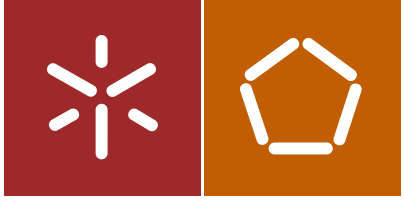
Universidade do Minho
Escola de Engenharia

Carlos Gustavo Ferreira Araújo Portela

Modelação de consumo de energia em Linux

UMinho | 2016 Carlos Gustavo Ferreira Araújo Portela Modelação de consumo de energia em Linux

julho de 2016



Universidade do Minho
Escola de Engenharia

Carlos Gustavo Ferreira Araújo Portela

Modelação de consumo de energia em Linux

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao
Grau de Mestre em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação do
Professor Doutor Victor Francisco Mendes de Freitas Gomes da Fonte

julho de 2016

Declaração

Nome: Carlos Gustavo Ferreira Araújo Portela
Endereço eletrónico: carlosportela@live.com.pt
Telefone: 914910221
Cartão de Cidadão: 13928012

Título da Dissertação: Modelação de consumo de energia em Linux
Orientador: Professor Doutor Victor Francisco Mendes de Freitas Gomes da Fonte
Ano de Conclusão: 2016
Designação do Mestrado: Mestrado Integrado em Engenharia de Telecomunicações e Informática

Universidade do Minho, ___ / ___ / ___

Assinatura: _____

Agradecimentos

Depois de terminada mais uma etapa crucial na minha vida académica e pessoal, tenho que deixar um agradecimento a todas as pessoas que possibilitaram e que contribuíram para a concretização deste projeto. Assim, gostaria de agradecer, de uma forma totalmente sincera, a todas as pessoas que, das formas mais variadas, me apoiaram em todo este processo.

Ao meu orientador, o Professor Doutor Victor Fonte, pelo tempo e pela paciência disponibilizada. Agradecer todas as suas sugestões que se revelaram bastante importantes para este trabalho, especialmente pela sua boa disposição natural.

Aos técnicos das oficinas de eletrónica, em especial ao Sr. Joel pela contagiante boa disposição e apoio técnico incondicional, na resolução dos problemas não só ao longo do projeto, mas também durante o percurso académico.

À Universidade do Minho e, em especial, aos membros dos Departamentos de Informática (DI), de Sistemas de Informação (DSI) e de Eletrónica Industrial (DEI), pela formação dada desde o inicio do Mestrado Integrado em Engenharia de Telecomunicações e Informática. Hoje assumo que todo este percurso foi produtivo e imprescindível para poder ser o profissional que sou hoje e o engenheiro que serei amanhã.

Aos meus amigos mais próximos que me acompanharam neste longo percurso universitário. Em especial àqueles com quem compartilhei e partilhei casa durante quatro inesquecíveis anos e a quem devo parte da minha formação como pessoa.

À minha família, em especial aos meus pais, Carlos e Rosa Portela, por me terem apoiado incondicionalmente desde a primeira decisão que tomei em ingressar neste curso. Pelo esforço para me darem todas as condições para que tudo corresse da melhor forma e para garantirem o meu bem-estar pessoal e profissional.

Por ultimo, mas não menos importante, à minha namorada, Sofia Brandão, pelo apoio incansável, pela motivação transmitida ao longo deste projeto e durante o percurso académico, pela companhia e pela amizade expressada nos melhores e piores momentos.

Resumo

Nestes últimos anos, a importância da eficiência energética nos sistemas informáticos tem vindo a crescer exponencialmente desde a área móvel até à computação de alto desempenho. O crescimento do mercado dos gadgets e a sua crescente dependência dos serviços de computação em Cloud são algumas das razões para este rápido avanço neste ramo tecnológico. A redução do consumo de energia e o aumento da produtividade de um sistema é, por várias razões, uma preocupação, tanto por parte do cliente como do fabricante. A exploração aprofundada do tópico pode contribuir para, por exemplo, o aumento da autonomia dos terminais móveis e a redução dos custos energéticos dos data centers e do cliente particular.

Neste sentido, a monitorização do consumo de energia de um sistema desempenha um papel fundamental para se aprimorar a eficiência energética do mesmo. No entanto, o grande desafio atual da monitorização passa por categorizar o consumo de energia a vários níveis como, por processo, por máquina virtual ou por subsistema de hardware. Esta escala de granularidade na análise energética permite o desenvolvimento de relatórios mais incisivos e conclusivos sobre a distribuição de consumo de energia do sistema.

No âmbito desta dissertação e tendo por base estes motivantes fatores, foi desenvolvido um modelo simples que visa estimar o consumo de energia do sistema na sua totalidade, categorizado por subsistema e, no caso do armazenamento secundário, também categorizado por processo.

Este documento apresenta um estudo sobre diferentes metodologias de medição assim como sobre as abordagens possíveis para o modelo em sistemas de teste com diferentes tipos de armazenamento secundário e com uma das ultimas gerações de processador da Intel. Numa fase inicial foi feita uma investigação a vários aspetos referentes à energia de um sistema, incluindo modelos de estimação do consumo de sistemas, métodos de medição de consumo e a precisão desses mesmos métodos.

No desenvolvimento do modelo, recorreu-se apenas a recursos existentes no sistema em causa para viabilizar um mais fácil investimento a larga escala. Por isso, o modelo recorre a interfaces de gestão e monitorização de energia como o RAPL e ACPI. Foram analisados os mais importantes aspetos energéticos de um sistema como a distribuição do consumo de energia estático e dinâmico pelos subsistemas e avaliou-se a eficiência e o desempenho dos mesmos nas mais diversas atividades. Desta forma, garantiu-se uma maior polivalência do modelo e, de um modo geral, uma maior precisão do mesmo.

O modelo foi validado com base em ferramentas disponibilizadas pelo sistema e através de medições físicas. Os resultados obtidos parecem satisfatórios, tendo sido registadas taxas de erro máximas de 5% no consumo total e de 10% no consumo do armazenamento secundário.

Este modelo desenvolvido pode ser adaptado a outro sistema, no entanto, necessita de executar uma ferramenta de calibragem que realiza todas as etapas que foram executadas na configuração usada para este projeto. Isto acontece essencialmente no subsistema de armazenamento secundário onde não se recorre a qualquer ferramenta existente para a estimação da sua energia. Desta forma, há uma etapa inicial que consiste na exercitação do subsistema de armazenamento secundário através de ferramentas de benchmarking e na recolha de dados estatísticos do custo das operações. Em seguida, é feito um estudo sobre esses mesmos dados e são atribuídos diferentes pesos energéticos para cada operação executada no subsistema. Depois, constrói-se o modelo e este é calibrado com recurso a interfaces de energia como o RAPL e ACPI. No fim, este modelo deve ser capaz de apresentar a fatura energética de cada processo que utiliza o subsistema de armazenamento secundário. Além disso, o modelo deve também estimar o consumo de energia total do sistema e a sua distribuição pelos principais subsistemas.

Abstract

In the past few years, the importance of energy efficiency in computer systems has been growing exponentially from the mobile area to the high-performance computing. The gadgets market growth and their increasing dependence on Cloud computing services are some of the reasons for this fast progress in this technological field. The power consumption reduction and the increasing productivity of a system is, for multiple reasons, a concern both for the customer and the manufacturer. For example, the in-depth research of this topic can contribute for increasing the mobile terminals autonomy and for reducing the energy costs on data centers and even on the costumer devices.

Altogether, the power consumption monitoring of a system plays an important role to enhance its energy efficiency. Some of the most challenging aspects of this research field are accurate power consumption measurement, estimation and categorization at different levels of the system, from hardware components to processes. This meticulous scale in the energy analysis allows the development of more incisive and conclusive reports about the system power consumption distribution.

Within this thesis and based on these challenging factors, it has been developed a simple model that aims to estimate the system power consumption in its entirety, categorized by subsystem and, in the case of secondary storage, also categorized by process.

This document presents an analysis of different measurement methods as well as about possible approaches for a model that run over experimental systems with different configurations of secondary storage and using a last generation Intel processor. Initially a research was made concerning many aspects about a system energy, including power consumption estimation models, consumption measurement methods and those methods accuracy.

For the model development, it has been used only the existing resources in the tested system to enable an easier investment on a large scale architecture. Therefore, the model implements energy management and monitoring interfaces like RAPL and ACPI. The most important system energy concerns were analysed, such as the power consumption breakdown of static and dynamic power subsystems. The efficiency and performance of some system activities were evaluated. This process ensured the development of a more versatile model, and globally, with a greater accuracy.

The model validation process was carried out based on tools provided by the system and by physical measurements. The obtained results seem adequate, having been recorded a maximum error rate of 5% concerning the total system consumption and 10% regarding the secondary storage consumption.

This power consumption model can actually be applied to any typical computer system through a calibration tool also developed in this project. This calibration step will estimate relevant power consumption parameters regarding secondary storage since we cannot rely on existing tools or APIs concerning this subsystem. This calibration will exercise the secondary under different usage patterns. The collected statistics will be processed and the cost of relevant operations regarding this secondary storage will be estimated. The next step involves the model development and its calibration using power interfaces as RAPL and ACPI. In the end, this model should be capable of presenting the energy bill of each process that uses the secondary storage subsystem. Additionally, the model should also be able to estimate the total system power consumption and its breakdown by the major subsystems.

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

ACPI – *Advanced Configuration and Power Interface*
AF – *Advanced Format*
API – *Application Programming Interface*
ARM – *Advanced RISC Machine*
C-*States* – *Processor Power States*
CEET – *Centre for Energy-Efficient Telecommunications*
CFQ – *Completely Fair Queuing*
CISC – *Complex Instruction Set Computing*
CPU – *Central Processing Unit*
D-States – *Device Power States*
DDR – *Double Data Rate*
DIMM – *Dual In-line Memory Modules*
DRAM – *Dynamic Random Access Memory*
G-*States* – *Global System States*
GPU – *Graphics Processing Unit*
GSMA – *GSM Association*
HDD – *Hard Drive Disk*
IO – *Input/Output*
IOPS – *Input/Output Operations Per Second*
JEDEC – *Joint Electron Device Engineering Council*
LLC Miss – *Last Level Cache Miss*
LPDDR – *Low Power Double Data Rate*
MSR – *Model Specific Registers*
MV – Maquinas virtuais
P-*States* – *Processor Performance States*
PDU – *Power Distribution Unit*
PMC – *Performance Monitor Counter*
PP – *Power Plane*
procfs – *proc filesystem*
RAM – *Random Access Memory*
RAPL – *Running Average Power Limit*
RISC – *Reduced Instruction Set Computing*
SATA – Serial AT Attachment
SDRAM – *Synchronous Dynamic Random Access Memory*

SMART – *Self-Monitoring, Analysis, and Reporting Technology*
SoC – *System-on-Chip*
SO- DIMM – *Small Outline Dual In-line Memory Modules*
SPM – *Spindle Motor*
SSD – *Solid State Disk*
T-*States* – *Throttling States*
TI – Tecnologias de Informação
TIC – Tecnologias de Informação e Comunicação
UPS – *Uninterruptible Power Supply*
USB – *Universal Serial Bus*
VCM – *Voice Coil Motor*

1. Introdução

1.1 Contextualização

Com a constante evolução das Tecnologias de Informação e Comunicação (TIC), a eficiência energética é um tema que tem vindo a preocupar cada vez mais tanto a industria como a investigação tecnológica. Vários investigadores concluíram que, atualmente, o consumo atual do ecossistema global TIC varia anualmente entre os 1,100 e 1,800 TWh. A fim de se entender a dimensão desta questão, estes valores ultrapassam o consumo anual total de energia gerada pelo Japão e Alemanha em conjunto. Estes cálculos envolvem o consumo energético dos *data centers*, infraestruturas de telecomunicações, terminais do consumidor final e, claro, todo o processo de fabrico destes instrumentos [1] [2]. No quotidiano, assistiu-se ao fim dos dispositivos móveis de utilização básica que dependiam de uma bateria ou de um conjunto de pilhas, como o *walkman*, o rádio ou o telemóvel de primeira e segunda geração, abrindo-se a porta a dispositivos mais sofisticados. Os novos *gadgets*, como aqueles que o mercado intitula comercialmente de *smartphones*, *ultrabooks* ou *tablets*, são um dos motores que estão a acelerar esta evolução, tanto para fins particulares como empresariais. As empresas, no âmbito de melhorar a sua capacidade governativa e de gestão de recursos, também acompanham a evolução procurando serviços mais informatizados, práticos e acessíveis de qualquer parte do mundo.

Para se perceber o crescimento e escala do investimento no ecossistema global de TIC, devemos considerar o facto de que, atualmente, esta área representa 10% do consumo mundial de energia [1]. O rápido aumento do trafego digital é a força condutora para o grande aumento de investimento global nas infraestruturas TIC. O Centro para Telecomunicações de Eficiência Energética (CEET) reportou que grande parte deste aumento de consumos é originado por *hardware* de comunicação de redes sem fios, como o Wi-Fi e o 3G, que, por sua vez, são utilizados para aceder a serviços Cloud. De acordo com os cálculos dos autores, os acessos sem fios à nuvem, em 2015, atingem consumos de 43 TWh, que, comparados com 9,2 TWh em 2012, representam um aumento de 460% [2].

A GSM Association (GSMA) concluiu que no final de 2014, metade da população mundial tinha pelo menos uma subscrição móvel. Desse conjunto, 40% eram detentores de um serviço móvel de terceira e quarta geração, valor esse que cresceu significativamente em comparação com os 10% referentes a 2008 [3]. Numa previsão para 2020, a Ericsson considera que mais de 80% dos subscritores de serviços móveis

serão utilizadores de serviços de terceira e/ou quarta geração ou superiores [4]. Verifica-se assim, nestes últimos anos, uma fase de transição para dispositivos móveis mais avançados com outras capacidades e serviços, que tornam as redes mais movimentadas e com maiores quantidades de dados. Por exemplo, a Cisco estima que os *smartphones* estejam a consumir até 37 vezes mais trafego de dados que os telemóveis tradicionais. No caso dos portáteis e *tablets*, a Cisco estima um consumo de 119 e 94 vezes superior respetivamente [5]. A GSMA prevê ainda que em 2016 o trafego de dados móveis aumente mensalmente 4,5 vezes comparativamente ao ano de 2013 [3].

Com esta evolução notável nas tecnologias móveis, nasce também um utilizador mais interativo com a possibilidade de, através do seu *gadget*, acompanhar a atualidade, transmitir e visualizar vídeos ou partilhar ficheiros. Atualmente, as transmissões de vídeo e as redes sociais representam 40% e 10% do trafego de dados das redes móveis respetivamente [3] [4]. A YouTube constatou mesmo que, em Outubro de 2014, 50% do seu trafego global era gerado por dispositivos móveis [6]. Este aumento de interatividade e uso de *smartphones* tem consequências interessantes a nível de consumos. A título de exemplo, um *tablet* ou *smartphone* que reproduza, em média, uma hora de vídeo por semana, consome anualmente mais energia que dois frigoríficos. Neste processo inclui-se as operações da rede e do terminal. [1] Considerando que um *smartphone* representa apenas um dispositivo num oceano do ecossistema de TIC, é de notar que as preocupações sobre os consumos energéticos vão muito para além dos dispositivos móveis.

Do lado do utilizador, as funcionalidades destes dispositivos são bastantes simples, mas, para isso, há uma enorme infraestrutura em nuvem que nos bastidores cria condições para que estes serviços sejam possíveis e capazes de dar resposta aos níveis de utilização.

Os serviços em nuvem estão neste momento a representar um papel essencial nas funcionalidades dos *gadgets*. Qualquer utilizador que adquira um *gadget* e pretenda usufruir das principais caraterísticas de sincronização do seu dispositivo, necessita obrigatoriamente de estar ligado a serviços nuvem. Estes factos comprovam que o crescimento tecnológico é muito mais do que aquilo que um utilizador de dispositivos consegue facilmente ver. O Facebook, a Apple, a Amazon, o Google e a Microsoft são algumas das marcas que estão a transformar os métodos de trabalho, de comunicação, de ver filmes ou televisão, ouvir musica e partilhar imagens pela Cloud. Estas empresas de computação em nuvem têm vindo a evoluir os seus *data centers*, isto é, a alargar as suas bases de armazenamento digital, para alimentar o desejo do utilizador de instantaneamente aceder a uma infinidade de conteúdos através dos seus computadores, telemóveis e de outros dispositivos. Em 2010, os *data centers*

contribuíram para aproximadamente 1,5% da eletricidade utilizada globalmente [7] [8] [9]. Num relatório de 2012, a Greenpeace Internacional previu que entre 2012 e 2020, os *data centers* aumentem a sua capacidade até 50 vezes [10] [1]. A computação em nuvem é uma das principais preocupações nos dias que correm em termos de consumos energéticos, sendo que, os seus principais custos recaem sobre as suas infraestruturas. Este mesmo relatório considera que os *data centers* são a maior razão para o aumento da necessidade de energia global. Os investigadores estimam que alguns destes possam estar a atingir consumos energéticos equivalentes ao consumo de 180 mil casas [10]. Grande parte destas centrais de dados localizam-se em cidades remotas e suburbanas e algumas são mesmo visíveis do espaço.

A evolução tecnológica tem sido espantosa, no entanto, há uma preocupação que está relacionada, de várias maneiras, com todas as inovações que surgem. Essa preocupação assenta sobre aquilo que alimenta todos os equipamentos eletrónicos: a energia. Não sendo uma opção pôr fim a este processo evolutivo, é importante procurar um caminho que seja mais controlado a nível energético. Por vezes, o caminho mais simples e linear não é aquele que deve ser percorrido com o objetivo de tornar os sistemas mais eficientes e, com isto, o planeta mais verde.

1.2 Motivação

Tendo em conta os aspetos referidos anteriormente, pode verificar-se que os *data centers* são o coração da Internet de hoje e dos sistemas de computação em nuvem. Apesar disso, este ramo está a ser cada vez mais prejudicado pelos seus consumos energéticos e custos de operação. Tornar estas plataformas mais eficientes é a solução tanto para encurtar as necessidades energéticas, como para reduzir significativamente os custos às organizações, para além de que é um contributo para o conceito de *green power*.

No processo evolutivo destas gigantes áreas de armazenamento digital, existem preocupações que motivam a exploração do tema. O aumento das capacidades de computação tem sido um dos fatores do crescimento do consumo de energia. São cada vez mais as empresas que procuram colocar e processar todos os seus dados remotamente sem a necessidade dos seus próprios recursos. Isto obriga a que as empresas de Tecnologias de Informação (TI) passem por constantes aumentos nos seus sistemas de computação, desde os *server racks* até ao *uninterruptible power supply* (UPS). Este investimento nos recursos dos *data centers,* para responder às necessidades dos clientes provoca um aumento do consumo de energia. De acordo com a IDC e a Intel, por cada dólar gasto em hardware de um servidor, um valor adicional de 51 cêntimos é despendido em energia e refrigeração [11] [12]. A IDC, neste

mesmo estudo prevê ainda que este rácio suba para 69 cêntimos num futuro próximo [12]. Outra grande preocupação das empresas de TI é o aumento progressivo do custo da energia nos últimos tempos [11]. A Emerson constatou que, entre 2002 e 2006, se verificou um aumento de 56% no preço da eletricidade a nível mundial [13]. Logicamente, este aumento tem mais impacto nas empresas dominantes deste mercado, no entanto, não deixa de ser uma preocupação para qualquer empresa detentora de *data centers*.

Para além das questões referidas anteriormente, existem outras limitações que podem surgir no crescimento de um *data center*. Um dos possíveis constrangimentos é, na zona onde estão instaladas estas bases, não haver disponibilidade, ou ser demasiado dispendioso, o investimento para aumentar a potência necessária para o crescimento da plataforma. Outro obstáculo é por vezes provocado por agencias do governo ou organizações ambientais que procuram, por exemplo, tornar a área mais verde, limitando as emissões de carbono. Têm surgido até ao momento bastantes artigos sobre o impacto ambiental que podem representar estas plataformas e sobre possíveis soluções para este problema [10] [14].

Com estas preocupações, surge uma maior necessidade de monitorizar a quantidade de energia consumida pelos sistemas informáticos. A modelação e previsão do consumo de energia nos *data centers* tem um papel fundamental neste contexto, que tem sido amplamente estudado pelas empresas dominantes deste mercado. A eficiência energética implica um maior rendimento dos recursos dessas empresas sendo, por isso, uma meta aliciante para as mesmas. São vários os artigos que verificam melhorias nesta área e mostram que estas empresas de TI têm apresentado resultados positivos, tanto a nível ambiental como económico [10] [11] [15] [16]. Estes resultados explicam-se devido às suas rigorosas capacidades de implantar medidas de eficiência energética nas suas bases [17]. O problema incide de uma forma mais preocupante em pequenos e médios *data centers,* nos quais as condições e margens de investimento não são equiparáveis às detidas pelas plataformas de um líder de mercado. A Anthesis indica que, perto de metade do consumo de eletricidade dos *data centers* nos Estados Unidos da América, é dominado por pequenos e médios *data centers*. Explica ainda que estes resultados se devem não só ao facto destas bases de dados serem tipicamente ineficientes, mas também por estas existirem na maior parte das vezes em edifícios de escritórios de pequenas e grandes organizações do país [17].

A monitorização aprofundada traz vantagens tanto para o cliente como para a empresa de TI, que obtém um controlo mais aprofundado sobre as lacunas nos consumos dos seus equipamentos. Estas análises podem ocorrer tanto a nível dos consumos por componentes das máquinas como por entidades que acedem aos *data*

centers, ou seja, a empresa de TI consegue obter relatórios mais personalizados onde pode identificar os componentes que estão a dissipar mais energia e os clientes que estão a usufruir dos seus recursos em maior escala. Desta forma, surge a oportunidade da empresa apresentar diferentes preços pelos seus serviços, consoante os consumos de energia e níveis de utilização de cada um dos seus clientes. Este tipo de monitorização tem a vantagem de a empresa poder cobrar pela prestação dos seus serviços o preço que é mais apropriado para o tipo de utilizador que é o cliente. O cliente acaba por ser igualmente beneficiado visto que não irá pagar um valor superior ao indicado pelo nível utilização que deu ao serviço. Atualmente, existem alguns planos de serviços de nuvem com preços tabelados consoante o nível de utilização [18]. No entanto, existe a este nível uma preocupação sobre os cálculos demasiado lineares apresentados pelas empresas, que se suspeita não apresentarem valores fiáveis. Um bom exemplo deste facto confirma-se pela falta de distinção entre os acessos aleatórios e sequenciais aos discos rígidos.

A energia consumida por um *data center* pode estar enquadrada em duas categorias: energia consumida pelo equipamento de TI (exemplo: servidores, armazenamento, rede) e energia consumida pelas infraestruturas complementares instaladas (exemplo: sistemas de refrigeração e fontes de alimentação de energia). A quantidade de energia utilizada por estes dois subcomponentes depende da arquitetura estabelecida, assim como da eficiência dos equipamentos. A Intel constata que grande parte dos Engenheiros que projetam *data centers* têm de confiar em informação inadequada sobre a utilização de energia dos servidores e das suas *racks*. Assim, estes profissionais necessitam de ser mais conservadores no desenvolvimento do projeto e estruturar as infraestruturas e servidores com base no rácio de energia declarado pela informação do equipamento [19]. Isto, geralmente, leva a uma utilização desproporcional das máquinas em relação à refrigeração estabelecida para as mesmas. Atualmente, grande parte dos artigos que analisam os consumos energéticos dos *data centers* averiguam que entre 40% a 50% da energia total utilizada por estas plataformas é consumida pelos sistemas de refrigeração instalados nas mesmas [1] [7] [13] [20]. Neste sentido, verifica-se a importância de estabelecer níveis de refrigeração proporcionais à utilização dos servidores, permitindo por um lado uma maior poupança energética e, por outro, a mesma performance. A instrumentação de consumo de energia a todos os níveis de um *data center*, desde o processador até à plataforma total, é a solução para se atingir uma maior eficiência energética.

Para além dos *data centers*, explorar e monitorizar os consumos energéticos traz também benefícios nos terminais móveis dos utilizadores. Como referido anteriormente na contextualização, os dispositivos móveis têm vindo a consumir

cada vez mais energia, particularmente devido à evolução das transmissões de dados móveis e à utilização que os consumidores dão aos mesmos. Sendo estes dispositivos desenhados para ter uma autonomia considerável, é do interesse das respetivas empresas o estudo intensivo de eficiência energética aquando do desenvolvimento dos mesmos. Num inquérito realizado pela Deloitte em 2014, o fator "vida da bateria" foi considerado como segundo fator mais importante na escolha de um telemóvel, sendo apenas ultrapassado pelo fator "ser um *smartphone*" [21]. Até aos dias de hoje, os *smartphones* têm em geral melhorado o seu tempo de autonomia. A Deloitte estimou que entre 2014 e 2015 fosse verificado um aumento de 15% da autonomia dos *smartphones* devido aos melhoramentos de processador, transmissores de rádio, ecrã e software [21]. Por exemplo, a Apple, na sua linha de computadores portáteis, apresenta resultados bastante positivos de autonomia e eficiência energética. Desde o primeiro MacBook Air, lançado em 2008, até ao modelo lançado em 2013, a Apple indica que foi reduzido o seu consumo em 45 pontos percentuais [22].

Estes factos comprovam que o consumo de energia se está a tornar uma preocupação crescente na área tecnológica, particularmente, nos *data centers*. Aliás, atualmente, verifica-se uma notável determinação para encontrar soluções capazes de moderar a crescente evolução deste problema. Apesar disso, grande parte destas soluções desvia as atenções de um subsistema que tem um grande impacto neste ramo, o armazenamento secundário [23]. A otimização do consumo de energia deste subsistema tem um papel importante neste desafio, visto que, é um dos três que mais energia consome nos *data centers* [13] [24] [25]. O aumento constante da capacidade de armazenamento destes centros de dados provoca um aumento na proporção de energia total consumida pelos mesmos [23]. Ao contrario do armazenamento secundário, o processador e memória primária, têm-se tornado bastante transparentes e com notáveis reduções no consumo de energia. No que toca ao armazenamento secundário, o disco rígido é o componente mais utilizado [24] [26] [27]. O subsistema faz-se acompanhar por uma estrutura mecânica que provoca um aumento nas variações de potência consumida consoante o tipo de operações. Além disso, há a grande desvantagem de este apresentar pouca ou nenhuma clareza sobre as suas caraterísticas de desenvolvimento e as operações que realiza [24] [28]. A maioria dos estudos de consumo energético e funcionamento apresentados é da origem dos próprios fabricantes, que obviamente apresentam um bom parecer acerca dos seus produtos [29].

Esta situação motiva uma exploração mais profunda sobre o verdadeiro impacto desta caixa selada aos olhos dos clientes. Desenvolver um modelo capaz de esboçar as operações realizadas pelo subsistema de armazenamento secundário por cada acesso ao Sistema Operativo, é um desafio aliciante que pode trazer grandes vantagens para

aqueles que procuram uma melhor monitorização sobre os seus sistemas, e, por sua vez, uma maior eficiência energética dos mesmos.

Neste sentido, a finalidade deste projeto, numa fase em que os custos energéticos se têm tornado cada vez mais importantes na computação, principalmente porque têm um impacto direto no custo do provisionamento energético para as infraestruturas de computação, nas despesas de operabilidade, não só para os *data centers* e para os edifícios empresariais, como também para a vida das baterias dos portáteis e de dispositivos móveis.

A modelação para estimativa energética vai ter um papel cada vez mais importante nos ecossistemas TIC. É essencial encontrar soluções para encurtar os consumos e os custos de forma significativa. Acredita-se que um modelo de consumo de energia seja a solução para que se possa identificar os custos de energia momentâneos, identificar potenciais reduções de custos, implementar projetos de eficiência energética e validá-los com o apoio das medições continuas de consumos. É importante fazer medições antes de identificar e retificar o problema. Na verdade, simplesmente não se pode melhorar alguma coisa, especialmente a eficiência energética, se não a estivermos a medir. Sem uma noção base e medições continuas de consumos, é impossível determinar onde é possível otimizar, avaliar resultados das otimizações ou mostrar as melhorias aos quadros superiores das empresas, agencias governamentais ou mesmo clientes.

1.3 Objetivos

Tendo por base os aspetos referidos anteriormente, o intuito primordial deste projeto passa por desenvolver um modelo de consumo de energia parametrizado em Linux. Esse modelo permitirá monitorizar os consumos de energia de um computador, tanto a nível geral (consumo total) como a nível detalhado, por componente ou processo. O modelo desenvolvido deve utilizar apenas funcionalidades internas da máquina em questão, tanto a nível de hardware como de software. As estimativas serão baseadas em valores apresentados por sensores incorporados nos componentes e estatísticas fornecidas pelo sistema operativo Linux instalado na máquina. Esta decisão deve-se ao investimento impraticável de utilizar hardware externo (ex: PDU's externos) em situações de larga escala. Seria também uma tarefa complexa e dispendiosa, desenhar uma rede de equipamentos de medição externa para servidores de diferentes tipos. Para além disto, os medidores de energia externos e independentes excluem a possibilidade de medir os consumos por componente ou processo e, com isso, limitam a capacidade de diagnosticar a causa dos consumos energéticos imprevistos. A opção de medição utilizando apenas

recursos internos da máquina apresenta a grande vantagem da gestão facilitada e de um investimento mais reduzido, tanto para este projeto como para uma visão empresarial futura.

O modelo deve ser capaz de se adaptar à realidade atual da tecnologia e, por isso, este deve fazer-se acompanhar das funcionalidades recentemente apresentadas pelos fabricantes. Para se conhecer melhor a atualidade tecnológica no ramo da otimização e monitorização energética, é primeiramente necessário ser realizado um estudo sobre as soluções existentes. Desta forma, numa fase inicial, são estabelecidos os seguintes objetivos:

- Estudo do impacto energético dos principais subsistemas de hardware num sistema;
- Estudo de técnicas de controlo e gestão de energia e desempenho;
- Estudo de fontes e métodos de monitorização de desempenho e energia.

Com o estudo realizado, verifica-se que, no ramo da gestão de energia, existem componentes que têm se tornado bastante transparentes e de fácil monitorização, ao passo que outros, têm continuado verdadeiras incógnitas na área do consumo de energia. No modelo a desenvolver, pretende-se incorporar a estimativa energética dos componentes mais relevantes a nível dinâmico na atividade de um sistema computacional. São vários os estudos que destacam que, durante os últimos 10 anos, os principais subsistemas geradores dos consumos de energia dinâmica são o *Central Processing Unit* (CPU), a Memória e o Disco [13] [24] [25]. Desta forma, para este projeto, pretende-se implementar um modelo de estimativa de energia baseado em três componentes essenciais:

- Processador;
- Memória primária;
- Armazenamento secundário.

Dos três componentes mais relevantes no impacto energético de um sistema, dois deles, nomeadamente o processador e a memória primária, apresentam atualmente uma notável evolução na gestão e monitorização de energia. Alguns fabricantes como a Intel e AMD dispõe de ferramentas que tornam possível a obtenção de detalhes sobre a energia consumida desses dois componentes através de sensores implementados nas novas arquiteturas, acompanhados de modelos de energia mais sofisticados e precisos [30] [31]. Por oposição, no armazenamento secundário, a situação é um pouco diferente e verifica-se alguma intransparência nas operações do componente. Este facto verifica-se principalmente no disco rígido ou *hard drive disk* (HDD) que, por sua vez, é atualmente o principal componente de memória secundária utilizado em servidores de *data centers* [24] [26]. Atualmente, são vários os artigos

que referenciam os discos como o subsistema mais difícil de modelar corretamente [24] [28]. Isto deve-se à sua dificuldade de consulta sobre os estados de energia do controlador de disco, o impacto do funcionamento da memória *cache* do componente, e, no caso dos discos HDD, as necessidades mecânicas do disco consoante as operações aleatórias e sequenciais de leitura e escrita [28] [32]. No caso dos *solid state disk* (SSD), a situação é facilitada pela linearidade do seu funcionamento e pelo similar impacto energético entre operações de escrita aleatória e sequencial. Desta forma, para o desenvolvimento deste projeto, acrescentam-se dois novos objetivos nesta fase inicial:

- Estudo de técnicas existentes de interceção do acesso ao armazenamento secundário;
- Desenvolvimento de técnicas de interceção do acesso ao armazenamento secundário.

O modelo é adaptável ao tipo de armazenamento secundário em questão. Desta maneira, deverão ser apresentadas duas soluções consoante o tipo de disco em questão, visto que, os HDD e SSD apresentam modos de funcionamento completamente distintos. Os subsistemas do processador e da memória primária serão estimados com base na ferramenta RAPL, disponibilizada nos mais recentes modelos de processador Intel. No caso do armazenamento secundário, este passa por uma fase de ensaios para se perceber os custos energéticos em cada tipo de acesso e de operações. Com isto, é realizada uma etapa de ensaios que se pode objetivar com os seguintes pontos:

- Estudo das condicionantes mais relevantes para o subsistema;
- Estudo de soluções para a realização dos ensaios;
- Execução dos vários ensaios de desempenho na máquina;
- Análise dos indicadores recolhidos nos ensaios;
- Estudar um modelo utilizando os indicadores adequados.

Com os custos energéticos das diversas atividades do sistema estudados ao pormenor, será então desenvolvido um modelo de estimativa de energia que engloba os três principais subsistemas. Desta forma, pretende-se que o modelo parametrizado desenvolvido neste projeto apresente:

- O consumo total do componente CPU;
- O consumo total do componente de memória primária, a *Random Access Memory* (RAM);
- O consumo total do componente de armazenamento secundário (HDD, SSD);

- O consumo energético de cada processo sobre o componente de armazenamento secundário (HDD, SSD);
- O consumo total do sistema em questão com base no cálculo destes três mais relevantes componentes.

Numa fase final, em que o modelo está totalmente definido, deverá ser realizada uma etapa de validação que, coloca à prova o potencial do modelo para estimar o consumo de energia parametrizado do sistema. Nesta fase, o modelo é executado em conjunto com ferramentas de *benchmarking* independentes que exercitem o sistema e, em especial, o subsistema de armazenamento secundário. Os resultados recolhidos do modelo neste processo são comparados com aqueles que são obtidos através de soluções a que o modelo não recorre. Entre essas soluções, surge a utilização de medições físicas externas ao sistema e a utilização do indicador de consumo de energia disponibilizado pela tecnologia ACPI Smart Battery. As análises de validação realizadas devem relatar os níveis de precisão e fiabilidade do modelo.

1.4 Estrutura

A presente dissertação está dividida em seis capítulos, os quais abordam cada uma das etapas constituintes deste projeto. Neste primeiro capitulo é apresentada uma breve introdução sobre a temática assim como a motivação, objetivos do projeto e a estrutura do documento da dissertação. Além deste capitulo introdutório, seguem-se os seguintes capítulos e apêndices:

Capitulo 2: São apresentados os conceitos e tecnologias envolvidas na realização deste projeto. Neste capitulo são abordados temas como a evolução dos sistemas e dos seus subsistemas (*hardware*) ao nível de desempenho e consumo energético. São exploradas tecnologias de monitorização e controlo de energia como o *Running Average Power Limit* (RAPL) e a *Advanced Configuration and Power Interface* (ACPI). Além disso, são estudados modelos que estimem o consumo de energia nos subsistemas que são mais relevantes para o modelo do projeto.

Capitulo 3: São expostas as metodologias de investigação e desenvolvimento utilizadas no desenrolar deste projeto. Este capitulo destaca-se por apresentar os métodos para o desenvolvimento do modelo e em que condições é que ele é desenvolvido. Por exemplo, nele é possível encontrar-se as técnicas de recolha de dados adotadas e a configuração do sistema em causa ao nível de hardware e software.

Capitulo 4: É explicado o processo de conceção e implementação do modelo de consumo de energia. Neste capitulo são explicados os tipos de ensaio executados e os resultados obtidos nos mesmos. Depois, com o estudo cuidado desses ensaios e

dos seus indicadores recolhidos, é implementado um modelo de consumo de energia parametrizado com duas vertentes, SSD e HDD.

Capitulo 5: Nesta fase é realizado um estudo de validação sobre o modelo desenvolvido. Numa primeira parte, são explicadas detalhadamente as metodologias utilizadas para se realizar esta atividade. Em seguida, com o processo de validação realizado, são então estudados os resultados obtidos e são calculados os níveis de precisão e estabilidade do modelo, tanto na vertente de SSD como de HDD. Por fim, é apresentada uma apreciação global sobre o potencial deste modelo de estimação de energia consumida.

Capitulo 6: No ultimo capitulo que compõe esta dissertação sumariza-se as principais conclusões obtidas com base no projeto elaborado. Além disso, inclui-se também uma subsecção que constata várias propostas de trabalho para dar continuidade ao projeto desenvolvido.

2. Estado da Arte

Neste capitulo será apresentada a revisão literária e o estado da arte das tecnologias envolvidas na realização deste projeto. Desta forma, serão apresentados vários conceitos, como ACPI ou RAPL, que colaboram para a melhor gestão e monitorização do consumo de energia. Numa primeira etapa, com base em suporte cientifico, será estudado o impacto energético de vários componentes que integram um sistema informático. Depois, serão estudadas as várias soluções para monitorizar o consumo de energia de um sistema, a nível de interfaces e ferramentas. Por fim, serão explorados alguns trabalhos científicos que envolvem o desenvolvimento de modelos de estimativa de consumo de energia.

2.1 A Eficiência Energética nos Sistemas, Subsistemas e Processos

Neste subcapítulo, serão enumerados os vários componentes que têm uma especial influência no impacto energético de um sistema informático. É explicado de que forma e por que razão é importante desagrupar o consumo de energia de um sistema, tanto a nível de *software* como de *hardware*. Adicionalmente, serão referidas algumas áreas exploráveis para que se obtenha uma maior eficiência energética num sistema. Entre elas, encontram-se a redução do consumo de energia estático de um sistema e o maior aproveitamento de recursos de *hardware* por parte do *software*. Para que haja um maior aproveitamento no estudo destas soluções, destaca-se neste capitulo a relevância do desenvolvimento de um modelo de consumo de energia parametrizado.

2.1.1 Energia e Potência

A Energia (E) é a quantidade total de trabalho desempenhado por um sistema sobre um período de tempo (T), ao passo que, a Potência (P) é a taxa a que esse trabalho é desempenhado [33]. Por outras palavras, a Potência entende-se como a quantidade de Energia consumida por segundo, cuja unidade se representa em Joules por Segundo (J/s) ou, simplesmente, Watts (W). A relação entre estas três variáveis pode ser expressada da seguinte forma,

$$E = PT \tag{1}$$

onde E corresponde à energia consumida pelo sistema em Joules, P é medido em Watts e T representa o período de tempo medido em segundos. A expressão acima

pode ser transformada, com a Potência a corresponder à Energia consumida num período de tempo de T_1 a T_2,

$$P = \frac{E}{T_2 - T_1} \tag{2}$$

onde T_1 é o tempo inicial e T_2 o tempo final, e a sua diferença é apresentada em segundos.

Para além desta expressão, é possível obter a Potência (P) quando são conhecidos os valores de corrente e voltagem. A expressão pode representar-se por,

$$P = VI \tag{3}$$

onde V corresponde ao valor da voltagem em Volts e I corresponde ao valor da corrente em Amperes. Neste projeto, estes termos e expressões serão utilizados frequentemente, sendo imprescindíveis para o desenvolvimento do mesmo, quando o principal tópico é a medição do consumo de energia.

2.1.2 As Categorias de Consumos de Energia

Desagrupar o consumo de energia de um sistema é o conceito fundamental para o desenvolvimento de um modelo parametrizado deste consumo. Existem várias formas de desagrupar o consumo total de energia de um sistema, sendo que, uma delas é através do cálculo do nível de utilização dos recursos de *hardware* do sistema. Para isso, neste estudo, distinguir-se-á o consumo de energia de um sistema informático em duas diferentes categorias: o consumo de energia estático e o consumo de energia dinâmico.

O **consumo de energia estático** refere-se ao consumo base necessário para garantir a "prontidão" do sistema, para que este se encontre disponível e capaz de responder a qualquer necessidade do utilizador de forma rápida e eficaz. Esta é a energia consumida pelo sistema independentemente do seu estado de operação. Este estado inclui a energia consumida em *idle* pelos discos, interfaces de rede, *central processing unit* (CPU), *caches*, memória, *motherboard*, *chips* complementares ao sistema e equipamentos de refrigeração que acompanham estes componentes. Desta categoria faz também parte a energia necessária para manter operacionais os processos básicos do sistema operativo e outras tarefas em execução como, por exemplo, a energia necessária para tratar interrupções do CPU e manter os relógios de *hardware*, portas de rede e discos ativos [34].

O **consumo de energia dinâmico** deve ser entendido como o consumo provocado por atividade útil e requisitada pelo utilizador do sistema. Nesta categoria inclui-se a energia consumida pelo CPU aquando da execução das instruções, pela atividade extraordinária da memória primária, pelo disco enquanto procura, lê ou

escreve dados, pela interface de rede a receber e enviar pacotes de dados e outras necessidades para responder aos pedidos do utilizador. Os valores atingidos pelo consumo de energia dinâmico dependem principalmente do tipo de carga de trabalho que é executada pelo computador e como essa carga utiliza o CPU, memória, *input/output* (IO), etc.

2.1.3 A Eficiência Energética

Eficiência, no dicionário, define-se como a capacidade de "obter a máxima produtividade possível com o mínimo esforço ou despesa despendidos" [35]. Em relação à eficiência energética de um sistema, a definição pode-se aplicar ao melhor uso dos recursos necessários à execução das tarefas, relativamente ao consumo de energia. Para isto, a monitorização dos consumos de energia estáticos e dinâmicos é essencial para melhorar os resultados a curto e longo prazo. Note-se que, um típico sistema informático que tenha um elevado consumo de energia estático tende a ser mais ineficiente a baixa utilização. Nos *data centers*, isto é especialmente desvantajoso porque os níveis de utilização destes sistemas são habitualmente baixos. Barroso reporta que, num grande *cluster* da Google, a utilização varia entre os 10% e 50% mas, refere também que a Google se encontra bastante desenvolvida no que concerne esta matéria [36]. Várias análises estimam que no vasto ramo desta industria, os valores variam entre 6% e 12% [37] [38] [39], e, no maior serviço de Cloud Publico (Amazon EC2), há ainda estudos que sugerem que a utilização está próxima de 7% [40]. No geral, um elevado consumo de energia estático contribui significativamente para a ineficiência energética de um *data center*. Barroso e Holzle introduziram este problema em "*The Case for Energy-Proportional Computing*" [36]. Este facto faz com que haja um maior esforço nesta industria para reduzir a energia estática e construir servidores *power-propotional*, isto é, servidores que consumam uma energia proporcional à utilização e carga do que estão a realizar. Apesar da redução no consumo de energia estático ter um papel importante para haver uma maior eficiência energética, por forma a construir um sistema *power-propotional*, é igualmente necessário que se conheça a energia dinâmica consumida pelo mesmo. É importante perceber a evolução do sistema consoante os níveis de trabalho do mesmo e, com isto, estudar até que nível este sistema é eficiente com uma elevada utilização. Desta forma, torna-se possível entender se o sistema se torna proporcional, relacionando a energia consumida desde a sua mais baixa carga de trabalho até ao seu nível máximo possível de utilização e, assim, confirmar a sua eficiência tanto a nível de consumo de energia estático como dinâmico. A monitorização do consumo dinâmico de energia, juntamente com a aplicação de técnicas de otimização do

sistema, é fundamental para minimizar o consumo de energia, garantir mais tempo de vida às baterias, ou controlar a dissipação térmica.

Num *data center* qualquer redução de consumo de energia, por mais pequena que seja, tem um grande impacto no restante equipamento das plataformas, uma vez que tem um impacto em todo o redor de um computador. A título de exemplo, a Emerson estima que, a redução de consumo de 1 Watt num componente de um servidor de um *data center* de 500 m^2, tem um efeito em cascata que resulta numa poupança adicional de 1,84 Watts em fontes de alimentação, sistemas de distribuição de energia, sistemas UPS, sistemas de refrigeração e transformadores de média voltagem [13].

Para se conseguir estas melhorias, é necessário um estudo aprofundado sobre os vários componentes de um sistema. Devido às variações consoante a sua carga de trabalho, temperatura, e dispositivos individuais, os modelos estáticos de consumo de energia (por exemplo, derivados dos *datasheets* dos dispositivos) não apresentam uma estimação realista da energia realmente consumida [41].

2.1.4 Consumo de Energia nos Diferentes Subsistemas de *Hardware*

Na monitorização de consumos de energia, a principal preocupação no processo de desenvolvimento de um modelo, incide na componente de consumo dinâmico de energia. A principal razão é bastante percetível já que, enquanto o consumo estático tem um valor base inalterável e de fácil captação, o consumo dinâmico varia constantemente e, por vezes, torna-se difícil de encontrar alguma coerência nessa variação. A irregularidade deste detalhe deve-se a inúmeros fatores, entre eles, destaca-se o escalonamento de processos que impossibilita a fácil previsão das tarefas a executar pelo sistema. Desta forma, é essencial fazer-se um estudo sobre os níveis de consumo de energia dinâmico de cada componente de hardware. Este estudo facilita o desenvolvimento de um modelo que incida sobre as principais causas das variâncias de consumos dinâmicos. O consumo de energia estático fará claramente parte da construção do modelo, no entanto, será um valor facilmente medido através de uma *power distribution unit* (PDU) ou através de um equipamento de medição externa.

Segundo vários estudos realizados durante os últimos 10 anos, verifica-se que os principais subsistemas geradores dos consumos de energia dinâmica são o CPU, a Memória e o Disco [13] [24] [25]. Kansal refere num artigo que os subsistemas, CPU, memoria e disco, representam respetivamente, 58%, 28% e 14% do impacto do consumo dinâmico [24]. A Google, num estudo sobre os seus *data centers*, considera

os mesmos três subsistemas de *hardware* como principais causas de consumo energético dos seus equipamentos TI [25]. Esta última indica ainda que o CPU detém 33% do consumo energético total desses equipamentos, seguindo-se a memória com 30% e o disco com 10%.

Com isto, é fácil perceber que as maiores preocupações durante o desenvolvimento deste modelo de estimação de energia devem incidir principalmente sobre estes três componentes, que são responsáveis por grande parte da variação de consumos energéticos de um ou vários sistemas informáticos. A monitorização de energia dinâmica, mais detalhada ao nível de componentes e processos, facilita também o estudo das causas das variações de energia. Com estas informações, torna-se possível modificar ou substituir determinados componentes e aplicações mais desvantajosas e tornar o sistema mais eficiente e *power-proportional*.

Apesar de se reconhecer a inviabilidade de trocar modelos de estimação de energia por medições físicas, é importante saber que existem habitualmente dois tipos de métodos para medir fisicamente a energia consumida nos vários componentes de um sistema. Um deles é utilizando medidores de energia externos, que intercetam a fonte de alimentação, sendo que o outro recorre à medição interna com a utilização de sensores de energia ou *motherboards* especiais. Um medidor de energia externo como o PDU tem a qualidade de ser flexível, podendo ser facilmente conectado ou desconectado de uma máquina, sem afetar, no entanto, as operações normais do sistema. Um dos PDUs mais reconhecidos e utilizados para este fim, é o equipamento Watts UP Series com a precisão de 0,1 watt [42]. O Watts UP PRO regista a informação de energia na memoria do PDU, que pode ser posteriormente transferida para o computador. Esta ferramenta dispõe de um acesso remoto em "tempo real" à informação de energia através de um cabo de rede. A informação de energia pode ser depois trabalhada informaticamente através da programação da leitura do PDU. Apesar da solução ser bastante fiável, é praticamente inviável aplicar um PDU externo em grandes escalas. Além disso, o custo e a complexidade de estruturar todo o equipamento para servidores é bastante excessivo. No caso da medição interna, apresenta a vantagem de vir, geralmente, implementada no *hardware* existente a fim de permitir uma fácil estruturação e gestão. Um exemplo entre muitos é o dos servidores Dell Power Series, que fazem-se acompanhar destes sensores que por sua vez permitem obter informação de vários componentes como o CPU, memoria, disco, rede ou até de ventoinhas de refrigeração [43]. Apesar de haver bastantes vantagens na simplicidade deste serviço, os seus custos são bastante elevados para instalações de grande escala e, além disso, verifica-se um decréscimo de desempenho no sistema com este serviço implementado. Desta forma, estimar a energia consumida por estes componentes é um desafio importante que tem vindo a mostrar resultados cada vez

mais positivos através dos estudos que têm sido realizados até ao momento [44]. Este tipo de monitorização permite que o cliente final evite o investimento noutras soluções de medição física.

2.1.5 Consumo de Energia por Processos

A distribuição do consumo de energia por processos é um dos maiores obstáculos e, ainda de incerta fiabilidade, na parametrização do consumo de energia de um sistema. Este desagrupamento é de interesse geral, principalmente a nível de TI empresarial. Através destas informações é possível monitorizar o consumo de energia a nível aplicacional e, assim, diagnosticar os processos com maior consumo energético no momento. Surge com isto, uma oportunidade de se conseguir, por exemplo, monitorizar a energia consumida por uma máquina virtual que esteja em execução sobre o sistema operativo. As empresas de TI são aquelas que detêm maior numero de sistemas instalados em enormes plataformas computacionais, o que por si só explica um grande interesse das mesmas na procura de, no que toca aos seus *data centers*, reduzir custos de operabilidade, uma maior eficiência energética e uma justa distribuição de preços atribuídos consoante o nível de utilização de cada cliente. Como referido anteriormente neste subcapítulo, os baixos níveis de utilização de cada sistema obrigam as empresas a procurar soluções eficientes para distribuir e conjugar melhor as várias máquinas virtuais e aplicações em execução. É indispensável implementar sistemas distribuídos, mais flexíveis, de maneira a reunir o maior numero de serviços em execução pelo menor número de máquinas possível, mantendo o mesmo desempenho. Soluções como estas surgem sempre com o suporte imprescindível de um modelo de energia e desempenho, capaz de relatar em tempo real os mais baixos e altos consumos e níveis de utilização de um sistema.

Atualmente, os principais fabricantes do ramo informático apresentam várias aplicações para gerir os consumos de energia tanto a nível particular, num computador individual, como a nível industrial, para plataformas de inúmeros sistemas instalados. Destacam-se neste tópico as seguintes aplicações: Activity Monitor (MacOS), PowerTop (Intel) e Joulemeter (Microsoft).

O **Activity Monitor** [45] é um dos exemplos de monitorização do consumo de energia para os utilizadores particulares que têm um computador portátil ou fixo. Esta aplicação está incluída nos sistemas operativos MacOS e é uma ferramenta bastante simples e de fácil perceção que, entre outras informações de desempenho, apresenta o impacto energético, tanto a nível geral como subcategorizado por aplicação. A informação apresentada não contém valores concretos de Energia e Potência, sendo que, é apenas apresentado ao utilizador um gráfico, sem unidades de medida, que

ilustra a variação de consumo de energia num intervalo de tempo, assim como o impacto médio e instantâneo de cada aplicação para a energia total consumida.

O **PowerTop** [46], uma aplicação desenvolvida pela equipa de *Open Source* da Intel, apresenta informação de energia e desempenho do sistema em questão. A Intel descreve a aplicação como uma "ferramenta para diagnosticar problemas relacionados com o consumo e gestão de energia". Este utensílio de diagnóstico, aperfeiçoado para sistemas com processadores Intel, reporta os valores de consumo de energia, desde as aplicações em execução até aos componentes ativos do sistema. A ferramenta realiza o seu processo de calibração para cada sistema através da informação da bateria ou fonte de alimentação disponibilizada pelo mesmo. Para cada processo, o PowerTop considera o nível de utilização do processador, disco, rede, *Graphics Processing Unit* (GPU) e outros dispositivos como por exemplo, uma *Universal Serial Bus* (USB) Pen Drive. É importante destacar que esta ferramenta distribui o consumo de energia do sistema por todos os processos que são acedidos pelo sistema durante a medição. Desde o seu aparecimento que tem sido intensivamente utilizado pela Intel, distribuidores Linux e varias entidades da comunidade Open Source [46].

O **Joulemeter** [24] é uma ferramenta destinada a apresentar as mesmas funcionalidades de medição de consumo de energia mas, distribuídas pelas várias máquinas virtuais (MV) em execução num servidor. Esta ferramenta de *software* é capaz de estimar a energia utilizada por uma MV, um computador, um servidor ou uma aplicação de software. A estas funcionalidades, junta-se a possibilidade de modelar também o impacto energético de vários componentes como o CPU, ecrã, memoria e armazenamento na energia total consumida. Os autores da ferramenta indicam que o programa necessita de ser utilizado em conjunto com um medidor externo no caso das plataformas *desktop*. No caso dos portáteis, a energia utilizada é determinada sem recurso a medidores de consumo externos visto que, estes fazem-se habitualmente acompanhar de medidores implementados internamente na bateria [47]. O Joulemeter é utilizado para atingir melhores resultados de eficiência energética em vários cenários como *data centers*, computação empresarial e maquinas de operação através de bateria.

A título de exemplo, estas três ferramentas são apenas um excerto de uma lista de soluções na área de monitorização de consumos energéticos a nível aplicacional. Nestes exemplos destaca-se a falta de documentação que explique, de forma transparente, o seu funcionamento e as técnicas utilizadas para a apresentação dos valores calculados. As empresas da área tendem a desenhar e limitar totalmente as aplicações desenvolvidas para os seus sistemas, tanto a nível de *software*, como é o caso da Apple e Microsoft, como a nível de *hardware*, como é o caso da Intel.

2.2 Componentes de *Hardware* Relevantes

Nesta secção pretende-se explorar, de uma forma mais aprofundada, os vários componentes de *hardware* que se considera relevantes num sistema dos dias de hoje. Serão estudados os três componentes que no capítulo anterior se consideraram de maior impacto no consumo energético de um sistema. Para além desse conjunto, será realizado um pequeno estudo sobre o impacto de alguns periféricos que são habitualmente utilizados em computadores portáteis e *desktop*. Os periféricos, ao fazerem parte do quotidiano de muitos utilizadores, podem ser também vistos como um componente extra de consumo energético e, é bastante útil, conhecer-se o consumo que estes provocam. Apesar do modelo a ser desenvolvido não incidir sobre estes periféricos, os seus valores estáticos podem ser futuramente adicionados a estes mesmos modelos. Todas estas análises são importantes para se conhecer melhor os componentes que farão parte da rotina de desenvolvimento de um modelo. No caso dos três componentes principais, é importante conhecer bem o seu funcionamento e entender os movimentos de mercado para se conseguir apresentar uma configuração de *hardware* atualizada aos dias de hoje.

Neste projeto, consideram-se os termos *Random Access Memory* (RAM) e memória, sinónimos de armazenamento principal ou primário. Entende-se por memória a área do computador onde os dados são armazenados para acesso rápido pelo processador do sistema. O termo armazenamento secundário associa-se ao armazenamento não volátil que, não está diretamente controlado pelo CPU ou não interage diretamente com uma aplicação, e requer operações IO. A memória primária é volátil enquanto o armazenamento secundário é persistente [48].

2.2.1 Processador

Reduzir a energia consumida no processador é um dos passos mais importantes para se obter uma maior eficiência energética, pois é um dos componentes que mais energia consome quando está em execução [6]. Para se perceber o melhor sentido para monitorizar este componente é essencial que, em primeiro lugar, se estude os diferentes tipos de modelos assim como as principais causas para um eventual consumo de energia desnecessário.

Um processador moderno pode conter biliões de transístores. Estes componentes realizam a computação fundamental no sistema, sendo que direta ou indiretamente controlam todos os outros componentes [49]. Uma das maiores causas para o grande consumo de energia de um CPU são os transístores nele incluídos. Quanto mais transístores, maior será a perda de potência e, consequentemente, maior

o aquecimento ao nível do processador. A perda de potência é causada pela resistência interna do transístor e pela passagem de corrente, situações que deixam o transístor quente. O aquecimento foi sempre o problema dos processadores, mas, principalmente no mercado móvel, é também a consequência de fabricar dispositivos mais pequenos [50]. Além do próprio CPU ser uma grande fonte de consumo de energia, existem também componentes adjacentes que têm bastante importância no desempenho do próprio CPU. Como é possível ver na Figura 1, existem dois tipos de memória que estão permanentemente em contacto com o CPU: a memória *cache* e memória primária [48]. A memória *cache* apresenta-se por ser mais pequena e mais rápida do que memória primária e contém porções de memória desse mesmo subsistema. Quando o processador pretende ler uma palavra da memoria, primeiro verifica se essa informação está disponível em *cache*. Se for o caso, a palavra é imediatamente entregue ao processador. Caso contrario, um bloco de memória primária que consiste num numero concreto de palavras é lido para o *cache* e, depois, entregue ao CPU. No caso de haver múltiplos níveis de *cache*, as velocidades de comunicação tendem a reduzir, quanto maior for o nível.

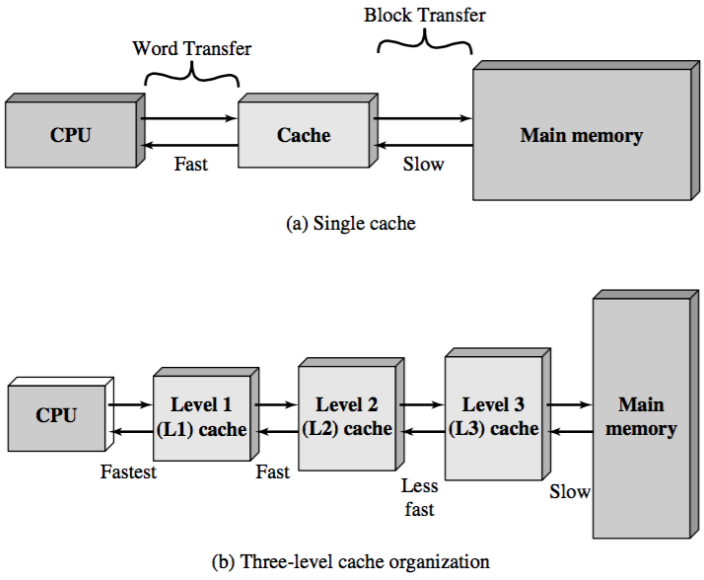


Figura 1- Relação entre CPU, Cache e Memória Primária [48].

Neste momento, um processador pode ser adaptado a inúmeras finalidades e, para isso, os seus fabricantes apresentam linhas diferentes que variam consoante as necessidades, por exemplo, de processamento ou de eficiência energética. Esta corrida parte da década de 80, com um debate acerca de duas diferentes filosofias de desenvolvimento de microcontroladores: a filosofia *Complex Instruction Set Computing* (CISC) e *Reduced Instruction Set Computing* (RISC) [51]. A filosofia CISC é baseada num conjunto de instruções de vários tamanhos em que, cada uma, pode

implicar a execução de várias operações complexas de baixo nível (*microcode*) e, por sua vez, originar vários ciclos [52]. A filosofia RISC, resume-se a um sistema que utiliza reduzidos e altamente otimizados conjuntos de instruções. Um *chip* CISC é relativamente lento na execução de cada instrução, pela complexidade e múltiplos ciclos que gerados pelo processo. No entanto, este tem a vantagem de, para realizar as mesmas tarefas, necessitar de executar menos instruções que um *chip* RISC. No caso da filosofia RISC, ao ser mais fácil de aplicar a técnica de *pipelining*, as operações podem ser mais facilmente executadas de forma paralela e, consequentemente, com mais rapidez. O conceito de RISC é também conhecido por resultar numa arquitetura mais escalável, devido à sua possibilidade de flexibilidade tecnológica, mantendo o mesmo design [52].

O destaque da arquitetura do tipo CISC é aplicar instruções complexas diretamente no *hardware* do PC, por isso as instruções por programa são mínimas, causando uma redução do tempo utilizado. Atualmente, estes dois conceitos têm as suas fronteiras um pouco apagadas e, os processadores mais modernos utilizam algumas caraterísticas de ambas as técnicas [53]. Apesar disso, o desenvolvimento dos processadores tende a ser baseado sempre num destes dois conceitos. Os processadores que são agora lideres do mercado, estão principalmente divididos em duas arquiteturas distintas: x86, baseado no conceito CISC, e Advanced RISC Machine (ARM), baseado no conceito RISC [51] [54]. Apesar de haver outras arquiteturas, a grande competição no mercado incide sobre esses dois tipos de processador. Os processadores de arquitetura x86 fazem parte do nosso quotidiano, maioritariamente em computadores *desktop* e servidores, ao passo que os processadores baseados em arquitetura ARM apresentam-se em grande parte nos dispositivos móveis, mais precisamente *tablets* e *smartphones*. Habitualmente, um processador baseado na filosofia de RISC requer bastante menos transístores do que um processador CISC [55]. Sendo a arquitetura ARM desenvolvida com o conceito de RISC, com processadores fisicamente mais simples que os x86, esta destaca-se por ser de menor custo e com menor temperatura e energia utilizadas [51]. Estas reduções são bastante apetecíveis no desenvolvimento de dispositivos portáteis alimentados com bateria, como *smartphones*, *tablets*, portáteis ou outros sistemas impregnados [54]. Os processadores para estes dispositivos móveis são tipicamente aplicados num *System-on-Chip* (SoC), que consiste num conjunto dos diferentes subsistemas do microprocessador agrupados juntamente com memorias e interfaces IO [56]. Por exemplo, num único SoC é possível encontrar múltiplos *cores* de CPU, GPU, *modems* de conexões sem fios (GPS, Wi-Fi, Bluetooth), controladores de memória, entre outros subcomponentes. Os SoCs são muito comuns no mercado dos dispositivos móveis devido ao seu baixo consumo de energia e formato compacto [54].

Atualmente, os SoC baseados na arquitetura ARM dominam o mercado de telemóveis e *tablets* e são produzidos por empresas como a Qualcomm Inc., MediaTek, Samsung Electronics Co. e TSMC. A Qualcomm Inc. domina o mercado móvel com uma quota de mercado atual de 70% [57] [58]. A Intel, fabricante líder do mercado dos processadores x86, detém neste momento apenas 1% de quota do mercado dos *smartphones* [59] [60]. Apesar disso, esta última detém uma quota de grande proporção do mercado em relação a processadores para computadores desktop, portáteis, e servidores com 88%, 90% e 99% respetivamente [59].

É interessante que, uma corrida que partiu da década de 80 esteja ainda ativa, de outros modos e com outras condições, mas, com a mesma ou até mais força que no passado. Os próximos anos continuarão a ser certamente interessantes para as duas principais arquiteturas do mercado.

2.2.2 Memória Primária

Como verificado no subcapítulo anterior, atualmente, o segundo maior consumidor de energia num servidor é a memoria. O rápido aumento da capacidade das memórias e da largura de banda contribuíram para este subsistema representar agora uma significativa porção do consumo energético total do sistema [61]. Os processadores são totalmente dependentes deste componente e, desta maneira, ele é imprescindível para o funcionamento de um sistema [48]. Os processadores armazenam na memória primária os dados que necessitam de executar. O acesso direto do processador à memória primária é chamado de RAM. O acesso à memória é *random* uma vez que o processador pode ler ou escrever, em qualquer momento que seja necessário, conjuntos de dados em múltiplos e diferentes intervalos de endereços de memória. Para a maioria dos computadores, a memoria primária é chamada de temporária já que o processador está constantemente a movimentar aplicações e dados que cada aplicação precisa, para dentro e para fora da memória primária. Esta qualidade temporária não é dependente de tecnologias especificas de memória. Para manter a velocidade do processador, as tecnologias de memória dos computadores têm utilizado memória volátil. Neste contexto, volátil significa que os dados desaparecem ao desligar o sistema [48]. Nos sistemas modernos, as células da *Dynamic Random Access Memory* (DRAM), têm que estar continuamente ativas ou os dados desaparecem.

Com o passar dos anos, o ramo da memória primária tem apresentado uma evolução nas suas interfaces normalizadas, para que os fabricantes, tanto de memória como de outros componentes relacionados, pudessem ambos disfrutar da economia de escala. Atualmente, a tecnologia *Double Data Rate* (DDR) é o tipo de memória

sincronizada dominante na memória primária dos sistemas informáticos [62]. Enquanto a versão primária de uma *Synchronous Dynamic Random Access Memory* (SDRAM), permite apenas um único envio de dados para o processador por ciclo de *clock* do *bus*, este novo conceito de DDR permite enviar dados duas vezes por cada ciclo de *clock*, na mudança de pulso superior e inferior [48]. Estas conexões normalizadas DDR surgem da associação de *solid state tecnology*, chamada de *Joint Electron Device Engineering Council* (JEDEC), que procura servir o interesse publico para eliminar desentendimentos entre fabricantes e compradores, facilitar a permutabilidade e melhorar os produtos [63]. À medida que as tecnologias de memória vão evoluindo, surgem também novos tipos de DDR com melhor desempenho e mais eficiência energética. Atualmente, o DDR4 é o tipo de memória mais atual desta evolução [64] [48].

As memórias DDR mais relevantes e utilizadas nos últimos modelos de computadores e servidores do mercado são tipicamente baseadas em DDR3 e DDR4. Nos servidores, onde se tira um melhor proveito das *Dual In-line Memory Modules* (DIMM), o consumo de energia das memórias DDR3 pode variar entre 5 e 21 w*atts* por módulo consoante o nível de utilização dos seus recursos [62]. No entanto, um módulo deste tipo de memória no estado ativo, tem uma média de consumo de energia que varia entre 5 e 12 *watts*. Como é possível visualizar na Figura 2, em comparação com a tecnologia anterior, DDR2, existe uma redução média no consumo de energia de aproximadamente 1 *watt* quando se mantém a mesma frequência e capacidade de armazenamento. Desta forma, verifica-se também uma maior eficiência energética com a evolução da tecnologia aplicada nos módulos das DIMM [62]. A tecnologia DDR3, para além da típica normalização, dispõe de um tipo de DIMMs de baixa voltagem que, por sua vez, apresenta um baixo consumo. Essa adaptação, conhecida por DDR3L (DDR3 - *Low Voltage*), é mais utilizada pelos computadores portáteis modernos e tem tido uma relevante atenção por parte das empresas de TI. A grande diferença entre os módulos DDR3 e DDR3L resume-se à gestão de voltagens aplicadas. Por exemplo, as memórias DDR3L são de dupla voltagem, isto é, suportam operações de 1.5V e 1.35V. No caso das memórias DDR3, estas são de uma única voltagem e suportam apenas operações de 1.5V [64] [65]. Atualmente, as empresas de TI têm estado atentas a esta nova tecnologia DDR3L e têm intenções de aplica-la nos seus servidores [66] [67] [68]. Considera-se que, este tipo de memória, utiliza menos energia sem comprometer o desempenho ou fiabilidade do sistema. Para além disso, estima-se que a redução do consumo de energia no subsistema de memória pode atingir os 40%. Por exemplo, os módulos DDR3L que foram utilizados no projeto apresentaram consumos que variam entre 0,5 e 2 *watts* por módulo, consoante o nível de atividade do componente. Esta tecnologia traz

grandes vantagens na redução dos custos de funcionamento dos servidores e dos sistemas de refrigeração dos mesmos [66].

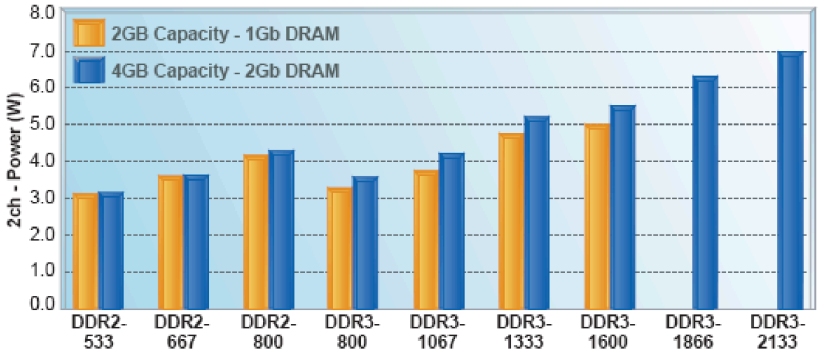


Figura 2 – Diferença do consumo de energia entre a tecnologia DDR2 e DDR3 [62].

Com a evolução rápida das memórias primárias, o tipo de memória DDR4 pode ser outra atração para as empresas e particulares. Esta tecnologia traz ainda melhor desempenho e funcionalidades de controlo mais robustas do que versões anteriores. Para além disso, é uma tecnologia com melhor economização de energia e uma voltagem de operações mais reduzida, de apenas 1.2V [64]. Por outro lado, este tipo de memória é ainda bastante dispendioso para um investimento em larga escala e não apresenta ainda nenhuma norma para funcionamento de baixa voltagem como existe com o tipo DDR3L. Nestes últimos tempos, os fatores que mais têm contribuído para a redução do consumo de energia nas memórias DRAM são ao mesmo tempo a redução de voltagem e as melhorias nas tecnologias de processamento. Como se pode ver na Figura 3, a voltagem das memórias tem vindo a reduzir, embora cada vez menos, e as melhorias das tecnologias de processamento têm sido menores. Existe atualmente uma maior preocupação em adotar novas tecnologias de funcionamento e tornar as memórias mais eficientes. Para isso, projetam-se, por exemplo, mecanismos para tornar as voltagens dinâmicas e mais adaptáveis às necessidades do sistema e o desenvolvimento de diferentes estados de gestão de energia [69].

Ao contrario dos servidores, desktops e portáteis, os smartphones utilizam a tecnologia de memória *Low Power Double Data Rate* (LPDDR) que se diferenciam das DDR por apresentarem uma voltagem, frequência e tamanho mais reduzido [70] [71]. Por exemplo, a Samsung afirma que uma memoria em LPDDR2 quando comparada com outra em DDR3, apresenta um consumo de energia 70% mais baixo [72]. Apesar disso, o desempenho de uma tecnologia DDR é bastante superior à tecnologia LPDDR, principalmente por apresentar uma maior taxa de transmissão do que a tecnologia de baixo consumo [71]. Nesta tecnologia o funcionamento difere em relação às conexões DDR, principalmente, nos registos de memória e protocolo de comandos. Para além

disso, ao contrario do habitual encaixe das DIMM, as memórias LPDDR são soldadas diretamente na placa do dispositivo móvel [73]. Os *tablets*, sendo uma evolução entre o mercado móvel e de computadores portáteis, têm opções que se situam entre os dois segmentos. As soluções apresentadas para os *tablets* estão entre os dispositivos móveis de alto desempenho e os portáteis de baixo consumo. Sendo assim, ao existir a possibilidade de configurar o sistema com processadores de diferentes tipos, as configurações de RAM podem variar entre o LPDDR e o DDR3 [71].

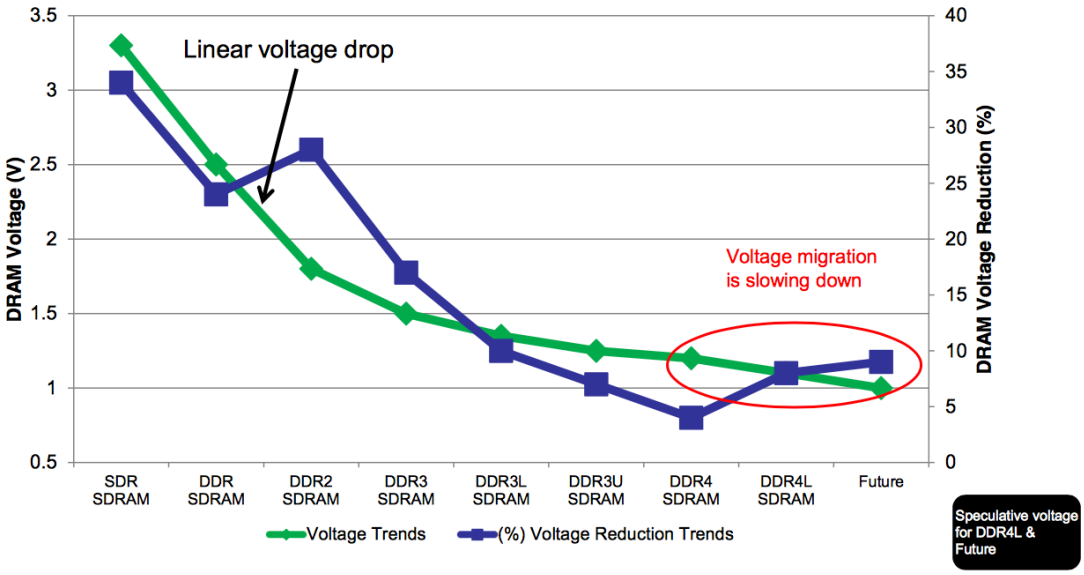


Figura 3 – Evolução dos processadores e respetivos efeitos na redução das voltagens da memória DRAM [69].

2.2.3 Armazenamento Secundário

Um computador armazena aplicações e dados em armazenamentos secundários quando estes não estão em processamento. O armazenamento secundário é persistente, o que significa que os dados mantidos neste armazenamento não desaparecem, à exceção de quando ocorrem erros. O termo "não volátil" é o mais indicado e utilizado para descrever o armazenamento secundário como sendo considerado persistente. Atualmente existem dois tipos de dispositivos para armazenamento secundário, o HDD e o SSD. O HDD é atualmente o principal entre os dois para utilização em Servidores de *data centers* e a sua taxa anual de erro está entre os 0,3% e 3% [24] [26] [27]. Apesar disto, os SSD estão a tornar-se um componente crucial das hierarquias modernas dos *data centers* [74] [75]. No que toca aos SSD existem ainda algumas dúvidas sobre a resistência e fiabilidade dos dados guardados, a falta de normalizações estabelecidas pela industria destes componentes e o elevado custo deste componente em relação aos HDD [76].

O subsistema de armazenamento secundário dispõe de vários estados de energia com tipos de funcionamento distintos [32]. Estes estados são selecionados consoante os níveis de utilização do subsistema e as configurações previamente definidas pelo utilizador no sistema operativo.

A Figura 4 mostra os estados possíveis de funcionamento de um disco HDD e as suas alterações de atividade entre cada estado. Os HDD modernos deste subsistema apresentam pelo menos três estados: *active, idle* e *standby* [32]. Na sua maioria, ainda se fazem acompanhar de um quarto estado, identificado como *sleeping*.

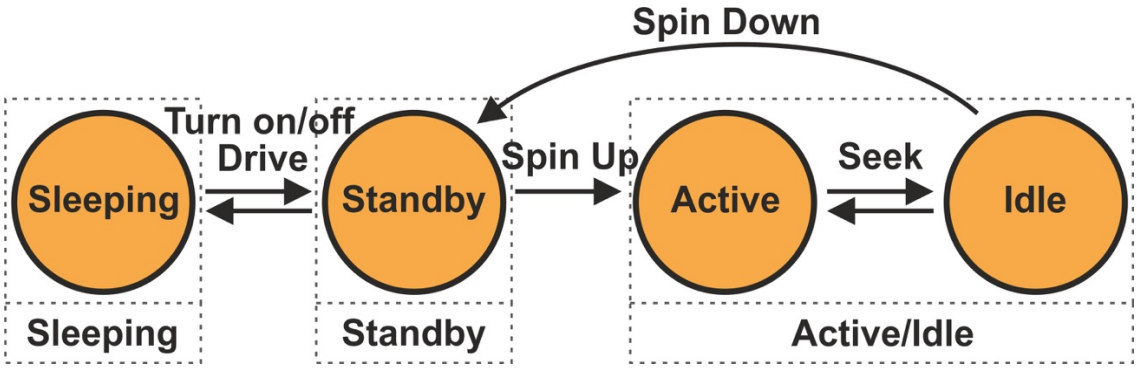


Figura 4 – Máquina de Estados de Energia e Operação do HDD (Adaptado de [32] [77]).

