Universidade do Minho
Escola de Engenharia

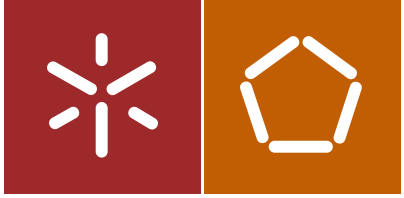
Sérgio Branco Amorim

Utilização de Computação de Elevado Desempenho para a simulação de problemas de hidrodinâmica costeira

outubro de 2015

Sérgio Branco Amorim Utilização de Computação de Elevado Desempenho para a simulação de problemas de hidrodinâmica costeira

UMinho | 2015



Universidade do Minho
Escola de Engenharia

Sérgio Branco Amorim

Utilização de Computação de Elevado Desempenho para a simulação de problemas de hidrodinâmica costeira

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Civil

Trabalho efectuado sob a orientação do
Professor Doutor José Luís Pinho

outubro de 2015

AGRADECIMENTOS

Agradeço ao meu orientador, o Professor Doutor José Luís da Silva Pinho, pelo apoio facultado ao longo deste trabalho, principalmente pela sua disponibilidade em ajudar e esclarecer ao longo de todo este processo.

Aos meus pais. Sem o seu apoio incondicional não teria conseguido acabar esta etapa da minha vida.

E finalmente, à minha namorada Rosa, que esteve sempre ao meu lado. A motivação e ânimo que ela me deu foram essenciais na realização desta dissertação.

RESUMO

Nesta dissertação pretende avaliar-se o desempenho da computação paralela na modelação hidráulica realizada com o programa Delft3D. A computação paralela permite o aumento da capacidade computacional com menores custos do que o simples recurso a unidades de *hardware* com maior potência (como processadores, memórias ou outros) o que a leva a ser, geralmente, uma solução atrativa. Nesta dissertação testou-se o uso deste recurso em duas vertentes possíveis. A primeira tratou-se da utilização de um *cluster* informático, um computador de alto desempenho formado por vários computadores individuais que podem trabalhar individualmente ou em conjunto, sendo essa uma forma usual de paralelismo informático em que várias unidades trabalham em conjunto num mesmo problema. Como o paralelismo chegou à arquitetura dos computadores pessoais mais comuns, sendo o processador de qualquer um deles constituídos por vários *cores* que podem funcionar como unidades de processamento individualizados, o uso desta ferramenta foi também testado neste tipo de equipamento para se aferir das possíveis melhorias de desempenho destas ferramentas num caso de modelação hidráulica. Tratou-se especificamente de um canal retangular hipotético, tendo estes testes numéricos envolvido as equações de conservação de massa, momento e modelo de turbulência, contemplados pelo módulo hidrodinâmico do Delft3D. Assim, pretendeu-se com esta dissertação testar estas ferramentas e registar as diferenças de desempenho das diferentes soluções possíveis.

Os resultados obtidos permitiram constatar que o recurso à computação paralela trouxe benefícios claros no desempenho do programa Delft3D. No exemplo simulado conseguiu-se uma execução treze vezes mais rápida no *cluster* SeARCH e três vezes mais rápida no computador pessoal com *core* quádruplo.

Palavras-chave: *cluster*, computação paralela, modelação, delft3d, hidráulica

ABSTRACT

The aim of this dissertation was to evaluate the performance of parallel computing in hydraulic modelling managed with the Delft3D software. Parallel computing allows the increasing the computing capability with fewer costs than the simple use of hardware units with bigger power (such as processors, memories and others) which makes it usually an attractive solution. In this dissertation we tested the use of this resource in two different ways. The first was the use of a computing cluster, a high performance computer made of several single computing units that can function together or individually, that being a usual method of computing parallelism in which several computing units solve a single problem together. Since parallelism is now available in the architecture of the most common personal computers, with the processors of any of them now constituted by several cores that can function as individual processing unit, the use of these tools was also tested in this type of hardware to study the possible performance improvements due to the use of these tools in a hydraulic modelling example. Specifically a hypothetical rectangular channel, the numerical tests involving the continuity, momentum and turbulence model equations, included in the Delft3D hydrodynamic module. Therefore, the aim of this dissertation was to test these tools and to record the performance of the different possible solutions.

The obtained results led to the conclusion that using parallel computing brought obvious benefits in the modelling performance with the Delft3D. It was managed, in the simulated example, an execution that was thirteen times faster in the SeARCH cluster and three times faster in a personal computer with four *cores*.

Keywords: cluster, parallel computing, modelling, delft3d, hydraulics

ÌNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ACRÓNIMOS

ALU	-	Arithmetic Logic Unit
CPU	-	Central Processing Unit
FPU	-	Floating-point Unit
MIMD	-	Multiple Instruction Multiple Data
MISD	-	Multiple Instruction, Single Data
MPI	-	Message Passing Interface
PBS	-	Portable Batch System
SeARCH	-	Services and Advanced Research Computing with HTC/HPC clusters
SISD	-	Single Instruction, Single Data
SIMD	-	Single Instruction, Multiple Data

1. INTRODUÇÃO

1.1. Enquadramento

Os sistemas aquáticos têm um lugar essencial na vida das pessoas. Para além de serem fontes da água utilizada para consumo humano e dos recursos piscatórios, têm outros usos importantes como a utilização para fins recreativos, como vias de transporte e um papel ecológico importante como ecossistemas cruciais para a vida de uma multitude de seres vivos.

As duas principais razões para a utilização de modelos computacionais são o melhor entendimento de processos físicos, químicos e biológicos, e também a sua utilização como ferramenta para ajuda à tomada de decisões relativas à gestão de sistemas aquáticos, permitindo simular esses mesmos sistemas.

Existem dois tipos principais de modelos: os modelos físicos e os modelos matemáticos. Um modelo físico permite realizar uma simulação à escala utilizando água (ou outro fluído) na qual é produzido um fluxo à escala que é medido e relacionado com as propriedades do escoamento no problema real. Os modelos matemáticos representam situações, como escoamentos de água, através de equações matemáticas que, no caso da hidráulica, são baseados em princípios físicos, químicos e biológicos que variam temporal e espacialmente, sendo usualmente resolvidas utilizando métodos computacionais (Ji, 2008).

A modelação computacional é uma área cada vez mais importante em pesquisas científicas e na resolução de problemas de engenharia. A necessidade de obtenção de modelos mais precisos e velozes para a resolução de problemas cada vez mais complexos faz a evolução tender para o uso de maior potência de computação, como o aumento da memória e da capacidade de processamento. Nas últimas décadas, o desenvolvimento de computação de elevado desempenho resultou em progressos no *hardware* paralelo e no *software* relacionado que tenta aproveitar estes recursos.

O fraco desempenho de um computador pode restringir a precisão dos modelos, impossibilitando a obtenção de resultados em tempo útil sem o recurso a soluções de modelação mais simplificadas, prejudicando os resultados finais. A maior capacidade computacional permite modelos mais complexos e logo melhores resultados.

Existem duas formas de aumentar a capacidade computacional, sendo uma a utilização de elementos de *hardware* (como por exemplo, os processadores ou memórias) mais poderosos. Alternativamente, o uso de computação paralela, ou seja, de várias unidades computacionais mais pequenas de modo a resolver problemas que, uma única dessas unidades, não teria capacidade de resolver sozinha (Buyya, 1999).

A vantagem da computação paralela em relação à primeira solução é principalmente o menor custo económico, o que faz com que agora, seja uma opção muito utilizada. Também é relevante o facto do aumento da capacidade das componentes do *hardware* ser cada vez mais limitada, pois é difícil o aumento da velocidade de frequência de relógio do *hardware* (especificamente dos processadores) sem provocar sobreaquecimento dos componentes.

A produção de processadores evoluiu no passado recente de modo que, apesar da evolução da frequência do relógio de processador ter tido a tendência para estagnar pelas razões acima referidas, permitiu a criação de processadores que têm evoluído no seu desempenho ao longo do tempo através do aumento do número de transístores. Infelizmente, o aumento do número de transístores não tem representado um aumento tão significativo na melhoria do desempenho dos processadores como foi o aumento de velocidade de frequência quando essa opção era mais viável (entre 1986 e 2003 o aumento do desempenho dos processadores foi em média de 50%, tendo diminuído a partir daí para um valor de 22%) (Rauber, 2012).

O paralelismo pode acontecer a vários níveis da arquitetura do processador de um computador:

- Paralelismo ao nível dos *bits*: o tamanho das instruções tem evoluído de modo a poder-se aumentar a exatidão dos valores calculados e também aumentar o tamanho do espaço de endereçamento;

- Paralelismo através de *pipelining*: o que significa o intercalar das instruções. A execução das instruções é dividida em vários passos que são processados por várias unidades de *hardware* dedicadas, podendo assim as instruções serem executadas em paralelo excetuando quando há dependências entre elas.

- Paralelismo através de múltiplas unidades funcionais: os processadores modernos são compostos por várias unidades - como o ALU (*arithmetic-logic unit*), FPU (*floating point unit*) entre outras - que podem executar diferentes instruções em simultâneo.

- Paralelismo ao nível do processo ou do *thread:* o paralelismo nas opções antes referidas é bastante limitado, sendo hoje em dia a opção utilizada a de criar processadores com vários *cores*. *Cores* esses que funcionam como processadores diferentes, apesar de poderem partilhar a mesma memória (mesmo no caso das memórias cache).

Para classificar as diferentes arquiteturas computacionais a ferramenta comummente utilizada é a Taxonomia de Flynn (Tabela 1). Esta classificação é baseada na hipótese de execução do fluxo de instruções e dados, no facto de poderem ser executadas vários ao mesmo tempo ou apenas um de cada vez (Quinn, 2003).

Tabela 1 – Taxonomia de Flynn

	Fluxo de Dados	
Fluxo de Instruções	SISD	SIMD
	MISD	MIMD

O SISD (*single instruction single data*) corresponde à computação sequencial. Apenas uma instrução e um fluxo de dados são executados de cada vez. As restantes arquiteturas correspondem a diferentes possibilidades de arquiteturas paralelas. A mais predominante utilizada hoje em dia é a MIMD (*multiple instructions multiple data*), em que ao mesmo tempo vários fluxos de instruções em várias unidades de processamento operam em dados diferentes ao mesmo tempo.

Os outros tipos de arquiteturas paralelas são o SIMD (*single instruction multiple data*) e o MISD (*multiple instructions single data*). O SIMD é um tipo de arquitetura paralela desenhada para problemas específicos caracterizados por um alto padrão de regularidade nos dados (como o processamento de imagens). Todas as unidades de processamento executam a mesma instrução a cada momento mas podem operar sobre diferentes fluxos de dados. O MISD é por sua vez uma arquitetura própria para problemas caracterizados por um alto padrão de regularidade funcional (isto é, processamento de sinal). Nela, cada unidade de processamento executa instruções diferentes em cada momento. Os computadores baseados nela são constituídas por uma das unidades de processamento independentes que operam sobre um mesmo fluxo de dados enviando os resultados de uma unidade para a próxima.

A computação paralela, como vai ser utilizada nesta dissertação, pode-se caracterizar por haver várias unidades de processamento (*cores*) ou processadores tendo como função executar uma mesma tarefa de modo mais veloz. Baseia-se na ideia de resolver um problema dividindo-o em tarefas menores, que podem ser realizadas simultaneamente através de algum tipo de coordenação, diminuindo o tempo total de execução desse mesmo problema.

Um dos principais exemplos de computação paralela, o *cluster* informático, é um sistema de dois ou mais sistemas computacionais independentes e subsistemas de armazenamento com o propósito de partilhar e aceder a recursos. Neste caso, os nós podem ser computadores totalmente independentes, podendo chegar a ser, por exemplo, utilizados durante horas de "expediente" nas suas funções usuais e usados para participar em computação paralela durante as horas em que estariam, noutra situação, inativos. Essencialmente são computadores

totalmente funcionais isoladamente que podem participar em operações de grande dimensão dentro da rede à qual estão ligados.

O uso da computação paralela tornou-se lugar-comum, tanto na computação pessoal como nas atividades científicas e de engenharia relativas à modelação que pedem recursos informáticos mais potentes devido aos problemas complexos e de grandes dimensões que são estudados. Isso permite obter resultados mais rapidamente, o que beneficia a investigação, já que permite a resolução de problemas cada vez mais complexos, e logo com resultados teoricamente mais próximos da realidade, sem tornar os tempos de computação incomportáveis.

Assim, a computação paralela é tremendamente útil para problemas de grandes dimensões, o que leva ao seu recurso em vários casos relacionados com estudos científicos e resolução de problemas recorrendo à modelação computacional. São muitos os exemplos da sua utilização em diferentes áreas do conhecimento, tanto na Hidráulica como em muitas outras áreas de engenharia e da ciência. A seguir referem-se alguns exemplos disso.

Já há vários anos que a computação paralela é tema de pesquisa, procurando-se determinar a viabilidade do seu uso em situações práticas. Uma das áreas estudadas foi o processamento e análise de imagens médicas (Barbosa, 2000). Outros dos exemplos encontrados pertenciam à área da ótica, havendo estudos que, por exemplo, comprovam a utilidade de computação paralela para a obtenção de leituras mais rápidas de um interferómetro de luz branca (Kuk *et al.*, 2012). Leituras mais rápidas com computação paralela também foram comprovadas na utilização de raios-x para criar imagens detalhadas de objetos (Giersch *et al.*, 2003), havendo estudos similares mas em relação a ultra-sons (Li, 2001). A computação paralela também foi comprovada como podendo ser aplicada à modelação do comportamento de materiais (Pinho da Cruz, 2007). A gestão pode encontrar utilidade nesta tecnologia, como é comprovado em Pinto (2011) em que se testou a gestão de sistemas hidrotérmicos com *clusters* computacionais.

Especificamente, na área da Hidráulica os exemplos do uso de computação paralela são múltiplos. Aí é comum o uso de modelação de modo a obter resultados relativamente precisos

para problemas em que a zona de estudo é vasta e/ou as variáveis são múltiplas. Para casos desses a utilização de *clusters* para modelação pode ser extremamente útil. Incluídos aqui estão problemas de hidrologia, como simulações de estuários, situações de cheias ou propagação de ondas marítimas, de qualidade de água como a dispersão de poluentes em rios ou no mar.

Wang *et al.* (2011) desenvolvem um processo para a modelação usando computação paralela em casos de bacias hidrográficas. Existem também exemplos de modelação da hidrodinâmica com utilização destes métodos, como um trabalho em que são desenvolvidos modelos de hidrodinâmica e transporte para rios usando paralelismo (Zhang *et al*., 2010).

1.2. Objetivos

Neste trabalho foi processado o módulo hidrodinâmico do programa Delft3D usando um *cluster* computacional e um computador pessoal com vários processadores, de modo a obter conhecimentos sobre o desempenho de sistemas de computação paralela para modelos desse tipo.

O objetivo deste trabalho consistiu na utilização de um *cluster* computacional da Universidade do Minho fazendo correr o módulo hidrodinâmico do Delft3D, através de um canal retangular hipotético, de modo a ver qual o desempenho da execução e qual o ganho em usar esta ferramenta para este tipos de problemas. Avaliou-se também o uso de computação paralela em computadores pessoais no processamento do modelo aplicado.

1.3. Organização da Dissertação

Esta dissertação está dividida em seis capítulos, cada um focando-se sobre um aspeto do trabalho que foi desenvolvido.

No primeiro capítulo – "Introdução" – é apresentado o trabalho realizado nesta dissertação, enquadrando a importância e a utilidade da computação paralela para a modelação e a sua utilização na pesquisa em ciência e engenharia hoje em dia. São também enunciados os objetivos e motivações por trás deste trabalho e as bases gerais da metodologia utilizada para cumprir esses mesmos objetivos.

No segundo capítulo – "Revisão do Estado da Arte" – são abordadas as bases para a modelação computacional, ou seja, as equações em que a teoria é baseada. Também aí são abordados casos passados em que a modelação computacional foi utilizada na área da hidráulica, especialmente recorrendo a arquiteturas computacionais em paralelo, como os clusters informáticos. Finalmente, é explicado em que consiste o software Delft3d (com os seus diferentes módulo) e as suas características e aplicações.

No terceiro capítulo – "Ambiente de Computação Paralela Para Execução do Delft3D" – é pormenorizada a metodologia para utilização do *cluster* SeARCH da Universidade do Minho em modelação usando o programa Delft3d. Também são abordados os procedimentos necessários para utilizar computação paralela em computadores pessoais para a modelação com Delft3d. Neste capítulo são descritas as ferramentas utilizadas para avaliar o desempenho computacional nos exemplos de modelação que foram testados neste trabalho.

No quarto capítulo – "Caso de Estudo" – é descrito o modelo que serviu de exemplo de estudo. Aqui também é abordado o Cluster SeARCH, sendo descritos os seus atributos de *hardware* e as suas capacidades. Finalmente é apresentado o MPI, um *software* essencial no uso de computação paralela.

