



Universidade do Minho
Escola de Engenharia

Francisca Leitão Gonçalves do Vale Lima

**Big Data Warehousing em Tempo Real:
Da Recolha ao Processamento de Dados**

**Big Data Warehousing em Tempo Real:
Da Recolha ao Processamento de Dados**

Francisca Leitão Gonçalves do Vale Lima

UMinho | 2017

outubro de 2017



Universidade do Minho
Escola de Engenharia

Francisca Leitão Gonçalves do Vale Lima

**Big Data Warehousing em Tempo Real:
Da Recolha ao Processamento de Dados**

Dissertação de Mestrado
Mestrado Integrado em Engenharia e Gestão de Sistemas de
Informação

Trabalho efetuado sob a orientação da
Professora Doutora Maribel Yasmina Santos

DECLARAÇÃO

Nome: Francisca Leitão Gonçalves do Vale Lima

Endereço eletrónico: a69163@alunos.uminho.pt

Telefone: 910535881

Número do Bilhete de Identidade: 14607140

Título da dissertação: *Big Data Warehousing* em Tempo Real: Da Recolha ao Processamento de Dados

Orientador: Professora Doutora Maribel Yasmina Santos

Ano de conclusão: 2017

Designação do Mestrado: Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação

É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA DISSERTAÇÃO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

Universidade do Minho, 31 / 10 / 2017

Assinatura: Francisca Vale Lima

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project n° 002814; Funding Reference: POCI-01-0247-FEDER-002814].

AGRADECIMENTOS

Entrei na universidade, nesta jornada pelo conhecimento para saber cada vez mais, para descobrir Sistemas de Informação, na esperança de ser aquilo que quero para o meu futuro. Os anos passaram rápido e quando dou conta estou no último ano a realizar a minha dissertação, mas contente com aquilo que aprendi, com aquilo que sei, pelos momentos que passei e pelas pessoas que conheci nesta jornada. Agora é o momento de agradecer a todas as partes importantes que contribuíram não só para este percurso, mas também para a realização desta dissertação.

Gostaria de agradecer à professora Maribel Yasmina Santos, que desde o primeiro momento se mostrou disponível e durante este percurso se revelou fundamental. Pela profissional exemplar que é, pela confiança que depositou em mim, pela paciência, pela ajuda, por todas as sugestões e pelos sorrisos que sempre acalmam os problemas que vão surgindo. Agradeço também a oportunidade de fazer parte do projeto de investigação, experiência que não estava à espera, mas que me ajudou a crescer e a descobrir cada vez mais sobre esta área.

Ao Carlos, que me fez aprender tanto sobre assuntos que olho para trás 1 ano e tão desconhecia. Por toda a paciência para explicar mais uma vez, por toda a disponibilidade, por te teres revelado a minha salvação em muitos momentos, sempre pronto a ajudar e com uma solução sempre à vista.

Aos meus coleguinhas do LID4, sempre cheios de energia e confiantes nas minhas capacidades. Por toda a partilha de conhecimentos e por todas as ajudas. Conjugiar a dissertação com o projeto de investigação não foi fácil, mas vocês estavam sempre prontos a ajudar.

Às minhas amigas e amigos mais chegados, aos que estão comigo nesta jornada e me acompanham quase desde o primeiro dia que entrei para a universidade. Pelas noites no buraco, pelos risos incontroláveis, pelos jantares e almoços todos juntos, sem dúvida muitos bons momentos e as dificuldades que surgiram foram ultrapassadas juntos. Porque sei que são pessoas que levo para a vida.

Aos meus pais e irmã, ao apoio incondicional que me deram, por toda a confiança para continuar e paciência que têm tido comigo. Por toda a motivação, a energia, tudo o que me proporcionaram e por acreditarem sempre em mim! Por estarem presentes e acreditarem em todas as minhas decisões!

Por fim agradeço a todas as pessoas que fizeram parte da minha jornada na Universidade do Minho, no Departamento de Sistemas de Informação, a todos aqueles que fizeram parte deste percurso. Bons ensinamentos me transmitiram, que levo nesta viagem!

Muito obrigada! ... e venha o próximo desafio!

RESUMO

A evolução tecnológica dos últimos anos tem captado o interesse das organizações na análise de dados, na extração de informação das grandes quantidades de dados geradas, surgindo o interesse em *Business Intelligence* e associado a este a componente de *Big Data*. *Big Data* veio assim revolucionar as tecnologias e técnicas tradicionais pela capacidade de lidar com o volume, velocidade e variedade dos dados. A utilização de *Data Warehouses*, em contexto de *Big Data*, os *Big Data Warehouses*, aumentam as perspetivas de obter os dados de forma rápida e atualizada, potenciando o acesso aos dados em tempo real. Assim, com a introdução de tecnologias em tempo real é possível capturar a mudança nos dados e obter uma análise com dados atualizados, cada vez mais importante na tomada de decisão. Nesta dissertação pretende-se compreender o papel dos diversos componentes e tecnologias na concretização de *Big Data Warehouses*, um conjunto de considerações para a implementação de *Big Data Warehouses*, pelo que são explorados os requisitos de tempo real para a concretização e avaliação de uma arquitetura para o processamento de dados. Vários trabalhos têm sido propostos na procura de soluções que permitam o fluxo de dados em tempo real, contudo esta dissertação distingue-se por permitir que a recolha, o processamento, o armazenamento e a análise de dados, tanto recolhidos em tempo real como armazenados numa componente histórica, possam ser feitos em poucos segundos recorrendo a um conjunto de tecnologias aqui testadas e validadas. Assim, neste trabalho é estudado o desempenho dos componentes que permitem a concretização do tempo real desejado, com dados em tempo real e históricos, os quais são concretizados num caso de demonstração que permite evidenciar as vantagens e desvantagens de cada tecnologia. Com dados do Twitter a fluir em tempo real, o comportamento das várias tecnologias em cenários distintos é avaliado de forma a estabelecer um conjunto de boas práticas que vão desde a recolha de dados com Kafka, o processamento de dados com Spark Streaming, ao armazenamento de dados com Hive e/ou Cassandra, sendo efetuadas consultas recorrendo ao Presto. O trabalho realizado permite analisar o comportamento do Kafka neste contexto, o desempenho do Spark Streaming com diferentes durações de pacotes de dados, o desempenho do Hive no armazenamento de dados em tempo real e no armazenamento de dados históricos, e do Cassandra no armazenamento de dados em tempo real. A análise dos resultados obtidos permite a uma organização compreender o papel de cada componente e tecnologia numa arquitetura para a concretização de *Big Data Warehouses*, sendo possível realizar consultas de dados que integram dados atuais, a fluir em tempo real, com dados históricos.

Palavras-Chave: *Big Data*, *Big Data Warehouse*, *Real-Time*, Hadoop, Hive

ABSTRACT

The technological evolution of the last years has called the attention of organizations for the analysis of data, with the aim of extracting information from the large volumes of generated data, increasing the interest in Business Intelligence and, more specifically, in Big Data. Big Data has revolutionized the traditional technologies and techniques with the ability to handle the volume, velocity and variety of data. The use of Data Warehouses, in the context of Big Data, the Big Data Warehouses, increases the ability to get faster access to updated data, enhancing data analytics in real-time. Thus, the introduction of real-time technologies allows capturing changes in data and provides analyses with updated data, a factor that is increasingly important in decision making. In this dissertation, the role of the components and technologies is explored and a set of considerations is established for the implementation of Big Data Warehouses, thus exploring the real-time requirements for the concretization and evaluation of an architecture for data processing. Related works have been proposed mainly enhancing data flowing in real-time. However, this dissertation is innovative by allowing the collection, processing, storage and analysis of data, a workflow that can be done in only a few seconds using a set of technologies tested and validated in this work. Thus, in this work the performance of the components that allow the concretization of the aimed real-time, processing real-time and historical data, are studied. A demonstration case shows the advantages and disadvantages of each technology. With Twitter data flowing in real-time, the technologies performance in distinct scenarios is evaluated establishing a set of best practices that use data collection with Kafka, data processing with Spark Streaming, data storage with Hive and/or Cassandra, being data queried by Presto. The work carried out allows the analysis of the performance of Kafka in this context, the performance of Spark Streaming with different micro-batches, the performance of Hive in real-time data storage and in historical data storage, and Cassandra in real-time data storage. The analysis of the results allows an organization to understand the role of each component and technology in an architecture for the implementation of Big Data Warehouses, being possible to perform data analytics that integrates current data, flowing in real-time, with historical data.

Keywords: Big Data, Big Data Warehouse, Real-Time, Hadoop, Hive

ÍNDICE

Agradecimentos.....	v
Resumo.....	vii
Abstract.....	ix
Índice.....	xi
Índice de Figuras.....	xiii
Índice de Gráficos.....	xv
Índice de Tabelas.....	xvii
Siglas e Acrónimos.....	xix
1. Introdução.....	1
1.1. Enquadramento e Motivação.....	1
1.2. Objetivos e Resultados Esperados.....	2
1.3. Abordagem de Investigação.....	2
1.3.1. Metodologia de Investigação.....	3
1.3.2. Processo de Revisão Bibliográfica.....	5
1.4. Contributos do Trabalho.....	6
1.5. Organização do Documento.....	6
2. <i>Big Data Warehouses</i>	9
2.1. Principais Conceitos.....	9
2.1.1. <i>Big Data</i>	9
2.1.2. <i>Data Warehouse</i>	19
2.1.3. <i>Big Data Warehouse</i>	21
2.2. Armazenamento de Dados.....	23
2.2.1. Bases de Dados.....	24
2.2.2. Modelos de Dados.....	26
2.3. Tempo Real em <i>Big Data Warehouses</i>	27
2.3.1. Características de Tempo Real.....	27
2.3.2. Trabalhos Relacionados.....	29
3. Tecnologias para a Concretização de <i>Big Data Warehouses</i>	35
3.1. Ecossistema do Hadoop.....	35

3.2. Repositório de Dados	37
3.3. Processamento de Dados em Tempo Real	40
4. <i>Big Data Warehouses</i> em Tempo Real	45
4.1. Fluxos de Dados	45
4.2. Arquitetura para o Processamento de Dados	47
4.3. Concretização da Arquitetura	50
4.3.1. Infraestrutura Utilizada	51
4.3.2. Componente Experimental.....	52
5. Caso de Demonstração	55
5.1. Recolha de Dados.....	55
5.2. Processamento de Dados	59
5.3. Armazenamento de Dados.....	61
5.3.1. Hive no Armazenamento de Dados em Tempo Real	64
5.3.2. Cassandra no Armazenamento de Dados em Tempo Real.....	71
5.3.3. Hive <i>versus</i> Cassandra no Armazenamento de Dados em Tempo Real	76
5.3.4. Hive no Armazenamento de Dados Históricos	89
5.4. Síntese de Resultados.....	94
6. Conclusão	99
6.1. Resultados Obtidos.....	100
6.2. Trabalho Futuro.....	101
Referências Bibliográficas	103

ÍNDICE DE FIGURAS

Figura 1 - <i>Design Science Research Methodology for Information Systems</i> . Adaptada de Peffers et al. (2007)	3
Figura 2 - Modelo dos 3 V's. Adaptada de P. Zikopoulos et al. (2011)	12
Figura 3 - Características entre os 3 V's. Adaptada de Krishnan (2013)	14
Figura 4 - <i>V's Big Data</i>	15
Figura 5 - Processo ELT. Adaptada de Krishnan (2013).....	19
Figura 6 - Arquitetura de DW. Adaptada de Kimball & Ross (2013)	20
Figura 7 - Framework RTDW. Adaptada de Li & Mao (2015)	31
Figura 8 - <i>Lambda Architecture</i> . Adaptada de X. Liu et al. (2014)	32
Figura 9 - Arquitetura do Facebook de tempo real. Adaptada de G. J. Chen et al. (2016)	33
Figura 10 - Arquitetura do HDFS. Adaptado de Krishnan (2013)	36
Figura 11 - Tecnologias de possível integração na arquitetura.....	41
Figura 12 - Fluxo de dados em <i>streaming</i> e em <i>batch</i>	46
Figura 13 - Arquitetura de processamento de dados. Adaptada de M. Y. Santos et al (2017)	48
Figura 14 - Cenários de tecnologias para testes	51
Figura 15 - Recolha de dados do Twitter	52
Figura 16 - Processamento e Armazenamento de dados em tempo real	53
Figura 17 - Recolha, Processamento e Armazenamento de dados em <i>batch</i>	53
Figura 18 - Recolha, Processamento e Armazenamento de dados em <i>batch</i> no Talend.....	54
Figura 19 - Processo do envio de mensagens através do Kafka para o Spark Streaming.....	57
Figura 20 - Processo do envio de mensagens através do Kafka para o Spark Streaming com mecanismo do Twitter	58
Figura 21 - Processamento de dados na aplicação Spark Streaming.....	60
Figura 22 - Fila de espera na aplicação Spark Streaming.....	60
Figura 23 - Consulta de dados na aplicação Spark Streaming.....	61
Figura 24 - Sistema de ficheiros da tabela particionada	64
Figura 25 - Ficheiros de pequena dimensão no sistema de ficheiros	66
Figura 26 - Dificuldade na inserção de dados na tabela não particionada do Hive	67
Figura 27 - Atraso no Spark Streaming na inserção de dados no Cassandra	76

Figura 28 - Espera no Spark Streaming.....	76
Figura 29 - Atraso no Spark Streaming na inserção de dados no Cassandra no <i>micro-batch</i> de 20 segundos	83
Figura 30 - Tamanho dos ficheiros de dois volumes de dados diferentes no Hive	84
Figura 31 - Atraso no Spark Streaming na inserção de dados no Cassandra	86
Figura 32 - Tempo de processamento no Spark Streaming com volume de dados V3 na tecnologia Hive	87
Figura 33 - Comportamento da tecnologia Cassandra no processamento no Spark Streaming com volume de dados V3	88
Figura 34 - Diferença da dimensão de ficheiros nas áreas de armazenamento de dados.....	90

ÍNDICE DE GRÁFICOS

Gráfico 1 - Média de mensagens consumidas pelo Kafka.....	57
Gráfico 2 - Média de mensagens consumidas pelo Kafka com mecanismo Twitter	58
Gráfico 3 - Resultados da Consulta 1 (C1) para a tabela particionada (CP) e não particionada (SP)	65
Gráfico 4 - Resultados da Consulta 2 (C2) para a tabela particionada (CP) e não particionada (SP)	68
Gráfico 5 - Resultados da Consulta 3 (C3) para a tabela particionada (CP) e não particionada (SP)	68
Gráfico 6 - Resultados da Consulta 1 (C1) para a tabela particionada (CP) e não particionada (SP) com formato ORC e TXT.....	70
Gráfico 7 - Resultados da Consulta 2 (C2) para a tabela particionada (CP) e não particionada (SP) com formato ORC e TXT.....	71
Gráfico 8 - Resultados da Consulta 1 (C1) para a tabela com chave de <i>clustering</i> (CC) e sem chave de <i>clustering</i> (SC).....	72
Gráfico 9 - Resultados da Consulta 2 (C2) para a tabela com chave de <i>clustering</i> (CC) e sem chave de <i>clustering</i> (SC).....	73
Gráfico 10 - Resultados da Consulta 3 (C3) para a tabela com chave de <i>clustering</i> (CC) e sem chave de <i>clustering</i> (SC).....	73
Gráfico 11 - Resultados da Consulta 1 (C1) para tabela Hive e Cassandra	77
Gráfico 12 - Resultados da Consulta 2 (C2) para tabela Hive e Cassandra	78
Gráfico 13 - Resultados da Consulta 3 (C3) para tabela Hive e Cassandra	78
Gráfico 14 - Resultados da Consulta 4 (C4) para tabela Hive e Cassandra	79
Gráfico 15 - Resultados Hive com <i>micro-batch</i> de 20 segundos	80
Gráfico 16 - Resultados Hive com <i>micro-batch</i> de 5 segundos	81
Gráfico 17 - Resultados Cassandra com <i>micro-batch</i> 20 segundos	82
Gráfico 18 - Resultados Cassandra com <i>micro-batch</i> de 5 segundos.....	82
Gráfico 19 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V2) no Hive	84
Gráfico 20 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V2) no Cassandra.....	85
Gráfico 21 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V3) no Hive	86