Os SSD apresentam-se habitualmente com dois estados: *active* e *idle* [78]. Os estados de energia são uma ajuda interessante para se estudar o modelo para a estimativa de consumo de energia. Não obstante, existe a desvantagem do subsistema sinalizar apenas três estados ao Sistema Operativo, independentemente do tipo de disco, isto porque, associa o conjunto dos estados *active* e *idle* num só [77]. Desta forma, é possível obter apenas informação de que o armazenamento secundário está em funcionamento sobre um dos seguintes estados:

- ***Active/idle***: modo de funcionamento normal. No caso dos discos HDD isto sugere que os pratos estão em rotação. Em ambos os tipos de disco, não é possível reconhecer se dados estão a ser lidos ou escritos;
- ***Standby***: modo de funcionamento de baixa potência. No caso dos discos HDD, os pratos não estão em rotação;
- ***Sleeping***: modo de consumo de energia mais baixo em que o dispositivo está completamente desligado.

Geralmente, os discos HDD e SSD fazem-se acompanhar de técnicas que aumentam o desempenho de escrita e leitura. A técnica de ***Write-Caching***, que se baseia na utilização de memória *cache* reservada do componente, torna os discos mais eficientes no seu procedimento de escrita, mas, ao mesmo tempo, mais vulneráveis a perdas de dados. Quando o sistema operativo pretende guardar algo em disco, se a

memória *cache* apresentar espaço suficiente, os dados são primeiramente guardados nesse espaço e, posteriormente, gravados em disco quando o controlador considerar oportuno. Embora o *cache* melhore o desempenho do disco, existe algum risco envolvido neste processo. Se um computador falhar (devido a uma falha de energia, por exemplo), o sistema pode não ter tempo de copiar os dados em *cache* para o disco. Nesse caso, as atualizações de dados realizadas serão perdidas [79]. A memória *cache* do disco está habitualmente incorporada no componente, no entanto, principalmente em discos SSD, pode ser utilizado um intervalo de memória livre em memória RAM destinado a este propósito [78] [48]. Além desta tecnologia no processo de escrita em disco, existe também uma técnica para melhorar o desempenho de leitura em disco. A técnica ***Read-Ahead*** [80] pode ser feita ao nível de *hardware*, no próprio disco, ou, a nível de *software*, pelo sistema operativo, e envolve a deteção ativa de um padrão de acesso ao disco e no tratamento de dados com base nesse mesmo padrão. Desta forma, é feita uma previsão sobre a quantidade e localização de dados que serão futuramente precisos d, estes serão previamente lidos antes de, sequer, serem requisitados pelo software. Esta técnica apresenta três vantagens a destacar. Primeiro, os atrasos de IO são omitidos à vista do *software*, visto que, quando uma aplicação requisita um bloco, este já foi previamente lido e está pronto a ser utilizado. Em segundo lugar, os discos são mais bem utilizados para grandes pedidos de leitura sequencial. Por ultimo, esta técnica amortiza o processo de *overhead* nos acessos ao componente.

Os discos HDD apresentam na sua estrutura física vários elementos imprescindíveis para o seu funcionamento. Na Figura 5 está representado um diagrama dos blocos que figura vários elementos fundamentais deste tipo de disco. Os HDD são constituídos por componentes como a cabeça, que é motorizada pelo *Voice Coil Motor* (VCM), os pratos, que se movimentam com o apoio *do Spindle Motor* (SPM), e todos os controladores e outros mecanismos de apoio ao funcionamento do disco. Entre diferentes modelos e opções dos fabricantes, surgem funcionalidades diferentes que podem ter tanto vantagens como desvantagens. As cabeças podem ser amovíveis ou não, sendo que, é uma desvantagem o facto de a cabeça estar fixa, devido a ser forçada a estar focada apenas numa faixa e, provavelmente por isso, tipicamente são usadas as amovíveis. Os discos podem ser compostos por um ou mais pratos e, para além disso, esses pratos podem ser de funcionamento único de um dos lados ou de funcionamento de duplo lado. Os pratos de duplo lado de funcionamento têm tendência a ser mais dispendiosos. Estas duas caraterísticas referentes aos pratos obrigam a configurações diferentes também das cabeças que, necessitam de ser múltiplas e funcionar em conjunto nas mesmas posições para todos os pratos. Cada prato apresenta várias faixas na sua superfície em que, cada uma, apresenta um

conjunto de blocos de dados. As faixas e os blocos separam-se por pequenos espaços a fim de prevenir ou minimizar erros de leitura da cabeça ou interferência de campos magnéticos.

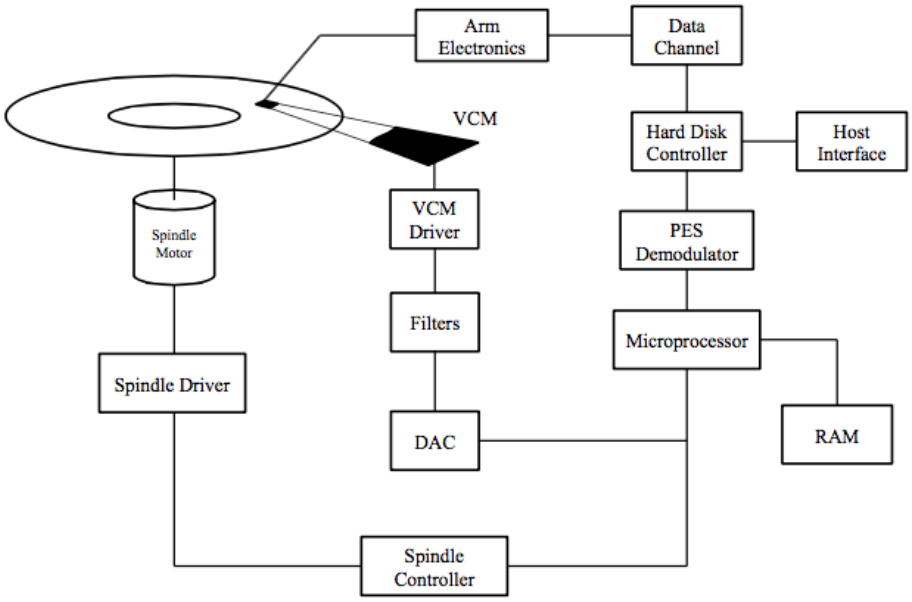


Figura 5 - Diagrama de blocos de um sistema de disco rígido [81].

Os dados são transferidos entre o HDD e o Sistema Operativo em blocos de tamanho fixo [48]. Tipicamente, os discos dividem-se em blocos físicos de 512 *bytes*. Apesar disso, os HDD mais recentes funcionam com tecnologia *Advanced Format* (AF), isto é, com blocos físicos de 4096 *bytes*. Apesar disso, os dispositivos têm a possibilidade de se exporem externamente como discos distribuídos em blocos lógicos de 512 *bytes* para serem compatíveis com determinados sistemas operativos [82] [83].

Nos discos HDD existem três fontes de consumo de energia: o SPM, o VCM e os componentes eletrónicos. [84] Da energia total de *idle*, 50% são consumidos pelos subsistemas eletromecânicos e os restantes 50% são consumidos pelos componentes eletrónicos [81]. A energia consumida pelo VCM geralmente faz parte da porção dinâmica do HDD, e ocorre apenas em certos momentos, durante a atividade de *seek* do disco. O consumo de energia do SPM varia consoante o numero de pratos montados no disco, o raio dos pratos e a velocidade angular do SPM que inclui, por exemplo, as rotações por minuto [85]. Além do consumo de energia, as rotações por minuto afetam também o desempenho do disco.

Os Discos SSD, ou memória *flash* NAND, apesar de terem a mesma finalidade que os discos HDD, não apresentam um funcionamento mecânico com pratos e cabeças. [76] Este tipo de disco armazena os seus dados em vários *flash packages* que

guardam os dados, ainda que, não exista energia a alimentar o dispositivo [86]. Cada *flash package* apresenta *chips* de memória *flash* interligados, com autonomia para realizar tarefas diferentes, sendo que, um *chip* pode estar a receber dados enquanto o outro está a realizar outras operações. A memória *flash* USB, apesar de se agrupar no mesmo tipo de memória, é tecnicamente incomparável com os discos SSD. Ambos utilizam memória *flash* NAND, no entanto, é a qualidade do NAND utilizado, assim como o controlador e a interface envolvida, aquilo que separa uma simples *flash* USB *drive* de um SSD. Para além disso, são tecnicamente incomparáveis devido à consideravelmente maior fiabilidade dos SSD e, consequentemente, estes são mais dispendiosos e de maior capacidade [76]. Existem dois tipos de memória não-volátil no mercado de memórias *flash*, NOR e NAND. Atualmente, as memórias NOR são pouco utilizadas no ramo, devido à introdução do tipo de memória NAND. A grande diferença revela-se no acesso às células, que nas memórias *flash* NOR é aleatória e, no caso das memórias *flash* NAND, o acesso é sequencial e em conjuntos, isto é, em blocos de célula em vez de aceder de forma individual [87] [76].

A comunicação entre o SSD e o exterior requer equipamento de compatibilidade e tradução incorporado. Como se pode ver na Figura 6, o SSD deve necessariamente conter um componente *Host Interface Logic* para suportar algumas conexões como USB ou *Serial AT Attachment* (SATA) e um *Logical Disk Emulation*, isto é, uma camada com mecanismos de tradução para permitir que o SSD se exponha para o exterior, de uma forma similar ao HDD. Na Figura, pode ainda encontrar-se um gestor interno do *buffer,* que interpreta os pedidos chegados do sistema operativo, e um multiplexador, que encaminha o transporte de dados pelas conexões de serie, de e para, os *flash packages*. Por sua vez, o processador e a RAM são blocos necessários a fim de gerir o ritmo dos pedidos e o mapeamento dos blocos lógicos do disco para a localização física de memória *flash* [86]. Os discos SSD têm também estruturas físicas, similares aos blocos nos discos HDD, denominadas de páginas [88]. Estes discos têm diferentes tamanhos de página dependendo do modelo. Estas paginas são depois organizadas em blocos. No topo destas estruturas existe uma camada abstrata que faz com que os SSD se exponham ao sistema operativo como um tradicional HDD [82].

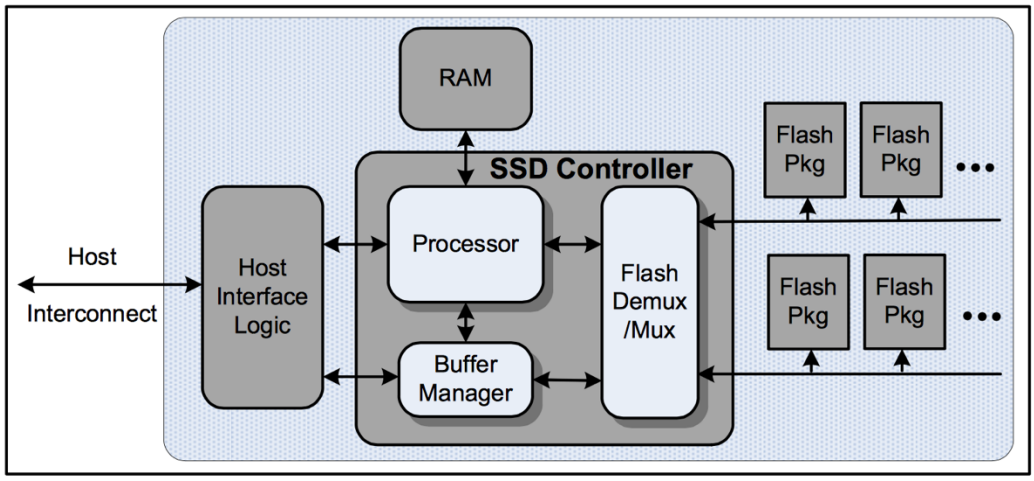


Figura 6 – Diagrama de blocos de um SSD [86].

Os SSD, ao não apresentarem componentes móveis no seu interior, tornam-se mais fiáveis no que toca a manter os dados seguros e menos expostos, numa queda imprevista do dispositivo ou, até, em períodos de simples turbulência, como por exemplo, numa viagem de comboio. As memórias *flash* têm vindo a impor-se no mundo da informática, sendo que, nos dispositivos móveis, são o único tipo de memória a ser utilizado devido ao seu tamanho adaptável. Ao contrário dos HDD, os SSD apresentam a vantagem de não utilizarem pratos que limitam bastante a redução de tamanho e compactação [89]. Nos HDD, por exemplo, a redução de tamanho não conseguiu transpor as 1,8 polegadas, onde apenas foi possível a obtenção de um máximo de 320GB de armazenamento.

Além destes dois tipos de disco, existem ainda os discos híbridos, que juntam a capacidade dos HDD com a velocidade dos SSD ao colocar os tradicionais pratos rotativos e uma pequena quantidade de memória *flash* de alta velocidade, tudo isto num único dispositivo. Os produtos de armazenamento híbrido monitorizam os dados que estão a ser lidos do disco rígido, e dirigem os *bits* que são mais frequentemente acedidos para a memória *flash* NAND. A memória *flash* NAND funciona como *cache* e, por isso, muda o seu conteúdo com o tempo. Uma vez que, os dados mais frequentes são armazenados na memória *flash*, estes vão ser servidos diretamente pela memória de SSD, resultando num desempenho idêntico ao de um SSD, para os ficheiros mais utilizados. [90] [91]

A nível comercial, apesar dos discos HDD serem lideres de mercado, os discos SSD têm vindo a tornar-se um forte candidato para primeira escolha de armazenamento, devido à sua maior eficiência energética e ao seu acesso aleatório mais rápido [74] [75]. No entanto, a diferença de preço entre os dois tipos de disco é significativa e, isso, é certamente um dos fatores que justifica o facto de o SSD ainda

não ter assumido o lugar pioneiro na escolha dos fabricantes. À vista do consumidor particular, numa analise sobre os HDD e SSD de 1TB mais procurados na loja online Amazon, verifica-se que o SSD apresenta um preço quase 8 vezes superior ao de um HDD [92] [93]. Neste exemplo em concreto, ao analisar o preço por *gigabyte* de cada um dos discos, verificou-se que o HDD apresenta um preço de 5 cent./gb e o SSD apresenta um preço de 38 cent./gb. No presente, as unidades de SSD estão disponíveis até à capacidade máxima de 4TB mas, essas são ainda soluções muito raras e dispendiosas [89]. O preço por *gigabyte* apresentado na análise justifica-se também por ser uma comparação de discos com capacidade 1TB, que, no caso dos SSD, têm um crescimento de preço por capacidade, bem superior ao que se verifica no caso dos HDD. Por exemplo, no exemplo do HDD da Amazon, entre a capacidade do HDD de 1TB para 2TB existe um aumento de preço de 150%. No caso de um SSD, abordando as mesmas capacidades do caso do HDD, verificamos um aumento de 220%. É certo que não existe só um produto, nem um vendedor, e é natural que as Empresas de TI tenham outras regalias e outros preços que os consumidores finais não conseguem ter.

2.3 Interfaces de Monitorização e Gestão de Energia

Como já se verificou anteriormente, a monitorização e gestão de energia é uma atividade importante e imprescindível para se estudar e controlar a eficiência energética de um sistema. Atualmente, a eficiência energética é um assunto que acarreta uma maior preocupação e, consequentemente, todos os fabricantes de componentes têm em atenção os consumos energéticos provocados pelos produtos que desenvolvem. Desde há alguns anos que existem especificações normalizadas, como o *Advanced Power Management* (APM) e o ACPI, que unificam os acessos e monitorização energética dos componentes desenvolvidos. No caso dos processadores, por outro lado, os seus fabricantes têm-se afastado um pouco dessas especificações e desenvolvido as suas próprias interfaces e técnicas de gestão. Estas empresas têm equipado os seus sistemas com novas ferramentas por forma a obter modelos de energia extremamente precisos, através de sensores aplicados no processador com o objetivo de medir o consumo de energia nos *cores* do CPU e da análise do código que é executado nesses componentes. Na gestão e monitorização de energia, a Intel introduziu uma interface chamada *RAPL,* a AMD apresentou a ferramenta *Bulldozer* e, numa vertente mais direcionada para os processadores gráficos (GPU), a NVIDIA lançou a ferramenta NVIDIA *Management Library* (NVML) [94] [95]. Estas interfaces privadas e, unicamente destinadas aos produtos do fabricante, têm a grande vantagem de serem aperfeiçoadas para determinados

componentes e permitem a melhoria de qualidade das funcionalidades de monitorização e gestão nos seus próprios produtos. No subsistema do processador, apesar de existirem ferramentas implementadas por diversos fabricantes, serão apenas estudadas aprofundadamente as soluções da fabricante Intel. Esta escolha deve-se ao facto de, depois de se estudarem as várias opções no subcapítulo 2.2, considerar-se que este se adequa melhor aos objetivos deste projeto.

2.3.1 *Advanced Power Management*

Uma das primeiras especificações a disponibilizar a gestão de energia foi a solução APM [96] [97]. O APM consiste num conjunto de camadas de *software* que suportam a gestão de energia em sistemas compostos por *hardware* compatível com essas funções. O APM define, independentemente do *hardware*, uma interface de *software* entre o *software* de gestão de energia específico do *software* e o *driver* de politicas de gestão de energia do sistema operativo. Por este meio, torna-se possível o controlo e monitorização de *software* de alto nível sobre o equipamento sem conhecimento especifico de configurações de fábrica do mesmo. Esta politica permite o controlo sobre o consumo destes dispositivos e facilita o acesso ao dispositivo que se torna indiferente, independentemente do fabricante em questão. A especificação desta interface define um conjunto de diferentes camadas em que, aplicações, sistemas operativos, *drivers* de dispositivos e a BIOS trabalham em conjunto de forma a reduzir o consumo de energia. Tanto a vida das baterias do sistema, como a sua eficiência e produtividade acabam por apresentar resultados bem mais positivos .

Apesar da especificação APM ter sido um passo importante para a poupança de energia num sistema, recentemente esta foi sendo ultrapassada. Esta mudança de corrente deve-se à dificuldade e inflexibilidade para os sistemas operativos, através do APM, gerirem a energia utilizada e as propriedades térmicas do sistema. O *hardware* era gerido pela BIOS e o utilizador era confrontado com importantes limitações de configuração e monitorização nas definições de gestão de energia. O APM BIOS é desenvolvido pelo fabricante e é específico para determinada BIOS, separadamente do sistema operativo. Por exemplo, o utilizador tem a possibilidade de definir valores de *idle* para o disco rígido na BIOS do APM, no entanto, quanto é atingido esse valor, a BIOS ordena a paragem de rotações do disco sem o consentimento do sistema operativo. Além disso, com estas limitações específicas ao fabricante, as atualizações de correção de defeitos na BIOS estavam a ser constantemente sobrepostas para agradar a cada fabricante e, existia dificuldade em resolver os variados problemas. Para além disso, sendo a logica da especificação APM totalmente implementada sobre a BIOS, os utilizadores apenas podiam corrigir os

problemas atualizando a BIOS, que por si só era um procedimento perigoso com risco de provocar danos irreparáveis ao *hardware*. Desta forma, era necessário o aparecimento de uma nova solução visto que a APM BIOS não tinha capacidade suficiente para implementar politicas de energia mais sofisticadas ou adaptáveis ao propósito do sistema em questão [97].

2.3.2 Advanced Configuration and Power Interface

O sucessor do APM é denominado de ACPI e surgiu com o intuito de contornar os problemas encontrados no primeiro caso. A interface não pode trabalhar em simultâneo com o APM devido a sobreposição de configurações [97]. Esta especificação de livre acesso para a industria, desenvolvida em conjunto pela Hewlett-Packard, Intel, Microsoft, Phoenix e Toshiba, está presente em todos os computadores modernos que apresentem processadores com arquitetura x86. O ACPI estabelece interfaces industrialmente normalizados e com configurações direcionadas ao Sistema Operativo. Essas interfaces possibilitam uma melhor gestão de energia e gestão térmica de plataformas móveis, desktops e servidores. A especificação permite que novas tecnologias de gestão de energia se envolvam independentemente nos sistemas operativos e *hardware* enquanto se mantêm a trabalhar em conjunto. Assim, o Sistema Operativo pode reter informação ao nível do *hardware* do sistema e do nível aplicacional de maneira a tomar melhores decisões de gestão de energia. A norma na integra apresenta muitas funcionalidades, incluindo a gestão de desempenho do processador, planos de controlo de energia consumida, zonas térmicas, variados sistemas de bateria, controladores embebidos, e enumeração de *bus*. A maior parte dos sistemas implementa apenas uma parte de todo este conjunto de funcionalidades. Por exemplo, um sistema *desktop* habitualmente implementa unicamente enumeração dos dispositivos e adaptadores do *bus,* ao passo que um portátil apresenta normalmente, para além disso, técnicas de gestão de refrigeração e bateria e métodos mais elaborados para suspensão e resumo do sistema [97]. A especificação do ACPI define o funcionamento do sistema em vários estados [98] [99]. Sendo esta especificação uma evolução da tecnologia APM, as duas não podem estar ativas em simultâneo no sistema, tal como já referido anteriormente.

2.3.3 Estados de Gestão de Energia num Sistema

A gestão de energia de um sistema é um domínio chave para que se obtenha melhores resultados de eficiência energética. Monitorizar e tomar decisões sobre os momentos mais indicados para "adormecer" um dispositivo, ou gerir frequências e

voltagens de um processador, são boas soluções para haver bons resultados na poupança de energia. Estas soluções podem, consequentemente, aumentar a vida da bateria dos dispositivos móveis, diminuir os exercícios das ventoinhas instaladas nos *desktops* e, claro, reduzir o consumo de energia e custos de refrigeração dos servidores.

A especificação APM define-se em quatro diferentes estados gerais do sistema: o *Ready, Stand-by, Suspended* e *Off*. Três destes estados aplicam-se simultaneamente aos componentes individuais do sistema e ao sistema no seu todo. O estado *Suspended* funciona numa condição especial de baixa potencia que se aplica ao sistema como um todo e não individualmente a cada componente. Para além deste grupo de estados, o controlo específico relativo aos componentes divide-se em dois tipos: o *Device Control* e o CPU *Core Control*. Ambos os tipos são controlados pelos quatro estados gerais da especificação que foram referidos anteriormente [96].

Esta secção estuda o atual funcionamento do ACPI em Linux na gestão de consumo energia do sistema. Desta forma, apresentam-se as decisões predefinidas que são tomadas pela interface e pelo sistema operativo afim de haver uma maior poupança de energia. Estas decisões ocorrem com base em vários fatores, tais como o nível de utilização do sistema e a temperatura dos componentes.

A Figura 7 resume simplificadamente a distribuição dos estados existentes e a relação entre eles. Ao analisar a mesma, é possível verificar que existem inúmeras combinações possíveis de estados de poupança de energia. Sendo a especificação desenvolvida com o intuito de universalizar a gestão de energia para os diversos fabricantes, estes seguem esta regra generalizada dos estados. De qualquer forma, pode sempre haver pequenas alterações que se aperfeiçoam aos seus próprios componentes desenvolvidos. A Intel, mais especificamente, apresenta uma gestão mais adaptada e estados mais aperfeiçoados aos seus processadores, nomeadamente nos estados C e P.

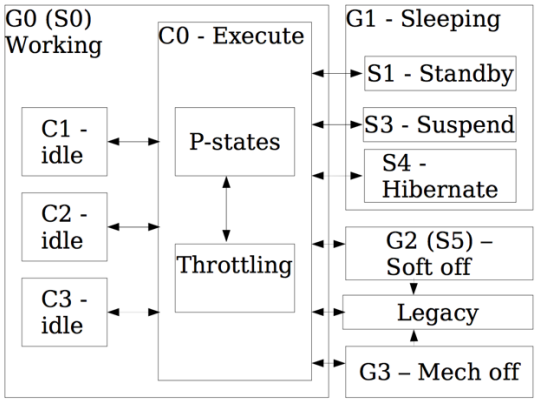


Figura 7 – Diagrama do ACPI Global, CPU, e *Sleep states* [99].

2.3.3.1 Global System States

Os *Global System States* (G-*States*) descrevem todo o sistema e, as transições entre estes estados, são tipicamente obvias para o utilizador. No estado G0, também identificado como *Working* ("em-serviço"), o computador está no seu modo normal de funcionamento e é possível executar *threads* de aplicações do utilizador. Neste modo, existe o maior conjunto de estados para gestão de processadores visto que o G0 é o estado mais importante e mais funcional do computador. O estado G1 intitula-se de *Sleeping* e, tal como o nome indica, o sistema está adormecido e aparenta estar desligado. No entanto, este pode ligar a qualquer momento consoante os eventos externos ocorridos e, com base na configuração previamente definida, isto é, o sistema neste estado não executa aplicações, mas pode acordar utilizando timers ou aquando da receção de um pacote de rede. Logicamente este estado tende a consumir bastante menos que o estado G0. Os restantes estados G entendem-se como estados de inatividade. No estado G2, ou *Soft-off*, o sistema está desligado de uma forma mais suave, podendo retornar a um modo funcional através de um sinal elétrico evitando qualquer ação física. O estado G2 apresenta um consumo de energia praticamente nulo. No caso do estado G3, ou *Mechanical Off*, este surge quando há uma desconexão elétrica mediante uma ação mecânica. Para além disso, para retornar a um estado de funcionamento, é também necessário que haja uma ação mecânica [98] [100].

2.3.3.2 Processor Power States

A especificação ACPI define quatro *Processor Power States* (C-*States*), de C0 a C3, para definir diferentes estados de *idle* para o processador, consoante o seu historial de atividade e as suas necessidades do momento. No contexto do estado G0, em que o computador está no seu normal funcionamento, o C-*State* C0 corresponde ao estado de execução do CPU e os estados superiores a C0 (C1-CN) são estados de *idle* do CPU, em que não há qualquer instrução a ser executada. Consequentemente, é esperado que o processador consuma menos. Quanto maior for o estado, maior é a poupança de energia, mas, por outro lado, maior é a latência para retornar desse estado para o modo mais operacional. As transições entre o C0 e o C1 ou outros C-*States* são realizadas pelo sistema operativo nos períodos de *idle*. Na especificação ACPI, apenas o estado C0 e C1 são necessários, sendo que os outros estados são opcionais.

Atualmente, a especificação ACPI tem sido evitada pela Intel para a governação dos C-*states* e P-*states*, abordados no próximo ponto. Len Brown, Engenheiro Principal da Intel *Open Source Technology Center*, afirma que o fabricante tem encontrado algumas desvantagens no uso da ACPI [101]. Destas desvantagens, Brown

destaca erros na BIOS, latências de C-*State* imprecisas e um conceito pouco esclarecedor de diferenciação entre *core* e *package*. Com estes vários obstáculos aos novos produtos do fabricante, a Intel apresenta agora o Intel_*idle driver* que se implementa nos seus modernos processadores por forma a substituir o *driver* mais comum, ACPI *processor_idle*. Este *driver*, desenvolvido pela Intel, torna o Linux mais eficiente nos processadores específicos do fabricante. Esta melhoria deve-se ao facto de este *driver* ter um maior conhecimento sobre o *hardware* que o ACPI e, para além disso, torna o Linux mais imune aos *bugs* da ACPI na BIOS [102] [103]. Nos processadores i7 verificou-se, com a utilização do driver intel_*idle*, uma redução de consumo de energia de 15% e uma melhoria no alto desempenho do sistema. [104]

2.3.3.3 Processor Performance States

Os *Processor Performance States* (*P-States*) são extremamente importantes na gestão de consumos de processadores, principalmente porque modelam simultaneamente as frequências e as voltagens do processador. Estes estados entram em funcionamento quando o *G-State* está em G0 e o *C-State* em C0. Os *P-States* foram desenvolvidos com o intuito de reduzir a energia do processador em execução e tornam-se bastante eficientes por estarem diretamente relacionados com as voltagens do processador. A norma ACPI define um conjunto de estados de desempenho que permitem que um sistema, em pleno funcionamento, varie o seu consumo de energia e desempenho ao funcionar com vários níveis de voltagem e frequência. Se um CPU for compatível e capaz de funcionar em vários modos de desempenho, estes podem ser mapeados utilizando estes *P-States*. O ACPI permite até um máximo de 16 estados de desempenho distintos. Por outro lado, existe a condição de que, à medida que o numero do estado aumenta, há um decréscimo de desempenho e consumo de energia. Desta forma, o estado P0 é aquele que apresenta um melhor desempenho e, consequentemente, um maior consumo de energia. Por sua vez, este é o estado que funciona com níveis de voltagem e frequências mais altas. A interface ACPI permite que componentes de alto nível (como o Sistema Operativo) controlem explicitamente o consumo de energia do sistema. O Sistema Operativo pode reunir informação tanto a nível do sistema (*hardware*) como a nível aplicacional, de maneira a tomar melhores decisões de gestão de energia [100] [98].

A partir da versão de *kernel* 3.10, o Linux inclui o *driver intel_pstate*, que se resume a uma nova plataforma de escalonamento de potência para processadores modernos de arquitetura Intel *Sandy Bridge* ou superior. Este *driver* gere os níveis de frequência e voltagem seguindo as capacidades especificas do CPU, em vez de depender da universal e mais limitada norma ACPI. O *intel_pstate driver,*

desenvolvido pela Intel, inclui algoritmos de controlo específicos e aperfeiçoados aos CPUs do fabricante. Esta aposta do líder de processadores do mercado, faz com que o sistema faça um julgamento mais preciso sobre como a plataforma *P-State* se deve adaptar, enquanto se mantem, por sua vez, uma maior poupança de energia. Neste modo, consoante o modelo do processador, o *driver* pode operar em dois diferentes modos, *legacy* e *Hardware P-State*. O modo *legacy* ramifica-se em dois diferentes tipos de governação: *performance* e *powersave*. Como os nomes sugerem, o modo *performance* maximiza o desempenho dos *cores* e, por sua vez, eleva ao máximo o consumo de energia do processador. No caso do *powersave*, este seleciona o estado P apropriado baseado nos níveis de carga e utilização do CPU. Nestes dois modos do grupo *legacy*, o controlo é feito pelo governador Intel e, por essa razão, o utilizador não tem controlo sobre estas decisões. O modo *Hardware P-State* é implementado no próprio processador. Ao contrario do modo *legacy*, neste modo as decisões são feitas com base nas preferências definidas pelo utilizador e a frequência reportada pelos *cores*. Neste caso, o governador Intel *P-State* está desativado [103] [105].

2.3.3.4 Throttling States

Os *Throttling States* (*T-States*) ajustam apenas a frequência do processador, deixando inalterada a sua voltagem. Atualmente o Linux utiliza apenas o *Throttling* para responder a eventos térmicos quando o processador dá sinais de estar demasiado quente. Os seus estados variam consoante a temperatura detetada pelo sensor térmico do processador em que, quão maior é a temperatura indicada, maior é o *T-State*. É importante referir que grande parte dos processadores também incluem um mecanismo de monitorização térmico resguardado no hardware, definido com maiores limites de temperatura que o *Throttling* do ACPI. Para além disso, estes acompanham-se também de um mecanismo de desligar automaticamente o sistema em caso de emergência térmica [98].

A Intel indica que, nos seus processadores, os *T-States* são atualmente irrelevantes. A empresa justifica-se apontando que antes do aparecimento e evolução dos *P-States* e *C-States*, os *T-States* existiam para salvar os processadores de queimarem em situações de alta utilização ou de uma avaria na sua refrigeração. A Intel revela ainda que algumas estruturas de dados de gestão de energia, como as definidas pelo ACPI, ainda incluem um campo não usado de *T-State* [106].

2.3.3.5 Sleep States

Os *Sleep States* (*S-States*) estão disponíveis assim que o sistema se apresenta no estado G1, ou *Sleeping State*. Estes estados podem entender-se como subestados do

estado G1 ou, diferentes formas do sistema se apresentar "adormecido" e da energia aplicada ao processador. Para conservar energia enquanto se está rapidamente disponível, os PCs compatíveis com ACPI utilizam os estados *Sleep* do sistema. A especificação ACPI define 5 destes estados, conhecidos por *S-States*. Estes estados foram modelados com base nos estados do sistema da normalização APM, um precedente do ACPI [107]. Nestes estados, à exceção do S0, não há qualquer operação realizada pelo sistema. Seguindo a ordem crescente de estados deste grupo, cada estado reflete uma maior poupança de energia, mas, em contrapartida, necessita de mais tempo para "acordar" e retomar o seu trabalho em normal funcionamento. O estado S0, ao contrario dos restantes, corresponde ao sistema no seu pleno funcionamento, isto é, quando o sistema se apresenta no estado G0. Excluindo o estado S0, entre todos os estados, o S1 é o mais rápido a retornar a um estado de funcionamento. Neste estado o processador e a *cache* suspendem a sua atividade, mas mantêm o seu contexto. O mesmo acontece com o restante *hardware*. O estado S2 é similar ao S1, no entanto, tanto o contexto do CPU como o conteúdo da *cache* são perdidos devido ao corte de alimentação de energia do processador. No caso do estado S3, toda a energia para o CPU é desligada, e o seu conteúdo dos registos é reencaminhado para a RAM. No estado S4, a energia consumida pelo CPU mantêm-se nula como no S3, mas a RAM é escrita para disco e é desligada da alimentação nos mesmos termos do CPU. Por fim, o estado S5 agrupa-se no estado G2, sendo que, ocorre quando o sistema é completamente desligado. Apesar disso, este pode ser ligado através de eventos elétricos externos como eventos de LAN ou USB [107] [47] [100].

2.3.3.6 Device Power States

Apesar de não estarem presentes na Figura 7, a especificação ACPI define estados de poupança de energia para dispositivos e denomina-os por *Device Power States* (*D-States*) [47] [98] [100]. Estes estados representam métodos de operação uniformizados para atribuir a diversos dispositivos que em pouco ou nada se relacionam entre si. Desta forma, cada estado de energia atribuído a um dispositivo atua apenas sobre o mesmo. Por exemplo, a relação entre um dispositivo de vídeo e um dispositivo de áudio é praticamente nula, e, estabelecer técnicas de poupança universais e versáteis torna-se bastante difícil. Assim, para ajudar a ultrapassar esta limitação, a especificação ramifica os estados em diferentes classes para diferentes tipos de dispositivos. Nesse seguimento, é possível encontrar classes de dispositivos de áudio, dispositivos de armazenamento ou até dispositivos de rede.

Para definir os procedimentos específicos de cada dispositivo nos diferentes estados, a especificação tem em consideração quatro critérios relevantes: o consumo de energia do dispositivo, o contexto e importância do dispositivo no sistema, o que implica ao controlador fazer para que o dispositivo volte ao seu normal funcionamento e ainda o tempo que isso pode demorar [47] [98].

O ACPI define três estados de poupança de energia nos dispositivos, sendo que, o D0 corresponde ao estado ativo, o D3 ao estado desligado, e D1 e D2 a estados intermédios de poupança [98]. Apesar das diferenças especificas variarem consoante a classe, comparado o estado D3 com o D2, verifica-se um consumo de energia mais reduzido, uma maior latência para retomar ao seu estado normal de funcionamento e uma perda de contexto do dispositivo no sistema [100].

2.3.4 *Running Average Power Limit*

O RAPL é uma interface normalizada desenvolvida pela Intel para gerir o consumo de energia de componentes como o processador e a memória primária. Esta interface, introduzida primeiramente nos processadores Intel da arquitetura *Sandy Bridge*, consiste em Model Specific Registers (MSR) que não fazem parte da arquitetura do processador, mas endereçam valores de energia necessários para a gestão de consumos nos componentes das várias plataformas Intel [31] [44]. O seu principal propósito é definir limites de consumo de energia do processador e modificar isto sempre que necessário, sendo que, as consultas de energia acessíveis ao utilizador são uma funcionalidade de bónus do RAPL [108]. Vários artigos dão uso à interface pela funcionalidade de medir o consumo de energia [94] [109] [110] [111]. O RAPL não apresenta os valores com base num medidor de energia analógico, pois utiliza um *software* que implementa um modelo de energia em execução no controlador do chip principal do *Package*. A energia é estimada através da análise de vários contadores de performance de *hardware*, sensores de temperatura, fugas de energia e modelos de IO. [108] [112] Os registos reservados para as leituras de energia são atualizados aproximadamente a cada milissegundo (1kHz) [31] [44] [108] [94].

Como é possível verificar na Figura 8, o RAPL apresenta várias interfaces MSR para cada um dos seus domínios, de forma a ser possível controlar e monitorizar o consumo de energia de uma forma mais específica. Existem algumas diferenças entre a lista de domínios disponíveis, consoante o tipo de plataforma Intel instalada na plataforma. Em plataformas destinadas ao Cliente estão disponíveis os domínios *Package*, *Power Plane* (PP) 0 e PP1. Por outro lado nas plataformas para fim de Servidor estão disponíveis os domínios *Package*, PP0 e DRAM. [31] [113]

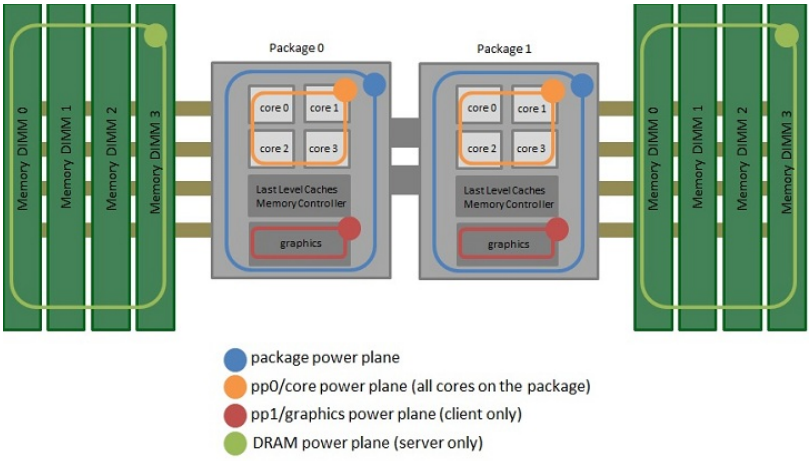


Figura 8 – Esboço ilustrativo dos quatro componentes disponíveis no RAPL [113].

Cada um destes domínios da interface RAPL é fonte de diferentes dados que distinguem conjuntos de componentes:

- ***Package***: Domínio referente aos consumos de energia do *package* (*socket*) inteiro, incluindo a energia do *core* e *uncore* [31].
- **PP0:** Domínio referente à energia consumida pelo PP0 que inclui todos os núcleos do CPU [31].
- **PP1**: Domínio referente à energia consumida pelo *PP1* que inclui algumas atividades *uncore* [31]. Em algumas versões da arquitetura *Sandy Bridge* este domínio reporta a informação de energia do GPU instalado *on-board* [108] [113] [114].
- **DRAM**: Domínio referente apenas à energia consumida nas DIMM. As interfaces MSR definidas para o domínio DRAM estão apenas disponíveis em plataformas de Servidor [31] [115] [116] [113].

As interfaces MSR disponíveis em cada um dos domínios previamente referidos são as seguintes:

- ***Power Limit***: Interface MSR para especificar a janela de tempo e o limite de energia a ser consumida pelo domínio. Para isso, o agente de *software* define os *watts* a ser consumidos e escolhe o espaço de tempo em que essa média de consumo necessita que ser controlada [31] [116].
- ***Energy Status***: Interface que disponibiliza informações da energia consumida que permite medir a potencia do domínio. O registo reporta a energia real utilizada pelo domínio. Este MSR é apenas de leitura [31] [116].

- ***Perf Status*** (opcional): Interface que disponibiliza informação dos efeitos de regressão de performance causados pelos limites impostos na potencia do domínio. Este MSR é apenas de leitura [31] [116].
- ***Power Info*** (opcional): Interface que apresenta informação de um conjunto de parâmetros para um dado domínio, como por exemplo, a potencia máxima e mínima ou até a temperatura do domínio em questão [31] [116].
- ***Policy*** (opcional): Interface que permite definir políticas de controlo de energia para distribuir os custos, isto é, balancear a potência consumida entre subdomínios do domínio [31] [116].

Com uma análise aos componentes que são abrangidos por cada domínio pode-se concluir que:

$$\text{energy (PP0)} + \text{energy (PP1)} \leq \text{energy (Package)} \quad (4)$$

Apesar da Intel apresentar uma secção dedicada ao RAPL no *Volume 3 of the Intel Architectures SW Developer's Manual* [31], encontra-se alguma inconsistência nas informações apresentadas sobre a ferramenta. Com isto, opera-se assim com uma perceção limitada sobre o que os registos do hardware nos podem transmitir, sendo que, alguns exemplos de inconsistência são:

- A documentação da Intel indica que uma plataforma com fim de Cliente inclui o domínio PP1, mas, não inclui o domínio DRAM e, no entanto, o sistema utilizado nesta investigação apresenta claramente valores para ambos os domínios [31] [113] [117].
- Em algumas plataformas (ex. Xeon E5 v3), o domínio PP0 deixou de ser suportado. Apenas os domínios *Package* e DRAM retornam valores diferentes de zero [118].
- Em algumas plataformas (ex. Xeon E5 v3), as unidades utilizadas para a energia da DRAM não são as mesmas unidades que são utilizadas para a energia do *Package* [118].
- As documentações Intel iniciais da arquitetura RAPL para plataformas com finalidade para Cliente indicam que a energia do GPU é incluída no PP1 [113], mas nas ultimas documentações apresentadas, a energia de GPU não é associada a este domínio [31]. Talvez pela incoerência existente neste tópico, surgem alguns artigos sem qualquer ligação à Intel que associam o consumo do GPU às descrições dos domínios RAPL [108] [114].

- Existem algumas dúvidas na energia reportada sobre o domínio DRAM na interface RAPL. Na documentação da interface do manual de desenvolvimento da Intel não é esclarecedor se, no registo da DRAM, o valor obtido é referente ao consumo da componente DIMM, do controlador de memoria pertencente ao *Package* ou até do *hardware* lógico que transfere os dados entre o *Package* e o *Bus*. São vários os artigos que tentam esclarecer esta questão, no entanto, dividem-se por diferentes opiniões que não permitem que se chegue a alguma conclusão [108] [113] [114] [94] [119]. A Intel apresenta apenas um indício sobre esta questão com uma simples imagem que é possível encontrar numa das suas ferramentas que utiliza a interface RAPL [113]. De maneira a tentar complementar esta informação, foi apresentada esta questão no portal de desenvolvimento da Intel. Em resposta a este assunto, *Patrick Fay* da Intel *Corporation*, esclarece que na interface RAPL, "o domínio de energia DRAM representa apenas as DIMM e não inclui qualquer energia no *Package* ou do controlador de memória" [117]. O colaborador da empresa acrescenta, ainda na mesma resposta, que a "energia DRAM é calculada através de fatores energéticos de várias operações da DRAM". Isto é, "trata-se da soma das multiplicações dos fatores pelas operações que interfiram com o componente" [117].

Na questão referente aos procedimentos de acesso à interface RAPL, em Linux, para consulta e controlo é necessário aceder aos registos MSR do CPU, o que requer suporte do sistema operativo. O Linux não tem atualmente *driver* e não é entendido que tenha no futuro próximo, no entanto, este sistema operativo suporta o *driver* MSR. Sendo estes acessos ao mais baixo nível de um sistema, ou seja, ao nível da arquitetura do CPU, é necessário ter as devidas permissões de leitura para que seja possível aceder aos MSR através do "/dev/cpu/*/msr". A *Application Programming Interface* (API) PAPI utiliza o MSR *driver* para obter os valores de RAPL [108] [116] [95]. Esta solução da ferramenta PAPI verifica-se como uma opção plausível para integrar no projeto a fim de consultar os dados disponibilizados pelo RAPL.

2.4 Ferramentas de Monitorização de Desempenho e Energia

Como referido anteriormente, o processo de monitorização do consumo de um sistema implica a recolha de dados estatísticos que possam dar indícios da atividade do mesmo. Num sistema, os dados podem ser obtidos com base em duas fontes: o

sistema operativo e em componentes de *hardware*. As métricas provenientes do sistema operativo são aquelas que indicam o nível de utilização do sistema, tanto a nível geral como categorizado por componentes e processos. Um exemplo típico é o indicador de utilização do CPU. Estas métricas não são diretamente disponibilizadas pelo *hardware*, mas sim calculadas pelo sistema operativo. As provenientes do *hardware* referem-se, habitualmente, aos *Performance Monitor Counter* (PMC) que são disponibilizados pelo CPU ou à ferramenta *Self-Monitoring, Analysis, and Reporting Technology* (SMART), que está presente nos discos rígidos. Este tipo de métricas é obtido com leituras ao próprio componente que, por sua vez, apresentam informações sobre a atividade do mesmo, tanto a nível instantâneo como em termos de estatísticas temporais. Ferramentas híbridas, caso existam, serão aquelas que procedam à recolha de métricas de ambas as fontes.

2.4.1 Ferramentas de Monitorização ao Nível do Sistema Operativo

As ferramentas de monitorização ao nível do sistema operativo representam um papel essencial na gestão de energia de um computador. Através das mesmas, é possível recolher dados detalhados sobre os vários componentes a que o sistema operativo se encontra a aceder e, de que forma está a utilizar esses recursos para responder aos pedidos do utilizador. Adicionalmente, o sistema operativo dispõe de informações especificas de cada processo em execução, o que permite ter uma perceção dos níveis de utilização de cada aplicação sobre o sistema. Estas informações que são disponibilizadas pelo sistema operativo podem ser maioritariamente recolhidas através do "pseudo" sistema de ficheiros *proc filesystem* (*procfs*). O *procfs* [120] [121] caracteriza-se como um sistema de ficheiros especial devido à sua inexistência na ocupação real de espaço de memória, sendo esse sistema de ficheiros, geralmente montado em */proc*, um simples interface de acesso às estruturas de dados do *kernel* Linux. Grande parte dos dados são apenas de leitura, contudo alguns ficheiros permitem a alteração de algumas variáveis do *kernel*. São também muitas as aplicações que confiam e utilizam este sistema de ficheiros *procfs*, pelo que, se torna uma fonte de dados essencial no quadro de interfaces Linux.

A Figura 9 apresenta o processo de comunicação utilizado através do *procfs*. O seu mecanismo de comunicação funciona através de ficheiros de leitura e escrita presentes na diretoria */proc*, onde é possível encontrar ficheiros de *kernel* e de processos do utilizado [120].

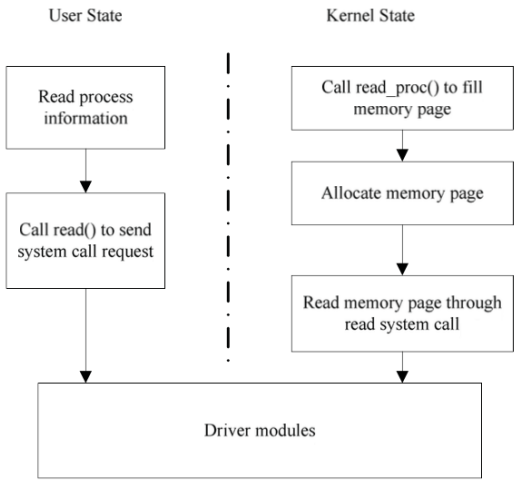


Figura 9 – Processo de Leitura através do procfs [120].

Os ficheiros do *procfs* contêm informação essencial sobre o sistema e sobre o seu estado momentâneo. Pode-se verificar de seguida algumas possíveis informações de obter:

- */proc/cpuinfo* – Informação do CPU (tipo, família);
- */proc/mounts* – Lista de sistemas de ficheiros carregados;
- */proc/diskstats* – Estatísticas IO do disco para cada dispositivo de disco;
- */proc/stat* – Estatísticas do sistema e *kernel* que podem variar com a arquitetura do sistema. Através desta entrada, é possível obter, por exemplo, o rácio de utilização do CPU.

O sistema de ficheiros */proc* contém também um diretório para cada processo em execução no sistema Linux que permite ao utilizador obter informações mais detalhadas sobre cada um deles. Estas diretorias são dinâmicas, sendo que aparecem e desaparecem consoante o estado de execução do processo. O nome dado a cada diretoria referente aos processos em execução é simplesmente o ID do processo. Uma outra diretoria do *procfs* é */proc/sys*, e, nela é possível encontrar todos os ficheiros de leitura e escrita. Esta diretoria difere das outras por, para além de apresentar informação do sistema, permitir ao administrador do sistema ativar ou desativar de imediato funcionalidades do *kernel*.

As ferramentas desta categoria que suscitam mais a atenção para serem aplicadas no desenvolvimento deste projeto são SYSSTAT, TOP, ATOP e VMSTAT.

O **SYSSTAT** [122] é um grupo de ferramentas que utiliza as informações fornecidas pelo sistema operativo para análise total do desempenho do sistema. Deste grupo, as ferramentas SAR, IOSTAT, MPSTAT e PIDSTAT são as ferramentas com mais relevo para o desenvolvimento de um modelo. Com a ferramenta SAR é possível

monitorizar o desempenho de vários subsistemas (CPU, memória, IO, etc.) do sistema operativo Linux em tempo real. Esta ferramenta permite também a recolha de todos os dados de desempenho para armazenar e posteriormente realizar uma analise sobre o historial de dados armazenados. O IOSTAT e o MPSTAT apresentam a mesma finalidade da ferramenta SAR, no entanto, ambos se restringem à estatística de dispositivos de IO e do CPU respetivamente, onde apresentam informação mais detalhada com atributos específicos de cada um desses componentes. Por exemplo, no IOSTAT, é possível verificar o numero total de blocos escritos ou o numero de setores lidos num determinado espaço de tempo. No caso do MPSTAT, podem encontrar-se detalhes como a percentagem de utilização de CPU, que surgiu durante o processamento ao nível do sistema (*kernel*). Por fim, no grupo de ferramentas do SYSSTAT, a PIDSTAT destaca-se por apresentar o nível de utilização do CPU e memória por processo, que é um ponto de partida atraente para desenvolver um modelo a esse nível [122]. Todas as ferramentas referidas dentro do grupo SYSSTAT têm a vantagem de serem de fácil armazenamento em ficheiro, isto porque, acompanhadas de uma coluna de tempo, apresentam um *output* acessível e uma frequência de amostragem definida pelo utilizador. Torna-se também acessível converter estes dados numa tabela de dados para uma posterior análise filtrada.

As ferramentas **TOP** [123] e **ATOP** [124] apresentam-se com um *output* um pouco diferente das ferramentas anteriormente referidas. Identificam-se ambas por exporem um quadro analítico constante dividido em diferentes áreas. Apresentam, num topo superior do quadro, dados generalizados referentes aos vários componentes e informações do sistema e, no espaço inferior, uma tabela ordenada pelos processos com mais atividade no sistema. Nos dados da área superior, estas duas ferramentas diferem, sendo que o TOP apresenta uma informação simplificada sobre processos e *threads* em execução e sobre níveis de utilização do CPU e memória RAM e *swap*. O ATOP distingue-se pelo seu maior detalhe, sendo possível encontrar informação elaborada sobre componentes de rede, discos e CPU ao nível de cada *core*. Na tabela de processos em execução verifica-se um maior detalhe nas informações apresentadas na ferramenta ATOP. Para as necessidades deste projeto qualquer uma destas duas ferramentas é útil para uma análise momentânea sobre a resposta do sistema a avaliações que se proporcionem.

O **VMSTAT** [125] é uma ferramenta que apresenta informações de processos, memória virtual e física, blocos IO, *traps* e atividade do CPU. Esta ferramenta tem a possibilidade de apresentar um *output* com uma determinada frequência de amostragem e apresenta estatísticas sobre o intervalo de tempo entre as amostras. Esta característica é bastante útil no caso de se pretender gravar os dados num ficheiro para consultar futuramente noutra ferramenta de tratamento de dados. Para este

projeto, a ferramenta destaca-se pela possibilidade de monitorizar a atividade de *paging*. Com isso, consegue-se entender algumas movimentações anormais e consumos excessivos de utilização da memória RAM que dão forçam a utilização da memória secundária (disco) como memória virtual (*swap*).

2.4.2 Ferramentas de Monitorização ao Nível de *Hardware*

Sem recorrer a equipamentos de medição externos, as ferramentas de monitorização ao nível de *hardware* são a solução mais direta para se obter dados concretos acerca do desempenho dos vários componentes. Nestas monitorizações ao nível de *hardware*, existe a vantagem de que os valores apresentados são calculados com medições físicas implementadas no fabricante. O fabricante tem a possibilidade de desenvolver tecnologias de medição nos seus componentes e criar uma interface que permita um fácil acesso do utilizador às mesmas. Nesta secção serão descritas facilidades de monitorização a nível de *hardware*, no processador, RAM e armazenamento secundário. As ferramentas que se destacam pela capacidade de aceder às informações fornecidas por esses componentes são a PAPI e a *smartctl*.

Monitorizar o desempenho de um processador e da memória primária utilizando contadores de monitorização de desempenho ou, por outras palavras, PMC, é uma opção a ter em conta para aplicar no modelo a desenvolver. Os contadores de desempenho no *chip* são geralmente precisos e, apresentam uma visão significante no desempenho do processador à frequência do ciclo de relógio do mesmo [126]. Os PMC já vêm incorporados no processador e expostos para o espaço do utilizador nas arquiteturas mais modernas. Os contadores existem como um pequeno conjunto de registos que contam eventos, ocorrências de sinais específicos relacionados com a função do processador [127]. A PAPI [108] é uma famosa API que torna possível o acesso a esses contadores. Para este projeto, a principal finalidade de recolher estes eventos prende-se com a monitorização e modelação de energia. Com a monitorização destes eventos é mais fácil de perceber a correlação entre a estrutura do código fonte e a eficiência de mapear esse código na arquitetura. Assim, surge uma variedade de finalidades na analise de desempenho onde se inclui, por exemplo, a otimização de compilação, *debugging*, *benchmarking*, monitorização e modelação de desempenho. A título de exemplo, os contadores de desempenho do *hardware* podem fornecer informações como instruções por ciclo e comportamento da memória *cache* [128]. Neste projeto, o PAPI será muito útil para obter dados do RAPL que, como referido na secção anterior, modela a energia consumida pelo processador e pela RAM. Para além de ser possível monitorizar o processador, através destes PMC pode

também perceber-se o nível de atividade da RAM através da quantidade de transbordos no ultimo nível de *cache*. Quando se verifica um elevado numero de falhas (MISS) no ultimo nível de *cache*, significa que o processador está a recorrer à RAM para recolher os dados que pretende [48] [129]. A PAPI pode ser uma solução muito útil para o desenvolvimento de um modelo, no entanto, existem outras soluções para explorar os PMC, como por exemplo, o OProfile que é também muito utilizado para o desenvolvimento de modelos [129]. Nas máquinas virtuais, os modelos desenvolvidos com estes parâmetros podem trazer grandes vantagens, isto porque a energia consumida por cada máquina virtual pode ser calculada ao analisar os PMC de cada máquina virtual no modelo de um servidor [129].

Para além do CPU e da memória RAM, o armazenamento secundário é outro componente de grande preocupação para ser monitorizado e incluído no modelo que se pretende desenvolver. Como verificado no subcapítulo 2.2, os discos HDD são atualmente o componente de armazenamento secundário mais utilizado no mundo da informática. Apesar disso existe sempre a possibilidade, mesmo que pequena, de ocorrerem falhas no componente. Se uma falha iminente pudesse ser prevista, podia surgir um aviso para o utilizador do *drive* realizar uma copia de segurança dos dados evitando deste modo perdas de informação. Em 1995, a industria de discos adotou a ferramenta SMART, uma especificação normalizada para prevenir os utilizadores de tais falhas [130]. O SMART é baseado na monitorização de várias medições da tecnologia do *drive* que sejam relevantes para impedir a ocorrência de falhas iminentes [131]. Para isso, é desenvolvido um algoritmo de aviso de falha no *drive* que se baseia no estabelecimento de um limite máximo para cada parâmetro que, assim que é excedido, processa-se ao aviso de erro em binário ("*won't fail*"/"*will fail*"). A ferramenta é de fácil implementação e os parâmetros podem ser monitorizados através das interfaces *computer-to-drive* ATA e SCSI com a utilização de aplicações como o *smartctl* (Linux) [132] [130]. A especificação permite até um total de 30 medições internas do *drive* que são atualizadas periodicamente. Na consulta à ferramenta é possível encontrar parâmetros como o *Read Error Rate*, que apresenta a taxa de erros de leitura do disco ou até a temperatura atual do componente. Certos parâmetros da lista normalizada são apenas utilizados por alguns fabricantes e não são possíveis de consultar nos restantes. Os valores dos atributos são armazenados numa área reservada de dados no *drive* com outros parâmetros operacionais do mesmo. Para se receber o aviso SMART, o sistema necessita de executar comandos específicos da interface do HDD para ativar o algoritmo e depois ler o resultado do aviso entre "*won't-fail*" e "*will fail*" [131] [130] [133].

Além da ferramenta *smartctl*, existem outras ferramentas relacionadas com os componentes de armazenamento secundário, como o *hdparm* que, apesar de ser uma

ferramenta de controlo sobre o mesmo componente, apresenta funcionalidades bastante diferentes e mais indicadas ao modelo que se pretende desenvolver neste projeto.

O *hdparm* é um utilitário de linha de comandos Linux capaz de comunicar com controladores de discos HDD, SSD ou, até mesmo, de outros dispositivos de armazenamento secundário. A ferramenta permite a configuração ou a consulta de vários parâmetros e informações do funcionamento do dispositivo [134]. Através dela é possível, por exemplo, verificar o estado atual de um dispositivo, definir níveis de poupança de energia, definir tempo de espera antes deste adormecer e medir a velocidade de leitura. Primeiramente, a aplicação foi utilizada como uma pequena ferramenta para testar os controladores Linux dos discos HDD. Desde então, o programa foi evoluindo e tornou-se uma ferramenta valiosa para diagnosticar e configurar dispositivos de armazenamento secundário.

O *hdparm* manipula o controlador diretamente, sendo por isso uma aplicação de uso arriscado pois pode levar à perda de dados ou, no pior dos casos, ao defeito do dispositivo. A documentação do programa refere que muitas das suas funções são experimentais ou perigosas. Desta forma, é aconselhável que o utilizador execute apenas funções da aplicação das quais conheça perfeitamente os respetivos efeitos e consequências [135] [77].

Neste projeto, existem funcionalidades do *hdparm* que são mais relevantes do que outras para o processo de testes e monitorização do componente de armazenamento secundário. Destacam-se as seguintes funcionalidades:

- **Controlo do nível de APM:** Configuração que permite definir valores de APM entre 1 e 255. Os valores mais baixos definem uma gestão de energia do disco mais rígida e os valores mais altos definem um melhor desempenho do dispositivo. Com o valor definido a 255, a gestão de energia é desligada [136].
- **Controlo do tempo para ativar o estado *standby* do disco:** Configuração que define o tempo de espera do *drive* para ativar modo *standby*. Por outras palavras, determina-se assim quanto tempo deve o disco esperar em *idle* (sem atividade do disco) antes de desligar o motor para poupar energia [77].
- **Consulta do estado atual do disco:** Funcionalidade que permite verificar o modo do disco no momento. Nesta consulta podem-se obter três diferentes estados: o estado *active/idle* (modo de funcionamento normal), o estado *standby* (modo de funcionamento de baixa potência) e o estado *sleeping* (modo de consumo de energia mais baixo em que o dispositivo está desligado) [77].

- **Controlo da funcionalidade de escrita em *cache* do disco:** Através desta configuração é possível definir se o processo de escrita em disco pode ser previamente feito em *cache* do dispositivo ou se é forçadamente um processo de escrita direta em disco. Sem recorrer à escrita em *cache* o desempenho em disco é inferior, no entanto, o risco de se perderem dados em processo de escrita é muito inferior [136].
- **Controlo da funcionalidade de *read-lookahead* em leituras realizadas em disco:** Através desta configuração é possível definir se o processo de leitura em disco pode recorrer à funcionalidade de *read-lookahead*. Esta funcionalidade determina se o disco, no seu processo de leitura, tem permissão para ler blocos seguintes e prever novos acessos ao mesmo pelo sistema operativo. Está predefinido que esta funcionalidade está ativa para que as consultas de leitura ao componente sejam otimizadas a fim de se conseguir um melhor desempenho de leitura. Este controlo, permite ainda definir o numero de blocos que são lidos para além dos realmente requisitados. No caso desta funcionalidade ser desativada, o disco lê apenas os blocos requisitados pelo sistema operativo [77].

Na fase de ensaios do projeto, as definições estabelecidas nestes parâmetros podem ser vantajosas a fim de tornar o processo o mais puro possível. Estas funcionalidades disponíveis no *hdparm* permitem um maior controlo e conhecimento acerca dos procedimentos de funcionamento estabelecidos pelo subsistema de armazenamento secundário nas escritas, leituras, e nos seus períodos de inatividade.

2.5 Modelos de Estimativa de Consumo de Energia

A energia consumida por um sistema de computação pode ser medida ou estimada. Como referido na secção 1.3, a grande desvantagem de utilizar técnicas de medição é a necessidade de instrumentação com equipamentos de *hardware* externos ao sistema. Além disso, a sua instalação pode tornar-se complexa e dispendiosa, principalmente a nível empresarial. Por outro lado, há a grande vantagem dos valores medidos serem mais precisos e fiáveis, ao serem baseados em cálculos de interceção da alimentação de energia. No caso de se aplicar um modelo de estimativa de consumo de energia, este pode não apresentar os níveis de precisão obtidos em medições físicas, mas, por outro lado, permite uma instalação mais simples e económica e, ao mesmo tempo, uma maior facilidade de escalabilidade. A estimação de energia pode ser realizada com base em métricas de *hardware* e em métricas disponibilizadas pelo sistema operativo. As métricas baseadas em dados recebidos de componentes de

hardware do sistema referem-se habitualmente PMC apresentados pelo CPU ou a detalhes fornecidos por ferramentas como o RAPL [95]. As métricas disponibilizadas pelo Sistema Operativo apresentam estatísticas que permitem calcular níveis de utilização de vários componentes como o CPU, disco ou dispositivos de rede [137]. O sistema operativo tem a possibilidade de fornecer essas informações com base nas operações por ele realizadas. Nos modelos apresentados, encontra-se algumas opções comuns e outras bastante distintas, sendo que, grande parte destes modelos recorrem a ambos os tipos de métricas. Esses modelos sugeridos podem depois ser aplicados a um *software* ou máquina virtual para estimar a sua potência consumida, com base no comportamento do sistema.

Como o modelo a ser desenvolvido integra, na componente de consumo de energia dinâmica, três subsistemas de *hardware* relevantes (CPU, Memória Primária e Armazenamento Secundário), será dada maior atenção aos modelos existentes que estudem esses mesmos subsistemas. Entre os três subsistemas, o subsistema de armazenamento secundário terá uma maior importância neste estudo. Este subsistema dispõe de um numero reduzido de estudos para estimar a energia por si consumida e os detalhes disponibilizados pelos fabricantes impedem que se conheça facilmente o impacto energético do componente. A maioria dos modelos propostos até ao momento destaca o CPU, a memória RAM e o armazenamento secundário como componentes essenciais para desenvolver um modelo rigoroso. Além disso, os modelos em estudo incluem unicamente o disco como componente de consumo de energia IO pelo facto de, a energia consumida pela rede ser relativamente baixa, podendo assim ser ignorada [129].

Para cada um dos três subsistemas definidos, serão estudadas várias abordagens divulgadas em diferentes modelos. Apesar disso, serão destacadas apenas as abordagens que mais se enquadram no trabalho definido a ser realizado neste projeto. Desta forma, para este estudo serão referenciados modelos de estimação de energia dos seguintes autores: Kansal [24], Krishman [138], Bohra [139], Stoess [140] e Dhiman [141]. Krishman apresenta um modelo bastante simples que se foca mais no CPU e na memória primária e avalia-o com uma taxa de erro de estimação que pode atingir os 10%. Bohra também segue ideais de análise idênticos aos de Krishman e, avalia o seu modelo com uma taxa de erro média de 7%. O modelo de Dhiman foca-se bastante nos eventos PMC e é avaliado com uma taxa de erro média de 9%. Os autores Bohra, Kansal e Stoess, comparativamente com os restantes, dão uma maior importância ao impacto do armazenamento secundário no consumo de um sistema. Bohra e Kansal, apresentam abordagens parecidas para os três subsistemas em questão. Bohra apresenta uma taxa de erro média de 7% no seu modelo. Kansal avalia a taxa de erro média do seu modelo de uma forma bastante particular constatando

que esta está entre 1,6 e 4,8 Watts. Por fim, Stoess apresenta um modelo mais elaborado e interessante no subsistema de armazenamento secundário e, no intuito deste projeto, torna-se uma peça fundamental para o seu desenvolvimento. O seu modelo é avaliado com uma taxa de erro média de 10%. Apesar de poderem ser vistos como aplicáveis a aplicações singulares, todos estes modelos têm como finalidade serem adaptados a ambientes virtuais sendo que, os seus autores pretendem atribuir determinados consumos energéticos a cada uma dessas máquinas virtuais.

2.5.1 Processador

O processador é um componente que, num sistema, apresenta um consumo de energia considerável e, em grande parte das situações, este é o subsistema que tem maior impacto energético tanto a nível dinâmico como estático. Por isso, quantificar o consumo de energia pode ser útil para vários propósitos, desde a monitorização de energia e temperatura até ao estabelecimento de politicas dinâmicas de gestão de energia a um baixo nível. O consumo de energia de um CPU depende de bastantes fatores como o tipo de instruções em execução, a *cache* utilizada, frequências de operação e numero de *cores* existentes no componente [48] [32]. Além disso, os estados C, P e T que são estudados na secção 2.3.3, têm um impacto indireto na energia consumida pelo subsistema ao controlarem configurações como a frequência do CPU e os momentos de *idle* do mesmo.

O autor do modelo Joulemeter, Kansal [24], para estimar o consumo energético do CPU por máquina virtual, recorre a um calculo dos tempos em que este componente está ativo e inativo. Estas informações são facilmente obtidas nas métricas do sistema operativo e do Xentrace do Hypervisor e torna-se assim possível de perceber os níveis de utilização do CPU. Para o calculo da utilização do CPU, o autor divide o tempo ativo pelo tempo total do processador durante um intervalo de tempo. O seu modelo destaca-se por apresentar uma alternativa bastante leve para medir o consumo. Dhiman [141], apresenta um modelo que, para estimar a energia do CPU, também recorre aos níveis de utilização do mesmo sugeridos pelo sistema operativo e, a isso, conjunta os eventos de PMC referentes ao numero de instruções por ciclo e às movimentações ocorridas em *cache*. Krishnan [138], na parcela do seu modelo destinada a estimar a energia do CPU, recorre a eventos PMC referentes a instruções executadas e aos acessos à *cache*. O autor considera que controlar apenas o evento *instruction_retired* (*inst_ret*) não é suficiente para estimar o consumo de energia. Este evento contabiliza as instruções obrigatórias a serem executadas para fluir o programa [142]. Apesar disso, o CPU executa muitas outras instruções, mas os resultados registados referem apenas as instruções mais fulcrais, possíveis de ser

consultadas no PMC *instruction_retired*. O autor sugere que, a introdução no modelo, dos acessos e transições entre os diferentes níveis de memória *cache* é importante para perceber o impacto energético de cada instrução realizada pelo CPU. Acrescenta ainda que, as instruções que não apresentem referencias de *cache* ou que apenas atinjam o primeiro nível de *cache* (L1 *Cache*), consomem bastante menos que aquelas que atingem o ultimo nível de cache. Bohra [139], aplica no seu modelo quatro eventos de PMC disponibilizados na sua configuração de teste para estimar a energia consumida pelo CPU. O autor recorre à contagem de ciclos de *clock* de quando o CPU não está pausado ou inativo (através do evento CPU_CLK_UNHALTED), aos acessos à memória *cache* e ao numero de instruções que referencia a memória *cache* na sua execução. O autor utiliza a aplicação Oprofile para monitorizar os PMC e perceber os coeficientes atribuídos a cada métrica estabelecida através destes contadores de desempenho de hardware.

2.5.2 Memória Primária

A memória primária, sendo uma memória de apoio direto ao CPU, tem uma atividade incansável em grandes cargas de utilização de um sistema [48]. Desta forma, é sem duvida um subsistema que tem um grande impacto energético e, por isso, é geralmente incluído nos modelos desenvolvidos para estimativa de energia consumida [24]. Numa fase primária, os autores de vários modelos de consumo de energia concluíram que o fator chave que influencia o consumo energético dinâmico deste subsistema é o débito de leitura e escrita [24]. Uma das soluções mais acessíveis para obter esta informação é com base no contador do evento *Last Level Cache Miss* (LLC Miss). Este evento faz parte dos PMCs que estão presentes em grande parte dos processadores. Desta forma, verifica-se que grande parte dos modelos recorre a esta solução para apresentar a sua formula de cálculo de estimativa de energia. Como verificado no capitulo referente aos componentes de *hardware* relevantes, a memória de um computador esta organizada em hierarquias. Um evento LLC Miss acontece quando uma operação de memória chegou ao ultimo nível de *cache* sem ter sido resolvida e, assim, o CPU acaba por recorrer à memória primária para terminar a operação [48]. Kansal [24] segue esta solução na sua abordagem à estimativa deste subsistema com um método leve. Bohra [139], apesar de também consultar PMCs, recorre a um evento diferente denominado de DRAM_ACESSES por apresentar o seu modelo em torno de um processador AMD. Este evento assemelha-se ao LLC Miss, sendo que, ambos se resumem ao numero de acessos à memória primária. Outros autores como Krishnan [138] e Dhiman [141] introduzem o evento de LLC Miss numa formula mais elaborada ao ser conjugada com as restantes transações de *cache*. Todos

estes modelos desenvolvidos, tiveram a finalidade de serem adaptados ao nível de máquinas virtuais, sendo que, os autores referem a possibilidade da ferramenta Hypervisor fornecer estas informações das PMCs divididas por cada máquina virtual.

2.5.3 Armazenamento Secundário

No desenvolvimento de um modelo de estimativa, o consumo de energia do subsistema de armazenamento secundário é normalmente enquadrado em conjunto com o consumo de outros componentes IO como dispositivos de rede. No entanto, grande parte dos estudos realizados apresenta apenas o disco como componente de consumo energético de IO. Isto acontece porque referem que o consumo de energia de componentes de rede é tão reduzido que pode ser ignorado [24]. A energia consumida pelo armazenamento secundário pode variar consoante o tipo de componente, o tipo de operação ou até mesmo o tipo de sistema de ficheiros nele existente. O subsistema torna-se assim uma maior interrogação no desenvolvimento de um modelo e, desta forma, é também um maior obstáculo para os autores de vários modelos. Num balanço geral sobre os modelos existentes, verifica-se neste subsistema alguma diversidade nas propostas apresentadas. Esta analise, faz crescer a motivação de explorar melhor este tópico e procurar melhores resultados no futuro da monitorização deste subsistema.

Nesta secção, apresentam-se varias abordagens sobre a monitorização do consumo de energia deste componente. O autor Bohra [139] apresenta uma solução baseada na quantidade de dados transitados entre o disco e a aplicação em questão através da utilização da ferramenta de IOSTAT. Esta decisão de utilizar uma ferramenta de monitorização é justificada pelos autores por considerarem a inexistência de PMCs que sugiram diretamente o numero de acessos realizados a disco. Além disso, estes consideram que, pelo numero de PMCs possíveis de ser lidos em simultâneo ser restrito, impede que haja espaço para explorar esta fonte de informação para este componente. O autor Kansal [24] no seu modelo desenvolvido para a ferramenta Joulemeter, utiliza para monitorizar o disco o numero de *bytes* lidos e escritos assim como o tempo serviço ocupado por essas mesmas operações. De qualquer forma, para o cálculo da estimativa ao nível de máquinas virtuais, o autor refere que a ferramenta hypervisor dispõe apenas de monitorização de bytes lidos e escritos. Com isto, Kansal calcula a quantidade de bytes movimentados durante um determinado espaço de tempo. O autor acrescenta ainda que este modelo apenas apresenta valores aproximados, visto que, as ações de movimento dos pratos do disco (*Spin Up/Down*) não são visíveis nem possíveis de ser consultadas no sistema. O impacto energético da variação de velocidades de rotação do disco não é considerado

neste modelo, isto porque o autor considera que esta situação é raramente aplicável a data centers.

Por fim, Stoess [140], numa abordagem de um modelo para máquinas virtuais, evita também a utilização de eventos de PMC neste subsistema e sugere uma solução baseada no tempo necessário para resolver cada pedido de IO. O autor atribui um consumo de energia a cada diferente estado do dispositivo e, além disso, calcula o tempo que o dispositivo necessita para transferir pedidos de um determinado tamanho. Como verificado anteriormente, o numero de estados de energia existentes é definido pelo fabricante, no entanto, o autor resume-se a distinguir apenas dois estados: *active* e *idle*. O autor exclui o estado de suspensão do disco por considerar uma abordagem irrealista para a monitorização do subsistema e, exclui ainda as variações de velocidade do disco. Para calcular o tempo de resolução de um pedido, o autor divide o tamanho do pedido pela taxa de transmissão do disco em *bytes* por segundo em intervalos de 50 pedidos. Stoess acrescenta que, neste modelo, apesar de serem ignorados vários parâmetros que afetam o consumo de energia dos pedidos como o tempo de seek e os atrasos de rotação, esta abordagem simples é suficientemente precisa.

2.6 Sumário

Na conclusão da etapa do levantamento do estado da arte, conseguiu-se obter um esclarecimento sobre o melhor caminho a percorrer para o desenvolvimento do modelo de estimativa do consumo de energia. Primeiramente, verificou-se que os subsistemas CPU, memória primária e armazenamento secundário, são aqueles que habitualmente provocam um maior impacto no consumo de energia dinâmico de um sistema. Em seguida, esses três componentes de hardware foram estudados mais aprofundadamente para se conhecer o modo de funcionamento e a interação que cada um tem com o sistema. Como o processo de desenvolvimento do modelo recai mais sobre o estudo do consumo de energia do armazenamento secundário, foi dada uma maior relevância a este componente e, assim, distinguiu-se as grandes diferenças de funcionamento entre os discos HDD e SSD. Posteriormente, estudou-se as interfaces e ferramentas que se relacionam de alguma maneira com o consumo de energia de um sistema. Entre todas, destacam-se as interfaces ACPI e RAPL, e as ferramentas *hdparm* e *procfs*. A interface ACPI tem a vantagem de permitir um maior controlo sobre o sistema ao nível de gestão de energia. O RAPL é uma solução que apresenta uma estimativa do consumo energético dos subsistemas CPU e memória primária e, por isso, é particularmente relevante para este projeto. A ferramenta *hdparm* alarga o leque de configurações acessíveis no subsistema de armazenamento secundário.