No quinto capítulo – "Análise e Discussão dos Resultados" – são apresentados e analisados os resultados. Analisam-se as diferenças de desempenho conforme se vão utilizando diferentes números de nós, sendo no tempo gasto ou na eficiência global do processamento dos modelos utilizando computação paralela.

No último e sexto capítulo – "Conclusões" – são apresentadas as conclusões que foram obtidas resultantes do trabalho realizado. São ainda apresentadas as possibilidades de desenvolvimento dos temas abordados nesta dissertação em futuros trabalhos.

2. REVISÃO DO ESTADO DA ARTE

2.1. Equações de Navier-Stokes

O tratamento matemático de problemas hidrodinâmicos tem como base as equações de Navier-Stokes que podem ser escritas do seguinte modo (Pinho, 2000),

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=\rho F_x-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) \quad (1)$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=\rho F_y-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) \quad (2)$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=\rho F_z-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right) \quad (3)$$

onde,

F_x, F_ye$F_z$são as componentes das forças de volume por unidade de massa [$N\ kg^{-1}$];

p é a pressão [Pa];

ρ corresponde à massa volúmica [$kg\ m^{-3}$]

μ é o coeficientes de viscosidade dinâmico [$kg\ m^{-1}\ s^{-1}$].

2.2. Equações de Reynolds

Das equações de Navier-Stokes são obtidas as equações de Reynolds, que caracterizam o movimento médio (média temporal) de uma partícula de fluido. As equações de Reynolds advêm da substituição do valor instantâneo da velocidade pela soma de um valor médio temporal com uma flutuação aleatória.

$$\rho\left(\frac{\partial \hat{u}}{\partial t} + u\frac{\partial(\hat{u}\hat{u})}{\partial x} + v\frac{\partial(\hat{u}\hat{v})}{\partial y} + w\frac{\partial(\hat{u}\hat{w})}{\partial z}\right) = \rho F_x - \frac{\partial \hat{p}}{\partial x} + \mu\left(\frac{\partial^2\hat{u}}{\partial x^2} + \frac{\partial^2\hat{u}}{\partial y^2} + \frac{\partial^2\hat{u}}{\partial z^2}\right) - \frac{1}{\rho}\left(\frac{\partial}{\partial x}\overline{u'u'} + \frac{\partial}{\partial y}\overline{u'v'} + \frac{\partial}{\partial z}\overline{u'w'}\right) \quad (4)$$

$$\rho\left(\frac{\partial \hat{v}}{\partial t} + u\frac{\partial(\hat{v}\hat{u})}{\partial x} + v\frac{\partial(\hat{v}\hat{v})}{\partial y} + w\frac{\partial(\hat{v}\hat{w})}{\partial z}\right) = \rho F_y - \frac{\partial \hat{p}}{\partial y} + \mu\left(\frac{\partial^2\hat{v}}{\partial x^2} + \frac{\partial^2\hat{v}}{\partial y^2} + \frac{\partial^2\hat{v}}{\partial z^2}\right) - \frac{1}{\rho}\left(\frac{\partial}{\partial x}\overline{v'u'} + \frac{\partial}{\partial y}\overline{v'v'} + \frac{\partial}{\partial z}\overline{v'w'}\right) \quad (5)$$

$$\rho\left(\frac{\partial \hat{w}}{\partial t} + u\frac{\partial(\hat{w}\hat{u})}{\partial x} + v\frac{\partial(w\hat{v})}{\partial y} + w\frac{\partial(\hat{w}\hat{w})}{\partial z}\right) = \rho F_z - \frac{\partial \hat{p}}{\partial z} + \mu\left(\frac{\partial^2\hat{w}}{\partial x^2} + \frac{\partial^2\hat{w}}{\partial y^2} + \frac{\partial^2\hat{w}}{\partial z^2}\right) - \frac{1}{\rho}\left(\frac{\partial}{\partial x}\overline{w'u'} + \frac{\partial}{\partial y}\overline{w'y'} + \frac{\partial}{\partial z}\overline{w'w'}\right) \quad (6)$$

onde,

$\hat{u}$, $\hat{v}$ e $\hat{w}$ são médias temporais das componentes da velocidade [m s^{-1}];

$\hat{p}$ é a média temporal da pressão [Pa];

u', v' e w' são flutuações das componentes da velocidade [m s^{-1}].

As equações de Reynolds na forma tridimensional e a equação da continuidade estabelecidas em termos de valores médios de $\hat{u}$, $\hat{v}$, $\hat{w}$, $\hat{p}$ e $\hat{c}$ são o ponto de partida para o estudo de escoamentos reais. Em domínios correspondentes a massas de água superficial, estas equações deverão sofrer as adaptações necessárias para a consideração das particularidades que lhes são inerentes: fundos pouco profundos predominantes e consideração de outras forças aplicadas, como sejam, forças de Coriolis devidas à rotação da Terra, variações da pressão atmosférica, atrito na superfície devido ao vento e a influência de gradientes de massa volúmica provocados pela presença de substâncias tais como o sal e poluentes. A integração das equações anteriores, segundo a direção vertical, permite obter sistemas de equações que podem ser resolvidas numericamente de modo eficiente em sistemas computacionais.

2.3. Modelação em Computação Paralela e *Clusters* Informáticos

Um *cluster* informático é um conjunto de dois ou mais sistemas computacionais independentes e subsistemas de armazenamento com o propósito de partilhar e aceder a recursos. São um tipo específico de arquitetura de computação paralela em que os nós podem ser computadores totalmente independentes, podendo chegar a ser, por exemplo, utilizados durante horas de "expediente" nas suas funções usuais e usados para participar em computação paralela durante as horas em que estariam, noutra situação, inativos. Essencialmente são computadores totalmente funcionais isoladamente que podem participar em operações de grande dimensão dentro da rede à qual estão ligados.

As características e vantagens da computação paralela com *clusters* podem-se resumir deste modo (Mauler, 2002):

- Alta disponibilidade, de modo a haver capacidade computacional disponível sempre que necessário;

- Expansibilidade, pois os *clusters* podem ver o seu tamanho ser expandido conforme as necessidades e também deve ser assim com o seu desempenho;

- Tolerância a falhas, já que possibilitam que as ocorrências de falhas não sejam detetadas pelos utilizadores pois existe sempre uma alternativa de *hardware* para resolver um pedido computacional, devido a existência dos múltiplos "nós";

- O controlo (a partilha e acesso aos recursos) e acesso ao *cluster* deve poder ser feito de qualquer ponto, ou seja, qualquer nó do cluster permite aceder às suas funções;

- Sistema com uma única imagem, pois cada ponto do *cluster* pode funcionar independentemente estando cada um desses pontos ligados de modo a oferecer um acesso unificado aos recursos do sistema;

- Desempenho de alto nível com uma relação custo/benefício favorável em relação a soluções mais "monolíticas".

O desempenho dos computadores é usualmente medida usando a Lei de Amhdal, que permite estudar como o desenvolvimento de um sistema computacional escalável varia à medida que se adicionam mais unidades de processamento, estabelecendo uma fronteira máxima na melhoria do desempenho com o aumento do número de nós (Quinn, 2003).

Na teoria, pode-se aumentar até o infinito a capacidade de um computador paralelo mas a verdade é que há várias limitações práticas a esse pressuposto. O que equivale a dizer que há um limite real ao incremento da velocidade de computação até ao ponto em que a adição de mais unidades de processamento não resultará em nenhum aumento da velocidade.

Quase todo o *software*, mesmo a correr num sistema paralelo, apresenta uma mistura de processos cuja execução poderá ser paralela e em série. A referida lei reflete o facto de, mesmo se teoricamente se poderia aumentar até ao infinito a velocidade da computação da fração paralela, não se poder alterar a velocidade da fração do código sequencial.

Os problemas resultantes da computação paralela são provocados, em parte, devido à necessidade neste tipo de arquiteturas computacionais de haver informação partilhada entre as diferentes unidades, o que provoca o aumento dos dados enviados entre eles, que por sua vez pode provocar um congestionamento no *bus*, que trata desta comunicação. Outro problema

vem da impossibilidade de dividir a aplicação em igualdade entre todos os processadores, o que vai fazer com que uns tenham mais "trabalho" a fazer do que outros, resultando que algumas unidades não serão aproveitadas em todo o seu potencial, pois estarão algum tempo sem atividade.

Problemas deste género poderão surgir ao longo dos testes e é importante saber identificá-los de modo a poder resolvê-los. Mas, apesar destes obstáculos, espera-se que o uso de um *cluster* para executar modelos hidráulicos resulte em aumentos significativos da velocidade de execução.

A modelação usando ferramentas informáticas pode utilizar vários métodos de aproximação numérica. Exemplo disso é a utilização do método das diferenças finitas no *software* Delft3D.

O método das diferenças finitas é baseado na resolução de equações diferenciais em pontos discretos. A aproximação de uma equação diferencial para uma forma discretizada e depois em um sistema de equações algébricas tendo em conta os valores das variáveis existentes em cada um dos pontos discretos, equivale à aproximação das derivadas parciais. Assim, sabem-se os valores apenas nesses pontos da malha discreta, o que equivale a uma aproximação do problema real. Quanto mais densa for a malha, mais próximo o modelo será da situação que é modelada, mas mais complexo será a sua resolução. A escolha da malha e da sua densidade é feita segundo critérios de exatidão e da praticabilidade da sua execução em ambiente digital.

2.4. Modelação de Problemas Hidráulicos em Computação Paralela

As zonas costeiras são importantíssimas nas mais variadas atividades humanas, sendo elas de lazer ou exploração de recursos, o que fazem delas áreas geográficas essenciais para o bem-estar económico e social das populações. Isso torna-se ainda mais evidente tendo em conta que cerca de metade da população mundial habita nessas zonas. As alterações climáticas e a ação do Homem têm levado a alterações morfológicas das zonas costeiras que têm vindo a afetar a atividade humana.

A modelação tem aqui um papel importante já que permite prever alterações de sistemas aquáticos e, assim, ajudar autoridades e técnicos na tomada de decisões para remediar e antecipar problemas variados, sejam em alterações de qualidade de água, morfologia da costa, entre outros. Portugal, pelas suas características geográficas, tem as alterações da costa como problema determinante. Como consequência, o estudo da costa portuguesa tem sido promovido em diferentes trabalhos como Granja *et al.* (2011) em que vários investigadores de áreas de diferentes domínios de conhecimento colaboram com o objetivo de criar um programa de monitorização para avaliar as alterações em zonas costeiras. Em Pinho (2000) e Pinho (2014) aplica-se modelação numérica para estudar sistemas costeiros.

Existem dois tipos de modelação. Os modelos físicos, usados há mais tempo, consistem em modelos em escala reduzida de situações reais. Os modelos computacionais consistem em simulações virtuais de situações reais, o que permite maior flexibilidade na alteração dos parâmetros e menores custos de simulação. Estas características fazem com que os modelos físicos se tenham tornado praticamente obsoletos e substituídos pelos modelos computacionais.

De seguida referem-se vários exemplos de diferentes situações e estudos em que a modelação se tornou essencial.

Os riscos futuros e as alterações das zonas costeiras podem ser analisados através da modelação como em Pereira (2010), em que é estudado a possível evolução do litoral de Aveiro. As alterações morfológicas em zona costeira também são analisadas em Wang (2014), especificamente para a zona de rebentação da costa belga. Outros trabalhos idênticos são, por exemplo, o de Jamal (2012) que modela uma praia formada principalmente por cascalho e o efeito a longo prazo de agitação e marés sobre ela.

Situações extremas, como a ocorrência de *tsunami* podem ser modeladas, podendo assim prever-se as suas consequências como em Gelfenbaum (2007) ou ainda a previsão dos efeitos de ondas resultantes de tempestade numa baía (Sedigh, 2014). As alterações sobre sistemas

naturais promovidas pelo Homem têm na modelação uma ferramenta essencial para as decisões relacionadas com elas, pois permite ter ideia das consequências dessas alterações. Há vários exemplos disso, relacionados com a construção de quebra-mares (Bos, 1996), quebra-mares submersos (Villani, 2012), esporões (Rocha, 2011), dispositivos de aproveitamento da energia das marés (Chatzirodou, 2014), alimentação de praias com areia (Yuan, 2014). Na dissertação de Reis (2010) é utilizada a modelação para se estudar se a opção de intervir na costa tem realmente benefícios que compensam os seus custos.

A alteração da morfologia de fundos marinhos também pode ser parcialmente consequência da ação biológica, sendo isso estudado e modelado para o caso do Mar Frísio em Bos (2007).

2.5. Programa Delft3d

O programa Delft3d é um *software* de modelação usado em problemas hidrodinâmicos, transporte de sedimentos, morfologia e qualidade de água em sistemas fluviais, estuarinos e costeiros.

O programa Delft3D é constituído por vários módulos, cada um abordando problemas diferentes (Figura 1). Esses módulos podem trocar informação e resultados entre si de modo a resolver problemas mais complexos, interpretando um grande número dos fatores que se vão alterando ao longo do tempo num sistema aquático (Deltares, 2014).

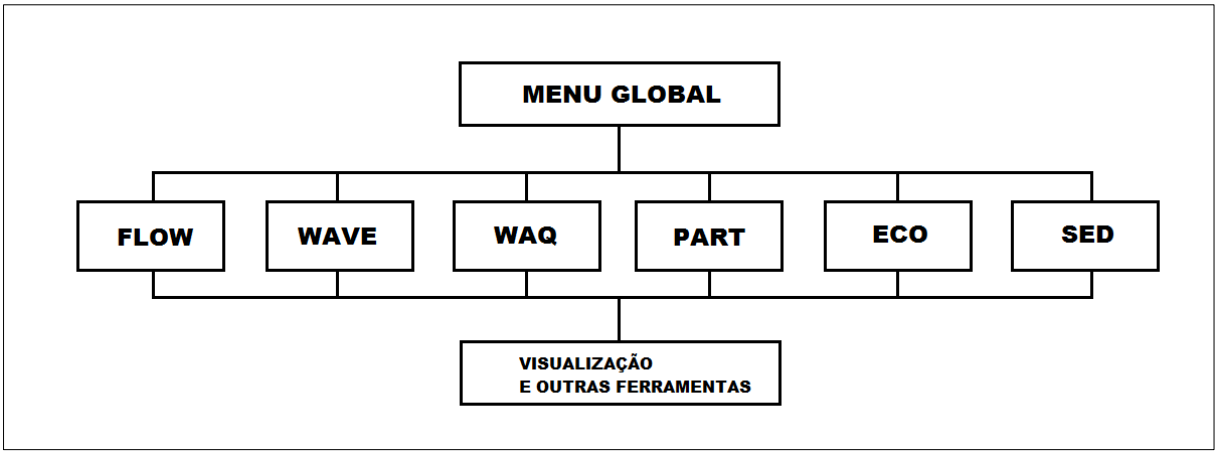


Figura 1 – Esquema dos módulos do Delft3d

O módulo Flow é um programa de simulação multidimensional hidrodinâmico que calcula escoamento variável e transporte de substâncias. É baseado nas equações de Navier-Stokes com a aproximação de águas pouco profundas. Pode-se aplicar em várias situações, tal como a simulação de caudais fluviais, simulação de lagos profundos e albufeiras, intrusões salinas em estuários, descargas de água doce em baías, transporte de sedimentos, material dissolvido ou poluente entre outros.

Alguns dos atributos que caracterizam o módulo FLOW são a inclusão de:

- Marés em fronteiras abertas;

- A força de Coriolis (efeitos resultantes da rotação da Terra);

- Correntes resultantes dos gradientes de densidade;

- Os efeitos resultantes da variação no tempo e espaço da atuação do vento e da pressão atmosférica;

- Tem em conta a turbulência e difusão baseada no conceito de viscosidade turbulenta;

- A variação no tempo de fontes e sumidouros;

- Simulação de descargas térmicas, descargas de afluentes em qualquer local e qualquer profundidade;

- Simulação de enchentes e vazantes devidas à maré;

- Possibilidade por optar por vários tipos de coordenadas (retilíneo, curvilíneo ou esférico);

- Possibilidade de ter em conta as trocas de calor através da superfície livre da água;

- Fluxos e forças resultantes da agitação;

- Influência da agitação nas forças de atrito em fundos;

- Possibilidade da inclusão de estruturas como pontes, esporões, entre outros.

Para além deste módulo, que é o mais pertinente para os trabalhos desta dissertação, explica-se logo a seguir os outros módulos e as suas funções e aplicações.

O módulo Wave tem como função a simulação de ondas geradas pela ação do vento em estuários, embocaduras, lagos e outros; para além da interação não linear entre ondas e a sua dissipação, tendo em conta a batimetria, o vento, nível da água e as correntes subaquáticas.