Gráfico 22 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V3) no Cassandra	88
Gráfico 23 - Resultados da consulta C1, C2, C3, C4, C9, C10, C11, C12 e C13 na tabela Hive do histórico	91
Gráfico 24 - Resultados das consultas C14, C15 e C16 com diferentes volumes de dados no armazenamento de dados em tempo real e histórico no Hive	93

ÍNDICE DE TABELAS

Tabela 1 - Consultas aos dados	62
Tabela 2 - Resultados das consultas 4, 5, 6, 7 e 8 na tabela particionada (CP) e tabela sem partições (SP) no Hive	69
Tabela 3 - Resultados das consultas 4, 5, 6, 7 e 8 na tabela com chave de <i>clustering</i> (CC) e tabela sem chave de <i>clustering</i> (SC) no Cassandra	74
Tabela 4 - Resultados da consulta 1 (C1), 2 (C2), 3 (C3) e 4 (C4) no Hive e Cassandra	91
Tabela 5 - Síntese de resultados obtidos	95

SIGLAS E ACRÓNIMOS

Este documento utiliza um conjunto de siglas e acrónimos apresentados de seguida:

ACID - *Atomicity, Consistency, Isolation, Durability*

BASE - *Basically Available, Soft state, Eventually consistent*

BDW - *Big Data Warehouse*

BI - *Business Intelligence*

CAP - *Consistency, Availability, Partition tolerance*

CC - Tabela com chave de *clustering*

CP - Tabela particionada

CRM - *Customer Relationship Management*

DSRM - *Design Science Research Methodology*

DW - *Data Warehouse*

ELT - Extração, Carregamento e Transformação

ERP - *Enterprise Resource Planning*

ETL - Extração, Transformação e Carregamento

GFS - *Google File System*

HDFS - *Hadoop Distributed File System*

HiveQL- *Hive Query Language*

NoSQL - *Not only SQL*

OLAP - *Online Analytical Processing*

OLTP - *Online Transaction Processing*

RTDW - *Real-Time Data Warehouse*

SC - Tabela sem chave de *clustering*

SP - Tabela não particionada

SQL - *Strutured Query Language*

TI - Tecnologias de Informação

1. INTRODUÇÃO

Neste capítulo apresentam-se o enquadramento e a motivação para o tema desta dissertação, os objetivos e resultados esperados no decorrer do seu desenvolvimento e a abordagem de investigação adotada. Encontram-se também os contributos do desenvolvimento deste trabalho e, por fim, a organização deste documento.

1.1. Enquadramento e Motivação

Grandes quantidades de dados são recolhidas todos os dias, associadas a novas fontes de dados, novos mecanismos de recolha e novas aplicações, pelo que as organizações começam a ter necessidade de extrair informação dos dados recolhidos. A análise dos dados permite conhecer, por exemplo, não só a opinião do cliente, como as necessidades e as oportunidades de negócio, representando uma vantagem competitiva (H. Chen, Chiang, & Storey, 2012).

A tecnologia que permite a extração de conhecimento tem sido cada vez mais procurada, havendo interesse na identificação de padrões e tendências que permitam suportar a tomada de decisão, trazendo grandes vantagens comparativamente às empresas que não usam este tipo de tecnologia (Di Tria, Lefons, & Tangorra, 2014b). O termo *Big Data* aplica-se aos dados que não podem ser processados ou analisados recorrendo a processos e ferramentas tradicionais (P. Zikopoulos, Eaton, DeRoos, Deutsch, & Lapis, 2011). As tecnologias de *Big Data* permitem que os dados possam ser mais facilmente acedidos e que sejam revelados aspetos até ao momento desconhecidos, havendo mais transparência nos dados que a organização tem disponíveis (Goss & Veeramuthu, 2013).

Para o armazenamento dos dados recolhidos são usados *Data Warehouses* (M. Y. Santos & Costa, 2016), contudo, o volume de dados que é necessário analisar está a alcançar tamanhos críticos para as abordagens tradicionais (Chevalier, El Malki, Kopliku, Teste, & Tournier, 2015). *Data Warehouses*, em contextos de *Big Data*, permitem a análise de grandes volumes de dados, extraindo informação relevante a partir dos mesmos, de forma a dar resposta às necessidades atuais das organizações.

Com a crescente utilização de *Data Warehouses*, são aumentadas as perspetivas de se ter dados mais atualizados, preparados para serem analisados, sendo criada a necessidade de existirem *Data Warehouses* em tempo real (Farooq & Sarwar, 2010). Com a introdução de tecnologia em tempo real, é possível capturar a mudança rápida nos dados e realizar uma análise atualizada a uma grande

quantidade de dados. A detecção de falha e factos suspeitos (Li & Mao, 2015) são exemplos de informação obtida com tecnologia em tempo real, importante para que possam ser tomadas decisões.

Porém, as ferramentas tradicionais não estão preparadas para o processamento de grandes volumes de dados em tempo real, para a atualização de dados de forma tão constante. É necessária uma aposta inteligente na tecnologia de forma a que não existam conflitos entre o processamento de interrogações aos dados e a extração de dados de outras fontes, nem grandes tempos de espera. É, por isso, importante que sejam tomadas algumas medidas que permitam que esse tempo real pretendido possa ser cumprido. O desafio coloca-se no tempo de acesso aos dados em tempo real, sem atrasos de processamento, na latência das operações (Li & Mao, 2015). É assim necessário que sejam conhecidos os requisitos de tempo real para a modelação e implementação de *Data Warehouses* em contextos de *Big Data*, permitindo obter informação dos dados de forma rápida e atualizada, o que trará diversas vantagens para o negócio.

1.2. Objetivos e Resultados Esperados

Com esta dissertação, pretende-se caracterizar os requisitos de tempo real que podem ser considerados na concretização de *Data Warehouses* em contextos de *Big Data* e propor uma arquitetura que permita a concretização deste tipo de repositórios. A arquitetura de processamento de dados utilizada será implementada e validada com um caso de demonstração, permitindo obter um conjunto de resultados e concluir acerca da adequabilidade da mesma.

Em termos de resultados espera-se a:

- Sistematização do estado de arte sobre a temática *Big Data Warehouses*;
- Análise das características de tempo real e trabalhos desenvolvidos na área;
- Proposta de uma arquitetura para a concretização de *Big Data Warehouses* em Tempo Real;
- Implementação e validação da arquitetura de processamento de dados com um caso de demonstração.

1.3. Abordagem de Investigação

Para o desenvolvimento desta dissertação, foi selecionada uma metodologia de investigação usada no desenvolvimento da investigação científica e que, entre outros, guia o processo de revisão bibliográfica usado na exploração dos conceitos e na descoberta dos trabalhos relacionados com o tema.

1.3.1. Metodologia de Investigação

A investigação elaborada nesta dissertação enquadra-se dentro de um processo de investigação mais completo baseado em *Design Science Research*, sendo utilizado, neste caso, como método de investigação uma experiência laboratorial (*benchmarking*). Assim, é utilizada a metodologia de investigação enquadrada na área de Sistemas de Informação, *Design Science Research Methodology (DSRM) for Information Systems*, de Peffers, Tuunanen, Rothenberger, & Chatterjee (2007) adequada à elaboração de investigação científica. Esta metodologia encontra-se dividida em seis fases, nomeadamente a identificação do problema e motivação, a definição dos objetivos, a conceção e desenvolvimento, a demonstração, a avaliação e, por fim, a comunicação, apresentando-se, de forma esquematizada, cada uma destas fases na Figura 1.

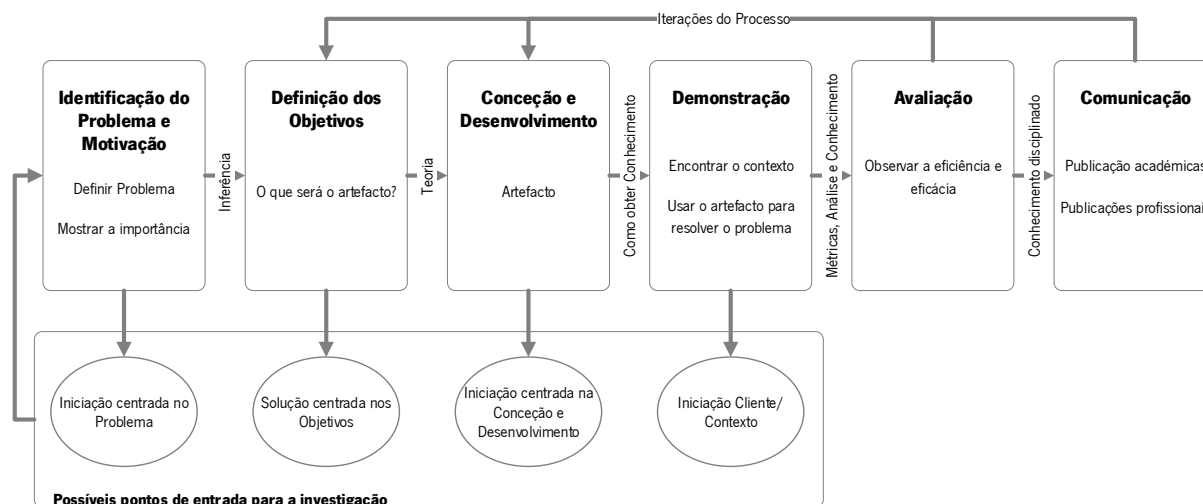


Figura 1 - *Design Science Research Methodology for Information Systems*. Adaptada de Peffers et al. (2007)

A utilização desta metodologia, como os autores desta sugerem, permite ter uma disciplina de trabalho orientada ao sucesso, importante no reconhecimento e avaliação dos resultados da investigação (Peffers et al., 2007). Além disso, segundo Geerts (2011), esta permite melhorar a produção, apresentação e avaliação da investigação científica e ao mesmo tempo ser coerente com os princípios e guias estabelecidos.

O fluxo de atividades e eventos, bem como os resultados que vão sendo atingidos nas várias fases, permitem elaborar um trabalho coerente, organizado e detalhado. Desta forma, o desenvolvimento desta dissertação está organizado num conjunto de tarefas orientadas à metodologia da Figura 1. No início da dissertação, é iniciada a leitura de documentos científicos, de forma a conhecer melhor os conceitos envolvidos na temática bem como a sua importância, permitindo a identificação do âmbito do trabalho. Inicia-se, assim, a elaboração do plano de trabalho, no qual é planeado o trabalho que se pretende vir a desenvolver, identificada a motivação, a importância e a necessidade de investigação neste

âmbito, os objetivos, a abordagem metodológica e as tarefas envolvidas neste trabalho. O ponto de entrada para a investigação na metodologia é centrado no problema, sendo que para cada uma das fases da metodologia existem um conjunto de tarefas que a regem e permitem a evolução do trabalho para as fases seguintes, nomeadamente:

- A primeira fase é a identificação do problema e motivação que consiste na identificação dos aspetos que marcam a diferença, que criam a oportunidade na definição do problema, que levarão ao desenvolvimento da solução (Peppers et al., 2007). Define-se a situação atual, de forma a motivar e dar relevância ao trabalho desenvolvido (Geerts, 2011), permitindo também esclarecer todo o processo.
- Segue-se a definição dos objetivos que consiste na definição daqueles que serão objetivos que permitam resolver o problema (Peppers et al., 2007) e obter o conhecimento necessário para pôr a solução em prática. É necessário conhecimento dos métodos, tecnologias e teorias da área em questão (Geerts, 2011).
- Assim que se encontra identificado o problema e a motivação e definidos os objetivos, é iniciada a revisão do estado de arte. Esta revela-se importante não só para se conhecer melhor os conceitos, como o trabalho já existente. Primeiramente, é efetuada uma leitura mais genérica e, depois de os conceitos estarem mais organizados, é elaborada a revisão de literatura. Durante a elaboração da revisão do estado de arte, é efetuado um enquadramento tecnológico de forma a se conhecer melhor cada uma das tecnologias, bem como a sua necessidade e utilidade no modelo desejado.
- Aquando conhecidas as tecnologias e, com base no trabalho já desenvolvido por outros autores, inicia-se a conceção e desenvolvimento. Esta fase visa a evolução e progresso dos vários métodos e modelos necessários para a solução do problema (Peppers et al., 2007). Começa a ser desenhada uma proposta de arquitetura, estando constantemente a ser alterada e alinhada com o que é desejado para o trabalho.
- A demonstração consiste na validação da solução desenvolvida, nomeadamente na implementação e validação da arquitetura proposta. Nesta fase, é provado se aquilo que foi desenvolvido é, ou não, capaz de resolver o problema (Geerts, 2011), envolvendo a definição de cenários de teste e a elaboração de testes.
- A avaliação é a fase em que são retiradas as conclusões do trabalho desenvolvido, podendo ser comparado com trabalhos já desenvolvidos na mesma área. É observado e avaliado em que medida é que a solução desenvolvida responde ao problema, comparando os objetivos com os

resultados atingidos (Geerts, 2011). Esta fase poderá levar a que tenham que ser feitos ajustes no que foi desenvolvido até ao momento, caso os resultados obtidos não sejam os desejados, ou se pretendam melhorias nos mesmos.

- A comunicação visa apresentar e divulgar os resultados que vão sendo obtidos no trabalho (Peffer et al., 2007), realçando a oportunidade criada e a utilidade do trabalho desenvolvido (Geerts, 2011). Esta fase inclui a escrita da dissertação, que se alonga ao longo de todo o trabalho, bem como outras oportunidades de divulgação do trabalho desenvolvido na comunidade científica.

1.3.2. Processo de Revisão Bibliográfica

Para a elaboração da revisão do estado de arte é necessária a pesquisa dos principais conceitos desta dissertação bem como os trabalhos relacionados com a sua temática. A definição dos critérios de pesquisa, fontes e palavras chave usadas são os principais aspetos para a pesquisa e recolha de documentos.

Para a pesquisa de documentos definiu-se as bases de dados de referência, nomeadamente Scopus, Web of Science, Google Scholar, IEEE Xplore, Springer e RepositóriUM, nas quais foram usadas um conjunto de palavras chave de pesquisa. As palavras chave advêm da temática da dissertação e foram “big data”, “data warehouse”, “big data warehouse” e “real-time”, sendo utilizada também a combinação entre as palavras chave. A recolha de documentos foi essencialmente entre outubro de 2016 e fevereiro de 2017, contudo, durante todo o período da dissertação houve procura de novos documentos relacionados com o tema.

A filtragem de documentos encontrados é inicialmente efetuada através da leitura do título e *abstract* do documento em questão, prestando também atenção ao número de citações. Com esta análise os documentos são considerados muito importantes e potencialmente importantes, selecionando assim os relevantes para esta dissertação. A pesquisa não necessita de estar limitada por data de publicação porque, como são conceitos recentes, a documentação é também recente.

Os documentos obtidos neste processo e os documentos fornecidos pela professora orientadora foram lidos e referenciados na dissertação, quando o conteúdo se revelava interessante e importante.