Além disso, esta ferramenta disponibiliza informações como o estado de funcionamento em que o disco se encontra. O *procfs* é uma solução instalada no sistema operativo Linux e permite conhecer uma estimativa dos níveis de utilização dos recursos de *hardware* do sistema. Entre outras funcionalidades, através desta solução é também possível conhecer-se o volume de dados escrito e lido em disco. Por fim, foram analisados alguns modelos existentes a título de se perceber o trabalho já desenvolvido no ramo e as soluções utilizadas pelos autores desses trabalhos científicos.

3. Metodologias de Investigação e Desenvolvimento

Este projeto é desenvolvido para plataformas Linux, onde se pretende estimar o consumo energético através de métricas calculadas com base em dados disponibilizados pelo sistema operativo ou por componentes do sistema. Assim, em seguimento do projeto que envolve os sistemas informáticos, os alvos de monitorização deste projeto são especificamente os componentes que mais impacto têm na atividade dinâmica da máquina.

À primeira vista, um modelo de estimativa do consumo de energia é mais atrativo a grandes superfícies computacionais que prestem serviços de TI do que, mais propriamente, ao cliente particular. As empresas que lideram esta área têm intensificado a investigação em *Green Power* e, a componente de monitorização de energia é uma peça essencial no progresso deste conceito. Neste projeto, o modelo a desenvolver permite vigiar a eficiência energética de um sistema, tanto a nível geral como a nível detalhado por componentes e, no caso do armazenamento secundário, também por processos. Este modelo permite ainda que haja uma melhor perceção sobre os componentes ou aplicações com consumos energéticos mais irregulares e invulgares, prevenindo graves acentuações de encargos energéticos às entidades detentoras destes sistemas.

A metodologia de investigação a utilizar neste projeto é a *action research* [143]. Este tipo de metodologia é baseado no aperfeiçoamento progressivo de um determinado trabalho assim que os problemas vão surgindo. Ou seja, a metodologia parte do planeamento de um trabalho, seguido um ciclo composto pelas fases de ação, observação, reflexão e revisão de planeamento. Esse ciclo, repetido as vezes consideradas necessárias, termina assim que se considera que foram atingidos os objetivos traçados. Desta forma, à medida que um modelo é desenvolvido para monitorização um sistema, ele é provisoriamente posto em prática e, seguidamente, analisam-se os resultados temporários e aplicam-se as ações necessárias para aperfeiçoar o modelo até serem atingidas as metas inicialmente traçadas.

O processo de desenvolvimento do modelo é feito sobre um sistema Linux com distribuição Lubuntu 14. O Linux destaca-se por apresentar um *kernel* muito utilizado na industria de TI, principalmente, em *data centers*. As aplicações para exercitar os diferentes subsistemas, calcular e recolher métricas são desenvolvidas em linguagem C. A linguagem C é uma referência por ser a linguagem base do Linux e torna-se assim uma vantagem para exercitar componentes ou manipular mais facilmente o sistema operativo.

No modelo a desenvolver será utilizada a interface RAPL para obter dados de consumo de energia dos componentes CPU e Memória Primária. Desta forma, o CPU escolhido para a configuração de *hardware* tem de se fazer acompanhar da funcionalidade. Para além disso, ao ser dada especial atenção ao armazenamento secundário, serão estudados quatro discos diferentes que abranjam os dois tipos de discos: HDD e SSD.

Este capitulo apresenta as metodologias de investigação e desenvolvimento a ser aplicadas para o desenvolvimento deste projeto. Nesta etapa serão destacados os métodos para o desenvolvimento do modelo e em que condições é que este será desenvolvido. Na primeira parte deste capitulo serão explicados os procedimentos definidos para desenvolver o modelo. Em seguida, serão esclarecidas as condições de *hardware* e *software* em que o modelo irá ser desenvolvido. Depois, serão estudados os indicadores mais úteis para recolher do sistema, as técnicas de recolha e a fiabilidade dos mesmos. Por fim, estabelecer-se-ão as condições da execução dos ensaios ao sistema para o desenvolvimento do modelo.

3.1 Procedimentos de Desenvolvimento

Num formato mais generalizado e, tal como explicado na secção 2.1.2, é essencial compreender que o consumo de energia do sistema se divide em duas grandes porções, o consumo de energia estático e dinâmico. O consumo de energia estático representa o consumo de todos os recursos necessários para o sistema se manter ligado e em serviços mínimos de funcionamento sem qualquer atividade requisitada pelo utilizador. O consumo de energia dinâmico representa o consumo de energia provocado por todas as atividades acionadas pelo utilizador, isto é, é o *offset* de energia que ultrapassa o consumo estático do sistema. Esta divisão de consumo energético explica-se pela equação,

$$P_{total} = P_{estática} + P_{dinâmica}, \tag{5}$$

em que P_{total} representa a potência total consumida pelo sistema e, $P_{estática}$ e $P_{dinâmica}$, representam a potência estática e dinâmica, respetivamente.

Para obter o **consumo de energia estático**, é lido, durante um intervalo de tempo, o consumo de energia estimado pela tecnologia ACPI *Smart Battery*. Durante esse intervalo o sistema mantém-se ligado com a menor atividade possível e sem qualquer interferência do utilizador. Recorrendo-se à ferramenta RAPL, é ainda possível separar o consumo $P_{estática}$ em três diferentes elementos, sendo possível obter a seguinte equação,

$$P_{estática} = P_{idle\_OTHER} + \left(P_{idle\_CPU} + P_{idle\_DRAM}\right), \quad (6)$$

em que P_{idle_CPU} representa o consumo estático do CPU, P_{idle_DRAM} representa o consumo estático da memória primária e, por fim, P_{idle_OTHER} representa a restante potência consumida estaticamente.

O **consumo de energia dinâmico** é explorado numa fase posterior ao calculo do consumo estático, a fim de se conseguir distinguir dessa energia consumida, os impactos energéticos provocados pelo utilizador. Nesta fase, qualquer aumento no consumo de energia total do sistema faz parte do consumo de energia dinâmico do sistema. A importância desta fase resume-se à repartição desse consumo dinâmico pelos três subsistemas principais que foram selecionados para este projeto: o processador, a memória primária e o armazenamento secundário.

Novamente, a ferramenta RAPL permite-nos realizar esta tarefa e compreender que impacto está a ter a atividade provocada pelo utilizador em cada um dos três principais subsistemas. Esta separação de elementos pode ser compreendida através da seguinte equação,

$$P_{dinâmica} = P_{active\_CPU} + P_{active\_DRAM} + P_{active\_DISK}, \quad (7)$$

em que P_{active_CPU} representa a potencia consumida pelo processador, P_{active_DRAM} representa a potência consumida pela memória primária e P_{active_DISK} a potência consumida pelo armazenamento secundário. Estas três variáveis apresentam apenas a potência consumida pela componente ativa do sistema, ou seja, a energia consumida pelos três componentes que ultrapassa os seus consumos estáticos. Para se obter as variáveis P_{active_CPU} e P_{active_DRAM}, recorre-se à diferença entre o consumo instantâneo atual e o consumo estático de ambos os componentes. Desta forma, é possível obter o consumo dinâmico dos mesmos através das equações,

$$P_{active\_CPU} = P_{now\_CPU} - P_{idle\_CPU}, \quad (8)$$

$$P_{active\_DRAM} = P_{now\_DRAM} - P_{idle\_DRAM}, \quad (9)$$

sendo P_{now_CPU} e P_{now_DRAM}, as variáveis de consumo atual dos subsistemas que a ferramenta RAPL disponibiliza.

Como o subsistema de armazenamento secundário não disponibiliza informações de consumo energético, é necessário realizar um calculo com base nos indicadores das tecnologias ACPI Smart Battery e RAPL para se obter o valor da variável P_{active_DISK}. Logo, recorre-se à seguinte equação,

$$P_{active\_DISK} = P_{now\_TOTAL} - P_{idle\_OTHER} - P_{now\_CPU} - P_{now\_DRAM}, \quad (10)$$

em que P_{now_TOTAL} corresponde ao valor do consumo energético momentâneo total do sistema, disponibilizado pela tecnologia ACPI Smart Battery. As variáveis P_{now_CPU} e P_{now_DRAM}, representam os consumos momentâneos de energia apresentados pela ferramenta RAPL referentes aos subsistemas processador e memória primária.

Numa fase posterior, já sendo possível extrair o consumo energético da atividade dinâmica do disco, inicia-se a etapa de ensaios a este subsistema. Nesta etapa, são realizados diversos exercícios que provocam diferentes procedimentos e operações no subsistema. Juntamente com esses exercícios, são estudados os diferentes impactos energéticos provocados pelo subsistema de armazenamento secundário ao se examinar os valores obtidos na variável P_{active_DISK}. Além disso, durante os exercícios realizados, são recolhidos certos indicadores do subsistema e é estudada a relação entre esses valores e as variações existentes no consumo energético do armazenamento secundário. Assim, é possível atribuir diferentes pesos de consumo de energia a cada operação e, perceber o impacto energético em disco, que certos indicadores recolhidos induzem com as suas variações.

Através destas análises realizadas nesta fase, é possível desenvolver-se um modelo que estime a P_{active_DISK} sem ser necessário recorrer à variável P_{now_TOTAL} que sugere o consumo total do sistema. Como referido anteriormente, o modelo final recorre apenas à ferramenta RAPL que permite consultar o consumo energético dos subsistemas do processador e memória primária. A estimativa da potencia consumida do subsistema de armazenamento secundário, na totalidade e categorizado por processo, é também incluída no modelo e baseia-se apenas nos indicadores recolhidos e nas operações executadas pelo sistema operativo. Desta forma, o modelo final deixa de parte o recurso da tecnologia ACPI Smart Battery que disponibiliza o consumo energético total do sistema.

3.2 Configuração de *Hardware*

A fase de configuração de *hardware* tem um papel fundamental para o desenvolvimento de um modelo. Esta fase deve ser processada sobre uma análise do atual mercado dos subsistemas mais relevantes e sobre os objetivos presentes neste projeto. Relembre-se que o principal foco deste projeto incide sobre o desenvolvimento de um modelo que estima o consumo de energia do sistema, distribuído pelos três subsistemas destacados e, no caso do disco, também por processo. Pretende-se que o modelo contribua para uma melhor monitorização dos sistemas existentes e que, consequentemente, seja um apoio para que se consiga

melhores resultados de eficiência energética. Atualmente, a principal fonte dos elevados consumos de energia na informática são os *data centers* onde existe uma grande quantidade de máquinas ativas em execução. No entanto, também os dispositivos móveis, como portáteis ou telemóveis, procuram uma maior eficiência energética para atingir melhores resultados na autonomia da bateria. É desta forma importante que se apresente uma configuração enquadrada na atualidade tecnológica para que, assim, seja possível desenvolver-se um modelo mais moderno e de fácil adaptação a qualquer sistema.

3.2.1 Processador

Sendo o CPU o principal componente para o bom desempenho de um sistema e também aquele que mais impacto energético tem, é fundamental ser o ponto de partida para se encontrar uma configuração de *hardware* mais adequada. Como estudado anteriormente, o fabricante que domina maioritariamente o mercado dos processadores é a Intel que, apesar de ainda não se ter assumido no ramo dos *smartphones* e *tablets*, tem uma quota enorme nos servidores e computadores desktop e portáteis [39] [52]. Deste modo, a utilização de um CPU Intel na configuração de Hardware é vantajosa para se atingir os objetivos previamente definidos neste projeto. Como é obvio, a Intel apresenta uma vasta gama de modelos para diferentes finalidades e diferentes tipos de sistemas. Sendo o RAPL uma interface que tem vindo a acompanhar todos os processadores do fabricante desde o aparecimento da geração *Sandy Bridge* em 2011, é importante que esta funcionalidade esteja presente no processador escolhido [109]. Atualmente, três em cada quatro sistemas de *data centers* e Super Computadores são compostos por processadores Intel Xeon com microarquitectura *Sandy Bridge* ou outra sucessora e, claro, todos eles têm disponível a ferramenta RAPL [137] [52]. É uma tecnologia bastante recente que se tem vindo a assumir e clarificar ao longo destes anos. Sendo assim, prevê-se que a interface tenha um papel cada vez mais fundamental na gestão energética de *data centers* e de computadores pessoais. Para este projeto será utilizado um processador Intel i5-4200M com 2.5 GHz e uma arquitetura *Haswell* (características representadas na Tabela 1). Esta microarquitectura, para além de apresentar uma melhoria significante de desempenho em relação a arquiteturas anteriores, consome vinte vezes menos do que a arquitetura anterior *Sandy Bridge* e duas vezes menos do que a arquitetura *Ivy Bridge* [138]. O processador utilizado é bastante atual, com apenas dois anos de vida no mercado e com um baixo consumo, visto que, é um processador de tipo *Mobile* [146]. Este tipo de processadores é desenvolvido para sistemas de baixo consumo e com alto desempenho como é o caso de portáteis e tablets.

Tabela 1 – Características do processador a utilizar [146].

Processador	
Fabricante	Intel
Modelo	i5-4200M
Microarquitectura	*Haswell*
Nº de Cores	2
Nº de *Threads*	4
Frequência Base	2.5 GHz
Frequência Máxima (Turbo)	3.1 GHz
Tipo de Instruções	64-Bit

3.2.2 Memória Primária

A memória primária introduzida na configuração de *hardware* necessita de ser compatível com o processador. Neste caso, a Intel indica que o processador não é compatível com um tamanho de memória primária superior a 32GB, e requer a utilização do tipo de memória DDR3L com uma frequência de 1333 MHz ou 1600 MHz [139]. A memória RAM escolhida é composta por dois módulos de 4GB com interface DDR3L e, com uma frequência de 1600 MHz. Como referido na secção 2.2.2, este tipo de memória é muito utilizado em computadores portáteis e servidores. É um tipo de memória que opera em duas diferentes voltagens consoante as necessidades do sistema. Desta forma, o DDR3L consegue apresentar excelentes resultados num balanço de relação entre o consumo de energia e desempenho. Sendo o sistema um computador portátil, são utilizadas *Small Outline Dual In-line Memory Modules* (SO-DIMM), que se caracterizam por serem mais pequenas. Este tipo de DIMMs é bastante escolhido pelos fabricantes destes dispositivos devido ao seu espaço limitado para desenhar a distribuição de componentes [152]. A Tabela 2 resume os dispositivos de memória primária utilizados na configuração do sistema.

Tabela 2 - Características dos dispositivos de memória primária a utilizar [148].

Memória primária				
Quantidade	**Fabricante**	**Frequência**	**Tamanho**	**Referência**
1	Hynix	1600 MHz	4GB	hmt451s6bfr8a-pb
1	Ramaxel	1600 MHz	4GB	rmt3170ME68F9F1600

3.2.3 Armazenamento Secundário

Os discos utilizados para este projeto foram escolhidos com a atenção de representarem um balanço evolutivo da tecnologia do subsistema e as características dos discos. Para esta seleção, foram tomadas em conta algumas especificações sobre

estes produtos, sendo as mais relevantes, o ano de lançamento, capacidade, e rotações por minuto, no caso dos HDD. O conjunto de componentes de armazenamento secundário escolhidos estão referidos na Tabela 3 [153] [81] [154]. Estes 2 componentes figuram a evolução tecnológica entre as duas tecnologias mais reconhecidas deste subsistema e têm várias caraterísticas interessantes para justificar as possíveis diferenças de consumo que se irão encontrar na análise sobre os testes efetuados. O componente da fabricante Seagate é do tipo HDD, o tipo de armazenamento secundário mais utilizado atualmente. No entanto, com o tipo de discos SSD a crescer no mercado de hoje, é também importante que se inclua esse tipo de disco no desenvolvimento do modelo de consumo de energia de um sistema. Por isso, procurou-se reproduzir, dentro dos possíveis, no disco SSD, as caraterísticas do HDD e assim fazer uma comparação equilibrada da tecnologia. Estes dois discos são bastante atuais e separam-se, na data de lançamento, pelo insignificativo intervalo de um ano. A acrescentar, utilizam ambos SATA 3 e apresentam a mesma capacidade de armazenamento de 500GB.

Tabela 3 - Especificações dos discos utilizados [78] [148] [149].

Armazenamento secundário		
Fabricante	SEAGATE	Samsung
Ano Lançamento	2013	2014
Modelo	*Momentus Thin*	850 EVO
Referência	ST500LM021	MZ-75E500B/AM
Interface	SATA 3	SATA 3
Tipo	HDD	SSD
Capacidade	500 GB	500 GB
RPM	7200	-
Pratos	1	-
Cabeças	2	-
Blocos Lógicos	512	512
Blocos Físicos	4096	-

Ambos os discos estão definidos para funcionarem em blocos lógicos de 512 *bytes*, sendo que, fisicamente, trabalham com tamanhos de dados diferentes. Os blocos lógicos são a unidade de alocação mais pequena em que um ficheiro está dividido num sistema de ficheiros [153]. O tamanho de um bloco lógico varia consoante o sistema de ficheiros e, cada bloco lógico, está mapeado para um único bloco físico. No caso dos blocos físicos, o disco HDD funciona em blocos físicos de 4096 *bytes*. No caso do disco SSD, o fabricante não disponibiliza essas informações, visto que, esses detalhes são menos relevantes por se tratar de uma memória *flash*.

Além destas especificações de fabrico, como é possível verificar-se na Tabela 4, os fabricantes analisaram e estimaram também o consumo de energia médio de cada

disco. Logicamente, nos discos HDD verifica-se um maior consumo em *idle*, visto que, o motor de rotação do prato se mantém ativo. Em períodos de atividade, o consumo do disco SSD atinge valores superiores, no entanto, é muito provável que se verifique uma maior eficiência do mesmo por este resolver mais rapidamente as operações requisitadas.

Tabela 4 – Estimativas de consumo de energia pelo fabricante do HDD [150] e SSD [149] [78].

Disco HDD	
Estado	**Média de Consumo**
Standby	0,18 W
Idle	0,9 W
Active - Read	1,70 W
Active – Write	1,90 W
Active - Seek	1,80 W

Disco SSD	
Estado	**Média de Consumo**
Idle	0,05 W
Active	3 W

Os valores apresentados pelos fabricantes serão analisados e confrontados depois dos ensaios realizados a cada um dos discos. Possivelmente, estas estimativas serão diferentes com aquelas que se irão encontrar ao longo deste projeto. Isto deve-se a vários fatores como as diferenças de temperatura ambiente e à diferente configuração estabelecida nos ensaios.

3.3 Configuração de *Software*

Para este projeto, optou-se por um Sistema Operativo com *kernel* Linux por várias razões, entre elas, a sua grande flexibilidade e transparência. Um Sistema Operativo Linux tem a grande vantagem de permitir um fácil e transparente acesso às bibliotecas do sistema, à sua infraestrutura e aos *drivers*. Nos sistemas de grande família UNIX, é possível ter acesso completo ao conteúdo do coração do sistema. Principalmente no desenvolvimento de software, estas caraterísticas são muito positivas, ao permitirem ao programador uma maior habilidade no desenvolvimento ou teste do *software*. Este sistema operativo permite uma customização profunda e um maior controlo no *hardware*, sendo estas as maiores razões para a grande proliferação do Linux.

O Linux tornou-se igualmente um dos elementos que acompanhou a modernização de um *data center* [141]. DiBona, diretor da área de *Open Source* da Google, indica que "todos os servidores da Google estão em funcionamento sobre o Sistema Operativo Linux" [142]. Além disso, os dois Sistemas Operativos da Google, Android e Chrome OS, são desenvolvidos com o núcleo Linux [142] [143]. Atualmente, o Sistema Operativo Android é o mais popular do mundo, não só na

categoria móvel como também na categoria global de sistemas operativos [144]. No ramo dos Super Computadores, também existe um total domínio dos Sistemas Operativos Linux [145] [137]. Neste último ramo, que se destaca pela elevada capacidade de computação e excelente desempenho, verifica-se que 99% destas máquinas funcionam com Sistema Operativo GNU/Linux [146]. Verifica-se assim que, em várias vertentes, o *kernel* Linux tem tido um papel fulcral no processo evolutivo da tecnologia destes últimos tempos, desde as grandes centrais de dados até ao pessoal *smartphone*.

Apesar desta conclusão, optar por um *kernel* Linux não é suficiente. O *kernel* Linux pode ser entendido como o coração do sistema operativo, no entanto, é necessário escolher uma distribuição. Uma distribuição Linux é composta por elementos como coleções de aplicativos, utilitários da Shell GNU (terminal), interface gráfica e, claro, o *kernel* (núcleo) do sistema operativo. Neste projeto, pretende-se implementar uma distribuição limpa, transparente, bastante leve e com poucos processos a executar nos bastidores do "olhar" e intenção do utilizador. Resumidamente, procura-se uma distribuição que interfira o mínimo possível com os ensaios a serem realizados. Desta forma, com uma análise sobre recentes estudos, optou-se pela distribuição Lubuntu 14 para desenvolver este projeto. A *techradar* avalia o Lubuntu como a distribuição Linux mais polarizadora nas variantes das soluções mais simples e leves [147]. O portal reporta ainda que esta distribuição é ideal para aqueles que procuram um Sistema Operativo limpo e que necessite de poucos recursos. Outra análise descreve o Lubuntu 14 como o melhor em termos de desempenho [148]. Nessa analise recomendam esta distribuição a quem procura "uma distribuição realmente eficiente e destinada a sistemas de baixos consumos". Estas duas análises foram bastante conclusivas e adaptam-se perfeitamente ao que se procura neste projeto. Os detalhes sobre a configuração de *software* definida para este projeto podem ser encontrados na Tabela 5.

Tabela 5 – Especificações da distribuição do sistema operativo a utilizar.

Distribuição do Sistema Operativo	
ID do Distribuidor	Lubuntu (Ubuntu)
Descrição	Ubuntu 14.04.4 LTS
Versão	14.04
Codename	trusty
Kernel	3.19.0-46-generic

Estas são algumas das motivações desta escolha, que permite que este projeto seja facilmente migrado para um fim industrial. Como se verificou anteriormente, é

um sistema operativo que se pode resumir como muito versátil e utilizado tanto a nível de grandes plataformas de computação e de dados como para um simples dispositivo móvel que faz parte do quotidiano de qualquer pessoa. Para além disso, é um sistema operativo limpo e com pouca atividade a não ser a provocada pelo utilizador.

3.4 Indicadores de Desempenho e Energia

Pode-se entender por indicadores de desempenho e energia todas as métricas e outras variáveis que descrevam ou indiciem um estado, exercício executado ou energia utilizada pelo sistema. No capítulo anterior, foram anunciadas algumas ferramentas que se entendem de fulcrais para obter indicadores de desempenho de um sistema. Recorde-se que estes indicadores podem surgir por matemáticas aplicadas pelo sistema operativo ou por informações retiradas diretamente do *hardware* do sistema. Para construir um modelo de estimativa de energia é necessário primeiramente selecionar indicadores que se associem, direta ou indiretamente, aos componentes que se pretende monitorizar. Os indicadores determinados são seguidamente implementados num modelo que define os seus pesos e quantifica os seus impactos energéticos num sistema, tanto ao nível de subsistemas como de processos.

Para este projeto, existe uma vasta quantidade de indicadores de desempenho disponíveis e é preciso realizar uma triagem para tornar o desenvolvimento do modelo mais simples e prático. Desta forma, foram estudadas várias ferramentas e os campos de dados que estas apresentam sobre os dispositivos que se pretende analisar. Nesta secção, apresentam-se os indicadores que serão utilizados direta e indiretamente no modelo. No seguimento da estratégia anteriormente definida, entende-se que o subsistema de armazenamento secundário deve ser monitorizado pormenorizadamente, isto porque, é o subsistema com mais escassez de informação energética e transparência de funcionamento [24]. Os restantes dois subsistemas serão monitorizados com base em informações energéticas já disponibilizadas por ferramentas estudadas e descritas no capítulo anterior. Assim, e como se verificou anteriormente, pode-se considerar que o modelo será trabalhado a fim de apresentar estimativas energéticas ao nível de quatro subsistemas de *hardware*: o consumo energético total do sistema e o consumo energético repartido pelos três principais subsistemas. A nível de *software*, será possível estimar o consumo de energia no subsistema de armazenamento secundário repartido por processos consoante os seus níveis de utilização do disco.

Sendo o modelo desenvolvido com base na técnica *Action Research*, este precisa de ser aperfeiçoado consoante as necessidades e objetivos. Posto isso e, para que o modelo seja fiável, este deve ser progressivamente comparado com o consumo de energia total de um sistema apresentado por fontes ou ferramentas externas a este modelo. Desta forma, este indicador terá um papel essencial na construção do modelo:

- **Potência Consumida pelo Sistema:** Energia total utilizada pelo sistema por segundo (*watts*). Este valor inclui toda a energia estática e dinâmica consumida.

Apesar de este indicador não ser incluído diretamente no modelo, ele participa no processo de desenvolvimento, aperfeiçoamento e, por fim, no processo de validação do modelo. Numa fase final, através desta informação, será possível comprovar que o modelo está a apresentar uma estimativa plausível e próxima do consumo real do sistema. O indicador contribui também para apresentar o rácio de precisão do modelo.

Ao contrario do indicador referente ao consumo total de energia do sistema, os indicadores que são apresentados em seguida, serão incluídos no modelo de estimativa de consumo de energia. Como referido anteriormente, o modelo final apresentará um submodelo no subsistema do armazenamento secundário. Este submodelo pode por sua vez apresentar o consumo energético total do subsistema ou o consumo energético ramificado por processos.

Nos subsistemas do **processador** e da **memória primária**, incluem-se dois parâmetros que expõem os seus consumos de energia. Depois de um estudo aprofundado sobre as interfaces e soluções de monitorização de energia existentes nos sistemas e, mantendo a ideia inicialmente traçada, verifica-se que, os subsistemas, processador e memória primária, já se fazem acompanhar de modelos integrados no seu *hardware* que se mostram capazes de estimar a energia consumida. Por esta razão, existem dois indicadores que podem ser calculados através de informação disponibilizada pelos subsistemas, nomeadamente:

- **Potência Consumida pelo Processador:** energia total utilizada pelo processador por segundo (*watts*);
- **Potência Consumida pela Memória Primária:** energia total utilizada pela memória primária por segundo (*watts*).

Estes dois indicadores, juntamente com o indicador da Potência Consumida pelo Sistema, são muito vantajosos para explorar pormenorizadamente o impacto energético do armazenamento secundário num sistema. Há muitas questões possíveis de ser estudadas através destes dados, desde o consumo estático de energia do sistema

e destes dois subsistemas, até ao impacto energético do restante subsistema consoante os seus níveis de utilização. Além disso, o modelo fica parcialmente construído com dois subsistemas com dados energéticos clarificados.

O **armazenamento secundário** é um subsistema que se mostra atualmente uma verdadeira incógnita a nível energético. Apesar disso, com base nos parâmetros anteriormente anunciados, o calculo do impacto energético do armazenamento secundário torna-se mais claro e percetível. Como este componente não apresenta dados de energia concretos, é necessária a recolha de parâmetros de desempenho capazes de caracterizar o funcionamento do disco e distinguir os seus diferentes estados de operação. Existem dois grandes objetivos da estimativa de energia consumida por este componente num sistema. Primeiramente, pretende-se apresentar o consumo de energia total do subsistema e, em seguida, particularizado por processo em execução no Sistema Operativo.

Neste subsistema, foram estudados dois diferentes caminhos para atingir os objetivos inicialmente estipulados, um caminho com indicadores apresentados pelo Sistema Operativo e por ferramentas que este disponibiliza e, outro caminho, em que há um processo de manipulação e supervisão sobre as operações realizadas pelo Sistema Operativo. No caminho mais generalizado e baseado em soluções disponibilizadas pelo sistema operativo, foram recolhidos os seguintes parâmetros de desempenho:

- **Numero de pedidos de leitura anunciados:** quantidade de operações de leitura requisitadas pelo sistema operativo ao subsistema de armazenamento secundário;
- **Numero de pedidos de escrita anunciados:** quantidade de operações de escrita requisitadas pelo sistema operativo ao subsistema de armazenamento secundário;
- **Numero de *kilobytes* escritos por segundo:** taxa de transmissão de escrita para o subsistema de armazenamento secundário por segundo (kB/s);
- **Numero de *kilobytes* lidos por segundo:** taxa de transmissão de leitura para o subsistema de armazenamento secundário por segundo (kB/s);
- **Tamanho médio de um pedido:** tamanho médio, em setores, de cada pedido realizado ao disco;
- **Tempo médio de resposta a um pedido:** tempo médio (em milissegundos) para que, os pedidos de IO enviados, sejam resolvidos pelo disco. Isto inclui o tempo utilizado pelos pedidos na fila de espera e o tempo decorrido para resolver os pedidos;

- **Estado de operação do disco:** estado que informa sobre modo atual de operação do disco a fim de gerir a energia. Caso o componente disponibilize esta informação, esta varia entre: *active/idle* (modo de operação normal), *standby* (modo de baixo consumo, com os HDD a desativarem as rotações) ou *sleeping* (modo de consumo mais baixo em que o disco é completamente desligado).

Os indicadores até ao momento apresentados foram desenvolvidos com base em soluções existentes. No sentido de se explorar soluções mais esclarecedoras para este subsistema, optou-se por investigar outro caminho mais intrometido no funcionamento do Sistema Operativo. Este caminho implica a implementação de novas soluções capazes de calcular e apresentar novos indicadores de desempenho. Esta necessidade deve-se ao facto de as soluções estudadas não apresentarem parâmetros categorizados por processo e, além disso, não detalharem os acessos realizados ao armazenamento secundário. Desta forma, este segundo caminho permite que a estimativa da energia consumida possa ser calculada com base nas operações realizadas ao subsistema, por processo e, além disso, com base na quantidade e tipo de acessos que estas fazem ao componente. O processo de acesso a um dispositivo de armazenamento secundário é comandado pelo Sistema Operativo à medida das suas necessidades através de *System Calls* (chamadas do sistema). Posto isto, serão utilizados, como parâmetros, as operações realizadas por determinadas *System Calls* que interfiram, direta e indiretamente, com o disco [149]. Neste caso, as *System Calls* de maior interesse para monitorizar os acessos a disco são:

- ***write*:** escrever num descritor de ficheiro;
- ***lseek*:** reposicionar o ponto para escrita ou leitura no ficheiro;
- ***read*:** ler um descritor de ficheiro.

Estas *System Calls* têm impactos energéticos variados consoante o seu objetivo e a quantidade de dados que movimentam no seu processo de IO. Apresentam-se assim os parâmetros que são utilizados nesta vertente do modelo:

- **Número de *System Calls "write()"*:** numero de *System Calls* executadas pelo Sistema Operativo por segundo;
- **Tamanho do Bloco da *System Call "write()"*:** tamanho dos blocos escritos através das *System Calls*;
- **Número de *System Calls "read()"*:** numero de *System Calls* executadas pelo Sistema Operativo por segundo;
- **Tamanho do Bloco da *System Call "read()"*:** tamanho dos blocos lidos através das *System Calls*;

- **Número de *System Calls* "*lseek*()":** numero de *System Calls* executadas pelo Sistema Operativo por segundo.

Com base neste conjunto de indicadores surge a possibilidade de caracterizar o desempenho e estimar o consumo energético de um sistema e dos seus principais subcomponentes. No desenvolvimento do modelo, estes indicadores apresentam valores que são divididos em dois estados fulcrais para se distinguir o impacto energético dos diferentes subsistemas, o consumo de energia estático e dinâmico. A distinção entre os valores dos parâmetros apresentados nestes diferentes estados, permite que se deduza os níveis de utilização, desempenho e energia consumida do sistema e seus subsistemas. Este estudo é feito com base na resposta dos subsistemas aos exercícios a que estes são submetidos. Ao ser conhecido o consumo total de energia consumida pelo sistema, processador e memória primária, é possível também entender o impacto energético dos exercícios característicos que são realizados ao subsistema de armazenamento secundário.

3.5 Técnicas de Recolha de Indicadores de Desempenho e Energia

Após serem revelados os indicadores chave para a construção de um modelo, é inevitável encontrar soluções através de práticas e ferramentas capazes de recolher esses indicadores. Enquanto uma parte destes indicadores é disponibilizado por ferramentas já desenvolvidas, a outra parte implica a implementação de novas técnicas e o desenvolvimento de ferramentas capazes de calcular estas métricas. Entretanto mesmo no caso da recolha de indicadores já disponibilizados, a solução requer um trabalho prático ao nível de desenvolvimento de *software* para que seja possível a colheita dos dados das ferramentas já existentes.

3.5.1 Técnicas com Base em Soluções Existentes

A recolha de indicadores de desempenho e energia através de soluções existentes tem a caraterística particular de apresentar mais funcionalidades do que aquelas realmente procuradas para o desenvolvimento de um modelo. Por essa razão, estas implicam o desenvolvimento de aplicações para atuar apenas sobre os dados pretendidos para o projeto. Por outro lado, têm a vantagem de já incorporarem modelos sólidos que garantem a fiabilidade dos valores que apresentam. As soluções existentes podem-se dividir em dois tipos diferentes, ***front-end*** e ***back-end***. Enquanto as soluções *front-end* são de fácil consulta e preparadas para o acesso direto do utilizador final, as soluções *back-end* necessitam da intervenção de um

programador para tirar proveito sobre os dados disponibilizados. Ao contrario das soluções *front-end*, as soluções *back-end* têm a vantagem de ser mais personalizáveis ao interesse do utilizador. Desta forma, nestas soluções é necessário o desenvolvimento de uma aplicação para que haja um controlo e monitorização constante sobre os indicadores. O resumo das soluções existentes utilizadas está explicito na Tabela 6. A aplicação *mywatter*, que foi desenvolvida para este projeto, reúne todos os indicadores back-end e, parte dos indicadores de front-end, numa única aplicação de recolha de indicadores.

Tabela 6 – Soluções utilizadas para recolha de indicadores de desempenho e energia de um sistema.

Tipo	Soluções	Tecnologias e Indicadores
Front-end	IOSTAT	kb/s escritos, kb/s lidos, pedidos de IO, etc.
	VMSTAT	Numero de blocos IO (disco), memória RAM utilizada, etc.
	HDPARM	Estado de operação do disco, etc.
Back-end	PAPI	PCM, RAPL, energia consumida pelo CPU e RAM, etc.
	procfs	Atividade de processos, nível de utilização do Disco, CPU, etc.
	sysfs	ACPI Smart Battery, potência consumida pelo sistema, etc.

As **soluções de *front-end*** utilizadas, exibem *outputs* previamente estabelecidos e apresentam mais informações do que aquelas que são realmente necessárias para este projeto. Desta forma, são desenvolvidas aplicações ou técnicas para filtrar a informação ao ponto de se recolher apenas os indicadores anteriormente definidos. As soluções de *front-end* incluídas para a recolha de alguns indicadores são o **IOSTAT**, o **VMSTAT** e o **HDPARM**. O IOSTAT e VMSTAT, são ferramentas que apresentam os seus próprios *output*s, no entanto, é possível predefinir alguns parâmetros que filtram parte da informação desnecessária para o projeto. No caso do IOSTAT [121], são utilizados os parâmetros "-x" e "-d" que permitem obter apenas dados detalhados referentes ao disco, excluindo-se assim dados dispensáveis sobre outros subsistemas com o processador. No VMSTAT [124] a situação é idêntica, sendo que, nesta ferramenta não são incluídos quaisquer parâmetros de predefinição e a triagem de dados será toda realizada por uma aplicação desenvolvida para este projeto. Estas duas ferramentas permitem recolher os indicadores relativos ao subsistema do armazenamento secundário que são calculados com base em dados fornecidos pelo Sistema Operativo, como a taxa de transmissão e o numero de pedidos de leitura e escrita no subsistema. Além destas duas ferramentas, o HDPARM [134] [77] tem um papel essencial neste projeto, isto porque, ele é utilizado para recolher o indicador do estado de operação atual do componente de armazenamento secundário. Para esta consulta, a ferramenta faz-se acompanhar do parâmetro "-C" na sua execução. Estas

três soluções apresentam os seus valores com base em cálculos sobre dados disponibilizados pelo Sistema Operativo nos seus sistemas de ficheiros procfs e sysfs.

As soluções de ***back-end*** utilizadas têm a vantagem de serem totalmente adaptáveis às aplicações desenvolvidas e, consequentemente, muito positivas às necessidades deste projeto. Através destas ferramentas, independentemente da lista de inúmeros dados que disponibilizam, o utilizador tem a possibilidade de recolher apenas aqueles que são os necessários. São ferramentas mais flexíveis para apenas fornecer os dados requisitados sem que haja cálculos e processos desnecessários que possam sobrecarregar a monitorização. Nestas soluções, a fim de obter os indicadores pretendidos, inclui-se a ferramenta **PAPI**, a interface **RAPL**, e os sistemas de ficheiros do Sistema Operativo, **procfs e sysfs**. As duas primeiras estão relacionadas com a monitorização ao nível de *hardware* enquanto os sistemas de ficheiros destinam-se à monitorização a nível de *software*.

Com referido no subcapítulo 2.4, a ferramenta PAPI [110] é fundamental para aceder às informações energéticas apresentadas pela interface RAPL. A ferramenta PAPI permite o acesso e a consulta aos registos de contadores de eventos de desempenho do subsistema do processador. Com isso, é possível aceder a dados como, o numero de acessos a *cache*, numero de instruções e aos dados energéticos fornecidos pela interface RAPL. No entanto, num esclarecimento aberto com colaboradores da ferramenta PAPI, verificou-se que os contadores referentes aos acessos ao *cache* do processador apresentam ultimamente algumas lacunas, sendo que, esses PMCs têm baralhado as análises destas equipas [150]. Neste projeto, este facto não tem qualquer impacto, isto porque, se optou pela utilização da estimativa de energia fornecida pela interface RAPL para o subsistema do processador e memória primária. A interface RAPL pode ser considerada o coração do processo de desenvolvimento do modelo parametrizado. Esta interface apresenta um modelo desenvolvido pela Intel capaz de estimar a energia consumida pelos dois componentes e, possibilita a obtenção de dados diretos de energia sobre dois dos três componentes estudados para este modelo, o CPU e a DRAM [108].

Os sistemas de ficheiros, *procfs* e *sysfs* [119], são muito vantajosos para obter os indicadores que se baseiam em dados disponibilizados pelo Sistema Operativo. Estes sistemas de ficheiros foram desenhados para fortificar a estrutura de dados do sistema e apresentar uma solução uniforme do *kernel* para a interface do utilizador [120]. Estas soluções expõem ao utilizador informações do sistema e permitem a definição de parâmetros do sistema e dos seus controladores. O *procfs* apresenta dados relacionados com níveis de utilização do processador e com operações de IO, alias, é através destas fontes que o IOSTAT e o VMSTAT apresentam os seus valores. Desta forma, o *procfs* é utilizado também para recolher os dados do Sistema Operativo

referentes ao subsistema de armazenamento secundário. O *sysfs* [151] é um sistema de ficheiros com uma finalidade idêntica à do *procfs*, sendo que, já fez parte do mesmo no passado e é utilizado para recolher o indicador de potência consumida pelo sistema. Este indicador é obtido através da tecnologia *Smart Battery* [42] que, está disponível na configuração de hardware previamente escolhida e faz parte da especificação ACPI. Esta tecnologia permite ao Sistema Operativo consultar, através do controlador de Smart Battery, detalhes físicos como, a taxa de carregamento da bateria e o estado atual de funcionamento. O *sysfs* disponibiliza também detalhes sobre o ritmo de funcionamento das ventoinhas existentes no sistema. Esse indicador foi também estudado como uma alternativa possível para se obter os níveis de consumo de energia, uma vez que as mesmas apresentam ritmos de rotação que variam consoante os níveis de utilização do sistema. No entanto, não se verificou a necessidade de recorrer a essa solução, visto que, a tecnologia *Smart Battery* disponibilizava essa informação de forma direta.

3.5.2 Técnicas com Base na Implementação de Novas Soluções

A Recolha de Indicadores de Desempenho e Energia através da implementação de novas soluções é necessária quando se verifica que as soluções existentes não apresentam os indicadores procurados para satisfazer a totalidade dos objetivos delineados para este projeto. Esta situação ocorre no subsistema de armazenamento secundário onde, é preciso procurar novas técnicas que apresentem outras métricas mais esclarecedoras sobre o estado real de funcionamento do componente em causa. Para que isto seja possível, devem ser estudadas novas soluções para obter mais dados relativos às operações realizadas em disco pelo sistema operativo. Durante o desenvolvimento deste projeto foram estudadas várias fontes de informações capazes de apresentar dados de monitorização como o sistema de ficheiros *procfs* ou os contadores de eventos de desempenho, isto é, os PMC. No entanto, estas soluções não declaram diretamente o tipo de acessos que o sistema operativo faz sobre o armazenamento secundário [93]. Verifica-se então a necessidade de aplicar neste projeto uma nova prática de monitorização que siga de perto todas as operações de acesso ao subsistema.

Parte dos indicadores escolhidos anteriormente para monitorizar o subsistema de armazenamento interno precisa de ser monitorizado com base nas *system calls*: *write*, *lseek* e *read*. É necessária a implementação de técnicas capazes de manipular o típico funcionamento e os acessos do Sistema Operativo e das suas aplicações em execução a esse subsistema. As *system calls* Linux não são invocadas diretamente,

mas sim, através de funções *wrapper* na biblioteca partilhada de C do Sistema Operativo (*glibc*) [149]. As funções de *wrapper* da *glibc* são bastante simples. Estas realizam poucas mais operações do que as de copiar os argumentos para os registos certos antes de invocar a *system call* e, posteriormente, definem apropriadamente a variável *errno* que a *system call* retorna. Na maior parte dos casos, o nome da função *wrapper* é o mesmo da *system call* que ela invoca.

A solução encontrada passa por manipular a *glibc* e monitorizar todas as operações dos processos que recorram às funções *write*, *read* e *lseek*. Para o desenvolvimento do modelo, é suficiente que esta monitorização seja feita ao nível das aplicações e não do sistema operativo. Desta forma, existem dois caminhos possíveis para intercetar acessos à *glibc* através do conector do GNU: a variável *LD_PRELOAD* ou a *flag -wrap* [152]. A variável do sistema LD_PRELOAD é utilizada na execução da aplicação para carregar a biblioteca partilhada a fim de disponibilizar as funções de *wrapper*. A *flag --wrap* é aplicada no momento em que a aplicação se liga à *glibc*.

Para este projeto, considerou-se o LD_PRELOAD como a solução mais prática. Isto porque, no caso de existirem inúmeras funções a ser manipuladas, não é necessário especificar, uma por uma, na fase da ligação das aplicações às funções.

Como é possível ver na Figura 10, o *LD_PRELOAD* atua entre a camada de aplicação e as bibliotecas do sistema operativo e permite que uma determinada biblioteca seja carregada primeiro que quaisquer outras bibliotecas.

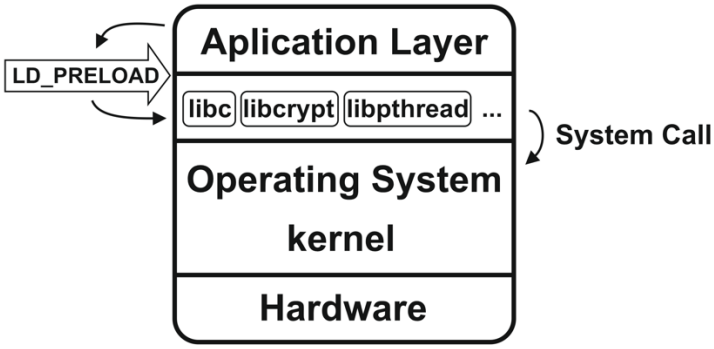


Figura 10 – Aplicação do *LD_PRELOAD* entre camadas do sistema.

No seguimento deste projeto, foi desenvolvida uma nova biblioteca que reescreve todas as funções referentes à escrita, leitura e reposicionamento da cabeça em disco. Depois de reescritas, através da função *dlsym()*, as funções devem ser reencaminhadas para as funções originais da *glibc* para executarem as devidas *system calls* [162]. Este processo é mais percetível ao visualizar-se a Figura 11, de seguida apresentada:

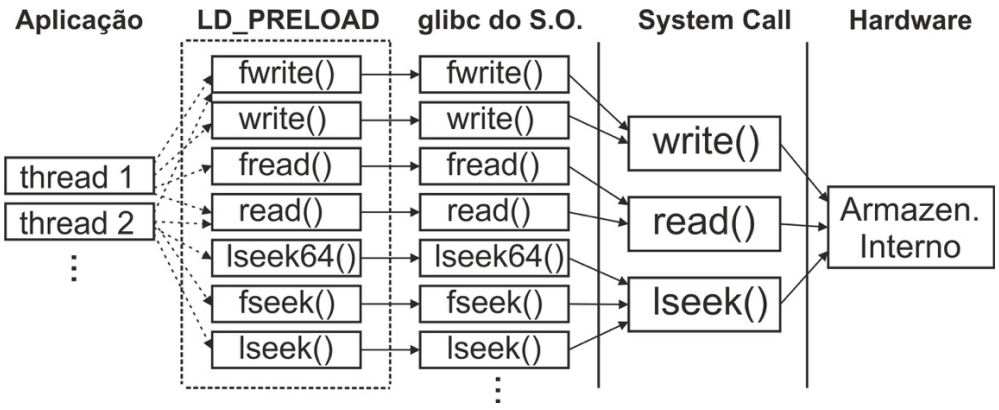


Figura 11 – Esboço Gráfico da Interceção da glibc através do LD_PRELOAD.

A biblioteca desenvolvida e introduzida através do *LD_PRELOAD*, tem a finalidade de registar a quantidade de vezes que determinadas funções são utilizadas e, nos casos de leitura e escrita, o tamanho dos blocos de cada função. Estes registos são atualizados constantemente num ficheiro que serve de interface para que os dados calculados sejam de possível acesso por outras aplicações exteriores à monitorização.

Esta técnica de interceção é apenas utilizada na validação do modelo final, quando este for colocado em prática em aplicações externas. Nas aplicações utilizadas nos ensaios, ao serem exclusivamente desenvolvidas para o projeto, são conhecidas todas as operações realizadas durante os exercícios. Desta forma, a técnica do LD_PRELOAD é colocada em prática na validação para contabilizar as funções utilizadas pelas aplicações *open source* que são independentes do projeto.

3.6 Fiabilidade das Ferramentas

A escolha de algumas ferramentas para sustentar o modelo desenvolvido obriga a um prévio estudo sobre a fiabilidade das mesmas. É importante que as ferramentas utilizadas no modelo sejam coerentes para desenvolver um modelo sólido e estável. O processo de comprovação da fiabilidade e precisão das várias ferramentas pode ser feito com base em dois métodos distintos: através da análise de estudos já realizados sobre esta matéria ou através de práticas próprias que impliquem a utilização das ferramentas. A análise de artigos existentes, permite que se obtenha uma visão construtiva sobre os níveis de precisão estimados e como estes qualificam a fiabilidade de uma ferramenta. O que ocorre, por vezes, é que os artigos sobre determinadas ferramentas são pouco esclarecedores ou inexistentes. Nesses casos, procuram-se soluções mais palpáveis, por isso, são realizados testes ao sistema e analisados os resultados apresentados pela ferramenta.

No desenrolar deste projeto, a interface RAPL tem um papel fulcral para se obter as informações de consumo de energia do CPU e memória RAM, isto porque através das informações desses dois componentes, é também possível obter um melhor parecer do impacto energético do restante subsistema que faz parte do modelo, o armazenamento secundário. Esta interface acaba por interferir bastante sobre a precisão do modelo que se tenciona desenvolver neste projeto e, por isso, é essencial explorar a fiabilidade da mesma. Apesar de ser utilizada sobre a ferramenta PAPI, a ferramenta PAPI não é colocada em exames de fiabilidade visto que, esta apenas consulta os valores dos registos disponibilizados pelo processador [110]. Desta forma, a análise incide totalmente sobre a interface RAPL. Para se obter uma maior confiança sobre esta interface, foi estudada a fiabilidade da mesma e foram recolhidos alguns resultados de outros estudos que podem ser úteis para se retirarem as devidas conclusões da mesma. Num estudo levado a cabo por Dongarra et al. [153], o RAPL foi comparado com medições reais de consumo de energia. A sua conclusão indica que o RAPL se apresenta como uma alternativa viável às medições físicas [153] [109]. Num outro estudo, foi comparada a potência consumida indicada pelas medições RAPL com as medições realizadas por instrumentação manual. Como resultado, verificou-se que os valores das medições de CPU calculadas pelo RAPL apresentam-se muito próximas daqueles que foram medidas por instrumentação externa ao sistema [94].

Como foi possível analisar neste estudo, a ferramenta RAPL apresenta várias referências positivas sobre a sua eficácia na monitorização de energia. Apesar de tudo decidiu-se examinar a ferramenta com exercícios recorrendo aos recursos disponíveis neste projeto. A fiabilidade da ferramenta foi analisada no seu desempenho a estimar um consumo de energia estático e dinâmico. Devido à falta de recursos, no consumo de energia estático, a ferramenta foi analisada unicamente sobre o subsistema de memória primária.

Desta forma, o **consumo de energia estático** que é estimado pela ferramenta para o subsistema de memória primária é confrontado com a documentação de estimativa energética fornecida por fabricantes deste componente. Ao existir coerência entre os valores apresentados pelos especialistas e os valores obtidos na execução desta ferramenta, assegura-se uma maior fiabilidade sobre estes indicadores utilizados. Para se realizar esta análise, os resultados do consumo de energia estática obtidos na ferramenta foram comparados com os valores apresentados por um simulador disponibilizado pelo fabricante destes componentes.

A Micron [155], dispõe de um simulador bastante utilizado no estudo de consumos energéticos do subsistema em questão. Este simulador estima a energia consumida por um subsistema de Memória RAM DDR3 através de um modelo baseado em especificações do componente como a frequência e o numero de pins do *bus* para

dados do mesmo. Esse artigo destinado ao calculo da energia consumida por um sistema de memória para DDR3 estima um valor de 0,44 *watts* por cada DIMM de memória primária. Ao adaptar-se o simulador à configuração de memória primária utilizada neste projeto, este sugere que o consumo energético deste subsistema apresenta um valor aproximado de 0,45 Watts por DIMM em estado *idle*. Ao confrontar estes valores com aqueles que a ferramenta RAPL estima no subsistema de memória primária deste projeto, verifica-se que a diferença é minima. Como se pode ver no gráfico da Figura 12, a ferramenta RAPL estima que o consumo da memória primária em *idle* é de 0,82 Watts e, sendo a configuração composta por duas DIMMs, este valor representa-se em 0,41 Watts por DIMM. Para conhecer o impacto energético do alto desempenho da memória, executou-se a ferramenta de *benchmark* RAMSpeed [170]. Esta ferramenta provocou um crescimento energético de 450% do subsistema, consumindo este uma média de 3,67 Watts durante este exercício de alto desempenho, tal como também se pode verificar na mesma Figura referida anteriormente.

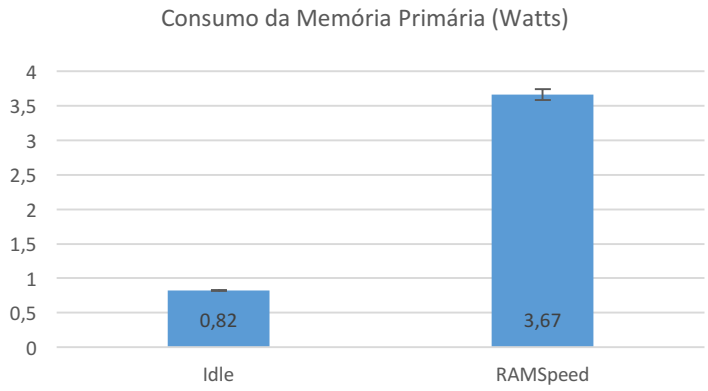


Figura 12 – Resultado de ensaio de consumo de energia da memória primária da configuração do sistema.

A verificação da fiabilidade da ferramenta na estimativa do consumo de energia dinâmico é uma tarefa complicada, principalmente, sendo esta feita sem recorrer a medidores físicos externos ao sistema. Ao contrario da verificação anterior, neste caso, a documentação existente não esclarece a atividade dinâmica do componente, até porque, esta pode variar bastante consoante a restante configuração de *hardware* e consoante as várias necessidades do utilizador e os tipos de exercícios realizados pelo mesmo. A solução encontrada para garantir alguma coerência da ferramenta passa por comparar com o indicador do consumo total de energia do sistema. Isto porque, apesar de não apresentar dados concretos de um subsistema, este indicador abrange estes dois subsistemas que, nos exercícios realizados são os únicos com impacto energético direto no sistema. Desta forma, no ato de exercitar um subsistema

ao máximo nível, o aumento da energia consumida que é notificado pela ferramenta RAPL deve ser idêntico ao aumento apresentado pelo indicador do consumo total de energia de um sistema. Para além disso, o facto de o indicador ser calculado através de uma solução diferente da ferramenta RAPL, isto é, com base na tecnologia existente para a gestão da bateria ACPI *Smart Battery*, permite que se confirme coerência entre as duas ferramentas e os seus indicadores em causa.

O gráfico da Figura 13 apresenta o consumo de energia estimado pelo RAPL (CPU, RAM) e pelo ACPI *Smart Battery* (TOTAL) em dois estados diferentes: *idle* e *active*. No estado *idle*, o sistema está completamente inativo, sendo que, utiliza apenas os recursos mínimos para se manter ligado sem qualquer atividade do utilizador. No estado *active*, é novamente executada a ferramenta de *benchmark* RAMSpeed que leva o CPU e a Memória Primária a um elevado desempenho e, consequentemente, provoca um maior consumo de energia nesses subsistemas. Através deste ensaio, é possível verificar-se no gráfico que, o consumo conjunto do CPU e RAM, é paralelo com o consumo total do Sistema. No entanto, foi calculada a variável OTHER que apresenta a diferença entre o consumo total do sistema e os subsistemas do CPU e Memória Primária. A variável forma no gráfico uma linha praticamente horizontal que comprova a coerência de ambas as ferramentas, RAPL e ACPI *Smart Battery*. Desta forma, é possível afirmar que existe uma conformidade entre a estimativa recorrendo à interface RAPL e a solução de ACPI *Smart Battery* que fornece o indicador da potência consumida no sistema.

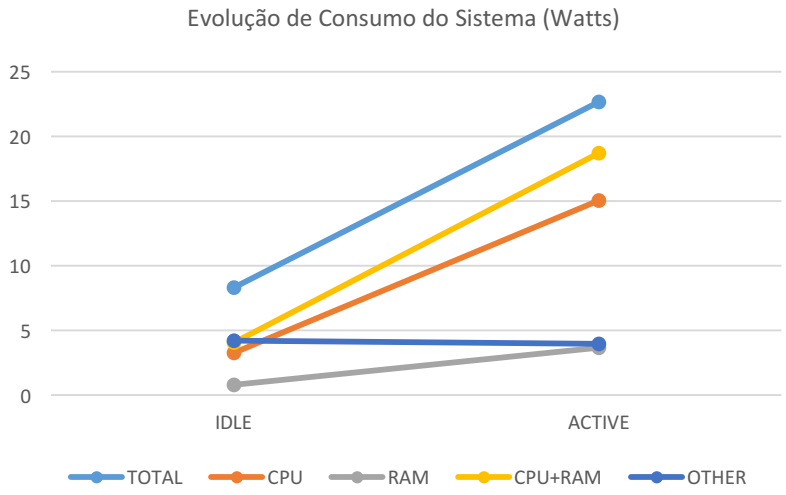


Figura 13 – Resultados do consumo de energia estimado pelo RAPL (CPU, RAM) e pelo ACPI *Smart Battery* (TOTAL) nos estados de *idle* e *active*.

3.7 Condições de Realização dos Ensaios

Estabelecer condições singulares para realizar todos os testes de exercício ao sistema é um passo imprescindível para fazer uma comparação imparcial e uma análise mais rigorosa dos seus resultados. Durante a realização de testes, o sistema deve-se apresentar estável e com o mínimo consumo de energia possível causado por efeitos ou periféricos alheios ao processo. Para isso, deve-se analisar bem os periféricos que se pretende utilizar e procurar o melhor método para tornar o sistema operativo o mais passivo possível e minimizar os processos de bastidores que o acompanham. Destaque-se que, nos ensaios realizados, **o Sistema Operativo será executado numa memória *flash* externa (*pendrive* USB)** para que haja o mínimo de interferência com o subsistema de armazenamento secundário que está a ser exercitado. Além disso, sendo este componente uma memória *flash*, o ruído energético que o mesmo provoca no consumo global do sistema é bastante reduzido.

A nível de software, o sistema deve-se apresentar o mais limpo possível e, para isso, decidiu-se executar o sistema operativo com o serviço LightDM desligado. O LightDM é um gestor de interface gráfica que é executado como *daemon* (em plano de fundo) e tem como finalidade apresentar a interface gráfica de comunicação com o utilizador. Este serviço, por apresentar uma interface leve, é utilizada em algumas distribuições Linux (e.g. Ubuntu, Arch, Debian) para *desktop* e portáteis e, muitas vezes, por servidores que necessitem da componente gráfica [158] [159]. Prescindir deste serviço que requer sempre mais recursos do sistema é uma opção bastante vantajosa para este projeto por vários motivos. Primeiro, apresenta um terminal de *shell* limpo e simples em vez da habitual interface gráfica prática e intuitiva para o utilizador. Segundo, numa comparação a nível de *software*, este modo apresenta-se em condições muito idênticas àquelas que são apresentadas pelos servidores em funcionamento nas enormes plataformas de *data centers*. Em terceiro e ultimo lugar, o numero de processos alheios ao utilizador e em execução pelo sistema operativo é bastante inferior comparado com o seu funcionamento em modo normal. Isto confirma-se após a realização de vários testes do sistema em *idle* nos dois modos, onde se verificou uma redução de 25% no numero de processos sem o serviço de LightDM ativo. Além de se prescindir deste serviço, decidiu-se realizar os testes com as placas de Ethernet e WiFi completamente desligadas. Isto faz com que se evite mais uma fonte de ruído alheio aos testes no consumo de energia do sistema.

A nível de hardware, a análise de condições recai totalmente sobre o debate de preferência entre periféricos externos ou periféricos incorporados no portátil para a execução dos testes. Apesar dos periféricos não serem uma fração importante para implementar no desenvolvimento de um modelo de consumos de energia, são

importantes a ser explorados por vários motivos. A principal finalidade de conhecer melhor o impacto destes periféricos a nível energético deve-se ao facto de estes poderem interferir negativamente nos exercícios praticados para o desenvolvimento do modelo. Por exemplo, é importante perceber que impacto tem, na realização de testes ao sistema, utilizar um monitor ou um teclado para facilitar a agilidade com que decorrem os testes. Para além disto, não é novidade que periféricos como um monitor, rato ou um teclado fazem parte do quotidiano de qualquer utilizador de um computador sendo, desta forma, importante perceber o consumo de energia provocado pela utilização dos mesmos. Como a eficiência energética é agora uma maior preocupação dos fabricantes de dispositivos móveis, interessa também perceber o impacto energético dos periféricos na autonomia de um portátil ou da energia consumida noutro tipo de computador. Numa primeira abordagem ao folheto [160] de detalhes de um rato e um teclado, verificou-se que ambos apresentam as características referidas na Tabela 7:

Tabela 7 – Especificações de periféricos analisados

	Rato Logitech B110	Teclado Logitech Media 600
Voltagem	5 V	5 V
Amperagem	100 mA = 0,1 A	100 mA = 0,1 A

Com base nestes dados, e aplicando a Equação (3) obteve-se uma potência de 0,5W para ambos os periféricos. Apesar da realização desta estimativa, decidiu-se confrontar estes valores com uma análise prática do impacto energético dos periféricos quando estes estão ligados ao sistema escolhido para o desenvolvimento do projeto. Essa análise comprova que, em ambos os periféricos, os consumos de energia variam consoante o nível de utilização dos mesmos. No caso do teclado, num teste de não utilização do mesmo, o seu consumo representou-se por apenas 0,08W. Aquando do aumento da utilização para uma escrita constante, este atingiu os 0,47W que, por sua vez, é um valor muito próximo ao obtido no calculo. Em relação ao rato, a situação encontrada foi idêntica, variando de 0,15W até 0,5W consoante os níveis de utilização. Conclui-se assim que, com a utilização destes dois periféricos, o impacto será reduzido para as condições em que se pretende realizar os testes. Isto é, não sendo necessária qualquer atividade com o rato ou teclado durante os testes, estes apresentaram um consumo energético bastante reduzido que não colocam em risco a integridade do processo de desenvolvimento do modelo.

Como o sistema escolhido é um computador portátil, existe a possibilidade de utilizar um monitor externo ou utilizar o próprio ecrã incorporado no portátil para a realização dos testes e desenvolvimento do modelo. É então necessário encontrar as

vantagens e desvantagens de cada uma das opções para que, se escolha a configuração mais estável e com o menor impacto energético possível. Para isso, foram analisados os vários níveis de brilho do ecrã do computador portátil Lenovo L440 e o impacto energético dos mesmos na bateria do portátil. O portátil apresenta um ecrã de 14 polegadas com uma resolução em 16:9 de 1600x900 pixéis [161]. Durante o teste, verificou-se um notável e crescente impacto energético desde o ecrã estar desligado até atingir o nível de brilho máximo de 100%. Como é possível verificar no gráfico da Figura 14, os valores variam entre 0 e 2,11 *watts*. Com esta análise conclui-se que o ecrã incorporado pode atingir uma quota de 25% do consumo total do portátil se este estiver no estado *idle*, com baixos níveis de utilização. Os consumos de energia apresentados no gráfico resumem-se a uma média calculada com base numa amostra de 30 valores para cada modo de operação do ecrã incorporado.

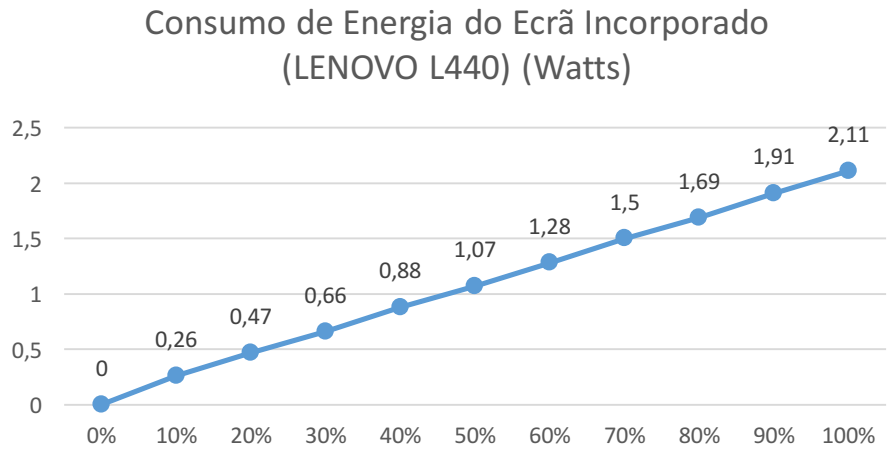


Figura 14 – Consumo de Energia do Ecrã de Portátil (Incorporado) – Modelo Lenovo L440.

No caso da opção de um monitor externo, o problema do consumo de energia do ecrã incorporado no portátil deixa de ser um problema. Foi realizado um teste com monitor externo e, concluiu-se que, o consumo do sistema é exatamente igual ao caso de nível de brilho de 0% do ecrã incorporado. Assim, pode-se concluir que o monitor externo é a melhor solução para o sistema durante os ensaios de desenvolvimento do modelo. O ecrã externo tem a vantagem de estar ligado diretamente à energia e, assim, não depender da energia do sistema nem interferir como ruído na energia consumida pelo mesmo.

Numa perspetiva híbrida, em que se relaciona o *hardware* com o *software*, a ferramenta HDPARM e as interfaces APM e ACPI têm um papel fundamental na realização destes ensaios. Nesta etapa, as definições estabelecidas nestes parâmetros devem ser vantajosas ao ponto de tornar o processo o mais puro possível. Os ensaios devem ser diretos o suficiente para não haver nenhuma ação complementar ao que

realmente se pretende exercitar nos componentes de armazenamento secundário. Desta forma, o controlo APM no disco é desativado a fim de colocar o componente no seu desempenho real sem qualquer interferência de gestão ou poupança energética. Também será desativado o acesso ao estado de *standby* do disco durante os testes para não haver o risco de o disco programar uma pausa imprevista no motor nas transições para os diferentes exercícios.

Resumindo, para o desenvolvimento do modelo, não serão utilizados quaisquer periféricos incorporados no portátil, a interface gráfica do sistema operativo não estará ativa e serão desativados os *screensavers* de poupança de energia do ecrã. Os testes serão realizados com o portátil a ser utilizado unicamente como um computador *desktop* onde se irá ligar o teclado, e monitor externo acima testados. O rato não será necessário visto que todo o processo de testes é realizado no terminal e sem a habitual interface gráfica. Verifica-se que nenhum destes periféricos causa instabilidade ao sistema na situação em que este será testado e, para além disso, torna o processo bastante mais pratico. As placas de rede serão também desativadas a fim de não interferirem com os ensaios. Além disto, é retirada a possibilidade de o disco estar no estado *standby* durante os ensaios. Este componente fica também com o controlo APM desativado. Na Figura 15, está apresentado o código-fonte que representa o script executado antes dos ensaios ao disco. Nele é possível verificar parte destas condições de realização de ensaio.

```
#!/bin/bash
#parametro que define disco em ensaio. ex: sdx
HDPATH="$1"

## RAPL & PMC: obter acesso total aos eventos do CPU através do PERF
sudo bash -c "echo -1 > /proc/sys/kernel/perf_event_paranoid"

## RAPL & PMC: carregar modulo MSR do kernel
sudo modprobe msr

## desativar controlos de screensaver
xset s off
sudo setterm -blank 0 -powersave off

##desativar interfaces de rede
sudo ifconfig wlan0 down
sudo ifconfig eth0 down

##prolongar modo ative/idle do disco para evitar o modo standby
sudo hdparm -S250 /dev/${HDPATH} ## 5 horas

## desativar o APM (Avanced Power Management) no disco
sudo hdparm -B255 /dev/${HDPATH}
```

Figura 15 - Código-Fonte, Ficheiro testcfg.sh, Configuração para os Ensaios.

4. Conceção e Implementação do Modelo de Consumo de Energia

Neste capitulo é apresentado o processo fundamental de desenvolvimento do modelo, desde a análise dos ensaios realizados até à construção e apresentação do modelo. Desta forma, numa primeira etapa, serão definidos os ensaios que se considera fundamentais para se conseguir um modelo solido, confiável e flexível para dar resposta às diferentes operações do sistema. Nesta fase, serão também apresentados os métodos estabelecidos para mecanizar os vários ensaios realizados. Em seguida, serão apresentados os resultados obtidos nos ensaios e tiradas as devidas conclusões em prol do modelo. Por fim, com base em todos os estudos e ensaios realizados sobre o sistema e suas tecnologias, será proposto o modelo que se considera mais indicado para a configuração do sistema que foi definida no capitulo anterior.

4.1 Ensaios Realizados e Especificações

Como referido anteriormente, o modelo de estimativa de consumo de energia que se pretende desenvolver neste projeto é baseado nos três subsistemas que, segundo os estudos realizados, apresentam um maior impacto sobre o consumo global de um sistema. Apesar do CPU e memória primária exibirem evoluções tecnológicas notáveis de gestão e monitorização de energia, o mesmo não acontece com o subsistema de armazenamento secundário. Desta forma, este subsistema necessita de ser estudado com mais profundidade para se conhecer melhor o seu desempenho e os recursos necessários para a sua atividade. Os componentes de armazenamento secundário podem ser acedidos de várias formas e para várias finalidades e, assim, com os estudos previamente realizados a este subsistema no projeto, acredita-se que todas as suas diferentes atividades tenham também impactos energéticos diferentes. Com isto, foram realizados vários ensaios personalizados para que sejam abrangidas as atividades consideradas fundamentais para os objetivos traçados para este projeto. As condicionantes que se consideram mais importantes para o impacto energético do subsistema são:

- estado de energia e operação;
- tipo de operação;
- tipo de acesso;
- politicas de utilização de *cache*;
- quantidade de dados por operação de acesso;
- utilização de sistema de ficheiros;

- IO *Schedulers*.

4.1.1 Estado de Energia e Operação

Conhecer o **estado de energia e operação** de um disco é o primeiro passo para se estimar o seu impacto energético no sistema. Como estudado anteriormente, o HDPARM disponibiliza esta informação, mas limitada apenas a três modos:

- ***active/idle;***
- ***standby;***
- ***sleeping/off.***

Estes três modos denunciam o estado atual de funcionamento deste subsistema. Apesar do sistema operativo caracterizar o estado de operação em três modos diferentes, existem modelos de discos que têm menos estados que não são possíveis diferenciar tão facilmente. Com os ensaios realizados, é possível verificarem-se diferenças entre os diferentes componentes examinados. É importante de referir que, o modo *sleeping/off* sugere sempre que o disco está completamente desligado e não tem qualquer impacto energético no sistema. O modo *standby* apresenta sempre um consumo reduzido com os requisitos mínimos ativos para o componente estar ligado. O modo *active/idle* apresenta um consumo mais alto do que os outros dois modos, no entanto, este modo é mais inconclusivo porque define um alargado número de atividades do disco. Neste último modo o disco tanto funciona com os serviços mínimos para dar uma resposta imediata ao sistema operativo, como funciona na sua atividade máxima em operações sequenciais ou aleatórias.

4.1.2 Tipo de Operação

O **tipo de operação** realizada em disco, é uma condicionante muito importante no impacto energético do subsistema e, pode dividir-se em apenas dois tipos:

- **operação de escrita;**
- **operação de leitura.**

Como é previsível, os dois modos de operação diferenciam-se pelo processo de gravação de novos blocos de dados e de leitura de blocos já existentes em disco e, consequentemente, pelo sentido que os dados tomam na comunicação com o sistema. Os processos de escrita e leitura no subsistema podem ser realizados de várias formas, sendo que, neste projeto o processo foi realizado através de um ficheiro e também, diretamente em disco (*raw mode*).

4.1.3 Tipo de Acesso

O **tipo de acesso** é outro dos fatores que mais influencia o consumo de energia provocado pelo armazenamento secundário. Entende-se que, para este projeto, o tipo de acesso pode ser distinguido em dois diferentes modos de funcionamento:

- **acesso sequencial;**
- **acesso aleatório.**

As operações, podem ser feitas de duas formas diferentes consoante a posição em que se pretender ler ou escrever em disco. Enquanto o acesso aleatório se traduz à execução de operações de forma ordenada e contínua, o acesso aleatório implica um exercício de reposicionamento por parte do disco e, consequentemente, um maior atraso das operações.

Com a vantagem da utilização da *cache*, estima-se que o acesso sequencial seja mais eficaz e apresente igualmente um menor custo por cada bloco de dados escrito ou lido. De qualquer forma, isso depende sempre das politicas de acesso definidas nestes ensaios e, mesmo sem recurso a esta curta memória *flash*, o acesso sequencial torna-se vantajoso por tomar uma ordem linear e mais acessível pelo componente. No caso do acesso aleatório há sempre uma obrigação do disco em aceder a uma posição de memória completamente diferente da anterior. Através do estudo já realizado, acredita-se que o impacto de operações de posicionamento nos discos HDD é bastante superior àquele que acontece nos discos SSD, principalmente, por este apresentar uma componente mecânica. Os discos SSD, sendo uma memória *flash* que apresenta apenas uma componente eletrónica, acredita-se que sejam mais eficazes em acessos aleatórios. Apesar disso, é possível que se verifique uma decadência de eficácia energética em operações de acesso aleatório, isto porque os discos SSD dispõem os seus dados de forma organizada para estarem preparados para obter melhores resultados em padrões de acesso sequenciais. Os acessos aleatórios prejudicam o processo de otimização das operações, isto porque, não se consegue prever facilmente quais os blocos de dados que possam ser futuramente acedidos.

Nas operações de acesso aleatório a discos HDD, sendo necessário que o disco opere em vários segmentos distantes, o exercício da cabeça é bastante superior e o desempenho do disco é bastante inferior, por se acreditar que estas operações aumentam o tempo de acesso e o consumo na realização das operações de escrita ou leitura. Como se verificou na secção 2.2.3, uma das maiores fontes de consumo de energia de um disco rígido durante atividades de *seek* do disco é a sua cabeça [86]. No caso dos acessos sequenciais dos discos HDD, os segmentos pretendidos estão posicionados de forma seguida e permitem uma maior rapidez e um menor exercício

por parte da mesma. Em ambos os casos, o funcionamento mecânico dos motores SPM que fazem mover os pratos giratórios é necessário. Assim, o consumo de energia desses movimentos e da componente eletrónica é bastante idêntica em ambos os casos.

4.1.4 Politicas de Utilização de *Cache*

As **politicas de utilização de *cache*** do disco são uma das fases do procedimento de leitura e escrita em disco que pode interferir bastante no consumo energético do componente. As políticas utilizadas e testadas no decorrer destes ensaios são:

- ***Read-Ahead***
- ***Write-Caching***

A memória *cache* deste subsistema pode ser utilizada em operações de leitura e escrita, através das técnicas de *read-ahead* e *write-caching*, respetivamente. Na leitura, a técnica do *read-ahead* resume-se à leitura de mais blocos do que aqueles realmente pedidos pelo sistema operativo. Para isso, o subsistema realiza um constante estudo de padrão nos últimos pedidos realizados para tentar prever os próximos blocos a serem requisitados. Neste processo, o subsistema, para além de ler os blocos requisitados pelo sistema operativo, lê também um intervalo de outros blocos que considera úteis para as próximas operações. Estes blocos extra lidos são então armazenados em *cache* do disco para estarem imediatamente disponíveis caso sejam futuramente pedidos pelo sistema operativo. A técnica de *write-caching* trata-se do processo de escrita em disco com o apoio da memória *cache*. Isto porque, os dados entregues pelo sistema operativo para serem escritos no subsistema, são primeiramente gravados em *cache* e, posteriormente num momento mais oportuno ao subsistema, são gravados em disco. Estas duas técnicas permitem que haja uma maior eficiência na escrita e leitura de dados. Por outro lado, impedem que se consiga ter uma melhor perceção sobre o impacto momentâneo do consumo energético de uma operação de escrita ou leitura. Desta forma, o subsistema irá ser posto à prova com e sem estas técnicas ativas. Assim, é possível verificar o impacto provocado pela *cache* do componente e, comprovar a gestão inteligente que o componente faz sobre os dados para os ler ou escrever no momento certo.

4.1.5 *Dataset* Utilizado por Operação de Acesso

O volume de dados requisitado por operação do utilizador ao sistema operativo é outro fator importante na eficiência de um disco, uma vez que, trabalhando com quantidades de dados mais altas, podemos evitar a execução de várias operações que

são realizadas por se trabalhar com quantidades mais reduzidas de dados. Consequentemente, evita-se um maior *overhead* e uma maior gestão de meta-dados na comunicação entre o sistema operativo e o subsistema de armazenamento secundário. Supõe-se, desta forma, que o processo fica mais limpo e simples quando as operações são realizadas com quantidades de dados superiores e, assim, também é possível que se verifique um menor impacto energético nessas mesmas condições. Para verificar a importância desta condicionante, decidiu-se realizar ensaios nas seguintes quantidades de dados por operação:

- **512 bytes;**
- **1024 bytes;**
- **2048 bytes;**
- **4096 bytes;**
- **8192 bytes.**

Como esclarecido anteriormente, na configuração de *software*, o sistema operativo está a funcionar sobre blocos lógicos de 512 *bytes*. No entanto, ambos os componentes de armazenamento secundário funcionam com blocos de 4096 *bytes* e acredita-se que haja uma diferença notável de impacto energético entre as operações de 4096 e 8192 *bytes*.