O módulo Water Quality tem como propósito situações de transporte de substâncias em águas superficiais ou subterrâneas. É baseada na equação de advecção-difusão e reação relativa à qualidade de água (as transformações que ocorrem em substâncias na água) e tem usos vários como a Avaliação de Impacto Ambiental, o estudo de descargas poluentes, eutrofização ou ciclo de nutrientes.

No módulo Ecology é tratada a modelação das características de massas de água referentes a algas em diferentes aspetos como na simulação de biomassa, dinâmica das comunidades de algas na competição por luz e nutrientes, adaptação a ecossistema e composição por espécies, concentração entre outros.

O módulo Particles permite a simulação da distribuição de concentração da libertação, pontual ou contínua de uma descarga de substâncias, tal como sal, petróleo ou outras.

O módulo Mor tem como função simular problema de morfodinâmica, ou seja, mudanças da morfologia de rios, estuários ou do litoral ao longo do tempo. Considera os efeitos das ondas, correntes, transporte de sedimentos de várias dimensões, indo de siltes até cascalho.

De seguida fala-se de como esses sistemas de equações são aplicados na criação de modelos no software Delft3D.

O Delft3D-FLOW resolve as equações de Navier-Stokes para um fluido incompressível, considerando as aproximações de águas pouco profundas e de Boussinesq, podendo ainda ser acoplados modelos de turbulência disponíveis para a solução hidrodinâmica. A aceleração vertical é ignorada na equação de conservação da quantidade de movimento segundo a vertical, o que leva ao uso da equação de pressão hidroestática. Em modelos 3D, a componente vertical da velocidade é calculada a partir da equação de continuidade. O conjunto de equações diferenciais parciais, em combinação com um conjunto apropriado de condições iniciais e de fronteira, são resolvidas numa grelha de diferenças finitas.

O Delft3D-Flow usa coordenadas ortogonais rectangulares ou curvilíneas na direção horizontal. Na direção vertical o Delft3D-FLOW permite o uso de dois tipos de sistemas de coordenadas: o sistema de coordenadas σ e de coordenadas cartesianas.

No sistema de coordenadas sigma a malha vertical consiste em camadas limitadas por dois planos que não serão restritamente horizontais, mas seguem sim a batimetria e a superfície livre. O número de camadas ao longo da área do modelo é constante, não levando em conta a variação local da profundidade. A espessura de cada camada não é uniforme, de modo a termos mais detalhes nas zonas que o pedem como as zonas próximas da superfície da água e próximos do leito. O sistema de coordenadas σ é definido como:

$$\sigma = \frac{z - \zeta}{d + \zeta} = \frac{z - \zeta}{H} \tag{7}$$

Em que:

z é a coordenada vertical no espaço físico

ζ é a elevação em relação ao plano de referência (z=0)

d é a profundidade em relação ao plano de referência (z=0)

As condições de fronteiras abertas são fronteiras virtuais "água-água". São introduzidos para limitar o tamanho do modelo e reduzir o esforço computacional. Numa fronteira aberta o nível da água, a velocidade ou uma combinação tem que ser definida tal como a velocidade para a fronteira onde ocorre uma descarga. A equação da continuidade apresenta a seguinte forma:

$$\frac{\partial \zeta}{\partial t} + \frac{1}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}} \frac{\partial \left[(d + \zeta) U \sqrt{G_{\eta\eta}}\right]}{\partial \xi} + \frac{1}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}} \frac{\partial \left[(d + \zeta) U \sqrt{G_{\xi\xi}}\right]}{\partial \eta} = Q \qquad (8)$$

U velocidade média na direção ξ

V velocidade média na direção η

ξ, η coordenadas nas direcções de X e Y, respectivamente

O termo Q representa a contribuição por unidade de área devida à descarga ou saída de água, precipitação ou evaporação:

$$Q = H \int_{-1}^{0} (q\ in\ -\ q\ out\ ) d\sigma\ +\ P\ -\ E \qquad (9)$$

Sendo o q_{in} e o q_{out} as fontes e saídas de água por unidade de volume respetivamente, P representa a precipitação e E as perdas devidas à evaporação.

As equações de conservação da quantidade de movimento segundo as direções xx e yy apresentam a seguinte forma:

$$\frac{\partial u}{\partial t}+\frac{u}{\sqrt{G_{\xi\xi}}}\frac{\partial u}{\partial \xi}+\frac{v}{\sqrt{G_{\eta\eta}}}\frac{\partial u}{\partial \eta}+\frac{\omega}{d\ +\ \zeta}\frac{\partial u}{\partial \sigma}-\frac{v^2}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}}\frac{\partial\sqrt{G_{\eta\eta}}}{\partial \xi}+\frac{uv}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}}\frac{\partial\sqrt{G_{\xi\xi}}}{\partial \eta}-fv$$
$$=-\frac{1}{\rho_0\sqrt{G_{\xi\xi}}}P_\xi+F_\xi+\frac{1}{(d\ +\ \zeta)^2}\frac{\partial}{\partial\sigma}\left(v_v\frac{\partial u}{\partial\sigma}\right)+M_\xi \quad (10)$$

$$\frac{\partial v}{\partial t}+\frac{u}{\sqrt{G_{\xi\xi}}}\frac{\partial v}{\partial \xi}+\frac{v}{\sqrt{G_{\eta\eta}}}\frac{\partial v}{\partial \eta}+\frac{\omega}{d\ +\ \zeta}\frac{\partial v}{\partial \sigma}+\frac{uv}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}}\frac{\partial\sqrt{G_{\eta\eta}}}{\partial \xi}-\frac{u^2}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}}\frac{\partial\sqrt{G_{\xi\xi}}}{\partial \eta}+fu$$
$$=-\frac{1}{\rho_0\sqrt{G_{\eta\eta}}}P_\eta+F_\eta+\frac{1}{(d\ +\ \zeta)^2}\frac{\partial}{\partial\sigma}\left(v_v\frac{\partial v}{\partial\sigma}\right)+M_\eta \quad (11)$$

A componente vertical da velocidade ω no sistema de coordenadas σ é obtida a partir da equação de continuidade:

$$\frac{\partial\zeta}{\partial t}+\frac{1}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}}\frac{\partial\left[(d+\zeta)u\sqrt{G_{\eta\eta}}\right]}{\partial\xi}+\frac{1}{\sqrt{G_{\xi\xi}}\sqrt{G_{\eta\eta}}}\frac{\partial\left[(d+\zeta)v\sqrt{G_{\xi\xi}}\right]}{\partial\eta}+\frac{\partial\omega}{\partial\sigma}$$
$$=H(q_{in}-q_{out}) \quad (12)$$

ω é a componente vertical da velocidade em relação ao plano σ.

3. AMBIENTE DE COMPUTAÇÃO PARALELA PARA EXECUÇÃO DO DELFT3D

3.1. *Software* Utilizado

Esta metodologia foi executada e testada para Windows 7 e com instalação de MPICH2 1.0.8p1. A instalação noutras versões de Windows e de outras versões de MPICH2 será similar, mas talvez não totalmente análoga.

3.1.1. Biblioteca MPI (*Message Passing Interface*)

O MPI surgiu da necessidade de se criar uma ferramenta para computação paralela que fosse compatível com vários tipos de arquitetura. A troca de dados é necessária em computação paralela com memória distribuída, ou seja, quando um processador não tem acesso direto ao espaço de endereçamento de outro processador.

MPI é o acrónimo de *Message-Passing Interface*. Este tipo de interface apresenta um conjunto de instruções e sub-rotinas que permite dividir uma aplicação para execução paralela. Os dados são divididos e transmitidos para outros processos (quer dizer, diferentes nós computacionais, diferentes *cores* dentro de um nó ou até no mesmo *core*, havendo a partilha dos recursos temporais) como mensagens (Dowd, 1998).

As mensagens de MPI, no caso simples de serem *point-to-point*, têm como característica, quando são corretamente criadas, responder a perguntas em relação a quais serão os processos de envio e receção, as características da informação transmitida (localização, tipo, tamanho, local de envio) e a quantidade de informação que o processo recetor está em condições de aceitar (Hager, 2011). Para casos mais complexos do que a transmissão de dados entre dois processos, os parâmetros das mensagens são similares.

O programa de instalação da biblioteca MPICH2 está contido num ficheiro compactado. Usando um *software* adequado para o efeito (WinRAR é uma das aplicações mais comummente utilizadas para isso) poderemos descompactar a pasta lá incluída intitulada MPICH2 (recomenda-se fazer isso para a raiz do disco rígido, ou para a raiz de uma das partições do disco, pois isso facilita os procedimentos de utilização do programa).

Depois deste passo inicial, deverá ser colocado o caminho da pasta de instalação do MPICH2 na variável de ambiente de definição do "Path".

Conforme se ilustra na Figura 2, ir a "Iniciar" e depois ir a "Painel de Controlo". Aí tem-se na barra lateral direita várias opções. Clicar em "Definições avançadas do Sistema" e ir para a janela "Avançadas".

Aí escolher a opção "Variáveis de Ambiente" onde se terá que colocar o endereço da pasta "bin" incluída dentro de MPICH2. No caso de MPICH2 estar na raiz do disco "C:" equivale a carregar em "Editar" na variável Path das "Variáveis de Sistema" e colocar "C:\mpich2\bin" na linha do valor da variável clicando finalmente em OK (Figura 3).

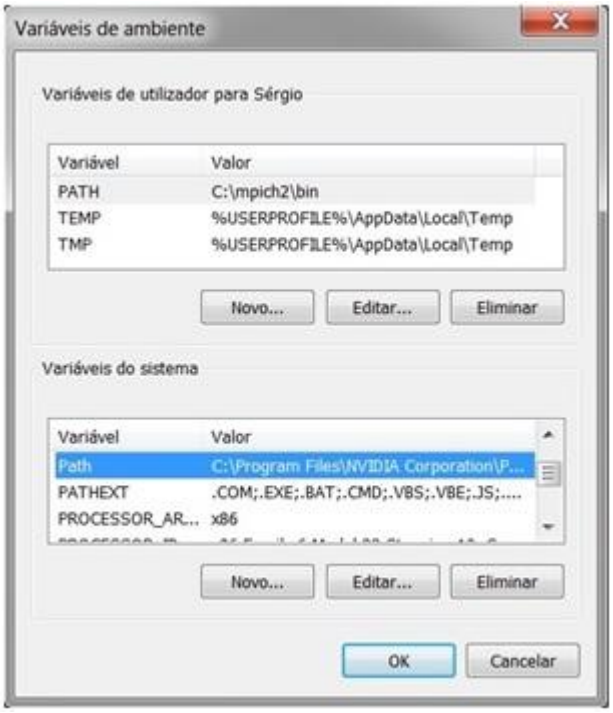


Figura 2 - Quadro de variáveis de ambiente

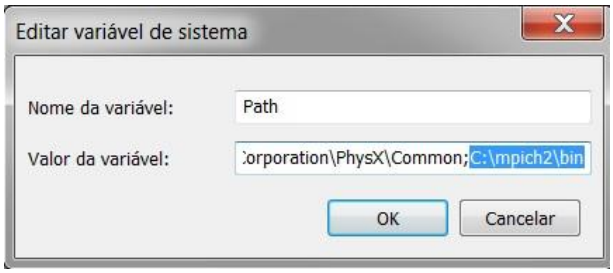


Figura 3 - Edição de variável de Sistema Path

No espaço para o valor de variáveis estão vários endereços de pastas do computador com diferentes funções. As que já se encontram lá são para manter e também tem que se certificar que o novo endereço está separado por um “;” dos outros endereços.

A instalação é realizada acedendo à pasta C:\mpich2\bin e fazendo correr os dois ficheiros executáveis lá inseridos. Isto é, o ‘mpich2.exe’ e o ‘smpd.exe’.

De seguida deverá fazer-se o registo do programa.

Abrir a Linha de Comandos como administrador, (opção disponível ao clicar com a esquerda na aplicação Linha de Comandos), e executar os seguintes comandos em qualquer localização:

b.1. "smpd -install"
b.2. "mpiexec -remove"
b.3. "mpiexec -register" – Aqui vai ser pedido onome de utilizador e a password. Se o PC não for parte de um domínio não é preciso especificar o nome de utilizador.
b.4. "mpiexec -validate" (Se tudo correu bem a resposta obtida será "SUCCESS")
b.5. "smpd -status" (A resposta esperada é "smpdrunningon<nome de host>")

Ao usar o smpd e o mpiexec, poderão surgir restrições da *firewall* aparecendo janelas de aviso do Windows, isso é usual e poderá ser facilmente resolvido. Basta dar a permissão de administrador para correr essas aplicações.

Completados os passos anteriores poder-se-á executar o MPI em qualquer pasta, (mas sempre no ambiente da linha de comandos).

A execução de um programa é efetuada com o seguinte comando:

mpiexec -n 2 <colocar aqui nome do programa que se quer correr>

onde:
mpiexec – corresponde à execução de um programa usando o mpi
-n 2 – corresponde ao número de processadores ou cores que pretendem ser usados (2,4, 8, por aí fora, dependendo do objetivo e da capacidade do sistema em que é executado). No caso apresentado os *cores* usados foram apenas dois.

A definição do máximo de *threads* de processamento para execução de um modelo depende das características de *hardware* do computador pessoal que será utilizado. A opção ideal será, à partida, utilizar o máximo de *threads* definidas no computador de modo a utilizar todas as

suas potencialidades. A definição de *threads* de processamento em maior número que esse em princípio não trará benefícios, já que a criação de *threads* será virtual não se adaptando às características do *hardware* do PC.

Se não se conhecerem as características do processador ao nível do número de *cores* bastará aceder ao Gestor de Tarefas do Windows e aí ao Monitor de Recursos em que se pode ver qual o número de processadores individuais que estão disponíveis para computação paralela. No caso da Figura 4 o processador do computador terá quatro *cores*.

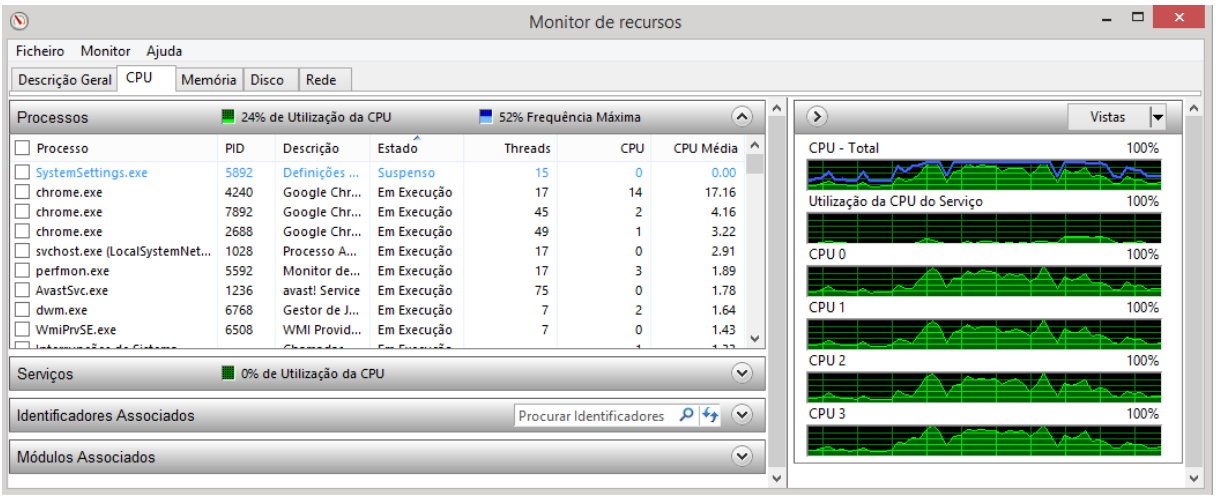


Figura 4 – Janela do monitor de recursos no Windows 7

3.1.2. Acesso ao cluster SeARCH da Universidade do Minho

Para além de se poder executar programas em paralelo num PC comum, existem recursos computacionais na Universidade do Minho de grande potência que permitem resolver problemas de maior dimensão em tempo muito mais curto. Fala-se aqui do *cluster* computacional SeARCH, gerido pelo Departamento de Informática da U.M. mas disponível para utilização de muitos diferentes departamentos desta instituição.

Utilitário Putty

O Putty é um *software* emulador de terminal disponível de modo gratuito e com código aberto. Suporta SSH, Telnet e RLogin, três modos de se aceder remotamente a computadores multi-utilizadores tal como o SeARCH.

Neste trabalho foi utilizada a versão Putty 0.63.

Descarregar o ficheiro Putty.exe do *website* seguinte: http://www.putty.org/

O programa não precisa de instalação e irá correr instantaneamente ao abrir-se o ficheiro.

Ao abrir, escreve-se no campo "Host Name (or IP Adress)" o seguinte: search.di.uminho.pt

No campo "Connection Type" escolher a opção "SSH".

Guardar estas opções clicando em "Save" fazendo delas as nossas configurações por defeito.