1.4. Contributos do Trabalho

O trabalho desenvolvido nesta dissertação permitiu já a publicação e submissão dos seguintes artigos científicos:

- Santos, M. Y., Costa, C., Galvão, J., Andrade, C., Martinho, B. A., Vale Lima, F. & Costa, E. (2017). Evaluating SQL-on-Hadoop for Big Data Warehousing on Not-So-Good Hardware. In Proceedings of the 21st International Database Engineering & Applications Symposium (pp. 242–252). New York, NY, USA: ACM. <https://doi.org/10.1145/3105831.3105842>;
- Santos, M. Y., Oliveira e Sá, J., Andrade, C., Vale Lima, F., Costa, E., Costa, C., Martinho, B. & Galvão, J. (2017). A Big Data system supporting Bosch Braga Industry 4.0 strategy. International Journal of Information Management. <https://doi.org/https://doi.org/10.1016/j.ijinfomgt.2017.07.012>;
- Santos, M. Y., Oliveira e Sá, J., Costa, C., Galvão, J., Andrade, C., Martinho, B., Vale Lima, F. & Costa, E. (2017). A Big Data Analytics Architecture for Industry 4.0. In WorldCist'17 - 5th World Conference on Information Systems and Technologies.

Estes artigos científicos enquadram-se no projeto de investigação “P30 - Business Intelligence Platform for Data Integration”¹ entre a Universidade do Minho e a BOSCH, na qual a autora desta dissertação se encontra desde junho de 2016. O seu contributo apresenta-se quer na análise de arquiteturas existentes, quer na compreensão e teste de diversas tecnologias de processamento de dados em *Big Data*.

Além destes, foi elaborado o seguinte capítulo de livro, cuja publicação se espera que venha a acontecer em 2018, no qual se encontram os principais contributos desta dissertação:

- Vale Lima, F., Costa, C. & Santos, M. Y. (2018). Real-Time Big Data Warehousing. In Emerging Perspectives in Big Data Warehousing. IGI Global.

1.5. Organização do Documento

Este documento apresenta-se organizado em seis principais capítulos. Neste primeiro capítulo encontra-se identificado o problema e a motivação para o desenvolvimento da dissertação nesta temática. Além disso, são identificados os objetivos e resultados esperados bem como a abordagem de

¹ O objetivo do projeto é desenvolver um sistema integrado de dados que permita, através de um processo interativo, o desenvolvimento do *Data Warehouse Organizacional*, contribuindo para o aumento da qualidade das operações de fábrica, em termos da eficiência do acesso e qualidade da informação crítica necessária para a tomada de decisão e envolvimento dos atores, a montante e jusante da cadeia de valor.

investigação adotada, nomeadamente a metodologia de investigação e o processo de revisão bibliográfica. Apresentam-se também os contributos do desenvolvimento deste trabalho bem como a estrutura que o documento do trabalho segue.

O segundo capítulo apresenta a revisão de literatura dos principais conceitos do tema desta dissertação, bem como trabalhos desenvolvidos por outros autores. São explorados os três principais conceitos, nomeadamente *Big Data*, *Data Warehouse* e *Big Data Warehouse*, as suas características e como o armazenamento dos dados pode ser efetuado. Para a implementação de um *Big Data Warehouse* em tempo real, são também explorados os requisitos em tempo real e conhecidos os trabalhos desenvolvidos neste sentido, tanto na abordagem tradicional como em contextos de *Big Data*.

No terceiro capítulo são exploradas as principais tecnologias que permitem a concretização de um *Big Data Warehouse* em tempo real. Apresentam-se as tecnologias do ecossistema do Hadoop², as tecnologias que permitem armazenar os dados e as tecnologias que permitem processar os dados em tempo real.

O quarto capítulo apresenta o trabalho experimental realizado, sendo apresentada a arquitetura e definidos os cenários de teste para a avaliação da solução. Esta solução recolhe dados em *streaming* e em *batch*, processa-os e armazena-os, estando assim disponíveis para a análise.

No quinto capítulo encontra-se o caso de demonstração, no qual é analisado o comportamento da tecnologia Kafka³, tecnologia de recolha de dados, o comportamento da tecnologia Spark Streaming⁴, que permite o processamento de dados, o comportamento da tecnologia Cassandra⁵, que permite o armazenamento de dados em tempo real, e o comportamento da tecnologia Hive⁶, para o armazenamento de dados em tempo real e de dados históricos, para permitir a concretização do *Big Data Warehouse* em tempo real. Para a análise de desempenho destas tecnologias são efetuadas um conjunto de consultas aos dados, recorrendo à tecnologia Presto⁷. Os resultados obtidos são analisados, pelo que são tecidas considerações sobre as tecnologias e sobre o contexto criado.

Por fim, este documento termina com as conclusões deste trabalho, onde são apresentados os principais resultados do desenvolvimento deste trabalho bem como o trabalho futuro, e as referências bibliográficas que suportam esta dissertação.

² <http://hadoop.apache.org/>

³ <http://kafka.apache.org/>

⁴ <https://spark.apache.org/streaming/>

⁵ <http://cassandra.apache.org/>

⁶ <https://hive.apache.org/>

⁷ <https://prestodb.io/>

2. BIG DATA WAREHOUSES

Neste capítulo encontra-se a revisão de literatura da temática *Big Data Warehouse*, inerente a esta dissertação. São apresentados os principais conceitos, explicado como o armazenamento de dados pode ser efetuado e apresentadas as principais características de tempo real e os trabalhos relacionados com a temática.

2.1. Principais Conceitos

Nesta secção, apresentam-se os principais conceitos envolvidos na temática desta dissertação, nomeadamente *Big Data*, *Data Warehouse* e a junção destes dois conceitos, *Big Data Warehouse*.

2.1.1. *Big Data*

Nos últimos anos, tem havido um grande aumento de fontes de dados e de mecanismos e aplicações de recolha de dados, que geram um grande volume de dados (Golab & Johnson, 2014), através de sensores, dispositivos eletrónicos, emails, redes sociais, páginas *web*, entre outros (P. Zikopoulos et al., 2011). Para as organizações, a análise destes dados revela-se importante, tendo havido cada vez mais interesse no conceito de *Business Intelligence* (BI). As ferramentas de BI permitem a análise dos dados disponíveis e com esta informação podem ser conhecidas as oportunidades do negócio bem como os interesses e opiniões do cliente (H. Chen et al., 2012; Goss & Veeramuthu, 2013).

Os dados gerados tornam-se cada vez mais variados e conseqüentemente mais complexos (P. Zikopoulos et al., 2011), levando, o seu crescimento, a uma integração para a análise cada vez mais complexa. Para este processo, as infraestruturas e tecnologias tradicionais começam a revelar-se limitadas, tornando-se, através delas, quase impossível a extração de conhecimento útil (Cuzzocrea, Song, & Davis, 2011). As tecnologias tradicionais deixam de conseguir responder aos pedidos a tempo, deixam de processar de forma eficiente a quantidade de dados envolvida (Durham, Rosen, & Harrison, 2014; Kaisler, Armour, Espinosa, & Money, 2013; Krishnan, 2013). O facto de estas não conseguirem adquirir, gerir e processar dados num tempo tolerável, gera a necessidade de uma aposta na tecnologia, na introdução de soluções baseadas em conceitos de *Big Data* (M. Chen, Mao, & Liu, 2014; Goss & Veeramuthu, 2013).

Big Data requer tecnologias de armazenamento, de gestão, de análise e de visualização, diferentes daquelas que provavelmente a organização possui (H. Chen et al., 2012). No seu ambiente, as bases de dados relacionais tradicionais e os sistemas de ficheiros existentes são aumentados,

permitindo que haja um maior volume, velocidade e capacidade de resposta transaccional, bem como um aumento da quantidade e variedade dos dados (Goss & Veeramuthu, 2013).

O *know-how* e as tecnologias de *Big Data* vieram revolucionar o que se possuía até ao momento, potenciando a tomada de decises nas organizaes e revelando-se uma oportunidade para os estudantes e para os profissionais de Tecnologias de Informaço (TI) (H. Chen et al., 2012). Isto deve-se não simplesmente pela tecnologia, mas pelo fenómeno que torna todo o processo de armazenamento e processamento de dados bastante mais eficiente. Inicialmente, era usado em ambiente de investigaço, mas mais tarde partiu para o ambiente comercial, para as organizaes (Mathur, Sihag, Bagaria, & Rajawat, 2014), devido a se revelar uma oportunidade de análise, de conhecimento do negócio (Golab & Johnson, 2014).

O termo começou a ser referenciado em 1970, tendo aumentado bastante a partir de 2008 (Mathur et al., 2014; Ularu, Puican, Apostu, & Velicanu, 2012). É um conceito bastante ambíguo e não possui uma definição entendida e aceite por todos, sendo que nem sempre foi visto de uma forma positiva (Ward & Barker, 2013). Apesar de o termo ser normalmente associado a grandes quantidades de dados, está longe de ser a sua única característica ou o seu único desafio (P. Zikopoulos et al., 2011).

De seguida, apresenta-se o conceito *Big Data*, as suas principais características, os seus desafios e como o processamento de dados é efetuado.

a) O Conceito

Big Data, segundo Krishnan (2013), é definido pelos volumes de dados disponíveis, na variedade de fontes de dados, que são gerados a diferentes velocidades e possuem diferentes graus de ambiguidade. *Big Data* representa mais variedade, mais quantidade, mais utilizadores, mais velocidade e mais complexidade (Goss & Veeramuthu, 2013). É o desafio da utilização de um grande volume de dados, mas ao mesmo tempo uma oportunidade para as organizaes aumentarem a sua eficácia (Mathur et al., 2014).

Segundo M. Chen et al. (2014), permite que surjam novas oportunidades, novos desafios, a potencialidade de conhecimento com mais detalhe e a descoberta de padrões e tendências escondidas nos dados. A análise dos dados possui grande e crescente poder em criar valor para o negócio e para os consumidores (M. Chen et al., 2014; Kaisler et al., 2013).

No entanto, antes de o *Big Data* surgir, o processamento automático de dados já existia, mas o que fez a mudança foi o facto de ser possível processar os dados de uma forma tão rápida, escalável e flexível (Krishnan, 2013). Com as técnicas de *Big Data*, é extraída informação útil de conjuntos de dados de grande dimensão, diversos, distribuídos e heterogéneos, o que acelera o progresso da descoberta

científica e inovação, levantando novas questões, potenciando o desenvolvimento de algoritmos e novas ferramentas de análise de dados, o crescimento económico e a melhoria da qualidade de vida e saúde (H. Chen et al., 2012). P. Zikopoulos et al. (2011) afirma que as técnicas usadas permitem retirar valor de grandes quantidades de dados, estejam eles em formatos estruturados, semiestruturados ou não estruturados, conhecendo os dados que são e não são importantes para a organização. Segundo o *McKinsey Global Institute*, a introdução de *Big Data* poderá melhorar também a produtividade e a competitividade, criando grandes benefícios para os consumidores (M. Chen et al., 2014; Manyika et al., 2011).

O termo *Big Data* é usado para conjuntos de dados que são difíceis de adquirir, armazenar, pesquisar, visualizar e analisar, com volume e complexidade de dados imensa e em rápido crescimento (Z. Liu, Yang, & Zhang, 2013). Segundo Oracle, as fontes de dados podem ser de três tipos, nomeadamente os dados tradicionais do negócio, ou seja, informação do cliente e das transações, como é o caso do ERP (*Enterprise Resource Planning*) e CRM (*Customer Relationship Management*), os dados gerados pelas máquinas e sensores e os dados das redes sociais como Facebook, Twitter, entre outras (Dijcks, 2013; Khan, Uddin, & Gupta, 2014; Z. Liu et al., 2013).

Todavia, seja qual for a área de negócio, o *Big Data* possui um papel importante de atuação, representando vantagem competitiva, para serem tomadas melhores decisões e conhecidas as tendências (Chandarana & Vijayalakshmi, 2014). Nos últimos anos, o crescimento rápido de dados deve-se principalmente às empresas com serviços na internet, como a Google e Facebook, o que levantou preocupações com a necessidade de infraestruturas com mais capacidade de armazenamento, para ser possível guardar todos os dados que são gerados (M. Chen et al., 2014). Khan et al. (2014) afirma que o *Big Data* tem muito potencial desde que a grande quantidade de dados não se torne um problema demasiado difícil para se resolver, pois aí não estará a ser extraído o valor desejado e importante dos dados.

b) Principais Características

Big Data normalmente surge associado às grandes quantidades de dados, contudo, este termo apresenta outras características que o explicam e ampliam as suas capacidades (Mathur et al., 2014; P. Zikopoulos et al., 2011). As suas três principais características, volume, velocidade e variedade, foram primeiramente referenciadas por Laney (2001) como as três dimensões dos desafios e oportunidades do crescimento de dados, os *3 V's*. O volume é referente à quantidade de dados que é gerada continuamente, a velocidade à rapidez com que os dados entram e saem, e a variedade aos diferentes

tipos e fontes de dados (Krishnan, 2013). De seguida, são descritas cada umas destas características, que se apresentam ilustradas na Figura 2.

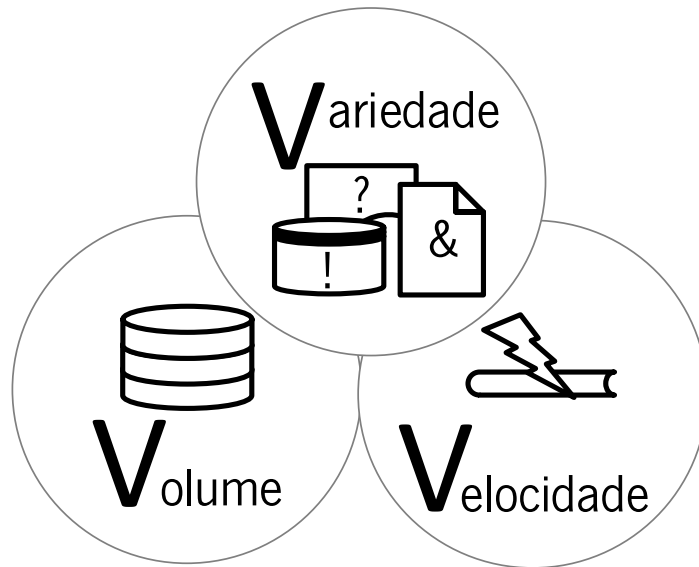


Figura 2 - Modelo dos 3 V's. Adaptada de P. Zikopoulos et al. (2011)

O volume caracteriza-se pela grande quantidade de dados existentes atualmente, e aqueles que se espera vir a recolher no futuro (Chandarana & Vijayalakshmi, 2014; Durham et al., 2014; Khan et al., 2014; Mathur et al., 2014), que são gerados e armazenados e têm vindo a crescer cada vez mais (M. Chen et al., 2014; P. Zikopoulos et al., 2011). Geram-se dados com cada vez mais detalhe e diferentes tipos de dados, o que representa muitos mais dados a ser analisados e geridos (Krishnan, 2013; Laney, 2001; Sagiroglu & Sinanc, 2013). Com o grande crescimento do volume de dados, as organizações revelam cada vez mais interesse na sua análise, no conhecimento do negócio, dos clientes e do mercado. O problema é que o volume de dados disponível nas organizações tem vindo a aumentar, enquanto que a percentagem de dados que se conseguem analisar tem vindo a diminuir (P. Zikopoulos et al., 2011). Outro problema é que o volume está a crescer rapidamente comparativamente aos recursos computacionais disponíveis nas organizações (Katal, Wazid, & Goudar, 2013), havendo a necessidade de encontrar uma solução de armazenamento. De forma a otimizar o processo, têm surgido várias soluções passando pela utilização de um *cluster* de máquinas com vários nós, garantindo o armazenamento e processamento dos dados (Laney, 2001; Z. Liu et al., 2013), pois as tecnologias tradicionais não são capazes de suportar o processamento (Khan et al., 2014). De destacar que o volume é a principal característica que distingue o *Big Data* das tecnologias e ferramentas tradicionais, e que representa o “Big” no termo (Demchenko, Grosso, de Laat, & Membrey, 2013; Katal et al., 2013).