4.1.6 Formatação do Subsistema do Disco

A **utilização de um sistema de ficheiros** num disco influencia todo o processo das operações realizadas no mesmo. Para os ensaios realizados, decidiu-se estabelecer dois tipos de provas:

- **Com Sistema de Ficheiros (EXT4);**
- **Sem Sistema de Ficheiros (RAW).**

Acredita-se que a utilização de um sistema de ficheiros pode ter influencia nas operações realizadas no subsistema devido à robusta estrutura de dados em que este está organizado para ser facilmente consultado pelo sistema operativo. No mundo UNIX, o ficheiro pode ser considerado a unidade mais pequena em que a informação pode ser gravada. Um sistema de ficheiros é habitualmente uma coleção de ficheiros ou, por outras palavras, uma forma logica em que uma coleção de ficheiros é gravada e organizada para uma mais fácil gestão e manutenção [163]. A acompanhar cada ficheiro existe um conjunto de objetos que acompanham a fim de armazenar meta-dados e detalhes de hierarquias [163]. Deste grupo de objetos, destacam-se o ***superblock***, o ***inode*** e o ***dentry***. O ***inode*** insere-se em varias situações, como num ficheiro ou diretoria, e armazena atributos do mesmo e o local onde os blocos estão

instalados no disco. O ***superblock*** é essencialmente composto por meta-dados do sistema de ficheiros e define o tipo de sistema de ficheiros, tamanho, estado e informação sobre outras estruturas de meta-dados. Este objeto é uma estrutura de dados única no sistema de ficheiros e, desta forma, apresenta várias copias a fim de ser salvaguardado. Por fim, o objeto ***dentry*** é aquele que o *kernel* Linux utiliza para manter o conhecimento do rasto da hierarquia de ficheiros nas diretorias. Cada *dentry* mapeia um numero de *inode* para um nome de ficheiro e para a sua diretoria superior. A realização dos ensaios em modo RAW, isto é, sem sistema de ficheiros, permite o acesso ao dispositivo de armazenamento de forma direta sem atravessar as *caches* e *buffers* do sistema operativo nem recorrer aos controlos de meta-dados do mesmo. Em várias aplicações como em sistemas de gestão de bases de dados, há a possibilidade de utilizar sistemas de ficheiros ou o modo RAW para armazenar e aceder aos dados [163]. Apesar disso, verifica-se uma preferência no uso do RAW, diretamente para efeitos de melhor desempenho, sendo que, neste modo, estes podem gerir totalmente como os dados são colocados em *cache* em vez de dependerem de um sistema operativo como no caso do sistema de ficheiros. Logicamente, podendo esta condicionante interferir com o desempenho do disco, interfere também com o impacto energético do mesmo no sistema.

4.1.7 IO *Schedulers*

Os **IO *Schedulers*** são algoritmos que definem a ordem com que o sistema operativo executa as operações de IO que são submetidas ao subsistema de armazenamento secundário [155] [165]. Estes algoritmos podem ter muitas finalidades dependendo do objetivo, sendo que, alguns dos objetivos comuns são minimizar o tempo utilizado pelo disco para realizar as operações de *seek*, priorizar certas operações de IO de processos específicos, apresentar uma partilha equilibrada da banda de comunicação do disco com cada processo em execução e garantir que certos pedidos vão ser executados antes de um tempo limite particularmente definido.

De uma forma simplificada, pode-se dizer que os IO *Schedulers* exercem duas operações básicas: fusão e ordenação [155]. A operação de fusão baseia-se em juntar dois ou mais pedidos adjacentes de IO num único pedido. A quantidade de dados de IO processados será a mesma, mas, o numero de operações IO será mais reduzida. O processo de ordenação é o mais importante de ambos e, tem o objetivo de ordenar pedidos de IO pendentes numa sequência estabelecida pelo algoritmo. Sendo os ensaios realizados em *kernel* Linux, os IO *Schedulers* de controlo colocados à prova nestes ensaios do subsistema são:

- *Completely Fair Queuing* (CFQ);
- *Noop*;
- *Deadline*.

O **IO *Scheduler* CFQ** destaca-se por manter uma ordem de IO escalável por processo e tentar distribuir a largura de banda de IO equitativamente por todos os pedidos de IO. Cada fila de processo contém pedidos síncronos dos processos e os espaços de tempo alocados a cada fila dependem da prioridade de cada processo pai. O **IO *Scheduler Deadline*** destaca-se por utilizar um algoritmo de tempo limite para minimizar a latência de IO para cada operação. O seu principal objetivo é garantir um tempo para iniciar o serviço relativo a um determinado pedido, isto porque, o algoritmo impõe um tempo limite em todas as operações IO para prevenir situações de *starvation* dos pedidos. Por fim, o **IO *Scheduler Noop*** apresenta uma simples ordem de FIFO (*First In, First Out*), e assume que o desempenho do IO foi ou vai ser otimizado no componente de armazenamento secundário.

Nas versões primárias do *kernel* Linux, este selecionava o IO *Scheduler Anticipatory* como a opção predefinida de origem nestes sistemas para controlo de IO. Este introduz um atraso controlado antes de expedir os pedidos de IO a fim de os agregar e reordenar por proximidade na memória e, assim, reduzir as operações de *seek* do disco [166]. Este algoritmo é mais indicado para otimizar sistemas com discos mais pequenos e com menores velocidades de operação. Apesar deste algoritmo se mostrar bastante capaz para determinadas cargas de trabalho de um computador, com o passar do tempo, foi-se verificando alguma instabilidade em algumas funções. Por exemplo nos serviços de base de dados verificou-se um desempenho bastante fraco em cargas de trabalho mais criticas. Numa análise realizada ao IO *Scheduler*, concluiu-se que este pode provocar um decréscimo de desempenho até 15% na execução de uma base de dados [166]. Com isto, a partir de 2006, o Linux decidiu que o IO *Scheduler Anticipatory* seria substituído pelo CFQ como IO *Scheduler* predefinido e, em 2010, este acabou mesmo por ser excluído das opções do *kernel* [166] [167] [169]. O IO *Scheduler* CFQ apresenta grandes capacidades, não só para as necessidades de *desktops* como também, de servidores. Segundo Shakshober, este oferece o mais alto e equilibrado desempenho para uma alargada e diversificada gama de aplicações e arquiteturas de IO [165].

Logicamente, com a evolução tecnológica do subsistema de armazenamento secundário, as preferências sobre os IO *Scheduler* também começam a tomar novos rumos e, esses mesmos IO *Schedulers*, acabam por se atualizar e adaptar a esta evolução. Desta forma, apesar do Linux definir um preferido de origem, acabam por ser as distribuições desenvolvidas com base no *kernel* Linux a selecionar o IO

Scheduler predefinido. Por exemplo, a distribuição *Red Hat Enterprise* Linux 4 apresenta o CFQ por definição [165]. No entanto, existem algumas distribuições, como é o caso do Ubuntu com *kernel* Linux 3.16, que definem o IO *Scheduler Deadline* como predefinido de origem no sistema operativo [166]. Este último tem vindo a assumir-se como principal escolha numa fase em que os SSD surgem em maior quantidade no mercado dos subsistemas de armazenamento secundário. Num conjunto de testes realizados pela equipa Phoronix, verificou-se que a utilização do IO *Scheduler Deadline* para os SSDs sugere geralmente um melhor desempenho, pelo menos para os sistemas em funcionamento com um único disco [170]. Desta forma, o modelo, tanto para o disco HDD como para o SSD, será desenhado em torno do IO *Scheduler Deadline*, sendo que, os restantes algoritmos serão apenas analisados para se estudar os diferentes impactos energéticos.

4.2 Soluções para Realização dos Ensaios

O processo de exercitar um determinado sistema a vários níveis obriga à instrumentação do sistema operativo a realização de certas ou nenhumas tarefas, consoante a finalidade dos ensaios. Para esses exercícios ao sistema, recorre-se a ferramentas de *benchmarking*. As ferramentas de *benchmarking* desempenham um papel bastante importante para se distinguirem os vários níveis de utilização de determinados subsistemas e, consequentemente, do sistema no seu todo. Os níveis de utilização do sistema permitem encontrar os indicadores de desempenho que melhor acompanham as variações e acertar os parâmetros para obter um modelo final calibrado. As ferramentas de *benchmarking* têm a possibilidade de incidir mais precisamente sobre um determinado componente e esta característica é bastante útil para se conseguir comparar os valores apresentados pelos indicadores e do consumo total, quando um componente está ativo ou adormecido. Outras soluções são importantes para automatizar todo o processo de ensaios, desde automatizar a execução de inúmeros exercícios de *benchmarking* como também para realizar relatórios finais sobre esses mesmos exercícios. **Nesta fase do projeto, foram utilizadas três linguagens de programação** distintas, destinadas a diferentes funções: C, Java e Shell Script. **A linguagem C** foi utilizada para desenvolver as aplicações de *benchmarking*. Esta opção justifica-se por ser a linguagem mais próxima das *system calls* do sistema operativo e, por ser também, a linguagem utilizada na base do sistema operativo Linux. **A linguagem Java** é utilizada para realizar relatórios finais sobre cada ensaio realizado. Optou-se por esta linguagem por ser bastante acessível no tratamento de dados e, pela sua alargada gama de soluções para realizar parsing de ficheiros de log. Por fim, **a Shell Script** foi utilizada para instrumentar e temporizar

todo o processo de ensaios e relatórios finais, sem que seja necessário o utilizador intervir no sistema entre cada novo ensaio ou relatório. Foi decidido que cada ensaio fosse repetido dez vezes para que se consiga obter uma estatística mais coerente sobre os valores máximos e o desvio padrão existente.

4.2.1 Técnicas de *Benchmarking*

Nas soluções de benchmarking, preferiu desenvolver-se aplicações totalmente novas que sejam personalizadas para as necessidades traçadas, a fim de se conseguir corresponder a todas as especificações e parâmetros definidos para os ensaios. Foram exploradas algumas ferramentas existentes e bastante utilizadas noutros artigos da área e, verificou-se que, estas apresentam modos de operação bastante predefinidos. Além disso, são aplicações robustas de mais e pouco personalizáveis para os exercícios específicos que se pretende fazer. Nestes ensaios pretende-se realizar determinados exercícios ao subsistema de armazenamento secundário que sejam focados num conjunto de fatores que foram referidos na secção 4.1. Os ensaios devem ser o mais transparentes possíveis para que sejam apenas efetuadas as tarefas pretendidas. Para isso, são proporcionados ciclos bastante simplificados com apenas funções de escrita (*write*), leitura (*read*), ou de posicionamento (*lseek*), e um contador para que seja possível apresentar resultados estatísticos no final do ensaio. Estes ciclos são finitos a um tempo previamente definido como parâmetro das aplicações desenvolvidas. Além destas operações, nas aplicações desenvolvidas são acrescentados, em tipos de ensaios a que a isso obriguem, outras funções que assegurem as condições necessárias para o tipo de exercício em causa. Para se conseguir uma maior facilidade logística de tratamento de dados, depois das aplicações terminarem os exercícios ao sistema, gravam os detalhes dos ensaios em ficheiros (*logs*). Os logs apresentam a hora inicial e final do ensaio (HH:MM:SS), o tamanho de bloco (*bytes*), o tempo de acesso por ciclo (ms), o numero de ciclos executados por segundo, quantidade de dados escritos ou lidos nesse intervalo (*kbytes*) e, por fim, a taxa de transferência de IO (kb/s). Através destes *logs*, é possível automatizar a criação de um relatório final sobre, cada indicador calculado nesse intervalo de ensaios.

Para satisfazer todas as condições e tipo de exercícios definidos na secção 4.1, foram necessárias algumas adaptações a cada ensaio realizado, tanto a nível de configurações do sistema como nas ferramentas de *benchmarking*.

No controlo do **estado de energia e operação** para o subsistema de armazenamento secundário, recorreu-se a uma solução externa às aplicações de *benchmarking* desenvolvidas. Desta forma, utilizou-se a ferramenta HDPARM e as suas funcionalidades disponibilizadas através das *flags* -S, -Y e -C. A *flag* -S permitiu

definir o estado do disco, entre *active/idle* e *standby*. A *flag* -Y coloca o disco no estado Sleep. A *flag* -C permitiu consultar o estado atual de operação e energia do disco.

Com o propósito de ser possível realizar dois diferentes **tipos de operação** e controlar o ***dataset* utilizado** nessas operações, as aplicações de *benchmarking* recorrem às funções de *write*() e *read*(). Sendo que, nessas mesmas funções é possível definir o volume de dados (*dataset*) escritos ou lidos por operação. Na necessidade de controlar o **tipo de acesso** pretendido, sendo ele sequencial ou aleatório, utilizou-se a função *lseek*(). Esta função permite que se altere a posição do descritor para que o próximo *write*() ou *read*() atue sobre essa nova posição. Nas aplicações desenvolvidas, o acesso aleatório recorre a um calculo de *random*() prévio para que a posição nova seja plenamente aleatória. No caso dos acessos sequenciais, a função *lseek*() não é utilizada. A Figura 16 apresenta um excerto de código da aplicação de leitura aleatória a disco.

```
for (;;) {
        offset = (off64_t) numblocks * random() / RAND_MAX;
        retval = lseek64(fd, (512 * offset)-BLOCKSIZE, SEEK_SET);
        handle("lseek64", retval == (off64_t) -1);
        retval = read(fd, buffer, BLOCKSIZE);
        handle("read", retval < 0);
        count++;
}
```

Figura 16 – Excerto de código-fonte da aplicação de leitura aleatória a disco.

O controlo das **politicas de utilização de cache** foi feito de duas formas: através da ferramenta HDPARM e através das aplicações de *benchmarking*. Assim, assegurou-se que o disco respeitava as politicas definidas tanto por configuração do sistema operativo como por ordem da aplicação desenvolvida. **No HDPARM**, as configurações utilizadas foram as presentes na Figura 17, em seguida apresentada.

```
sudo hdparm -W0 /dev/sdx      #Write-Caching Desativo
sudo hdparm -W1 /dev/sdx      #Write-Caching Ativo

sudo hdparm -A0 /dev/sdx      #Read-Ahead Desativo
sudo hdparm -A1 /dev/sdx      #Read-Ahead Ativo
```

Figura 17 – Configurações de HDPARM utilizadas para controlo das politicas de utilização de *cache*.

Nas aplicações desenvolvidas, era necessário acrescentar certas operações a fim de desativar as técnicas de *Write-Caching* ou de *Read-Ahead*. No caso da **técnica *Write-Caching***, foram estudadas várias soluções como o *fsync*(), *fdatasync*() e *sync*() [183] [184]. Considerou-se que *fdatasync*() era a função mais indicada para a necessidade de que os dados fossem realmente escritos em disco, isto porque, é aquela

que menos meta-dados escreve por operação de escrita de dados e a que mais se limita a escrever os dados realmente pretendidos. Para se evitar a execução da função *fdatasync*() ao fim de cada operação de *write*(), encontrou-se a o *flag* O_DSYNC que realiza o mesmo serviço com apenas uma utilização na operação de *open*().

No caso da **técnica de *Read-Ahead***, para a sua desativação através das aplicações de *benchmarking*, como complemento ao comando da ferramenta HDPARM, utilizou-se a função *posix_fadvise(fd, 0, 0, POSIX_FADV_RANDOM)* [185]. Esta função desativa as operações de *read-ahead* realizadas pelo *driver* de *kernel* do sistema de ficheiros. O "*advice*" POSIX_FADV_RANDOM sugere que aquela área do sistema de ficheiros não será mais lida após este pedido.

A **formatação do subsistema de armazenamento secundário** é realizada com recurso à ferramenta *fdisk* [186]. A ferramenta tem a capacidade de manipular as tabelas de partições dos dispositivos de armazenamento secundário em Linux. No projeto, é utilizada para formatar o disco com o sistema de ficheiros Linux EXT4. Para recorrer a este sistema de ficheiros nas operações a disco, este necessita de ser montado numa diretoria do sistema operativo e, todas as escritas ou leituras são feitas em ficheiros nessa diretoria. No caso de se pretender utilizar o disco com formato RAW, o disco não necessita de ser montado pois, é diretamente acedido através da diretoria /dev/sdx. Qualquer escrita através deste modo, anula a formatação do sistema de ficheiros EXT4 que tenha sido previamente criada e impossibilita o acesso desse modo a todos os dados previamente armazenados no sistema de ficheiros do disco.

A **seleção do IO *Scheduler*** no sistema é feita na fase de carregamento do sistema operativo através da configuração do *bootloader* GRUB. Para isso, edita-se o ficheiro de configuração do GRUB "*/etc/default/grub*". Por exemplo, se for intenção definir o IO *Scheduler Deadline* como predefinido a ligar o sistema, acrescenta-se na linha referente ao *GRUB_CMDLINE_LINUX_DEFAULT*, o comando "*elevator=deadline*". Caso se pretenda escolher outro algoritmo, é só alterar a palavra "*deadline*". Por fim, é sempre necessário atualizar a configuração alterada através da execução do comando "*update-grub*" na *shell*.

4.2.2 Automatização dos Ensaios

O processo de automatização dos ensaios realizados é um passo imprescindível para haver uma maior flexibilidade no tratamento de dados. Os ensaios obrigam a que haja um funcionamento mecanizado de todo o processo por refletirem um elevado volume de dados para ser filtrado e analisado. Como referido anteriormente, nesta etapa foram introduzidas soluções em diferentes linguagens de programação que são

mais eficazes em determinadas tarefas. Como é possível ver na Figura 18, a *Shell script* comanda todo o processo, sendo através da mesma que são preparadas as configurações dos ensaios, o controlo da monitorização e a sequencia de exercícios de *benchmarking* realizados. No fim, o mesmo *script* conjuga todos os ficheiros resultantes das soluções executadas num programa JAVA desenvolvido para fazer separação, leitura e calculo de dados (*parsing*).

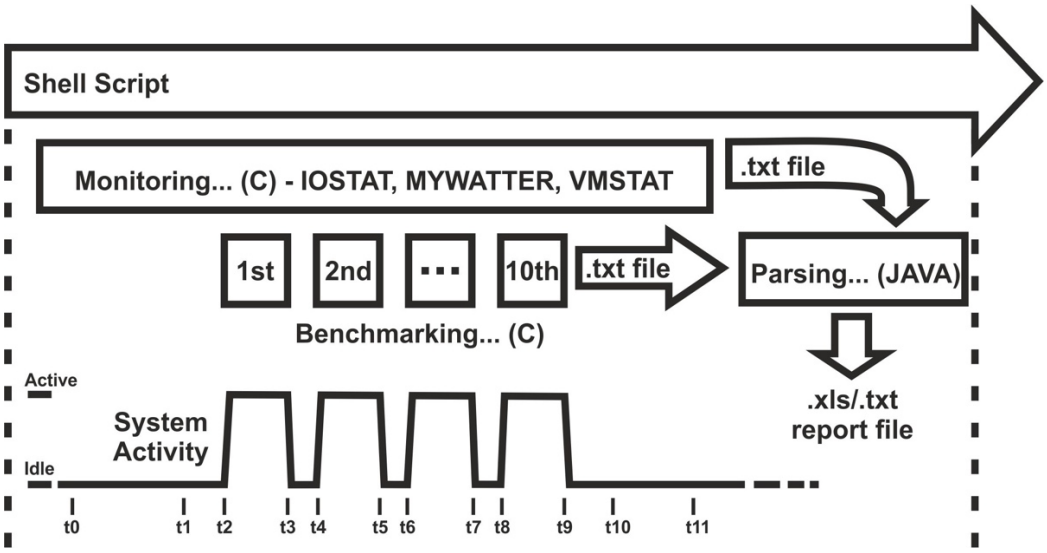


Figura 18 – Exemplo de ocorrências na execução da Shell script para os ensaios ao sistema.

Em cada ensaio realizado pela *Shell script* desenvolvido, repete-se dez vezes o mesmo exercício a fim de ser conseguir um resultado mais coerente. Além disso, como o exercício é realizado para 5 *datasets* diferentes (512, 1024, 2048, 4096, 8192 *bytes*), pode-se dizer que cada *script* processa 50 exercícios de *benchmarking*. Cada um desses 50 exercícios de *benchmarking* tem a duração de 1 minuto. Em cada *script*, composto por 50 exercícios, os indicadores de monitorização são calculados sete vezes em estados de *idle*. Por exemplo, na Figura 18, os períodos entre t0 e t1 ou t10 e t11, são associados a momentos em que o sistema está em *idle*. A Figura 19 expõe um excerto de código da parte mais importante da script. Aqui é possível verificar a realização dos cinquenta (5x10) exercícios e a monitorização dos períodos de *idle* entre os mesmos exercícios. Neste exemplo, o exercício realizado é sobre a escrita sequencial ao disco.

```
blocksize=( 512 1024 2048 4096 8192 )
sleeptime=60
for i in "${blocksize[@]}"
do
        for (( z = 1 ; z <= 10; z++ ))
        do
                echo "date && sudo ./writer_seq /dev/${HDPATH} $TIMEOUT
                        $i $ODSYNC $FDATASYNC $WRITE && date"
                date && sudo ./writer_seq /dev/${HDPATH} $TIMEOUT $i
                         $ODSYNC $FDATASYNC $WRITE && date
                printf "\n\n####### Test $z is over. \n\n"
                sleep $sleeptime
        done
        printf "\nSTARTING IDLE MONITORING - 2mins left.\n"
        date +%T | tr '\n' ' ' >> ${FILENAME}_log_idle.txt
        sleep 120
        date +%T | tr '\n' ' ' >> ${FILENAME}_log_idle.txt
        ps aux | wc -l >> ${FILENAME}_log_idle.txt
        printf "\nIDLE MONITORING IS OVER.\n\n"
        sleep 5
done
```

Figura 19 – Excerto de código-fonte do script que executa os ensaios de desempenho e monitorização.

Terminado o período de ensaios, os ficheiros de *log* são automaticamente carregados num programa em JAVA que calcula a média de valores dos indicadores de monitorização entre determinados intervalos. Todo este processo da *script*, dura proximamente 2 horas e 10 minutos até estar completamente concluído. Em seguida, os resultados finais são importados para o Excel onde são analisados e estudados ao pormenor. Na Figura 20, é possível visualizar um excerto de *output* no Excel, referente à monitorização através da solução *mywatter* que foi desenvolvida para o projeto. Neste exemplo, apenas são apresentadas parte dos indicadores analisados nessa solução de monitorização.

	HH:MM:SS	rapl:::PACKAGE_	rapl:::DRAM_	rapl:::PP1_	rapl:::PP0_	POWER_NOW
Idle	AVG_185105_185605	3,44	0,79	0	0,03	9,98
	AVG_191641_191841	3,44	0,79	0	0,03	9,94
	AVG_193851_194051	3,42	0,79	0	0,03	9,94
	AVG_200102_200302	3,36	0,79	0	0,03	9,89
	AVG_202314_202514	3,33	0,79	0	0,03	9,89
	AVG_204519_204719	3,23	0,79	0	0,03	9,89
	AVG_204724_205224	3,21	0,79	0	0,03	9,89
	Média	3,35	0,79	0,00	0,03	9,92
	Desvio Padrão	0,096214047	0	0	0	0,036384193
512	AVG_185638_185736	4,55	0,88	0	0,32	12,6
	AVG_185839_185937	4,45	0,86	0	0,3	12,4
	AVG_190040_190138	4,54	0,88	0	0,32	12,45
	AVG_190241_190339	4,51	0,88	0	0,3	12,57
	AVG_190442_190540	4,44	0,86	0	0,29	12,41
	AVG_190643_190741	4,46	0,86	0	0,31	12,43
	AVG_190843_190941	4,44	0,86	0	0,3	12,52
	AVG_191043_191141	4,49	0,86	0	0,33	12,41
	AVG_191243_191341	4,51	0,87	0	0,32	12,45
	AVG_191443_191541	4,5	0,88	0	0,3	12,59
	Média	4,49	0,87	0,00	0,31	12,48
	Desvio Padrão	0,040124805	0,00994429	0	0,012867	0,07930952

Figura 20 – Excerto de um ficheiro excel criado depois da execução do script de um ensaio.

Com o desenvolvimento deste sistema de automatização dos ensaios e da análise da monitorização dos mesmos, existe uma maior facilidade de explorar os dados, verificar as variações dos indicadores recolhidos e concluir o impacto energético dos exercícios realizados.

4.3 Análise dos Ensaios e dos Indicadores Recolhidos

Os ensaios realizados ao subsistema de armazenamento secundário têm uma elevada importância para a elaboração de um modelo sólido e fiável. Através destes ensaios e da análise realizada aos mesmos, torna-se possível obter um parecer sobre as diferentes situações a que um disco se pode sujeitar e, assim, fazer com que o modelo seja capaz de estimar mais corretamente a energia consumida nessas diferentes situações. Nesta fase, serão analisados os ensaios realizados e os indicadores destes recolhidos. A análise é feita uniformemente para cada disco e, em seguida, há uma análise geral sobre os diferentes componentes colocados em exame.

O IO *Scheduler Deadline* será o algoritmo utilizado para as analises principais do modelo. Desta forma, os restantes algoritmos são apenas utilizados para estudar a importância energética deste fator no subsistema de armazenamento secundário. A análise realizada aos ensaios é feita com base na variação de energia dinâmica dos três mais relevantes subsistemas através das variáveis: P_{active_CPU}, P_{active_DRAM}, P_{active_DISK}.

4.3.1 Ensaios de Inatividade do Disco

Como referido anteriormente, o elevado consumo de energia estático de um sistema é um dos grandes problemas das grandes centrais de dados dos dias de hoje [36]. Atualmente, existe uma maior preocupação em reduzir o consumo energético dos componentes quando estes estão inativos. Nesta etapa, pretende-se conhecer o custo da inatividade de dois componentes de armazenamento secundário. Para isso foram realizados ensaios aos mesmos nos seus diferentes estados de energia e operação. O sistema operativo só permite distinguir o funcionamento de um disco entre três diferentes estados:

- *Sleeping/Off;*
- *Standby;*
- *Active/Idle.*

Apesar disso, com o objetivo de ser possível distinguir o seu estado de *idle* com o seu estado *active*, foram realizados dois exercícios diferentes ao disco. Um primeiro exercício chamado de Base, consistiu em não realizar qualquer operação ao disco no

estado *active/idle* para que ele estivesse em pleno estado de *idle*. Na outra situação, foram feitos variados testes ao disco neste estado e foi capturado consumo médio e máximo obtido em situações de desempenho críticas.

Analisando-se o gráfico da Figura 21 relativo aos ensaios realizados sobre este tópico, é possível fundamentar várias situações que foram anteriormente previstas. No caso do estado *standby*, os dois discos exibem valores similares de consumo. Ao apresentar a sua componente mecânica desligada, o disco HDD apresenta uma redução maior entre este estado e aqueles de mais atividade. No caso do disco SSD, como é um disco composto por apenas componentes eletrónicos, apresenta um consumo base bastante invariante entre o estado *standby* e *idle* (Base *active/idle*). Na situação de atividade máxima de ambos os discos, verifica-se um maior consumo energético por parte do disco SSD. No entanto, será possível ver nos ensaios seguintes que esta situação do disco SSD é compensada pelo seu desempenho e eficiência de resolução de pedidos, tanto de escrita como de leitura.

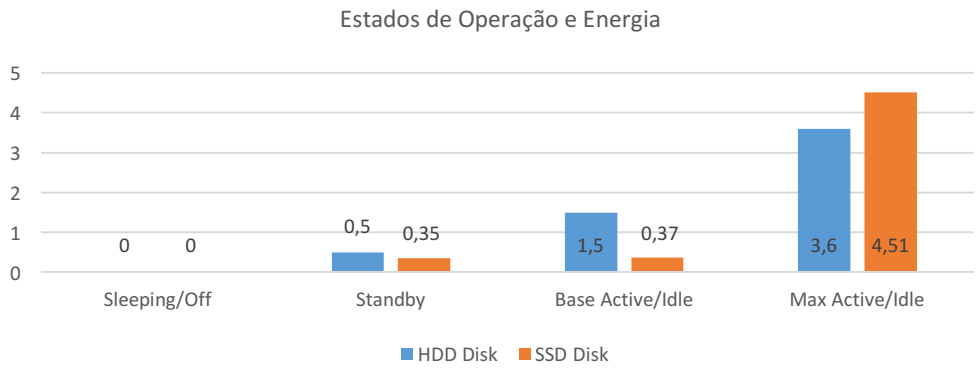


Figura 21 - Consumo de energia dos discos HDD e SSD nos vários estados.

Nesta primeira etapa, pode-se citar que, o consumo de um disco SSD apresenta resultados bem mais positivos na sua inatividade tanto em *standby* como em *idle*. O disco HDD encontra-se numa posição bastante desvantajosa nesta matéria, podendo consumir até mais do dobro do disco SSD. Esta situação, apesar de não ser tão preocupante em computadores pessoais, torna-se mais alarmante em grandes centrais de dados onde reduzir o consumo estático de qualquer subsistema em 50% é uma meta aliciante. Sendo o consumo estático um dos grandes temas das empresas de TI, segue-se para os exercícios de operações em disco com o cenário de preferência para a tecnologia de memórias *flash*.

4.3.2 Ensaios de Leitura em Disco

Os ensaios de leitura em disco são uma peça fulcral para o desenvolvimento do modelo de consumo de energia delineado para este projeto. É com base nestes resultados que será possível identificar os diferentes consumos de energia de leitura em disco e perceber quais os procedimentos de leitura em disco que podem causar mais impacto no consumo energético de um sistema. Com base no estudo realizado, decidiu-se dividir os ensaios de leitura em dois capítulos:

- Ensaios de Leitura Sequencial;
- Ensaios de Leitura Aleatória.

Acredita-se assim que, para este modelo, é importante haver uma distinção entre estes dois modos de leitura, até porque se prevê que a leitura sequencial seja mais barata e eficiente do que a leitura aleatória. Nestes ensaios, optou-se por recorrer apenas à leitura em modo *raw*. Esta decisão acontece porque se acredita que, nas operações de leitura, não existe uma diferença relevante entre a utilização deste modo e utilização do formato de sistema de ficheiros.

4.3.2.1 Modo Sequencial

A leitura sequencial a um subsistema de armazenamento secundário é um exercício que apresenta grandes vantagens relativamente às técnicas de otimização que são utilizadas para se atingir um melhor desempenho. Neste modo de acesso, o disco pode prever os próximos blocos a serem requisitados e, através da técnica de *read-ahead*, prepara-os antes do sistema operativo os pedir. Desta forma, para se analisar melhor o potencial deste subsistema na leitura sequencial e verificar-se o seu desempenho energético, optou-se por exercitar os discos com e sem a utilização desta técnica.

As principais análises para adaptar às necessidades do desenvolvimento do modelo são realizadas com os exercícios que utilizam a técnica *read-ahead*. Desta forma, pode-se verificar os diferentes desempenhos e impactos energéticos dos dois discos na leitura sequencial.

4.3.2.1.1 Disco HDD

No disco HDD, a leitura sequencial foi comparada entre duas diferentes situações: com e sem recurso à técnica de *read*-ahead. Isto porque, com o estudo já realizado, percebeu-se que a otimização do desempenho do disco pode depender bastante desta técnica.

Numa primeira análise, **recorrendo à técnica de *read-ahead***, verificou-se uma pequena variação de eficiência energética entre os diferentes *datasets* aplicados ao exercício. O gráfico da Figura 22 representa a evolução do consumo de energia dos três principais componentes e o volume de dados lido durante os vários exercícios realizados. Através deste gráfico, é possível verificar que o consumo de energia do disco é praticamente o mesmo independentemente do exercício. O volume de dados lido também é praticamente o mesmo, independentemente do tamanho de *dataset* em causa.

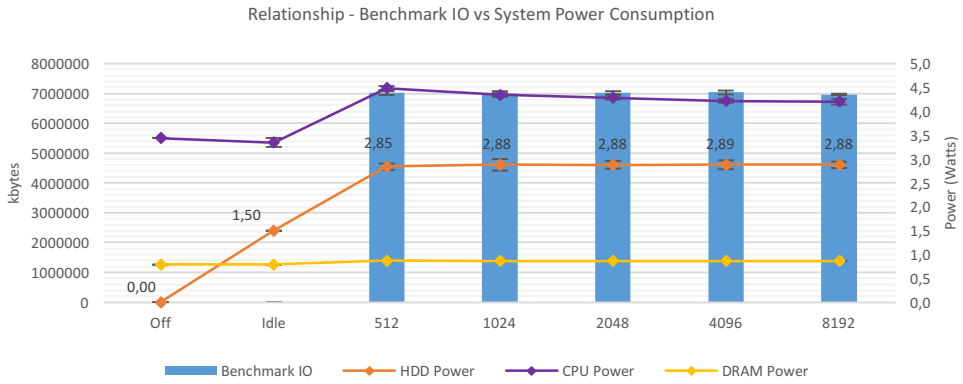


Figura 22 – Ensaio de leitura sequencial ao disco HDD com técnica de read-ahead ativa. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados lido nos diferentes datasets.

No consumo de energia do CPU, a situação é um pouco diferente, sendo que se apura uma pequena redução de potência consumida com o aumento do *dataset* aplicado. Esta situação pode ser mais facilmente percebida com a análise do gráfico da Figura 23.

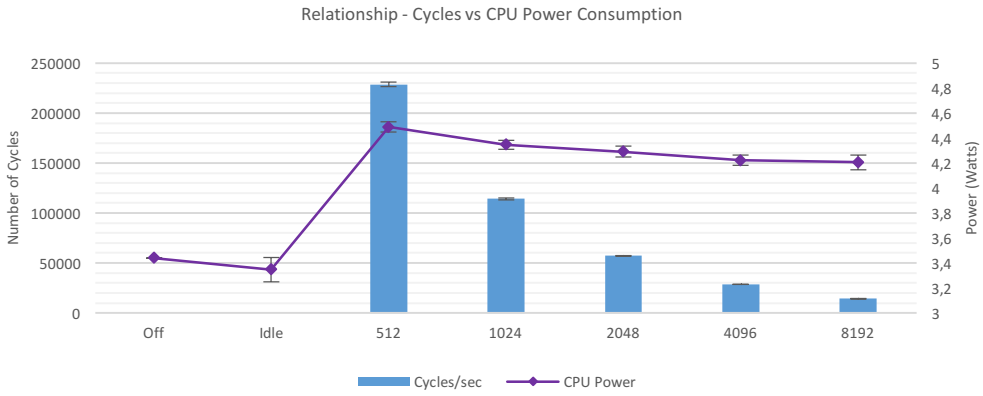


Figura 23 – Ensaio de leitura sequencial ao disco HDD com técnica de read-ahead ativa. Relação entre o número de ciclos por segundo de leitura executados e o consumo de energia do CPU.

Neste gráfico, é possível verificar que o consumo de energia do CPU reduz assim que o numero de ciclos de leitura diminui. Apesar da quantidade de dados lidos ser a mesma, em *datasets* mais pequenos, realizam-se mais pedidos ao sistema operativo para se obter os mesmos dados. Executando-se os pedidos, provoca-se uma maior carga de utilização ao sistema operativo que se reflete num maior consumo de energia do CPU. No caso do disco, apesar de o numero de ciclos executados causar mais pedidos de IO, o componente, não sofre variações mecânicas momentâneas e, apenas se limita a entregar os dados requisitados. Isto porque, como a técnica de *read-ahead* se encontra ativa e, o exercício de leitura tem um padrão sequencial, o disco já leu antecipadamente os dados.

No gráfico da Figura 24 é possível analisar o custo energético, por bloco e por operação, calculado com os exercícios realizados. O custo energético da leitura de um bloco sequencialmente é bastante estável. No caso do custo por operação, encontra-se uma tendência de este acompanhar proporcionalmente o aumento do *dataset*. Isto porque, as operações com *datasets* maiores, requisitam mais dados por operação e, consequentemente, ocupam mais tempo de acesso e consomem mais energia.

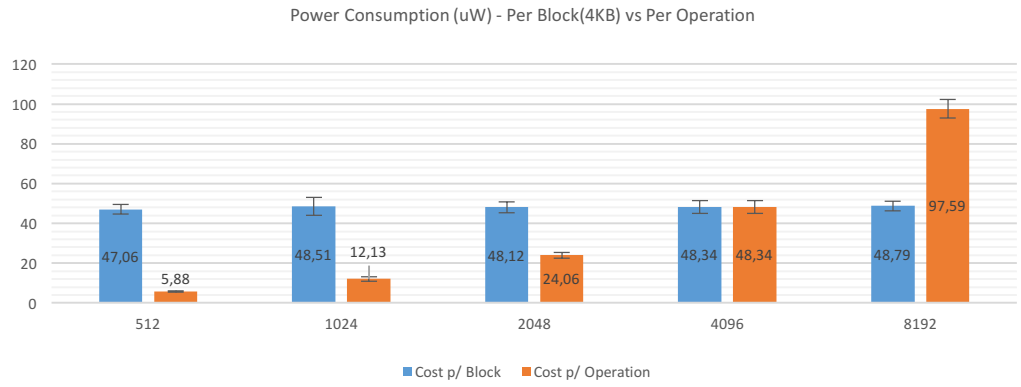


Figura 24 - Ensaio de leitura sequencial ao disco HDD com técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

O gráfico da Figura 25 apresenta uma análise sobre o output apresentado pelo IOSTAT. Nesta análise, através da variável IO-ReqSize, é possível verificar-se que a quantidade de blocos lógicos requisitados pelo sistema operativo é sempre 256. Cada bloco lógico representa 512 *bytes*, sendo que, a técnica de *read-ahead* está a provocar uma leitura fixa a rondar os 130 *kilobytes* de dados. Posto isto, numa operação de *dataset* de 4096 *bytes*, para além do *dataset* pedido, vão ser disponibilizados os 32 *datasets* seguintes a esse.

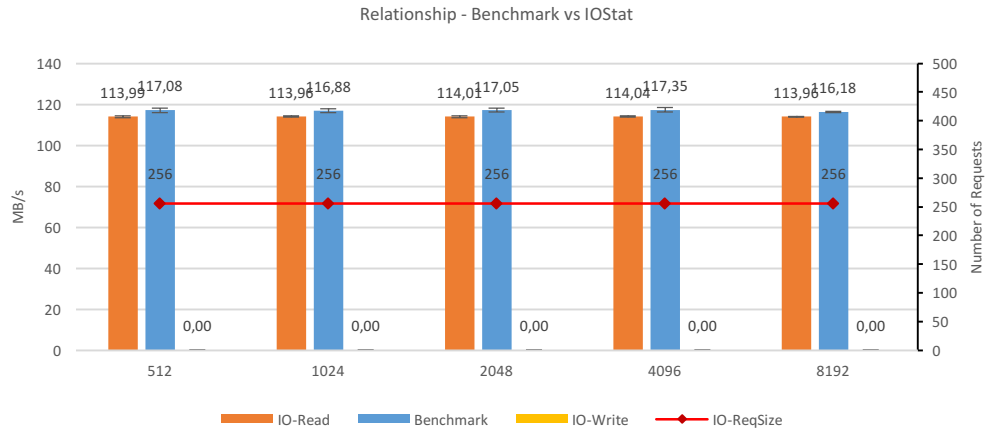


Figura 25 – Ensaio de leitura sequencial ao disco HDD com técnica de read-ahead ativa. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Na situação de leitura sequencial do disco HDD **sem recorrer a técnica de *read-ahead***, verifica-se um aumento no custo energético por operação e por bloco lido. O gráfico da Figura 26 comprova que o custo energético por bloco pode ser até 85 vezes superior no caso desta técnica estar desativada.

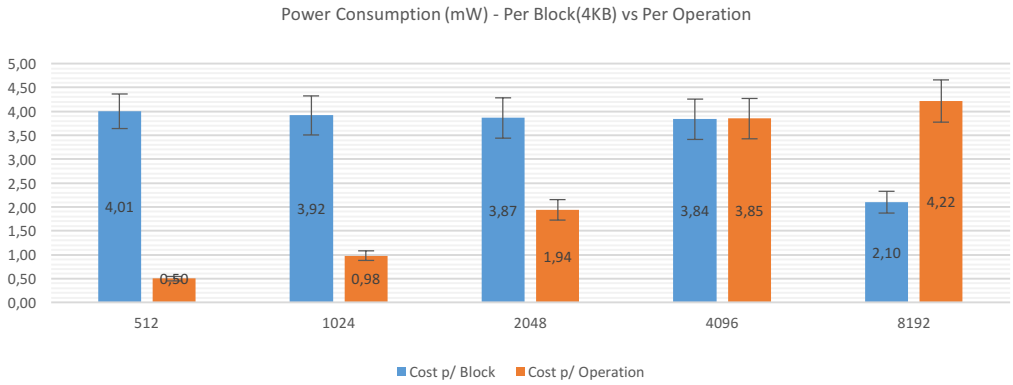


Figura 26 – Ensaio de leitura sequencial ao disco HDD sem a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Neste gráfico, é ainda possível verificar-se que o custo energético da leitura de um bloco, nos *datasets* inferiores a 8KB, é bastante idêntico. Isto deve-se à necessidade de o disco ler obrigatoriamente o bloco físico de 4KB, independentemente dos dados requisitados serem de um volume inferior a esse. Por exemplo, na leitura de um *dataset* de 2048 *bytes*, o disco lê fisicamente os 4096 *bytes* e entrega-os ao sistema operativo. Desta forma, no pedido seguinte do dataset de 2048 *bytes*, o sistema já tem esse volume de dados disponível e não obriga o disco a realizar novamente exercícios mecânicos. O *dataset* de 8192 apresenta um custo

próximo de metade dos restantes *datasets*. Isto acontece porque cada operação executada corresponde a duas leituras físicas sequenciais de 4KB, e, o disco, apresenta uma maior capacidade na realização desse exercício.

O gráfico da Figura 27 comprova o aumento de desempenho nas operações com *datasets* de 8192. A variável IO-ReqSize é disponibilizada pela ferramenta IOSTAT e, permite consultar a quantidade de blocos lógicos requisitados, tanto para leitura como para escrita, pelo sistema operativo ao disco. Através da variável, verifica-se que nos exercícios de *datasets* inferiores a 8192, a quantidade de blocos lógicos (512 *bytes*) requisitados, é sempre 8 e equivale a um bloco físico de 4KB. O exercício mecânico do disco é invariável nesses quatro *datasets*. No caso das operações de 8192 *bytes*, o sistema operativo lê 16 blocos lógicos ou, por outras palavras, 2 blocos físicos, do disco.

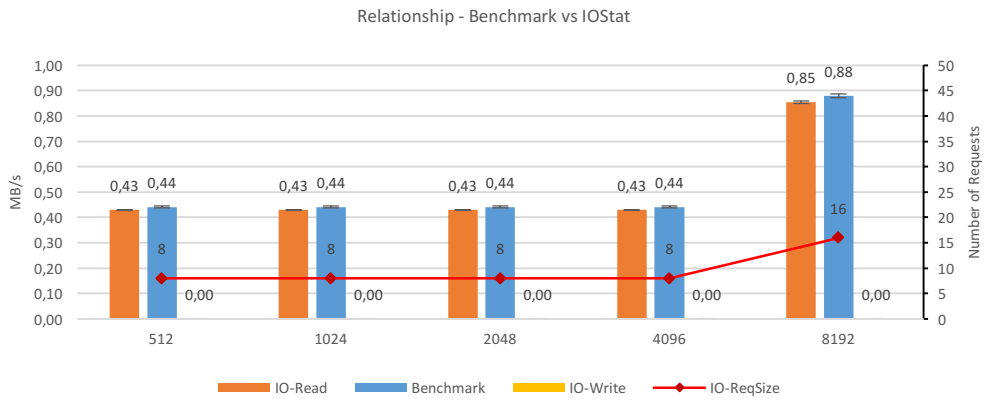


Figura 27 – Ensaio de leitura sequencial ao disco HDD sem a técnica de read-ahead ativa. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

O gráfico da Figura 28 comprova o consumo energético invariável do disco nos *datasets* entre 512 e 4096. No *dataset* de 8192, verifica-se um aumento pouco percetível do consumo energético do disco, por haver uma maior quantidade de dados lidos.

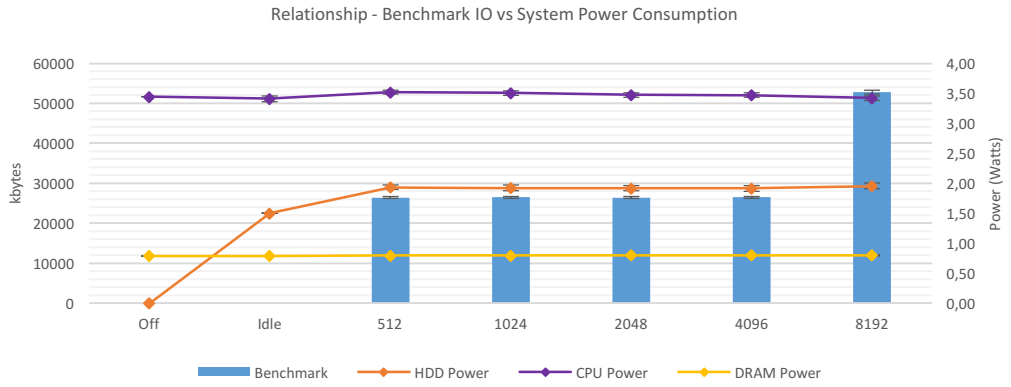


Figura 28 – Ensaio de leitura sequencial ao disco HDD sem a técnica de read-ahead ativa. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados lido nos diferentes datasets.

4.3.2.1.2 Disco SSD

Numa primeira análise sobre a leitura sequencial ao disco SSD, utilizou-se a técnica de *read-ahead*. Neste estudo verificou-se bastante estabilidade nos resultados obtidos pelo mesmo nos diferentes *datasets* colocados em exercício. O gráfico da Figura 29 comprova isso ao mostrar uma reduzida variação no custo energético por bloco nos diferentes *datasets*. No caso do custo por operação, verifica-se um crescimento proporcional da variável ao *dataset* em causa.

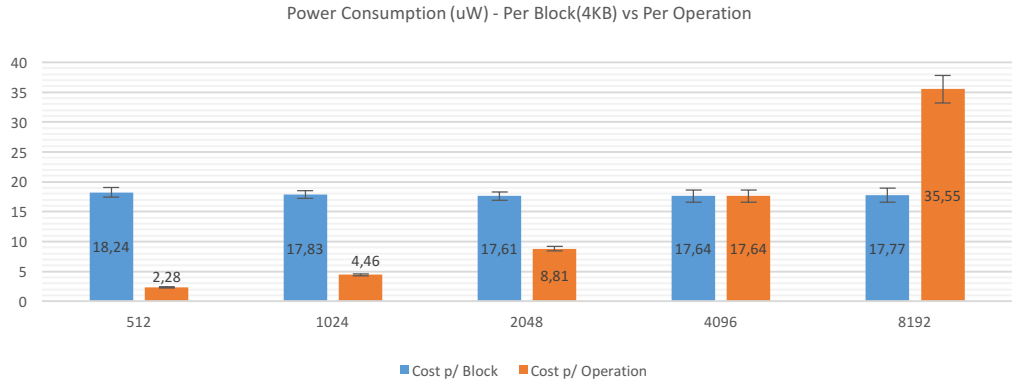


Figura 29 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

O gráfico da Figura 30 comprova esta estabilidade no impacto energético em disco entre todos os *datasets* examinados, isto porque, o consumo energético extraordinário do componente varia pouco e apresenta uma média de 2,38 *watts*. Além disso, o volume de dados lido num minuto é bastante idêntico nos diferentes exercícios e está próximo de 33GB.

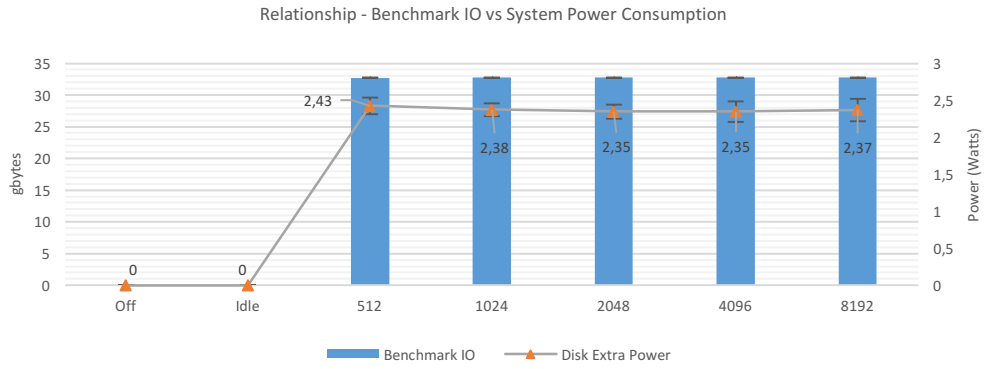


Figura 30 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa. Relação entre o volume de dados lido pela ferramenta de benchmark e o consumo extraordinário de energia do disco.

Apesar disso, ao analisar-se os dois restantes componentes destacados no consumo dinâmico, através do gráfico da Figura 31, verifica-se que o CPU consome mais energia em *datasets* mais pequenos. Isto acontece pelo maior numero de operações (ciclos) executadas para que se obtenha o mesmo volume de dados escrito. O consumo da memória primária é pouco variável durante os diferentes ensaios.

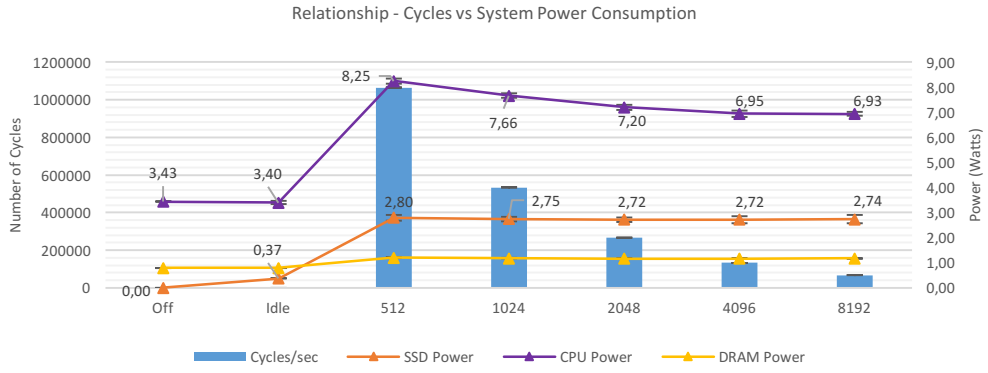


Figura 31 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa. Relação entre o consumo de energia dos principais subsistemas e o número de ciclos por segundo de leitura executados nos diferentes datasets.

Como é possível ver no gráfico da Figura 32, este ensaio decorreu com taxas de transferência de leitura do disco por volta dos 540 MB/s. O numero de blocos lógicos pedidos manteve-se sempre nos 256, garantindo assim o perfeito funcionamento da técnica de *read-ahead* em que está configurada para ler 256 blocos lógicos (equivalente a 32 blocos físicos).

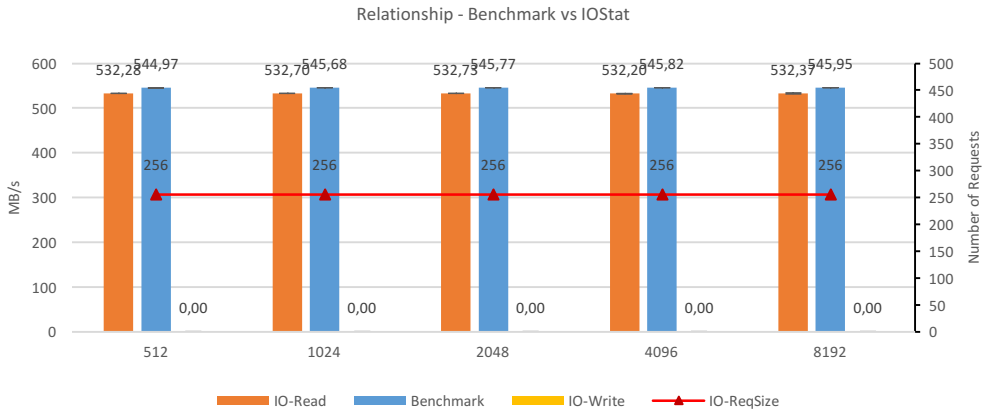


Figura 32 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Numa análise sobre outro ponto de vista de leitura sequencial, desativa-se a técnica de *read-ahead* nos ensaios. Desta forma, o disco não faz leituras que não tenham sido ainda requisitadas sobre volumes de dados que prevê que sejam futuramente lidos. O gráfico da Figura 33 representa o custo por bloco e por operação nessas condições. Através do mesmo, verifica-se que, sem utilizar a técnica de *read-ahead*, o custo por bloco é duas vezes mais caro. O custo por operação tem um crescimento proporcional ao crescimento do dataset. Isto justifica-se com o volume de dados processado em cada uma das operações.

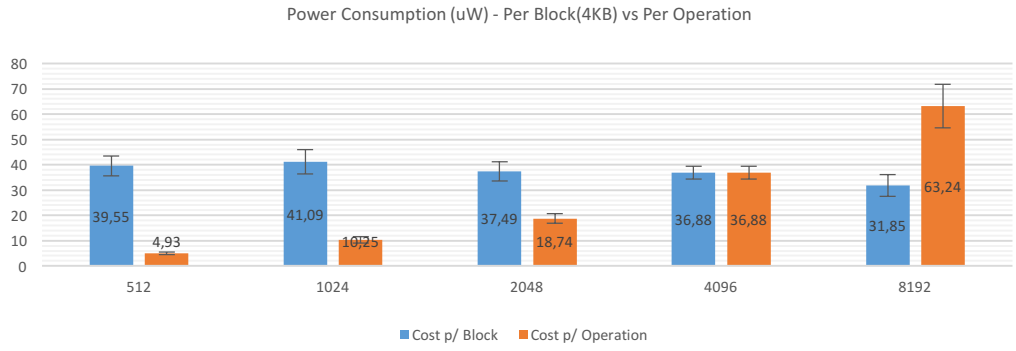


Figura 33 – Ensaio de leitura sequencial ao disco SSD sem a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

A maior variação de desempenho e custo por bloco verificou-se entre o *dataset* 8192 e os restantes *datasets*. Esta situação é visível no gráfico da Figura 34 e acontece pelo numero de blocos lógicos que o sistema operativo requisita ao disco em cada uma das situações. Enquanto que, nos *datasets* iguais ou inferiores 4096, o sistema operativo requisita sempre a leitura de 8 blocos lógicos de 512 *bytes* (total de 4096

bytes), no pedido do *dataset* de 8192, este requisita 16 blocos lógicos. Isto acontece porque o disco SSD se figura como um disco de blocos físicos de 4KB e, é apenas possível ao sistema operativo requisitar volumes de dados que sejam múltiplos desse tamanho. Entre os *datasets* de 512 e 4096, o desempenho do disco é idêntico porque o sistema operativo pede o mesmo numero de blocos lógicos independentemente do *dataset*. Por exemplo, no exercício de *dataset* de 512, o sistema operativo requisita a leitura de 8 blocos lógicos de 512 e fornece à aplicação apenas o conjunto de dados pedido. Os restantes blocos lógicos são guardados em memória para serem disponibilizados em futuros pedidos.

A taxa de transferência de leitura, sem recurso à técnica de *read-ahead*, atingiu uma média mínima de 120 MB/s no *dataset* de 512. Este resultado chega a ser quatro vezes inferior à mesma análise com recurso à técnica de *read-ahead*. Apesar disso, o custo por bloco não chega a crescer tanto com a anulação da técnica. Isto é justificado com o não tão elevado consumo de energia do disco nestes ensaios.

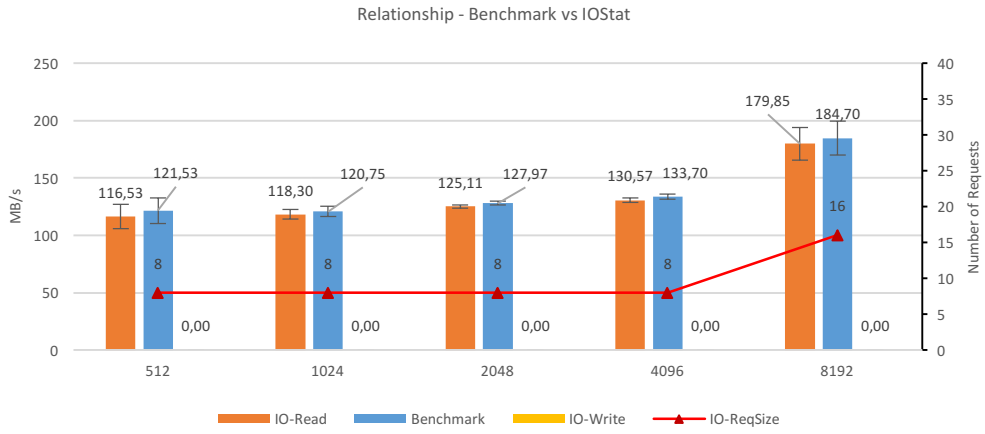


Figura 34 – Ensaio de leitura sequencial ao disco SSD sem a técnica de read-ahead ativa. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

O gráfico da Figura 35 apresenta o consumo de energia dos três principais componentes de atividade dinâmica. Nele é possível verificar que o disco tem um crescimento de consumo mais acentuado quando se exercita no *dataset* de 8192 *bytes*. Isto acontece devido a este *dataset* dobrar o numero de blocos lidos comparativamente aos restantes *datasets*. Consequentemente, o volume de dados lidos pela ferramenta de *benchmarking* também cresce de forma notável neste *dataset*.

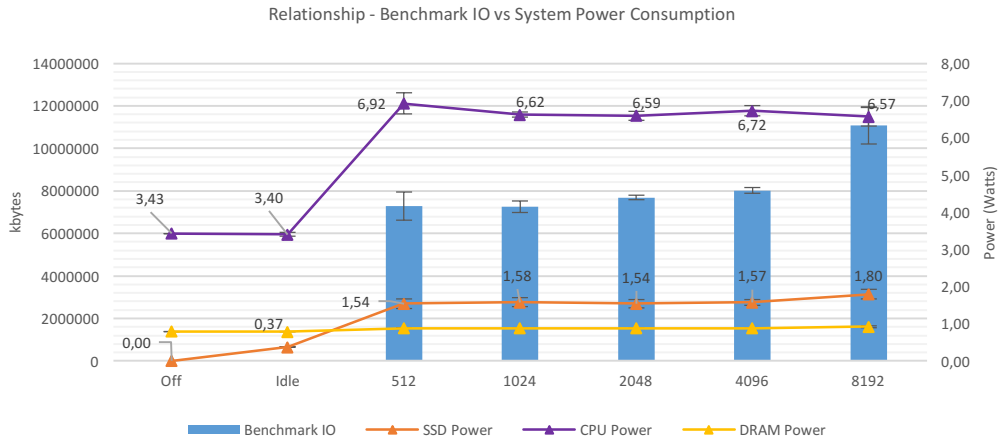


Figura 35 - Ensaio de leitura sequencial ao disco SSD sem a técnica de read-ahead ativa. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados lido nos diferentes datasets.

4.3.2.2 Modo Aleatório

Habitualmente, o exercício de leitura aleatória no subsistema de armazenamento secundário é mais intenso e exaustivo, comparativamente com o exercício de leitura sequencial. Isto porque, este exercício é realizado sem qualquer padrão que permita que o disco otimize os seus procedimentos de leitura.

O recurso à técnica de *read-ahead* será também analisado para verificar as capacidades de otimização dos discos em causa. Esta análise sobre este subsistema permite prever a improdutividade da técnica neste tipo de operações. Independentemente da utilização de técnicas de otimização, é possível que o desempenho do subsistema piore neste modo em relação às operações de leitura sequencial. Sendo que, o tempo de acesso ao componente será mais alto e a taxa de transferência de dados será bem mais reduzida. Com estas consequências, o custo energético acaba por se tornar bastante alto para uma simples operação de leitura.

4.3.2.2.1 Disco HDD

No disco HDD, o exercício aleatório de leitura é sempre uma tarefa árdua para o componente. Como verificado no estudo prévio sobre o mesmo, a sua estrutura mecânica torna este tipo de operações mais lento e com um maior custo energético por cada bloco lido. Estes factos podem ser averiguados nestas análises realizadas ao componente, sendo que, estas dividem-se numa primeira etapa com utilização da técnica de *read-ahead*, e uma segunda etapa em que se abdica dessa mesma técnica.

Com recurso à técnica de *read-ahead*, o custo energético por bloco e por operação pode ser verificado no gráfico da Figura 36. Em comparação com a leitura

sequencial do mesmo disco recorrendo a esta técnica, verifica-se um aumento no custo energético por bloco que pode atingir um valor duas mil vezes superior.

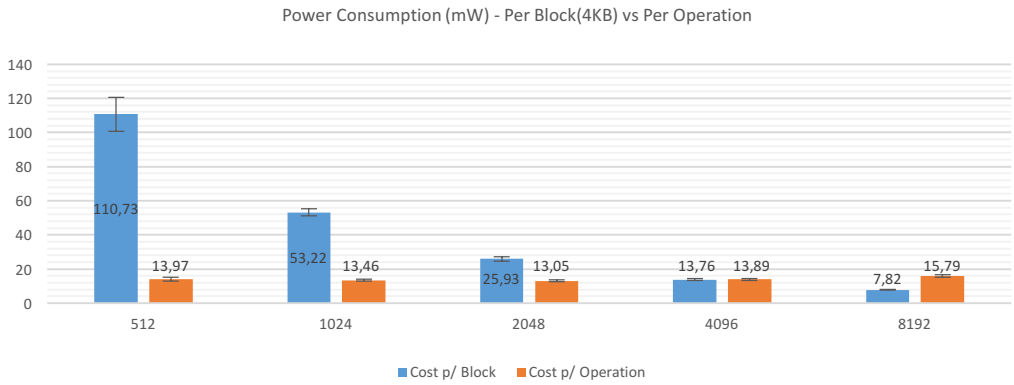


Figura 36 - Ensaio de leitura aleatória ao disco HDD com a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Neste exercício de leitura aleatória, o impacto energético da leitura de um bloco varia consoante o tamanho de *dataset* utilizado, sendo que, o valor decresce proporcionalmente à medida que o *dataset* é maior. O custo da execução de cada operação não apresenta grandes variações de valor entre diferentes *datasets*. Estes dois acontecimentos devem-se ao facto do custo maior deste exercício incidir sobre as operações de alteração de posição de leitura. Desta forma, o custo de operação é estável entre os diferentes exercícios, pois, a única situação variável entre eles é o *dataset*, sendo que a *system call lseek()* é sempre executada. O custo por bloco sofre um decréscimo por cada aumento do *dataset*, isto porque, existe uma maior facilidade em ler uma maior quantidade de dados entre cada *system call lseek()*.

Estes factos podem ser mais facilmente percetíveis através do gráfico da Figura 37. Este gráfico apresenta a relação entre o volume de dados lido entre diferentes *datasets* e o impacto energético para os principais subsistemas de energia dinâmica. Pode-se verificar que, o volume de dados lido aumenta proporcionalmente com o aumento do *dataset* e, o consumo energético dos três componentes não apresenta variações notáveis independentemente do exercício em causa. Esta situação faz com que o custo por bloco lido de 4KB seja menor à medida que o *dataset* aumente. Comparativamente com a leitura sequencial com recurso à técnica de *read-ahead*, verifica-se, na leitura aleatória, uma enorme redução do volume de dados lidos num minuto de exercício.

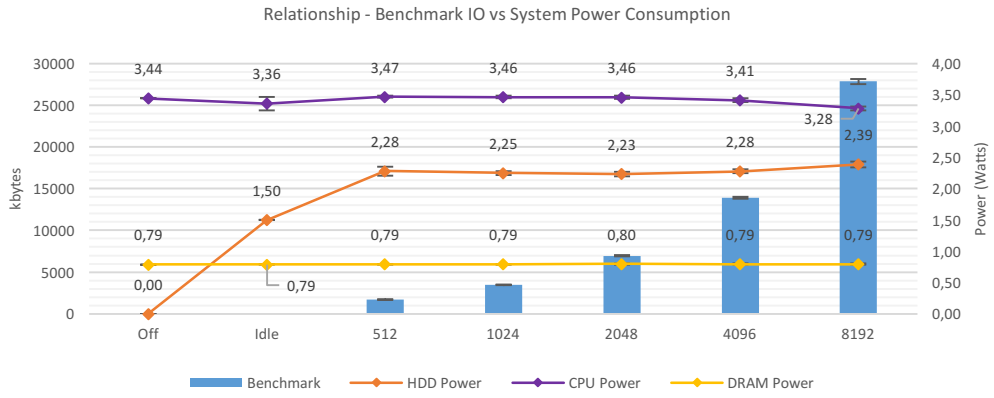


Figura 37 - Ensaio de leitura aleatória ao disco HDD com a técnica de read-ahead ativa. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados lido nos diferentes datasets.

Um outro fator importante para um menor desempenho do disco neste exercício surge com o desalinhamento entre a posição do *dataset* requisitado e a distribuição desse volume de dados pelos blocos físicos do disco. Isto é, neste exercício, o *dataset* é lido numa posição completamente aleatória, sendo que, este pode não começar no inicio de um bloco físico e necessitar da leitura de mais um bloco físico onde se encontra parte dos dados requisitados. Além disto, no caso dos *datasets* de 512, 1024 e 2048, a quantidade de dados lida é obrigatoriamente maior do que aquela que é realmente necessária devido aos blocos físicos serem sempre de 4KB [182].

Estes acontecimentos podem ser todos verificados no gráfico da Figura 38. A variável IO-ReqSize representa a média de blocos lógicos requisitados ao disco durante o minuto de exercícios. A variável Benchmark representa a taxa média de leitura de dados realmente requisitados pela aplicação de *benchmarking* desenvolvida. Por fim, as variáveis IO-Read e IO-Write apresentam a taxa real de transmissão de dados do disco de leitura e escrita, respetivamente. Por exemplo, no exercício de *dataset* de 512, verifica-se que a média de blocos lógicos requisitados é de 8, sendo que, cada um dos blocos lógicos corresponde a 512 *bytes*. Esta é a quantidade mínima de blocos lógicos possível de ser lida, devido a representar um bloco físico de 4KB no disco. Desta forma, neste *dataset*, apenas 1/8 dos dados lidos do disco é realmente aproveitado [182]. Nos *datasets* entre 1024 e 4096, pode-se confirmar o desalinhamento entre os *datasets* e os blocos físicos através da analise da variável IO-ReqSize. Esta variável apresenta sempre valores médios superiores a 8 e, no entanto, a leitura de 8 blocos lógicos seria suficiente para servir o volume de dados requisitados. No caso do *dataset* de 8192, a leitura de 16 blocos lógicos poderia ser suficiente para servir os dados requisitados. Apesar disso, o IO-ReqSize apresenta um

valor bastante superior, isto porque existe uma probabilidade muito reduzida da posição do *dataset* aleatoriamente requisitado coincidir exatamente com o conjunto de dados de dois blocos físicos de 4KB.

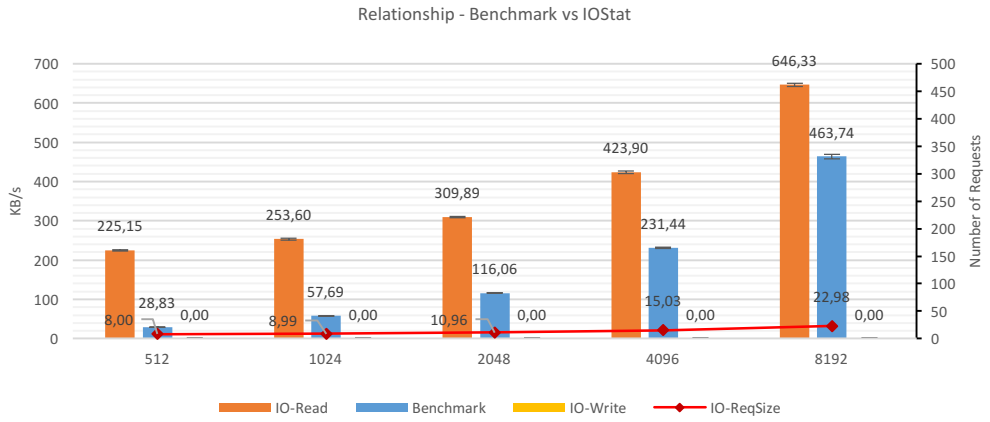


Figura 38 – Ensaio de leitura aleatória ao disco HDD com a técnica de read-ahead ativa. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Sem recorrer a técnica de *read-ahead*, obteve-se resultados bastante idênticos em todas as análises realizadas. Esta conclusão era previsível, visto que, a técnica de *read-ahead* não tem a possibilidade de otimizar as operações de leitura em situações em que estas não tenham qualquer padrão estabelecido. Desta forma, o custo energético, por operação e por bloco, é bastante idêntico e pode ser verificado no gráfico da Figura 39.

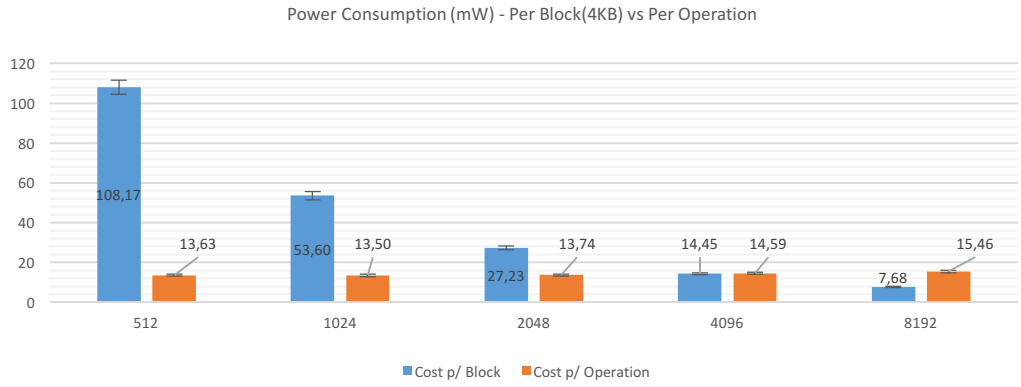


Figura 39 – Ensaio de leitura aleatória ao disco HDD sem a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Nestas condições, as restantes análises sobre os indicadores não são apresentadas por serem muito idênticas àquelas expostas com recurso à técnica de Read-Ahead.

4.3.2.2.2 Disco SSD

Nesta etapa são analisados os resultados obtidos nos ensaios relativos à leitura aleatória no disco SSD. Acredita-se que, nesta tarefa, este disco tenha um desempenho muito melhor comparativamente com o disco HDD. Numa primeira fase, **estudou-se os resultados obtidos no exercício com recurso à técnica de *read-ahead***. O gráfico da Figura 40 apresenta o custo médio por bloco e por operação de leitura. Através do mesmo, é possível verificar-se que, ao contrario da leitura sequencial, o custo por operação é muito idêntico entre todos os *datasets*. No caso do custo por bloco, verifica-se um decréscimo à medida que o *dataset* aumenta. Isto acontece porque grande parte do custo energético de uma operação é provocado pela mudança da posição de leitura e não pelo volume de dados lido. Consequentemente, o custo por bloco é também maior em *datasets* mais pequenos porque, nesses casos, é exigido um maior numero de operações e reposicionamentos para ler aleatoriamente o mesmo volume de dados de *datasets* maiores.

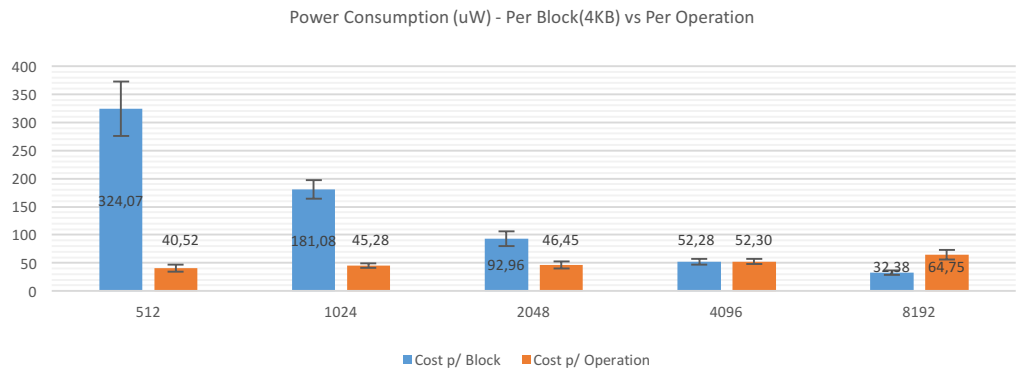


Figura 40 – Ensaio de leitura aleatória ao disco SSD com a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

As operações de leitura aleatória são imprevisíveis para o disco. Sendo que, neste tipo de ensaios, o processo é bastante lento e cada operação demora bastante tempoa ser resolvida comparativamente com as leituras sequenciais. Desta forma, quantos mais dados forem em recolhidos entre cada reposicionamento do disco, maior é também o volume de dados lido num minuto. Esta situação pode ser comprovada no gráfico da Figura 41 que, para além de apresentar o volume de dados lido em cada um dos *datasets*, também apresenta o consumo dos três principais componentes com impacto no consumo de energia dinâmico do sistema. O consumo dos três componentes é bastante invariável independentemente do *dataset* utilizado. No entanto, o desempenho de leitura do disco é totalmente diferente entre *datasets*.

Assim, pode-se confirmar que o reposicionamento de leitura causa atrasos e reduz a eficiência energética do disco.

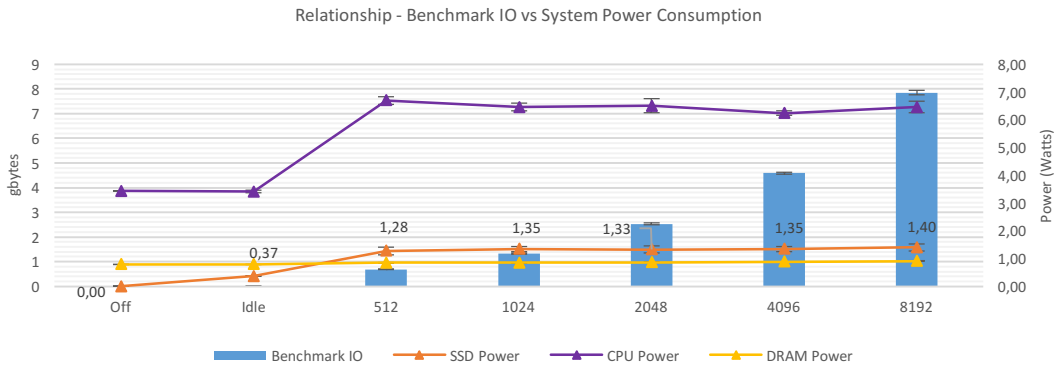


Figura 41 - Ensaio de leitura aleatória ao disco SSD com a técnica de read-ahead ativa. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados lido nos diferentes datasets.