Depois de isso clicar, na opção "Open" no fundo da janela (Figura 5).

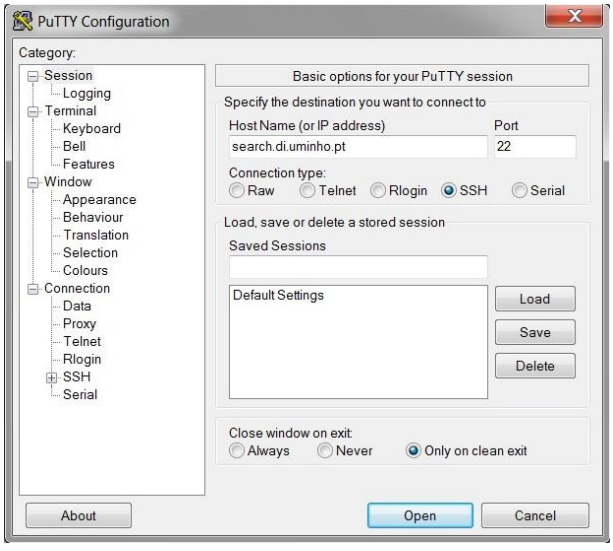


Figura 5 – Visualização do *software* Putty

Daí acede-se ao servidor. (Poderá surgir um aviso a dizer que a chave de acesso ao *host* não está no nosso registo, mas pode-se continuar sem qualquer problema.)

Colocar o “username” e a “password” necessários para entrar no servidor.

Instalação do WinSCP

Esta ferramenta é utilizada aqui para a transferência de ficheiros entre o servidor do *cluster* SeARCH e um computador pessoal. A versão utilizada, e consequentemente à qual se aplicaram estas instruções correspondentes, foi a 5.7 podendo ser necessário procedimentos diferentes para outras versões deste programa.

Este *software* pode-se instalar gratuitamente acedendo à seguinte página: http://winscp.net/eng/download.php

Aí existem várias hipóteses para a sua instalação. Sugere-se a versão “*portable*” que não precisa de instalação.

O ficheiro descarregado é um ficheiro “zip” havendo no seu interior um executável chamado “WinSCP.exe”. Bastará abrir esse ficheiro para aceder ao programa.

Aí existem vários campos que deverão ser preenchidos (Figura 6):

- o nome do servidor (“HostName”) que será nesse caso: search.di.uminho.pt

- “User Name” e “Password” que correspondem ao nome de utilizador e palavra-passe e terão que ser obrigatoriamente preenchidos para obter acesso ao SeARCH.

- Não esquecer o campo “File Protocol” onde terá que ser selecionado a opção SFTP.

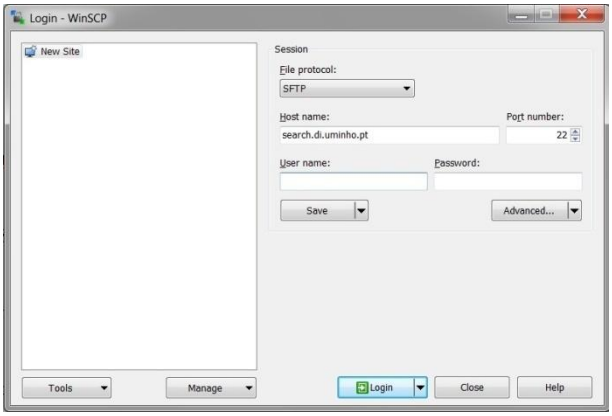


Figura 6 – Visualização do *software* WinSCP - login

Abrir-se-á uma janela em que, do lado esquerdo está a janela dos ficheiros no PC do qual estamos a aceder ao SeARCH e, do lado direito estão os ficheiros do servidor correspondente ao nosso nome de utilizador.

A cópia de ficheiros entre um computador e outro pode-se efetuar pelo simples arrastamento dos mesmos entre as janelas (Figura 7).

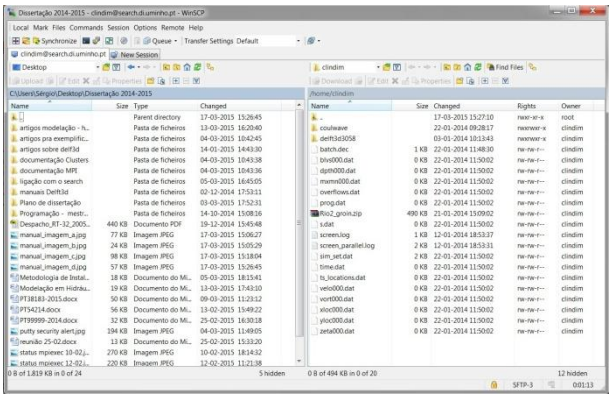


Figura 7 - Visualização do *software* WinSCP

Utilização do Delft3D no *cluster* SeARCH

Para se executar o programa em paralelo no *cluster* SeARCH deverão ser evocados os seguintes comandos:

1) module add intel/intel-11
2) module add intel/mpich2-1.5-intel-11 (Adiciona as bibliotecas dos compiladores Intel Fortran 11 e o Mpich2 1.5.)
3) source export.sh (Este comando permite executar o que está contido no ficheiro export.sh)
4) qsub rio2.sub

O comando qsub permite colocar um *job* (programa ou tarefa) em fila para ser executado no *cluster* quando houver disponibilidade para tal.

A linha de código deste ficheiro com extensão "*.sub" que corresponde à definição do nº de nós a ser utilizados na execução é a seguinte:

./run.sh 2

O número "-2" corresponde ao nº de nós a serem utilizados. Podemos optar por diferentes números de nós (2, 4, 8 etc) a serem utilizados.

Ao executar este comando temos como retorno o número da aplicação dentro do *cluster* de modo a sabermos identificá-la ou aplicar nela algum comando. Como por exemplo:

450863.search.di.uminho.pt

Para saber o estado do programa em relação à sua execução pode-se usar o comando "Qstat".

Ele permite-nos ver todas as tarefas que estão na fila de execução ou já a ser executados pelo SeARCH, tal como o tempo na qual tem sido executado para além de mais informação (fila de execução, nome, utilizador).

Se, afinal, não quisermos que um programa seja executado podemos apagá-lo usando:

qdel <nº de identificação>

O ficheiro rio2.sub é composto por comandos de PBS que serão executados sequencialmente de modo a definir as características da execução do modelo e os resultados finais obtidos.

PBS é um acrónimo de *Portable Batch System*, que designa o *software* que permite o agendamento de tarefas. Torque é o nome do PBS usado especificamente no *cluster* SeARCH.

Em seguida vemos qual é a função dos diferentes comandos de PBS utilizados no código utilizado no ficheiro rio2.sub (assinalados no início de cada linha com um "#PBS").

"-N" serve para especificar o nome da tarefa.

"-M" serve para especificar os e-mails para os quais são enviados avisos relativos à tarefa.

"-m e" define as condições em que um e-mail é enviado, sendo o "e" para especificar que um e-mail será enviado quando a tarefa terminar.

"-j eo" serve para definir que o ficheiro de output e o ficheiro de erro

"-V" declara que as variáveis executadas pelo qsub são exportadas para o batchjob.

"-l walltime=20:00:00" declara o tempo que é pedido para a operação

"-W x=NACCESSPOLICY:SINGLEJOB" serve para restringir cada nó utilizado à tarefa que estamos a processar.

“cd $PBS_O_WORKDIR # /home/cpd25326/PI/rio2” define o endereço de trabalho para o comando qsub.

O output da execução inclui ficheiros onde as características do desempenho em cada um dos nós de execução são descritas. Esses ficheiros são intitulados tri-diag.rio2-001, tri-diag.rio2-002, …, tri-diag.ri02-00n.

Cada um deles corresponde a um nó e, no fim da execução, temos aí a informação sobre a execução em cada nó. A informação mais relevante pode-se encontrar num quadro deste tipo (Figura 8).

```
Timer name            |       wall clock      |       CPU time
                      |-----------------------|-----------------------
                      |       sec     |  %    |       sec     |  %
-------------------------------------------------------------------------
Initialization        |         5.23  |  0.0  |         1.82  |  0.0
Simulation            |     11368.93  | 99.9  |     11343.56  |100.0
Close and stop        |         3.70  |  0.0  |         1.39  |  0.0
                      |-------------- +|----- +|-------------- +|----- +
Total                 |     11377.87  |100.0  |     11346.77  |100.0
------------------------------------------------|-----------------------
Momentum eq.          |      3400.79  | 29.9  |      3400.42  | 30.0
Continuity eq.        |      2103.50  | 18.5  |      2103.37  | 18.5
Transport eq.         |         0.00  |  0.0  |         0.00  |  0.0
Turbulence            |      3356.59  | 29.5  |      3356.35  | 29.6
3D Morphology         |         0.00  |  0.0  |         0.00  |  0.0
Wait (dd   module)    |         0.00  |  0.0  |         0.00  |  0.0
Wait (ext. modules)   |         0.00  |  0.0  |         0.00  |  0.0
-------------------------------------------------------------------------
 Performance = CPU time / (TimeSteps*MMAX*NMAX*KMAX*LMAX)
   TimeSteps   :        2880
   MMAX        :         754
   NMAX        :          97
   KMAX        :          10
   LMAX        :           2
   Performance : 0.26934E-05 [sec]
```

Figura 8 – Quadro do *output* da computação de um nó

De seguida explicam-se a que correspondem os dados aí incluídos:

Wall clock – corresponde ao tempo total utilizado na execução do programa, incluindo atrasos programados, ou tempo para os recursos ficarem disponíveis/ tempo de Input/Output e atrasos no canal de comunicação.

CPU time – tempo usado especificamente para o processamento do programa.

Initialisation – tempo usado para inicializar a execução da simulação.

Close and stop – tempo usado para finalizar a simulação.

Através destes ficheiros de output também podemos saber qual o tempo de execução de cada uma das equações utilizadas na modelação do problema, em tempo utilizado e em percentagem.

Por fim, temos acesso à informação sobre o desempenho na modelação, dado esse baseado em determinados dados relativos à modelação em cada nó. Estes dados (Kmax, Nmax, Mmax, Lmax) são relativos à dimensão da grelha calculada pelo nó específico, que é uma parcela de todo o modelo, sendo este o modo como as tarefas são divididas entre os nós, dividindo o domínio em subdomínios parcelares.

Cada quadro de dados é obtido para o desempenho em cada nó, ou seja, numa computação paralela em que são utilizados, por exemplo, 8 nós, teremos 8 conjuntos de resultados. Desta forma, para comparar os desempenhos entre as diferentes hipóteses que testamos em computação paralela, escolhemos o valor mais desfavorável de cada conjunto de dados em cada estatística obtida, para análise e comparação dos resultados com outros casos.

Instalação do Delft3d

O Delf3d GUI (*Graphic User Interface*) permite o uso de todas a potencialidades do Delft3d.

1) Instalar o "Microsoft Visual C++ 2008 SP1 Redistributable Package" que consiste em componentes das bibliotecas de Visual C++ permitindo correr aplicações criadas usando o Visual C++ 2008, o que o torna obrigatório para correr este programa em específico. O ficheiro que tem que ser executado para proceder à instalação é o "vcredist_x86 (2008 SP1).exe". Ele pode ser encontrado no *website*: https://www.microsoft.com/en-us/download/details.aspx?id=5582

2) Outra instalação preliminar necessária é o "Matlab Runtime". Trata-se de um conjunto de bibliotecas que permite a execução de aplicações e componentes compiladas de Matlab. Isso é conseguindo descarregando e instalando o ficheiro MCRInstaller.exe que se pode encontrar no *website* http://www.mathworks.com/products/compiler/mcr/

3) A instalação do programa é feita executando o setup.delft3d.openhydromorpho.exe.

Utilização do Quickplot

A utilização do Quickplot, aplicação utilizada para a obtenção de visualizações dos modelos e alterações das suas propriedades ao longo do tempo, também depende da instalação das bibliotecas de Visual C++ e Matlab para o seu funcionamento.

3.2. Ferramentas de avaliação de desempenho na execução do modelo

A eficiência de uma execução paralela pode ser obtida com a seguinte fórmula:

$$\frac{S}{n * T_{(n)}} \tag{13}$$

Em que S é o tempo total de execução do modelo numa arquitetura sequencial, n é o número de processadores (nós) utilizado na máquina paralela que executa o modelo e T(n) o tempo total de execução demorado (Hager, 2011).

Isso permite-nos comparar o tempo necessário para a execução de uma tarefa num computador sequencial com a soma total do tempo demorado por cada nó de um computador paralelo para o mesmo problema. Isso permite saber se o uso dos recursos computacionais é eficiente. Ou seja, comparar a ocupação dos recursos de *hardware*. Se o resultado for maior que um, quer dizer que a eficiência do processamento paralelo é maior do que do sequencial.

Para se analisar o quão significativo é a melhoria do desempenho conforme se aumenta o número de nós da execução do modelo, usa-se a relação entre o tempo total da execução do modelo num único nó do *cluster* (*T1*) e o tempo de execução no cluster usando vários nós (*Tn*).

$$\frac{T_n}{T_1} \tag{14}$$

Assim, a razão entre estes dois intervalos de tempo também pode ser uma ferramenta útil para analisar a pertinência do uso de computação paralela.

$$tempo\ útil\ em\ percentagem = \frac{Tempo\ de\ processamento}{Tempo\ Total} * 100 \quad (15)$$

O tempo de processamento é o tempo utilizado na execução no processador, (sem contar com o tempo gasto em tarefas de *Input/Output* ou em atrasos, programados ou não), enquanto o tempo total corresponde ao tempo total demorado para a execução do programa. Quanto mais próximo de 1 for a razão entre os dois, menor será o tempo usado para tarefas que não o processamento dos dados, tempo que pode ser considerado como sendo "tempo útil".

Desempenho no Delft3d

O desempenho na execução de um modelo com o Delft3d é avaliado segundo a seguinte expressão:

$$\frac{Tempo\ de\ processamento}{N * Mmax * Nmax * Kmax * Lmax} \quad (16)$$

Em que N equivale ao número de intervalos de tempo de execução (corresponde à divisão do problema em diferentes intervalos de tempo de modo a descrever a evolução de um problema ao longo do tempo), Nmax e Mmax correspondem ao número de pontos da malha do modelo na direção dos XX e dos YY, respetivamente, Kmax é o número de camadas da malha na vertical e Lmax é o número de constituintes analisados (referindo-se aos constituintes que podem ser analisados no Delft3d Flow que são salinidade, temperatura, sedimentos, poluentes e traçadores) (Deltares, 2014). Os valores de cada uma dessas variáveis correspondem aos que são calculados em cada um dos nós computacionais. Pois a malha do modelo pode ser dividido pelos mais variados nós, sendo os resultados finais objeto da conjugação dos valores obtidos em cada um desses nós.

4. CASO DE ESTUDO

4.1. Descrição do modelo Rio2

O modelo Rio2 refere-se a um canal fluvial com 30 km de comprimento e 2 km de largura com a presença de esporões. Os esporões estão localizados em ambas as margens do canal, estando espaçados entre si em intervalos constantes em cada uma das margens e tendo como características uma extensão de 800 metros com profundidade que varia entre os 7 metros junto à margem reduzindo-se linearmente até os zero metros a uma distância de 800 metros da margem.

Foram consideradas as seguintes condições nas fronteiras abertas: a montante um caudal de 100 m^3/s e a jusante um nível constante. O modelo de turbulência tinha como característica os coeficientes de difusão turbulenta são calculados com base num modelo de turbulência do tipo k-ε.

A grelha do modelo tem uma resolução de 20 m x 20 m (Figura 9). Correspondendo a 96 de pontos na direção da largura e 1501 pontos na direção do comprimento do canal.

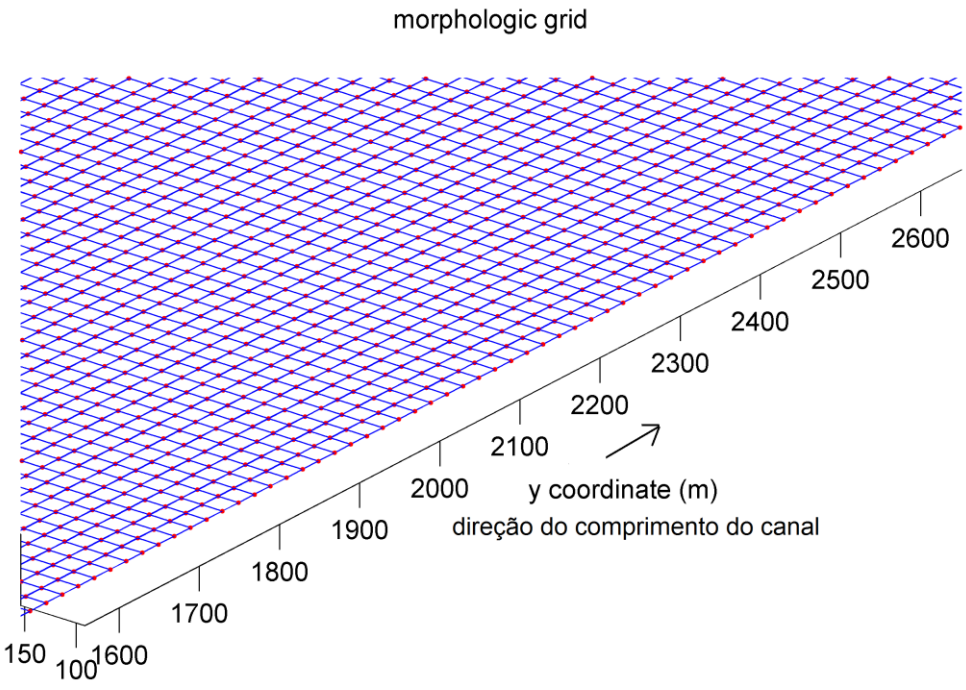


Figura 9 – Visualização parcial da grelha do modelo estudado

A secção transversal apresenta a forma representada na Figura 10.