A velocidade caracteriza-se pelo rápido crescimento de dados, por estarem constantemente a ser gerados mais dados, pelo acumular de grandes quantidades de dados, que são recolhidos e

armazenados (Chandarana & Vijayalakshmi, 2014; Durham et al., 2014; Katal et al., 2013; Z. Liu et al., 2013; Mathur et al., 2014). O facto de representar uma oportunidade na recolha e análise de dados, de forma rápida e oportuna (M. Chen et al., 2014), leva ao interesse numa análise em tempo real para dar suporte rápido e valioso na tomada de decisão (Khan et al., 2014; Z. Liu et al., 2013; Mathur et al., 2014). Porém, a velocidade com que os dados são gerados e fluem, implica a necessidade de gerir esta velocidade e também a capacidade de processamento (Khan et al., 2014; Laney, 2001). Segundo a IBM, a velocidade é entendida como o tempo que demora a processar grandes quantidades de dados, sendo que o processamento rápido maximiza a eficiência do negócio (Ularu et al., 2012; P. C. Zikopoulos et al., 2012).

A variedade representa os diferentes formatos, as estruturas e as inconsistências na semântica dos dados (Demchenko et al., 2013; Laney, 2001). Caracteriza-se por os dados estarem em formas distintas, podendo ser estruturados, não estruturados ou semiestruturados, estarem em texto, multimédia, provir de emails, redes sociais, sensores, entre outros. Estes dados podem ter origem em fontes muito distintas, estando envolvidos sistemas, bases de dados e aplicações diferentes (Chandarana & Vijayalakshmi, 2014; M. Chen et al., 2014; Durham et al., 2014; Khan et al., 2014; Krishnan, 2013; Mathur et al., 2014). As inconsistências na semântica dos dados torna necessária a sua integração e junção, o que se revela um desafio quando os dados não estão documentados (Z. Liu et al., 2013; Sagioglu & Sinanc, 2013), principalmente quando necessário que os dados sejam acedidos em *streaming* (Z. Liu et al., 2013). O facto de existir grande variedade nos dados afeta diretamente a integridade destes, podendo os dados conter erros (Khan et al., 2014).

Do cruzamento entre os 3 *V's* surgem outras três características, segundo Krishnan (2013), nomeadamente a ambiguidade, a viscosidade e viralidade, como se apresenta na Figura 3. A ambiguidade manifesta-se entre a variedade e o volume pela falta de metadados. Veja-se que, por exemplo, um “M” e “F” podem ser entendidos como Masculino e Feminino ou Segunda (em inglês *Monday*) e Sexta (em inglês *Friday*). A viscosidade manifesta-se entre o volume e a velocidade medindo a resistência para o fluxo no volume de dados, podendo haver resistência ao nível da tecnologia ou regras de negócio. A viralidade manifesta-se entre a velocidade e a variedade medindo e descrevendo o quão rápido os dados são partilhados na rede.

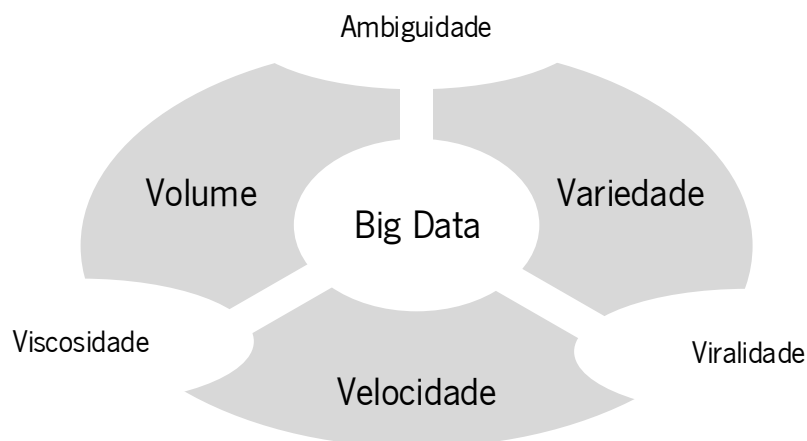


Figura 3 - Características entre os 3 V's. Adaptada de Krishnan (2013)

A primeira abordagem de características de *Big Data* consistia em apenas três características, os tão conhecidos 3 V's, mas vários autores defendem que estas não são suficientes para caracterizar o termo. Chandarana & Vijayalakshmi (2014) desenvolveram o modelo de 5 V's no qual são adicionadas duas propriedades, nomeadamente, a veracidade e o valor. Mais tarde, Khan et al. (2014) sugerem a validade e a volatilidade, apresentando o modelo dos 7 V's. Estas características são descritas de seguida, pelo que se apresentam ilustradas na Figura 4.

A veracidade acaba por ser transversal às outras propriedades, caracterizando-se pela incerteza dos dados, dada a necessidade de gerir dados de diferentes níveis de confiabilidade, previsibilidade e tipos de dados (Demchenko et al., 2013; Mathur et al., 2014). Esta característica está intimamente relacionada com a qualidade dos dados e advém principalmente do facto de os dados serem gerados com tanta velocidade, que nem há tempo para estes serem limpos. É criada a necessidade da existência de mecanismos que lidem com dados precisos, imprecisos, corretos ou incorretos (Chandarana & Vijayalakshmi, 2014; Du, 2015). Esta é a característica mais preocupante ao nível do processamento de *Big Data*, da análise que é efetuada e resultados obtidos (Khan et al., 2014), pois uma análise baseada em dados errados poderá levar a más decisões (Du, 2015).

O valor representa o que se obtém com a recolha de dados e posterior análise (Demchenko et al., 2013). É conhecida informação que até então não era, tendo como desafio a identificação, transformação e extração de dados para que possa ser retirada informação útil para a tomada de decisão (Z. Liu et al., 2013). O conhecimento escondido do negócio traz vantagem competitiva e representa o valor do *Big Data* (Chandarana & Vijayalakshmi, 2014; M. Chen et al., 2014; Du, 2015), compensando o investimento efetuado (Khan et al., 2014).

A validade representa a exatidão dos dados relativamente ao uso que lhes se pretende dar. Os dados têm que ser devidamente compreendidos de forma a perceber se fazem ou não sentido no contexto em que se inserem (Khan et al., 2014).

A volatilidade representa o tempo no qual os dados armazenados estão válidos, sendo de especial atenção nas análises em tempo real. Para isso, é necessário considerar o armazenamento disponível e a segurança dos dados (Du, 2015; Khan et al., 2014).

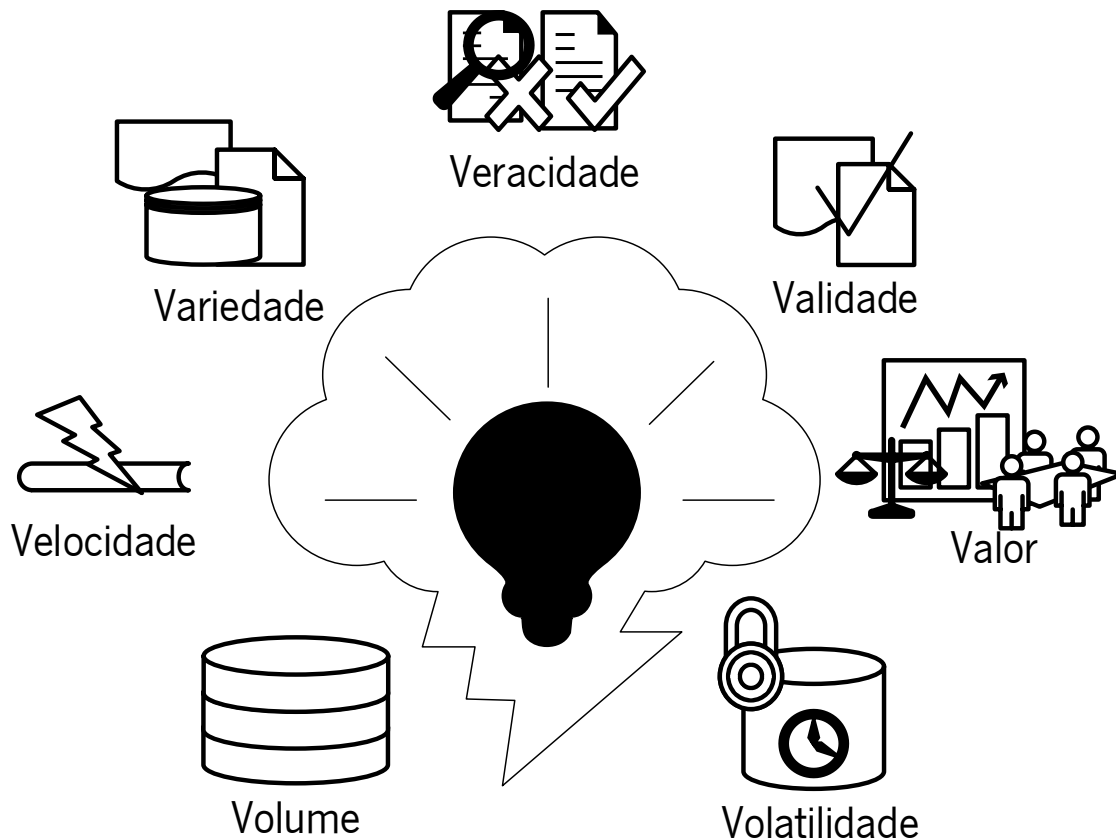


Figura 4 - *V's Big Data*

Além destas, duas propriedades também importantes são a variabilidade, relacionada com a alteração dos dados e o valor dos dados dependendo do contexto, e a visualização, relacionada com a forma como os dados devem ser compreendidos (Du, 2015; Katal et al., 2013). Segundo Kaisler et al. (2013) e Katal et al. (2013), a complexidade é também uma característica importante, relacionada com a integração das várias fontes de dados e o relacionamento dos dados, o que pode levar a conclusões até ao momento desconhecidas.

c) Desafios

Os desafios de *Big Data* começaram a surgir devido à necessidade de tecnologia e métodos para se recolher, armazenar, gerir e analisar os dados, que podem ser heterógenos, algo que não é suportado pelos sistemas tradicionais (M. Chen et al., 2014). Para isso, é importante que seja usada a potencialidade das ferramentas com que se lida, para que seja possível retirar o maior partido delas e obter resultados de forma mais rápida (Durham et al., 2014).

Com a rápida evolução do conceito de *Big Data*, existem vários desafios ou obstáculos com que as tecnologias e os especialistas têm que lidar, os quais se apresentam de seguida:

- Privacidade, Segurança e Confiança - Os dados recolhidos possuem informações acerca dos utilizadores, informações que são importantes para se conhecer as tendências e opiniões, o que traz valor para o negócio (Kaisler et al., 2013). Contudo, é necessário que haja atenção com a proteção de conteúdos, pois são armazenados dados sensíveis, como é o caso dos dados das redes sociais. A organização terá que ser capaz de proteger a privacidade e segurança dos utilizadores, pois estes já não a conseguem controlar (Chandarana & Vijayalakshmi, 2014; M. Chen et al., 2014). A precisão e fiabilidade dos dados também são um desafio, como refere o desafio da confiança de Z. Liu et al. (2013), sendo de alguma dificuldade validar os dados disponíveis (Kaisler et al., 2013).
- Competências Tecnológicas e Analíticas - *Big Data* cria grandes expectativas em tecnologias e ferramentas capazes de armazenar, processar e analisar grandes quantidades de dados e dados complexos (Chandarana & Vijayalakshmi, 2014). Para lidar com estas tecnologias, há a necessidade de as organizações possuírem os conhecimentos necessários, com especialistas de TI nas organizações (Ularu et al., 2012) e formando especialistas competentes nas universidades (Katal et al., 2013). Além disso, segundo Kaisler et al. (2013), com a aposta na tecnologia e dadas as devidas formações aos utilizadores, estes serão mais bem sucedidos, mais produtivos e poderão criar impacto no negócio.
- Armazenamento e Processamento de Dados - As tecnologias tradicionais suportam a inserção de dados e armazenamento, contudo, para um volume de dados tão elevado a inserção, atualização e seleção de dados, bem como a utilização de dados sem estrutura, fica comprometida (Chandarana & Vijayalakshmi, 2014). As tecnologias emergentes proporcionam a capacidade de se processar dados de múltiplos formatos e estruturas, com capacidade para os dados do presente e do futuro (M. Chen et al., 2014; Krishnan, 2013). Atualmente, com a quantidade de fontes de dados, como é referido no desafio do processamento de Z. Liu et al.

(2013), é necessário que haja armazenamento para esses dados e que seja possível a sua análise. Porém, a disponibilidade e eficiência do processamento torna-se algo crítico, bem como a escalabilidade do armazenamento (Z. Liu et al., 2013).

- Dados Estruturados e Não Estruturados - Com a introdução de *Big Data*, os dados não estruturados têm vindo a aumentar e não podem ser analisados como dados estruturados (Z. Liu et al., 2013). O problema é que trabalhar com dados não estruturados é mais complexo, e, por isso, seria interessante a capacidade de automaticamente gerar os metadados para descrever os dados (Chandarana & Vijayalakshmi, 2014; Katal et al., 2013; Labrinidis & Jagadish, 2012). Há ainda que considerar que os dados vão sendo recolhidos ao longo do tempo, pelo que podem alterar o seu formato, mudar a necessidade de serem analisados ou a forma como são analisados (Kaisler et al., 2013).
- Mecanismos Analíticos - A análise de dados deve estar desenhada de acordo com os dados que se possui e com o negócio. Deve existir um equilíbrio entre a abordagem tradicional e as bases de dados não relacionais, para que a análise aos dados possa ter bom desempenho (M. Chen et al., 2014). Há também que ter em atenção que a informação que é analisada é aquela que está nos dados, e nem sempre as fontes de dados estão corretas (Labrinidis & Jagadish, 2012).
- Utilidade dos Dados - Decidir que dados são úteis para responder a uma questão analítica é um importante desafio, porque a dificuldade não está em recolher dados, mas sim em saber quais os dados úteis e importantes (Durham et al., 2014). A recolha de dados tem se tornado um “vício”, mas é importante conhecer aqueles que devem ser armazenados e quais os que podem ser descartados (M. Chen et al., 2014; Kaisler et al., 2013). *Big Data* foca-se na qualidade dos dados, dos resultados e conclusões, devendo a seleção de informação ser a mais adequada e com autenticidade (Chandarana & Vijayalakshmi, 2014; Katal et al., 2013). Há, por isso, a necessidade de filtrar apenas informação útil, que poderá ter interesse para o negócio (Labrinidis & Jagadish, 2012), pois, afinal de contas, nem todos os dados possuem o mesmo valor ou são criados da mesma forma, sendo importante distinguir a sua análise (Kaisler et al., 2013).

d) Processamento de Dados

O processamento de dados pode ser efetuado de forma centralizada, numa abordagem tradicional, na qual os dados são recolhidos para um único local de armazenamento centralizado e aí processados até estar terminado, ou de uma forma distribuída, na qual os dados são processados pelas várias máquinas (os nós) do *cluster* distribuindo as suas tarefas. Ambas as abordagens têm vantagens

e desvantagens, assim como vários desafios (Krishnan, 2013). Os dados podem ser processados de várias formas, segundo Du (2015), nomeadamente em:

- *Batch* - Neste tipo de processamento, os dados são lidos da fonte de dados, processados e armazenados na base de dados ou ficheiro de destino. As tarefas são executadas uma de cada vez, de forma sequencial. Este tipo de processamento é implementado no Hadoop através do paradigma MapReduce, sendo usado o sistema de ficheiros distribuído *Hadoop Distributed File System* (HDFS).
- *Real-Time* - Neste tipo de processamento, os dados são processados à medida que são recolhidos, depois de os dados estarem criados ou serem alterados, obtendo-se resultados quase de forma imediata. Caracteriza-se por executar as tarefas em paralelo, de forma a conseguir apresentar resultados rapidamente.
- *Streaming* - Neste tipo de processamento, os dados estão continuamente a ser processados, obtendo-se resultados à medida que os novos dados surgem, à medida que estes eventos acontecem.