Depois desta análise mais idêntica à realidade, **realizou-se este mesmo ensaio sem recorrer à técnica de *read-ahead*** para se perceber o impacto da técnica neste tipo de leitura. O gráfico da Figura 42 comprova que o custo por bloco e operação é bastante idêntico àquele encontrado nas condições anteriores.

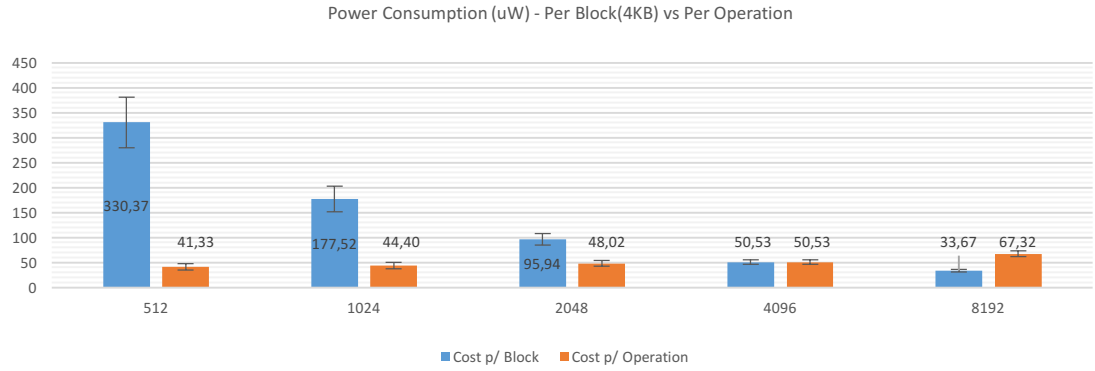


Figura 42 - Ensaio de leitura aleatória ao disco SSD sem a técnica de read-ahead ativa. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Com isto, é possível confirmar que a eficiência obtida na técnica de *read-ahead* é importante apenas para a leitura sequencial e bastante indiferente para leitura aleatória. Isto deve-se principalmente à não existência de padrão de leitura que, por sua vez, impede que o componente faça algum tipo de previsão sobre os próximos blocos de dados a serem requisitados.

O consumo energético dos três principais componentes de atividade dinâmica pode ser verificado no gráfico da Figura 43 juntamente com a quantidade de dados lida em cada *dataset*. Como no caso em que se recorre à técnica de *read-ahead*, a variação do

consumo entre *datasets* é mínima e irrelevante neste estudo. Verifica-se assim que a técnica não traz qualquer eficiência no processo de leitura aleatória do disco SSD.

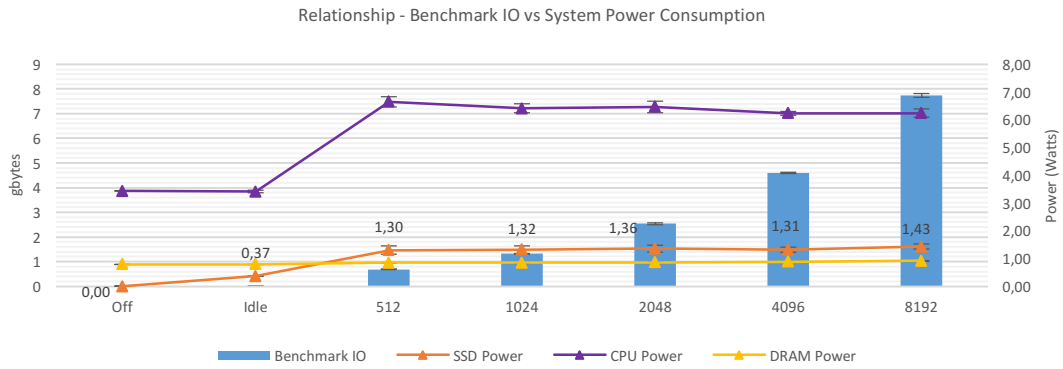


Figura 43 -Ensaio de leitura aleatória ao disco SSD sem a técnica de read-ahead ativa. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados lido nos diferentes datasets.

4.3.3 Ensaios de Escrita em Disco

Os ensaios de escrita em disco permitem que se perceba os diferentes impactos energéticos dos vários tipos de operação de escrita ocorridos no componente. Com isto, é possível completar o modelo de estimativa do consumo de energia para os dois componentes escolhidos para representar o subsistema de armazenamento secundário no sistema. O estudo realizado permitiu concluir que é possível haver uma diferença notável entre os acessos sequenciais e aleatórios e, desta forma, optou-se por dividir esta análise nestes dois diferentes modos. Como analisado anteriormente, acredita-se que os discos SSD apresentem um desempenho de escrita bem superior aos discos HDD e, também assim, é possível que apresentem uma maior eficiência energética neste tipo de operações.

4.3.3.1 Modo Sequencial

As operações de escrita em modo sequencial são habitualmente utilizadas como marco de referência para promover as melhores velocidades atingidas pelos discos. Como nas leituras, o melhor desempenho dos discos na escrita é em modo sequencial por várias razões como a otimização da localização física dos blocos e pelas técnicas implementadas em disco para este tipo de escritas.

A primeira análise incide sobre a escrita sequencial mais idêntica com aquela que o sistema operativo e as suas aplicações recorrem no seu funcionamento de quotidiano. Desta forma, a primeira análise recorre à escrita em sistema de ficheiros com a utilização da técnica *write-caching*.

4.3.3.1.1 Disco HDD

A primeira análise sobre a escrita sequencial no disco HDD é feita **com recurso à técnica de *write-caching* e em sistema de ficheiros EXT4**. Esta análise é a mais importante por ser aquela que mais se identifica com a convencional escrita de dados em disco a nível sequencial. No entanto, verificou-se alguma instabilidade e incoerência nos ensaios realizados. O gráfico da Figura 44 prova isso mesmo, verificando-se um alargado desvio padrão no custo por bloco dos exercícios com um *dataset* igual ou inferior a 2048. Os restantes *datasets*, 4096 e 8192, deste exercício apresentam resultados mais coerentes e estáveis. Acredita-se que, esta instabilidade é justificada pela grande quantidade de ciclos executados nos *datasets* mais reduzidos. Além destes resultados instáveis, considera-se que, no custo por operação, o crescimento devia ser mais proporcional ao aumento do *dataset*.

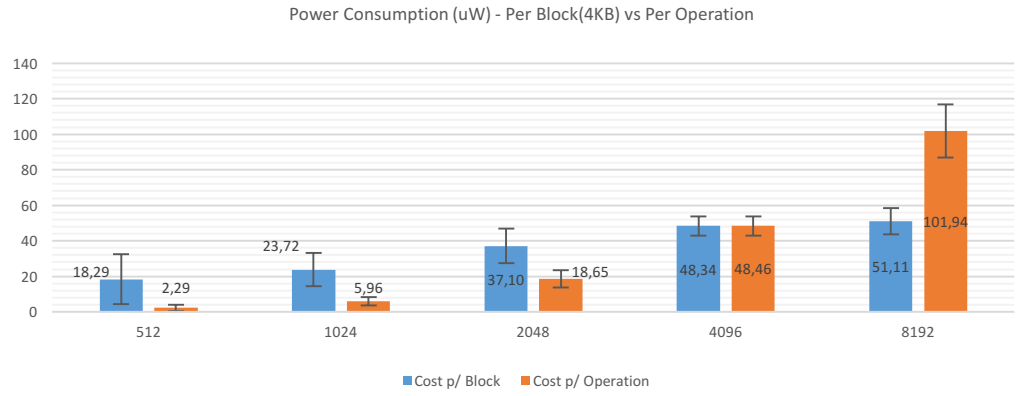


Figura 44 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

No gráfico da Figura 45 verifica-se que os exercícios referentes ao *dataset* de 512 apresentam o numero de ciclos por segundo mais elevado que, por sua vez, provoca um maior consumo energético do CPU. Com este alto nível de utilização do CPU, torna-se mais difícil filtrar o consumo energético do disco, sendo por isso que, nos *datasets* menores, este apresenta valores tão alargados de desvio padrão.

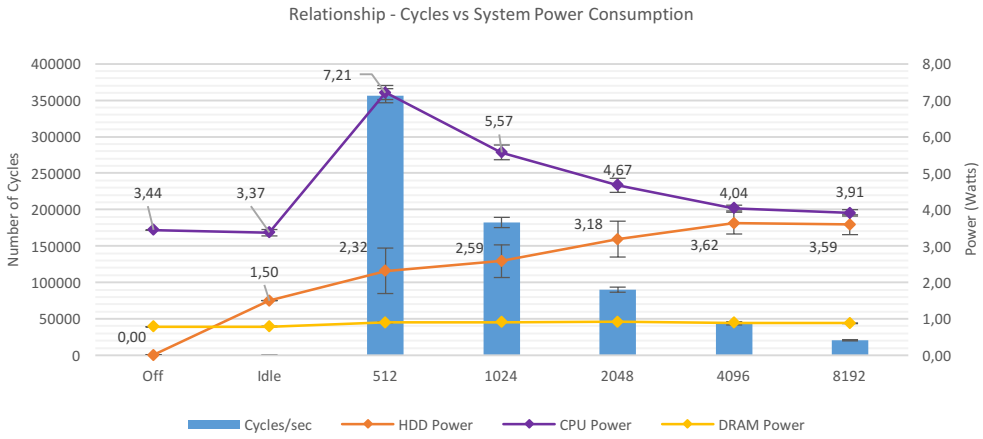


Figura 45 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o consumo de energia dos principais subsistemas e o número de ciclos de escrita executados nos diferentes datasets.

Depois desta etapa, decidiu-se explorar o impacto da **desativação da técnica da *write-caching* mantendo a utilização de sistema de ficheiros** em disco. O gráfico da Figura 46 representa o custo por bloco e por operação nos vários *datasets* em que foram executados os exercícios. O custo por bloco, no caso do *dataset* de 512, pode ser até 5000 vezes superior. Verifica-se assim que existe um enorme impacto energético caso seja desativada esta técnica. À medida que o *dataset* aumenta, constata-se um decréscimo proporcional no custo por bloco devido à redução no numero de operações que são requisitadas. O custo por operação é bastante estável entre *datasets*, isto porque, a exaustão do disco acontece na garantia de que os dados estão realmente escritos no momento. Desta forma, as operações têm todas um custo elevado e semelhante que sobrepõe a diferença mínima de impacto energético do volume de dados escrito. Este custo pode-se traduzir ao custo exato da execução de uma operação de escrita em modo *raw* se, esta, fosse instantaneamente atendida pelo disco. Com a utilização da técnica de *write-caching*, verifica-se sempre uma otimização que faz com que o disco escreva determinados volumes de dados num momento mais oportuno.

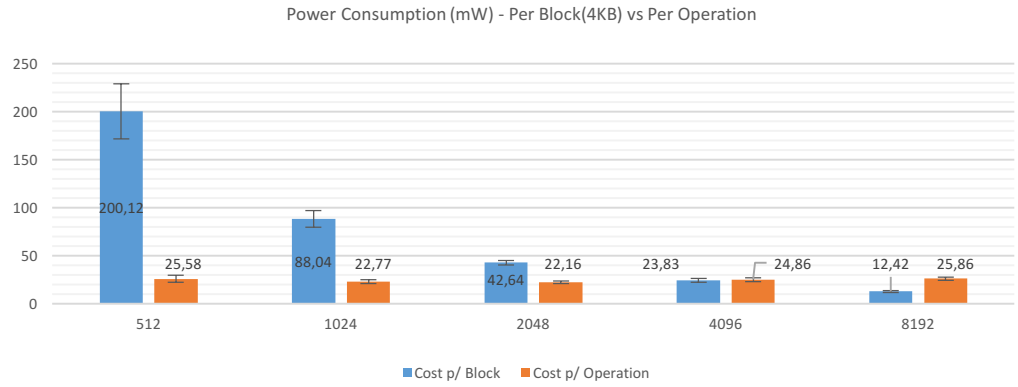


Figura 46 – Ensaio de escrita sequencial ao disco HDD sem a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Esta análise pode ser confrontada também com o consumo energético do CPU, da RAM e do Disco, através do gráfico da Figura 47. Verifica-se um consumo estável de energia por parte dos três principais componentes do sistema em análise.

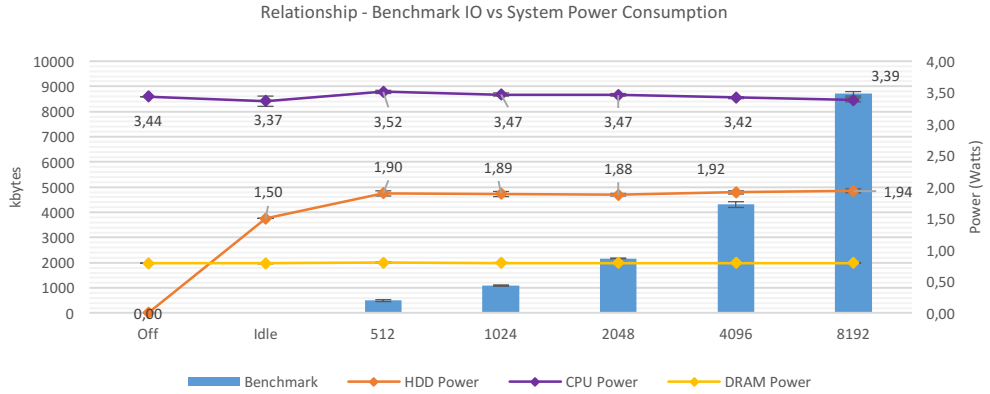


Figura 47 – Ensaio de escrita sequencial ao disco HDD sem a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

O CPU apresenta esta estabilidade devido ao numero de ciclos entre diferentes *datasets* ser muito idêntico. Desta forma, o volume de dados escrito sequencialmente cresce de forma proporcional com o *dataset* em prática, isto porque, a quantidade e o tempo de acesso dos ciclos, são invariáveis entre os diferentes *datasets*. Este facto acontece porque, depois do disco estar preparado para escrever os primeiros 512 bytes, está igualmente capaz de escrever rapidamente os restantes blocos lógicos requisitados. Apesar disso, o desempenho do disco na escrita de dados é extremamente baixo comparativamente com o mesmo ensaio recorrendo à técnica de

write-caching. Constata-se, neste caso, uma diferença entre 9 *megabytes* e 9 *gigabytes*, escritos num minuto de exercício.

Numa análise diferente, **sem utilizar sistema de ficheiros e com recurso à técnica de *write-caching***, verificou-se, em certos *datasets*, melhores resultados comparativamente ao uso de sistema de ficheiros. O gráfico da Figura 48 apresenta o custo por operação e por bloco dos 5 diferentes *datasets* colocados em prática no ensaio.

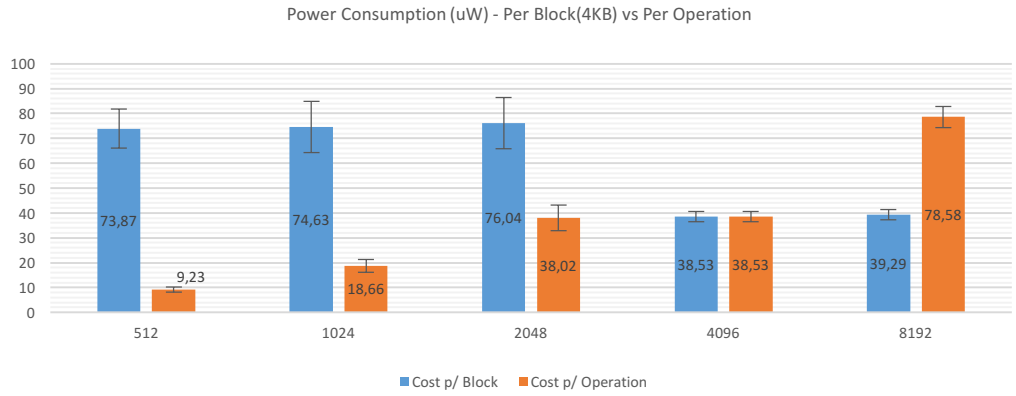


Figura 48 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Neste gráfico é possível verificar que nos *datasets* 512, 1024 e 2048, existe um elevado custo por bloco comparativamente com os restantes *datasets*. Esta questão foi explorada ao pormenor a fim de se encontrar a resposta para este facto. A resposta remete à análise do gráfico da Figura 49, que apresenta indicadores recolhidos na ferramenta IOSTAT e as taxas de escrita em disco da ferramenta de *benchmarking*. No gráfico é possível constatar que, em *datasets* inferiores a 4096, existe uma taxa de leitura alta que é provocada pela escrita de blocos de tamanho inferior ao tamanho físico de disco.

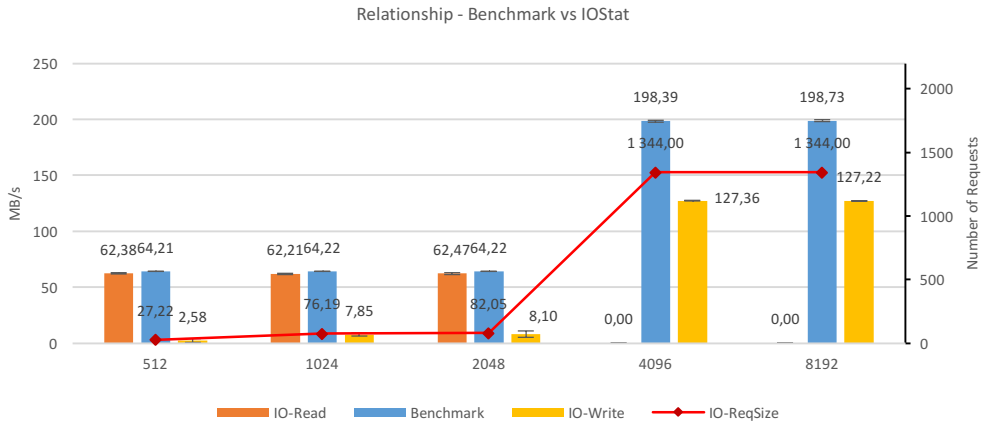


Figura 49 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em modo raw. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Isto acontece devido à técnica de *read-modify-write* aplicada pelo disco [182]. A técnica consiste em ler os dados existentes, modificar o intervalo de dados pretendido e escrever todo o conjunto de dados lido de novo no disco. Por exemplo, no *dataset* de 512, quando o sistema operativo tenta escrever o bloco lógico de 512 *bytes*, o disco vai primeiramente ler o setor físico de 4KB onde estão presentes os 512 *bytes* que vão ser alterados. Depois, são inseridos os 512 bytes de novos dados e, seguidamente, o bloco de 4KB é escrito no dispositivo. Esta situação acontece para todos os *datasets* inferiores ao tamanho de um bloco físico que são escritos em *raw* no disco. Em caso de utilização de sistema de ficheiros, com as técnicas de otimização aplicadas pelo sistema operativo, esta situação não acontece.

Com esta atividade mais agitada de leitura e escrita do disco, o consumo energético, tanto do CPU como do disco, acaba por se diferenciar, entre *datasets*. No gráfico da Figura 50, é possível verificar um maior consumo energético do CPU para arbitrar todas as atividades alternadas de IO para os *datasets* inferiores a 4KB. No caso do disco, por haver uma redução de desempenho na quantidade de dados escrita em disco, verifica-se também uma pequena redução no seu consumo energético nesses mesmos *datasets*. Nestas condições, isto é, em modo *raw* e para *datasets* superiores ou iguais a 4KB, a quantidade de dados escritos num minuto atinge 12GB. Valor esse que, comparativamente com as mesmas condições utilizando o sistema de ficheiros, é bastante superior. Em sistema de ficheiros, o valor máximo aconteceu no primeiro ensaio, no *dataset* de 2KB, onde se atingiu a marca de 11GB escritos. Esta situação é aceitável, visto que, empresas como a Microsoft destaca o melhoramento de desempenho do componente quando este funciona em *raw* [183]. Esta solução, é por isso bastante recorrente para aplicar em servidores de bases de dados. Numa análise

que relaciona o desempenho e o custo energético da escrita sequencial, verifica-se que os melhores resultados foram obtidos na escrita sequencial, nestas condições, e nos *datasets* de 4KB e 8KB.

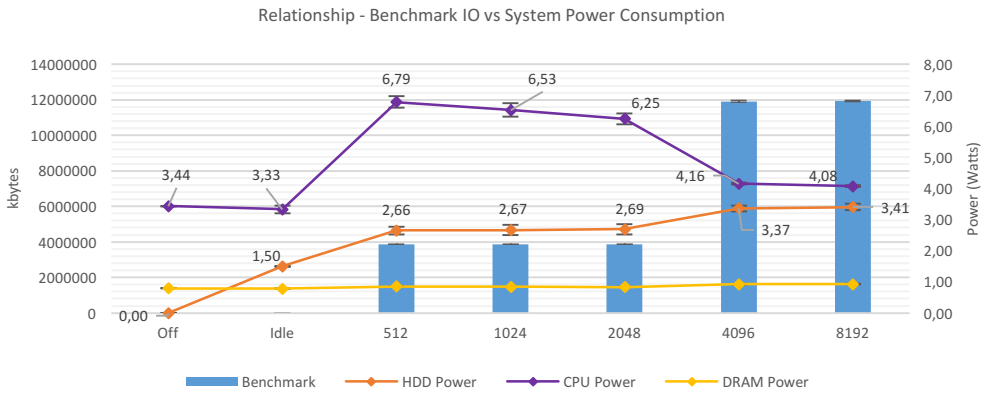


Figura 50 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Numa ultima etapa, analisou-se o desempenho do disco com a **desativação da técnica de *write-caching* com o funcionamento em modo *raw***. O consumo energético por bloco e por operação pode ser consultado no gráfico da Figura 51. Através dele, é possível verificar que, como na utilização de sistema de ficheiros sem a técnica, o custo de operação é bastante estável entre diferentes *datasets*. Isto deve-se ao enorme impacto energético que tem a execução de uma operação sem a técnica de *write-caching* ativa. A quantidade de dados escrita em cada operação tem um impacto reduzido para o custo energético do disco neste exercício. Desta forma, verifica-se um decréscimo proporcional no custo por bloco de 4KB à medida que o *dataset* utilizado aumenta. Esta situação acontece devido ao reduzido impacto que a quantidade de dados escritos por operação tem, no grande impacto energético, por si só da execução de uma operação. Este custo pode-se traduzir ao custo exato da execução de uma operação de escrita em modo *raw* se, esta, fosse instantemente atendida pelo disco. Com a utilização da técnica de *write-caching*, verifica-se sempre uma otimização que faz com que o disco escreva determinados volumes de dados num momento mais oportuno.

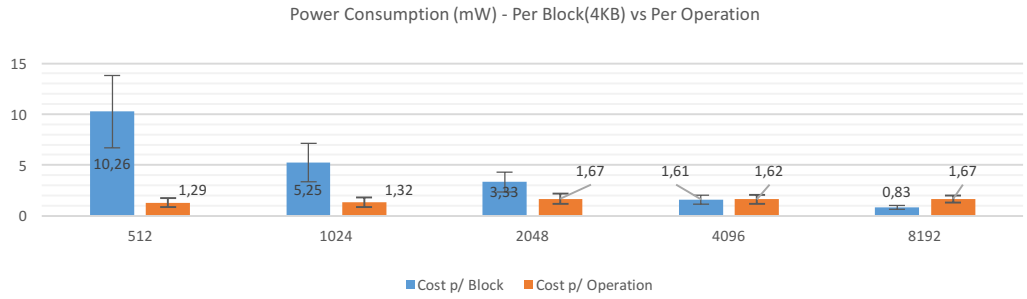


Figura 51 – Ensaio de escrita sequencial ao disco HDD sem a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

No gráfico da Figura 52 é possível consultar o consumo energético dos três principais componentes em relação a atividade da ferramenta de *benchmarking* nos diferentes *datasets*.

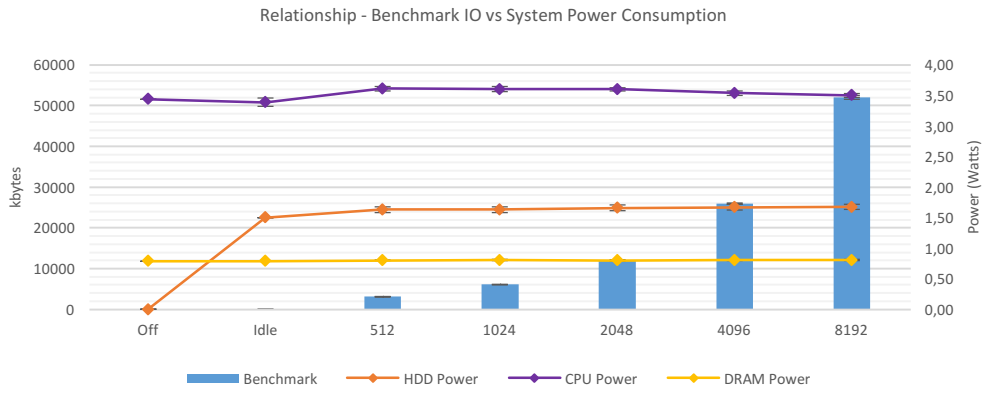


Figura 52 – Ensaio de escrita sequencial ao disco HDD sem a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Neste ensaio verifica-se que, como o numero de ciclos executados por segundo é idêntico entre os *datasets*, a quantidade de bytes escrita num minuto é proporcional aos *datasets* utilizados, isto é, à quantidade de dados escrita em cada ciclo. Como o custo por operação e o numero de operações (ciclos) por segundo são idênticos entre diferentes *datasets*, o consumo de energia do CPU e do disco é pouco variável. O consumo do disco é reduzido devido à pequena quantidade de dados que se está a escrever devido a impossibilidade de otimizar a escrita.

4.3.3.1.2 Disco SSD

Numa primeira fase, optou-se por analisar os resultados dos ensaios referentes à escrita sequencial com as condições mais aproximadas à realidade de um sistema em

funcionamento convencional. Desta forma, **este tipo de escrita sequencial ocorre com recurso à técnica de *write-caching* e utilizando um formato de sistema de ficheiros EXT4**. O gráfico da Figura 53 representa o custo por bloco e por operação do disco SSD em tarefas de escrita sequencial. Nestes cálculos de custo por bloco, verifica-se alguma estabilidade com custos pouco variantes entre diferentes *datasets*. Como tem acontecido noutros ensaios de atividades sequenciais com esta técnica ativa, o custo de operação tem um crescimento proporcional à evolução do *dataset* utilizado. Isto acontece porque, ao contrário das atividades aleatórias em que o maior custo é proporcionado pela *system call lseek*, o impacto energético destas operações varia consoante o volume de dados escrito nelas.

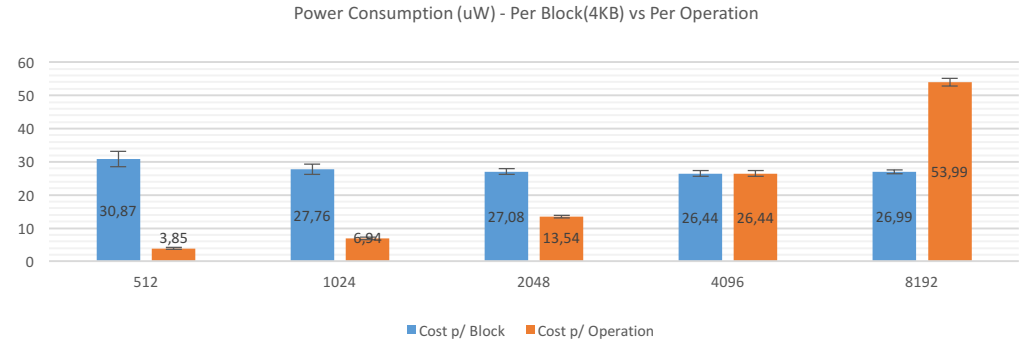


Figura 53 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

O gráfico da Figura 54 apresenta o impacto energético que este ensaio provocou nos três principais subsistemas de energia dinâmica. Nos *datasets* mais curtos, verifica-se um menor volume de dados escrito, no entanto, o custo por bloco para o disco mantém-se estável devido a este componente consumir menos também nestas condições. Apesar disso, o CPU apresenta um consumo mais elevado em *datasets* mais pequenos devido à necessidade de execução de mais operações de escrita nesses *datasets* comparativamente com *datasets* maiores. A memória primária tem um consumo pouco variável que não transpõe a barreira de 1 *watt* durante os exercícios todos realizados. O maior volume de dados escrito acontece nos *datasets* de 4096 e 8192, com valores entre os 30 e 35 GB escritos.

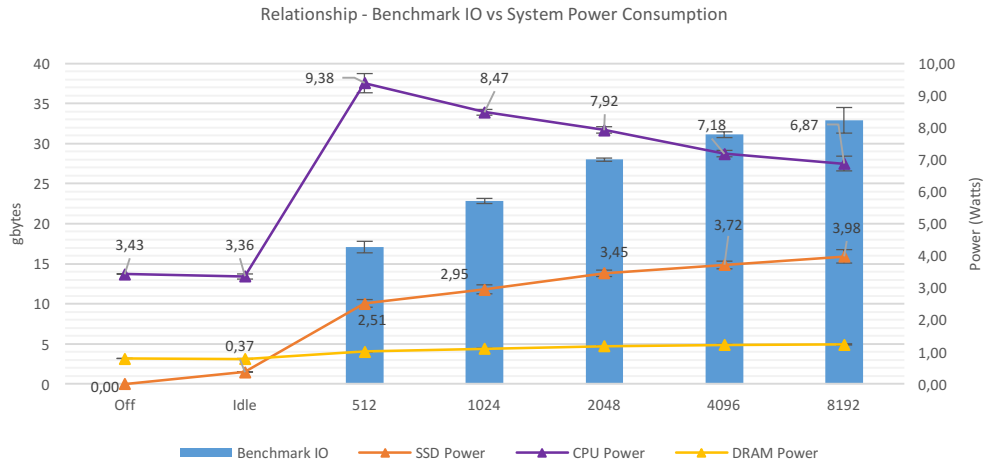


Figura 54 – Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

O desempenho do disco nestas condições é bastante positivo, no entanto, reconhece-se que a técnica de *write-caching* está a otimizar este processo. Desta forma, **para se conhecer melhor o real impacto instantâneo destas operações, desativou-se essa técnica e manteve-se a utilização de sistema de ficheiros**. Nestas condições, verificou-se que o custo por bloco pode ser até 500 vezes superior. Neste caso, o custo por bloco decresce com o aumento do *dataset* utilizado. Isto acontece porque, em *datasets* maiores, o disco realiza menos operações para escrever o mesmo volume de dados. Nesse caso, por ser necessário garantir a escrita dos dados é realizada no momento, cada operação tem um elevado tempo de acesso. Por isso, quando se escreve um maior *dataset* em cada uma das operações, verifica-se uma maior eficiência do disco em cada procedimento de escrita realizado. Pela mesma razão, o custo por operação é idêntico entre todos os *datasets*, visto que, uma operação por si só, tem um elevado impacto energético e encobre o impacto mínimo causado pelos diferentes *datasets* em causa. O gráfico da Figura 55 expõe todos estes resultados referentes ao custo por bloco e operação.

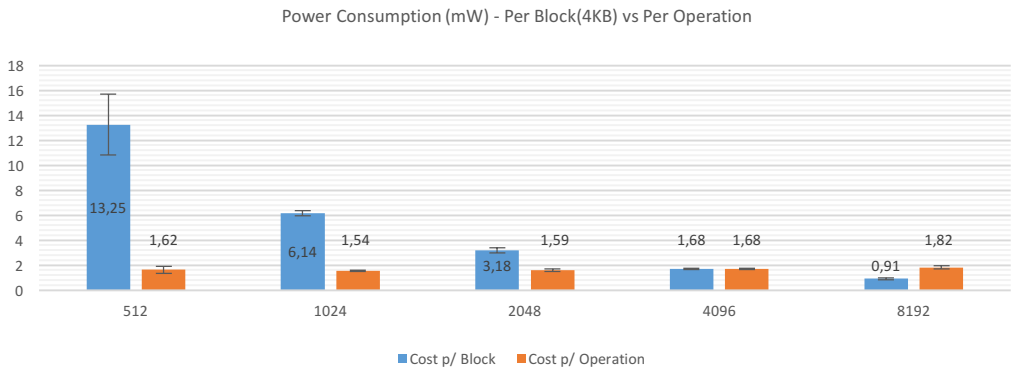


Figura 55 - Ensaio de escrita sequencial ao disco SSD sem a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Por estas razões expostas anteriormente, o volume de dados escrito num minuto é também bastante variável entre os diferentes *datasets*. Como se pode ver no gráfico da Figura 56, no *dataset* de 8192 *bytes*, a ferramenta de benchmarking conseguiu escrever 250MB num minuto. Já o *dataset* de 512 *bytes* não conseguiu transpor os 20MB escritos num minuto de exercício. O impacto energético destes exercícios é estável nos três principais componentes durante os diferentes *datasets*. Com a execução dos exercícios, verificou-se tanto no CPU como no disco SSD um crescimento de 1 *watt* no consumo energético.

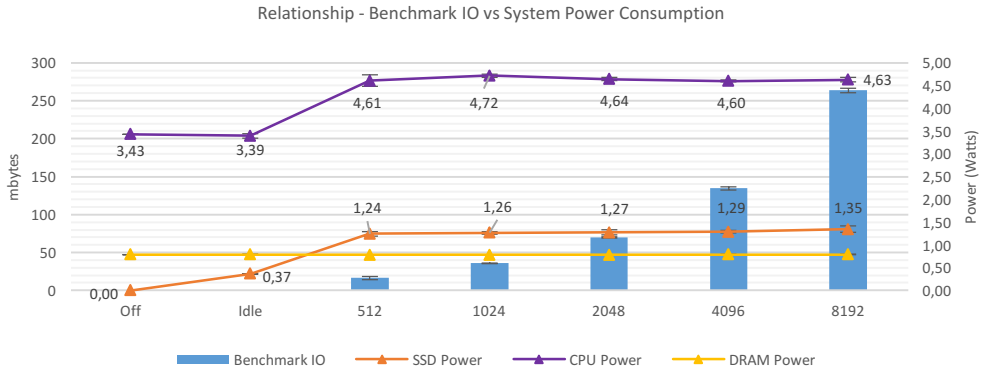


Figura 56 - Ensaio de escrita sequencial ao disco SSD sem a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Numa outra configuração, **analisa-se os resultados obtidos na escrita sequencial sem utilizar sistema de ficheiros e com a técnica de *write-caching* ativa**. Nesta perspetiva, verificam-se reações e resultados do disco um pouco diferentes do ensaio anterior com sistema de ficheiros e a técnica de *write-caching* ativa. O custo por bloco e por operação deste ensaio para os diferentes *datasets* pode ser consultado no gráfico

da Figura 57. No caso dos *datasets* de 4096 e 8192 *bytes*, verifica-se uma eficiência energética um pouco maior comparativamente com os resultados obtidos com o uso de sistema de ficheiros. O custo por operação apresenta também resultados bastante positivos nesses mesmos dois *datasets*.

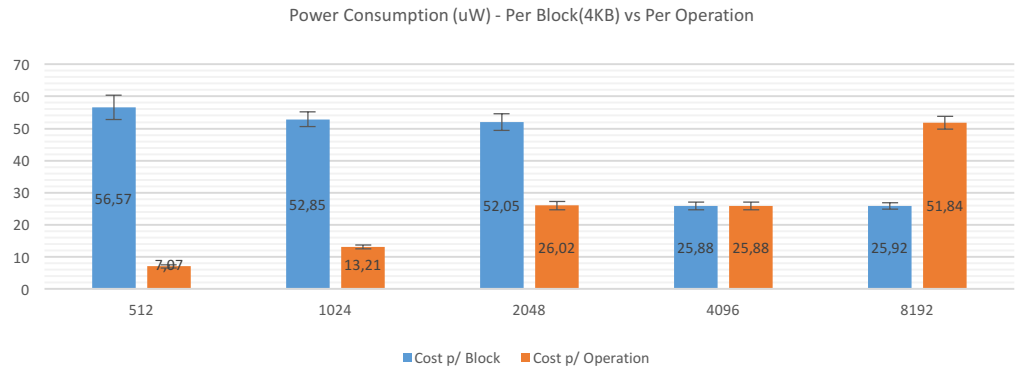


Figura 57 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Nos restantes *datasets*, verifica-se um custo energético por bloco a rondar o dobro do valor obtido nos *datasets* de 4096 e 8192 *bytes*. Isto acontece no disco SSD porque, como aconteceu no disco HDD, existe a necessidade de nestas operações ocorrer o processo de *read-modify-write*. O gráfico da Figura 58 comprova a situação sucedida. Nele, é possível verificar-se um misto de operações de escrita e leitura para os *datasets* inferiores a 4096, quando se deveria apenas verificar operações de escrita.

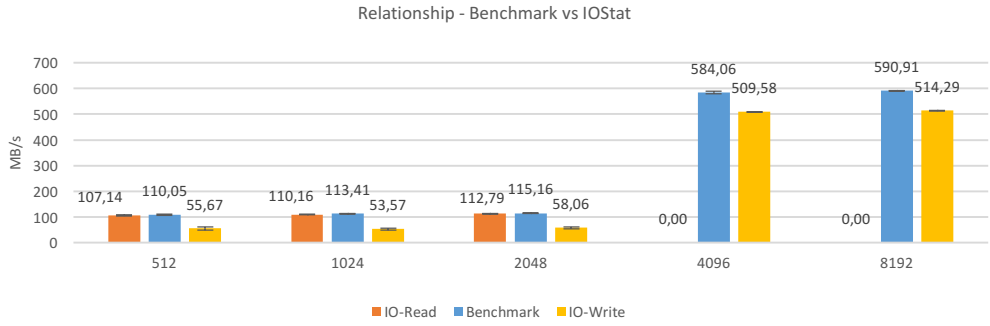


Figura 58 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Logicamente, este processo de leitura e escrita em simultâneo afeta o desempenho do disco, isto é, o volume de dados escrito através da ferramenta de *benchmarking* durante o minuto de exercício. O gráfico da Figura 59 mostra isso, e

relaciona esses dados com a média de consumo energético dos três principais componentes durante um minuto de atividade.

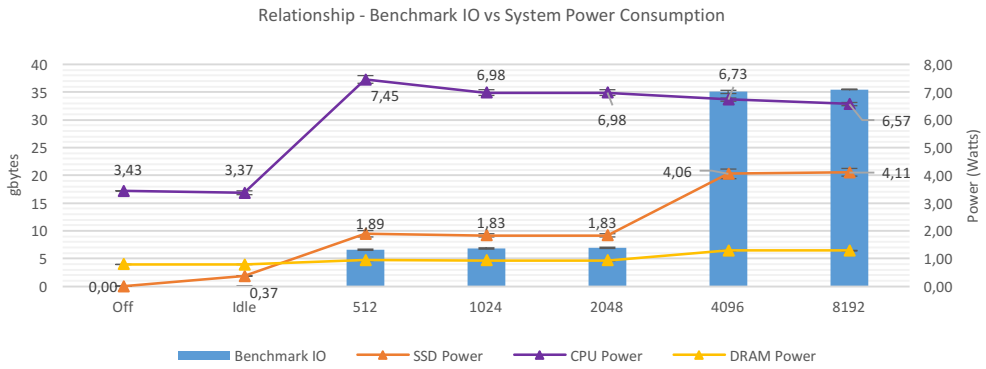


Figura 59 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Nos *datasets* inferiores a 4096, verifica-se um menor consumo por parte do disco que, no entanto, é acompanhado pelo reduzido volume de dados escrito. Nesses mesmos *datasets*, o CPU apresenta um consumo energético mais elevado devido à necessidade do sistema operativo realizar uma gestão de IO mais robusta para resolver os pedidos da ferramenta. A memória RAM apresenta variações reduzidas em que, o maior impacto energético acontece nos *datasets* de 4096 e 8196 *bytes*. Esta situação acontece devido à necessidade de o sistema operativo recorrer à memória primária para administrar o elevado volume de dados para escrita em disco. O gráfico da Figura 60 mostra os níveis de utilização do CPU, memória primária e armazenamento secundário.

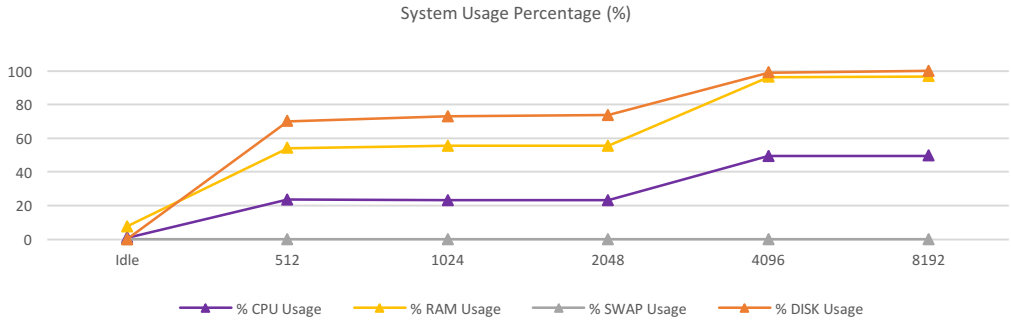


Figura 60 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico dos níveis de utilização de recursos do CPU, memória primária, disco e memória SWAP.

Repetiu-se a análise sobre a escrita sequencial em modo *raw* mas, desta vez, sem recorrer a técnica de *write-caching*. Nestas condições, o custo por bloco chegou a atingir um custo 100 vezes superior. O gráfico da Figura 61 apresenta esses valores em

conjunto com o custo por operação de cada um dos *datasets*. Estes resultados mostram que existe um custo bastante superior na execução destas operações sem recorrer a técnica *write-caching*.

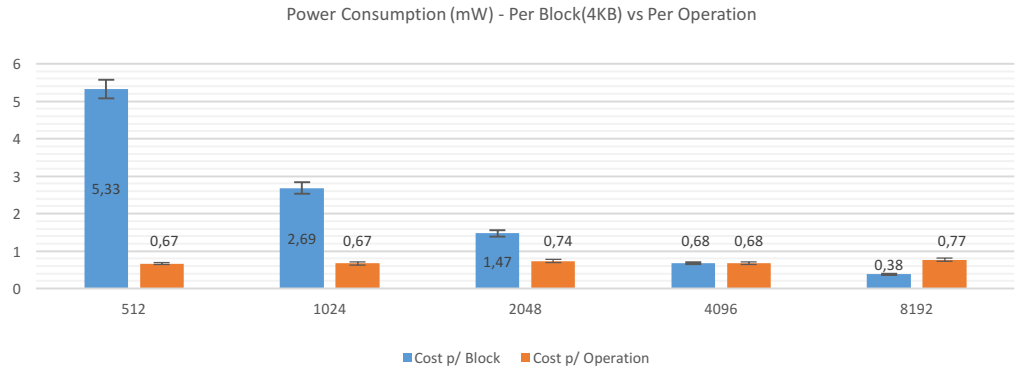


Figura 61 - Ensaio de escrita sequencial ao disco SSD sem a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

O elevado custo energético por bloco e operação é provocado, principalmente, pelo desempenho do disco no processo de escrita. O gráfico da Figura 62 prova essa situação ao expor a relação entre o consumo médio de energia dos três principais componentes e o volume de dados escrito no disco através ferramenta de *benchmarking*. Como é possível verificar, o marco mais alto de escrita em disco acontece no *dataset* de 8192 *bytes*, onde se atingiu uma escrita total de dados úteis em disco de quase 600 MB. Em comparação com o ensaio em que se utilizou a técnica de *write-caching*, verifica-se um desempenho até 55 vezes inferior. Num ponto de vista energético, o consumo do disco foi constante nos 5 diferentes *datasets*, apresentando valores sempre muito próximos de 1,2 *watts*. O CPU e a Memória DRAM também não apresentam variações significativas entre *datasets*.

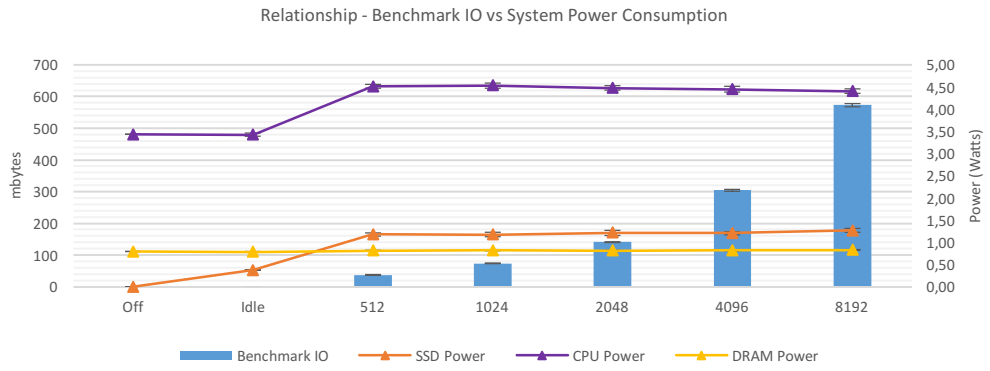


Figura 62 - Ensaio de escrita sequencial ao disco SSD sem a técnica de *write-caching* ativa e em modo *raw*. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes *datasets*.

Com estes dados recolhidos e apresentados, é possível concluir-se que o impacto energético do disco nestas condições é reduzido. No entanto, por haver um baixo desempenho de escrita do disco, o subsistema torna-se ineficiente ao se optar por desativar esta técnica.

4.3.3.2 Modo Aleatório

4.3.3.2.1 Disco HDD

A primeira análise realizada ao disco incide sobre os **exercícios de escrita aleatória que utilizam sistema de ficheiros e a técnica de *write-caching***. Nestes exercícios verificou-se alguma instabilidade de valores nos resultados obtidos. O gráfico da Figura 63 representa o custo energético por bloco e por operação de escrita realizada ao disco. Na analise sobre o custo por bloco do gráfico, verifica-se um instável conjunto de resultados obtidos em cada *dataset* que forma este alargado desvio padrão, principalmente nos *datasets* mais curtos. No caso do custo por operação, os resultados obtidos em cada *dataset* são menos variáveis.

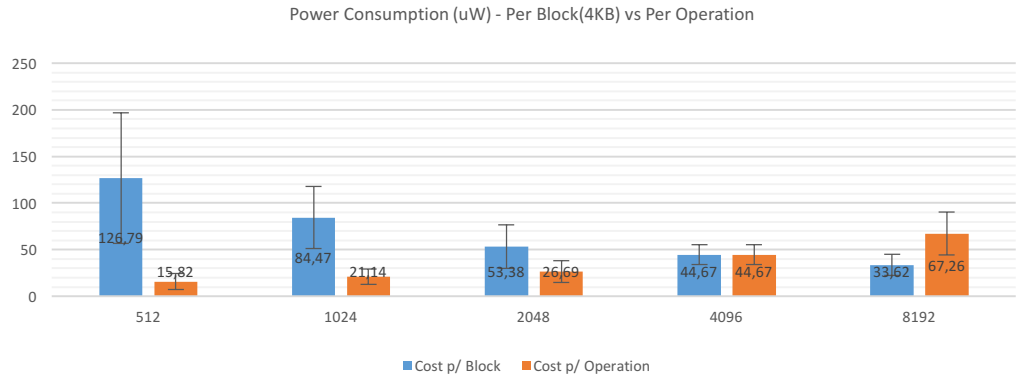


Figura 63 - Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

No ensaio realizado, conclui-se que ambas as métricas calculadas de custo por bloco e operação não são muito esclarecedoras para o desenvolvimento do modelo. Uma das maiores razões para este facto prende-se na ativação da técnica de *write-caching* que, permite que o disco otimize todo o processo e opte por realizar certas operações em momentos mais tardios. No seguimento desta situação, optou-se por explorar melhor outros pontos de referencia capazes de sugerir o consumo energético do disco em exercícios de escrita aleatória, utilizando sistema de ficheiros e recorrendo à técnica de *write-caching*.

O gráfico da Figura 64 apresenta as variações de consumo energético dos três principais componentes do sistema e o numero médio de operações realizadas por segundo. Verifica-se que, o consumo energético do disco é pouco variável entre os diferentes *datasets*. Em relação ao subsistema do processador, verifica-se uma variação de energia reduzida, entre *datasets* que, é provocada por fatores como momentos de espera em operações de IO e pelo numero de ciclos que o subsistema processa por segundo.

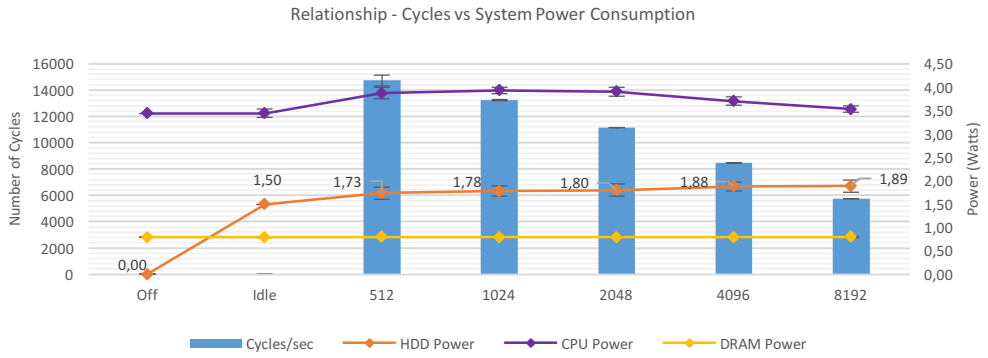


Figura 64 - Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o consumo de energia dos principais subsistemas e o número de ciclos de escrita executados nos diferentes datasets.

Para que seja possível analisar melhor a variação do consumo de energia ocorrente no disco examinado, construiu-se o gráfico da Figura 65. Este gráfico mostra, nos diferentes *datasets*, a diferença potencial entre o disco em *idle* e em atividade, e ainda apresenta o volume médio de dados escrito em cada um dos exercícios executados. Nesta atividade de escrita aleatória, o disco atua com o melhor desempenho possível e apresenta consumos de energia muito idênticos entre *datasets*. Verificou-se ainda que, durante o intervalo de inatividade entre exercícios, o disco reduziu apenas 20% o consumo de energia extra de energia que apresentou na sua atividade. Isto sugere que, o disco, nas condições em que estes ensaios foram realizados, mantem um maior consumo de energia nos dois minutos seguintes ao exercício ter terminado.

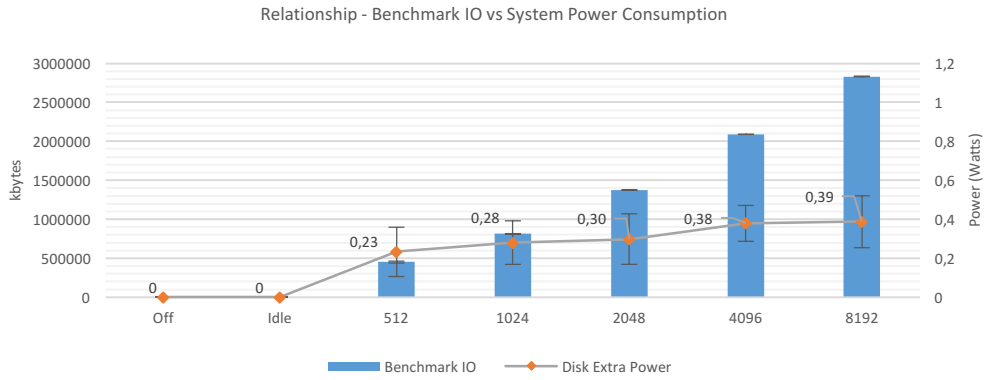


Figura 65 – Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o volume de dados escrito pela ferramenta de benchmark e o consumo extraordinário de energia do disco.

Numa outra etapa, **em que se remove a técnica de *write-caching* e se mantém a utilização do sistema de ficheiros**, é analisada a dependência que o componente tem da técnica para apresentar melhorias de desempenho. Além disso, com esta análise é possível perceber o custo total de uma operação de escrita aleatória, que seja realizada no momento pretendido, utilizando o sistema de ficheiros. A utilização da técnica *write-caching*, acaba por ter um maior impacto nos discos HDD, que necessitam de um intervalo de tempo mais longo para realizar estas operações.

O gráfico da Figura 66 resume o custo real de uma operação e da escrita de um bloco no disco através do sistema de ficheiros. Verifica-se no gráfico que se não se recorrer à técnica de *write-caching*, o consumo energético por bloco pode ser 1000 vezes superior. Este crescimento deve-se principalmente à notável queda de desempenho do disco nestas condições. O gráfico da Figura 67 comprova esta situação. Nele é possível verificar-se que, em comparação com as condições anteriormente revistas, o consumo do disco é praticamente o mesmo em atividade, mas, no entanto, o desempenho de escrita é bastante inferior. Por exemplo, enquanto que no *dataset* de 8192 dos ensaios anteriores era possível atingir a escrita de 1GB, neste caso, não se passa de uns meros 10MB.

Neste mesmo gráfico, é também possível concluir-se que as variações de consumo energético provocadas por estas operações são bastante reduzidas. Isto justifica-se pelo reduzido número de ciclos que o disco executa por segundo nos diferentes *datasets*.

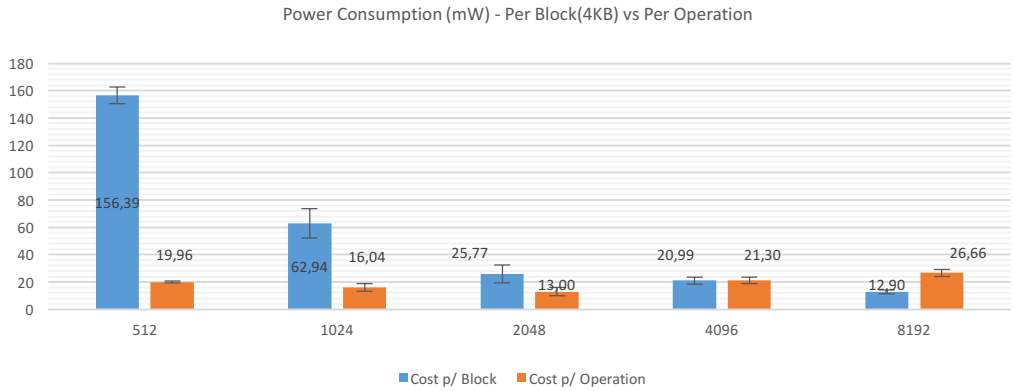


Figura 66 – Ensaio de escrita aleatória ao disco HDD sem a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

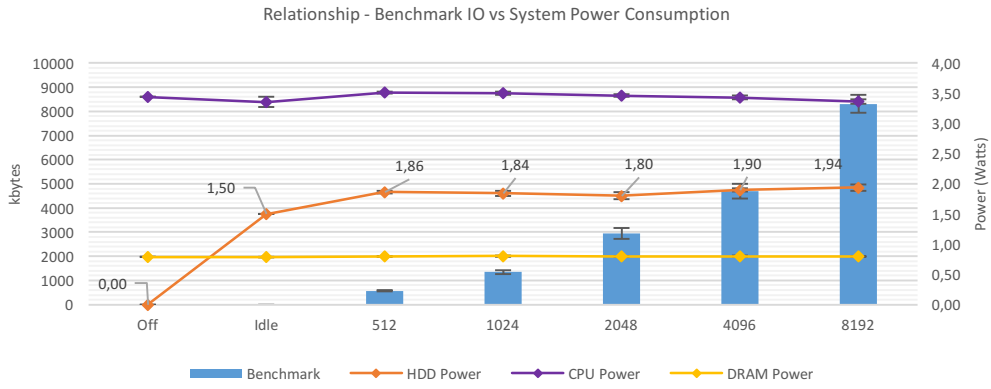


Figura 67 – Ensaio de escrita aleatória ao disco HDD sem a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Além dos ensaios com utilização do sistema de ficheiros, foram executados ensaios sem recorrer a essa opção. Este debate entre o funcionamento do disco sobre dois formatos diferentes permite que se perceba a diferença de desempenho e, por sua vez, de custo energético, entre ambas as opções. Desta forma, explorou-se o desempenho e consumo energético do disco, com o mesmo **a funcionar em modo *raw***. Neste formato, o gráfico da Figura 68 representa o custo por operação e por bloco **recorrendo à técnica *write-caching*** e, o gráfico da Figura 69 representa os mesmos detalhes **sem recorrer à mesma técnica**.

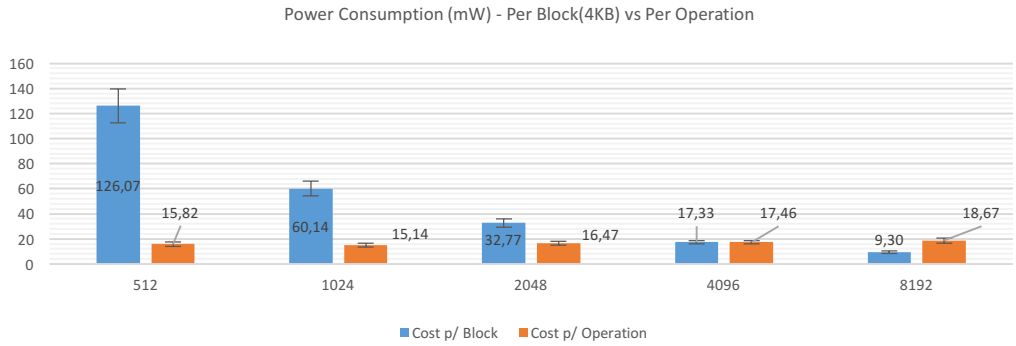


Figura 68 – Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

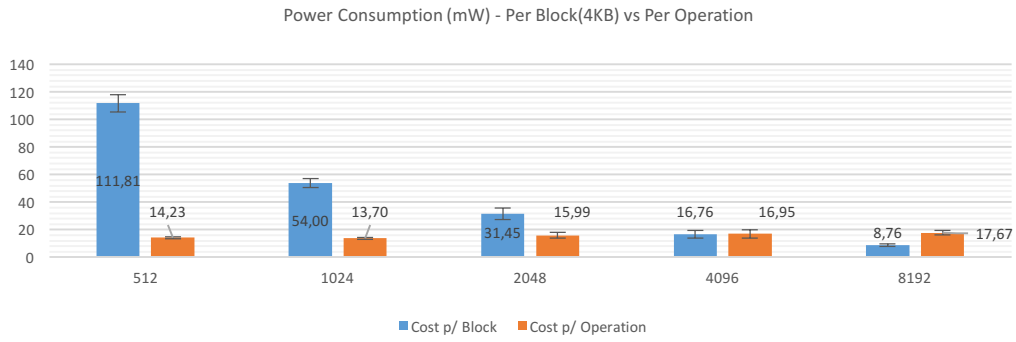


Figura 69 – Ensaio de escrita aleatória ao disco HDD sem a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Numa análise sobre estes dois gráficos, é possível verificar-se que ambos apresentam resultados muito idênticos, sendo que, o ensaio sem recurso à técnica de *write-caching* sugere custos um pouco menores nas suas operações. Esta situação não era a mais espectável, visto que, a técnica de *write-caching* tem mostrado uma notável otimização na escrita em disco, tanto a nível aleatório como sequencial. No entanto, a situação foi explorada com algum cuidado com o objetivo de se perceber o porquê de uma variação tão reduzida entre os dois ensaios de modo *raw*.

Os gráficos das Figura 70 e Figura 71 apresentam detalhes sobre as taxas de transmissão de dados entre o sistema operativo e o disco, divididas em três variáveis, *Benchmark*, IO-*Read* e IO-*Write*. A variável *Benchmark* representa a taxa média de dados úteis escritos pela aplicação em disco por segundo. As variáveis IO-*Read* e IO-*Write* representam as taxas de escrita e leitura, disponibilizadas pela ferramenta IOSTAT, que realmente ocorrem em disco para além dos dados úteis escritos pela aplicação. O gráfico da Figura 70 é referente ao ensaio de escrita em disco em modo

raw com recurso à técnica de *write-caching*. O gráfico da Figura 71 representa o ensaio de escrita em disco em modo *raw* sem recurso à mesma técnica.

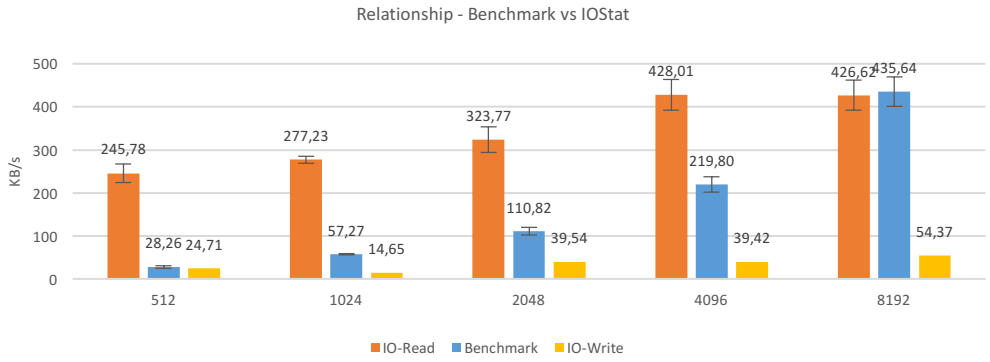


Figura 70 – Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em modo raw. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

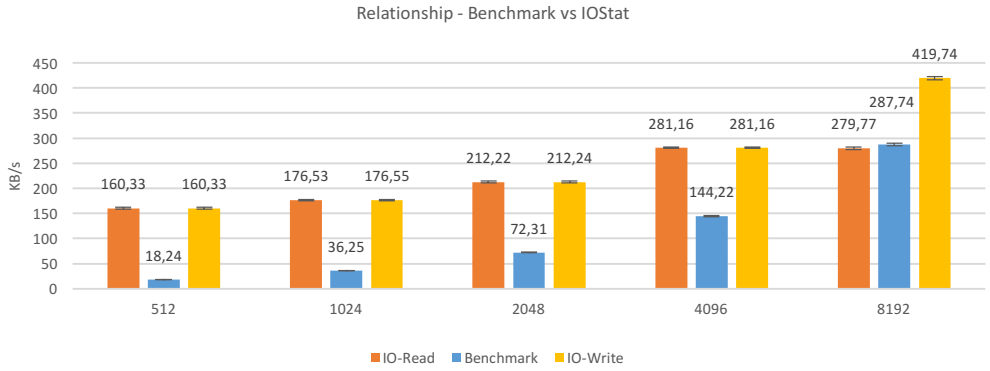


Figura 71 – Ensaio de escrita aleatória ao disco HDD sem a técnica de write-caching ativa e em modo raw. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Em ambos, é possível verificar uma elevada taxa de leitura que afeta todo o processo de escrita em disco. Este acontecimento não surgiu nas situações anteriormente analisadas deste disco referentes a operações de escrita aleatória. Isto é, nas operações de escrita aleatória com a utilização de sistema de ficheiros, a taxa de transferência de operações de leitura é nula. Acredita-se assim que, esta situação, atrase todo o processo de escrita em disco e tenha impacto no desempenho e consumo de energia do mesmo.

Esta agitação de leitura e escrita simultânea em disco acontece quando se escreve em áreas não alinhadas por blocos. Isto porque, nesse caso, é necessário realizar o processo *read-modify-write* em que, como o nome indica, o sistema precisa de ler um ou mais blocos, modificar os dados, e escrever novamente esses blocos

[182]. No caso da escrita para um ficheiro montado num sistema de ficheiros, não é necessário realizar leituras se o ficheiro estiver presente na memória cache da memória primária.

4.3.3.2.2 Disco SSD

A escrita aleatória é geralmente a etapa mais deficiente para um disco, no entanto, pelos resultados obtidos, o disco SSD demonstra uma grande capacidade de responder bem a este processo. O disco apresenta a facilidade de ser totalmente eletrónico e de excluir qualquer componente mecânico da sua composição. É essencialmente esta a razão para o tão bom desempenho do disco nestas condições.

Numa primeira fase, analisou-se o disco SSD com exercícios de escrita em sistema de ficheiros recorrendo à técnica de *write-caching*. O gráfico da Figura 72 apresenta o cálculo do custo por bloco de 4KB e por operação. O custo energético nos *datasets* de 4KB e 8KB é bastante reduzido, atingindo, no máximo, um custo três vezes superior àquele calculado na escrita sequencial com as mesmas condições. Em *datasets* mais curtos, o custo energético por bloco fica maior devido à existência de mais operações *lseek()* por cada 4KB escritos. Por exemplo, no *dataset* de 512, ocorrem 8 *lseeks()* para escrever 4096 *bytes*. O custo por operação é pouco variável entre os *datasets* inferiores a 4KB, visto que, o maior impacto da operação incide sobre o reposicionamento e não sobre o volume de dados escrito. A partir dos *datasets* de 4KB, já existe um maior relevo sobre o tamanho do *dataset* que começa a sobrepor o custo do *lseek()*.

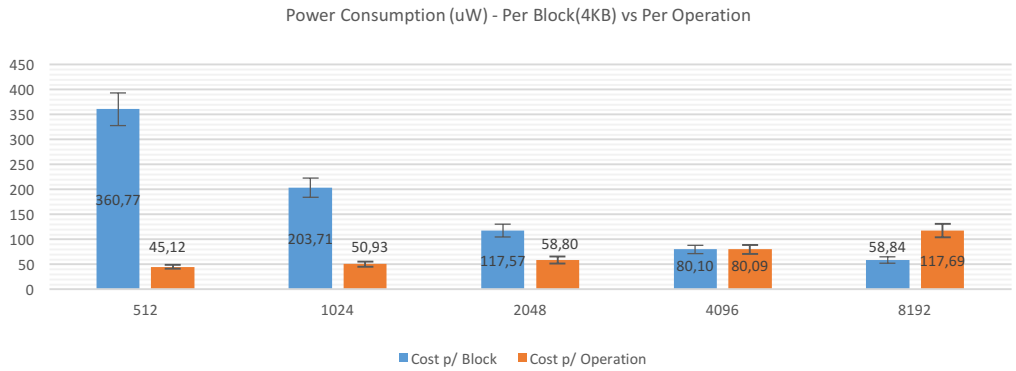


Figura 72 - Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

O volume de dados escrito em conjunto com o consumo energético do disco, tem uma relevância essencial para o calculo do custo por bloco de 4KB. Desta forma, a consulta do gráfico da Figura 73 permite-nos obter uma melhor perceção sobre esses

detalhes. Nele é possível consultar o consumo energético médio dos mais importantes subsistemas nos diferentes *datasets* e relacionar essas variáveis com o volume de dados escrito em cada um deles. Os resultados obtidos constatam um elevado consumo de energia do CPU durante os exercícios e um notável aumento de 200% no consumo da memória primária. Quanto ao disco, verifica-se um modesto crescimento que o leva a estar próximo dos 1,8 *watts* nos vários *datasets*, sem nunca ultrapassar esse valor. Destes valores, aqueles que mais suscitam a atenção são os do subsistema do CPU e memória primária que têm uma variação mais significante comparativamente com os passados ensaios realizados neste disco e no HDD. O volume de dados escrito em cada um dos *datasets* é incrivelmente bom para uma tarefa de escrita aleatória. Apesar disso, estes resultados de escrita acabam por ser um pouco fictícios visto que o disco recorre primeiramente à memória cache para os armazenar temporariamente e, depois, escreve-os em disco num momento mais oportuno.

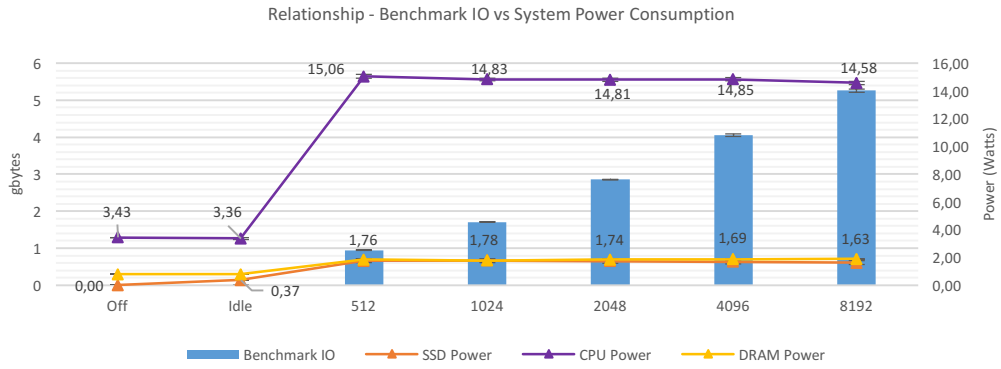


Figura 73 – Ensaio de escrita aleatória ao disco SSD com a técnica de *write-caching* ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes *datasets*.

O CPU e a memória primária têm um papel fundamental neste processo de escrita aleatória com *write-caching* no disco SSD. Como indica no *datasheet* [78] do disco SSD, o componente utiliza parte da memória primária como armazenamento de cache. Com isto, a técnica *write-caching* recorre à memória primária para armazenar temporariamente determinados dados e, dessa forma, aumenta o nível de utilização e de consumo desse mesmo componente. O gráfico Figura 74 descreve os níveis de utilização dos três principais componentes em análise durante os ensaios. Através dele é possível verificar-se que a memória RAM atinge níveis de utilização elevados durante os ensaios por estar a ser ocupada por dados úteis que serão futuramente escritos em disco. O consumo do CPU é também afetado por este processo, visto que, este é o subsistema responsável pela gestão da memória primária. Este facto faz com

que o seu consumo energético atinja valores próximos de 15 *watts* durante os diferentes ensaios.

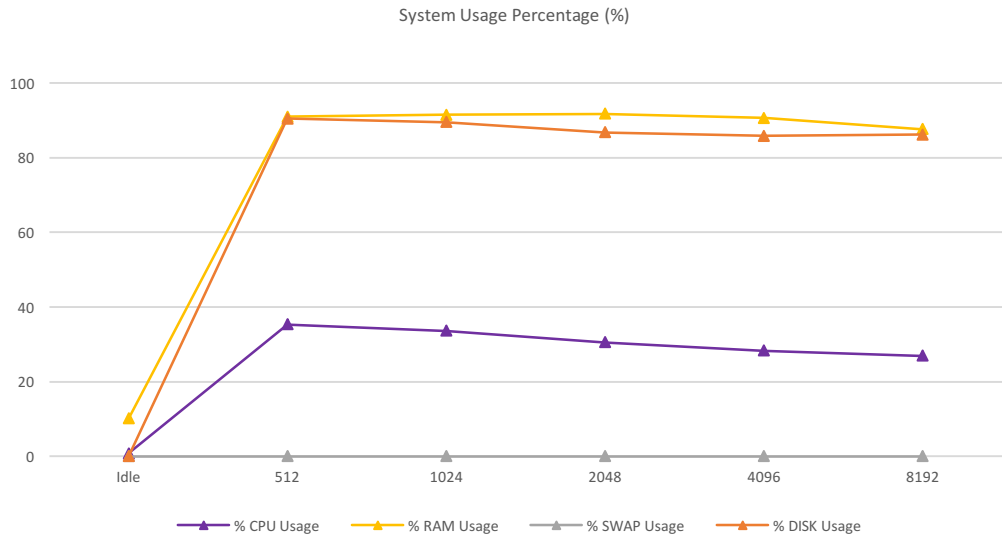


Figura 74 - Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico dos níveis de utilização de recursos do CPU, memória primária, disco e memória SWAP.

Com estes resultados analisados, verifica-se um notável aumento de desempenho de escrita aleatória em sistema de ficheiros com a utilização da técnica *write-caching*. A elevada utilização da memória primária para armazenar temporariamente determinados pedidos é essencial para se obter estes resultados de desempenho.

Numa diferente análise, decidiu-se manter a utilização de sistema de ficheiros e remover a técnica de *write-caching*. Desta forma, consegue ter uma perceção do real custo momentâneo da escrita de determinados conjuntos de dados em disco de forma aleatória. O gráfico da Figura 75 apresenta o custo energético médio por bloco de 4KB e por operação calculado com os resultados obtidos nos diferentes *datasets*. Através destes cálculos, é possível concluir-se que o impacto energético por bloco de 4KB pode ser até 60 vezes superior àquele existente nas mesmas condições com recurso à técnica de *write-caching*. O custo por bloco de 4KB tem tendência a diminuir em *datasets* maiores devido a existir um menor numero de operações de *lseek* por um volume de dados escrito. O custo por operação é idêntico entre os diferentes *datasets*, isto acontece porque, o maior custo da operação é provocado pelo *lseek* e não pela instrução de escrita. As duas variáveis de custo energético calculadas apresentam desvios padrão pequenos, o que se traduz a uma maior estabilidade e fiabilidade nos valores recolhidos.

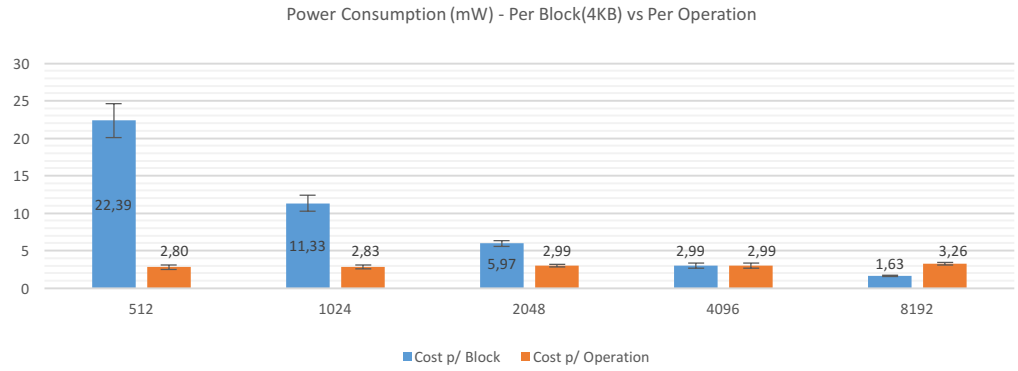


Figura 75 – Ensaio de escrita aleatória ao disco SSD sem a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

O consumo energético dos principais subsistemas está sinalizado no gráfico da Figura 76, em conjunto com o volume de dados escrito, durante os diferentes *datasets* do ensaio. No consumo energético do CPU, é possível verificar-se um crescimento de aproximadamente 3 *watts* do estado *idle* para os vários exercícios executados. O aumento do consumo energético do componente é principalmente justificado pelo serviço que o subsistema presta ao sistema operativo para resolver os pedidos feitos pela aplicação e pelo elevado tempo de espera de IO causado pelo tempo de acesso da escrita aleatória. O consumo de energia médio do disco SSD não ultrapassa os 1,68 *watts* durante os vários exercícios executados. A memória primária, neste caso, não presta qualquer serviço de *cache* e, desta forma, não apresenta variações significantes de consumo energético. Em comparação com as condições anteriormente analisadas, com a técnica de *write-caching* ativa e utilizando o sistema de ficheiros, o desempenho do disco SSD foi bastante reduzido, chegando mesmo a ser 200 vezes inferior no *dataset* de 512 *bytes*.