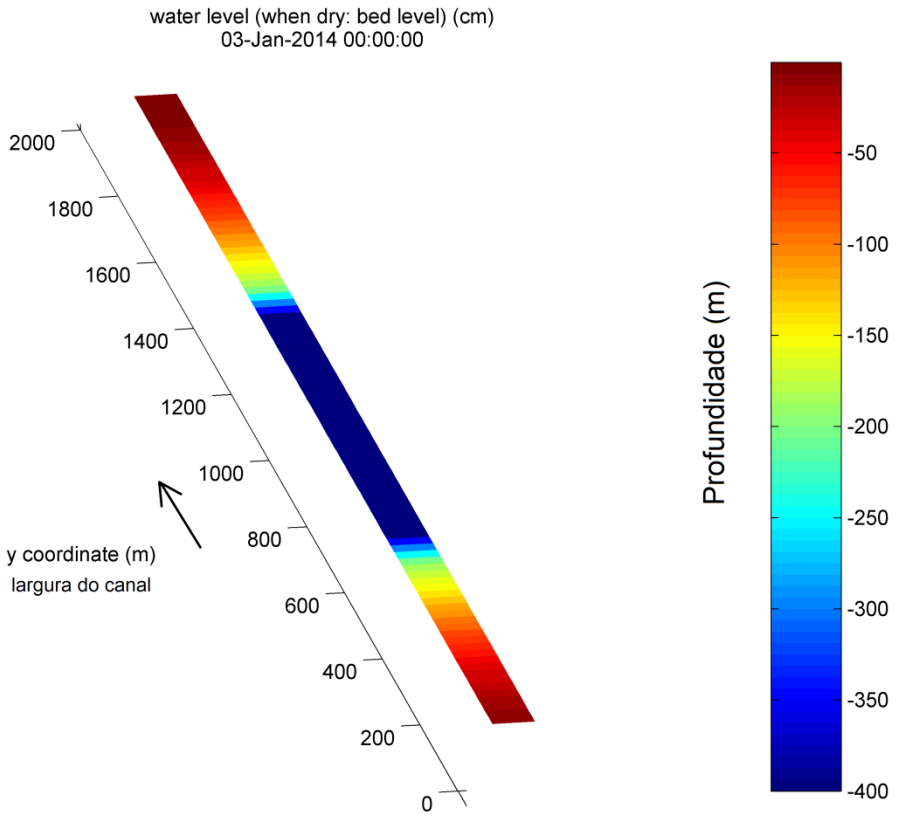


Figura 10 – Batimetria do canal num perfil transversal do modelo

4.2. Cluster SeARCH

O *cluster* SeARCH (Figura 11), responsabilidade do Departamento de Informática da Universidade do Minho, é um *cluster* informático utilizado para vários problemas de investigação relacionados com bastantes áreas de conhecimento estudadas nesta universidade.

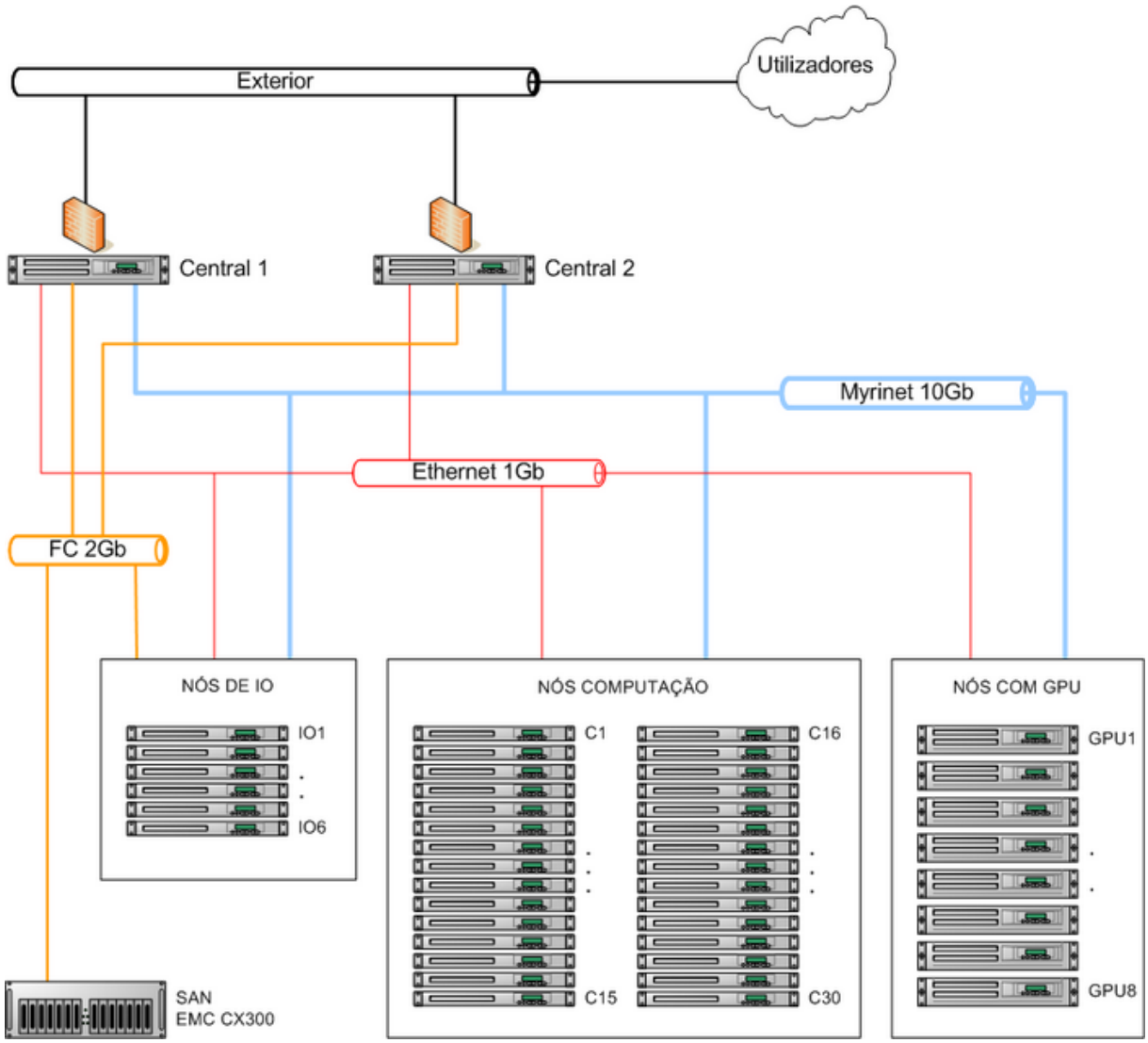


Figura 11 – Esquema do *cluster* SeARCH (fonte: search.di.uminho.pt)

Os seus elementos mais importantes são 46 nós, sendo dois deles nós centrais de gestão e os restantes sendo computacionais (havendo três subtipos: genéricos, com acesso privilegiado ao espaço de armazenamento ou orientados à computação gráfica). As características dos nós genéricos (e os outros, excetuando os nós para computação gráfica) são a existência de dois processadores de 3,2 GHz e 2 MB de Cache, tendo memórias RAM de 2 GB DDR2-400. Isto para além de uma memória central com grande capacidade e as redes de conexão que os ligam, do tipo Ethernet e Myrinet.

Esta é uma das ferramentas em que foi testado o desempenho de modelos hidráulicos quando são executados em *hardware* do tipo de computação paralela, sendo um caso particular pois é

uma ferramenta de computação de elevado desempenho ao contrário dos outros exemplos que foram testados na execução de computação paralela, que são simples computadores pessoais.

5. ANÁLISE E DISCUSSÃO DOS RESULTADOS

5.1. Desempenho no *cluster* computacional SeARCH

Para a análise da utilidade do recurso a *clusters* para a modelação de problemas hidráulicos complexos, o processo utilizado foi o de fazer correr um modelo, recorrendo a uma grande variedade de possibilidades no uso de recursos em relação ao número de nós computacionais utilizados.

Infelizmente, não foi possível a realização de mais testes no SeARCH devido a alterações deste *cluster* alheias a este trabalho e ocorridas durante a sua realização, que resultaram na ausência de bibliotecas necessárias à execução de computação paralela.

O modelo que foi testado como exemplo, designado de Rio2 simula um canal fluvial com 30 km de comprimento e largura constante de 2 km, foi discretizado utilizando uma grelha com 1501x96 pontos, foi executado utilizando diferentes números de nós do *cluster* SeARCH. Ou seja, executa-se o modelo usando diferentes quantidades de nós computacionais do *cluster* de modo a permitir a análise de quais são os benefícios obtidos pela aplicação deste tipo de recurso.

Assim, a resolução do problema foi testada utilizando um nó (o que equivale à execução num PC individual, de modo a termos uma base de comparação com os outros resultados obtidos), 2, 4, 6, 8, 10, 12, 14, 16 e 18 nós. Algumas das simulações foram repetidas uma segunda vez de modo a assegurar a consistência dos tempos de execução obtidos.

Pode-se, assim, verificar que a velocidade de execução do modelo em computação paralela (Figura 12) apresenta uma tendência para aumentar quase uniformemente o desempenho com o aumento de nós de computação. Logo na passagem de um para dois nós de computação há uma diminuição do tempo demorado em 64,9%. Isto acontece mais evidentemente na progressão de nós entre os dois e os doze, chegando o tempo de execução a ser

aproximadamente treze vezes mais rápido do que usando um computador individual. De notar também que, na passagem de dois para quatro nós, a diminuição do tempo de execução é muito menor do que a anterior (16,1%).

Ao chegar aos doze nós, o aumento de *clusters* deixa de resultar em diminuição do tempo de execução do modelo, havendo sim um aumento súbito (aproximadamente 37 %) ao passarmos dos doze aos catorze nós. Por causa disso, tentando descrever melhor o desenvolvimento dessa situação, também foi executado o modelo usando treze nós do SeARCH, observando-se que é na passagem da utilização de doze para treze nós que se encontra o aumento abrupto no tempo de processamento do modelo.

É difícil identificar as razões que levam a esta degradação das condições de execução do modelo no *cluster*, implicando esta pesquisa conhecimentos e análise mais aprofundadas do software Delft3d Flow e sobretudo da arquitetura de hardware do cluster o que não é objeto desta dissertação.

Depois dessa diminuição brusca, o aumento da velocidade de execução até aos dezoito nós é praticamente constante mas sempre inferior à conseguida com doze nós. Isso faz do uso de doze nós a melhor opção possível das que foram testadas se apenas se tiver em conta o tempo de execução.

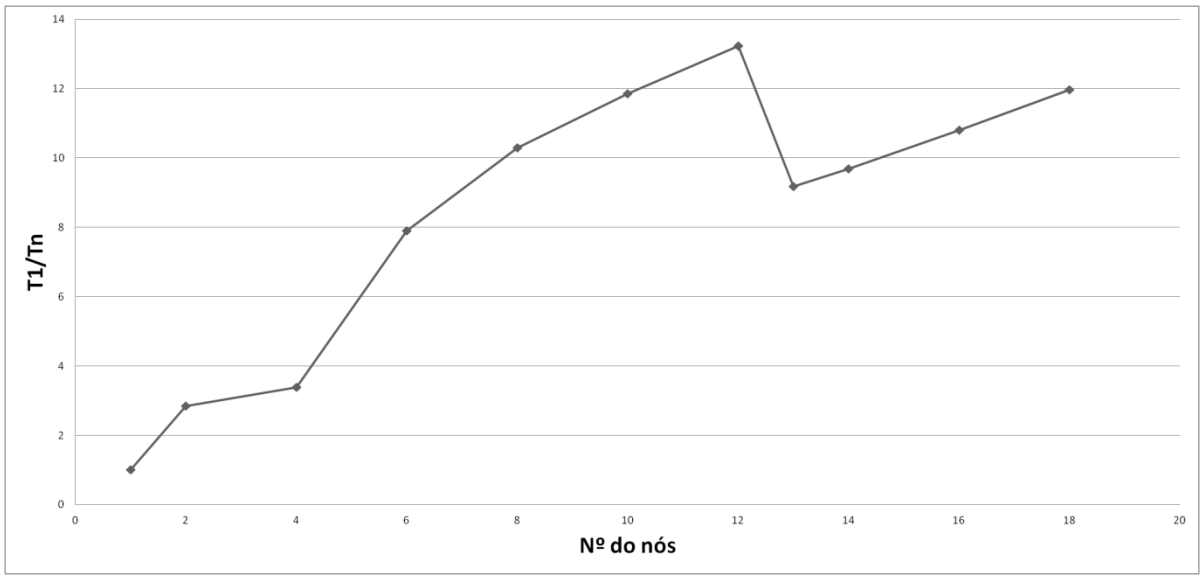


Figura 12 - Desempenho na execução do modelo no SeARCH variando o número de nós utilizados

Analisando as outras estatísticas obtidas, vê-se que existe forte correlação entre elas. A eficiência, Figura 14, permite estabelecer a relação entre a quantidade de recursos computacionais empregues e o ganho efetivo no desempenho. Ou seja, ao adicionar mais nós para a resolução do modelo ficar-se a saber se cada nó está a ter, individualmente, melhor desempenho ou não do que teria um computador sequencial.

Ao passar da execução sequencial para um computador paralelo com dois nós, nota-se que, para além de um incremento de velocidade de execução (Figura 12), há também um aumento da eficiência de cada nó (Figura 13). Isso acontece porque, ao haver dois nós a resolver o mesmo problema, pode-se dividir facilmente as tarefas entre nós. Divisão que se torna mais onerosa, possivelmente devido ao aumento das necessidades de comunicação, com o aumento do número de nós utilizados, o que se reflete na diminuição progressiva da eficiência (tal como ela é definida no capítulo 3). A partir dos catorze nós, o aumento dos nós levou à diminuição brusca do tempo de execução, sendo que a eficiência também diminui bruscamente ao passar-se de doze para catorze nós havendo, a partir daí, diminuição constante e progressiva e logo, valores de eficiência sempre menores às de um computador sequencial. De notar também, que ao passar-se de 2 para 4 nós computacionais ocorreu uma diminuição brusca na eficiência que não conseguimos explicar, principalmente tendo em conta que existe um aumento de seguida, ao passar-se para 6 nós.

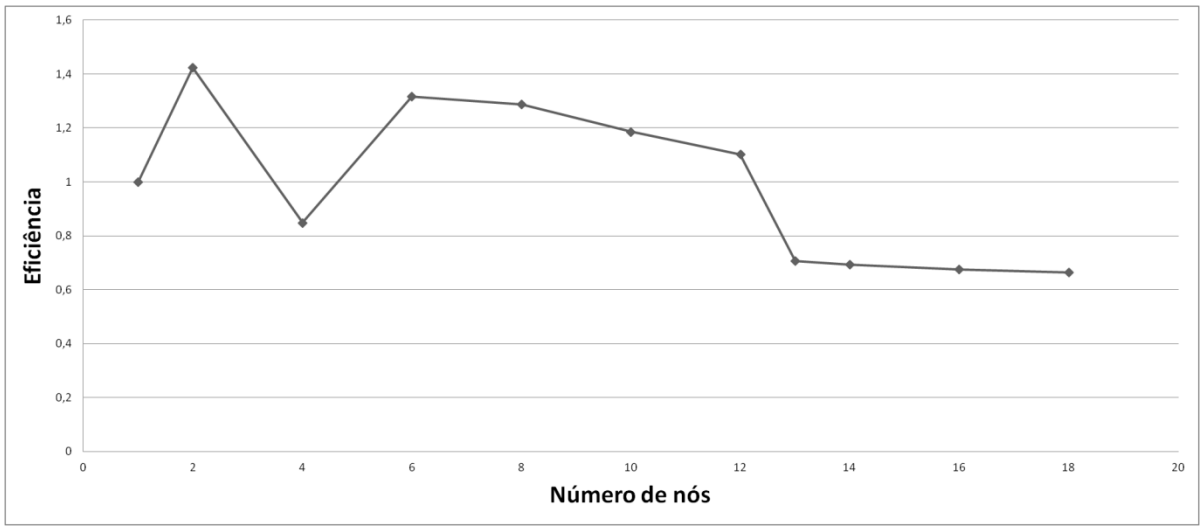


Figura 13 - Eficiência na execução do modelo no SeARCH variando o número de nós utilizados

Onde se poderia refletir a diminuição de eficiência seria na diferença entre o tempo total de execução e o tempo utilizado apenas para o processamento (tempo de CPU) (Figura 14). Mas os testes realizados com este modelo nunca resultaram em grandes diferenças entre esses tempos (o diferencial máximo percentual resultante foi de 0,1%). Isso quer dizer que o tempo necessário para atividades não relacionadas com o processamento é sempre muito pequeno, podendo-se considerar insignificante. O que significa que o tempo necessário para processos de *input/output* e tempos de espera voluntários ou não, como a espera devida ao congestionamento das ligações entre elementos do *cluster* nunca é problema.