Tal como referido anteriormente, constitui um dos objetivos desta dissertação a utilização da arquitetura para processamento de dados que permita a concretização do conceito de *Big Data Warehouse* com processamento de dados em tempo real. Para que tal seja possível, é necessário compreender o fluxo de dados do processamento de *Big Data*. Ao contrário do tradicional processo de Extração-Transformação-Carregamento (ETL), o *Big Data* passa pelo processo de Extração-Carregamento-Transformação (ELT). O fator diferenciador é que no processo de ELT os dados são recolhidos e armazenados numa área de estágio e só quando transformados é que são carregados no *Data Warehouse* (Freudenreich et al., 2013; Waas et al., 2013).

O fluxo de dados de processamento de *Big Data*, segundo Krishnan (2013), possui quatro fases, nomeadamente, a recolha, o carregamento, a transformação e a extração, como se apresenta na Figura 5. A recolha é efetuada a diferentes fontes de dados tipicamente para uma área de estágio. No carregamento é aplicada uma estrutura aos dados e associados os metadados, podendo ser particionados horizontalmente (limitar as linhas) ou verticalmente (limitar as colunas). Na transformação é aplicado um conjunto de regras de negócio aos dados, de forma a eliminar problemas nos dados. Por fim, os dados são extraídos para poder ser feita a sua análise, visualização e elaboração de *reports*, para integração num *Data Warehouse*.

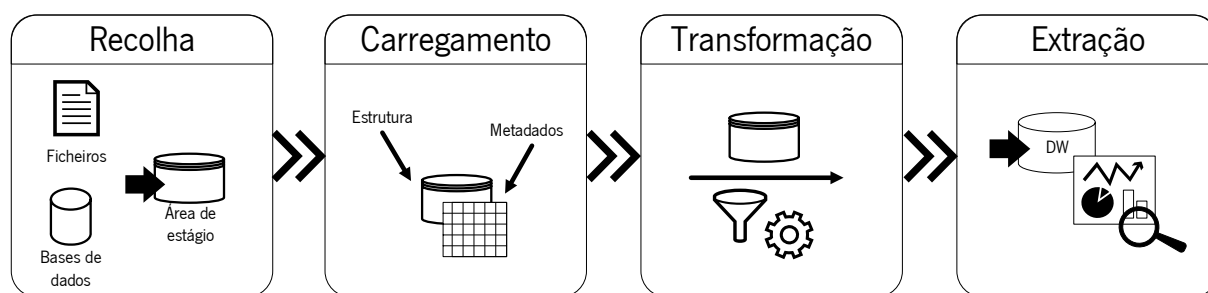


Figura 5 - Processo ELT. Adaptada de Krishnan (2013)

2.1.2. Data Warehouse

Nesta secção é assumido que o leitor está familiarizado com o conceito de *Data Warehouse*, pelo que de seguida apenas é apresentada uma breve síntese do seu objetivo e enquadramento num sistema de BI.

Data Warehouse (DW) é o componente essencial de uma infraestrutura de BI, muito pelo valor dos indicadores estratégicos que este traz para a tomada de decisão (Vaisman & Zimányi, 2012). Revela-se importante em análises estatísticas, para a extração de informação relevante, descoberta de relações escondidas e padrões nos dados (Di Tria, Lefons, & Tangorra, 2014a). Permite recolher, armazenar e gerir os dados, o que suporta o processo de tomada de decisão (Krishnan, 2013).

Um DW deve ser consistente, facilmente se adaptar à mudança e apresentar informação atualizada, permitindo fornecer informação para o negócio (Kimball & Ross, 2013). No sistema tradicional, existem um conjunto de dados que são armazenados em sistemas *Online Transaction Processing* (OLTP), que depois periodicamente são exportados para sistemas *Online Analytical Processing* (OLAP) através de um processo de ETL. No tradicional DW armazenam-se os dados, como se apresenta na Figura 6, que provêm de um conjunto de fontes de dados sobre as quais é executado um processo ETL, sendo que parte dos dados também podem ser armazenados nos *Data Marts*. Sobre o DW podem ser utilizadas um conjunto de ferramentas e algoritmos que efetuam consultas aos dados, não só dados atuais, como dados históricos, do qual resulta a análise de dados, como *reports* e *dashboards* (Kimball & Ross, 2013; Krishnan, 2013; Vaisman & Zimányi, 2012; Waas et al., 2013).

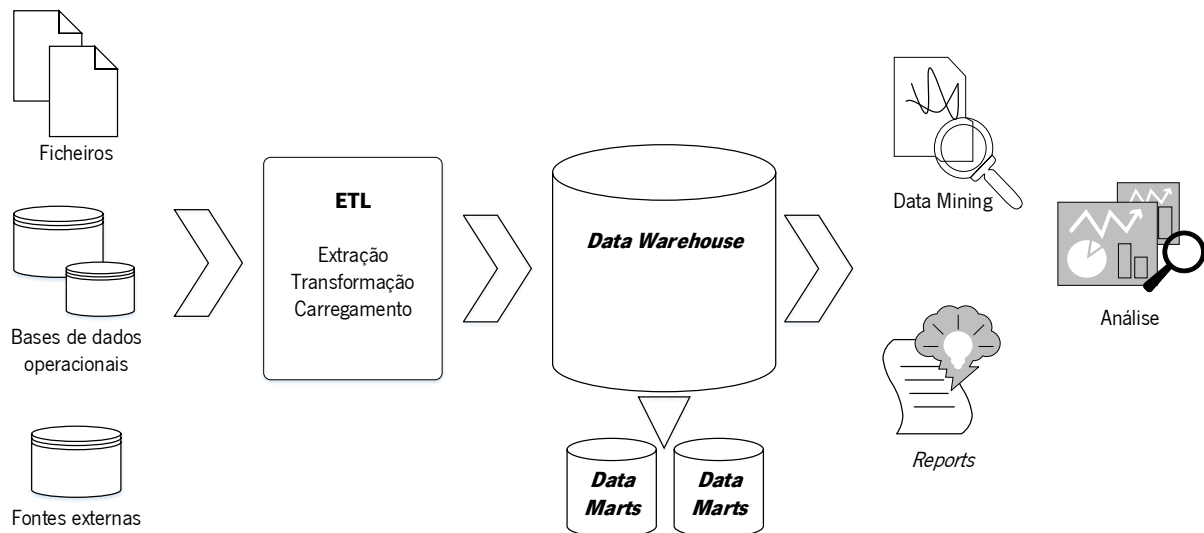


Figura 6 - Arquitetura de DW. Adaptada de Kimball & Ross (2013)

Tipicamente, uma infraestrutura de BI é responsável pelo processo de ETL das fontes de dados para o DW, o qual passa por três fases (Waas et al., 2013). A extração consiste em ler e perceber os dados das fontes de dados, copiando esses mesmos dados para a área de estágio para futura manipulação. A transformação consiste nas alterações que são feitas aos dados, como a limpeza de dados, combinação entre dados e eliminação da duplicação de dados. Os dados, depois de transformados, poderão trazer mais valor para o negócio, pois foi melhorada a sua qualidade. O carregamento consiste no armazenamento dos dados no DW, devidamente transformados, e a estruturação física dos dados, para o futuro acesso (Kimball & Ross, 2013).

O processo ETL tem que ser repetido periodicamente, de forma a atualizar a informação disponível até ao momento. A periodicidade dependerá da taxa de refrescamento dos dados em questão, sendo que o sistema terá que ser suficientemente eficiente para que estes refrescamentos não prejudiquem o normal funcionamento do sistema de BI. É desejável que a periodicidade do refrescamento seja cada vez menor para que a informação analisada reflita mais convenientemente a realidade (Waas et al., 2013).

Contudo, Freudenreich et al. (2013) apresentam algumas críticas ao processo de ETL, dado o processo não disponibilizar os dados rapidamente, ser efetuado o processamento de dados que não são utilizados, possuir escalabilidade limitada e não permitir facilmente a integração de novas fontes de dados. Estas críticas demonstram que há a necessidade de apostar numa solução mais eficiente para as organizações.

Para o DW se adequar à organização, é necessário que esteja modelado de forma a armazenar os dados para a análise do negócio desejada. A modelação adequada para que este seja capaz de responder às consultas é a modelação multidimensional. A modelação multidimensional do DW pode ser

modelo em estrela, floco de neve ou constelação, sendo que qualquer uma das abordagens de organização dos dados assenta em tabelas de factos e dimensões. O modelo em estrela é constituído por tabelas de factos e dimensões, que permitem fazer análises sobre diversas perspetivas. O esquema em floco de neve apresenta uma estrutura mais complexa, dado que as dimensões estão completamente normalizadas. O esquema em constelação integra múltiplas tabelas de factos que partilham dimensões comuns (Kimball & Caserta, 2004; Kimball & Ross, 2013).

Para a conceção de um DW tradicional, segue-se, habitualmente uma destas abordagens: *data-driven* e *requirement-driven*. Com a abordagem *data-driven*, há uma abordagem orientada para os dados, a preocupação com a utilização das fontes de dados, sendo criado um modelo multidimensional com base nas fontes de dados disponíveis. Nesta abordagem é minimizada a interação com os decisores da organização, o que poderá levar a que os dados para análise não correspondam às suas necessidades nem correspondam aos objetivos do negócio. Com a abordagem *requirement-driven*, a preocupação é nos objetivos de negócio resultantes das necessidades da tomada de decisão, sendo desenvolvido o modelo multidimensional com base nos objetivos do negócio. As fontes de dados são consideradas mais tarde havendo um mapeamento entre os objetivos e as fontes de dados, sendo que o pior que pode acontecer é os dados necessários não se encontrarem nas fontes de dados disponíveis, ou, por outro lado, fontes de dados com informação relevante não estarem a ser usadas. Contudo, qualquer uma destas abordagens possuem as suas vantagens, tendo emergido uma abordagem híbrida que permite que sejam recolhidas as potencialidades de cada uma das abordagens, ter que integrar e reconciliar dados com requisitos do negócio (Di Tria et al., 2014a, 2014b).

Segundo Di Tria et al. (2013), a automação do DW seria muito valiosa, tanto pelo esforço reduzido como pela simplificação do processo. Contudo, ainda há partes da construção do DW que não conseguem ser automatizadas, como é o caso do processo de integração das diferentes fontes de dados que, com a heterogeneidade, variedade e complexidade dos dados, cada vez se torna mais complexo. Seria necessário que todos os dados já possuíssem uma semântica associada, que facilitasse o seu mapeamento, que existisse uma ontologia de dados.

2.1.3. *Big Data Warehouse*

Nesta dissertação pretende-se implementar um *Data Warehouse* em contextos de *Big Data*, revelando-se necessário o conhecimento do conceito que os une, mais especificamente, *Big Data Warehouse*. A exploração do conceito e suas características permitirá a proposta e a concretização de uma arquitetura de *Big Data Warehouse* em tempo real, correspondendo aos objetivos desta dissertação.

Big Data Warehouse (BDW) surge no âmbito de BI, dado a análise de *Big Data* poder passar pela adoção de um DW que permite a extração e disponibilização de informação para a tomada de decisão (Di Tria et al., 2014b). BDW permite recolher, integrar e armazenar grandes volumes de dados, de diferentes fontes de dados, estruturados ou não estruturados, que são gerados a grande velocidade, possuindo variedade e complexidade nos dados. Além disso, é capaz de facilmente integrar uma nova fonte de dados através de um processo de modelação reduzido (Di Tria et al., 2014a).

Há quem pense no DW como a cópia dos dados de transações de forma estruturada, para poder ser feita a sua consulta e análise. Contudo, face a grandes quantidades de dados, o DW não é suficiente. Deixa de ser capaz de armazenar dados para consulta e análise, de possuir ferramentas e base de dados analíticas que mostrem ser eficientes. Começa a revelar-se limitador para responder a questões de análise de dados (Jacobs, 2009). BDW surgiu com o objetivo de alterar o tradicional DW para que se obtenham novas propriedades, novas características, as características do *Big Data*. Distingue-se das soluções tradicionais pelo facto de ser magnético, atraindo todas as fontes de dados disponíveis na organização, ser ágil, suportando a contínua e rápida evolução, e permitir o detalhe, suportando análises mais sofisticadas (Di Tria et al., 2014a). Além disso, apresenta como características o facto de estar mais orientado para análise e para a descoberta de padrões escondidos, havendo mais flexibilidade e permitindo, por exemplo, a adição de novas fontes de dados, a capacidade de escalar facilmente, a capacidade de integrar dados com variadas estruturas, a capacidade de analisar grandes volumes de dados e processamento paralelo e distribuído (Mohanty, Jagadeesh, & Srivatsa, 2013).

A implementação de BDW pode passar por duas estratégias, nomeadamente *the lift and shift strategy* e *the rip and replace strategy*. Na estratégia *the lift and shift strategy*, as capacidades do DW tradicional são aumentadas com tecnologias de *Big Data*, sendo adicionados componentes de baixo custo, rápidos e que permitem melhores resultados. Esta estratégia é desenvolvida para casos específicos, sendo a implementação muito adequada à organização e à infraestrutura possuída por esta (Clegg, 2015). Na estratégia *the rip and replace strategy*, o DW tradicional é abandonado e substituído por tecnologias de *Big Data*, apresentando uma arquitetura moderna. É procurada esta estratégia quando a solução atual possui deficiências ou não serve os objetivos do negócio. Contudo, esta estratégia põe em causa o investimento da empresa em tecnologia tradicional, sendo uma estratégia dispendiosa e disruptiva (Russom, 2016). De acordo com as intenções da organização, deverá ser escolhida a estratégia que melhor se adegue à modernização do DW.

Com a evolução de BDW, segundo Di Tria et al. (2014), surge a necessidade da conceção de uma metodologia de BDW. A conceção de uma metodologia exige o redesenho de novos princípios, como

a automação e as técnicas ágeis, que terão consequências nos requisitos do negócio e poderão resolver alguns problemas da organização. A automação permite a redução de esforços, sendo evitados erros e tarefas repetitivas, e as técnicas ágeis são importantes para lidar com as mudanças nos requisitos, permitindo o menor impacto na concepção do processo. Di Tria et al. (2014) apresentam uma metodologia de BDW, GrHyMM (*Graph-based Hybrid Multidimensional Model*), que se destaca por ser capaz de considerar fontes de dados e requisitos de forma simultânea e por usar técnicas de automação.

No contexto organizacional deverá ser seguida esta estratégia, no sentido de preparar o sistema para novas fontes de dados, para novos dados e construir um modelo que considere as fontes de dados e as necessidades do negócio, de forma a conseguir implementar um BDW. Contudo, nesta dissertação, dada a concretização do BDW em tempo real, não haverá tanto foco no contexto do negócio, mas sim na construção de uma solução capaz de ser adaptada a qualquer organização, que ofereça uma resposta rápida às consultas de dados com dados atuais. De forma a integrar grandes volumes de dados provenientes de diferentes fontes de dados serão usadas tecnologias como Hive ou bases de dados como Cassandra ou HBase⁸ para o armazenamento de dados (Alekseev et al., 2016), podendo até existir mais do que uma destas tecnologias na arquitetura definida.