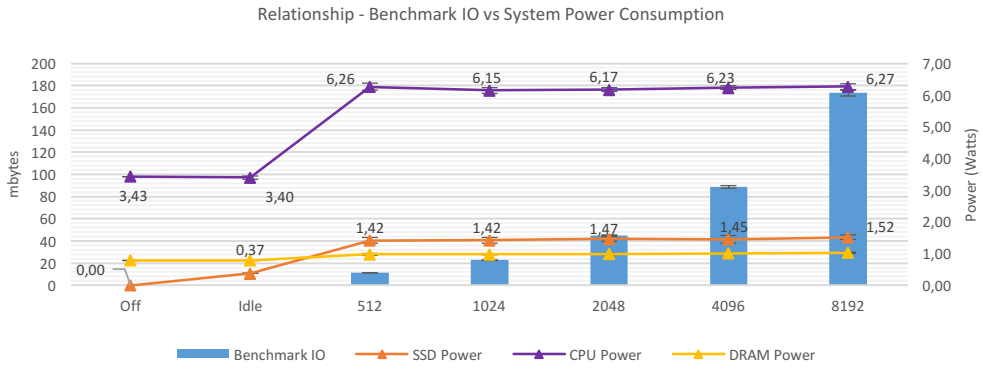


Figura 76 – Ensaio de escrita aleatória ao disco SSD sem a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Como é possível verificar no gráfico da Figura 77, o numero de ciclos executado por segundo é idêntico entre os diferentes *datasets*. Isto deve-se ao elevado tempo de execução de cada operação de escrita aleatória. Desta forma, em *datasets* maiores, existe uma maior oportunidade de escrever um maior volume de dados.

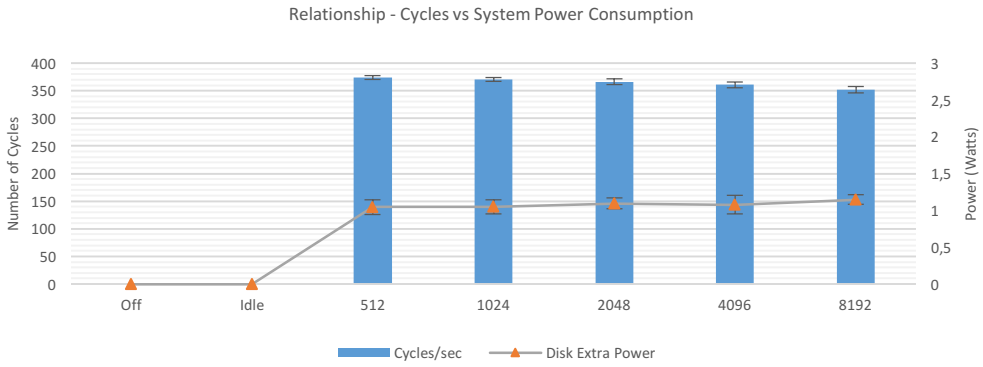


Figura 77 – Ensaio de escrita aleatória ao disco SSD sem a técnica de write-caching ativa e em sistema de ficheiros. Relação entre o número de ciclos de escrever executados por segundo pela ferramenta de benchmark e o consumo extraordinário de energia do disco.

O consumo de energia provocado por este ensaio é pouco variável independentemente do *dataset* em causa. Isto acontece porque, em geral, o volume de dados escrito é bastante pequeno, variando entre 10 e 200 megabytes, e o numero de ciclos é idêntico entre *datasets*.

Numa fase posterior, realizou-se o ensaio de escrita aleatória em *raw* recorrendo à técnica *write-caching*. O gráfico da Figura 78 apresenta o custo energético, por bloco de 4KB e por operação, calculado nos diferentes *datasets* utilizados. Ao contrario do que se passou no ensaio de escrita sequencial, na escrita

aleatória, em qualquer *dataset*, o custo por bloco em modo *raw* é superior àquele obtido utilizando sistema de ficheiros. Como é habitual na escrita aleatória, o custo por bloco decresce proporcionalmente com o crescimento do *dataset*. Já o custo por operação cresce minimamente à medida que o *dataset* aumenta.

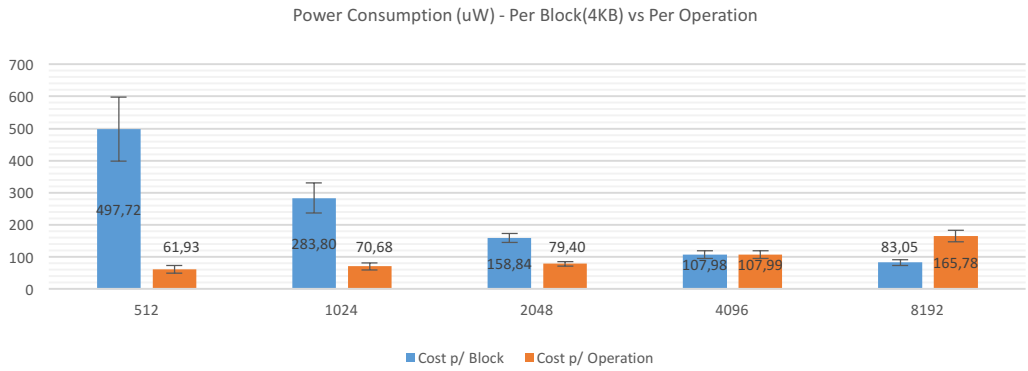


Figura 78 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

A principal razão para o modo *raw* apresentar um custo de escrita superior ao obtido com a utilização de sistema de ficheiros é explicada com os resultados obtidos no gráfico da Figura 79. O gráfico apresenta a média, de um minuto de exercício, da taxa de transferência de leitura e escrita disponibilizada pelo IOSTAT, e da taxa de transferência de escrita de dados úteis da aplicação de benchmarking. Através dos resultados obtidos, é possível verificar-se que estes exercícios provocam a execução do processo de *read-modify-write* no disco [182].

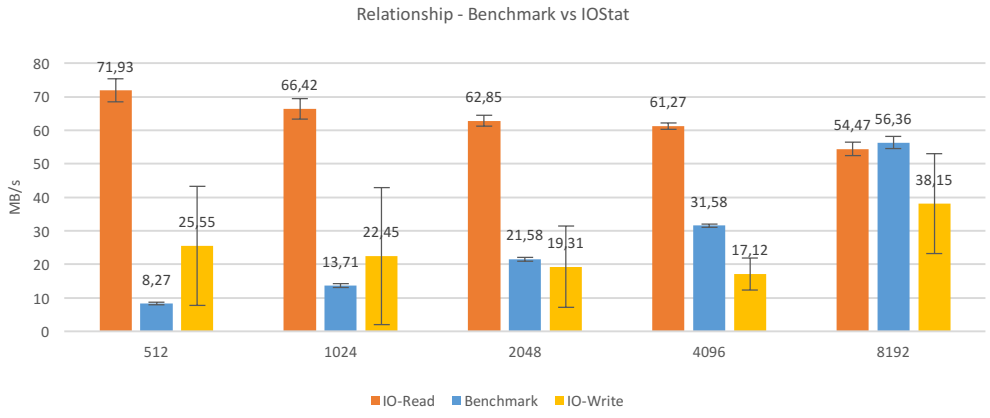


Figura 79 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

Esta situação acontece porque a escrita dos *datasets* é aleatória e existe uma grande probabilidade de falha de alinhamento para com o disco. Quando acontece

essa falha, um bloco lógico pode residir em vários setores físicos. Nesse caso, uma simples escrita de um bloco lógico pode-se transformar numa leitura e escrita de vários setores que não são totalmente preenchidos. A Figura 80 demonstra um exemplo desse acontecimento, quando se pretende escrever um bloco lógico de 4KB desalinhado.

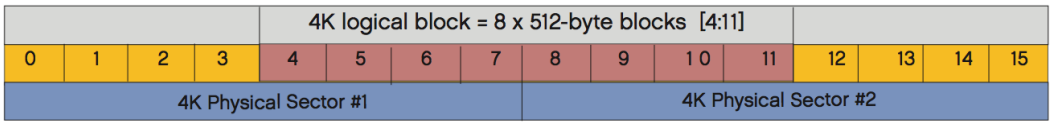


Figura 80 - Exemplo da escrita de um bloco lógico desalinhado com os blocos físicos de um disco [182].

O consumo energético dos três principais componentes é apresentado no gráfico da Figura 81, em conjunto com o volume de dados escrito em cada um dos diferentes *datasets* utilizados. Nestes resultados verifica-se que o consumo energético médio do disco varia entre 1,2 e 1,6 *watts* e o do CPU varia entre 6 e 6,27 *watts*. A memória primária aumenta minimamente o seu consumo, com uma média de 0,6 Watts entre o *idle* e os exercícios. O volume de dados escrito neste ensaio varia entre 500 *megabytes* e 3,5 *gigabytes*.

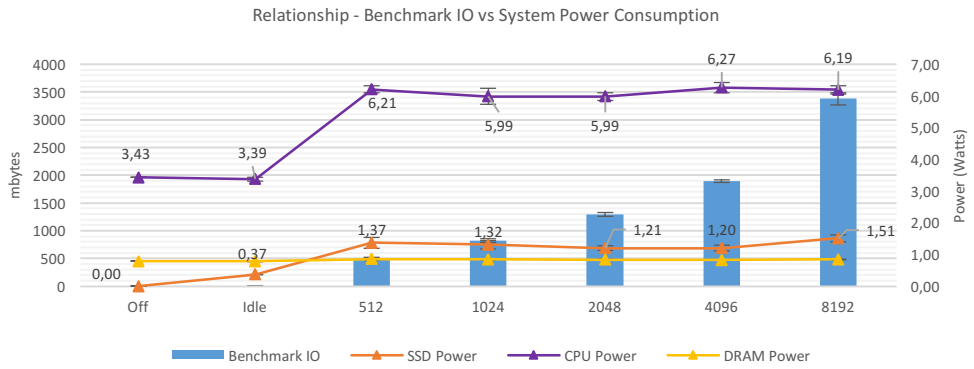


Figura 81 -Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

Comparativamente com o mesmo ensaio utilizando sistema de ficheiros, verifica-se um decréscimo de desempenho causado pelo desalinhamento da escrita de blocos lógicos nos setores físicos. Na utilização de sistema de ficheiros, o desempenho mantém-se a bom nível porque, a ferramenta utilizada para a criação do formato EXT4, encarregou-se de fazer o alinhamento entre o sistema de ficheiros e a estrutura física do disco. Desta forma, os resultados obtidos nessas condições foram bastante superiores, variando entre 1 e 5 *gigabytes* escritos aleatoriamente. O consumo do CPU e da memória primária, neste caso, é inferior devido a não existir uma tão elevada utilização desta memória para operações de cache. O disco, por ter

um menor desempenho em modo *raw*, também apresenta um consumo um pouco mais reduzido.

Numa fase posterior, desativou-se a técnica de *write-caching* na escrita aleatória em modo *raw*. Em comparação com o ensaio com a utilização da técnica em modo *raw*, verificou-se um crescimento no custo por bloco até 16 vezes. O gráfico da Figura 82 apresenta o custo por bloco e por operação deste ensaio. O custo por operação teve valores bastante idênticos entre *datasets* que variam entre 0,77 e 1 *miliwatt*.

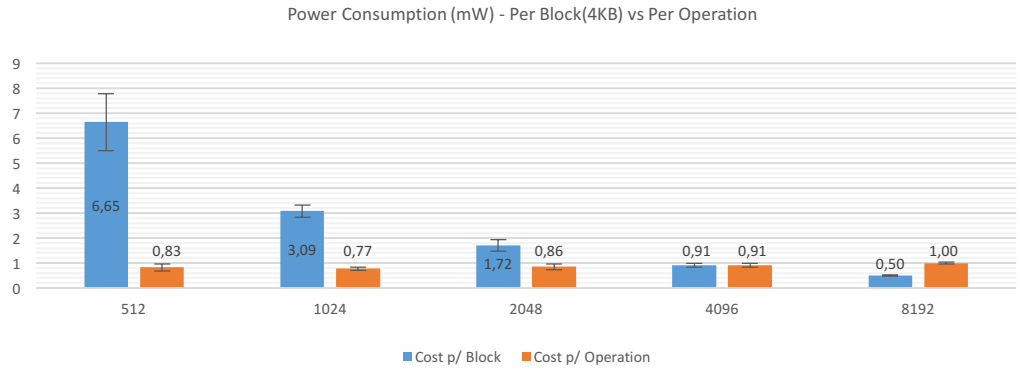


Figura 82 -Ensaio de escrita aleatória ao disco SSD sem a técnica de write-caching ativa e em modo raw. Relação entre o custo por bloco de 4KB e por operação nos diferentes datasets.

Este ensaio é aquele que maior impacto energético apresenta por volume de dados escrito entre todos os ensaios realizados e, isso deve-se a duas diferentes razões. A primeira é referente à existência de desalinhamento dos blocos lógicos na escrita aleatória em modo *raw*. Nesta situação, um bloco lógico ao ser escrito aleatoriamente numa alargada gama de posições, tem uma grande probabilidade de corresponder a um numero superior de blocos físicos. Com isso, e como comprova o gráfico da Figura 83, é necessário que o disco realize o processo de *read-modify-write*. A segunda razão está relacionada com a impossibilidade de utilização de cache nas condições definidas para este ensaio. A não utilização da memória *cache*, impede que o disco armazene previamente os pedidos e os resolva num momento mais oportuno. Desta forma, é garantido que os dados são realmente escritos no momento do pedido e só serão resolvidos outros pedidos assim que estes estiverem seguramente escritos em disco.

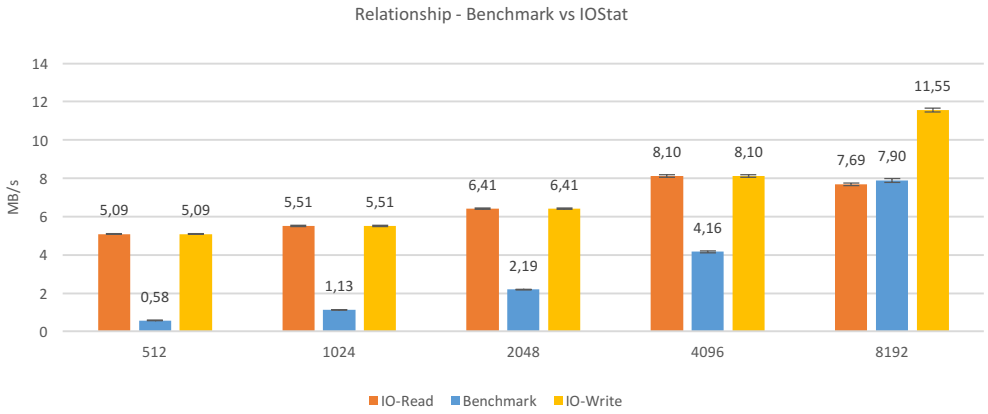


Figura 83 - Ensaio de escrita aleatória ao disco SSD sem a técnica de write-caching ativa e em modo raw. Relação entre os dados de IO da ferramenta de benchmark e da ferramenta IOSTAT.

O consumo energético dos diferentes subsistemas durante os diferentes exercícios realizados foi bastante estável e pouco variável. O gráfico da Figura 84 comprova isso e mostra que, entre o *idle* e a média dos exercícios realizados, o CPU e o disco aumentam aproximadamente 1,5 *watts* e 1 *watt*, respetivamente. A memória primária tem um valor praticamente invariável entre o estado *idle* e os restantes exercícios. Isto deve-se principalmente à não utilização de *cache* neste ensaio.

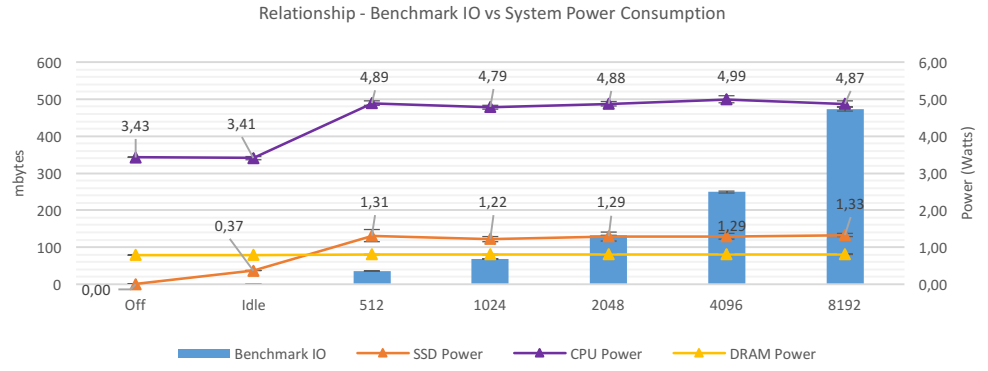


Figura 84 - Ensaio de escrita aleatória ao disco SSD sem a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia dos principais subsistemas e do volume de dados escrito nos diferentes datasets.

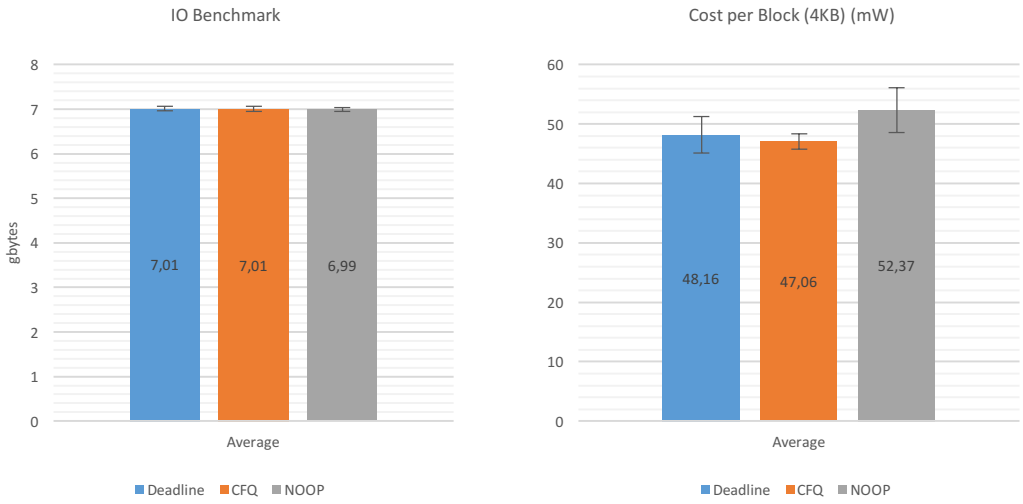
O volume de dados escrito neste ensaio é muito reduzido, variando entre 35 e 475 *megabytes*. Nos ensaios todos realizados a este disco, este é o valor seguramente mais baixo pelos mesmos motivos que o custo por bloco é o mais elevado nestas condições.

4.3.4 Ensaios de IO *Schedulers*

4.3.4.1 Disco HDD

Durante todos os ensaios realizados ao disco HDD, verificou-se que a variação de impacto energético utilizando diferentes IO *Schedulers* é reduzida. No entanto, qualquer otimização a nível energético é importante e, principalmente a larga escala, acaba por fazer de facto uma grande diferença de consumo.

Na leitura sequencial, os resultados dos ensaios realizados foram muito idênticos entre os três IO *Schedulers*. O volume de dados escrito e o custo por bloco foram muito idênticos nos vários *datasets* de cada IO *Scheduler* e, por isso, optou-se por calcular uma média total. O gráfico 1 da Figura 85 apresenta a média do volume de dados lido pelos vários IO *Schedulers*. Verifica-se uma pequena superioridade por parte dos algoritmos *Deadline* e CFQ onde se obteve uma média de 7,01 GB lidos. Como é possível observar no gráfico 2 da Figura 85, o custo energético por bloco de 4KB mais pequeno é conseguido com o algoritmo CFQ e, o algoritmo *NOOP* obteve o pior resultado.



1 2

Figura 85 - Ensaio de leitura sequencial ao disco HDD com a técnica de read-ahead ativa e em modo raw. (1) Média de volume de dados lido nos diferentes IO Schedulers experimentados. (2) Média de custo por bloco de 4KB lido nos diferentes IO Schedulers experimentados.

O consumo energético dos diferentes IO *Schedulers* durante os vários estados de atividade e *datasets* é apresentado no gráfico da Figura 86. Através dele é possível verificar-se que o maior consumo energético acontece utilizando o IO *Scheduler NOOP*. Este facto, juntamente com o pior desempenho no volume de dados lido, faz

deste algoritmo o menos eficiente no exercício de leitura sequencial por uma margem reduzida.

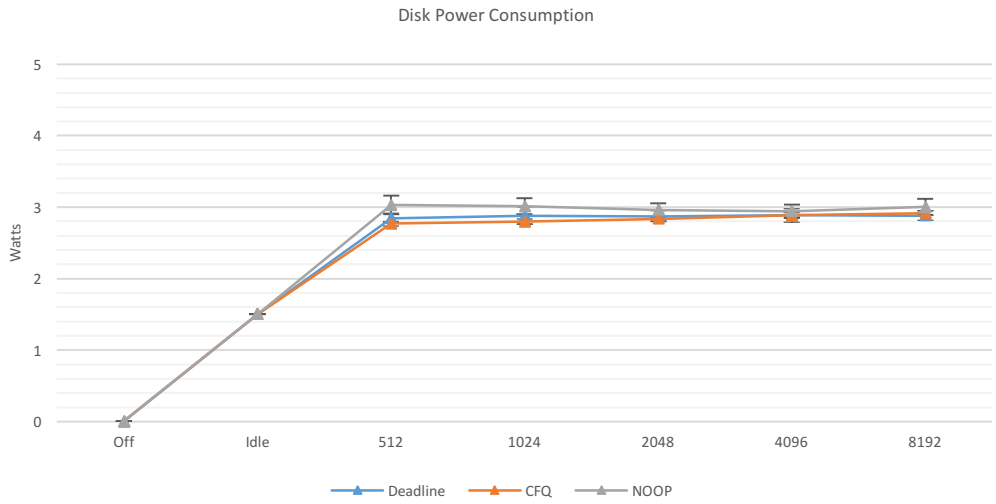
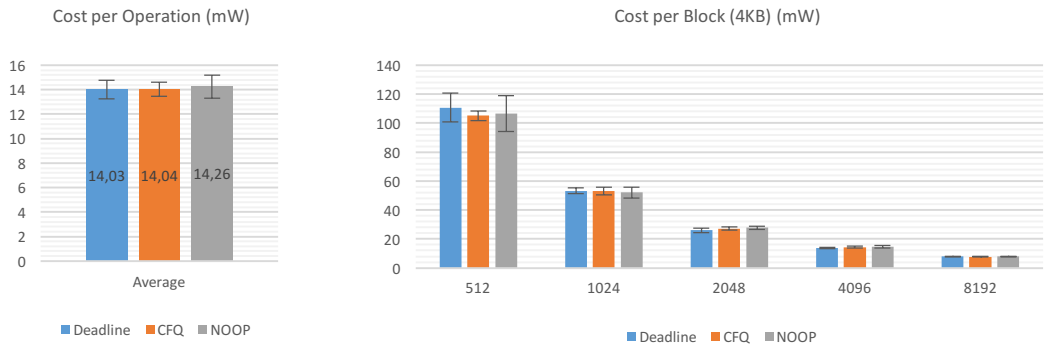


Figura 86 - Ensaio de leitura sequencial ao disco HDD com a técnica de read-ahead ativa e em modo raw. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

Na leitura aleatória, a análise realizada deve priorizar o custo por operação perante o custo por bloco lido. Isto porque, o maior impacto energético deste ensaio é provocado pelo numero operações de posicionamento (*lseek*) e não sobre o volume de dados lido em cada operação. Neste ensaio, o custo por operação foi bastante invariável independentemente do *dataset* em causa. O gráfico 1 da Figura 87 apresenta o custo energético por operação dos diferentes IO *Schedulers* examinados no disco HDD. Os resultados são idênticos, embora o algoritmo *NOOP* apresente um custo minimamente superior, de 0,20 mW.

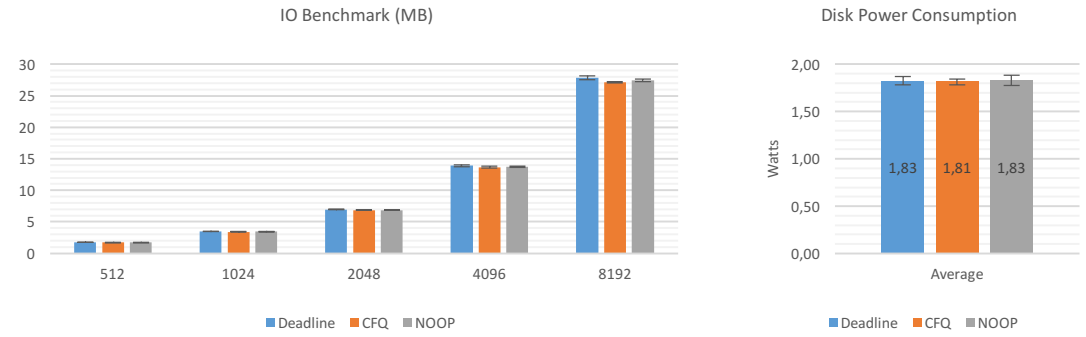


1 2

Figura 87 - Ensaio de leitura aleatória ao disco HDD com a técnica de read-ahead ativa e em modo raw. (1) Média de volume de dados lido nos diferentes IO Schedulers experimentados. (2) Média de custo por bloco de 4KB lido nos diferentes IO Schedulers experimentados.

No gráfico 2 da Figura 87, onde é apresentado custo por bloco é possivel verificar-se que existe uma similaridade de valores entre os diferentes algoritmos. No entanto, o *NOOP* apresenta um maior desvio padrão que sinaliza alguma instabilidade em relação aos outros protocolos.

O grafico 1 da Figura 88 mostra que, durante os diferentes exercicios executados ao disco, o algoritmo *Deadline* apresenteu o melhor desempenho, seguido do algoritmo CFQ. O algoritmo *NOOP* acaba por ter o pior desempenho em todos os exercicios dos diferentes datasets. O gráfico 2 da Figura 88 mostra que o consumo dos IO *Schedulers*, *Deadline* e *NOOP*, é ligeiramente superior ao do CFQ. No entanto, o IO *Scheduler Deadline* foi considerado o protocolo mais eficiente neste ensaio.



1 2

Figura 88 - Ensaio de leitura aleatória ao disco HDD com a técnica de read-ahead ativa e em modo raw. (1) Gráfico da média de volume de dados lido para os três IO Schedulers experimentados nos diferentes datasets. (2) Média do consumo de energia do disco nos ensaios de cada um dos IO Schedulers.

Na escrita sequencial em modo *raw*, o algoritmo *NOOP* apresentou o custo mais elevado por uma notável diferença. O gráfico da Figura 89 comprova a situação e mostra que o algoritmo CFQ foi o mais eficiente a nível energético.

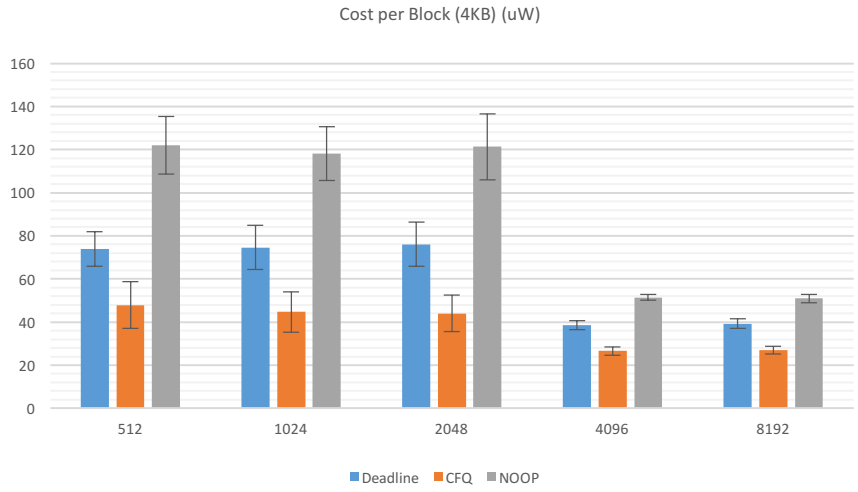


Figura 89 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em modo raw. Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets.

A nível de desempenho de escrita, verificou-se que tanto o algoritmo *Deadline* como CFQ apresentam resultados muito idênticos. O algoritmo *NOOP* tem um rendimento um pouco mais reduzido em comparação com os restantes. O gráfico da Figura 90 mostra todos os resultados de desempenho de escrita durante os diferentes *datasets*.

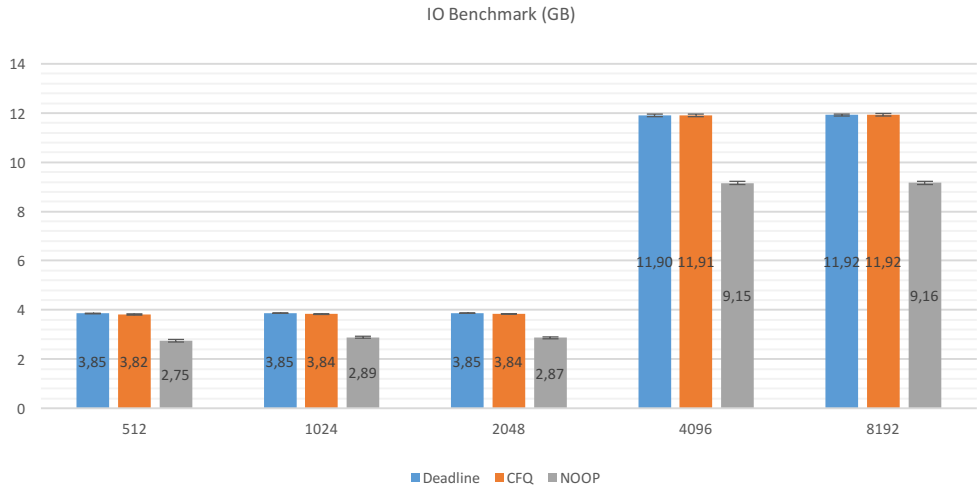


Figura 90 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em modo raw. Gráfico da média de volume de dados escrito pela ferramenta de benchmark para os três IO Schedulers experimentados nos diferentes datasets.

Apesar dos IO *Schedulers Deadline* e CFQ apresentarem um rendimento idêntico no volume de dados escrito, o consumo energético de ambos difere durante esses ensaios e faz com que o CFQ seja o algoritmo mais eficiente. Como é possível ver

no gráfico da Figura 91, no decorrer dos ensaios, o algoritmo CFQ apresenta o consumo energético mais reduzido. O algoritmo *NOOP*, para além do seu menor desempenho, exibe também o maior consumo energético nos ensaios, o que faz deste IO *Scheduler* o menos eficiente neste tipo de operações.

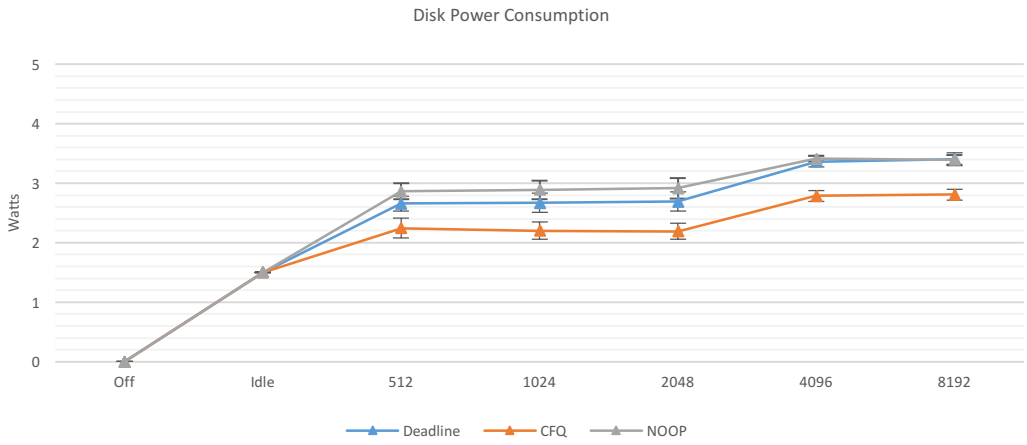
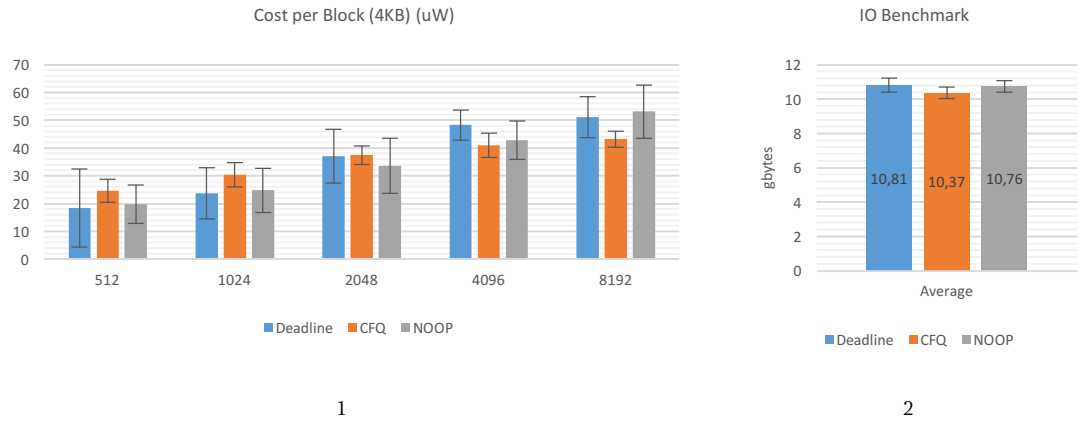


Figura 91 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

Na escrita sequencial utilizando sistema de ficheiros EXT4, verificou-se que em *datasets* superiores ou iguais a 4KB, o algoritmo CFQ foi o mais eficiente. Nos restantes *datasets*, verificou-se uma maior eficiência por parte do algoritmo *Deadline*. O gráfico 1 da Figura 92 comprova esta situação ao apresentar o custo por bloco de 4KB dos diferentes IO *Schedulers* nos vários *datasets* utilizados.



1 2

Figura 92 – Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. (1) Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets. (2) Gráfico da média de volume de dados escrito pela ferramenta de benchmark para os três IO Schedulers experimentados.

O gráfico 2 da Figura 92 apresenta os resultados obtidos sobre o desempenho dos ensaios de escrita nos três IO *Schedulers* examinados. Nele é possível verificar-se

que o melhor desempenho é conseguido pelo algoritmo *Deadline*, com a escrita de 10,81GB durante 1 minuto. Apesar disso, e como comprova o gráfico da Figura 93, esse melhor desempenho não é suficiente para que este seja o mais eficiente. É possível verificar-se pelos dados graficamente expostos que, tanto o algoritmo *Deadline* como *NOOP*, em *datasets* maiores, exibem um consumo energético mais elevado que o CFQ. Num balanço sobre este ensaio, é possível concluir-se que o algoritmo CFQ foi o mais eficiente a nível energético.

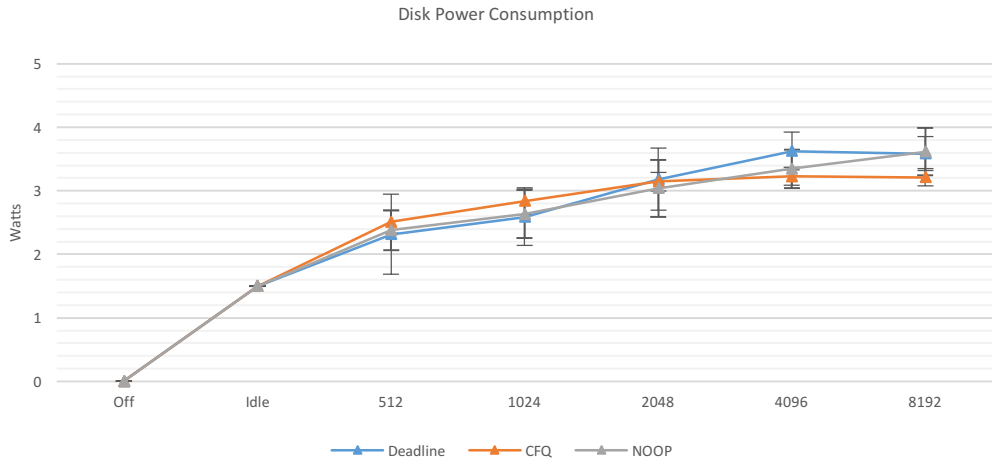
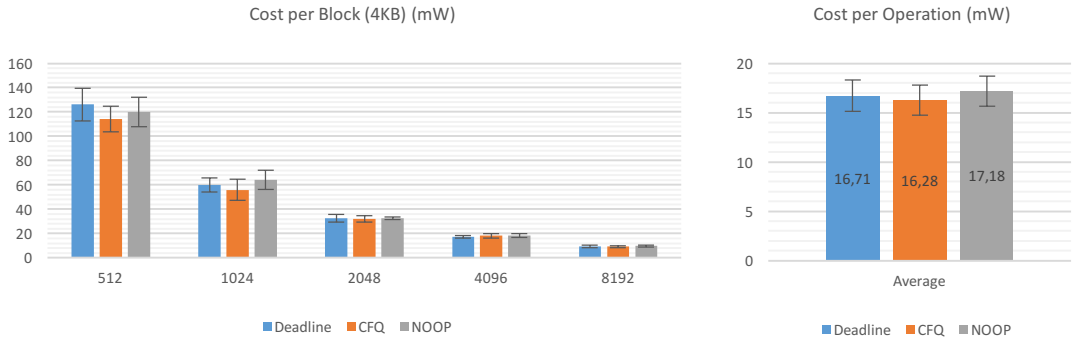


Figura 93 - Ensaio de escrita sequencial ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

Na escrita aleatória em modo raw, a maior referência para se analisar a eficiência energética é o custo por operação. Isto porque, o maior impacto no consumo energético é provocado pelas operações de reposicionamento e não pelo *dataset* de operação. De qualquer forma, o custo por bloco de 4KB pode ser observado no gráfico 1 da Figura 94. Nele, é possível verificar-se um menor impacto energético por parte do algoritmo CFQ em *datasets* mais curtos e, um equilíbrio entre IO *Schedulers* em *datasets* maiores. O gráfico 2 expõe o custo por operação onde é possível verificar-se uma maior eficiência do algoritmo CFQ em comparação com os restantes IO *Schedulers*. Apesar disso, a diferença entre todos eles é bastante reduzida.



1 2

Figura 94 – Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em modo raw. (1) Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets. (2) Gráfico da média do custo por operação de escrita para os três IO Schedulers experimentados.

O volume de dados escrito nos vários IO *Schedulers* e *datasets* é apresentado no gráfico da Figura 95. Novamente, o melhor desempenho de escrita é conseguido pelo algoritmo *Deadline*, sendo que, pelo seu consumo energético mais elevado, acaba por não ser suficiente para que este seja o algoritmo mais eficiente neste ensaio.

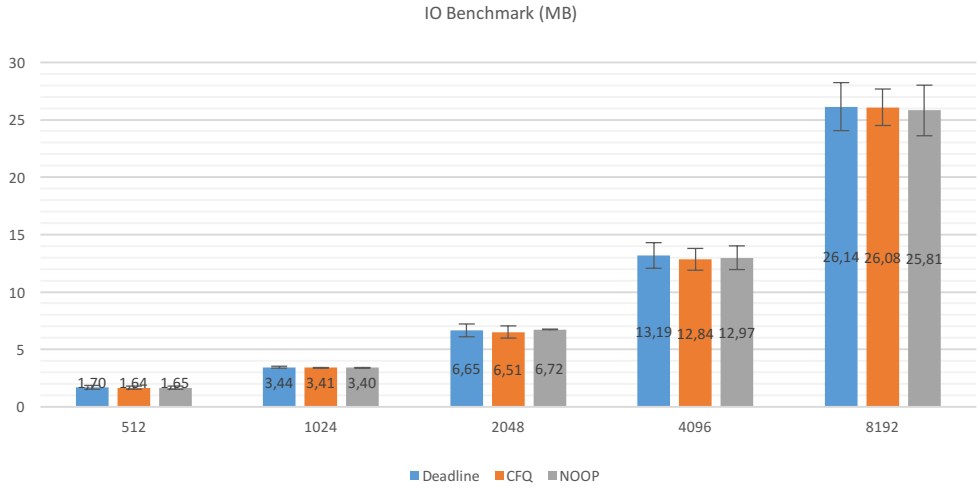


Figura 95 – Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em modo raw. Gráfico da média de volume de dados escrito pela ferramenta de benchmark para os três IO Schedulers experimentados nos diferentes datasets.

Na escrita aleatória em sistema de ficheiros, o calculo do custo por bloco apresentou alguma instabilidade de resultados nos *datasets* mais reduzidos. Isto acontece por haver várias fontes de instabilidade como o volume de dados escrito e o consumo energético do disco nestes *datasets*. Nos *datasets* de 4096 e 8192 bytes, o custo energético por bloco já apresenta valores mais concentrados e próximos dos 40

uW. O gráfico da Figura 96 expõe todos estes resultados referentes ao calculo do custo por bloco.

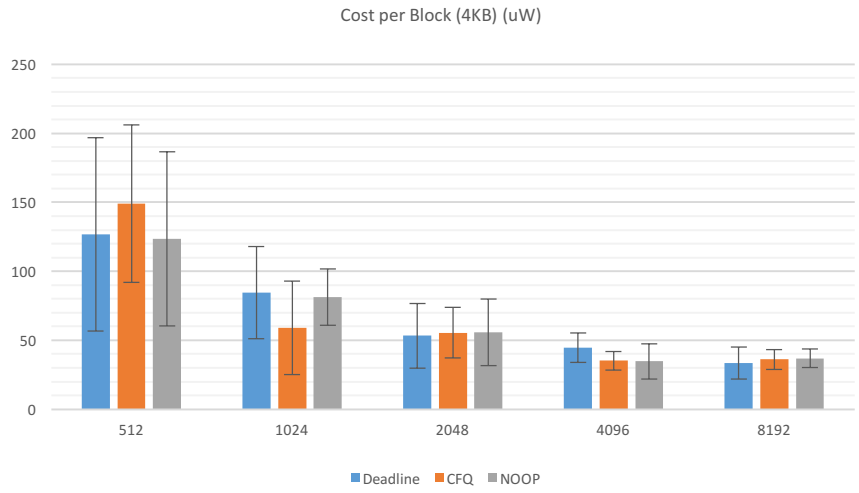


Figura 96 - Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets.

Os resultados de desempenho utilizando diferentes IO *Schedulers* são bastante idênticos verificando-se diferenças insignificantes no rendimento da escrita. O gráfico da Figura 97 comprova esta situação.

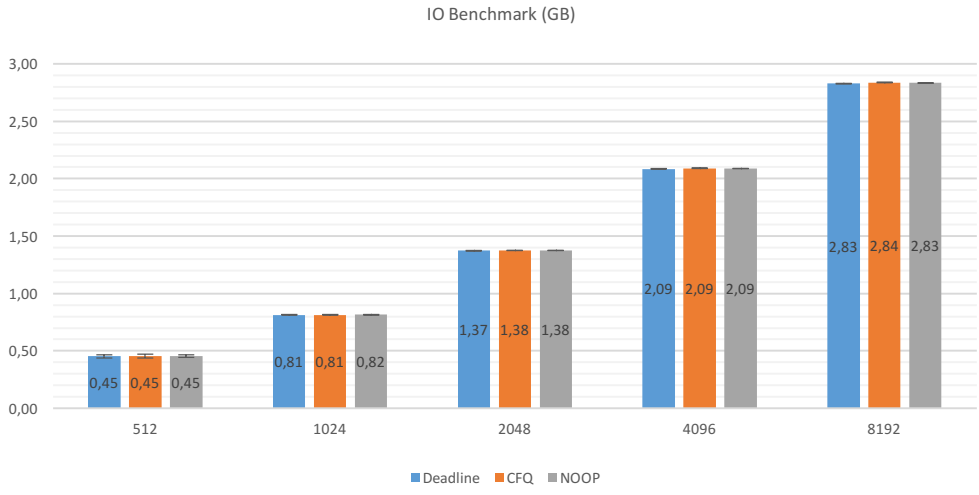


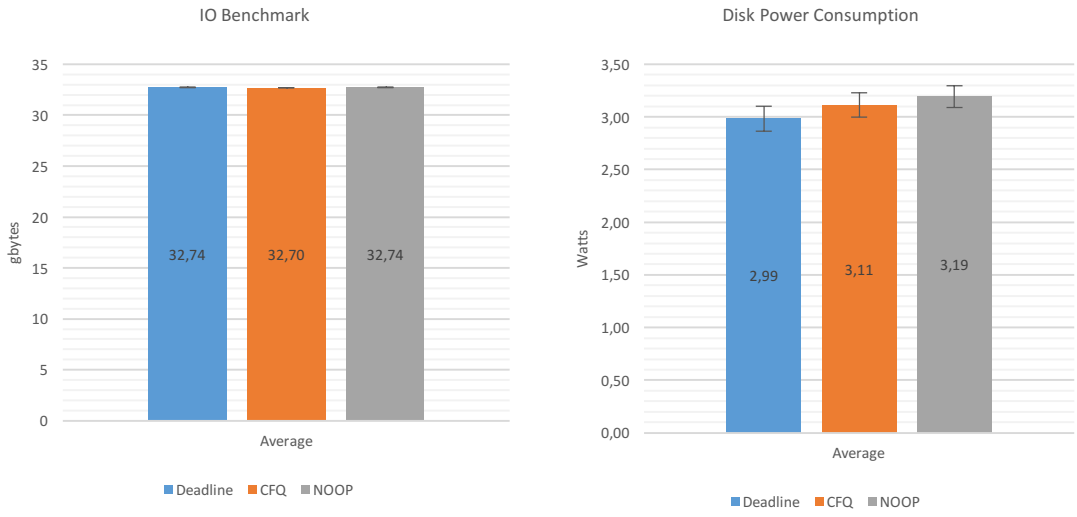
Figura 97 - Ensaio de escrita aleatória ao disco HDD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da média de volume de dados escrito pela ferramenta de benchmark para os três IO Schedulers experimentados nos diferentes datasets.

Com estes resultados analisados, não é possível consagrar algum destes algoritmos como preferido a nível de eficiência energética ou de desempenho. Os resultados de custo por bloco são muito instáveis e inconclusivos em qualquer um dos

datasets. A nível de desempenho, os três IO *Schedulers* apresentam valores muito similares, em cada um dos *datasets* examinado.

4.3.4.2 Disco SSD

No ensaio de leitura sequencial ao disco SSD, os três IO *Schedulers* apresentaram resultados de desempenho muito idênticos. Como é possível visualizar no gráfico 1 da Figura 98, a média da marca atingida esteve muito próxima dos 33GB. Este valor é apresentado numa média única porque com os 5 *datasets* utilizados no ensaio apresentaram um volume de dados com uma variação praticamente nula.



1 2

Figura 98 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa e em modo raw. (1) Gráfico da média de volume de dados lido para os três IO Schedulers experimentados. (2) Média do consumo de energia do disco nos ensaios de cada um dos IO Schedulers.

Como é possível verificar-se no gráfico 2 da Figura 98, o disco apresenta uma média de consumo mais reduzida na utilização do IO *Scheduler Deadline*. No caso de se analisar o impacto energético deste subsistema ao logo dos diferentes *datasets*, verifica-se que o algoritmo *Deadline* apresentou os melhores resultados em todas as frentes. O gráfico da Figura 99 demonstra esta situação.

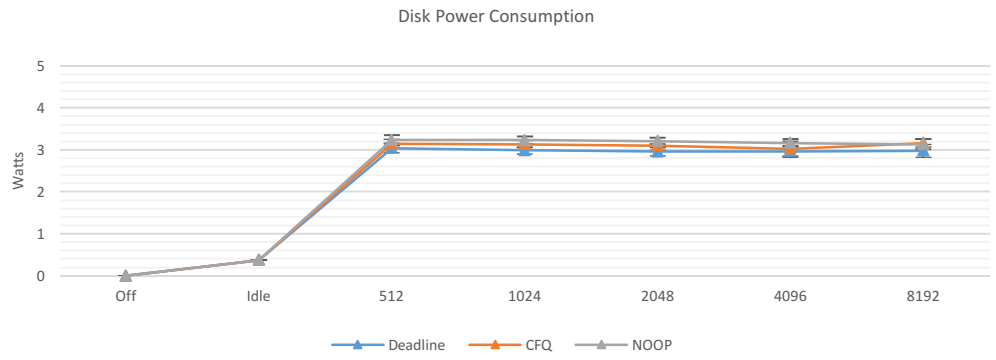


Figura 99 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa e em modo raw. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

Numa pequena análise sobre a resposta dos outros subsistemas a este ensaio, verificou-se que o algoritmo *Deadline* provocou um consumo ligeiramente maior ao nível da memória DRAM. Como se observa no gráfico da_Figura 100, apesar dessa pequena superioridade de impacto energético por parte de um dos algoritmos, todos eles provocaram um aumento de 50% no consumo de energia do subsistema.

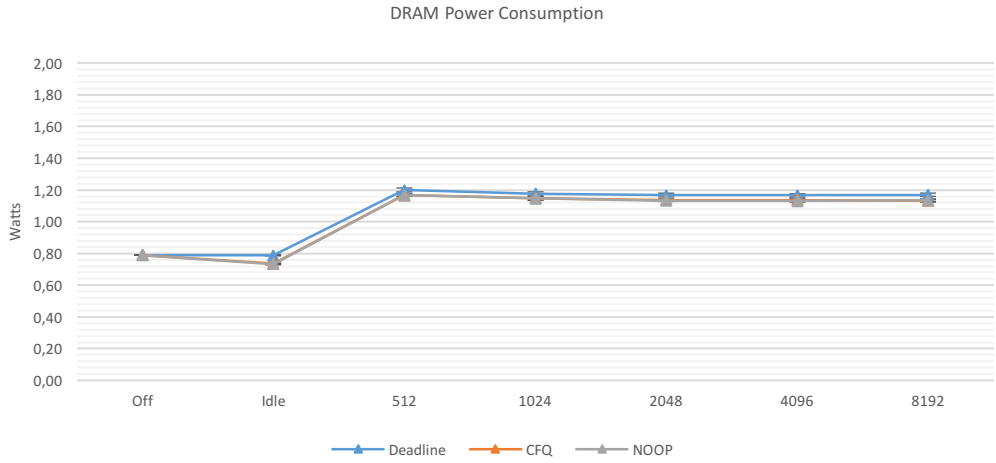


Figura 100 – Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa e em modo raw. Gráfico da evolução do consumo de energia da memória RAM para os três IO Schedulers experimentados nos diferentes datasets.

No caso do CPU, ocorreu um consumo energético muito idêntico entre os três diferentes IO *Schedulers*. É possível verificar-se no gráfico da Figura 101 que, em *datasets* inferiores, o consumo do CPU aumentou até 250%. Sendo que, este impacto acabou por ir reduzindo em *datasets* maiores devido à redução do numero de operações executadas num minuto.

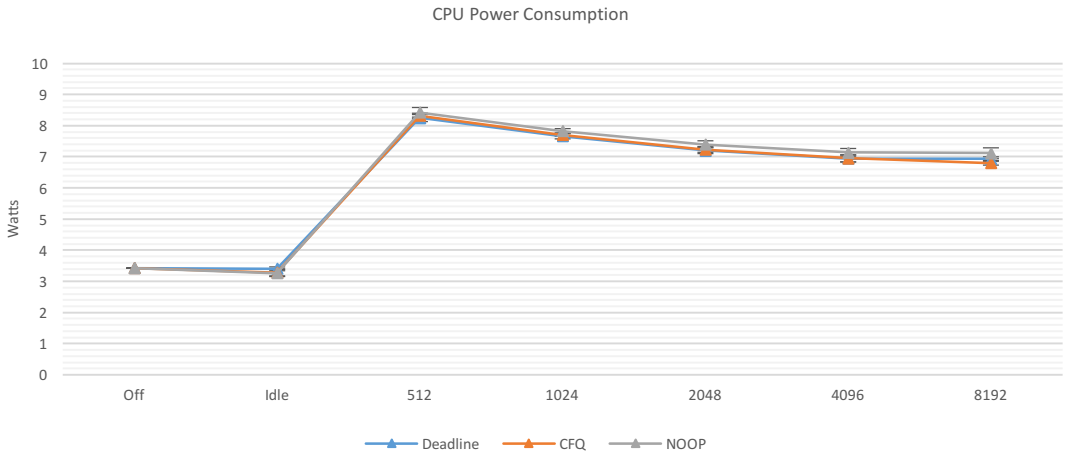


Figura 101 - Ensaio de leitura sequencial ao disco SSD com a técnica de read-ahead ativa e em modo raw. Gráfico da evolução do consumo de energia do CPU para os três IO Schedulers experimentados nos diferentes datasets.

Na leitura aleatória, verificou-se uma diferença destacável de desempenho por parte do IO *Scheduler Deadline*. Através do gráfico 1 da Figura 102 é possível consultar o volume de dados lido nos diferentes *datasets* utilizando os vários IO *Schedulers* e, assim, comprovar o facto constatado. Os algoritmos CFQ e *NOOP* exibem resultados de desempenho que representam à volta de metade daqueles conseguidos pelo *Deadline*.

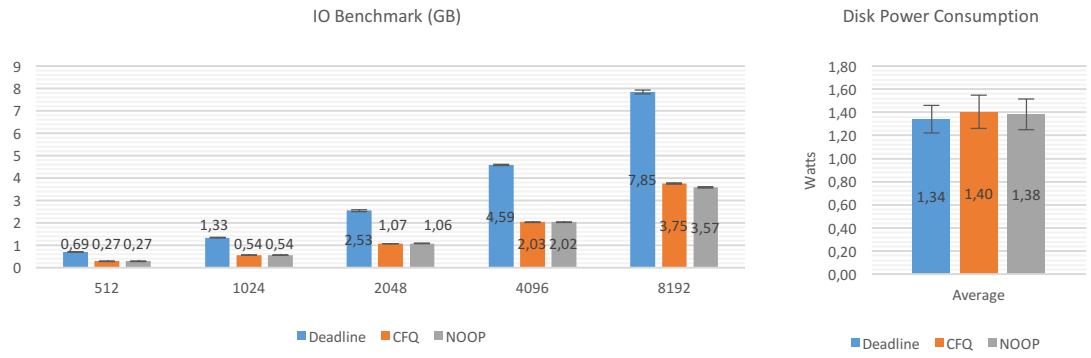


Figura 102 - Ensaio de leitura aleatória ao disco SSD com a técnica de read-ahead ativa e em modo raw. (1) Gráfico da média de volume de dados lido para os três IO Schedulers experimentados nos diferentes datasets. (2) Média do consumo de energia do disco nos ensaios de cada um dos IO Schedulers.

O consumo de energia do disco durante os diferentes *datasets* exercitados foi bastante estável. No entanto, entre IO *Schedulers*, verificou-se um menor consumo por parte do *Deadline*, comparativamente com o algoritmo CFQ e *NOOP* que

apresentaram resultados idênticos. O gráfico 2 Figura 102 expõe essa informação de forma mais detalhada.

Com estes resultados obtidos, é fácil de se concluir que existe uma notável superioridade de eficiência energética por parte do IO *Scheduler Deadline* em relação aos restantes. No entanto, o gráfico confirma essa expetativa e mostra o custo energético por bloco lido nos diferentes *datasets*. Em todos os *datasets* exercitados é possível verifica-se uma maior eficiência energética do IO *Scheduler Deadline*.

Neste tipo de leitura, ao ser aleatório, existe um maior peso energético sobre cada operação realizada do que propriamente sobre o tipo de *dataset* utilizado. Desta forma, existe alguma relevância em se analisar o custo por operação que está apresentado no gráfico da Figura 103. Logicamente, o IO *Scheduler Deadline* é o que apresenta os resultados mais positivos, consumindo um valor entre 40 e 60 uW para os vários *datasets* exercitados.

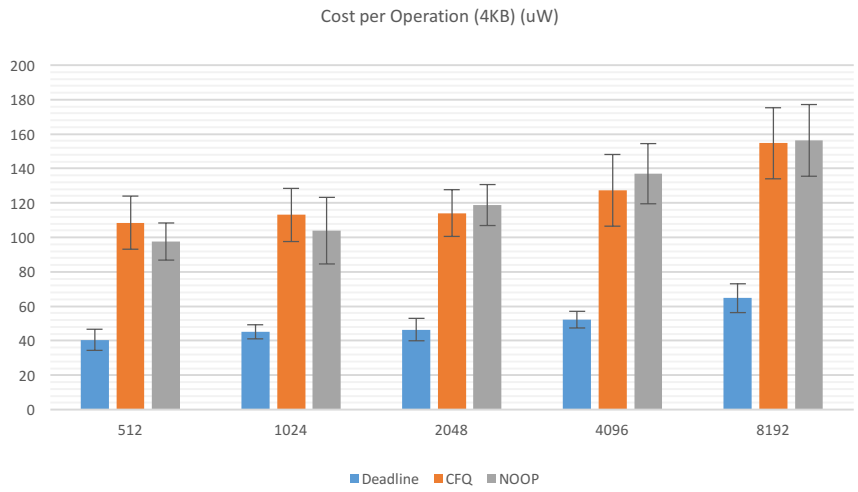
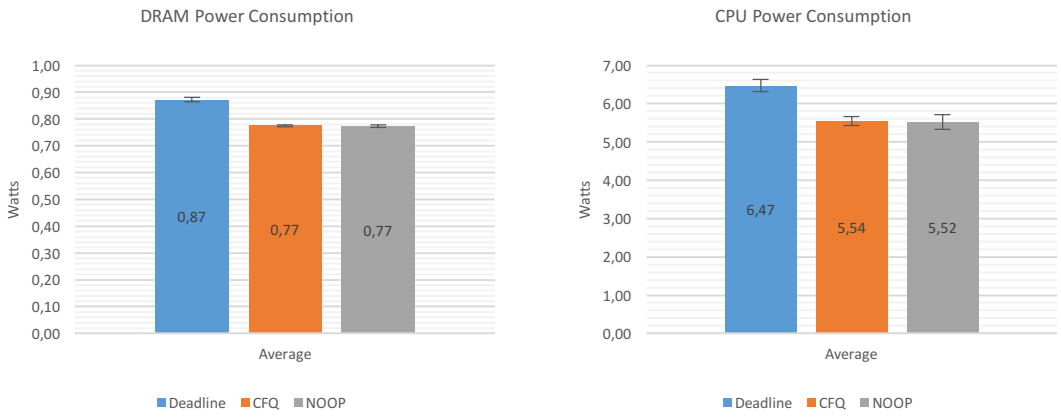


Figura 103 - Ensaio de leitura aleatória ao disco SSD com a técnica de read-ahead ativa e em modo raw. Média de custo por operação de leitura para os três IO Schedulers nos diferentes datasets.

Apesar de se verificar resultados bastante positivos por parte do IO *Scheduler Deadline* durante este ensaio, este provou um crescimento do consumo energético dos restantes subsistemas com maior atividade dinâmica. Como é possível verificar-se no gráfico 1 e 2 da Figura 104, os consumos de energia, tanto da memória RAM como do CPU, têm valores bastante superiores com este maior desempenho do disco SSD.



1 2

Figura 104 - Ensaio de leitura aleatória ao disco SSD com a técnica de read-ahead ativa e em modo raw. (1) Média do consumo de energia da memória RAM nos ensaios de cada um dos IO Schedulers. (2) Média do consumo de energia do CPU nos ensaios de cada um dos IO Schedulers.

Na escrita sequencial em raw, destaca-se novamente o desempenho do IO *Scheduler Deadline* pela positiva. Pela negativa, destacou-se o IO *Scheduler* CFQ, ao apresentar os piores resultados de desempenho entre os três IO *Schedulers*. O gráfico da Figura 105 comprova esta situação. Como explicado anteriormente, os três *datasets* mais curtos apresentam valores bastante inferiores de desempenho devido a estarem a escrever blocos lógicos de tamanho inferior aos blocos físicos.

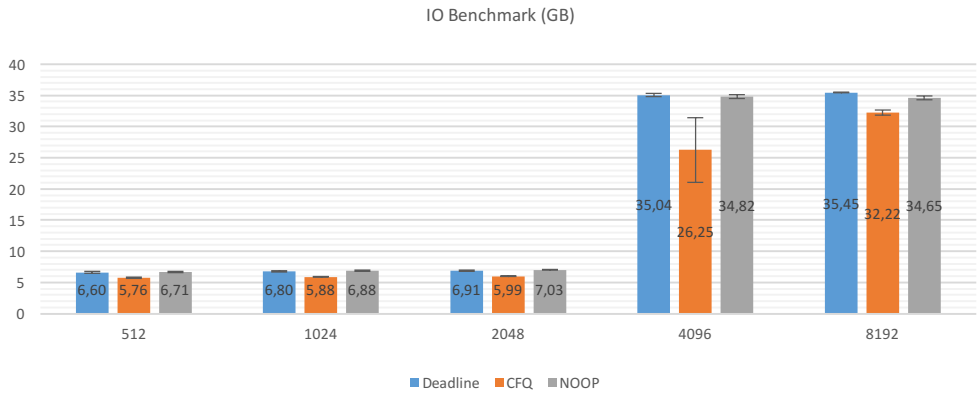


Figura 105 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da média de volume de dados escrito para os três IO Schedulers experimentados nos diferentes datasets.

A melhor forma de se perceber a eficiência energética dos diferentes protocolos é com base no custo por bloco. O gráfico da Figura 106 mostra esses cálculos realizados e, através dele, é possível confirmar-se que o IO *Scheduler Deadline* apresenta os melhores resultados de eficiência energética, seguido do CFQ e do *NOOP*. O algoritmo CFQ foi o menos eficiente neste ensaio.

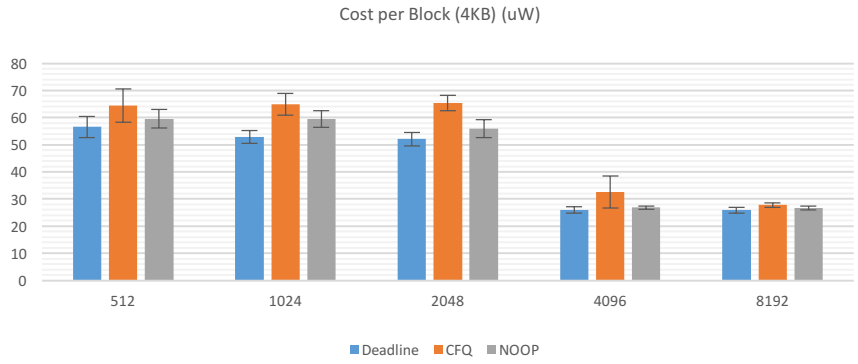


Figura 106 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets.

A eficiência energética do disco é calculada com base no consumo de energia do disco e do volume de dados por ele escrito. Neste gráfico, presente na Figura 107 é possível analisar-se o consumo do disco nos vários *datasets* exercitados. Com os dados calculados, verifica-se que o aumento de desempenho do disco nos dois maiores *datasets*, reflete-se também no consumo do mesmo. O consumo de energia do disco apresenta os valores mais reduzidos no algoritmo *Deadline* e CFQ.

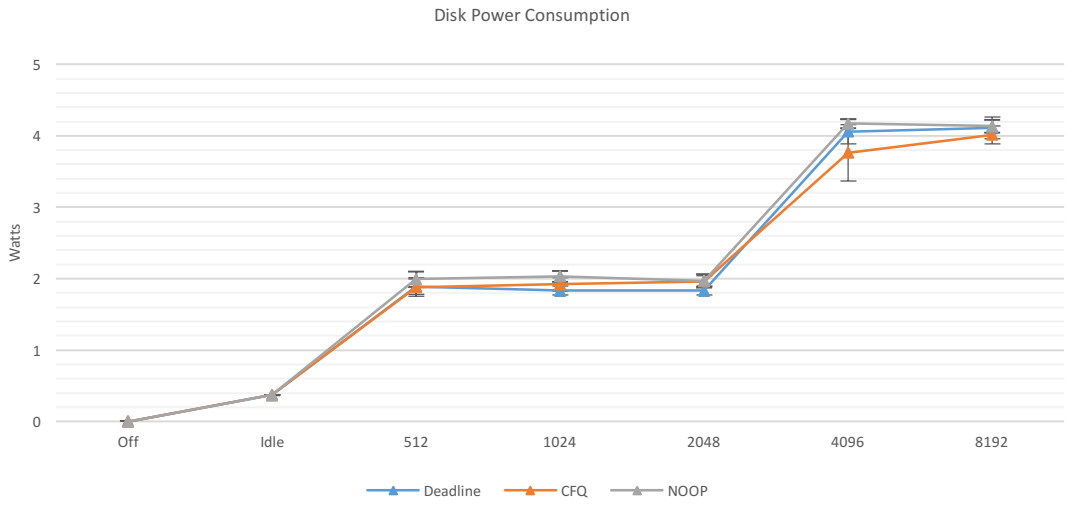


Figura 107 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

A memória RAM também sofreu um pequeno crescimento de potência aquando do exercício dos *datasets* 4096 e 8192, como se pode observar no gráfico da Figura 108. O consumo de energia do subsistema, que nos primeiros três *datasets* já tinha aumentado 0,1 Watts, aumentou ainda mais 0,4 Watts nos dois maiores *datasets*. A nível do impacto energético dos diferentes IO *Schedulers*, verifica-se um maior

consumo por parte do algoritmo *Deadline* que acaba por ser compensado pelo seu notável desempenho.

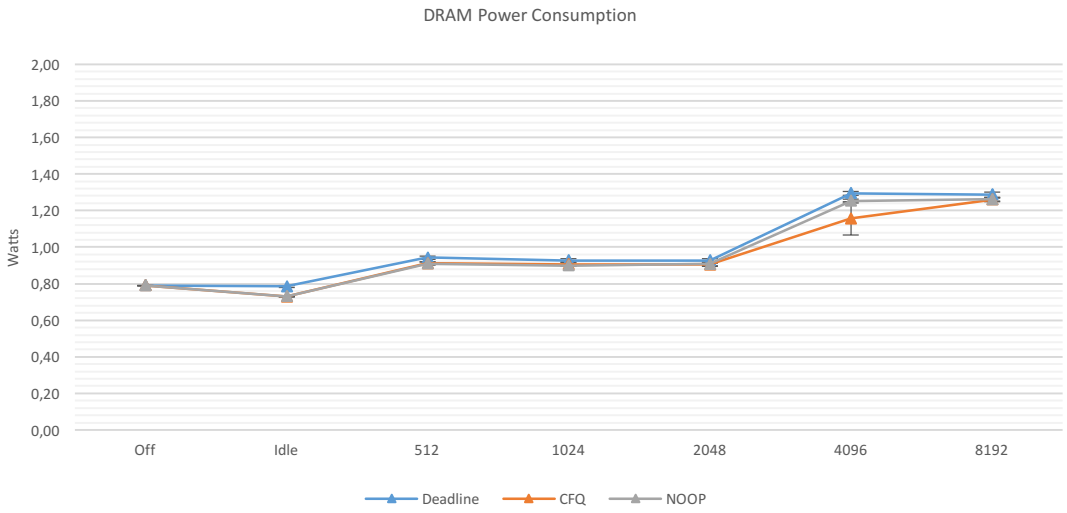


Figura 108 - Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia da memória RAM para os três IO Schedulers experimentados nos diferentes datasets.

Na escrita sequencial em sistema de ficheiros EXT4, a eficiência energética dos três diferentes IO *Schedulers* é bastante idêntica. Nestes ensaios realizados, o IO *Scheduler* mais eficiente varia consoante o *dataset* em causa. Como se pode visualizar no gráfico da Figura 109, os melhores resultados de eficiência são obtidos pelos algoritmos *Deadline* e CFQ por margens muito reduzidas. No *dataset* de 512, 2048 e 4096, o *Deadline* apresenta os melhores resultados de eficiência. Nos restantes *datasets*, os resultados mais positivos são obtidos pelo algoritmo CFQ. Os melhores resultados de eficiência são obtidos no *dataset* de 4096 e 8192 com um custo por bloco de aproximadamente 27 uW.

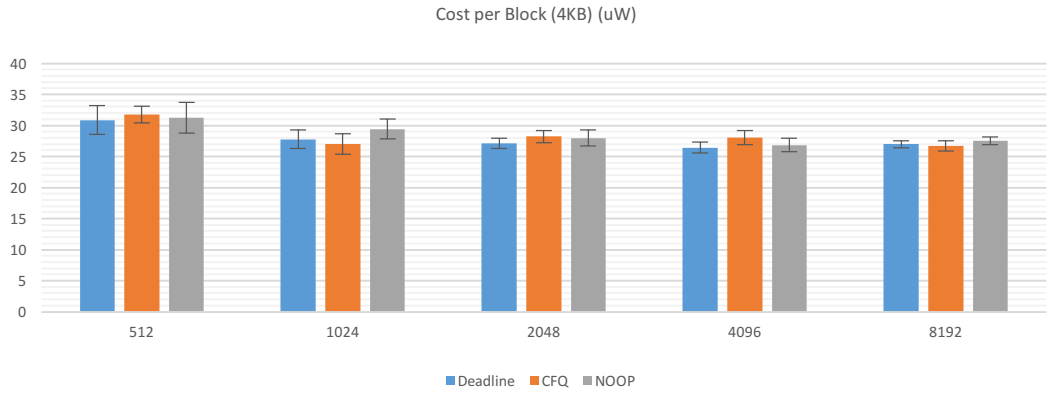


Figura 109 – Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets.

Em relação ao desempenho na escrita, destaca-se o algoritmo CFQ com os melhores resultados. O algoritmo atinge, no seu melhor rendimento durante o ensaio, uma média de 34,5 GB escritos. No gráfico da Figura 110 Figura 110, é possível observar-se todos os resultados obtidos nesta variante.

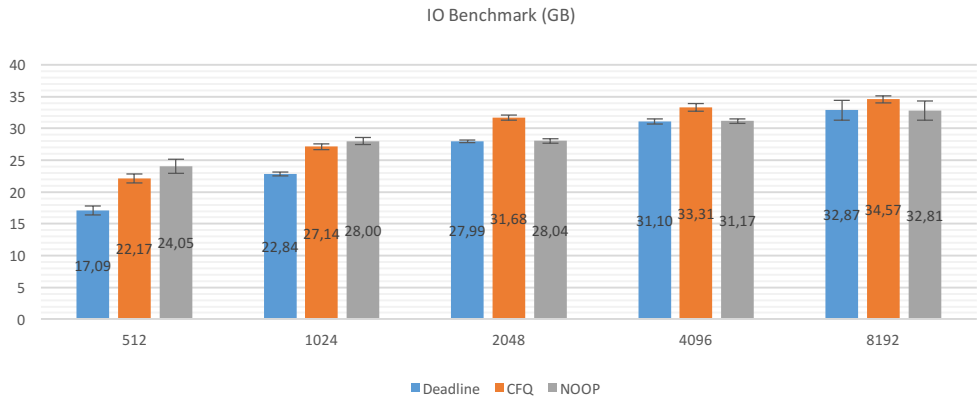


Figura 110 – Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da média de volume de dados escrito para os três IO Schedulers experimentados nos diferentes datasets.

O consumo energético do disco nos vários exercícios executados é demonstrado no gráfico da Figura 111. Em todos os *datasets*, o IO *Scheduler* que mais energia consumiu foi aquele que teve um melhor desempenho com um maior volume de dados escrito. O marco mais elevado de consumo de energia foi atingido no *dataset* 4096, pelo IO *Scheduler* CFQ com o valor aproximado de 4,2W.

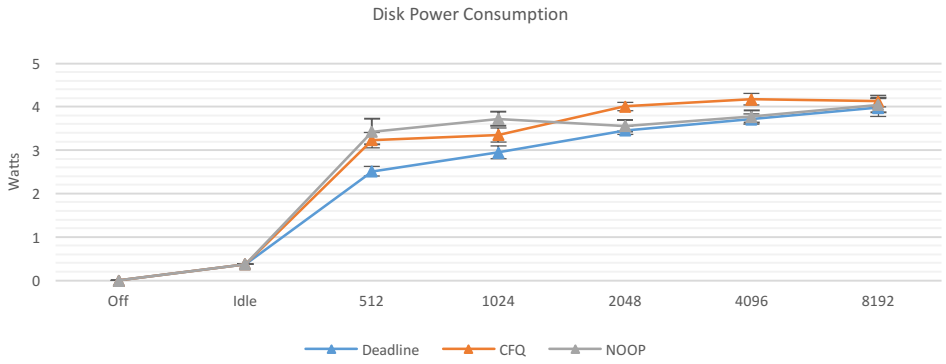


Figura 111 – Ensaio de escrita sequencial ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

Na escrita aleatória em modo raw, verificou-se uma destacável superioridade na eficiência energética do IO *Scheduler Deadline* quando comparado com os restantes. O custo por bloco demonstrado no gráfico da Figura 112 comprova esta situação onde o algoritmo *Deadline* tem um custo aproximado de metade daquele obtido com o algoritmo CFQ ou *NOOP*. A maior eficiência acontece no *dataset* de 8192 bytes, onde o algoritmo atinge o consumo de aproximadamente 80 uW.

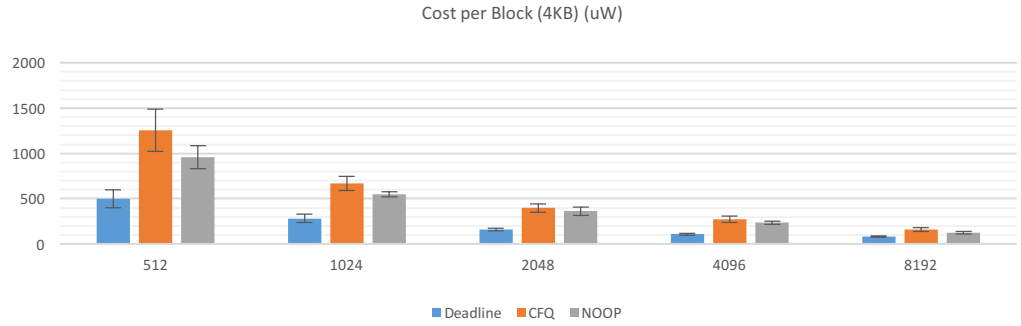


Figura 112 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets.

Este impressionante resultado deve-se maioritariamente ao desempenho de escrita que o disco consegue atingir com este algoritmo. O gráfico da Figura 113 demonstra todos os resultados médios obtidos nos diferentes *datasets*. O melhor resultado surge por parte do algoritmo *Deadline* no *dataset* de 8192 bytes, onde este atinge o marco de 3,4GB escritos aleatoriamente.