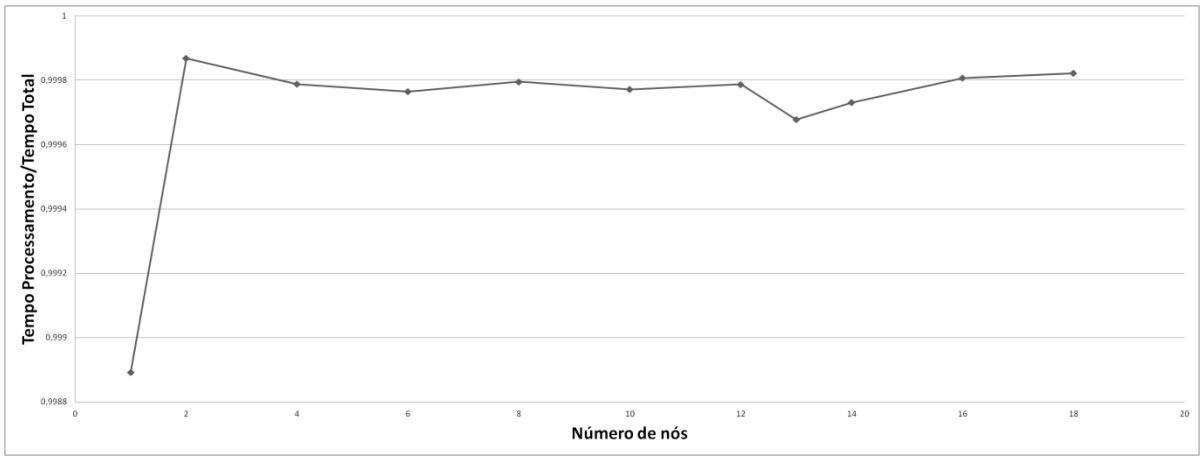


Figura 14 - Relação entre o tempo de processamento e o tempo total na execução do modelo no SeARCH variando o número de nós utilizados

Em algumas situações específicas houve uma diminuição da eficiência bastante abrupta que é mais difícil de explicar. É o caso da passagem de dois para quatro nós e a passagem de doze para treze nós.

O método para avaliar o desempenho do Delft3d (Figura 15) que é disponibilizado como *output* da execução do modelo teve os seus resultados também analisados. Sendo que o único elemento que varia nessa equação é o tempo de processamento, obtemos um gráfico com grande semelhança com o inverso de outros gráficos como da eficiência ou do tempo de execução. E tal como no gráfico da eficiência, constatou-se um pioramento considerável do desempenho na passagem dos 2 para os 4 nós computacionais.

O que este método nos permite observar é a qualidade do desempenho do *software* conforme se vão utilizando mais nós computacionais para a resolução do modelo.

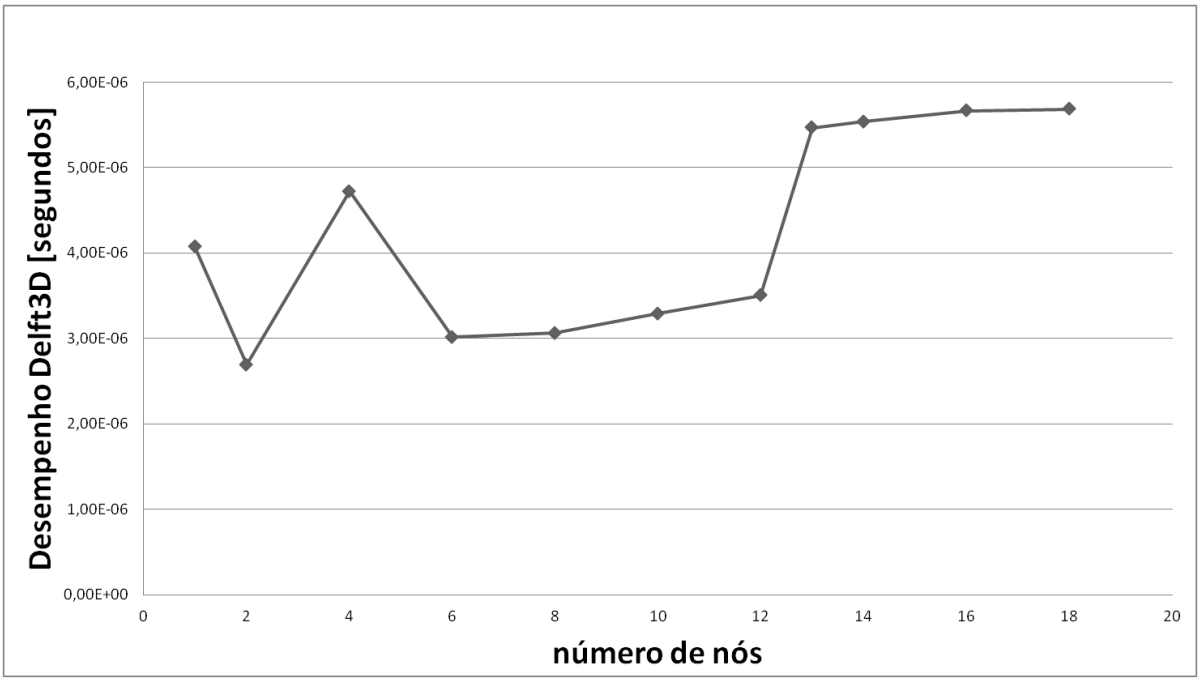


Figura 15 - Performance Delft3D na execução do modelo no SeARCH variando o número de nós utilizados

A análise realizada também permitiu avaliar a utilização dos recursos computacionais na resolução de cada uma das equações usadas na modelação (Figura 16). Uma análise mais exaustiva permitiu-nos ver que havia pouca variação na divisão do tempo de execução por cada uma das equações no modelo, ou seja, pouca influência das mudanças na arquitetura do *hardware* utilizado. A divisão de tempo de computação fazia-se entre as equações da continuidade (UZD) e de turbulência, ambas tendo sempre valores próximos de 30% da ocupação do tempo de execução (em média, 30,4 % e 29,5 % respetivamente sem grande variação). As sub-rotinas de resolução das equações de conservação da quantidade de movimento (SUD) apresentam valores sempre perto à volta dos 18%, sendo a média de 17,86%.

A ocupação do resto do tempo de computação (Outros) não é especificada no output do DELFT3D e, analisando todos os testes realizados, terá um valor médio de 22,19%, não havendo grande variação dessa percentagem entre as diferentes soluções testadas.

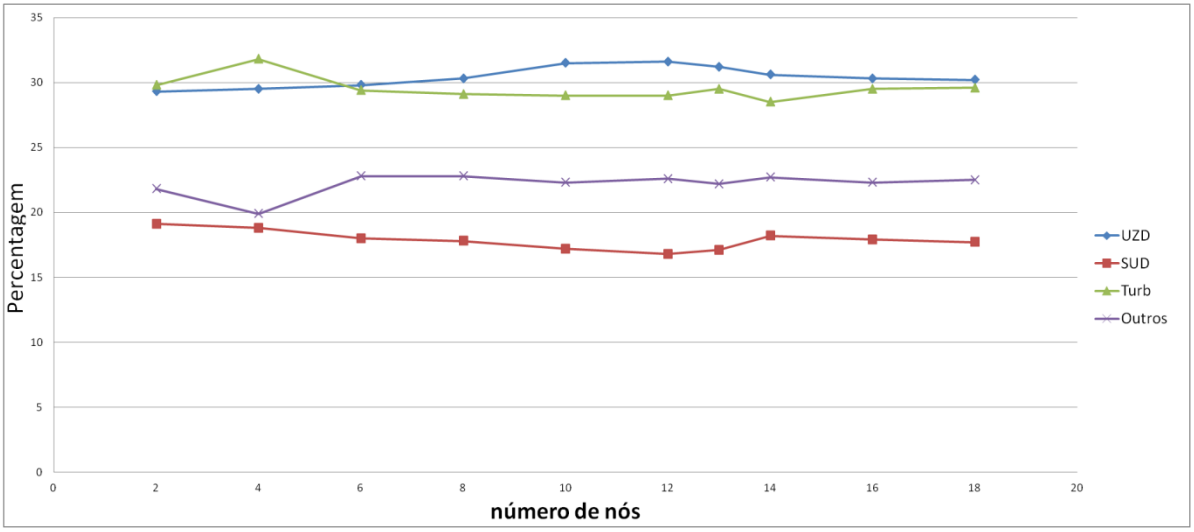


Figura 16 – Percentagem da utilização das equações no Delft3D

5.2. Desempenho do programa DELFT3D usando computação paralela num computador pessoal

Para a execução foi utilizado um computador pessoal com um processador Intel Core i7-2600, com 3,4 GHz de frequência, tendo 4 *cores* que permitem correr até 8 *threads* de execução independente.

De notar que só foi possível correr o modelo com até 6 *threads* e não até aos 8 como seria expectável, devido a problemas que não foram possíveis identificar. Ao dividirem-se as tarefas entre as diferentes *threads*, de modo a comportarem-se como processadores individualizados, obteve-se um aumento claro de desempenho comparativamente ao desempenho regular do computador (Figura 17).

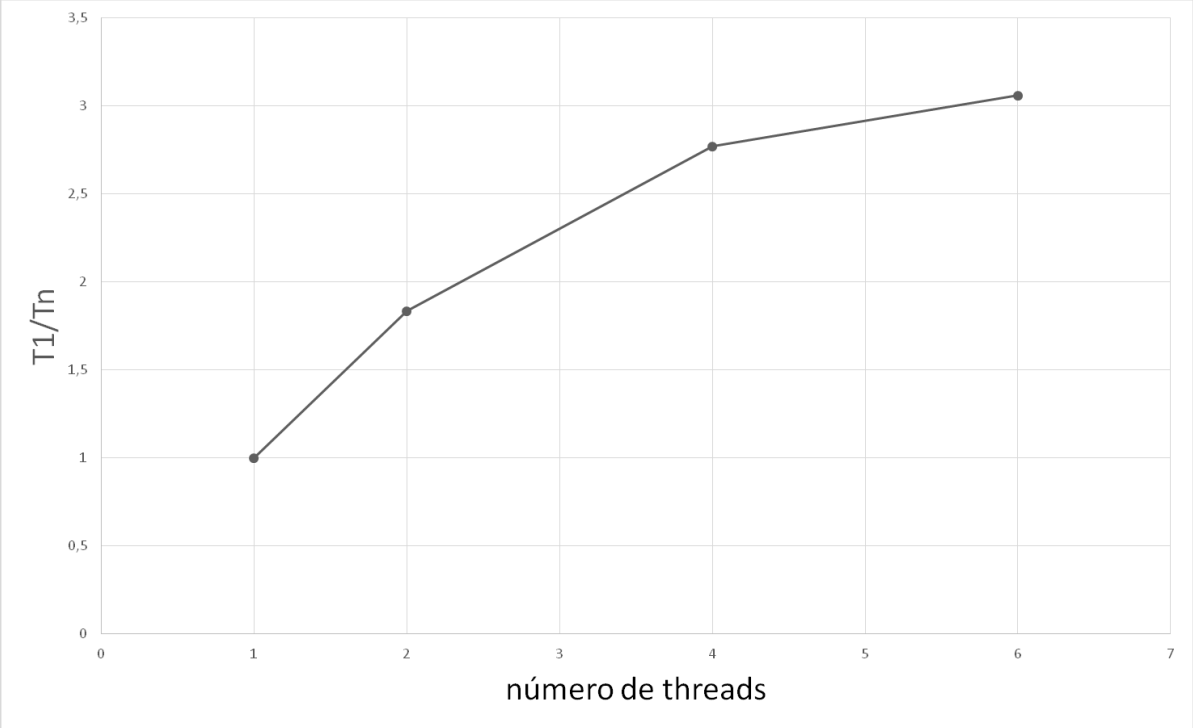


Figura 17 – Desempenho na execução do modelo num computador pessoal variando o número de nós utilizados

A passagem de um para dois *threads* diminuiu logo o tempo de execução significativamente, resultando numa diminuição de quase metade do tempo (Figura 18). A transição de dois para quatro *threads* também resulta em diminuição do tempo de execução/aumento da eficiência,

não sendo a diferença tão significativa como na transição anterior. Pode-se dizer o mesmo da transição de quatro para seis, parecendo haver a tendência para o tempo de execução se estabilizar à medida que se aumentam o número de *threads*.

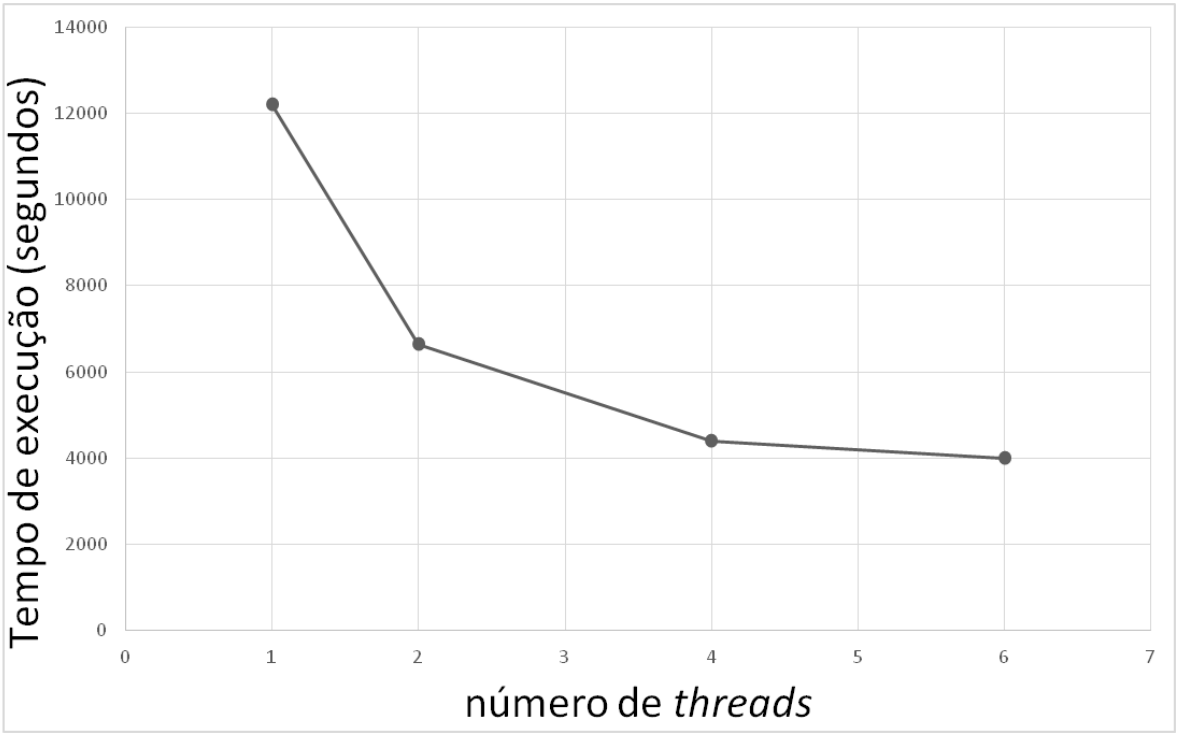


Figura 18 – Tempo de execução do modelo num computador pessoal variando o número de nós utilizados

A diferença entre o tempo total e o tempo de processamento vai aumentando conforme se vai aumentando o número de *threads*. O que quer dizer que o tempo necessário para processos de *input/output* e tempos de espera voluntários ou não, como a espera devida ao congestionamento do *bus* de ligação, torna-se cada vez maior, tornando o uso de computação paralela cada vez mais ineficaz. Com seis *threads* esse "tempo perdido" já contabiliza quase 15% do tempo de execução.