2.2. Armazenamento de Dados

O volume de dados é cada vez maior, os dados movem-se com grande velocidade e não se encontram sempre com estrutura, nem um modelo de dados definido, havendo a necessidade de escolher uma alternativa para armazenamento e processamento de dados (Goss & Veeramuthu, 2013).

Para dar resposta aos desafios de *Big Data*, começaram a surgir novas formas de armazenar e processar de forma eficiente, para além das tradicionais bases de dados relacionais conhecidas. Sistemas de ficheiros distribuídos, tecnologias Hadoop, bases de dados NoSQL (*Not Only SQL*) ou novas arquiteturas de bases de dados revelam-se soluções possíveis para adoção na organização (Pokorny, 2014). Estas formas de armazenar e processar dados devem ser capazes de lidar com a heterogeneidade dos dados, sendo que os dados gerados podem ser organizados em três categorias, nomeadamente estruturados, semiestruturados e não estruturados. Os dados estruturados são aqueles que podem ser incorporados num modelo relacional, apresentando estrutura, consistência e a configuração para responder a consultas de dados. Os dados semiestruturados são dados estruturados que não tem um

⁸ <http://hbase.apache.org/>

esquema particular ou definitivo. Os dados não estruturados são complexos, sem estrutura explícita e de difícil compreensão e análise para a consulta de dados (Mathur et al., 2014; Mohanty et al., 2013).

De forma a enquadrar o armazenamento de dados, de seguida, são apresentados os principais tipos de bases de dados utilizados em contextos de *Big Data* e, ainda, as principais diferenças que o paradigma *Big Data* traz à definição dos seus modelos de dados.

2.2.1. Bases de Dados

As tradicionais bases de dados relacionais são centralizadas ou distribuídas, com uso de linguagem SQL (*Structured Query Language*) e propriedades ACID (*Atomicity, Consistency, Isolation, Durability*) de processamento de transações. Estas bases de dados têm a capacidade de suportar grandes quantidades de dados estruturados, tendo alguma capacidade de dar resposta a consultas de forma relativamente interativa e rápida (Pokorny, 2014). O armazenamento dos dados é feito em tabelas, com um modelo de dados baseado em tabelas e as relações entre elas, e apenas armazenam dados estruturados (Mohanty et al., 2013).

Contudo, face à revolução de *Big Data*, os DWs tradicionais começam a não conseguir suportar a análise aos volumes de dados devido a estes exigirem requisitos de escalabilidade e flexibilidade elevados, o que motiva o surgimento de novas abordagens capazes de o fazer, como as bases de dados NoSQL (Chevalier et al., 2015; Krishnan, 2013).

As bases de dados NoSQL pretendem simplificar os modelos de dados, são usadas para recolher e armazenar dados, possuem alta escalabilidade e estrutura de dados dinâmica, permitindo obter melhores resultados nas consultas (Di Tria et al., 2014b; Dijcks, 2013; Pokorny, 2014). Contudo, NoSQL não apresenta propriedades ACID ao contrário das bases de dados tradicionais (Özcan et al., 2014). Com a evolução do número de utilizadores e volume de dados envolvidos, torna-se necessária a utilização de princípios diferentes de ACID, tendo sido introduzidos o teorema CAP (*Consistency, Availability, Partition Tolerance*) e BASE (*basically available, soft state, eventually consistent*) (Mohanty et al., 2013).

O teorema de CAP surge associado à extensão dos sistemas de base de dados distribuídas, estando estas num *cluster* de máquinas com vários nós. Segundo este teorema, é importante garantir duas das suas três características, nomeadamente consistência, disponibilidade e tolerância a partições, como se apresenta de seguida (Krishnan, 2013; Mohanty et al., 2013; Pokorny, 2014):

- Consistência (em inglês *Consistency*) (C) - Esta característica permite garantir que os dados estão disponíveis em todos os nós do *cluster* de forma igual, podendo os dados ser escritos da mesma forma que são lidos.

- Disponibilidade (em inglês *Availability*) (A) - Esta característica permite garantir que todos os pedidos efetuados terão uma resposta, havendo leitura e escrita de dados quando desejado.
- Tolerância de partições (em inglês *Partition Tolerance*) (P) - Esta característica permite garantir que o sistema mantém o seu funcionamento face a falhas, mesmo sob casos de perda de dados ou perda de comunicação, de um ou mais nós. No caso de falha, a base de dados fica particionada.

BASE surge associado ao armazenamento de dados em NoSQL, focado na tolerância de partições e disponibilidade, caracterizando-se por *basically available* (o sistema não garante a disponibilidade), *soft state* (o sistema pode mudar ao longo do tempo) e *eventually consistent* (o sistema torna-se consistente em alguns casos) (Krishnan, 2013; Mohanty et al., 2013).

Com base nestes princípios, mas com necessidades diferentes, surgiram vários tipos de bases de dados NoSQL, nomeadamente (M. Chen et al., 2014; Hecht & Jablonski, 2011; Krishnan, 2013; Mohanty et al., 2013):

- Base de dados *key-value* – Esta base de dados caracteriza-se pela existência de uma tabela que cruza as chaves com os valores, sendo que cada chave única aponta para um determinado valor. Permite o rápido acesso aos dados, contudo os dados armazenados não possuem esquema. Um exemplo de uma base de dados deste tipo é Voldemort⁹.
- Base de dados *column-oriented* – Esta base de dados caracteriza-se por organizar os seus dados em colunas e famílias de colunas. Apresenta-se otimizada para a leitura de colunas, reduzindo o tempo de consulta dos dados pelo rápido acesso às colunas questionadas. Exemplos deste tipo de base de dados são HBase e Cassandra.
- Base de dados *document-oriented* – Esta base de dados caracteriza-se por armazenar dados como documentos, normalmente representados em JSON ou XML, relevando-se capazes de suportar dados em formas mais complexas. Apresenta flexibilidade no esquema de organização e armazenamento de dados. Exemplos deste tipo de bases de dados são Riak¹⁰, CouchDB¹¹ e MongoDB¹².
- Base de dados de grafos – Esta base de dados caracteriza-se por se basear na teoria dos grafos. Os dados são armazenados em vértices e arestas (ligações direcionadas entre vértices), sendo

⁹ <http://www.project-voldemort.com/voldemort/>

¹⁰ <http://basho.com/products/>

¹¹ <http://couchdb.apache.org/>

¹² <https://www.mongodb.com/>

que ambos possuem propriedades associadas. Um exemplo deste tipo de base de dados é Neo4J¹³.

2.2.2. Modelos de Dados

De forma a garantir o adequado armazenamento dos dados e as desejadas necessidades analíticas são utilizados modelos de dados concedendo estrutura aos dados (M. Y. Santos & Costa, 2016). Apesar de este ser um aspeto central de sistemas de informação, devido a este trabalho não apresentar como objetivo o estudo de modelos de dados, estes apenas são discutidos sucintamente.

Tradicionalmente, num DW, é utilizada a modelação multidimensional que assenta em tabelas de factos e dimensões e as relações entre estas, sendo principalmente utilizados o modelo em estrela, floco de neve ou constelação (Kimball & Caserta, 2004; Kimball & Ross, 2013), brevemente explicados na secção 2.1.2.

Em contextos de *Big Data*, com dados estruturados e não estruturados, os modelos de dados também são utilizados (M. Y. Santos & Costa, 2016). Num BDW, como referido no trabalho de M. Y. Santos & Costa (2016), podem ser utilizados modelos como o modelo colunar e tabular. O modelo de dados colunar caracteriza-se por incluir um conjunto de tabelas, apresentando esquemas de dados que não refletem as entidades relevantes, mas sim as consultas de dados que têm que ser implementadas, sendo a redundância de dados utilizada como mecanismo de otimização do tempo de resposta. O modelo de dados tabular caracteriza-se também por incluir um conjunto de tabelas, no entanto, estas tabelas encontram-se organizadas de forma a realçar o papel descritivo ou analítico de cada atributo. Existem tabelas e colunas descritivas, as que acrescentam detalhe à análise, e tabelas e atributos analíticos, que possuem dados relevantes para a análise e tomada de decisão. Estes modelos permitem facilitar a execução de consultas de dados e a gestão de grandes conjuntos de dados.

Contudo, tanto nos modelos de um DW como de um BDW, é importante que os sistemas de armazenamento sejam escaláveis. No entanto, mesmo com a inovação que se tem verificado no armazenamento de grandes volumes de dados existem períodos de latência na resposta, sendo importante a otimização dos modelos de dados, para mais facilmente se ter acesso aos dados necessários a responder às consultas (Marz & Warren, 2015). As tecnologias do ecossistema do Hadoop e a base de dados Cassandra são exemplos disso, apresentando limitações em casos em que se pretende respostas rápidas. Desta forma, por exemplo, no caso da tecnologia Hive, o acesso aos dados pode ser

¹³ <https://neo4j.com/>

otimizado com a criação de *buckets* e partições que permitem a organização dos dados e a criação de ficheiros de menor dimensão, com apenas parte dos dados, para que o acesso aos dados seja mais eficiente (Costa, Costa, & Santos, 2017; Rutherglen, Wampler, & Capriolo, 2012).

Sendo o foco desta dissertação o acesso aos dados, é importante que o armazenamento permita uma análise dos dados, estando os modelos de dados preparados para otimizar o processamento de consultas. Assim, a tecnologia, base de dados e/ou modelo de dados selecionados devem ser adequados para armazenar dados históricos e dados que são recebidos em tempo real, de forma ser possível concretizar um ambiente de recolha, processamento, armazenamento e análise de dados que satisfaz os requisitos estabelecidos por cada organização.

2.3. Tempo Real em *Big Data Warehouses*

Atualmente as organizações cada vez mais se preocupam com BI, com a análise e ferramentas para reportar os dados, mas, com o crescimento do volume de dados dos próximos anos, a integração dos dados para análise será cada vez mais complexa. As infraestruturas tradicionais tornam-se limitadas, sendo necessária a introdução de soluções baseadas em conceitos de *Big Data*. Os utilizadores requerem acesso aos dados em tempo real, estando assim lançado o desafio de conseguir lidar com a complexidade e volume de dados, considerando que os dados estão constantemente a mudar e por isso é necessário análises e resultados rápidos (M. Chen et al., 2014; Goss & Veeramuthu, 2013).

De forma a se conseguir responder aos requisitos em tempo real, é necessário que sejam conhecidas as principais características de tempo real e trabalhos desenvolvidos neste sentido, como se apresenta nesta secção.

2.3.1. Características de Tempo Real

O acesso aos dados em tempo real revela-se uma oportunidade pois permite que seja conhecido o valor dos dados que estão constantemente a ser gerados e ainda que se consiga reagir rapidamente na tomada de decisão com mais informação do negócio. Assim, como exemplo, na produção de um determinado produto, com dados de testes de controlo do produto e informação de sensores de monitorização atempada será possível a deteção de problemas, permitindo uma tomada de decisão com informação mais atualizada do negócio. Contudo, o que acontece em muitas organizações é que os seus processos não estão preparados para o tempo real pois os dados apenas são refrescados, por exemplo, ao fim de algumas horas, não sendo assim possível a tomada de decisão em tempo real (Goss & Veeramuthu, 2013; R. J. Santos, Bernardino, & Vieira, 2011). Mas, mesmo sendo uma mais valia para

a organização, é necessário que cada negócio avalie a sua necessidade de tempo real, pois esta não é necessariamente igual para qualquer área de negócio.

De forma a acompanhar as alterações, é necessário que as tecnologias de BI e os DWs propaguem os dados pelas suas fontes e o acesso aos dados tenha o mínimo de latência, algo que não acontece nos DWs tradicionais. É necessário o conceito de latência zero, que permite que as organizações tenham disponível informação o mais rápido possível para a tomada de decisão, informação das necessidades dos clientes e do negócio (Bruckner, List, & Schiefer, 2002). Contudo, para isso, é necessário que as organizações possuam soluções capazes de responder ao requisito temporal, um obstáculo à mudança, dados os custos envolvidos na aquisição de tecnologia. Além disso, muitas organizações não conhecem as potencialidades destas soluções e não há muitos profissionais com *know-how*.

Para aqueles que precisam de tomar decisões em tempo real, de acordo com os seus requisitos temporais, a disponibilidade é um aspeto essencial porque se a recolha e análise do negócio for mais tardia, há uma redução do seu desempenho, da sua capacidade de tomada de decisão atempada. A disponibilidade é a característica que permite a melhoria do desempenho nas operações e serviços do negócio e o aumento dos ganhos (Mohamed & Al-Jaroodi, 2014).

Tendo-se tornando o tempo real um aspeto tão importante, é também necessário ter-se em atenção como se vai gerir os dados que advêm de fontes tão diferentes e apresentar uma análise destes de forma rápida, pois à medida que o tempo passa, os dados têm menor valor (Lebdaoui, Orhanou, & Elhajji, 2014). A integração de dados é realizada pelo processo de ETL/ELT, sendo que, em tempo real, este será contínuo. Em organizações como Ebay e Amazon, onde as transações são todas feitas *online*, é importante que haja acesso aos dados de forma muito rápida (X. Liu, Iftikhar, & Xie, 2014).

A dificuldade em analisar os dados em tempo real prende-se com o facto de o volume, velocidade e variedade de dados ser tanta que se torne difícil o processamento e análise dos dados. A arquitetura em tempo real, a propor nesta dissertação, deverá ser capaz de integrar algumas preocupações e desafios, no âmbito de *Big Data*, nomeadamente:

- A análise dos dados deverá disponibilizar informação útil para o negócio, existindo um processo de tratamento para evitar uma análise errada aos dados;
- O processamento de dados em tempo real deverá ser rápido, de forma a conseguir cumprir com o tempo desejado, dado que a rapidez no processamento maximiza a eficiência do negócio;
- Não hajam conflitos entre processamento de dados, consulta de dados e extração de dados de outras fontes, para que não hajam grandes tempos de espera;

- A análise de dados deverá contemplar dados históricos e dados em tempo real, para que as decisões possam ser tomadas com informação atualizada do negócio.

O conceito *near real-time surge* associado ao tempo real pretendido, no qual se afirma que o tempo real será aquele que permitir o mais rápido possível fornecer informação relevante para o negócio. No contexto desta dissertação, procura-se implementar um BDW que responda à necessidade em tempo real considerando uma latência mínima de processamento de dados. Cada consulta de dados deverá demorar o mínimo possível, pelo que só depois de ser conhecido o comportamento das tecnologias é que este poderá ser definido.

2.3.2. Trabalhos Relacionados

O crescente interesse dos utilizadores nas tecnologias de BI leva a que haja uma preocupação para que a tomada de decisão possa ser feita com informação mais atualizada. Isto despoletou um interesse dos utilizadores no acesso de dados em tempo real, havendo uma preocupação no desenvolvimento de soluções que o possibilitem. Vários trabalhos têm sido concretizados nesse sentido numa abordagem mais tradicional e, mais recentemente, com o uso de *Big Data*. De forma a serem conhecidos os trabalhos que têm sido desenvolvidos na área, primeiro foram explorados os trabalhos propostos numa abordagem mais tradicional e depois numa abordagem com uso de *Big Data*, de forma a ser compreendida a evolução e as limitações dos trabalhos desenvolvidos.

a) Abordagem Tradicional

Numa abordagem tradicional, surgiu o *Real-Time Data Warehouse* (RTDW), que se considera uma extensão do DW tradicional, pois permite capturar a mudança rápida nos dados, assim como o processo de análise em tempo real. O tempo de acesso aos dados sem atrasos de processamento é o seu grande desafio (Farooq & Sarwar, 2010; Li & Mao, 2015).