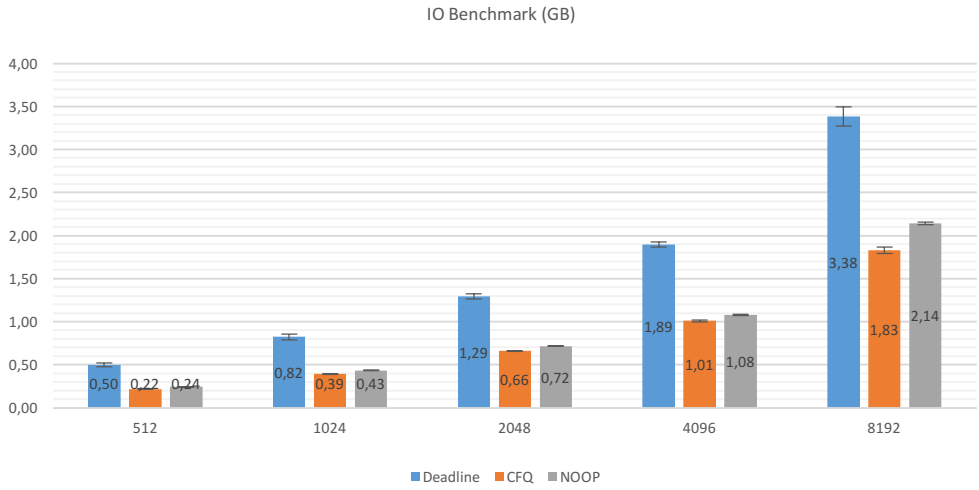


Figura 113 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da média de volume de dados escrito para os três IO Schedulers experimentados nos diferentes datasets.

O consumo energético do disco nos diferentes *datasets* e IO Schedulers apresenta valores muito idênticos. No entanto, como comprova o gráfico da Figura 114, o IO Scheduler *Deadline* apresenta resultados ligeiramente melhores com consumos um pouco mais reduzidos em alguns dos *datasets*.

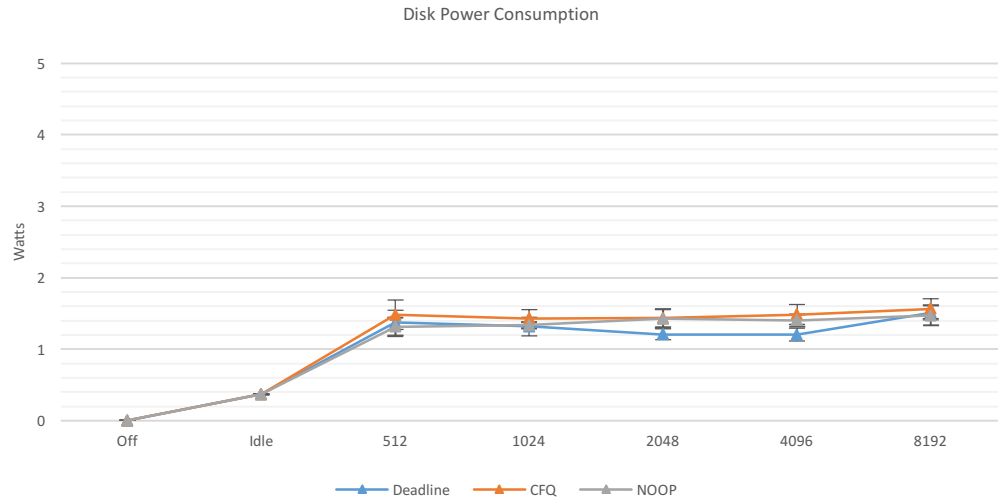
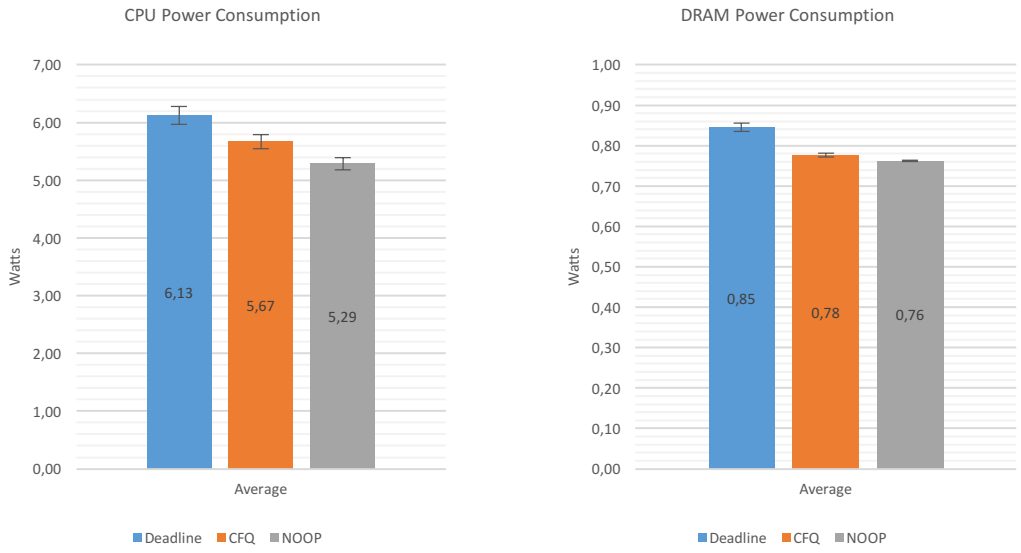


Figura 114 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

Ao contrario do que acontece no consumo energético do disco, o protocolo *Deadline* provoca um maior impacto energético no CPU e na memória DRAM. Isto deve-se principalmente ao seu bem superior rendimento comparado com os restantes IO *Schedulers*. No gráfico 1 da Figura 115, é possível verificar-se o valor

médio consumido do CPU com os diferentes algoritmos executados. Através desses dados apresentados é possível verificar-se que o maior marco é atingido pelo algoritmo *Deadline* com o CPU a consumir 6,13 *watts*. Na memória DRAM, como indica o gráfico 2 da Figura 115, o algoritmo *Deadline* provoca um consumo de energia na marca média de 0,85W.



1 2

Figura 115 - Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em modo raw. (1) Média do consumo de energia da memória RAM nos ensaios de cada um dos IO Schedulers. (2) Média do consumo de energia do CPU nos ensaios de cada um dos IO Schedulers.

Na escrita aleatória em sistema de ficheiros, verifica-se um melhor desempenho comparativamente com o mesmo modo de escrita em *raw*. Isto acontece principalmente porque existe uma melhor gestão sobre o sistema de ficheiros para não haver desalinhamentos de escrita, entre os blocos lógicos e físicos. Como se vê no gráfico da Figura 116, o melhor marco de volume de dados escrito atingido nestes ensaios acontece no *dataset* de 8192 com os algoritmos CFQ e *NOOP* a atingir 5,5GB. Os três algoritmos estudados apresentaram resultados muito idênticos nesta matéria. Apesar disso, o IO *Scheduler Deadline* apresenta valores ligeiramente inferiores, na casa dos 5% de decréscimo, nos *datasets* de 4096 e 8192.

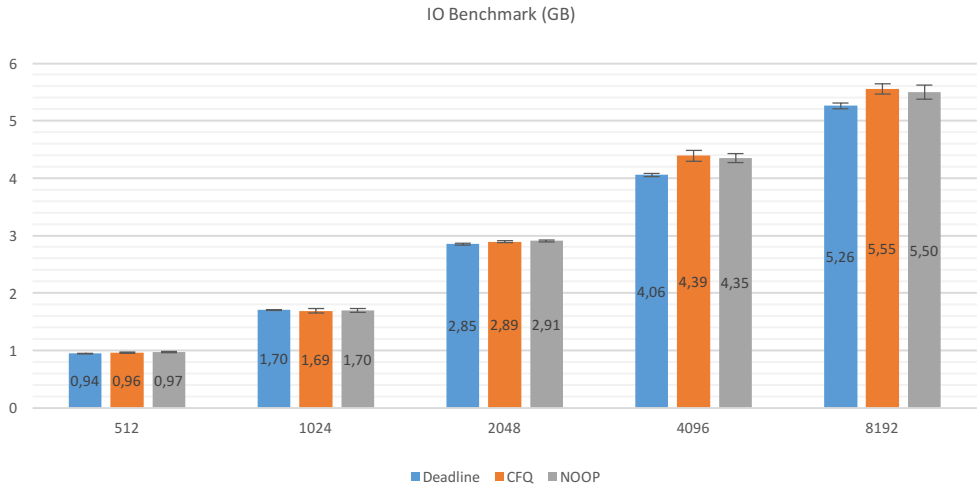


Figura 116 - Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da média de volume de dados escrito para os três IO Schedulers experimentados nos diferentes datasets.

No consumo do disco durante os vários *datasets* exercitados, o algoritmo *Deadline* destaca-se pela positiva. Recorrendo a este algoritmo, o disco SSD não ultrapassa os 2 *watts* de consumo em qualquer um dos *datasets*. No caso do *dataset* de 512 *bytes*, todos os IO *Schedulers* causam um consumo de energia ao disco que se fixa nos 2 *watts*. Os IO *Schedulers*, CFQ e *NOOP*, não apresentam resultados tão satisfatórios, verificando-se assim alguma variação entre datasets na potência consumida, com valores a oscilar entre os 2 e 3,5 *watts*. O gráfico da Figura 117 expõe esta situação de uma forma mais esclarecedora.

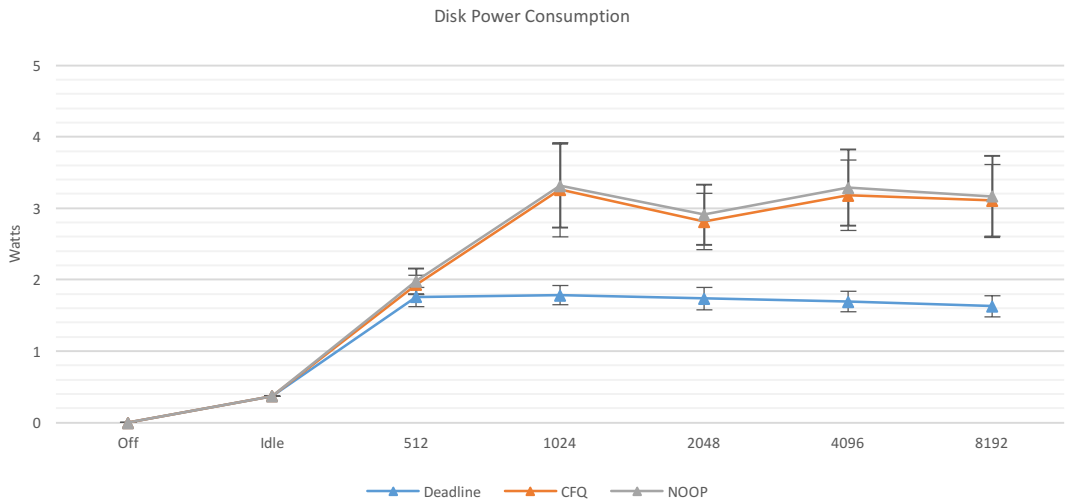


Figura 117 - Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia do disco para os três IO Schedulers experimentados nos diferentes datasets.

No seguimento dos bons resultados obtidos pelo IO *Scheduler Deadline* no consumo do disco, confirma-se nesta etapa que este é o algoritmo mais eficiente para o disco. O gráfico da Figura 118 demonstra o cálculo do custo energético por bloco de 4KB nos diferentes *datasets*. Através do mesmo, é possível comprovar que, à exceção do sucedido no *dataset* de 512, o IO *Scheduler Deadline* provoca um impacto energético no disco 50% inferior aos restantes IO *Schedulers*. No caso do *dataset* de 512, os resultados entre algoritmos são bastante parecidos, com uma reduzida superioridade de eficiência por parte do algoritmo *Deadline*. O melhor custo energético por bloco de 4KB é conseguido no *dataset* de 8192 pelo mesmo algoritmo com um valor de 50 uW.

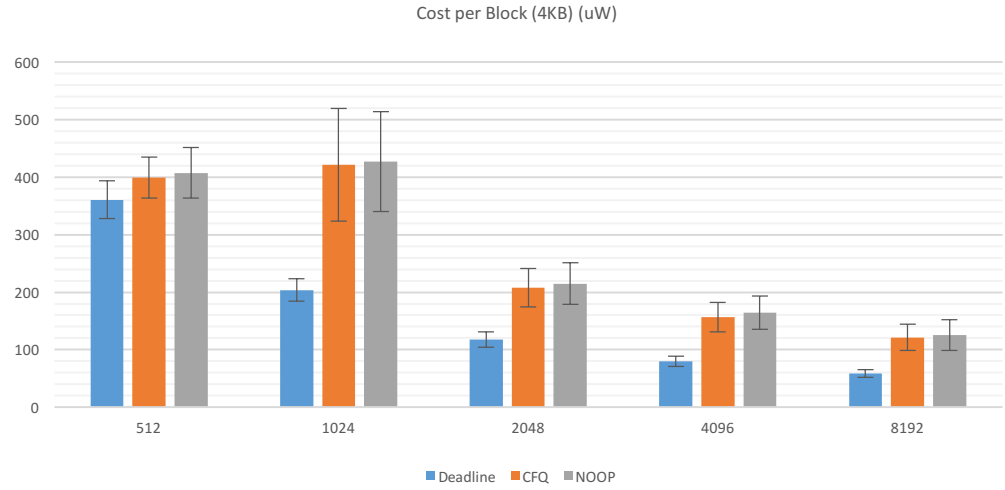


Figura 118 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Média de custo por bloco de 4KB escrito para os três IO Schedulers nos diferentes datasets.

Como o disco SSD recorre à memória DRAM para operações de cache, esta memória também tem um crescimento de consumo de energia durante os ensaios. O gráfico da Figura 119 mostra esse impacto na memória primária para os diferentes *datasets* exercitados. Constata-se nestes resultados obtidos que, este ensaio, provocou um crescimento de aproximadamente 1 *watt* no consumo de energia da memória para qualquer um dos IO *Schedulers* examinados.

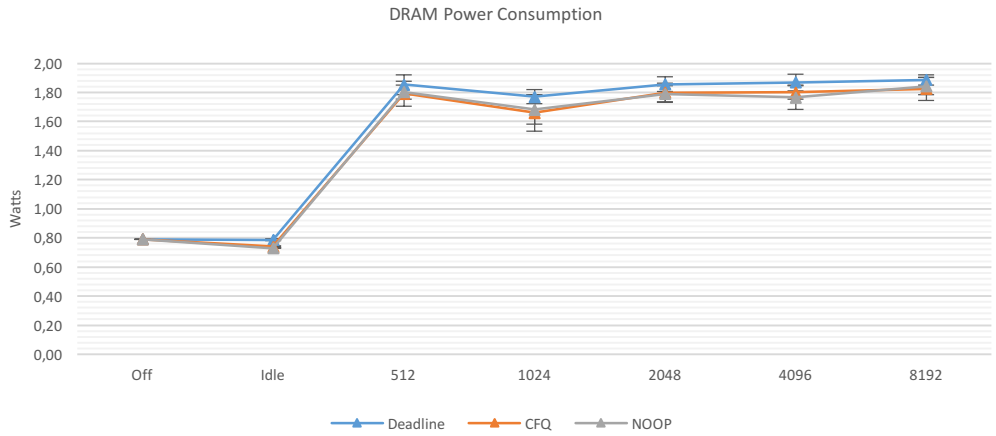


Figura 119 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia da memória RAM para os três IO Schedulers experimentados nos diferentes datasets.

O consumo energético do CPU também se viu afetado com a execução deste ensaio. Como se pode constatar no gráfico da Figura 120, existe um crescimento de 12W no consumo do componente entre o estado *idle* do disco e qualquer exercício ao disco. Isto acontece com a elevada utilização da memória RAM para serviços de cache e à gestão de execução da aplicação durante os ensaios.

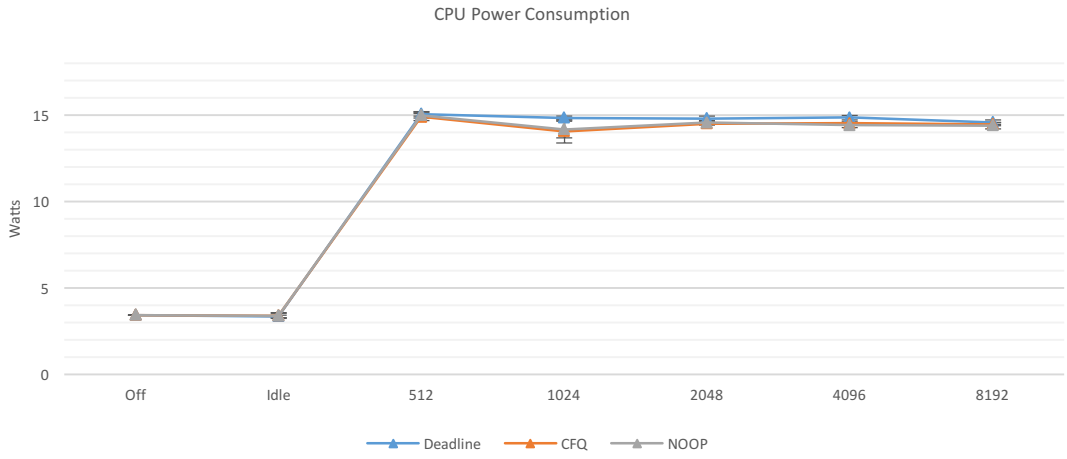


Figura 120 – Ensaio de escrita aleatória ao disco SSD com a técnica de write-caching ativa e em sistema de ficheiros. Gráfico da evolução do consumo de energia do CPU para os três IO Schedulers experimentados nos diferentes datasets.

4.3.5 Balanço Final dos Ensaios

4.3.5.1 Distribuição do Consumo de Energia no Sistema

A distribuição do consumo de energia pelo sistema foi analisada durante os ensaios realizados ao mesmo. Esta analise permite perceber-se quais os componentes que mais consomem estaticamente e, qual o impacto do disco entre todos os subsistemas do sistema global tanto a nível estático como a nível dinâmico. Nesta análise, foram colocados em exame ambos os discos, o disco HHD e o SSD.

Na configuração complementada com o disco HDD, os resultados do sistema global em estado estático podem ser verificados nos gráficos da Figura 121. Estes dados foram contabilizados com o disco HDD em estado de *idle* e com o sistema sem qualquer atividade provocada pelo utilizador. Nestas condições, é possível verificar que o disco representa 16% do consumo total do sistema e, posiciona-se como segundo maior consumidor de energia do sistema entre os restantes subsistemas em análise. Foi analisada a mesma situação com a diferença de que o disco HDD estava em estado de *standby* e, verificou-se um impacto mais reduzido de apenas 6% que tornou este subsistema o terceiro maior consumidor de energia do sistema.

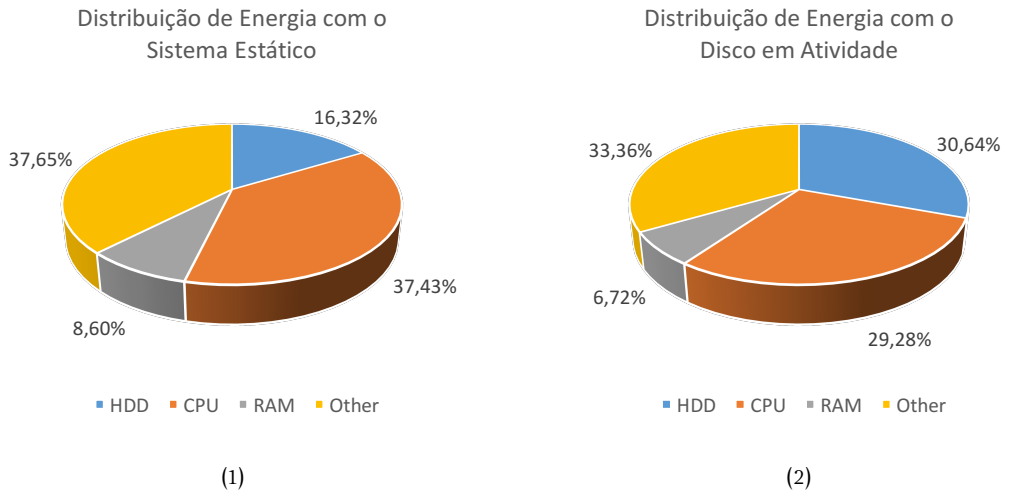


Figura 121 - Distribuição do consumo de energia do disco HDD e dos restantes principais subsistemas analisados num sistema. (1) Distribuição de energia com o sistema estático. (2) Distribuição de energia com o disco em atividade.

Quando o disco HDD foi colocado com elevadas cargas de trabalho, verificou-se que este pode chegar mesmo a ser o maior consumidor de energia de um sistema ao representar 30% do consumo total. Esta situação pode ser consultada no gráfico 2 da Figura 121.

Na configuração complementada com o disco SSD e com o sistema em estado estático, os resultados da distribuição do consumo de energia pelos subsistemas apresentam-se nos gráficos da Figura 122. Neste componente de memória *flash*, é indiferente configurar o seu estado para *idle* ou *standby* através da ferramenta HDPARM, sendo que, para o mesmo, só existe o estado *idle*. Verifica-se com estes dados que, o disco SSD, ao ter apenas o estado *idle*, apresenta um consumo uniforme em períodos de inatividade. Neste estado, o disco SSD destaca-se por apresentar um consumo extremamente baixo, representando apenas 4% do consumo total do sistema. Na situação de alto desempenho deste componente, este pode chegar a representar 35% da energia total consumida do sistema.

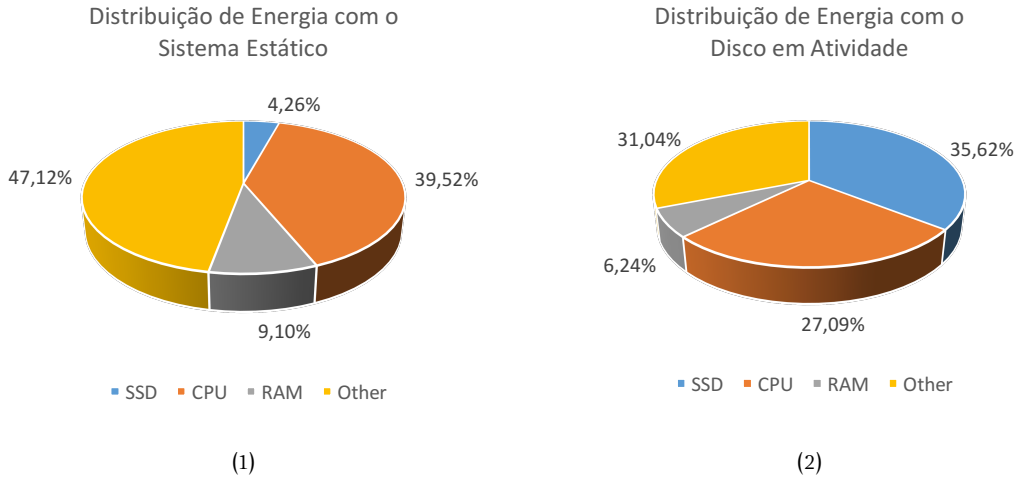


Figura 122 - Distribuição do consumo de energia do disco SSD e dos restantes principais subsistemas analisados num sistema. (1) Distribuição de energia com o sistema estático. (2) Distribuição de energia com o disco em atividade.

Numa analise comparativa entre os dois discos, verifica-se que, com ambos os componentes em estado *idle*, o impacto energético do disco SSD no sistema é praticamente metade daquele apresentado com o disco HDD. Além disso, existe ainda o maior impacto energético na transição de estado que não é possível contabilizar com os recursos utilizados. Esta variação ocorre sobre o disco HDD em situações como a de ligar o VCM ou alterar a sua velocidade de rotação. Na situação comparativa entre os dois discos em altos níveis de atividade, o disco SSD acaba por ter um maior impacto sobre o consumo total de energia do sistema. No entanto, este valor acaba por ser compensado pela quantidade de dados bem superior que lê ou escreve em relação ao disco HDD. Desta forma, pode-se considerar o disco SSD bastante mais cumpridor do termo *power-porpotional*. Além disto, esta análise comprova que o subsistema de armazenamento secundário pode ser uma grande preocupação ao nível

do impacto energético de um sistema, podendo mesmo chegar a ser o maior consumidor de energia do mesmo.

4.3.5.2 Potência Estimada pelos Fabricantes

Numa fase em que os discos já foram colocados à prova em vários exames e os seus consumos energéticos já foram estudados aprofundadamente, é altura de se verificar a veracidade dos detalhes energéticos apresentados pelos fabricantes dos mesmos. Esta análise é interessante de se realizar para se perceber até que ponto se consegue estudar o consumo de um sistema recorrendo apenas ao folheto informativo do fabricante do componente. Entre todos os ensaios realizados, procurou-se recorrer àqueles que mais se identificam com os estados de atividade do disco que o fabricante estima. Desta forma, nas operações de leitura optou-se por recorrer aos ensaios em modo *raw* com recurso à técnica de *read-ahead*. Nas operações de escrita, optou-se por recorrer aos ensaios em sistema de ficheiro com a utilização da técnica *write-caching*.

No disco HDD, o fabricante disponibiliza detalhes energéticos bastante esclarecedores, identificando assim as diferentes situações de operação do disco. Para as situações de inatividade do disco, o fabricante disponibiliza uma estimativa de consumo para o estado *idle* e *standby*. A Tabela 8 apresenta a média de consumo de energia apresentada pelo fabricante, comparada com a média obtida nos ensaios realizados ao disco. No caso das situações de atividade do disco, existe informação de consumo para os estados e escrita, leitura e *seek*. No estado *standby*, os valores obtidos no exame realizado apresentam uma média de 0,5 *watts*. O consumo energético recolhido é superior, em mais do dobro, àquele estimado pelo fabricante. Já no estado de *idle*, o resultado da comparação entre o consumo energético recolhido e estimado é mais satisfatório. Neste estado, o consumo energético recolhido no ensaio de inatividade apresenta-se razoavelmente superior, com uma diferença de 0,6 *watts*.

Tabela 8 – Tabela de comparação dos detalhes de consumo estimados pelo fabricante do disco HDD e dos consumos calculados nos ensaios do projeto.

Disco HDD		
Estado	**Fabricante**	**Ensaios**
Standby	0,18 W	0,5 W
Idle	0,9 W	1,5 W
Active - Read	1,70 W	2,4 W
Active - Write	1,90 W	2,5 W
Active - Seek	1,80 W	1,80 W / 1,35 W

Nas atividades de leitura, o modo sequencial apresentou uma discrepância de resultados, enquanto o modo aleatório apresentou valores mais aceitáveis que se aproximam daqueles estimados pelo fabricante. Durante os ensaios de leitura sequencial, a média de valores recolhidos de consumo de energia do disco ronda os 2,4 *watts*, valor este que é 40% superior ao estimado. No caso da leitura aleatória, o valor médio de consumo de disco recolhido nos ensaios foi de 1,80W. Nesta atividade, existe um estado de execução do disco que varia entre *seek* e *read*. Analisando a estimativa do fabricante, verifica-se que os valores estimados para estes dois estados vão de encontro com os valores médios obtidos durante os ensaios.

No ensaio de escrita sequencial obteve-se um consumo médio de 2,5 W e o consumo energético do disco nesta atividade variou entre os 2 e 3 *watts*. Desta forma, este valor pode chegar a ser mais de 50% superior ao indicado pelo fabricante, visto que, no estudo do fabricante, os valores desta atividade têm uma média de 1,90 *watts*. Na escrita aleatória verificou-se um valor bastante abaixo daquele estimado pelo fabricante. Esta atividade resume-se a uma constante variação entre operações de *write* e *seek*. Os resultados recolhidos apresentam uma média de 1,35W, valor este que é 25% inferior ao estimado para este tipo de atividade.

No disco SSD, o fabricante apenas dividiu o consumo de energia do componente por dois diferentes estados, *idle* e *active*. Enquanto o estado *idle* se resume ao consumo estático do disco em qualquer período de inatividade, o estado Active resume-se ao consumo energético médio de qualquer atividade do disco. A Tabela 9 apresenta a média de consumo de energia apresentada pelo fabricante, comparada com a média obtida nos ensaios realizados ao disco. O fabricante do disco não diferencia a situação de *idle* e *standby*, visto que, a principal diferença destes estados acontece nos discos mecânicos, com a rotação dos pratos. Isto é confirmado nos ensaios de inatividade realizados sobre estes dois estados, ao verificar-se que ambos apresentam valores idênticos. O valor médio de consumo de energia nestes estados rondou os 0,37 *watts*, valor este que é sete vezes superior àquele constatado na estimativa energética do fabricante.

Tabela 9 - Tabela de comparação dos detalhes de consumo estimados pelo fabricante do disco SSD e dos consumos calculados nos ensaios do projeto.

Disco SSD		
Estado	**Fabricante**	**Ensaios**
Idle	0,05 W	0,37 W
Active	3 W	3,5 W

Nas operações de leitura, verificou-se alguma coerência entre os dados recolhidos e a estimativa apresentada pelo fabricante. No ensaio de leitura sequencial obteve-se uma média de 2,9 *watts*. Na leitura aleatória, obteve-se um menor volume de dados lido e por isso o consumo energético foi também mais reduzido, com uma média de 1,5 *watts*. A estimativa do fabricante numa situação ativa do disco é de 3 *watts*. Uma informação limitada e pouco concreta sobre várias situações. Desta forma, impede que se desenvolva um modelo com base nesses detalhes.

Nas operações de escrita, constatou-se a mesma situação que, a estimativa energética apresentada pelo fabricante no estado *active*, não cobre as diversas operações e os seus diferentes impactos energéticos. Na escrita sequencial obteve-se um consumo de energia variável, entre os 2,7 e 4,1 *watts*, consoante o volume de dados escrito entre *datasets*. Nas operações de escrita no modo aleatório verificou-se um valor a rondar 1,9 *watts*.

4.4 Proposta do Modelo de Consumo de Energia Parametrizado

Depois de um estudo sobre vários fatores que podem interferir no consumo energético de um sistema, surge a etapa de apresentação de um modelo capaz de estimar o consumo de energia desse mesmo sistema. Foram estudadas as tecnologias e indicadores mais adequados para se atingir o principal objetivo com os melhores resultados possíveis. O sistema destaca-se por apresentar a tecnologia RAPL no processador, o que permite o desenvolvimento de um modelo mais fiável com uma colaboração indireta do fabricante deste componente. A etapa mais árdua da construção deste modelo surge na estimação do consumo energia do subsistema de armazenamento secundário. Por isso, é nessa mesma etapa que os ensaios realizados têm uma maior relevância para se obter um modelo preciso e estável.

Neste modelo proposto, a componente de estimativa do consumo de energia do armazenamento secundário é especialmente desenhada para apresentar melhores resultados quando este subsistema recorre à utilização de sistema de ficheiros, apresenta as partições alinhadas e atua com as técnicas de *write-caching e read-ahead* ativas. Esta decisão surge ao se reconhecer que esta configuração é a mais utilizada no funcionamento de um sistema. No entanto, o estudo das restantes condicionantes foi bastante enriquecedor para se compreender o diferente impacto energético do subsistema consoante as configurações utilizadas.

4.4.1 Base do Modelo

Como referido ao longo do projeto, o modelo desenvolvido apresenta duas vertentes ao nível do disco, podendo este ser um disco HDD ou SSD. Desta forma, independentemente do disco em causa, a restante estrutura do modelo mantem-se e pode ser compreendida pela equação,

$$P_{total} = P_{idle\_other} + P_{CPU} + P_{DRAM} + P_{HDD} + P_{SSD} \quad (11)$$

onde, P_{idle_other} representa o consumo em *idle* do sistema sem incluir qualquer consumo de energia do CPU, memória primária e armazenamento secundário. O P_{CPU} e P_{DRAM} representam o consumo de energia estático e dinâmico dos subsistemas CPU e memória primária, respetivamente. O P_{HDD} e P_{SSD} representam o consumo de energia estático e dinâmico do disco HDD e SSD, respetivamente.

No modelo do disco que é proposto, tanto para o disco HDD como para o SSD, separou-se a estimativa do consumo de energia em duas componentes, o consumo estático e dinâmico. Assim, o modelo base pode-se compreender da seguinte forma,

$$P_{disco} = P_{disco\_estática} + P_{disco\_dinâmica} \quad (12)$$

em que $P_{disco\_estática}$ e $P_{disco\_dinâmica}$ representam, respetivamente, o consumo de energia do subsistema em *idle* e o consumo suplementar de quando este se encontra em atividade. O valor atribuído ao consumo de energia estático é invariável e, com base nos ensaios realizados, é de 1,5 *watts* para o disco HDD e de 0,37 *watts* para o disco SSD. Estes valores foram obtidos com um ensaio de *idle* ao sistema para cada disco. No caso deste modelo ser adaptado a outras configurações com outros discos, é necessário realizar-se novamente este ensaio para se conhecer o consumo de energia do subsistema.

A estimativa do consumo de energia dinâmico do subsistema é também desenvolvida com base nos ensaios realizados e pode ser representada através da equação,

$$\begin{aligned} P_{disco\_dinâmica} = {} & W_{seq}\left(W_{ops}\times W_{cost(seq)}\right) + W_{ran}\left(W_{ops}\times W_{cost(ran)}\right) \\ & + R_{seq}\left(R_{ops}\times R_{cost(seq)}\right) + R_{ran}\left(R_{ops}\times R_{cost(ran)}\right) \end{aligned} \quad (13)$$

onde W_{seq} e W_{ran} representam as *flags* sinalizadoras do tipo de operação de escrita realizada e, no caso da leitura, a representação é feita através de R_{seq} e R_{ran}. Os parâmetros W_{ops} e W_{cost} representam o numero de operações de escrita e o custo de cada operação, respetivamente. As operações de leitura utilizam os parâmetros R_{ops} e R_{cost} para apresentarem os mesmos valores.

As *flags* que sinalizam o tipo de operação realizada, tanto para a escrita como leitura, estão numa escala entre 0 e 1 que assinala a percentagem sobre o modo de

acesso em que as operações atuaram. Por exemplo, se metade das operações recolhidas forem de escrita sequencial e outra metade de escrita aleatória, pode-se atribuir o valor 0.5, tanto ao parâmetro W_{seq} como W_{ran}. Desta forma, utilizando as seguintes equações,

$$W_{seq} = 1 - W_{ran} \quad (14)$$

$$R_{seq} = 1 - R_{ran} \quad (15)$$

pode-se limitar a atribuição de um valor apenas a um modo de acesso, tanto para a escrita como para a leitura. Para se perceber melhor o funcionamento das *flags*, é apresentada a Tabela 10 que atribui valores às *flags* consoante o tipo de operação e o modo de acesso.

Tabela 10 - Exemplo de flags do modelo que identificam determinados exercícios de um disco

Operation	*Mode*	W_{seq}	R_{seq}
Write	*Sequential*	1	N.A.
	Random	0	N.A.
Read	*Sequential*	N.A.	1
	Random	N.A.	0
Read'n'Write	*Sequential*	1	1
	Random	0	1

No caso das operações de *Read'n'Write* aleatório, considera-se apenas que a escrita é aleatória porque habitualmente existe um padrão de leitura para que algo seja escrito em diferentes posições de memória. Desta forma, neste modo de operação, optou-se por considerar a leitura sequencial e a escrita aleatória.

Nas duas secções seguintes são assinalados os valores atribuídos às variáveis W_{cost} e R_{cost}, valores estes que variam consoante o disco e a operação em causa.

4.4.2 Disco HDD

Depois de serem realizados todos os ensaios relativos ao disco HDD, conjugou-se os resultados mais relevantes dos ensaios em duas tabelas para tornar o processo de desenvolvimento do modelo mais simples e prático. Essas mesmas duas tabelas estão representadas na Figura 123. A primeira tabela relata o custo energético por bloco de 4KB distribuído por *dataset*, tipo de operação (*read* ou *write*) e por modo de acesso (sequencial ou aleatório). A segunda tabela relata o custo energético por operação distribuído também da mesma forma. Através destas tabelas, é possível verificar-se que o consumo de energia dos acessos aleatórios é muito superior ao dos acessos sequenciais. Nos dados recolhidos de leituras, o acesso aleatório chega a ser 2300

vezes superior ao acesso sequencial, ao passo que, na escrita, o acesso aleatório chegou a ser 7000 vezes superior.

	Cost p/ Block (4KB)			
	Read		Write	
	Sequential	Random	Sequential	Random
Dataset Size	uW	mW	uW	mW
512	47,06	110,73	18,29	126,79
1024	48,51	53,22	23,72	84,47
2048	48,12	25,93	37,1	53,38
4096	48,34	13,76	48,46	44,67
8192	48,79	7,82	51,11	33,67

(1)

	Cost p/ Operation			
	Read		Write	
	Sequential	Random	Sequential	Random
Dataset Size	uW	mW	uW	mW
512	5,88	13,97	2,29	15,82
1024	12,13	13,46	5,96	21,14
2048	24,06	13,05	18,65	26,69
4096	48,34	13,89	48,46	44,67
8192	97,59	15,79	101,94	67,26

(2)

Figura 123 – Resultados dos ensaios ao disco HDD. Custo energético por bloco de 4 KB (1) e por operação (2) distribuídos por *dataset*, tipo de operação e modo de acesso.

Neste conjunto de dados, existem outras conclusões úteis que se podem retirar para o desenvolvimento do modelo. No caso das operações de leitura sequencial, verifica-se que o custo por bloco é pouco variável entre *datasets*. O mesmo acontece no custo por operação das leituras aleatórias, isto é, o consumo de energia por operação varia pouco nos vários *datasets*. Estas duas situações podem ser verificadas no gráfico 1 e 2 da Figura 124.

No caso da escrita, tanto em operações aleatórias como em operações sequenciais, verificou-se que os valores apresentados, tanto no custo por bloco como no custo por operação, são variáveis entre *datasets* e não apresentam um padrão evolutivo uniforme. Noutra perspetiva, esta situação pode ser demonstrada através do gráfico 3 e 4 da Figura 124, onde se apresenta o custo por bloco da escrita sequencial e o custo por operação da escrita aleatória, respetivamente.

Para se preencher as variáveis W_{cost} e R_{cost} do modelo, consoante o *dataset* e modo de operação em causa, optou-se por usar os valores apresentados na Figura 124. No entanto, esses dados não abrangem todos os *datasets* possíveis das operações executadas e, por isso, procurou-se outro método capaz de responder a mais *datasets* que podem surgir nas operações utilizadas. Por uma questão de gestão de tempo e por este projeto ter tomado uma dimensão maior do que a esperada, optou-se por não se realizar novos ensaios para implementar novos *datasets*. Não obstante, acredita-se que a adição de mais *datasets* tornaria o modelo mais robusto e mais refinado.

Figura 124 – Gráficos dos resultados dos ensaios ao disco HDD. Custo por bloco (4KB) das operações de leitura sequencial (1); Custo por operação de leituras aleatórias (2); Custo por bloco (4KB) das operações de escrita sequencial (3); Custo por operação de escrita aleatória (4).

Posto isto, recorreu-se à ferramenta *CurveExpert* [184] para se obter uma função que se ajuste aos valores obtidos nos ensaios e realize uma estimativa dos restantes valores. Em seguida são apresentadas as funções utilizadas para o modelo do disco HDD, sendo que, em todas elas, a variável α representa o *dataset* em causa.

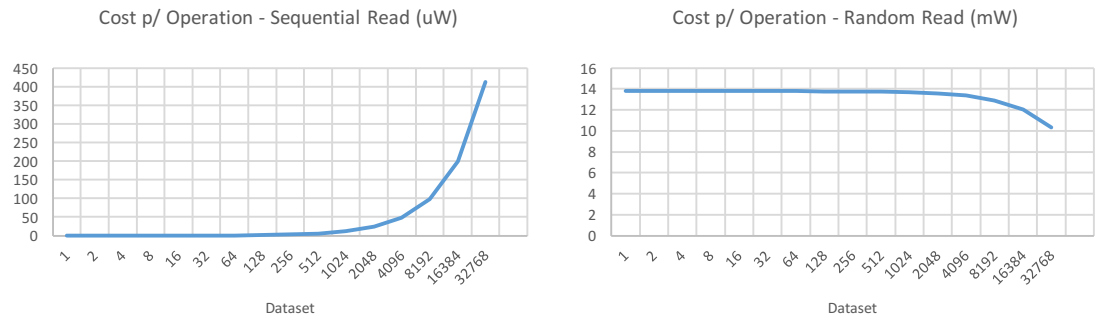
Para a leitura sequencial, a ferramenta utilizou o modelo *Saturation Growth-Rate* para gerar a fórmula:

$$R_{cost(seq)} = 10^{-6}\left(\frac{-5409{,}06\ \alpha}{-462261{,}17 + \alpha}\right) \tag{16}$$

No caso da leitura aleatória, a ferramenta utilizou o modelo de função hiperbólica e sugeriu a seguinte equação:

$$R_{cost(ran)} = 10^{-3}\left(\frac{-0{,}44\ \alpha + 56454{,}77}{4096}\right) \tag{17}$$

Para se perceber melhor o custo por operação, optou-se por traduzir estas fórmulas em gráficos, cujos resultados obtidos podem ser visualizados na Figura 125.



(1) (2)

Figura 125 – Gráficos obtidos através de fórmulas geradas pela ferramenta *CurveExpert* a partir dos dados obtidos nos ensaios do disco HDD. Custo por por operação de leitura sequencial (1) e de leitura aleatória (2).

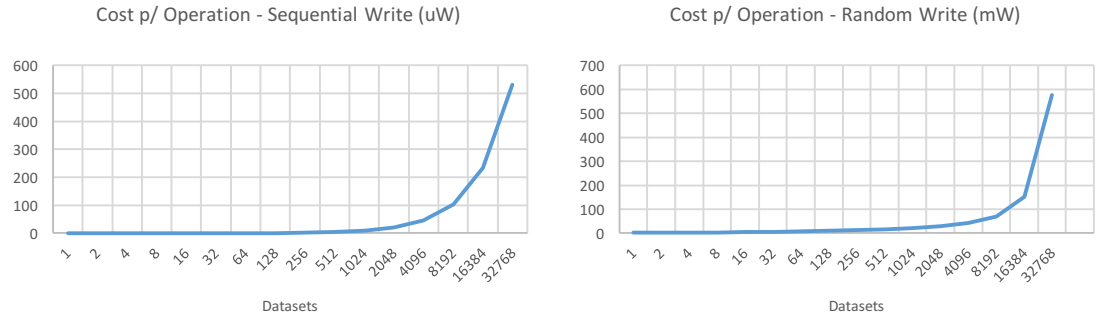
Para se representar a escrita sequencial, utilizou-se o modelo de potência e a fórmula representa-se da seguinte forma:

$$W_{cost(seq)} = 10^{-6}(2{,}37\times10^{-3}\times \alpha^{1{,}19}) \quad (18)$$

A escrita aleatória representou-se com o modelo Hoerl através da fórmula:

$$W_{cost(ran)} = 10^{-3}\left(\frac{7257{,}43 \;\; \alpha - \alpha^{0{,}35}}{4096}\right) \quad (19)$$

Utilizando as duas funções de escrita obtidas na ferramenta *CurveExpert*, construíram-se dois gráficos que estão apresentados na Figura 126.



(1) (2)

Figura 126 – Gráficos obtidos através de fórmulas geradas pela ferramenta *CurveExpert* a partir dos dados obtidos nos ensaios do disco HDD. Custo por operação de escrita sequencial (1) e de escrita aleatória (2).

4.4.3 Disco SSD

Depois de serem realizados todos os ensaios ao disco SSD, foram apontados os resultados que se consideraram mais relevantes para o desenvolvimento do modelo.

Desta forma, a Figura 127 Figura 127 apresenta duas tabelas que resumem o balanço do consumo de energia do disco nos diferentes tipos de operação. Neste disco, apesar da diferença não ser tão grande como no disco HDD, volta-se a reconhecer o maior consumo de energia das atividades de acesso aleatório em relação as atividades de acesso sequencial. Na leitura e escrita, o acesso aleatório pode chegar a ser 18 e 12 vezes superior ao acesso sequencial, respetivamente.

	Cost p/ Block (4KB)			
	Read		Write	
	Sequential	Random	Sequential	Random
Dataset Size	uW	uW	uW	uW
512	18,24	324,07	30,87	360,77
1024	17,83	181,08	27,76	203,71
2048	17,61	92,96	27,08	117,57
4096	17,64	52,28	26,44	80,1
8192	17,77	32,28	26,99	58,84

(1)

	Cost p/ Operation			
	Leitura		Escrita	
	Sequential	Random	Sequential	Random
Dataset Size	uW	uW	uW	uW
512	2,28	40,52	3,85	45,12
1024	4,46	45,28	6,94	50,93
2048	8,81	46,45	13,54	58,8
4096	17,64	52,3	26,44	80,09
8192	35,55	64,75	53,99	117,69

(2)

Figura 127 - Resultados dos ensaios ao disco SSD. Custo energético por bloco de 4 KB (1) e por operação (2) distribuídos por *dataset*, tipo de operação e modo de acesso.

Através de uma análise sobre as tabelas da Figura 128, verifica-se que o consumo de energia por bloco apresentado, tanto para as operações de acesso sequencial de leitura como escrita, tem uma variação reduzida entre diferentes *datasets*. No caso das operações de acesso aleatório, estas apresentam valores de maior variação consoante o *dataset* em causa. Estas situações podem ser analisadas através da Figura 128. Os gráficos 1 e 3, que correspondem respetivamente ao custo por bloco de leitura e escrita sequencial, apresentam variações pequenas entre *datasets*. Os gráficos 2 e 4, representam respetivamente o custo por operação de leituras e escritas aleatórias e, apresentam diferenças maiores entre *datasets*.

Para se desenvolver o modelo relativo ao disco SSD recorreu-se ao mesmo procedimento do utilizado para o modelo do disco HDD. Desta forma, para se preencher as variáveis W_{cost} e R_{cost} do modelo, consoante o dataset e modo de operação em causa, optou-se por usar os valores apresentados na Figura 128.

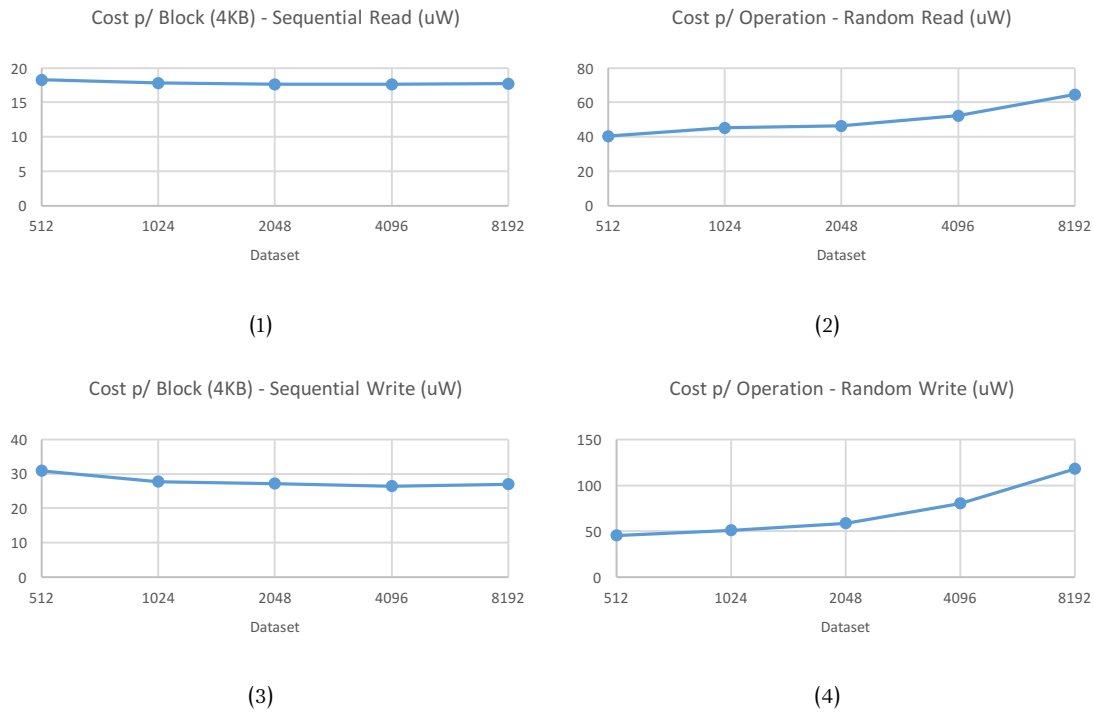


Figura 128 – Gráficos dos resultados dos ensaios ao disco SSD. Custo por bloco (4KB) das operações de leitura sequencial (1); Custo por operação de leituras aleatórias (2); Custo por bloco (4KB) das operações de escrita sequencial (3); Custo por operação de escrita aleatória (4).

No entanto, tal como no caso do disco HDD, esses dados não abrangem todos os *datasets* possíveis das operações executadas e, por essa razão, recorreu-se novamente à ferramenta *CurveExpert* [184] para se obter uma função que se ajuste aos valores obtidos nos ensaios e realize uma estimativa dos restantes valores. Em seguida são apresentadas as funções utilizadas para o modelo do disco SSD, sendo que, em todas elas, a variável α representa o *dataset* em causa.

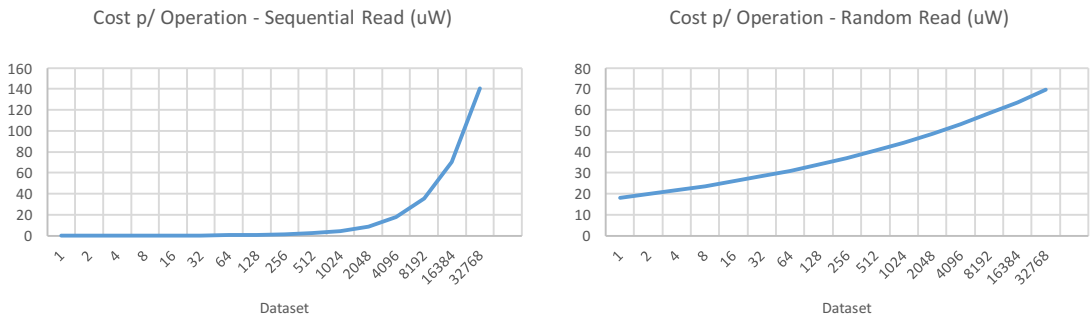
Para a leitura sequencial, a ferramenta utilizou o modelo *Weibull* para gerar a fórmula:

$$R_{cost(seq)} = 10^{-6}\left(\frac{63{,}67\ \alpha - 46{,}10\ \alpha\ e^{-3{,}43\times\alpha^{-0{,}89}}}{4096}\right) \quad (20)$$

A leitura aleatória representou-se com a função de Potência através da fórmula:

$$R_{cost(ran)} = 10^{-6}(18{,}06\ \alpha^{0{,}13}) \quad (21)$$

Utilizando as duas funções de leitura obtidas na ferramenta CurveExpert, construíram-se dois gráficos que estão apresentados na Figura 129.



(1) (2)

Figura 129 – Gráficos obtidos através de fórmulas geradas pela ferramenta *CurveExpert* a partir dos dados obtidos nos ensaios do disco SSD. Custo por operação de leitura sequencial (1) e de leitura aleatória (2).

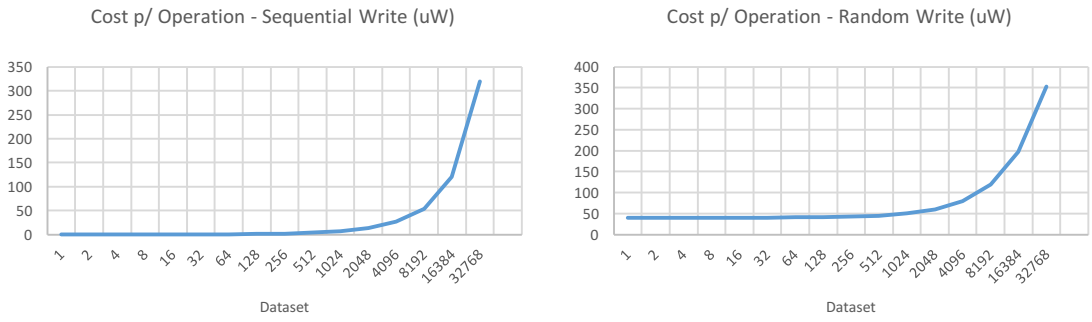
Na escrita sequencial, a ferramenta recorreu ao modelo *Hoerl* para chegar à seguinte fórmula:

$$W_{cost(seq)} = 10^{-6}(1{,}3\ \alpha^{0{,}9}) \tag{22}$$

No caso da escrita aleatória, através da função hiperbólica, obteve-se a seguinte fórmula:

$$W_{cost(ran)} = 10^{-6}\left(\frac{39{,}15\ \alpha + 1{,}65}{4096}\right) \tag{23}$$

Através destas duas formulas, construíram-se os dois gráficos de custo de operação que representam a escrita sequencial e aleatória. Estes gráficos podem ser visualizados na Figura 130.



(1) (2)

Figura 130 – Gráficos obtidos através de fórmulas geradas pela ferramenta *CurveExpert* a partir dos dados obtidos nos ensaios do disco SSD. Custo por operação de escrita sequencial (1) e de escrita aleatória (2).

4.5 Sumário

Depois de ultrapassada esta etapa, há várias conclusões possíveis de se retirar dos ensaios realizados, tanto ao nível do disco HDD como do disco SSD.

Inicialmente foram definidas as especificações essenciais a serem estudadas nos ensaios ao subsistema, nomeadamente, o estado de energia e operação, o tipo de operação, o tipo de acesso, as politicas de utilização de *cache*, o *dataset* utilizado por operação, a formatação do subsistema e o IO *Scheduler* em uso. Posteriormente, numa análise sobre estas especificações na execução dos ensaios, constatou-se que as politicas de utilização de *cache* e o tipo de acesso utilizado são as condicionantes com maior destaque para o impacto energético dinâmico do armazenamento secundário.

Como já era esperado depois do levantamento do estado da arte, assegurou-se que o tipo de acesso tem um impacto elevado no consumo de energia deste subsistema que, no entanto, é maior no disco HDD. Este facto deve-se ao facto do disco HDD apresentar uma estrutura de funcionamento mecânica que o coloca em desvantagem nas operações de mudança de posição. Por exemplo, no caso da leitura e escrita do disco HDD, o impacto energético do acesso aleatório foi de 2000 e 7000 vezes superior ao sequencial, respetivamente. No caso do disco SSD, este impacto do acesso aleatório foi de 18 e 12 vezes superior, respetivamente.

Em ambos os discos analisados, verificou-se uma notável melhoria na eficiência energética dos mesmos com a aplicação das técnicas de utilização da *cache*: *write-caching* e *read-ahead*. A técnica *write-caching* faz com que os dados a serem escritos, sejam previamente armazenados em *cache* e possam ser escritos no disco num momento mais oportuno. A técnica de *read-ahead* permite que o subsistema de armazenamento secundário dê uma resposta mais rápida aos pedidos de leitura realizados. Isto ocorre porque, com base no padrão de leitura, o subsistema tenta prever os próximos blocos a serem requisitados e lê e armazena esse volume de dados para pedidos futuros.

Neste capitulo constatou-se também que o alinhamento das partições do subsistema desempenha um papel essencial para um maior desempenho na escrita e leitura em disco. Um dos principais problemas do desalinhamento de uma partição acontece quando o sistema operativo pretende escrever em disco um bloco lógico que, ingenuamente se distribui em dois blocos físicos. Nesta situação, ao contrario de uma suposta operação simples de escrita, acontecem duas operações de "*read-modify-write*" que causam uma redução grande de desempenho e, consequentemente, uma menor eficiência energética do subsistema.

Num outro campo de estudo, depois dos ensaios realizados ao disco, pôde-se concluir duas situações interessantes relativas a dois indicadores analisados no decorrer dos mesmos. Primeiro, verificou-se que o indicador *await* fornecido pela ferramenta IOSTAT apresenta valores sugestivos sobre o tipo de acesso dos pedidos que o disco está a resolver. Com base na informação sobre o tempo de espera de resolução de um pedido enviado, existe a possibilidade de se prever se o disco está a

realizar escritas aleatórias ou sequencias com base neste indicador. Um outro facto que se concluiu foi sobre a solução pouco viável de estimar o consumo de energia do subsistema de armazenamento secundário unicamente com base no indicador de taxa de transferência de leitura e escrita em disco. Este indicador, também fornecido pela ferramenta IOSTAT, suscitou maior interesse de estudo neste projeto depois de ter sido referido como o único recurso utilizado para estimar o consumo de energia do subsistema, num modelo relacionado com este projeto, por parte do autor Bohra [139].

Numa análise generalizada que compara o impacto energético entre a tecnologia HDD e SSD, constatou-se que o disco HDD apresenta uma potência consumida bem mais elevada nas suas operações. Na configuração mais tradicional, que inclui a utilização das técnicas de *cache* e de sistema de ficheiros, o disco HDD apresenta um consumo de energia 3 vezes superior ao do disco SSD no caso do acesso sequencial e 300 vezes superior no caso do acesso aleatório.

Finalmente, os ensaios realizados foram muito proveitosos para o desenvolvimento do modelo de consumo de energia. Através deles, foram analisadas várias situações de funcionamento do subsistema de armazenamento secundário e explorou-se melhor algumas condicionantes importantes que podem interferir na eficácia e fiabilidade deste modelo.

5. Validação do Modelo

Para que um modelo seja fiável e haja uma apreciação global mais concreta sobre o mesmo, é necessário que este passe por um processo de validação. O objetivo deste capitulo é explorar os dados recolhidos e validar o modelo desenvolvido. A avaliação e validação do modelo será realizada com base numa ferramenta que dispõe de um conjunto extensivo de variados exercícios de desempenho. Sendo o modelo mais focado no subsistema de armazenamento secundário, optou-se por recorrer a exercícios que coloquem este subsistema a realizar os seus diversos tipos de operação, tanto a nível de leitura como de escrita. Posteriormente pretende-se encontrar pontos de possíveis melhorias.

O presente capitulo encontra-se dividido em três secções. Na primeira secção, é apresentada a metodologia utilizada para realizar a validação do modelo. Esta metodologia é feita com base em duas diferentes ações de validação, com base na tecnologia ACPI *Smart Battery* e na medição física do subsistema de armazenamento secundário através de equipamento externo. Na secção seguinte, são apresentados os resultados obtidos da validação com base nas duas técnicas anteriormente apresentadas. Por ultimo, na terceira secção, é analisada, de uma forma global, a fiabilidade e precisão do modelo e o seu potencial para uma adaptação a outros ambientes informáticos.

5.1 Metodologia

A metodologia adotada, para a validação do modelo de duas vertentes (HDD e SSD), baseia-se na tecnologia ACPI *Smart Battery* e em medições físicas ao disco através de equipamento externo ao sistema. O exercício de validação foi feito com a execução de ferramentas de *benchmarking* independentes ao projeto. As ferramentas de *benchmarking* a utilizar deviam apresentar exercícios focados no subsistema de armazenamento secundário. Durante esse período de execução das ferramentas, os dados fornecidos pelo modelo e pelas duas técnicas de validação foram recolhidos e estudados. A técnica de validação com recurso ao indicador de ACPI *Smart Battery* é proveitosa para a análise do modelo no seu desempenho de estimar o consumo energético total do sistema. A técnica de validação com recurso a medições físicas externas, permite a validação da componente do modelo que estima a energia consumida pelo subsistema de armazenamento secundário. Desta forma, estas duas validações completam-se ao cobrirem duas componentes diferentes de estimativa de energia do modelo.

Neste processo de validação, a técnica de redireccionamento da biblioteca do libC (LD_PRELOAD) foi utilizada para se capturar as funções executadas pela aplicação de *benchmarking*. Através da recolha destes dados, é possível perceber-se que tipo de operações são executadas sobre o sistema operativo e, consequentemente, que movimentos de IO ocorrem com essa atividade. No modelo, o cálculo do consumo de energia do componente de armazenamento secundário foi feito com base nos dados recolhidos através desta técnica.

5.1.1 Ferramentas de *Benchmarking*

Para validar o modelo, recorreu-se a aplicações de *benchmarking* desenvolvidas por outras entidades que exercitem o subsistema de armazenamento secundário de forma espontânea. Ao serem aplicações externas ao projeto, tornam a validação mais desafiante, não havendo um conhecimento aprofundado acerca da forma que exercitam o disco e o submetem a elevadas cargas de trabalho. As aplicações escolhidas para a etapa de validação foram a *Bonnie* e *Bonnie++*.

A ferramenta *Bonnie++* [185] é desenvolvida em C++ e dá continuidade à sua versão inicial denominada de *Bonnie*. A ferramenta destina-se a executar exercícios simples de desempenho ao subsistema de armazenamento secundário e ao seu sistema de ficheiros implementado. A primeira versão, o *Bonnie*, foi escrito em C e incluía uma série de testes de IO que simulavam vários tipos de aplicação de bases de dados. O *Boonie++* divide-se em duas sequências de testes. A primeira é a série original de testes do *Bonnie* com operações de IO direcionadas para bases de dados, enquanto que a segunda sequência testa a leitura e escrita de muitos ficheiros pequenos. No total, são executados 12 testes, incluindo três tipos de escrita e leitura sequencial e de *seeks* aleatórios. Todas as operações realizadas através desta ferramenta recorrem ao sistema de ficheiros e abdicam das escritas e leituras em modo *raw*, algo que se identifica bastante com o tipo de utilização tradicional do subsistema. Além disto, neste processo de validação, para que haja uma grande semelhança ao quotidiano da utilização de um sistema, tanto a funcionalidade de *write-caching* como de *read-ahead*, são utilizadas normalmente e sem qualquer limitação.

5.1.1.1 Testes de Validação

Nesta etapa de validação, omitiram-se os testes de desempenho do sistema de ficheiros e, o foco da análise resumiu-se no comportamento e consumo energético do disco. Desta forma, os testes com utilidade foram separados e nomeados de forma a se distinguir a reação do disco e o seu impacto energético de uma forma mais fácil. O processo de validação distribuiu-se assim por 8 fases:

- **Primeira fase (T1):** a ferramenta de *benchmarking* ainda não está ativa e o sistema apresenta todos os subsistemas em estado *idle*. Esta etapa é importante para se perceber a fiabilidade do modelo ao nível mais estático, em que nenhuma atividade provocada pelo utilizador ocorre no sistema. Neste caso, o modelo só é avaliado ao nível da estimativa do disco, isto porque, esta etapa é utilizada para calibrar o mesmo ao nível do RAPL e, assim, obter-se o consumo estático do subsistema do CPU e da memória primária. No teste final (**T8**), volta-se a realizar o teste de *idle* com o modelo já calibrado e é possível verificar-se a sua fiabilidade na estimação do consumo total do sistema.
- **Segunda fase (T2):** a ferramenta de *benchmarking* atua sobre o sistema com operações "*putc*", em que, cada uma destas operações consiste na escrita de apenas um *byte* em disco. Esta fase é especialmente difícil por não ter sido estudada tão aprofundadamente nos ensaios realizados sobre ambos os discos. A execução das operações é feita ciclicamente por um determinado tempo e a escrita é sequencial. Como as operações escrevem a mínima quantidade de dados possível, há a tendência para ocorrer um elevado numero de operações por segundo. Consequentemente, verifica-se habitualmente um elevado consumo de energia por parte do CPU para a resolução do elevado numero de pedidos.
- **Terceira fase (T3)**: a ferramenta de *benchmarking* continua a atuar em operações de escrita sobre o disco, no entanto, recorre à escrita através da *syscall "write*()" com *dataset* fixo de 8KB. Esta atividade é das mais comuns na escrita em disco e foi previamente estudada nos ensaios realizados. As operações de *write* são executadas ciclicamente e a escrita é sequencial.
- **Quarta fase (T4)**: são executadas operações de leitura e escrita alternadamente em disco. O *dataset* estabelecido para ambas as operações é de 8192 *bytes*. Nesta etapa, os dados são lidos numa determinada posição de um ficheiro e escritos noutra posição diferente, desta forma, verifica-se uma recolocação dos dados. Logicamente, para que aconteça esta reposição, ocorre a execução da *system call "lseek"* entre a leitura e a escrita dos dados. Apesar disso, na análise realizada, considera-se este exercício como sequencial devido ao padrão fixo existente nas operações. Ou seja, existem duas posições de memória, uma de escrita e outra de leitura, que se movimentam sequencialmente e, assim, é criado um padrão de operações ao qual o disco consegue prever futuras requisições e tem a possibilidade de otimizar o seu desempenho.
- **Quinta fase (T5)**: a ferramenta de *benchmarking* executa sucessivas operações sequenciais de leitura. Estas operações são feitas utilizando a função de C *getc*()

em que é lido apenas um *byte* por execução. À semelhança do que acontece no T2, o CPU tem tendência a ter um consumo bastante elevado devido à resolução do elevado numero de pedidos que acontece no exercício.

- **Sexta fase (T6)**: a ferramenta de *benchmarking* provoca uma leitura sequencial ao disco, através da função *read*() pertencente à *libC*. Estas operações são realizadas de forma contínua durante um certo período de tempo e, cada uma destas, apresenta um *dataset* de 8192 bytes. Este exercício assemelha-se ao ensaio que foi realizado neste projeto, relativo à leitura sequencial utilizando a técnica de *read-ahead*.
- **Sétima fase (T7)**: foram executadas operações de escrita e leitura, de forma alternada e totalmente aleatória. Nas operações de leitura era utilizado um *dataset* de 4096 *bytes* e, nas operações de escrita, o *dataset* era de 8192 *bytes*. Este exercício é totalmente inovador em relação a todos os ensaios realizados, no entanto, o modelo foi desenhado de forma a corresponder positivamente à atividade.
- **Oitava fase (T8)**: é novamente realizado um exercício de *idle* ao sistema que coloca à prova a potencialidade do modelo a estimar o consumo de energia do disco e do sistema total nestas condições. Enquanto no T1, devido ao modelo necessitar de ser calibrado, este teste só permitia a estimativa do consumo do disco, nesta fase, é possível verificar-se a qualidade de desempenho do mesmo na totalidade. Neste teste, o utilizador não pode provocar qualquer atividade no sistema operativo e o sistema deve apresentar uma atividade estática.

5.1.2 Validação com Recurso ao Indicador ACPI *Smart Battery*

A validação com base nos valores fornecidos pelo indicador ACPI *Smart Battery* é um processo que promove a não dependência de *hardware* adicional para se obter uma garantia de fiabilidade do modelo. Durante todo o projeto, definiu-se um padrão que passa por evitar qualquer investimento a nível de *hardware* para se chegar a um produto final. Desta forma, esta validação acaba por ser essencial para se provar a possibilidade de validar um modelo com base nos recursos disponibilizados pelo *hardware* existente no sistema. A larga escala, um modelo que seja desenvolvido nestas condições, requer um investimento muito mais pequeno e acessível a qualquer solução.

Como foi explicado anteriormente, o ACPI *Smart Battery* disponibiliza a informação do consumo total do sistema. Como o modelo também deve ser capaz de

disponibilizar essa informação, este indicador é um ponto de referência comparativo para análise da eficácia deste mesmo modelo. Para a validação do modelo com base na tecnologia ACPI *Smart Battery*, são executadas ferramentas de *benchmarking* e, em simultâneo, são recolhidos os vários indicadores necessários para o processo. Depois, realiza-se uma análise comparativa entre os resultados de consumo total obtidos pelo modelo e pelo indicador da tecnologia de ACPI.

5.1.3 Validação com Recurso a Medições Físicas Externas

A validação do modelo elaborado com base em medições físicas externas ao subsistema de armazenamento secundário é uma etapa que se pode considerar de importância para esta fase final do projeto. Como referido anteriormente, as medições físicas têm um custo elevado em grande escala e torna-se inviável que esses equipamentos façam parte de um investimento para monitorizar grandes infraestruturas. Desta forma, neste projeto, evitou-se criar qualquer dependência desta solução para o desenvolvimento do modelo, visto que, o desafio era cortar nesses custos e fazer um modelo leve e eficaz a larga escala. No entanto, a medição física desempenha nesta etapa um papel fundamental para se assegurar que este modelo pode crescer a larga escala sem a necessidade de qualquer instrumentação externa ao *hardware* existente.

Neste projeto, recorre-se a esta solução para se assegurar a eficácia, precisão e fiabilidade do modelo com base em medições físicas destinadas especificamente para a validação. Assim, colocou-se o modelo à prova, comparando-o com valores efetivamente medidos através de equipamento externo a fim de se garantir que a estimativa do consumo de energia está próxima da realidade. Esta validação é realizada com base na execução de ferramentas de *benchmarking* para se comparar a resposta do modelo e da medição física recolhida às várias situações de exercício do disco.

5.1.3.1 Conexão de Alimentação Energética SATA

Como se sabe, este modelo foi desenvolvido em torno de dois componentes diferentes de armazenamento secundário. Ambos os componentes apresentam uma interface de conexão SATA e utilizam os mesmos recursos energéticos através da mesma. A conexão SATA apresenta duas entradas diferentes: uma com a finalidade de alimentar energeticamente o componente e, a outra, sendo responsável pela troca de

dados entre o componente e o sistema. Para se atingirem os objetivos traçados neste setor, é necessária apenas a utilização da conexão de energia SATA.

Como se pode ver na Figura 131, este conector apresenta uma estrutura *pin-out* especial de 15 pinos que pode chegar a ser alimentado pelo máximo de 5 fios que se dividem em três diferentes voltagens para além do sinal GND (Ground) [186]. No conector SATA de energia, cada cabo é ligado a três *pins* terminais e, essa numeração de *pins* não tem a mesma ordem dos fios que a ele chegam.

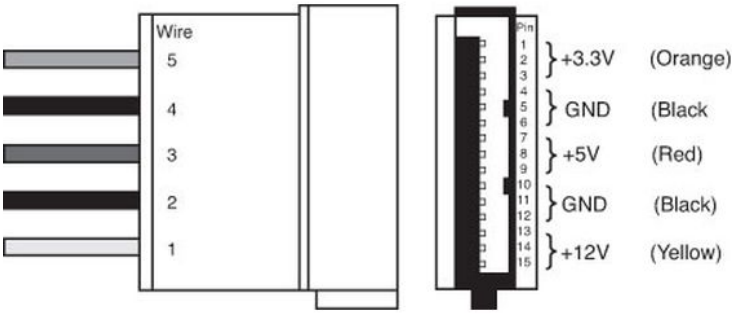


Figura 131 - *Pin-out* do conector SATA de Energia [186].

O cabo de conexão SATA de energia utilizado neste projeto está representado na Figura 132.

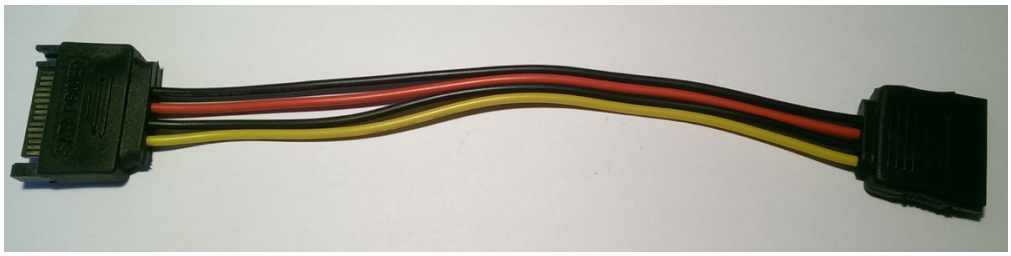


Figura 132 - Cabo SATA de energia utilizado.

Como se pode verificar, este cabo apresenta apenas quatro fios diferentes: dois de cor preta, um vermelho e outro amarelo. Desta forma, os sinais existentes para alimentar o subsistema são apenas os indicados na Tabela 11.

Tabela 11 - Tabela descritiva do pin-out do cabo SATA de energia.

Cor	Sinal
Preto	GND
Vermelho	+5V
Preto	GND
Amarelo	+12V

Apesar deste cabo não incluir o sinal de 3.3V, isto não é um problema visto que grande parte dos equipamentos atuais não necessitam do sinal de +3.3V e utilizam apenas +5V e +12V consoante as suas necessidades [186]. Segundo as fichas de especificações dos dois componentes de armazenamento secundário que foram colocados em prova, ambos sugerem que recorrem apenas ao sinal de +5V para o seu funcionamento [78] [150].

5.1.3.2 Equipamento de Medição

O processo de medir fisicamente o consumo de energia do armazenamento secundário requer alguns cuidados para que seja possível reter e avaliar os resultados no final. A solução escolhida passa por intercetar o cabo SATA de alimentação que parte do sistema e termina no componente. Desse cabo já são conhecidas as tensões de funcionamento de cada fio e, por isso, resta apenas conhecer a corrente lá existente para se possa concluir a potência consumida. Nos dois discos que estão em estudo, ambos apenas utilizam o fio de 5V. Segundo as análises realizadas para o projeto e os dados apresentados nos *datasheets* dos discos, a potência de ambos não ultrapassa os 5W. Para que seja possível entender melhor as variações, é dada uma margem de intervalo e, por isso, o sensor deve ser capaz de ler a corrente compreendida entre 0A e 1,2A.

Logicamente, por ser necessário tratar os dados no momento, a fim de armazenar corretamente os mesmos, a opção de utilizar um amperímetro vulgar não é uma solução viável. Desta forma, para realizar estas medições, optou por se utilizar um Arduino UNO e um sensor de corrente. O esquemático da relação entre estes dois componentes e o cabo SATA de energia pode ser consultado na Figura 133.

Numa primeira fase, pela sua facilidade de aquisição, foi experimentado utilizar o sensor de corrente Atto Pilot de 90 Amperes [187]. No entanto, verificou-se que este sensor era insuficiente para correntes tão reduzidas como as procuradas para este projeto. Este sensor de *output* analógico até 3,3V, está preparado para correntes que possam atingir um máximo de 90 Amperes e, a sua precisão para correntes pequenas é relativamente baixa. O intervalo de corrente procurado para este projeto encontra-se entre 0 e 1,2 Amperes e, no sinal analógico de output do sensor Atto Pilot, isso representa um pequeno intervalo de 0 a 0,05V.

Optou-se por explorar melhor este mercado para encontrar um sensor suficientemente preciso para as necessidades do projeto. O sensor de corrente que mais se identificou com o procurado foi o *Low Current Sensor Breakout* [188]. Este sensor é comercializado pela empresa SparkFun e tem a capacidade de medir a corrente independentemente de o sinal ser contínuo ou alternado. O sensor utilizado

recorre a um circuito integrado ACS712 destinado a medir uma corrente até 5 amperes. O sensor faz se acompanhar de uma fase de amplificador operacional para controlar o ganho, de maneira a ser possível medir correntes mais baixas de uma forma mais precisa. O sensor *Low Current Sensor* expõe uma voltagem analógica que varia linearmente com a corrente medida.