O desempenho de execução do DELFT3d tem um aumento significativo conforme se vai progressivamente aumentando o número de *threads* utilizadas para o processamento deste modelo.

Quanto aos outros dados obtidos, vemos que a eficiência tem tendência para diminuir conforme se vai aumentando o número de *threads* em que é dividido a computação (Figura 19). Os dados referentes ao desempenho no Delft3D permitem constatar o aumento do tempo de execução de cada parcela em que o modelo é dividido (Figura 20). Outra caraterística que tende a deteriorar-se ao se aumentarem as *threads* de execução é a relação entre o tempo de processamento e o tempo total de execução (Figura 21). Neste caso, quando se trata de computação sequencial, o tempo de processamento equivale praticamente ao tempo total da modelação efetuada. Quanto mais se aumentou o número de *threads*, maior se tornou o tempo necessário para outras atividades que não são de processamento. Estes dados não significam o aumento de tempo de resolução já que, como se trata de computação paralela, a execução está dividida entre vários *threads* e constatou-se que apesar de cada *thread* ser menos eficaz do que o processador em execução sequencial, o facto de ser usado paralelismo compensa largamente este aparente defeito.

Assim, temos uma diminuição significativa do tempo de execução ao utilizar o MPI e ao dividir a execução por seis *threads* de execução diminui-se o tempo de execução para menos de um terço do tempo de execução sem utilizar esse recurso (32,69%). É assim uma melhoria significativa na velocidade de execução deste *software* de modelação sem qualquer tipo de alteração no *hardware*.

Na figura 22 pode-se constatar a variação do recurso a cada uma das equações conforme o grau de paralelismo usado na resolução do modelo. A parcela de tempo intitulada “Outros” refere-se ao tempo que não é atribuído, no *output* do *software*, a qualquer uma das equações.

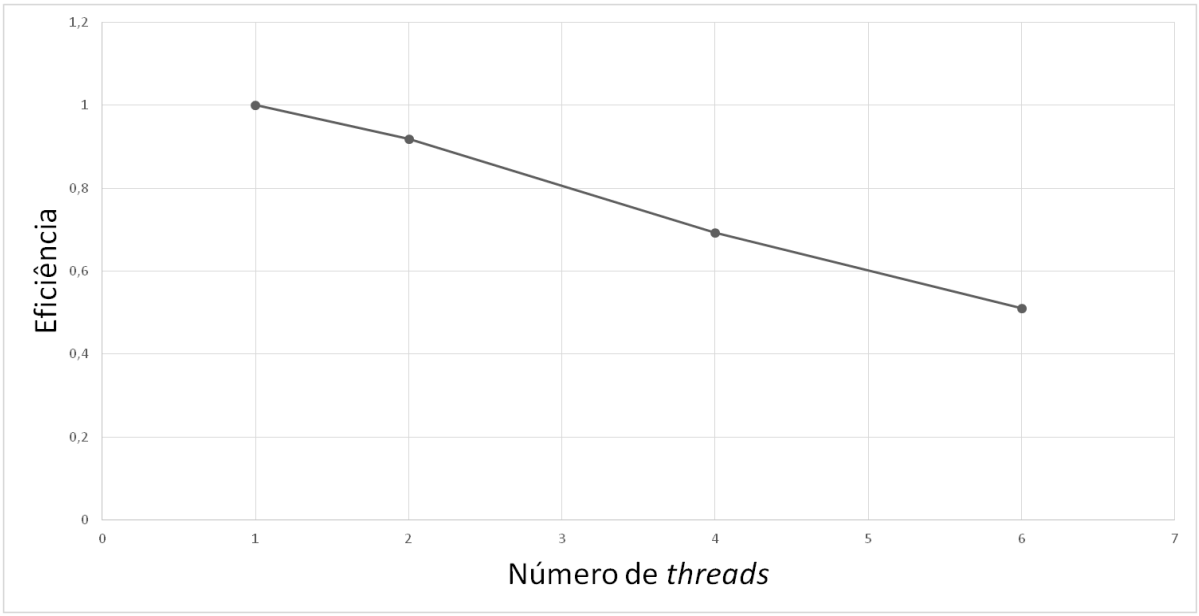


Figura 19 – Eficiência na execução do modelo num computador pessoal variando o número de nós utilizados

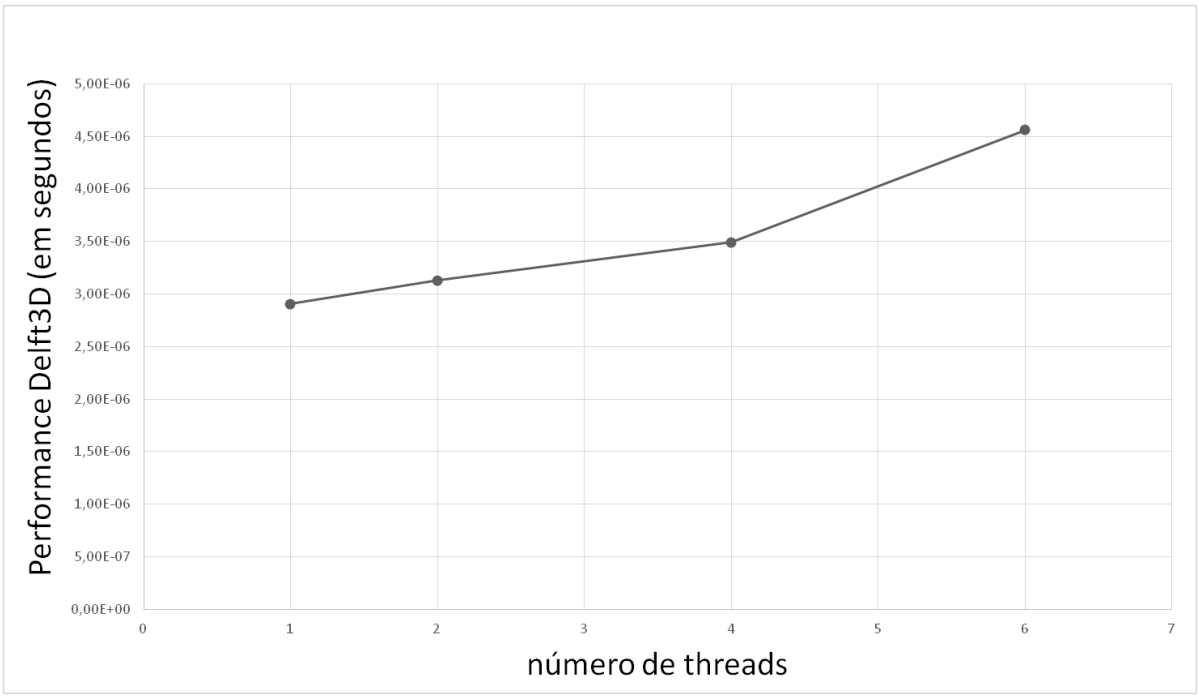


Figura 20 – Desempenho do Delft3D na execução do modelo num computador pessoal variando o número de nós utilizados

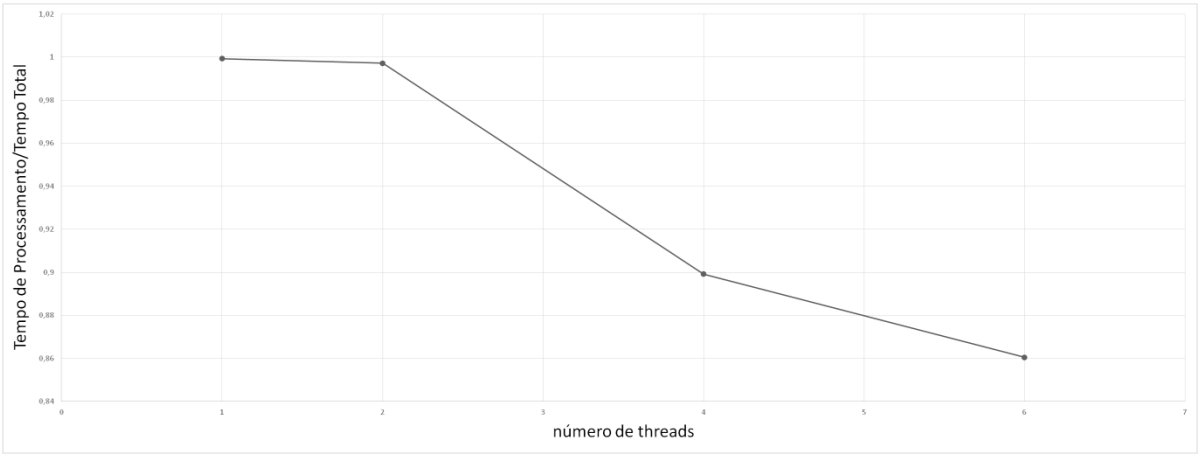


Figura 21 – Relação entre o tempo de processamento e o tempo total na execução do modelo num computador pessoal variando o número de nós utilizados

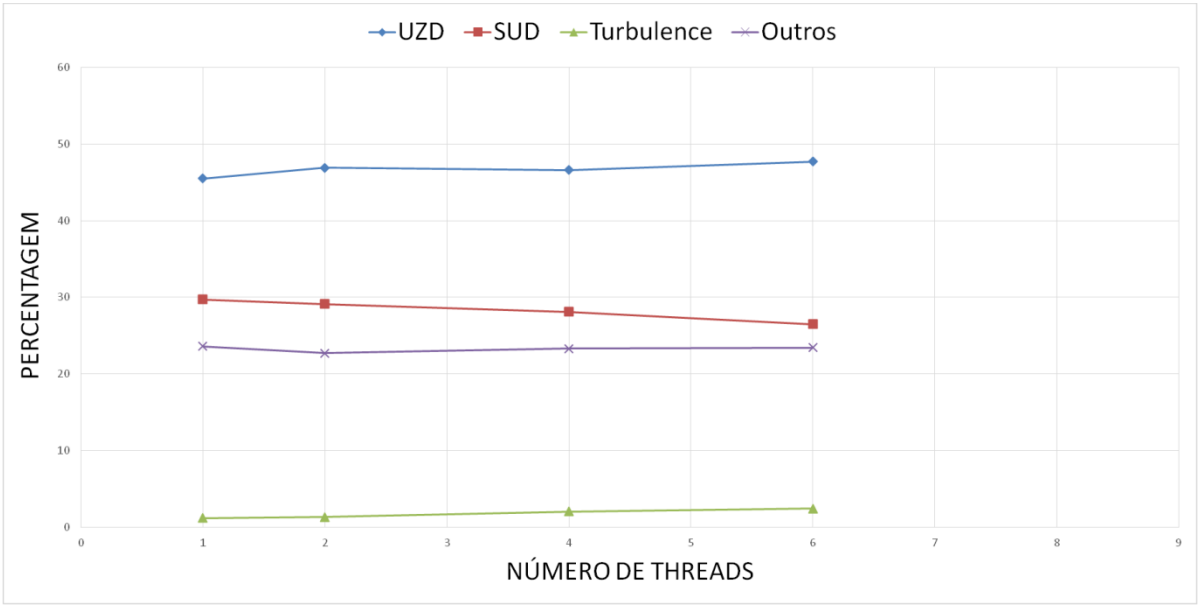


Figura 22 – Percentagem das equações utilizadas na execução do modelo num computador pessoal variando o número de nós utilizados

A execução deste modelo no Delft3D usando as potencialidades da arquitetura paralela foi também realizada utilizando outro computador pessoal. A unidade de processamento deste equipamento tratou-se de um *quadcore* (quatro *cores* de processamento) com frequência de 2,l6 GHz.

O tempo de execução do modelo Rio2, dispensando as possibilidades oferecidas pela computação paralela, é de 14129 segundos (aproximadamente quatro horas). O processador do computador testado permitiu a possibilidade de testar a execução paralela usando até quatro *threads*.

O tempo de execução com dois e quatro *threads* possibilitou a obtenção de resultados em, respetivamente, 70,7% e 59,8% do tempo de execução sequencial (Figura 23 e 24).

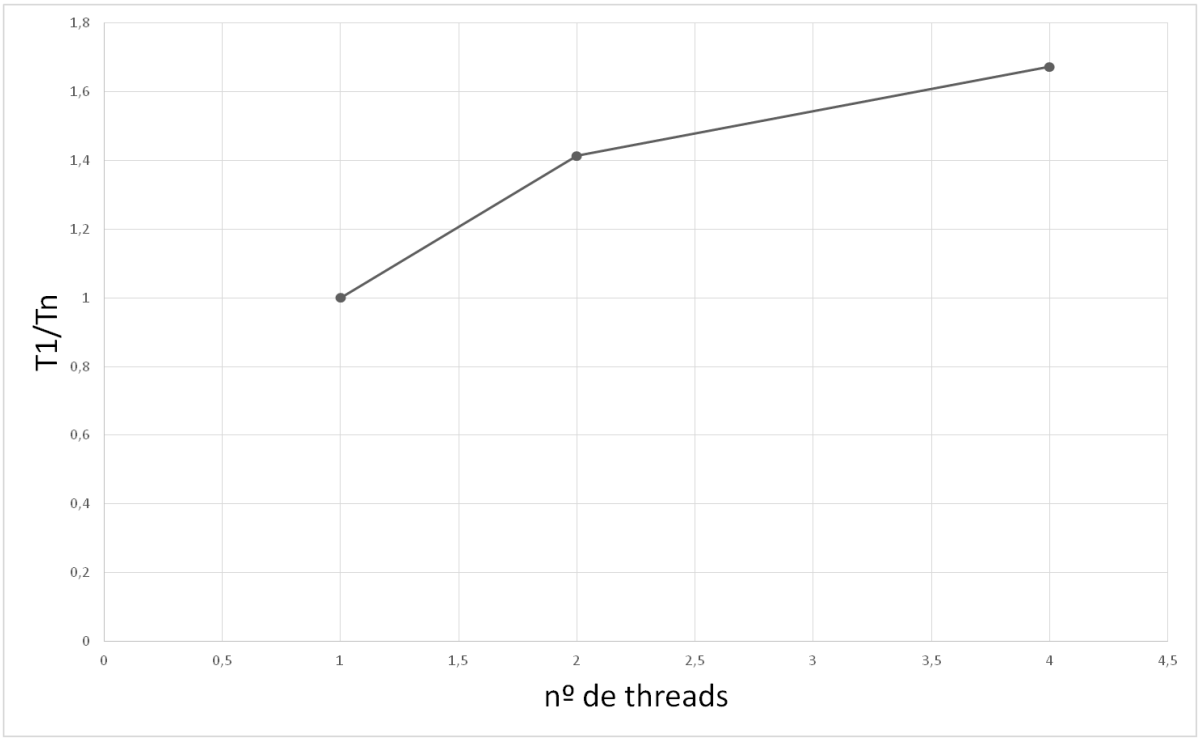


Figura 23 – Evolução do desempenho

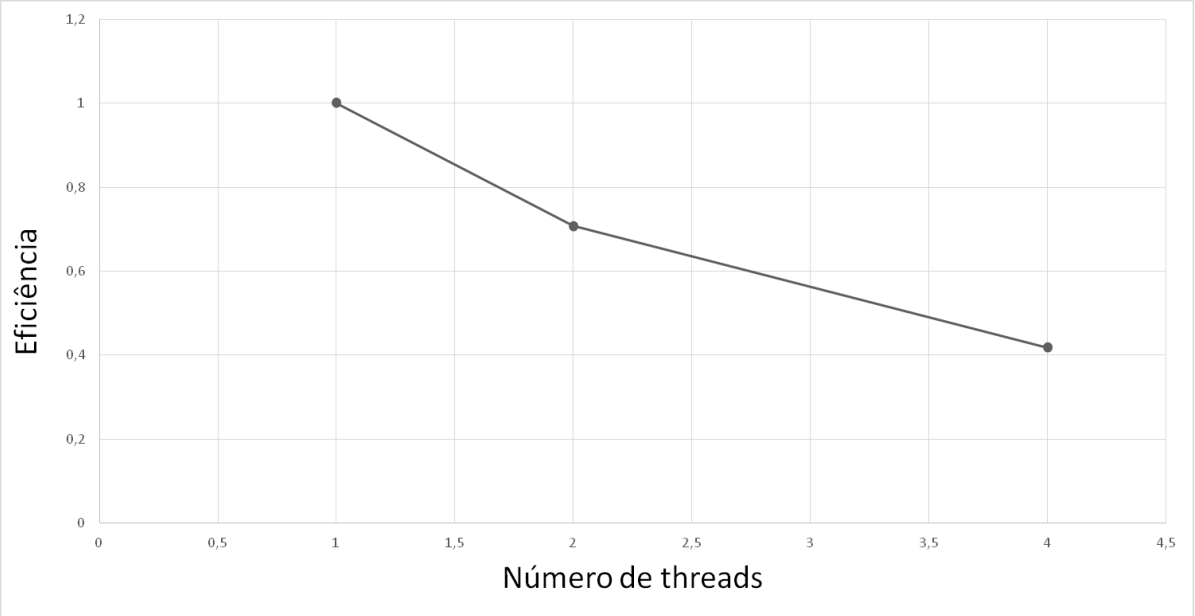


Figura 24 – Evolução da eficiência

A eficiência de cada *thread* individual reduz-se ao aumentar o paralelismo, mas isso não impede a diminuição significativa do tempo de execução (Figuras 25 a 27). Uma das razões possíveis para esta diminuição da eficiência encontra-se na relação entre o tempo total de execução e o tempo de processamento. Ao ir-se aumentando o número de *threads* a diferença entre esses tempos vai aumentando também, com quatro *threads* o tempo de processamento equivale apenas a aproximadamente 94% do tempo total. Assim o tempo para atividades que não sejam de processamento do modelo pretendido vai aumentando e prejudicando a eficiência na utilização do paralelismo.