No processo tradicional de processamento, os dados são extraídos das suas fontes através de uma ferramenta de ETL, passando por uma fase de transformação e limpeza, para posteriormente serem inseridos num DW (Lebdaoui et al., 2014; Vaisman & Zimányi, 2012). De acordo com o período da sua extração (diário, mensal, anual, entre outros), os dados são refrescados, permitindo aos utilizadores ter dados atuais e atualizados com informação útil para o processo de tomada de decisão, havendo sincronismo da informação atualizada com a que está no DW. A necessidade de uma análise de dados com mais brevidade motivou a redução da latência existente no processo de ETL, pois é esta que impossibilita a disponibilização de serviços em tempo real. A redução da latência no processamento de

dados revela-se o requisito chave e poderá representar uma enorme vantagem competitiva nas organizações (Freudenreich et al., 2013; Vaisman & Zimányi, 2012).

Vários autores têm trabalhado neste sentido, na descoberta de possíveis soluções que permitam reduzir a latência para que possa haver acesso a dados em tempo real. Normalmente as soluções são mais conceptuais e não apresentam um modelo tecnológico específico de como o fluxo de dados poderia ser processado rapidamente. Segundo Vaisman & Zimányi (2012), podem ser seguidas estratégias como o aumento da frequência do processo ETL, a eliminação da necessidade de os dados serem agregados e a introdução de dados em tabelas de tempo real. O trabalho de Freudenreich et al. (2013) defende que o processo de ETL pode ser acelerado separando o processo e efetuando paralelismo no processamento. A abordagem definida por estes autores permite o carregamento dos dados diretamente para o DW, sem limpeza ou transformação. Além disso, a criação de vistas acedidas pelas interrogações aos dados permitiria minimizar o refrescamento.

A abordagem de processamento em tempo real de Kimball & Ross (2013) é motivada pela necessidade de tomada de decisão ativa, pela necessidade de serem conhecidos os vários estados que os dados podem tomar. A sua solução é a construção de uma partição em tempo real, pelo que a partição em tempo real terá a mesma estrutura das tabelas do DW histórico, mas terá disponível apenas os dados de um dia, de forma a garantir a rápida resposta a consultas. Os dados das tabelas em tempo real precisam de ser atualizados com mais frequência, sendo que, ao fim do dia normalmente, os dados são movidos para as tabelas de dados históricos. Para a análise dos dados, são combinados os dados estáticos e em tempo real numa vista. Contudo, o aspeto mais crítico é o facto de a qualidade de dados poder ficar comprometida. Uma abordagem semelhante a esta é a metodologia de carregamento contínuo de dados de R. J. Santos & Bernardino (2008), que foi apresentada no sentido de assegurar que a consulta ao DW apresenta um bom desempenho, com os dados históricos e de tempo real. Esta metodologia optou pela criação de tabelas temporárias com a informação mais atualizada, de forma a permitir que seja possível a tomada de decisão com informações atualizadas. O RTDW será obtido se se conseguir a integração dos dados no DW de forma eficiente, sem haver grande atraso, e sem o desempenho ficar prejudicado.

Li & Mao (2015) desenvolveram um trabalho que consiste na utilização de uma *framework* de ETL em tempo real que processa dados históricos e em tempo real, como se apresenta na Figura 7. A principal motivação para este desenvolvimento consiste nas dificuldades que as ferramentas tradicionais encontram em carregar e consultar, em simultâneo, dados em tempo real. Surge a necessidade de

augmentar a capacidade de processamento das bases de dados tornando-as mais escaláveis, separar os dados em tempo real e juntar os dados históricos e em tempo real para análise.

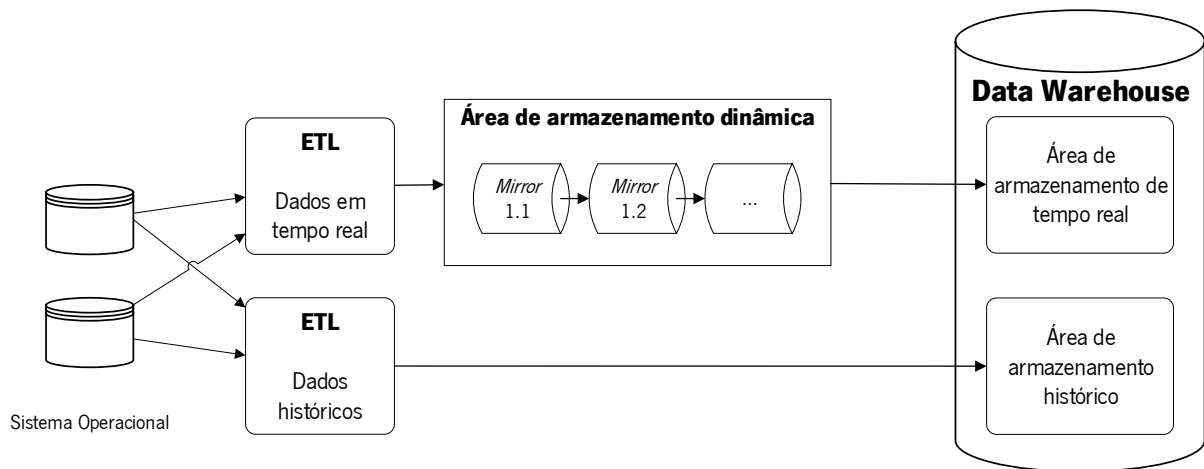


Figura 7 - Framework RTDW. Adaptada de Li & Mao (2015)

Este trabalho pretende superar as dificuldades das ferramentas tradicionais sendo, para isso, criado um processo ETL para dados históricos e outro para dados em tempo real. Os dados em tempo real serão combinados na área de armazenamento dinâmica, que armazena os dados em ficheiros e possui um gestor de ficheiros que permite a gestão e manutenção para o seu bom funcionamento, havendo replicação de dados. Os dados são armazenados num DW, existindo uma área de dados históricos e outra de dados em tempo real, onde são feitas as consultas. Esta abordagem evolui em parte de Mao, Min, Wang, Jia, & Jie (2014).

A abordagem de Golab & Johnson (2014) também afirma que faz sentido dividir o sistema de tempo real daquele que possui os dados históricos, mas que esta divisão ainda não está completamente clara, pois os dados antigos permitem fornecer contexto para os novos dados. O facto de se ter dados em tempo real e dados históricos exige técnicas ao nível da organização dos dados, como, por exemplo, o particionamento horizontal dos dados de acordo com o atributo temporal. Além disso, os dados em tempo real exigem técnicas de controlo de forma a garantir que as consultas não são bloqueadas por atualizações de dados. Ao nível da distribuição do armazenamento é preciso ter em atenção como este é feito, para que se possa melhorar o desempenho das consultas pois, se existem várias máquinas, a junção de dados entre diferentes tabelas poderá ser ineficiente.

b) Abordagem com *Big Data*

Numa abordagem com *Big Data*, as tecnologias estão preparadas para o grande volume de dados, a diversidade e a elevada velocidade, mas a baixa latência nas consultas de dados continua a ser um desafio. A análise em tempo real entende-se pela análise aos dados depois da integração dos

mesmos, dados esses que poderão estar armazenados ou poderão ser acedidos por *streaming*. O objetivo é que, seja qual for a consulta desejada aos dados, o resultado possa ser apresentado em segundos (X. Liu et al., 2014).

A *Lambda Architecture* proposta por Marz & Warren (2015) é uma arquitetura desenhada para lidar com grandes conjunto de dados, sejam eles dados históricos ou dados recolhidos em tempo real. Esta arquitetura procura encontrar um balanço entre a latência, a taxa de transferência e a tolerância a falhas. De forma responder às consultas de dados, esta apresenta três camadas, nomeadamente *batch layer*, *speed layer* e *servicing layer*, nas quais são computadas as consultas de dados num conjunto de vistas (X. Liu et al., 2014; Marz & Warren, 2015). Na Figura 8 apresenta-se esta arquitetura.

A *batch layer* é a camada onde são armazenados os dados e computadas as vistas dos dados já recolhidos, sendo usado um sistema de processamento distribuído, Hadoop, em *batch*, capaz de lidar com grandes quantidades de dados. Este processo está constantemente a ser repetido de forma a incluir os novos dados. A *speed layer* é a camada onde são processados os novos dados, os dados em tempo real, com Storm¹⁴ ou Spark¹⁵. Estes dados fluem quase imediatamente para uma base de dados NoSQL e, por nem sempre estarem completos, são depois substituídos quando processados em *batch* pelo Hadoop. A *servicing layer* é a camada onde são carregadas as vistas e efetuadas as consultas aos dados, havendo uma junção de resultados históricos e resultados em tempo real, numa base de dados NoSQL, como Cassandra ou HBase (X. Liu et al., 2014; Marz & Warren, 2015). A análise de dados é feita com base nas vistas criadas.

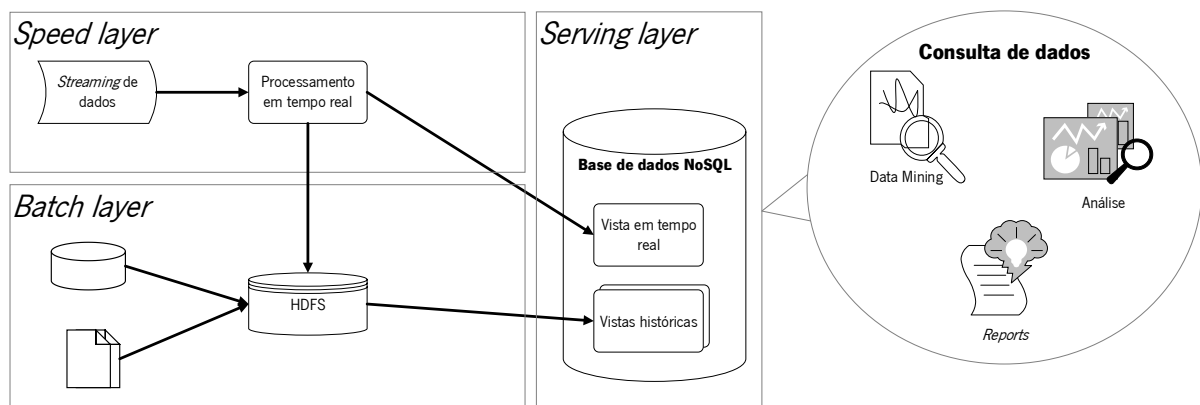


Figura 8 - *Lambda Architecture*. Adaptada de X. Liu et al. (2014)

O facto de serem utilizadas vistas permite que, quando se pretende conhecer o valor de uma determinada consulta, apenas se questione a vista sem ter a necessidade de estar a filtrar todos os

¹⁴ <http://storm.apache.org/>

¹⁵ <http://spark.apache.org/>

dados. O problema é que a criação de vistas é sempre uma operação com alguma latência e assim que a criação termina já existem novos dados, estando assim a vista desatualizada. Com a *batch layer* e *servicing layer*, propriedades de sistemas de *Big Data*, como a robustez, a tolerância a falhas e a escalabilidade estão garantidas, contudo continua a existir latência entre o tempo de computação da vista e o carregamento desta. Esta latência leva a que os dados que surgem, enquanto está a ser feita a computação das vistas, não sejam considerados. Para garantir o tempo real, a *speed layer* necessita de garantir que os novos dados estão nas vistas de tempo real, de forma rápida, para responder às consultas a tempo. Para que se possa obter a menor latência possível, em vez de os dados novos serem olhados todos de uma vez, sempre que é recebido um novo dado, as vistas de tempo real são atualizadas, em vez de serem novamente computadas as vistas históricas. Isto permite que, quando são feitas consultas nas vistas históricas e de tempo real, haja junção dos resultados e os resultados representarão também os dados em tempo real (Marz & Warren, 2015).

Segundo G. J. Chen et al. (2016), uma outra solução para processamento de dados em tempo real é a desenvolvida pelo Facebook. Esta solução em tempo real fornece informação valiosa à medida que os próprios eventos estão a acontecer. Na Figura 9 apresenta-se a representação desta solução, onde se encontra o fluxo de dados entre os vários sistemas que esta engloba, nomeadamente as fontes de dados do Facebook, os sistemas de *streaming* e o armazenamento de dados.

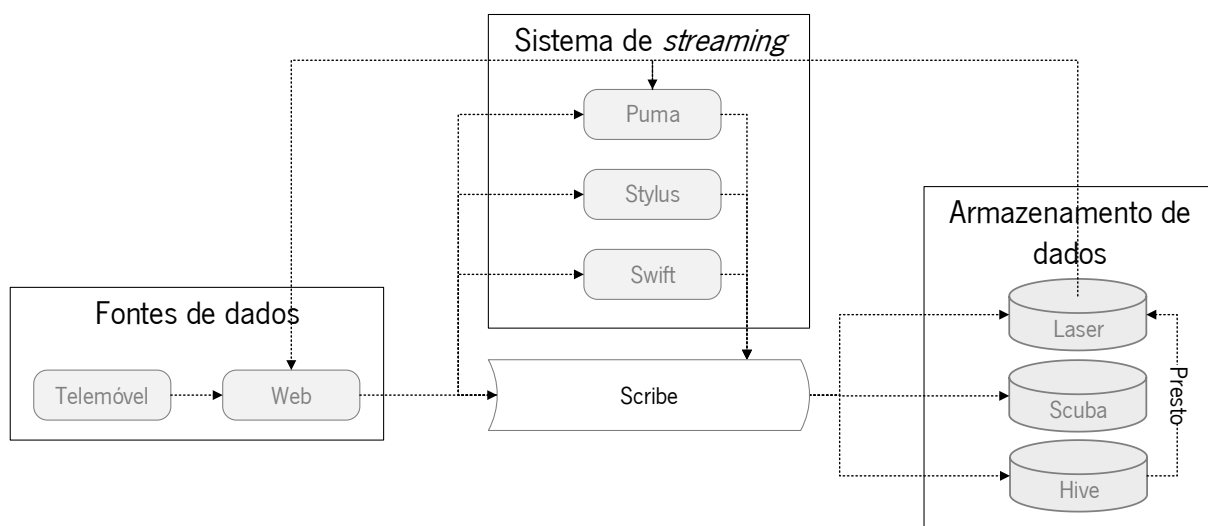


Figura 9 - Arquitetura do Facebook de tempo real. Adaptada de G. J. Chen et al. (2016)

Nesta arquitetura, os eventos que vão acontecendo nas fontes de dados, nomeadamente no telemóvel e na web, são registados e enviados pela tecnologia de distribuição de mensagens, Scribe¹⁶,

¹⁶ Tecnologia em desenvolvimento, sem URL oficial.

para os sistemas de *streaming* e para o armazenamento de dados. Com esta tecnologia de transporte de mensagens, os dados são recolhidos, agregados e entregues em tempo real. Os sistemas de *streaming*, nomeadamente Puma¹⁶, Stylus¹⁶ e Swift¹⁶ leem e escrevem os dados para o Scribe. Estes sistemas são responsáveis por processar os dados garantindo o bom desempenho e tolerância a falhas. O armazenamento de dados é feito no Laser¹⁶, Scuba¹⁶ e Hive, dependendo das consultas de dados desejada, sendo que estas tecnologias armazenam os dados e respondem de forma rápida com os resultados. A tecnologia Presto também entra neste fluxo, visto ser através desta tecnologia que se faz consultas ao DW no Hive.

A análise dos trabalhos desenvolvidos por outros autores representa um bom ponto de partida para a proposta da arquitetura de BDW em tempo real desta dissertação. Os trabalhos aqui apresentados expõem os principais problemas, desafios e dificuldades de uma arquitetura em tempo real, nas quais a latência pode traduzir-se numa análise menos correta. Contudo, mesmo sendo as soluções encontradas mais de foro conceptual e pouco tecnológico, foram encontradas algumas soluções que conjugam os dois aspetos, ajudando na concretização de uma solução de BDW em tempo real.