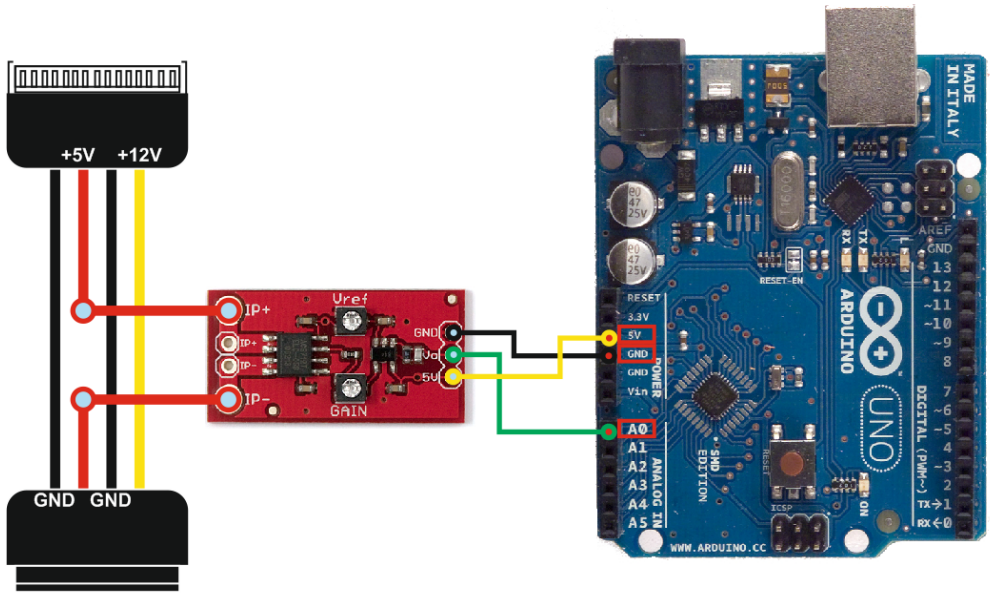


Figura 133 – Esquemático de ligações entre o cabo SATA de energia, o medidor de corrente e o Arduino (da esquerda para a direita) com o objetivo de medir a corrente existente no cabo de +5V.

Além disto, foi utilizado um Arduino UNO para ler a voltagem analógica apresentada pelo sensor e comunicar esses valores ao computador a ele conectado pela porta série. Esta foi a solução mais simples e suficiente para atingir os objetivos pretendidos de ler, informaticamente e de forma constante, o consumo energético do subsistema de armazenamento secundário. Para que o Arduino funcione conforme o pretendido deste projeto, foi desenvolvido um programa para este equipamento que fizesse leituras constantes à voltagem analógica e as enviasse de segundo a segundo. Como é possível ver no código-fonte presente na Figura 134, a leitura da entrada analógica A0 é feita a cada milissegundo e, seguidamente, é calculada a média dos últimos mil milissegundos que é, por fim, enviada para o sistema informático conectado ao Arduino. Decidiu-se implementar um filtro de leitura que se baseia em ler o valor analógico, uma vez por cada milissegundo e, ao fim de um segundo, é apresentada a média desta leitura. Esta implementação foi necessária por se ter verificado previamente alguma instabilidade nos valores obtidos aquando de uma única leitura por segundo. Depois desta alteração, o sinal recebido do sensor tornou-

se mais estável, isto é, deixaram de existir variações bruscas a cada segundo de leitura do valor digital obtido do conversor ADC.

```
int sensorPin_0 = A0;
int sensorValue_0 = 0;

void setup() {
        Serial.begin(9600);
}

void loop() {
        float average_a0 = 0;
        for(int i = 0; i < 1000; i++) {
                average_a0+=analogRead(sensorPin_0);
                delay(1);
        }
        average_a0/=1000;
        Serial.println(average_a0);
}
```

Figura 134 – Excerto de código-fonte do Arduino para leitura do sinal analógico do sensor de corrente.

Inicialmente, o sensor necessita de ser calibrado, a fim de se definirem limites máximos de medição de corrente para tornar o sensor o mais preciso possível para as correntes pretendidas. Além disso, o Arduino também precisa de ser consultado nestas calibragens do sensor para se perceber a relação dos valores apresentados pelo mesmo comparativamente com os amperes realmente obtidos. Através de todo este processo, é possível ajustar e, assim, conhecer o sinal analógico equivalente às correntes medidas pelo sensor. A calibragem do sensor é um procedimento que requer a utilização de algum equipamento como multímetros e uma fonte de alimentação. O circuito utilizado pode ser verificado na prática laboratorial da Figura 135. O circuito resume-se a uma simples fonte de tensão em que, entre o sinal positivo e negativo, há uma resistência onde acontece toda a queda de tensão. A resistência precisa de ser escolhida cuidadosamente visto que há o interesse em experimentar potências altas a ser dissipadas no componente. Primeiro, estudou-se as convencionais resistências que atingem uma potência dissipada máxima que varia entre 0,125W e 1W [189]. Verificou-se que essas resistências são insuficientes para adaptar no circuito de calibragem que apresenta tensões até 5V e correntes até 1,5A. Desta forma, utilizou-se uma resistência de potência *Vitrohm* 216-8, que apresenta as seguintes características: 1 *Ohm*, 10% de tolerância e potência dissipada máxima de 11W [190].

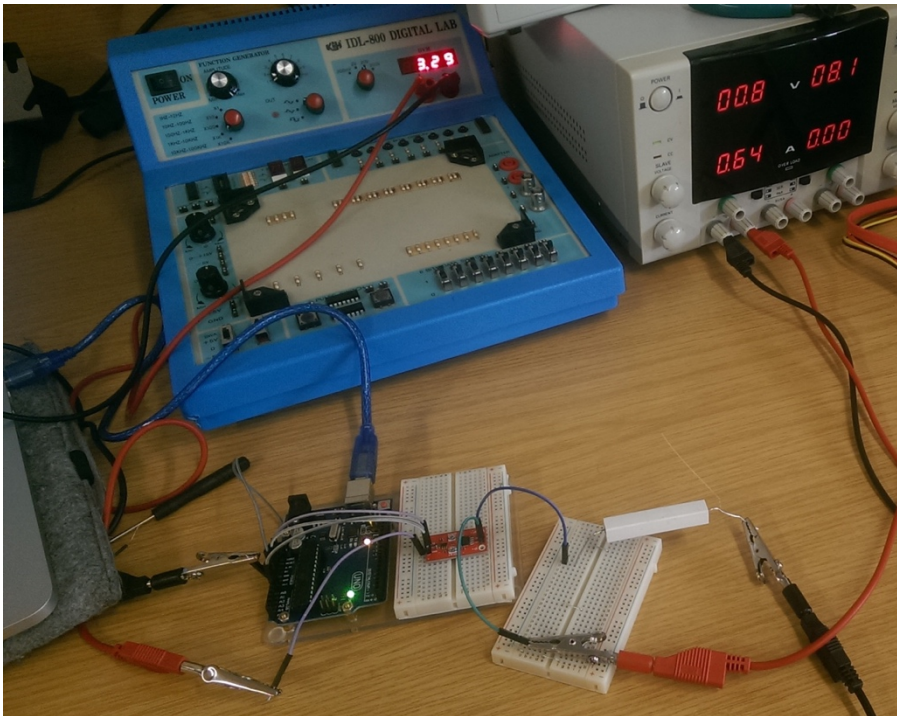


Figura 135 – Prática laboratorial - circuito utilizado para a calibração do sensor de corrente.

Para utilizar o sensor no circuito, este é aberto entre o sinal positivo e a resistência e coloca-se o sensor a ligar esses dois componentes. Então, o sinal positivo da fonte de tensão passa a ligar ao sinal positivo do sensor e a resistência liga-se ao sinal negativo do sensor. Depois, ajustando a tensão da fonte, é possível obter diversos valores de corrente para que se conseguia calibrar o sensor. O V_{out} do sensor para além de estar ligado ao Arduino durante este exercício, é ligado também a um voltímetro para haver uma melhor interpretação do sinal analógico de saída.

O documento informativo do sensor [188] sugere que, a consulta de apenas um valor de corrente, para além da corrente nula, seria suficiente por se tratar de uma função linear. No entanto, para se assegurar que o sensor é fiável nas medições que realiza, optou-se por experimentar vários valores de corrente e recolher cada V_{out} correspondente. Como é possível verificar no gráfico da Figura 136, foi medido o sinal analógico de saída do sensor para cada 0,1 amperes entre 0 e 1,8. Este foi o intervalo escolhido porque, para além de já ser um intervalo suficientemente preciso para as necessidades pretendidas, verificou-se alguma dificuldade no ajuste do ganho e V_{ref} para se conseguir um intervalo mais curto. Nesta análise, confirma-se também a linearidade da função, o que permite que se transcreva a função numa equação $y=mx+b$. O gráfico apresenta esses valores recolhidos e, através dele, pode-se também confirmar que se forma uma função linear. A sensibilidade média do sensor, depois de calibrado nestas configurações, é de 2,83 V/A. Desta forma, a função linear

pode ser descrita através da equação V_{out}=-2,83A+5,1. Por exemplo, na Figura 136 é possível verificar que, quando a corrente do circuito é de 0,64A, o V_{out} é de 3,29V. Com o medidor corretamente calibrado e preparado para medir a corrente, segue-se para a etapa de execução, em que se exercita o disco e se recolhe os devidos indicadores, tanto do modelo, como do instrumento de medição física.

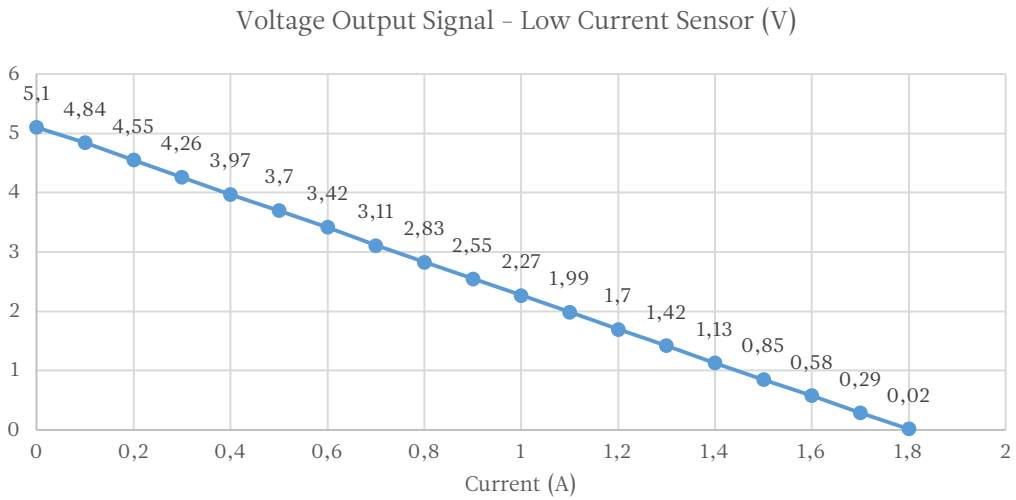


Figura 136 –Gráfico resultante da calibragem do sensor de medição de corrente, para medir correntes no intervalo entre os 0 e 1,8 A.

5.2 Resultados

Depois de se estabelecer uma metodologia, o processo de validação foi posto em prática e os resultados foram recolhidos durante o desenrolar do mesmo. Agora, apresentam-se os resultados recolhidos nas várias situações de operação que são causados pelas ferramentas de *benchmarking* escolhidas. Os resultados mais relevantes nesta etapa são, o indicador ACPI *Smart Battery*, as medições físicas realizadas ao subsistema de armazenamento secundário, e claro, o *output* do modelo de consumo de energia desenvolvido, tanto ao nível do disco como a nível global.

5.2.1 Disco HDD

No disco HDD, o processo de validação foi realizado sem qualquer imprevisto e os resultados finais foram calculados com sucesso. Neste processo o disco passou por um conjunto de testes distribuídos em várias etapas temporais. As várias etapas diferenciam-se pelo tipo de atividade que é realizada no sistema, no seu todo e, em particular, no disco. Consoante o exercício executado no sistema, cada etapa apresentou resultados diferentes que acabaram por corresponder à expetativa e estão

apresentados na Figura 137. Para uma análise mais aprofundada sobre a estimativa de consumo de energia do disco desenvolveu-se um gráfico que demonstra o consumo de energia medido fisicamente e o consumo estimado pelo modelo para o disco HDD, ao longo dos 8 exercícios de validação. Este gráfico encontra-se apresentado na Figura 138.

Na primeira etapa (**T1**), com o sistema em *idle* e com nenhuma atividade por parte do disco à exceção da rotatividade dos pratos, verificou-se que o modelo apresenta um valor extremamente próximo do realmente obtido através do ACPI e da medição física. Nesta etapa, não há qualquer ocorrência de *Input/Output Operations Per Second* (IOPS) e, por isso, o modelo baseia-se apenas no consumo base do disco em estado *idle*. Utilizando a solução de medição física desenvolvida, verificou-se que, durante este exercício de *idle*, o consumo do disco apresentou uma média de 1,5W, valor este que é muito idêntico àquele apresentado pelo modelo desenvolvido. No caso do consumo total do sistema, tanto a ferramenta ACPI como o modelo desenvolvido, sugerem um consumo de 10,21W. O modelo apresenta-se com uma taxa de erro muito próxima de 0 tanto ao nível do disco como ao nível do consumo total do sistema. Neste exercício, os resultados obtidos são bastante positivos e isso deve-se à inatividade do disco que permite uma maior facilidade de estimação de consumo de energia.

No segundo exercício (**T2**), as operações realizadas foram apenas de escrita e destacam-se pela simples escrita de um *byte* por cada execução. Durante esta atividade, foram executadas 53956875,6 operações por segundo. A maior evolução do impacto energético durante este exercício ocorreu no subsistema do CPU, sendo que, o subsistema aumentou em 400% o seu consumo e atingiu os 12,33W. O subsistema de armazenamento secundário apresentou uma média de consumo de 2,13W. O modelo estimou que o consumo deste subsistema tivesse uma média de 2W. Segundo o indicador do ACPI, o consumo total do sistema apresentou uma média de 20,33W. Já o modelo desenvolvido calcula um consumo de 19,69W do sistema global. O modelo, neste exercício em específico, apresenta uma taxa de erro de 6,19% para o armazenamento secundário e de 3,17% para o consumo energético total.

O exercício que se segue (**T3**), consiste numa atividade de escrita sequencial em blocos de 8192 *bytes* e é bastante idêntico àquele que foi ensaiado e estudado no capítulo anterior. Desta forma, os resultados obtidos através do modelo responderam a esta etapa positivamente. Durante este período, foram executadas 18391 operações de escrita por segundo, em que, cada uma, apresentou um *dataset* de 8192 *bytes*. Com base nas técnicas recorrentes, em média, o disco HDD consumiu 3,1 W e o sistema consumiu 13,1 Watts. Confrontando estes valores com aqueles obtidos no modelo de consumo de energia do HDD, verifica-se um erro máximo de 5,82% sobre o

subsistema de armazenamento secundário e um erro máximo de 1,42% sobre a estimativa do sistema total.

O exercício **T4** destaca-se por provocar uma atividade que não foi testada nos ensaios realizados no capitulo anterior. Nesta atividade, verificou-se a ocorrência de 9045 operações por segundo, tanto a nível de leitura como de escrita, com o *dataset* de 8KB. Neste teste, o consumo do CPU foi aquele que apresentou um maior crescimento comparativamente com o seu consumo em *idle*. No armazenamento secundário, a medição física apresentou um consumo médio de 2,95W e o modelo sugeriu um consumo de 3,25W. No caso do consumo total, a ferramenta ACPI *Smart Battery* indicou uma média de consumo de 12,74W, valor esse que é muito idêntico àquele apresentado pelo modelo. Com estes resultados, o modelo atingiu um erro de 10% no consumo total do disco e 0,02% no consumo total do sistema.

No exercício **T5**, realizou-se um tipo de operações que não foi previamente ensaiado e consiste na leitura sequencial em que, cada operação lê apenas um *byte*. A média de operações de leitura por segundo foi de 54504661 (IOPS). Nesta atividade, tal como na atividade T2, devido ao elevado numero de operações, o CPU atinge o maior crescimento de consumo de energia, com uma média de 12,76W. Em relação a outros indicadores, verificou-se que, o disco consumiu em média 2,25W e o sistema total consumiu uma média de 21,12W. O modelo estimou 2,13W para o disco e 20,23W para o sistema total. Com estes resultados, a taxa de erro do modelo foi de 5,31% para o disco e 4,21% para o sistema total.

Na etapa de *benchmarking* (**T6**), o disco executou uma média de 14549 operações por segundo com um *dataset* de 8192 *bytes*. Consequentemente, o disco apresentou o maior aumento de consumo, ao ascender mais de 1W, comparativamente com o seu estado em *idle*. No armazenamento secundário, a média do consumo fisicamente medido foi 2,71W, enquanto que, no modelo, obteve-se uma estimativa de 2,91W. No consumo global do sistema, a ferramenta recorrente do ACPI estima 12,67W, enquanto o modelo sugere 12,53W. Nesta etapa, o modelo apresentou uma taxa de erro de 7,26% para o consumo do disco e de 1,08% para o consumo do sistema no seu todo.

Na penúltima atividade (**T7**) ocorreu um processo aleatório de escrita e leitura. Assim, atingiu-se uma média de 146 operações de leitura por segundo, cada uma com um *dataset* de 4096 *bytes*. No caso da escrita, foram realizadas 7,33 operações por segundo, com *datasets* de 8192 *bytes*. Com todas as operações ocorridas, verificou-se através da medição física uma média de 2W de consumo de disco. Já o modelo estimou este subsistema com um consumo médio de 2,16W. No caso do consumo total do sistema, a ferramenta do ACPI sugeriu um consumo de 10,86W, enquanto que, o

modelo desenvolvido, estimou 10,85W. A taxa de erro obtida na estimativa do consumo através do modelo foi de 8,71% para o disco e de 0,14% para o sistema total.

Na atividade final (**T8**) em que o sistema está em *idle* e não há qualquer operação ativa em disco, obteve-se uma média de consumo de 1,512W através das medições físicas e de 1,5W através do modelo de estimação de consumo de energia. A interface ACPI indicou um consumo total de 10,23W, valor esse que está próximo dos 10,14W estimados pelo modelo desenvolvido neste projeto. A taxa de erro existente nesta etapa é de 0,8% para o consumo do disco e de 0,9% para o consumo do sistema total.

Average Calculation	Time	T1	T2	T3	T4	T5	T6	T7	T8
	Test	idle	putc	write	read'n'write	getc	read	read'n'write	idle
	Mode	-	seq	seq	seq (*)	seq	seq	ran	-
READ	IOPS	0	0	0	9045	54504661,12	14549,275	146,167	0
	Bytes per Op	0	0	0	8192	1	8192	4096	0
	Bytes per Sec	0	0	0	74096640	54504661,12	119187661	598700,03	0
WRITE	IOPS	0	53956875,6	18390,854	9045	0	0	7,333	0
	Bytes per Op	0	1	8192	8192	0	0	8192	0
	Bytes per Sec	0	53956875,6	150657876	74096640	0	0	60071,936	0
POWER	System (ACPI)	10,21	20,33	13,059	12,74	21,12	12,67	10,86	10,233
	CPU (RAPL)	3,393	12,327	4,14	4,073	12,757	4,233	3,367	3,325
	DRAM (RAPL)	0,734	0,774	0,868	0,836	0,765	0,81	0,733	0,736
	DISK (PHYS)	1,502	2,134	3,102	2,95	2,246	2,71	1,989	1,512
MODEL	Disk	1,5	2,00179894	3,2824416	3,251112	2,126803603	2,90662391	2,1621959	1,5
	Total	10,21	19,6857989	12,873442	12,743112	20,2318036	12,5326239	10,845196	10,144
ERROR	Disk	0,13%	6,19%	5,82%	10,21%	5,31%	7,26%	8,71%	0,79%
	Total	0,00%	3,17%	1,42%	0,02%	4,21%	1,08%	0,14%	0,87%

Figura 137 –Tabela com os resultados dos 8 exercícios realizados ao disco HDD, acompanhados pelo consumo de energia estimado pelo modelo.

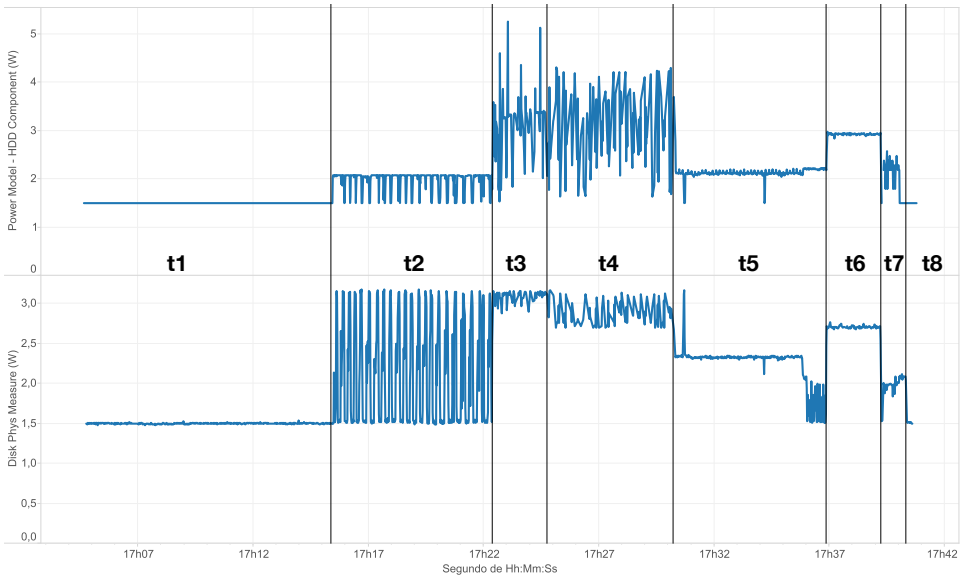


Figura 138 – Gráfico que apresenta o consumo de energia medido fisicamente e o consumo estimado pelo modelo para o disco HDD, ao longo dos 8 exercícios de validação (t1-t8).

5.2.2 Disco SSD

No disco SSD, o processo de validação foi realizado sem grandes problemas e os resultados finais foram calculados com sucesso. Apesar disso, o teste **T7** foi excluído das análises de resultados, isto porque, este teste era executado num demasiado reduzido espaço de tempo e impedia a obtenção de uma quantidade de resultados suficiente e esclarecedora.

Neste disco existe uma maior facilidade de funcionamento do modelo, isto porque, o disco tem um funcionamento totalmente eletrónico e os imprevistos de escrita ou leitura têm por essa razão um impacto menor.

As várias etapas diferenciam-se pelo tipo de atividade que é realizada no sistema, no seu todo e, em particular, no disco. Consoante o exercício executado no sistema, cada etapa apresentou resultados diferentes que, tal como no disco HDD, acabam por corresponder à expetativa. Esses resultados estão apresentados na Figura 139. Para uma análise mais aprofundada sobre a estimativa de consumo de energia do disco desenvolveu-se um gráfico que demonstra o consumo de energia medido fisicamente e o consumo estimado pelo modelo para o disco SSD, ao longo dos 7 exercícios de validação. Este gráfico encontra-se apresentado na Figura 140.

Na primeira etapa (**T1**), o sistema apresentou-se em modo *idle*, ou seja, o utilizador não requisitou qualquer atividade ao sistema operativo. Neste modo, o modelo é habitualmente mais fiável devido à reduzida atividade dos subsistemas que, por sua vez, torna a estimação de energia consumida mais fácil. Nestas condições, o disco SSD consumiu uma média de 0,352W. No caso do consumo de energia do sistema total, a ferramenta *Smart Battery* da interface ACPI sugeriu uma média de consumo de 8,97W. Através destes valores, foi possível calibrar o modelo, ao nível do subsistema do CPU e de memória primária, para as análises futuras.

Na segunda etapa (**T2**), foi realizada a atividade de escrita sequencial em disco com operações de apenas um *byte*, através da operação *putc()*. Neste exercício a média de consumo de energia do CPU foi a mais elevada, com 13,39W, devido ao alto número de operações executadas. Através da técnica de medição física aplicada, verificou-se que o consumo médio do disco SSD foi de 1,13W. O modelo desenvolvido apresenta um valor idêntico, sugerindo um consumo médio do disco de 1,22W. A interface ACPI constata um consumo médio do sistema de 20,53W, valor este que é um pouco superior aos 20,04W apresentados pelo modelo. A taxa de erro do modelo nesta etapa foi de 8,06% para o disco e de 2,38% para o sistema total.

Na fase seguinte de *benchmarking* (**T3**), foi realizada a escrita sequencial em disco com o dataset de 8192 *bytes*. Neste exercício, segundo a medição física realizada, o consumo médio do disco foi de 3,22W, enquanto que, o modelo

desenvolvido apresentou uma média de 3,36W. O consumo total do sistema foi de 15,35W segundo a interface ACPI e, de 15,79W segundo o modelo. A taxa de erro do modelo nesta etapa foi de 4,47% para o disco e de 4,87% para o sistema total.

No exercício **T4**, foi realizado um conjunto de operações de leitura e escrita que se intercalam entre si. Nesta atividade, foram realizadas 30840 operações por segundo, tanto a nível de escrita como de leitura, com um padrão estabelecido e um *dataset* de 8192 *bytes*. Com base nas medições realizadas, o consumo médio do disco SSD foi de 2,92W, um valor que é pouco inferior aos 3,1W estimados pelo modelo. No caso do consumo total do sistema, a interface ACPI sugeriu um consumo de 15,35W, enquanto o modelo estima um consumo de 15,13W. A taxa de erro neste exercício foi de 5,78% para o consumo do disco SSD e de 1,41% para o consumo total do sistema.

Na etapa seguinte (**T5**), o disco SSD realizou operações sequenciais de leitura em que cada uma lia apenas um *byte*. Esta atividade destaca-se por executar um mais elevado numero de operações e provocar um maior consumo do CPU, que neste caso atingiu os 14,1W. O medidor físico indicou que o disco consumiu uma média de 1,14W, enquanto que, o modelo, sugeriu uma média de 1,09W. No consumo total do sistema, a interface ACPI indica 21,38W e o modelo apresenta uma estimativa de 20,62W. A taxa de erro do modelo para este teste é de 5,78% para o disco e de 1,41% para o sistema total.

No exercício **T6**, foram realizadas operações de leitura sequencial ao disco, com um *dataset* de 8192 *bytes*. Nesta atividade, foram lidos 515MB por segundo e foram executadas 62890 operações de leitura por segundo. Com isto, o disco e o sistema, apresentaram um consumo energético médio de 2,42W e 15,95W, respetivamente. No caso do modelo, estimou-se um consumo de 2,59W para o disco e de 15,39 para o sistema. Desta forma, a taxa de erro do modelo para o disco foi de 6,93% e, para o sistema, foi de 3,53%.

Finalmente, o exercício T**8** foi realizado com o sistema em *idle* e sem qualquer atividade provocada pelo utilizador. Depois dos indicadores do RAPL terem sido calibrados no teste T1, é nesta etapa que se verifica a fiabilidade do modelo em *idle* para o indicador do consumo de energia total do sistema. Neste exercício, o disco apresentou uma média de consumo de energia de 0,36 e o sistema apresentou um consumo de 8,86W. Comparativamente com os dados estimados pelo modelo, verifica-se uma taxa de erro de 3,35% para o consumo do disco e um erro de 0,18% para o consumo total do sistema.

Average Calculation	Time	T1	T2	T3	T4	T5	T6	T8
	Test	idle	putc	write	read'n'write	getc	read	idle
	Mode	-	seq	seq	seq (*)	seq	seq	-
READ	IOPS	0	0	0	30840,2941	60282411	62889,5	0
	Bytes per Op	0	0	0	8192	1	8192	0
	Bytes per Sec	0	0	0	252643689	60282411	515190784	0
WRITE	IOPS	0	56811737	56536,425	30840,3177	0	0	0
	Bytes per Op	0	1	8192	8192	0	0	0
	Bytes per Sec	0	56811737	463146394	252643882	0	0	0
POWER	System (ACPI)	8,97	20,53	16,6	15,351	21,382	15,95	8,86
	CPU (RAPL)	3,21	13,387	6,543	6,2	14,103	7	3,117
	DRAM (RAPL)	0,734	0,777	1,233	1,19	0,768	1,143	0,733
	DISK (PHYS)	0,352	1,131	3,216	2,92	1,142	2,421	0,358
MODEL	Disk	0,37	1,222176055	3,35964615	3,08888157	1,09338893	2,58874156	0,37
	Total	8,97	20,04217606	15,7916462	15,1348816	20,6203889	15,3877416	8,876
ERROR	Disk	5,11%	8,06%	4,47%	5,78%	4,26%	6,93%	3,35%
	Total	0,00%	2,38%	4,87%	1,41%	3,56%	3,53%	0,18%

Figura 139 - Tabela com os resultados dos 7 exercícios realizados ao disco SSD, acompanhados pelo consumo de energia estimado pelo modelo.

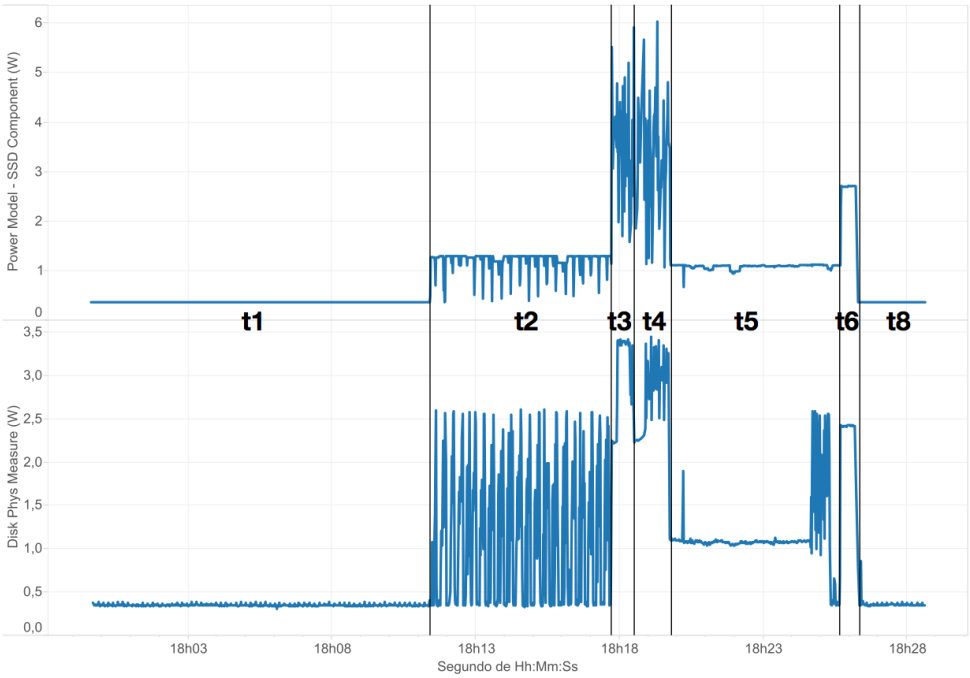


Figura 140 - Gráfico que apresenta o consumo de energia medido fisicamente e o consumo estimado pelo modelo para o disco SSD, ao longo dos 7 exercícios de validação (t1-t6, t8).

5.3 Apreciação Global

O modelo desenvolvido, tanto para o disco HDD como para o disco SSD, apresenta resultados bastante positivos. Verifica-se através deste processo de validação que o modelo demonstra ser capaz de responder bem a diferentes desafios daqueles estudados na fase de ensaios e estudo dos discos. Desta forma, considera-se que, a quantidade e o tipo de ensaios definidos, foram suficientes para se desenvolver

um modelo estável e capaz de estimar o consumo de energia do subsistema de armazenamento secundário. O *output* do modelo ao nível do consumo de energia total do sistema é também outro ponto fulcral para o sucesso deste projeto. A estimativa desse indicador depende do bom funcionamento da ferramenta RAPL e, também, da qualidade do modelo a estimar o consumo de energia do subsistema de armazenamento secundário. Segundo vários artigos, a ferramenta RAPL é bastante fiável e, por isso, se o modelo apresentar bons resultados de estimação de energia ao nível do subsistema de armazenamento secundário, verifica-se também a existência de bons resultados ao nível do consumo global do sistema.

Neste processo de validação, em ambos os discos, verificou-se que as taxas de erro obtidas são bastante reduzidas nas etapas de *idle*, sendo um pouco mais elevadas nas etapas em que o disco é exercitado. Isto acontece porque, no estado passivo do sistema, os níveis de consumo de energia e de atividade do sistema são pequenos e a oscilação é bastante reduzida. Esses fatores permitem que, nestas condições, o modelo apresente um melhor desempenho e se aproxime mais dos consumos reais realmente ocorridos. Para além disto, verificou-se também que o modelo apresenta menores taxas de erro sobre o consumo total do sistema do que sobre o subsistema de armazenamento secundário em específico. Isto deve-se à dificuldade obtida na estimação de energia do subsistema de armazenamento secundário e ao bom desempenho da ferramenta RAPL na estimativa do consumo de energia dos restantes subsistemas incluídos. Desta forma, a taxa de erro incide maioritariamente sobre a estimativa do consumo de energia do disco e, por sua vez, tem um impacto mais pequeno no calculo da estimativa do consumo total energia do sistema.

O **modelo desenvolvido para o disco HDD** apresentou taxas de erro bastante positivas, no entanto, estas variam consoante o exercício em causa. Por exemplo, na execução do modelo aquando do sistema está em *idle*, a taxa de erro do disco e do consumo total do sistema varia entre 0 e 1%. No caso do sistema estar ativo, isto é, com o utilizador a provocar atividade no mesmo, a taxa de erro do modelo para estimar o consumo do disco pode variar entre 5% e 10%. Já a estimação do consumo global de energia do sistema varia apenas entre 1 e 4%. Os melhores resultados foram obtidos nos exercícios de desempenho mais simples com atividades de escrita ou leitura sequencial.

O **modelo desenvolvido para o disco SSD** apresentou resultados um pouco mais fiáveis do que o modelo para o disco HDD. Há várias razões possiveis para este sucedido, sendo que se destaca o facto do equipamento ser totalmente elétrico e haver uma menor oscilação energética. Com o sistema em *idle*, o modelo apresenta uma taxa de erro entre 3 e 5% para a estimativa do consumo do disco e entre 0 e 1% para a estimativa do consumo do sistema. No caso de existir atividade no sistema, o modelo

apresenta uma taxa de erro entre 3% e 8% para o consumo do disco e entre 1% e 5% para o consumo total do sistema.

6. Conclusão

Este projeto enquadra-se numa importante temática da atualidade - a eficiência energética de plataformas computacionais de larga escala - que ao longo dos anos se tem tornado mais relevante com o elevado crescimento do volume de dados processado através de serviços *Cloud*. Neste sentido, e seguindo a metodologia de investigação *action research*, foi desenvolvido um modelo para estimar o consumo de energia em sistemas Linux. O modelo apresenta duas vertentes, dependendo da tecnologia utilizada no subsistema de armazenamento, sendo que, os discos a testar podem ser tanto SSD ou HDD.

Este capitulo divide-se em duas secções. Na primeira secção será apresentada uma síntese do trabalho prático efetuado. Na segunda secção são apresentadas as linhas de investigação destacáveis que poderão ser executadas num trabalho futuro sobre este tema.

6.1 Considerações Finais e Contributos

Como já foi referido ao longo deste documento, a gestão e monitorização de energia num sistema apresenta-se como um desafio da atualidade e um importante tópico ao qual é preciso dar resposta. Este tema tem vindo a crescer juntamente com o aumento exponencial do número de sistemas computorizados no mundo. Atualmente, existe uma grande afluência aos gadgets e, também por isso, há uma maior sincronização de dados em *Cloud*, sendo esta uma das causas do aumento do número de *data centers* no mundo. Com isto, verifica-se também uma maior preocupação por parte dos fabricantes destes equipamentos a nível da eficiência energética dos mesmos. Por exemplo, a importância de reduzir o custo energético de um data center de TI ou de se obter um maior rendimento e autonomia da bateria de um portátil, são razões cada vez mais fundamentais para que o tema de gestão e monitorização do consumo de energia faça parte das politicas das empresas tecnológicas. Com uma visão sobre a atualidade e o futuro deste ramo, considera-se que qualquer contributo para este tema pode ser importante não só para o ramo de TI como também para o ambiente.

Deste modo, o trabalho realizado consistiu no desenvolvimento de um modelo de estimativa de consumo de energia parametrizado. Através desse modelo consegue-se obter uma estimativa do consumo energético de um sistema, tanto a nível global como categorizado pelos principais subsistemas. No caso da estimativa do modelo no subsistema de armazenamento secundário, é também possível consultar o seu consumo energético classificado pelos vários processos ativos.

Para a realização deste projeto, houve um estudo prévio sobre os componentes, tanto a nível de *hardware* como de *software*, que têm uma relação com o tópico de energia num sistema. Desta forma, foi estudada a distribuição do consumo de energia de um sistema pelos vários subsistemas e o impacto que estes subsistemas têm para o consumo dinâmico de energia. Nesta etapa concluiu-se que os principais subsistemas que deveriam ser incluídos no modelo para representar o consumo de energia dinâmico são o CPU, memória primária e armazenamento secundário. Depois, foram estudadas as interfaces e ferramentas existentes para gestão e monitorização de energia no sistema, e em particular, nos mais relevantes subsistemas. Entre elas, destacou-se o ACPI, RAPL, HDPARM, *sysfs* e *procfs.* Por fim, estudou-se os modelos já desenvolvidos que estão minimamente relacionados com o que se pretende realizar. No final desta etapa, concluiu-se que o armazenamento secundário é o subsistema com menos informação energética disponível. Desta forma, este subsistema acabou por ser estudado aprofundadamente para se conseguir estimar o seu consumo de energia a vários níveis.

Com base nestas conclusões obtidas no levantamento do estado da arte, desenvolveu-se um modelo com base no estudo da resposta do sistema, ao nível energético, a determinados exercícios de desempenho. Para se estudar essa resposta, foram recolhidos os indicadores considerados mais adequados às necessidades do projeto. Entre eles, destaca-se a recolha do indicador de consumo de energia total do sistema, do consumo de energia do CPU e da memória primária. Nesta fase, deu-se especial importância à identificação de padrões relevantes de acesso a disco para se perceber melhor o seu real comportamento do mesmo. Este passo foi importante para se conseguir estimar o consumo de energia relativo a esses padrões do subsistema.

Depois do estudo do consumo energético do sistema por componente de *hardware*, estudou-se e definiu-se um método de interseção dos processos no acesso a disco. Através disto, tornou-se possível descodificar que tarefas está cada processo a gerar no disco e, assim, há uma maior facilidade em atribuir diferentes pesos energéticos a cada um dos processos ativos. Este procedimento foi essencial e bastante conclusivo para que o modelo desenvolvido fosse capaz de calcular a fatura energética de acesso a disco por processo.

Após a fase de testes e ajustes em ambiente laboratorial, resolveu-se um modelo final capaz de estimar o consumo de energia de um sistema em várias categorias. O modelo desenvolvido apresenta taxas de erro bastante positivas para o âmbito do projeto e dos objetivos traçados. A taxa de erro de estimativa do consumo de energia pode chegar aos 5% no consumo total do sistema. No caso do consumo apenas do subsistema de armazenamento secundário, a taxa de erro pode chegar aos 10% no caso do disco HDD e aos 8% no caso do disco SSD.

Por último, pode-se ainda dizer que foi concretizada a ideia de implementar um modelo que depende apenas dos recursos disponibilizados pelo sistema, o que permite uma larga redução do investimento para o estudo e execução do modelo. Este facto é ainda mais vantajoso quando se pretende que o modelo seja introduzido num conjunto de sistemas em larga escala, em que a utilização de qualquer recurso externo ao sistema tem um custo bem maior.

6.2 Trabalho Futuro

Este projeto entende-se como uma base de suporte para desenvolver e avaliar estratégias de gestão de energia para sistemas informáticos. O crescimento dos serviços em nuvem e o aumento das infraestruturas computacionais de larga escala, tornam o estudo de eficiência energética mais relevante e prioritário a curto prazo. A fim de se conseguir um futuro mais verde para os *data centers*, é importante estabelecer mecanismos de poupança de energia como politicas de gestão de recursos e um melhor controlo de operações dos sistemas.

Os resultados de precisão obtidos sobre o modelo são positivos para os objetivos traçados e para o sistema em causa. Apesar disso, existe sempre a possibilidade de se aperfeiçoar os níveis de precisão e garantir um modelo ainda mais fiável. A larga escala, estes pequenos detalhes têm sempre um impacto maior e podem induzir a estudos menos precisos. Existem vários caminhos de trabalho possíveis para que se consiga aperfeiçoar o modelo e obterem-se melhores resultados de precisão. Por exemplo, através da introdução de novos *datasets* ou através do estudo de novos casos de operação do disco como a variação sequencial entre escritas e leituras ou operações de escrita alternadas em duas posições de memória. Além disso, pode-se estudar a introdução de novas variáveis relevantes para este tipo de estimativa como o indicador *await* da ferramenta IOSTAT que apresenta o tempo ocupado a resolver um pedido de IO.

A fim de se obter uma maior flexibilidade a larga escala, ambiciona-se realizar, num trabalho futuro, a automatização da etapa de calibração. Este processo de automatização, permite que haja uma maior facilidade na preparação do modelo para diferentes sistemas com diversas configurações. O modelo tornar-se-ia mais ágil para qualquer ambiente, o que permitiria uma instalação rápida do mesmo nessas condições mais adversas. Este caso iria obrigar a um estudo mais alargado do impacto energético de mais equipamentos de *hardware*, de outros sistemas operativos que não sejam Linux e, claro, da relação existente entre os mesmos.

Uma outra matéria de interesse para trabalho futuro revela-se com o alargamento do numero de componentes possíveis de se estimar no modelo, tanto a

nível de *hardware* como *software*. A nível de hardware, existe o interesse de alargar a cobertura do modelo a outros componentes que também têm alguma relevância no impacto energético do sistema, como as placas rede Ethernet e antenas de conectividade WiFi.

A nível de *software*, da mesma maneira que se proporcionou o consumo energético por processo para as atividades em disco, pretende-se atingir o mesmo objetivo para o processador e para a memória primária. As soluções a que se recorre para se atingirem estes objetivos poderão não ser as mesmas. Uma das soluções possíveis pode passar pela recolha de indicadores fornecidos pelo sistema operativo sobre o nível de utilização de recursos destes subsistemas por aplicação. Além destes indicadores, a nível de *hardware*, através dos contadores PCM, estes subsistemas disponibilizam informações que podem ser úteis para um trabalho futuro.

Numa vertente mais comercial, e para que o utilizador não necessite de grandes conhecimentos técnicos da área informática, procura-se futuramente que o modelo seja implementado numa interface gráfica simplificada e flexível. Existe também o interesse de desenvolver uma API que permita que este modelo seja facilmente inserido em aplicações externas ao projeto. Desta forma, existe uma maior facilidade para que terceiros implementem outros tópicos interessantes através do modelo como economização de energia, controlo de faturas energéticas e programação para delimitar consumos energéticos.

7. Bibliografia

[1] M. P. Mills, "The Cloud Begins with Coal," Digital Power Group, E.U.A., 2013.

[2] CEET, "The Power of Wireless Cloud," University of Melbourne, Australia, 2013.

[3] GSMA Intelligence, "The Mobile Economy 2015," GSM Association, Londres, Inglaterra, 2015.

[4] Ericsson, "Ericsson Mobility Report," Ericsson, Suécia, 2014.

[5] Cisco, "Cisco VNI Global Mobile Data Traffic Forecast," em *Transformation through Innovation, MWC World Congress 2015*, Barcelona, Espanha, 2015.

[6] S. Wojcicki, "YouTube CEO Talks," 2014.

[7] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters,* vol. III, pp. -, 23 Setembro 2008.

[8] J. G. Koomey, "Growth in Data Center Electricity Use 2005 to 2010," EUA, 2011.

[9] A. Andrae e P. M. Corcoran, "Emerging Trends in Electricity Consumption for Consumer ICT," *National University of Ireland Rep.,* nº Tech, 2013.

[10] G. Cook, "How Clean is Your Cloud?," Arc Communications, Holanda, 2012.

[11] Intel, "Reducing Data Center Energy Consumption," Intel Corporation, E.U.A., 2008.

[12] IDC, "Virtualization and Multicore Innovations Disrupt the Worldwide Server Market," *IDC Document #206035,* Março 2007.

[13] Emerson, "Energy Logic: Reducing Data Center Energy Consumption by Creating Savings that Cascade Across Systems," Emerson Network Power, E.U.A., 2009.

[14] CGI, "Green Data Centers," CGI GROUP INC., 2014.

[15] Google, "Google's Green PPAs: What, How, and Why," Google Inc., 2013.

[16] Apple, "Environmental Responsibility Report, 2015 Progress Report, Covering FY2014," Apple Inc., E.U.A., 2015.

[17] Anthesis, "Data Center Efficiency Assessment," www.suerossi.com, E.U.A., 2014.

[18] M. Wachs, L. Xu, A. Kanevsky e G. R. Ganger, "Exertion-based billing for cloud storage access," em *HotCloud'11 Proceedings of the 3rd USENIX: Hot topics in cloud computing*, E.U.A..

[19] D. Jenkins e Intel, "The Efficient Data Center," Intel, EUA, 2009.

[20] J. Pflueger, A. Esser e Dell, "The Energy Smart Data Center," Dell, 2008.

[21] Deloitte, "Smartphone batteries: better but no breakthrough," The Creative Studio at Deloitte, Londres, Inglaterra, 2015.

[22] Apple, "13-inch MacBook Air: Environmental Report," Apple Inc., E.U.A., 2013.

[23] A. Hylick, R. Sohan, A. Rice e B. Jones, "An Analysis of Hard Drive Energy Consumption," 2008.

[24] A. Kansal, F. Zhao, J. Liu, N. Kothari e A. A. Bhattacharya, "Virtual Machine Power Metering and Provisioning," *SoCC '10 Proceedings of the 1st ACM symposium,* nº Cloud Computing, pp. 39-50, 2010.

[25] Google, L. A. Barroso e U. Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, vol. 6, Morgan & Claypool Publishers, 2009.

[26] E. Grochowski e R. F. Hoyt, "Future Trends in Hard Disk Drives," *IEEE Transactions on Magnetics,* vol. 32, pp. 1850-1854, 1996.

[27] J. Yang, Q. C. M. C. U. Component & Product Reliability Eng. e F.-B. Sun, "A comprehensive review of hard-disk drive reliability," *Reliability and Maintainability Symposium, 1999. Proceedings. Annual,* pp. 403 - 409, 1999.

[28] E. Pinheiro, W.-D. Weber, L. A. Barroso e Google, "Failure Trends in a Large Disk Drive Population," *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07),* Fevereiro 2007.

[29] D. Anderson, J. Dykes e E. Riedel, "More than an interface - scsi vs ata," em *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST'03)*, 2003.

[30] AMD: Advanced Micro Devices, "BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors," 2013.

[31] Intel Corporation, Intel® 64 and IA-32 Architectures Software Developer's Manual: System Programming Guide (Vol. 3A, 3B & 3C), vol. 3, Intel, 2015.

[32] M. Dayarathna, Y. Wen e R. Fan, "Data Center Energy Consumption Modeling : A Survey," *IEEE Communications Surveys & Tutorials,* Setembro 2015.

[33] Organisation Intergouvernementale de la Convention du Mètre, "The International System of Units (SI)," 2008.

[34] R. Lent, "A model for network server performance and power consumption," *Sustainable Computing: Informatics and Systems,* vol. 3, 2012.

[35] Oxford, The New Oxford Dictionary of English, Oxford, New York: Oxford University Press, 1998.

[36] L. Barroso, M. V. Google e U. Holzle, "The Case for Energy-Proportional Computing," *IEEE Computer Society,* vol. 40, pp. 33 - 37, 2007.

[37] W. F. N. K. James M. Kaplan, "Revolutionizing Data Center Energy Efficiency," McKinsey&Company, 2008.

[38] Gartner, "Gartner Says Efficient Data Center Design Can Lead to 300 Percent Capacity Growth in 60 Percent Less Space," 18 Novembro 2010. [Online]. Available: http://www.gartner.com/newsroom/id/1472714. [Acedido em Novembro 2015].

[39] A. Vasan, A. Sivasubramaniam, V. Shimpi, T. Sivabalan e R. Subbiah, "Worth their Watts? - An Empirical Study of Datacenter Servers," em *Proceedings of the 16th International Symposium on High Performance Computer Architecture*, India, 2010.

[40] H. Liu e S. J. C. U. Accenture Technol. Labs., "A Measurement Study of Server Utilization in Public Clouds," em *Proceedings of the 9th International Conference on Dependable, Autonomic and Secure Computing*, Sydney, NSW, 2011.

[41] L. Wanner, L. A. C. U. Univ. of California, R. Balani, S. Zahedi e C. Apte, "Variability-aware Duty Cycle Scheduling in Long Running Embedded Sensing Systems," em *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, França, 2011.

[42] Watts Up Series, "Watts Up Meters - About Us," [Online]. Available: https://www.wattsupmeters.com/secure/about.php. [Acedido em Fevereiro 2016].

[43] J. Jenne, V. Nijhawan e R. Hormuth, "Dell Energy Smart Architecture (DESA) for 11G Rack and Tower Servers," 2009.

[44] D. Hackenberg, T. Ilsche, R. Schone, D. Molka, M. Schmidt e W. E. Nagel, "Power Measurement Techniques on Standard Compute Nodes: A Quantitative Comparison," *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on,* pp. 194-204, 21-23 Abril 2013.

[45] Apple, "Use Activity Monitor on your Mac," [Online]. Available: https://support.apple.com/en-us/HT201464. [Acedido em Novembro 2015].

[46] Intel Open Source Technology Center, "PowerTOP," Novembro 2015. [Online]. Available: https://01.org/powertop/.

[47] Hewlett-Packard Corporation;Intel Corporation;Microsoft Corporation;Phoenix Technologies Ltd.;Toshiba Corporation, Advanced Configuration and Power Interface Specification, Revision 5.0, Errata A ed., 2013.

[48] W. Stallings, Computer Organization And Architecture - 8th Edition, Prentice Hall, 2009.

[49] B. Kjell e CCSU, "Introduction to Computer Science using Java," 2014. [Online]. Available: https://chortle.ccsu.edu/java5/index.html. [Acedido em Outubro 2015].

[50] A. Snavely, "Instruction Set Architecture".

[51] E. Blern, l. Menon e K. Sankaralingarn, "Power Struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and x86 Architectures," IEEE, 2013.

[52] P. R. Lakhe, "A Technology In Most Recent Processor Is Complex Reduced Instruction Set Computers (CRISC): A Survey," International Journal of Innovative Research & Studies, India, 2013.

[53] H. Krad e A. Y. Al-Taie, "A New Trend for CISC and RISC Architectures," 2007.

[54] D. Stevenson e PCAdvisor, "AMD vs Intel: Why AMD will always be the underdog but ARM still beats Intel for mobile," Setembro 2015. [Online].

Available: http://www.pcadvisor.co.uk/feature/pc-components/amd-vs-intel-3528212/. [Acedido em Dezembro 2015].

[55] E. Roberts e S. University, "RISC vs CISC," [Online]. Available: http://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/riscisc/. [Acedido em Novembro 2015].

[56] D. J. Greaves, "System on Chip Design and Modelling," 2011.

[57] TREFIS, "Why Qualcomm Remains The No.1 Player In Cellular Baseband," Julho 2014. [Online]. Available: http://www.trefis.com/stock/qcom/articles/246477/why-qualcomm-remains-the-no-1-player-in-cellular-baseband/2014-07-10. [Acedido em Novembro 2015].

[58] fool.com, "3 Promising Markets for Qualcomm Inc.," Outubro 2015. [Online]. Available: http://www.fool.com/investing/general/2015/10/28/3-promising-markets-for-qualcomm-inc.aspx. [Acedido em Novembro 2015].

[59] TREFIS, "Intel (INTC)," Outubro 2015. [Online]. Available: http://www.trefis.com/company?hm=INTC.trefis&. [Acedido em Novembro 2015].

[60] T. Culpan, I. King, B. Womack e Bloomberg, "Intel Seen Threatened as Google Mulls Own Server Chips," Dezembro 2013. [Online]. Available: http://www.bloomberg.com/news/articles/2013-12-13/intel-seen-threatened-as-google-mulls-own-server-chips. [Acedido em Novembro 2015].

[61] J. Lin, H. Zheng, Z. Zhu, H. D. E. Gorbatov e Z. Zhang, "Software thermal management of dram memory for multicore systems," em *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, E.U.A., 2008.

[62] Intel Corporation, "The Problem of Power Consumption in Servers," 2009.

[63] JEDEC, "Standards & Documents Search: JC-10: Terms, Definitions, and Symbols," [Online]. Available: http://www.jedec.org/standards-documents/results/taxonomy%3A1?order=field_doc_full_number_value&sort=asc. [Acedido em Novembro 2015].

[64] MICRON, "DDR4 – Advantages of Migrating from DDR3," [Online]. Available: https://www.micron.com/products/dram/ddr3-to-ddr4. [Acedido em Dezembro 2015].

[65] DELL, "What is DDR3L memory?," Outubro 2015. [Online]. Available: http://www.dell.com/support/article/us/en/19/SLN153768/EN. [Acedido em Dezembro 2015].

[66] Crucial, "Crucial energy-efficient DDR3 server memory," [Online]. Available: http://www.crucial.com/usa/en/memory-server-energy-efficient. [Acedido em Novembro 2015].

[67] Kingston, "What's New in Greener DDR3 Server Memory," [Online]. Available: http://www.kingston.com/us/community/articledetail?ArticleId=14&Article-Title=Whats-New-in-Greener-DDR3-Server-Memory. [Acedido em Novembro 2015].

[68] DATARAM, "4GB and 8GB PC3L-10600 RDIMMs Validated on Intel Xeon 5600 Series Processors," [Online]. Available: http://memory.dataram.com/products-and-services/energy-efficient-ddr3l-memory-for-latest-ibm-system-x-servers. [Acedido em Novembro 2015].

[69] S. Patel e AMD, "Server Memory Power Trends," em *Server Memory Forum Shenzhen 2012*, China, 2012.

[70] S. Lee, S. h. Inc. e JEDEC, "JEDEC: Introduction to LPDDR3," *LPDDR3 Symposium 2012,* 2012.

[71] A. B. Kahng e V. Srinivas, "Mobile System Considerations for SDRAM Interface Trends," em *System Level Interconnect Prediction (SLIP), 2011 13th International Workshop on*, E.U.A., 2011.

[72] Samsung USA, "DDR3 vs. LPDDR2 Power Comparison," Dezembro 2011. [Online]. Available: https://www.youtube.com/watch?v=_-Hx81ZE-D8. [Acedido em Novembro 2015].

[73] MICRON, "LPDRAM," [Online]. Available: https://www.micron.com/products/dram/lpdram. [Acedido em Dezembro 2015].

[74] K. Kant e Intel, "Data center evolution A tutorial on state of the art, issues, and challenges," *Elsevier,* vol. 53, nº Data Centers, p. 2939–2965, 2009.

[75] D. Andersen e S. Swanson, "Rethinking Flash in the Data Center," *IEEE Micro,* vol. 30, pp. 52 - 54, 2010.

[76] Seagate, "The Top 20 Things to Know About SSD," 2011.

[77] DIE, "hdparm - get/set SATA/IDE device parameters," [Online]. Available: http://linux.die.net/man/8/hdparm.

[78] Samsung, "Samsung SSD 850 EVO," 2014.

[79] G.-P. Musumeci e M. K. Loukides, System Performance Tuning, Segunda Edição ed., M. Loukides, Ed., O'Reilly Media, 2002.

[80] F. Wu, H. Xi, J. Li e N. Zou, "Linux readahead: less tricks for more," *Proceedings of the Linux Symposium,* vol. II, 2007.

[81] M. Sri-Jayantha, "Trends in mobile storage design," *IEEE Symposium on,* nº Low Power Electronics, pp. 54-57, Outubro 1995.

[82] Toshiba, "4K Sector Disk Drives: Transitioning to the Future with Advanced Format Technologies," 2011.

[83] Western Digital, "Advanced Format Technology," 2010.

[84] S. Gurumurthi e A. Sivasubramaniam, "Energy-Efficient Storage Systems for Data Centers," em *Energy-Efficient Distributed Computing Systems*, John Wiley & Sons, Inc., 2012, pp. 361-376.

[85] I. Sato, T. J. NTT Appl. Electron. Lab., K. Otani, M. Mizukami e S. Oguchi, "Characteristics of heat transfer in small disk enclosures at high rotation speeds," em *InterSociety Conference on Thermal Phenomena in Electronic Systems, 1990. I-THERM II.*, 1990.

[86] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse e R. Panigrahy, "Design Tradeoffs for SSD Performance," *Proceedings of the USENIX Technical Conference,* Junho 2008.

[87] A. Tal, "Two Flash Technologies Compared: NOR vs NAND," 2002.

[88] Intel, "Partition Alignment of Intel® SSDs for Achieving Maximum Performance and Endurance," Intel Corporation, 2014.

[89] PCMag e J. S. Domingo, "SSD vs. HDD: What's the Difference?," Fevereiro 2015. [Online]. Available: http://www.pcmag.com/article2/0,2817,2404258,00.asp. [Acedido em Novembro 2015].

[90] A. L. Shimpi e AnandTech, "Seagate 2nd Generation Momentus XT (750GB) Hybrid HDD Review," Dezembro 2011. [Online]. Available: http://www.anandtech.com/show/5160/seagate-2nd-generation-

momentus-xt-750gb-hybrid-hdd-review. [Acedido em Dezembro 2015].

[91] P. Schmid e A. Roos, "Momentus XT 750 GB Review: A Second-Gen Hybrid Hard Drive," Fevereiro 2012. [Online]. Available: http://www.tomshardware.com/reviews/hybrid-hard-drive-flash-ssd,3116.html. [Acedido em Dezembro 2015].

[92] Amazon Europe, "Los más vendidos en Discos duros sólidos internos," [Online]. Available: http://www.amazon.es/gp/bestsellers/electronics/937918031/. [Acedido em Novembro 2015].

[93] Amazon Europe, "Los más vendidos en Discos duros internos," [Online]. Available: http://www.amazon.es/gp/bestsellers/electronics/937917031/. [Acedido em Novembro 2015].

[94] M. Hahnel, B. Dobel, M. Volp e H. Hartig, "Measuring Energy Consumption for Short Code Paths Using RAPL," em *GREENMETRICS' 12*, Londres, Reino Unido, 2012.

[95] V. Weaver, U. o. M. O. M. U. Electr. & Comput. Eng., D. Terpstra, H. McCraw e M. Johnson, "PAPI 5: Measuring power, energy, and the cloud," em *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013.

[96] Intel Corporation;Microsoft Corporation, Advanced Power Management (APM) BIOS Interface Specification, vol. 1.2, Intel APM Support Desk, 1996.

[97] FreeBSD, H. Pandya e T. Rhodes, "Chapter 11. Configuration and Tuning: Power and Resource Management," [Online]. Available: https://www.freebsd.org/doc/handbook/acpi-overview.html. [Acedido em Dezembro 2015].

[98] A. L. Brown e I. Corporation, "The State of ACPI in the Linux Kernel," *Linux Symposium 2004,* vol. I, pp. 121-132, 2004.

[99] L. Brown, A. Keshavamurthy, D. S. Li, R. Moore, V. Pallipadi, L. Yu e I. O. S. T. Center, "ACPI in Linux," *Proceedings of the Linux Symposium,* vol. I, pp. 51-68, 2005.

[100] P. Barry e P. Crowley, Modern Embedded Computing: Designing Connected, Pervasive, Media-rich Systems, Elsevier, 2012, p. 518.

[101] L. Brown, Intel e I. O. S. T. Center, "Linux Idle Power Checkup," em *Linuxcon*, Boston, 2010.

[102] L. Brown e Intel, "Linux/drivers/idle/intel_idle.c," 2013. [Online]. Available: http://lxr.free-electrons.com/source/drivers/idle/intel_idle.c. [Acedido em Dezembro 2015].

[103] K. C. Accardi e A. Yates, "POWERTOP User's Guide," Intel Corporation, 2014.

[104] L. Brown e Intel, "Linux Idle Power Checkup," em *Linuxcon 2010*, Boston, MA, 2010.

[105] Intel, "Linux/Documentation/cpu-freq/intel-pstate.txt," [Online]. Available: http://lxr.free-electrons.com/source/Documentation/cpu-freq/intel-pstate.txt. [Acedido em Novembro 2015].

[106] Intel, "Power Management States: P-States, C-States, and Package C-States," Abril 2014. [Online]. Available: https://software.intel.com/en-us/articles/power-management-states-p-states-c-states-and-package-c-states. [Acedido em Novembro 2015].

[107] L. Yu e I. Corporation, "Documentation: The file /sys/power/state || /proc/acpi/sleep," 2006. [Online]. Available: http://acpi.sourceforge.net/index.html. [Acedido em 2015].

[108] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra e S. Moore, "Measuring Energy and Power with PAPI," em *Parallel Processing Workshops (ICPPW), 41st International Conference on*, Pittsburgh, PA, EUA, 2012.

[109] T. Rauber, G. Runger, M. Schwind, H. Xu e S. Melzner, "Energy Models for DVFS Processors," em *9th Scheduling for Large Scale Systems Workshop*, Lyon, França, 2014.

[110] H. McCraw, J. Ralph, A. Danalis e J. Dongarra, "Power Monitoring with PAPI for Extreme Scale Architectures and Dataflow-based Programming Models," 2014.

[111] A. Cabrera, F. Almeida, J. Arteaga e V. Blanco, "Measuring Energy Consumption with the Energy Measurement Library," Espanha, 2014.

[112] Intel, E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan e E. Weissmann, "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," em *Micro, IEEE*, 2012.

[113] Intel, "Intel® Power Governor," 20 Julho 2012. [Online]. Available: https://software.intel.com/en-us/articles/intel-power-governor. [Acedido em 2015].

[114] L.-I. H. Simonsen, "Increasing SpMV Energy Efficiency," Noruega, 2013.

[115] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna e C. Le, "RAPL: Memory Power Estimation and Capping," EUA, 2010.

[116] J. Demmel e A. Gearhart, "Instrumenting Linear Algebra Energy Consumption via On-chip Energy Counters," Berkley, EUA, 2012.

[117] Intel, Patrick Fay, "Intel Developer Zone Topic: RAPL analysis/tests on a laptop," Março 2015. [Online]. Available: https://software.intel.com/pt-br/forums/software-tuning-performance-optimization-platform-monitoring/topic/543879. [Acedido em Março-Dezembro 2015].

[118] Intel, Intel Xeon Processor E5-1600, E5-2400, and E5-2600 v3 Product Families: Datasheet, document 330784, revision 002 ed., vol. 2, Intel Corporation, 2015.

[119] B. Subramaniam e W.-c. Feng, "Towards Energy-Proportional Computing Using Subsystem-Level Power Management," EUA, 2015.

[120] B. Wang, W. C. Wuhan Res. Inst. of Posts & Telecommun., B. Wang e Q. Xiong, "The Comparison of Communication Methods between User and Kernel Space in Embedded Linux," em *Internation Conference on Computational Problem-Solving (ICCP)*, Lijiang, 2010.

[121] T. Bowden, B. Bauer, J. Nerin, S. Feng e S. Seibold, "THE /proc FILESYSTEM," Junho 2009. [Online]. Available: https://www.kernel.org/doc/Documentation/filesystems/proc.txt. [Acedido em Outubro 2015].

[122] SYSSTAT, "SYSSTAT Documentation," [Online]. Available: http://sebastien.godard.pagesperso-orange.fr. [Acedido em Outubro 2015].

[123] DIE, "top - display Linux tasks," [Online]. Available: http://linux.die.net/man/1/top. [Acedido em Outubro 2015].

[124] DIE, "atop - AT Computing's System & Process Monitor," [Online]. Available: http://linux.die.net/man/1/atop. [Acedido em Outubro 2015].

[125] DIE, "vmstat - Report virtual memory statistics," [Online]. Available: http://linux.die.net/man/8/vmstat. [Acedido em Outubro 2015].

[126] V. Weaver, C. U. I. N. Comput. Syst. Lab. e S. McKee, " Can hardware performance counters be trusted?," *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on,* pp. 141-150, 2008.

[127] K. Singh, M. Bhadauria e S. A. McKee, "Real Time Power Estimation and Thread Scheduling via Performance Counters," *ACM SIGARCH Computer Architecture News,* vol. 37, nº 2, pp. 46-55, 2009.

[128] S. Moore, "PAPI - Performance API," Virtual Institute - High Productivity Supercomputing, 2011.

[129] C. Gu, H. I. o. T. S. G. S. S. C. Department of Computer Science and Technology, H. Huang e X. Jia, "Power Metering for Virtual Machine in Cloud Computing—Challenges and Opportunities," *Access, IEEE,* vol. 2, pp. 1106-1116, 2014.

[130] F. I. Gordon F. Hughes, J. F. Murray, I. Kenneth Kreutz-Delgado Senior Member e C. Elkan, "Improved Disk-Drive Failure Warning," *IEEE Transactions on Reliability,* vol. 51, pp. 350-357, 2002.

[131] B. Zhu, N. U. T. C. Nankai-Baidu Joint Lab., G. Wang, X. Liu e D. Hu, "Proactive Drive Failure Prediction for Large Scale Storage Systems," em *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, E.U.A., 2013.

[132] DIE, "smartctl - Control and Monitor Utility for SMART Disks," [Online]. Available: http://linux.die.net/man/8/smartctl.

[133] B. Allen, "Monitoring Hard Disks with SMART," Janeiro 2004. [Online]. Available: http://www.linuxjournal.com/magazine/monitoring-hard-disks-smart. [Acedido em Outubro 2015].

[134] N. Matthew e R. Stones, Beginning Linux Programming - 4th Edition, Wiley Publishing, Inc., 2007.

[135] T. Schürmann, "Tune Your Hard Disk with hdparm," [Online]. Available: http://www.linux-magazine.com/Online/Features/Tune-Your-Hard-Disk-with-hdparm. [Acedido em Dezembro 2015].

[136] archlinux, "hdparm," [Online]. Available: https://wiki.archlinux.org/index.php/Hdparm. [Acedido em Dezembro 2015].

[137] P. Mochel, "The sysfs Filesystem".

[138] B. Krishnan, H. Amur, A. Gavrilovska e K. Schwan, "VM Power Metering: Feasibility and Challenges," Atlanta, 2010.

[139] A. E. H. Bohra e V. Chaudhary, "VMeter: Power Modelling for Virtualized Clouds," IEEE, Nova Iorque, 2010.

[140] J. Stoess, C. Lang e F. Bellosa, "Energy Management for Hypervisor-Based Virtual Machines," Alemanha, 2010.

[141] G. Dhiman, K. Mihic e T. Rosing, "A System for Online Power Prediction in Virtualized Environments Using Gaussian Mixture Models," San Diego, E.U.A., 2010.

[142] D. Bricker e Intel, "What does "Instruction Retired" mean exactly?," 2003. [Online]. Available: https://software.intel.com/en-us/forums/intel-vtune-amplifier-xe/topic/311170.

[143] D. Tripp, "Action research: a methodological introduction," University of Murdoch, Australia, 2005.

[144] S. J. Vaughan-Nichols, "Linux still rules supercomputing," Julho 2015. [Online]. Available: http://www.zdnet.com/article/linux-still-rules-supercomputing/. [Acedido em Novembro 2015].

[145] D. Perlmutter e Intel, "Haswell Presentation," em *Intel Developer Forum 2012*, São Francisco, USA, 2012.

[146] Intel, "Intel® Core™ i5-4200M Processor (3M Cache, up to 3.10 GHz)," [Online]. Available: http://ark.intel.com/products/76348/Intel-Core-i5-4200M-Processor-3M-Cache-up-to-3_10-GHz. [Acedido em Dezembro 2015].

[147] MICRON, "SODIMM," [Online]. Available: https://www.micron.com/products/dram-modules/sodimm. [Acedido em Dezembro 2015].

[148] Lenovo, "ThinkPad L440 Laptop," [Online]. Available: http://shop.lenovo.com/us/en/laptops/thinkpad/l-series/l440/. [Acedido em Outubro 2015].

[149] Samsung, "Technical Specification," [Online]. Available: http://www.samsung.com/global/business/semiconductor/minisite/SSD/global/html/ssd850evo/specifications.html. [Acedido em Dezembro 2015].

[150] Seagate, "Momentus® Thin SATA - 7200 RPM - ST500LM021 & ST320LM010," 2014.

[151] R. Love, Linux System Programming, O'REILLY, 2007.

[152] A. Gillen e IDC, "The Role of Linux in Datacenter Modernization," 2013.

[153] C. DiBona, Interviewee, *Google: "Android is the Linux desktop dream come true".* [Entrevista]. Maio 2011.

[154] A. McPherson, "What a Year for Linux: Please Join us in Celebration," Dezembro 2012. [Online]. Available: http://www.linuxfoundation.org/news-media/blogs/browse/2012/12/what-year-linux-please-join-us-celebration. [Acedido em Novembro 2015].

[155] F. Manjoo, "A Murky Road Ahead for Android, Despite Market Dominance," Maio 2015. [Online]. Available: Farhad Manjoo. [Acedido em Novembro 2015].

[156] S. J. Vaughan-Nichols, "For Linux, Supercomputers R Us," Agosto 2015. [Online]. Available: http://www.computerworld.com/article/2960701/linux/for-linux-supercomputers-r-us.html. [Acedido em Novembro 2015].

[157] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon e M. Meuer, "TOP500 - Statistics on high-performance computers," Setembro 2015. [Online]. Available: http://www.top500.org/statistics/list/. [Acedido em Novembro 2015].

[158] S. Sharma e techradar, "10 of the most popular lightweight Linux distros," Março 2015. [Online]. Available: http://www.techradar.com/news/software/operating-systems/10-of-the-most-popular-lightweight-linux-distros-1295034. [Acedido em Dezembro 2015].

[159] A. Sen, "Ubuntu 14.10 vs Kubuntu 14.10 vs Xubuntu 14.10 vs Lubuntu 14.10 vs Ubuntu GNOME 14.10: A Comparison," Novembro 2014. [Online]. Available: http://mylinuxexplore.blogspot.pt/2014/11/ubuntu-1410-vs-kubuntu-1410-vs-xubuntu.html. [Acedido em Novembro 2015].

[160] man7, "syscall - indirect system call," Março 2015. [Online]. Available: http://man7.org/linux/man-pages/man2/syscall.2.html. [Acedido em Dezembro 2015].

[161] J. McCalpin, "PAPI Mailing List," 2015. [Online]. Available: http://ptools-perfapi.eecs.utk.narkive.com/kULDsRih/llc-events-question. [Acedido em 2015].

[162] C. Mobius, T. U. o. D. D. G. Dept. of Comput. Networks, W. Dargie e A. Schill, "Power Consumption Estimation Models for Processors, Virtual Machines, and Servers," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, pp. 1600-1614, 28 Julho 2013.

[163] D. S. Myers e A. L. Bazinet, "Intercepting Arbitrary Functions on Windows, UNIX, and Macintosh OS X Platforms," University of Maryland, E.U.A., 2004.

[164] J. Dongarra, U. o. T. K. T. U. Innovative Comput. Lab., H. Ltaief, P. Luszczek e V. Weaver, "Energy Footprint of Advanced Dense Numerical Linear Algebra Using Tile Algorithms on Multicore Architectures," em *Second International Conference on Cloud and Green Computing (CGC)*, Xiangtan, China, 2012.

[165] MICRON, "Calculating Total DRAM Power: Calculating Memory System Power for DDR3," 2007.

[166] Alasir, "RAMspeed, a cache and memory benchmarking tool," [Online]. Available: http://alasir.com/software/ramspeed/. [Acedido em Janeiro 2016].

[167] freedesktop.org, "The Light Display Manager (LightDM)," [Online]. Available: http://www.freedesktop.org/wiki/Software/LightDM/. [Acedido em Dezembro 2015].

[168] Ubuntu, "LightDM," Novembro 2014. [Online]. Available: https://wiki.ubuntu.com/LightDM. [Acedido em Dezembro 2015].

[169] Logitech, "Logitech Support," Dezembro 2015. [Online]. Available: http://support.logitech.com/en_us/home.

[170] SAP, "SAP Sybase Adaptive Server Enterprise: Raw Devices vs File System," 2013.

[171] M. T. Jones, "Anatomy of the Linux virtual file system switch: Abstractions and high-level concepts," 2009.

[172] D. J. Shakshober, "Choosing an I/O Scheduler for Red Hat® Enterprise Linux® 4 and the 2.6 Kernel," [Online]. Available: http://www.redhat.com/magazine/008jun05/features/schedulers/. [Acedido em Janeiro 2016].

[173] R. Love, Linux Kernel Development, Pearson Education, 2010, p. 480.

[174] J. B. Layton, "2.6.33 is Out! Say Good Bye to the Anticipatory Scheduler," Março 2010. [Online]. Available: http://www.linux-mag.com/id/7724/. [Acedido em Janeiro 2016].

[175] J. Axboe, "[PATCH] Make CFQ the default IO scheduler," Junho 2006. [Online]. Available: http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=b17fd9bceb99610f6dc7998c9a4ed6b71520be2b. [Acedido em Janeiro 2016].

[176] Linux Kernel Newbies, "Linux 2 6 33," 2010. [Online]. Available: http://kernelnewbies.org/Linux_2_6_33. [Acedido em Janeiro 2016].

[177] M. Larabel, "Linux 3.16: Deadline I/O Scheduler Generally Leads With A SSD," Junho 2014. [Online]. Available: http://www.phoronix.com/scan.php?page=article&item=linux_316_iosched&num=1. [Acedido em Janeiro 2016].

[178] Ubuntu, "fsync, fdatasync," [Online]. Available: http://manpages.ubuntu.com/manpages/raring/man2/fsync.2.html. [Acedido em Dezembro 2015].

[179] Y. Xia, "Does Fsync() Ensure Data Persistency When Disk Cache Is Enabled?," 2014. [Online]. Available: http://xiayubin.com/blog/2014/06/20/does-fsync-ensure-data-persistency-when-disk-cache-is-enabled/. [Acedido em Dezembro 2015].

[180] DIE, "posix_fadvise - predeclare an access pattern for file data," [Online]. Available: http://linux.die.net/man/2/posix_fadvise. [Acedido em Janeiro 2016].

[181] DIE, "fdisk - Partition table manipulator for Linux," [Online]. Available: http://linux.die.net/man/8/fdisk. [Acedido em Janeiro 2016].

[182] HGST, "Advanced Format Technology Brief," HGST, 2014.

[183] Microsoft, "Using Raw Partitions," [Online]. Available: https://technet.microsoft.com/en-us/library/aa933078(v=sql.80).aspx. [Acedido em Março 2016].

[184] CurveExpert, "CurveExpert and GraphExpert Software," [Online]. Available: https://www.curveexpert.net/.

[185] T. B. Russell Coker, "Bonnie++ Homepage," [Online]. Available: http://www.coker.com.au/bonnie++/.

[186] D. t. Woligroski, "Power Supply 101: A Reference Of Specifications," Dezembro 2011. [Online]. Available: http://www.tomshardware.com/reviews/power-supply-specifications-atx-reference,3061-11.html. [Acedido em Março 2016].

[187] SparkFun, "AttoPilot Voltage and Current Sense Breakout - 90A," [Online]. Available: https://www.sparkfun.com/products/9028. [Acedido em Março 2016].

[188] SparkFun, "SparkFun Low Current Sensor Breakout - ACS712," [Online]. Available: https://www.sparkfun.com/products/8883. [Acedido em Abril 2016].

[189] SparkFun, "Resistors," [Online]. Available: https://learn.sparkfun.com/tutorials/resistors/power-rating. [Acedido em Março 2016].

[190] VITROHM, "Series KH - Power Wirewound Ceramic Resistors," ARTLAND, 2000.