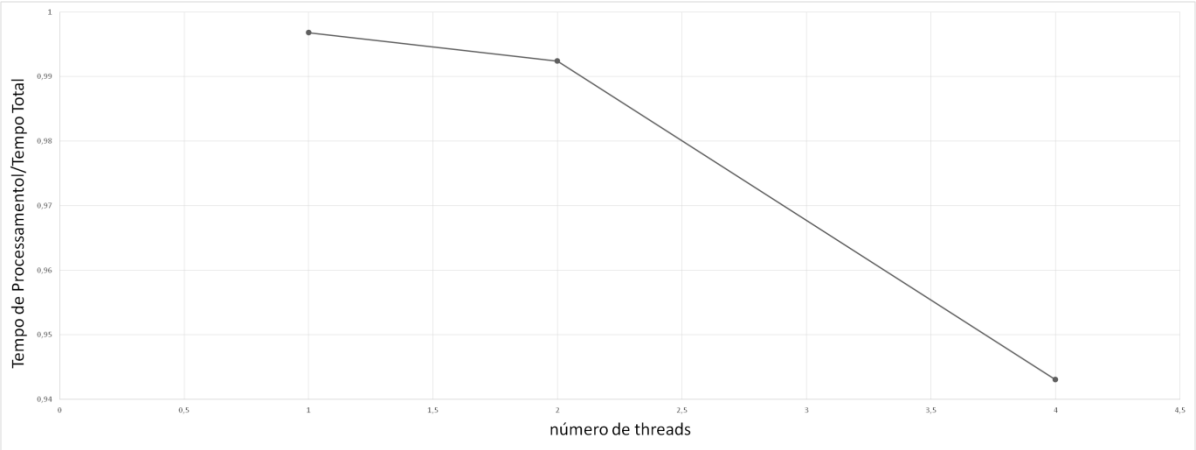


Figura 25 – Evolução da relação entre tempo de processamento e tempo total demorado

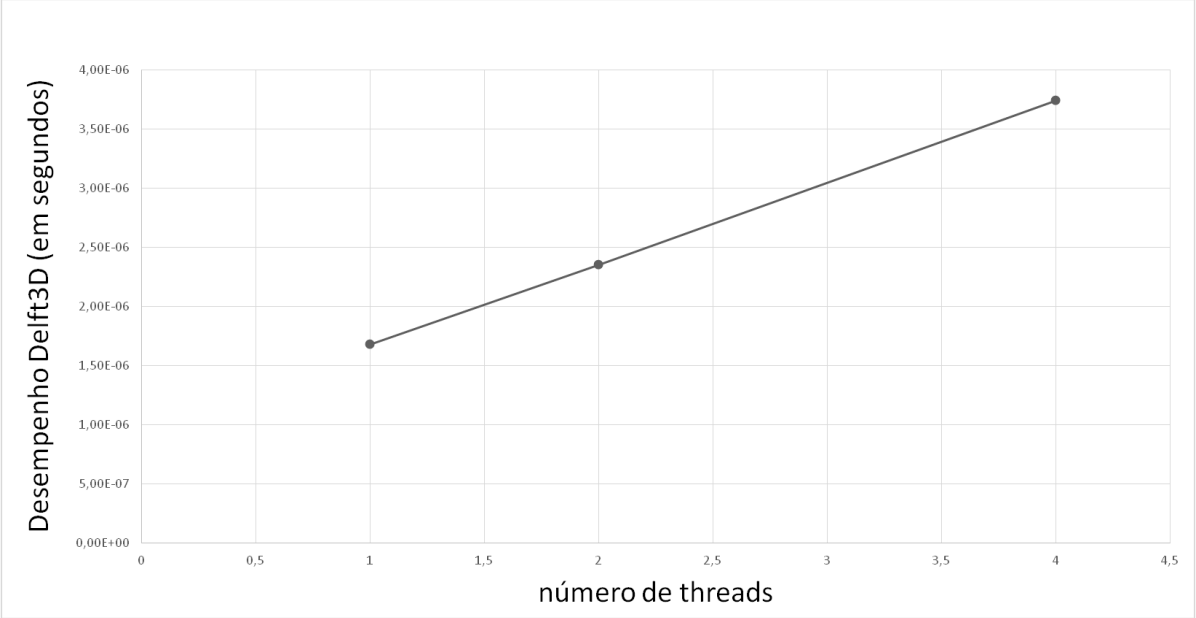


Figura 26 – Desempenho no Delft3D

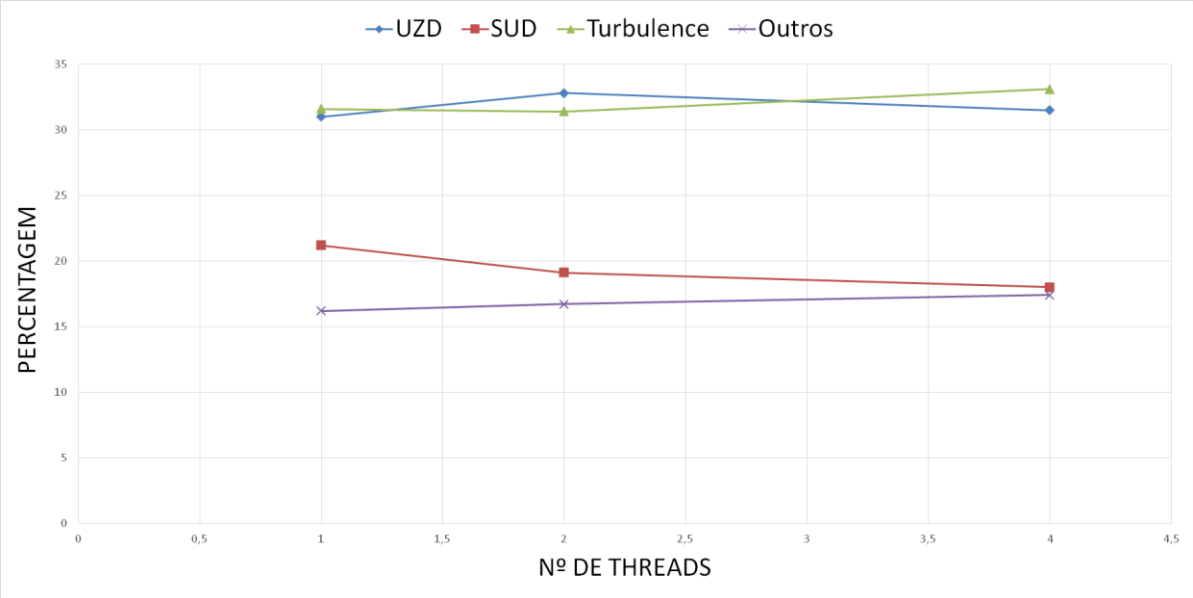


Figura 27 – Percentagem de utilização de cada equação

6. CONCLUSÕES

A eficiência da computação paralela aplicada ao módulo hidrodinâmico do programa Delft3D foi avaliada apontando para resultados viáveis para a sua utilização.

O modelo foi testado em dois computadores pessoais diferentes havendo, nos dois casos, uma diminuição substancial do tempo de execução dos modelos. Isto torna a utilização da programação paralela em computadores pessoais muito vantajosa já que potenciou as capacidades para a modelação hidráulica, tornando-as ferramentas que permitem obter resultados de modelação mais rapidamente e/ou resolver modelos mais complexos e exatos em tempo útil do que sem recorrer ao paralelismo.

Os resultados, em todos os casos testados, comprovaram que a computação paralela traz benefícios significativos no desempenho informático. O *cluster* SeARCH, ao se utilizarem vários nós computacionais deste *hardware* em vez de apenas um nó individual, permitiu recorrer a maiores recursos computacionais para um mesmo problema, o que levou à obtenção dos resultados da modelação numa pequena porção do tempo necessário no caso de recurso a computação sequencial (especificamente mais de 13 vezes mais rápido).

Mesmo nos casos de computadores pessoais, em que o *hardware* não é alterado, a computação paralela traz um benefício claro, havendo um melhor aproveitamento dos recursos existentes, o que traz uma diminuição muito evidente do tempo de resolução da hidrodinâmica em todos os casos testados. Os resultados obtidos com o primeiro PC testado permitiram constatar uma diminuição para 32,7% no tempo de execução em relação ao tempo demorado sem recurso à computação paralela. Quanto ao segundo PC de teste, o tempo de execução diminuiu para 69,8% do tempo inicialmente obtido.

Assim, comprovou-se que a computação paralela é uma ferramenta que traz benefícios claros na modelação hidráulica computacional, principalmente em problemas que envolvem uma

escala temporal de anos ou até mesmo de décadas, muito comuns na gestão e projetos de sistemas reais. Isto permite o uso mais pleno das potencialidades do *hardware*, seja através de *clusters* em que diferentes nós computacionais contribuem para a resolução de problemas de modelação, possibilitando a criação de modelos mais complexos e precisos, ou potenciando as capacidades paralelas de um simples computador pessoal, tornando-o uma ferramenta claramente mais rápida apenas graças à utilização de *software* para esse fim.

7. REFERÊNCIAS BIBLIOGRÁFICAS

Barbosa, Jorge Manuel Gomes. 2000. *Paralelismo em Processamento e Análise de Imagens Médicas*. Universidade do Porto.

Bos, K.J., J.A. Roelvink, & M.W. Dingemans. (1996). Modelling The Impact Of Detached Breakwaters On The Coast.25th International Conference on Coastal Engineering; Orlando, Florida, United States, September 2-6 19961.25 (1996): n. pag. Web. 10 Mar. 2015

Buyya, Rajkumar. 1999. *High Performance Cluster Computing: Architectures and Systems*. Prenctice Hall.

Chatzirodou, A., & Karunarathna, H. (2014). Impacts Of Tidal Energy Extraction On Sea BedMprphology. Coastal Engineering Proceedings, 1(34), sediment.33.

Deltares. 2014. *Delft3d-Flow - Simulation of multi-dimensional hydrodynamic flows and transport phenomena, including sediments. - User Manual - Hydro-Morphodynamics*. Delft.

Dowd, Kevin & Severance, Charles R. 1998 . *High Performance Computing*. O'Reilly.

Gelfenbaum, G., Vatvani, D., Jaffe, B., and Dekker, F. (2007) Tsunami Inundation and Sediment Transport in Vicinity of Coastal Mangrove Forest. Sixth International Symposium on Coastal Engineering and Science of Coastal Sediment Process, New Orleans, Louisiana, United States, May 13-17, 2007: pp. 1117-1128.

Giersch, Jürgen, Weidemann, Andreas & Anton, Gisela. 2003. *ROSI – An Object-Oriented and Parrallel-Computing Monte Carlo Simulation for X-Ray Imaging.* Volume 509, Issues 1‑3, 21 August 2003, Pages 151‑156.

Granja, Helena Maria, Bastos, L. ,Pinho, José L. S.,Gonçalves, J., Henriques, Renato F., Bio, A., Mendes, J., Magalhães, A. Métodos de monitorização e análise de risco de erosão costeira: dois casos de estudo portugueses. *Revista de Gestão Costeira Integrado,* **Volume 15, Número 1**: Páginas 47-63.

Hager, Georg e Wellein, Gerhard, 2011. *Introduction to High Performance for Scientist and Engineers*. Taylor & Francis.

Granja, Helena Maria *et al*. .Integração de metodologias no estabelecimento de um programa de monitorização costeira para avaliação de risco. VII Conferência Nacional de Cartografia e Geodesia: CNCG2011: Coordenadas para o Futuro Livro de Resumos, Faculdade de Ciências da Universidade do Porto, 5 e 6 de Maio de 2011

Jamal, M., Simmonds, D., e Magar, V. (2012). Gravel Beach Profile Evolution In Wave And Tidal Environments. Coastal Engineering Proceedings, 1(33), sediment.15.

Ji, Zhen-Gang. 2008. *Hydrodynamics and water quality : modeling rives, lakes, and estuaries*. Wiley-Interscience.

Kuk, Won Ko, Sim, Jae Hwan e Kim, Min Young. 2012. A High-speed Whitelight Scanning Interferometer using On-The-Fly Imaging and Parallel Processing. *Simpósio Internacional de Tecnologias de Optometria, 2012*. Pag 1-4

Li, Yadong. 2001. *Computer Simulation of Linear and Harmonic Ultrasound Imaging.* University of Wisconsin-Madison.

Mauler, Gary e Beebe, Milton. 2002. *Clustering Windows Server – A Road Map for Enterprise Solutions*. Butterworth–Heinemann.

Pereira, Carla Alexandra Santos. 2010. *Risco de Erosão para Diferentes Cenários de Evolução do Litoral de Aveiro*. Universidade de Aveiro

Pinho, José L. S. . 2000. *Aplicação de modelação matemática ao estudo da hidrodinâmica e da qualidade da água em zonas costeiras*. Universidade do Minho.

Pinho, José L. S. 2014. Monitorização e modelação da morfodinâmica costeira. Ingenium**141**: 38-39.

Pinho, José L. S. e Vieira, José M. P. 2005. *Mathematical modelling of salt water intrusion in a Northern Portuguese estuary*. Universidade do Minho.

Pinho, José L. S.. 2000. *Aplicação de Modelação Matemática ao Estudo da Hidrodinâmica e da Qualidade da Água em Zonas Costeiras*. Universidade do Minho.

Pinho da Cruz, Joaquim Alexandre Mendes. 2007. *Caracterização Termomecânica de Materiais Multifásicos Utilizando Procedimentos de Homogeneização*. Universidade de Aveiro.

Pinto, Roberto José. 2011. *Aplicação de Processamento Paralelo ao Problema de Planejamento da Operação de Sistemas Hidrotérmicos Baseado em Clusters de Computador*. Universidade Federal do Rio de Janeiro.

Quinn, Michael J. (2003). *Parralel Programming In C With MPI and Open MP*. 1º Edição. McGraw-Hill. New York

Rauber, Thomas e Rünger, Gudula. 2012. *Parallel Programing for Multicore and Cluster Systems – Second Edition*. Springer.

Reis, Eduarda Maria Oliveira. 2010. *Evolução da linha de costa e defesa das zonas costeiras: análise custo/benefício*. Universidade de Aveiro

Da Rocha, Mariana Vieira Lima Matias. 2011. *Modelação Numérica do Impacto dos Esporões na Hidrodinâmica Costeira*. Universidade de Aveiro.

Sedigh, M., Tomlinson, R., Etemad-Shahidi, A., e Cartwright, N. (2014). Modelling The Morphological Response Of An Ebb Tidal Delta To Storm Wave Forcing. *Coastal Engineering Proceedings*, *1*(34)

Villani, M., Bosboom, J., Zijlema, M. e Stive, M. (2012). Circulation Patterns And Shoreline Response Induced By Submerged Breakwaters. Coastal Engineering Proceedings, 1(33), structures.25.

Wang, Hao, Fu, Xudong, Wang, Guangqian, Li, Tiejian e Gao, Jie. 2011. A Common Parallel Computing Framework for Modeling Hydrological Processes of River Basins. *State Key Laboratory of Hydroscience and Engineering, Tsinghua University.*

Wang, L., Zimmermann, N., Trouw, K., De Maerschalck, B., e Vanlede, J. (2014). Numerical Modelling Of Long-Term MorphologyIn The Surf Zone Of The Belgian Coast. Coastal Engineering Proceedings, 1(34), sediment.42.

Yuan, F., e Cox, R. (2014). Modelling Beach Morphodynamics For Southern Gold Coast Beach Nourishment Project At Bilinga Beach. Coastal Engineering Proceedings, 1(34), sediment.34.

Zhang, Yan Jun, Jha, Manoj, Gu, Roy, Wensheng, Luo e Alin, Lei. 2010. A DEM-based Parallel Computing Hydrodynamic and Transport Model. *River Research and Applications, volume 25, Número 5*, pag 647-658.

8. LISTA DE SITES CONSULTADOS

http://search.di.uminho.pt/