De uma forma geral estes trabalhos respondem aos seus objetivos, contudo, pensando naquilo que se pretende desenvolver nesta dissertação, a máxima preocupação é com os dados que fluem em tempo real, para que o processamento destes seja o mais rápido possível e que a análise permita resultados interessantes e em tempo útil. A arquitetura utilizada será uma evolução da *Lamba Architecture* considerando as suas três principais camadas que permitem o acesso aos dados em tempo real, tendo em conta a abordagem de processamento em tempo real de Kimball & Ross (2013).

Este capítulo de revisão de literatura demonstra a importância da temática desta dissertação e a vantagem que poderá trazer para as organizações, na tomada de decisão. Terminado este capítulo, é possível afirmar que a exploração dos conceitos chave representa uma base de conhecimento importante para entender o contexto em que esta dissertação se insere. Além disso, a análise dos requisitos de tempo real e trabalhos desenvolvidos mostram a relevância e a motivação do estudo nesta área. Com esta base de informações importantes sobre o tema, será possível o alcance do objetivo pretendido, a concretização de um DW em contextos de *Big Data* em tempo real. Para isso, terá que ser definida uma arquitetura, sendo necessária a exploração das tecnologias que poderão ser utilizadas para atingir o objetivo desta dissertação.

3. TECNOLOGIAS PARA A CONCRETIZAÇÃO DE *BIG DATA WAREHOUSES*

Ao longo dos últimos anos, o volume de dados recolhido e analisado para BI tem vindo a aumentar rapidamente, fazendo com que as soluções tradicionais de armazenamento possam não satisfazer os requisitos do negócio (Thusoo et al., 2010). Começaram, por isso, a surgir ferramentas capazes de dar resposta às necessidades de armazenamento e processamento de grandes quantidades de dados.

Neste capítulo são exploradas as principais tecnologias de armazenamento e processamento de dados, que poderão ser incluídas na arquitetura a propor e implementar nesta dissertação.

3.1. Ecossistema do Hadoop

Apache Hadoop é uma *framework* que permite o processamento distribuído de grandes conjuntos de dados num *cluster* de computadores (Hadoop, 2017). Possui um ecossistema de variados componentes, cada um deles com uma responsabilidade diferente, capazes de armazenar e processar grandes quantidades de dados (Fan, Zhao, & Wang, 2015; Thusoo et al., 2010). Foi desenhado para o processamento de dados não estruturados, mas é capaz de lidar com dados estruturados e com as funcionalidades de SQL permite a resposta a consultas muito rapidamente, comparativamente aos sistemas relacionais (Y. Chen et al., 2014). De forma a explorar uma das plataformas de *Big Data* mais utilizadas, nesta secção é explorado o ecossistema do Hadoop e as suas principais características.

O Hadoop caracteriza-se pela sua flexibilidade, escalabilidade, eficiência e capacidade em ambiente distribuído (Fan et al., 2015; Thusoo et al., 2010). Foi concebido com a capacidade de escalar de pequenos servidores para grandes conjuntos de máquinas, garantindo a elevada disponibilidade, e é responsável por executar um conjunto de tarefas, por distribuir as tarefas em pequenas tarefas, e por armazenar os dados, de forma paralela e distribuída (Hadoop, 2017). Além disso, enquadra-se num sistema MAD, por se revelar: magnético (em inglês *magnetism*) pois é capaz de atrair todos os tipos de dados, ágil (em inglês *agility*) pois é capaz de adaptar o seu mecanismo de armazenamento de acordo com a evolução das fontes de dados, e permitir o detalhe (em inglês de *depth*) pois é capaz de suportar uma análise aos dados muito mais detalhada do que era possível com ferramentas tradicionais, considerando os vários tipos de dados (Cuzzocrea et al., 2011).

O ambiente de computação do Hadoop assenta num sistema de ficheiros distribuídos, o *Hadoop Distributed File System* (HDFS). Este sistema de ficheiros foi desenhado especificamente para grandes quantidades de dados e inspirado no trabalho da Google, no *Google File System* (GFS), e no paradigma

de programação MapReduce, que usa as tarefas de *Map* e *Reduce* para a manipulação de dados num *cluster* de computadores que trabalham em paralelo (P. Zikopoulos et al., 2011).

O HDFS é responsável pela gestão e armazenamento de dados de forma eficiente e permite um acesso rápido aos dados. Possui a capacidade de filtrar grandes conjuntos de dados e produzir resultados, sendo o seu processamento em *batch*. Este sistema de ficheiros distribuído distingue-se dos já existentes pela sua elevada capacidade de tolerância a falhas, pela grande quantidade de dados que suporta e por poder ser usado em ambientes de *hardware* de baixo custo. Além disso, caracteriza-se também pela sua redundância, dado o facto de armazenar cópias dos dados nos nós do *cluster*, havendo armazenamento repetido em muitos locais do *cluster*. Permite, também, que facilmente possa haver recuperação a falhas. A sua arquitetura é *master-slave*, como representado na Figura 10, apresentando um *NameNode* (o *master*) e vários *DataNodes* (os *slaves/ workers*). O *NameNode* é responsável por gerir o sistema de metadados, o sistema de ficheiros e regular o acesso aos clientes. Os *DataNodes*, um por nó do *cluster*, são responsáveis pela gestão e armazenamento dos dados. Por exemplo, quando a aplicação cliente lê ou escreve dados no HDFS, primeiro tem que haver comunicação com o *NameNode* para obter a localização de onde poderá escrever ou ler no *DataNode* (Cuzzocrea et al., 2011; Hadoop, 2017; Katal et al., 2013; Krishnan, 2013; P. Zikopoulos et al., 2011).

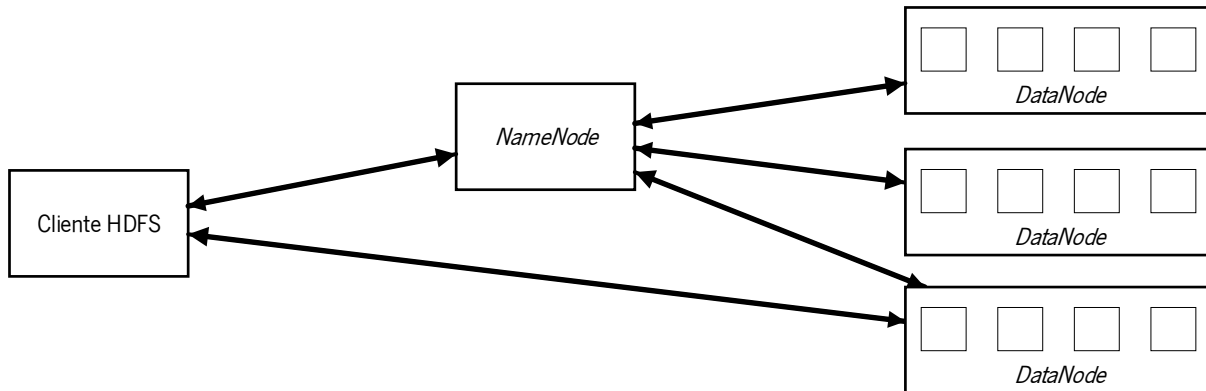


Figura 10 - Arquitetura do HDFS. Adaptado de Krishnan (2013)

O paradigma de computação usado pelo Hadoop é o MapReduce, que permite o processamento paralelo de grandes conjuntos de dados, e é baseado no YARN, que é o responsável pela gestão de tarefas e de recursos do *cluster*. MapReduce possui uma grande rapidez no processamento, é disponível e suporta diferentes tipos de dados, fazendo uso de uma infraestrutura distribuída para o processamento paralelo em *batch*. Neste paradigma, o trabalho é dividido na fase *Map*, que é responsável por dividir as tarefas computacionais em tarefas menores, sendo as pequenas tarefas executadas em paralelo nos diferentes nós do *cluster*, associando uma chave a cada um dos valores, cada linha de dados, e *Reduce*, que é responsável por combinar as respostas dos vários nós do *cluster*, combinando os valores que

partilham a mesma chave (Cuzzocrea et al., 2011; Hadoop, 2017; Krishnan, 2013; X. Liu et al., 2014; Z. Liu et al., 2013; Mohanty et al., 2013; Rutherglen et al., 2012; P. Zikopoulos et al., 2011).

Porém, o ambiente Hadoop engloba muitos outros componentes, responsáveis pela coordenação, monitorização, segurança e acesso, e integração de dados, alguns deles bastante relevantes para o contexto de *Big Data*. Cada um dos componentes apresenta um papel distinto, mas importante, alguns dos quais (Hadoop, 2017; X. Liu et al., 2014; Mohanty et al., 2013; Rutherglen et al., 2012):

- Ambari é a interface web que permite a gestão e monitorização do *cluster*, suportando HDFS, MapReduce, Hive, HBase, ZooKeeper, entre outros.
- Cassandra é uma base de dados escalável e de elevada disponibilidade, sem pontos de falha.
- HBase é uma base de dados distribuída, escalável e *column-oriented*, onde as colunas são organizadas em famílias de colunas. Suporta o armazenamento de dados em grandes tabelas, permitindo a realização de consultas e transações.
- Hive é uma infraestrutura de DW que permite consultas dos dados e a gestão dos dados. As suas tabelas são uma abstração do HDFS, o que torna mais rápido o carregamento de dados.
- Spark foi desenhado com o objetivo de permitir uma análise aos dados mais rápida, possuindo um modelo de execução que permite a otimização, suportando computação em memória. Spark Streaming permite o processamento em *streaming*, de forma semelhante ao Spark.
- ZooKeeper é responsável por coordenar os serviços garantindo o elevado desempenho das aplicações distribuídas e permite que haja uma grande confiança na coordenação distribuída.

3.2. Repositório de Dados

Para a concretização de BDW em tempo real, é necessário o armazenamento de dados no qual são utilizadas tecnologias, não sendo necessariamente a mesma tecnologia para armazenar dados históricos e os dados em tempo real. Tecnologias como Hive, HBase, Cassandra e Kudu¹⁷ são exploradas de seguida, para depois ser estudada a possibilidade de estas serem usadas na solução desta dissertação.

- **Hive**

Hive é uma ferramenta de BI utilizada para a execução de consultas ao armazenamento em HDFS, revelando-se uma ferramenta essencial no ecossistema Hadoop. Permite obter uma componente

¹⁷ <https://kudu.apache.org/>

análítica dos dados processados, materializados e armazenados no Hadoop (Cuzzocrea et al., 2011), facilitando a consulta e gestão de grandes quantidades de dados (Y. Chen et al., 2014). A linguagem usada pelo Hive para efetuar consultas é o HiveQL (*Hive Query Language*) bastante semelhante ao tão conhecido SQL. Hive é mais adequado para DW, quando os dados não mudam rapidamente e quando não são necessárias respostas rápidas, pois a análise de dados é sobre dados relativamente estáticos, sendo mantidos para *reports*, *dashboards* e análise de tendências (Du, 2015; Rutherglen et al., 2012; Thusoo et al., 2009, 2010). Porém, o facto de o Hadoop ser um sistema em *batch* leva a que as consultas no Hive tenham latência. É utilizado o MapReduce como mecanismo para execução de consultas, contudo com o surgimento do Tez¹⁸, um mecanismo de execução flexível para processar dados em *batch*, o Tez substitui o MapReduce em soluções como o Hive. Esta mudança permitiu a redução da latência das consultas (Hadoop, 2017).

Esta tecnologia não foi desenhada no sentido de serem eliminados e atualizados registos, assim como não foi desenhado para substituir o sistema transacional, sendo que, para a concretização de transações, considera-se a utilização de bases de dados NoSQL (Rutherglen et al., 2012). Contudo, o Hive já permite a realização de atualizações aos dados com a utilização de tabelas que suportam transações ACID (Hadoop, 2017).

A estrutura dos dados em Hive é em tabelas, colunas e linhas, uma estrutura bastante comum, sendo os dados armazenados nas tabelas e as tabelas armazenadas diretamente no HDFS. Tendo em vista a otimização de resultados nas consultas, as tabelas poderão ter partições ou *buckets*, estando as partições de uma tabela armazenadas numa subdiretoria da tabela e os *buckets* armazenados num ficheiro na diretoria da partição ou tabela, dependendo se a tabela se encontra particionada ou não (Du, 2015; Rutherglen et al., 2012; Thusoo et al., 2009, 2010). Segundo Thusoo et al. (2009, 2010), para a criação de partições, aquando a criação da tabela, terão que ser identificadas as colunas que se pretende particionar, sendo que as colunas particionadas não são as colunas da tabela, mas derivam delas. Para a criação de *buckets*, aquando a criação da tabela, terão que ser identificados o número de *buckets* necessários e as colunas pelas quais os *buckets* vão ser criados. Assim, em qualquer um dos casos, o desempenho será melhorado e o tempo de processamento das consultas diminuirá.

¹⁸ <http://tez.apache.org/>

- **HBase**

HBase é uma base de dados orientada a colunas que usa o HDFS como armazenamento, sendo classificada como base de dados NoSQL. Esta foi desenhada tendo em vista o bom desempenho e elevada disponibilidade e não suporta a linguagem SQL, pois não é uma base de dados relacional. Porém, o HBase suporta transações ACID, permitindo a realização de transações (Cattell, 2011; Krishnan, 2013; White, 2012; P. Zikopoulos et al., 2011).

As suas tabelas contêm colunas e linhas, muito semelhante às bases de dados tradicionais, e ainda uma chave primária. Os atributos das tabelas são agrupados em famílias de colunas, sendo especificadas as colunas que são incluídas, podendo ser adicionadas outras mais tarde. Além disso, esta base de dados faz particionamento e distribuição de dados de forma transparente e as alterações aos dados ficam guardadas em memória e periodicamente são armazenadas em disco (Cattell, 2011; Krishnan, 2013; White, 2012; P. Zikopoulos et al., 2011).

Esta tecnologia adequa-se quando existem requisitos de tempo real, em ler e escrever grandes conjuntos de dados, pois esta consegue ler e escrever sem precisar de procurar e processar todo o conjunto de dados (Cattell, 2011; Krishnan, 2013; White, 2012; P. Zikopoulos et al., 2011).

- **Cassandra**

Cassandra é uma base de dados distribuída caracterizada pela sua elevada disponibilidade e escalabilidade, sendo classificada como base de dados NoSQL. As alterações aos dados são guardadas em memória e periodicamente armazenadas em disco, sendo o armazenamento de todos os seus dados em disco. Além disso, há replicação de dados e particionamento nos vários nós do *cluster* e a deteção e recuperação de falhas é feita automaticamente, não existindo ponto de falha único, pois todos têm a mesma função. Ao contrário do HBase, esta base de dados suporta transações ACID mas de forma muito limitada (Cattell, 2011; M. Chen et al., 2014; Krishnan, 2013).

É orientada a colunas e possui um esquema bastante flexível, o que permite adicionar ou eliminar algum atributo de forma rápida. As suas tabelas contêm linhas, colunas, famílias de colunas e super colunas, cada um destes identificado através de uma chave-valor (Cattell, 2011; M. Chen et al., 2014; Krishnan, 2013).

- **Kudu**

Kudu é um sistema de armazenamento de rápido acesso, com baixa latência. O desenvolvimento deste sistema de armazenamento foi motivado pela lacuna entre o desempenho analítico no HDFS e a baixa latência de acesso aos dados do HBase e Cassandra, o que permite que esta solução afirme ser

mais completa que as restantes. A análise rápida aos dados, a baixa latência de atualizações e a consistência do seu desempenho são as suas principais características (Lipcon et al., 2015).

Esta tecnologia armazena os dados em tabelas como as bases de dados relacionais, existindo uma chave primária para cada uma das linhas, contudo os dados estão organizados orientados a colunas e não a linhas. Cada uma das linhas pode ser lida, atualizada e eliminada, caracterizando-se pela rápida leitura de dados. De forma a armazenar grandes volumes de dados, as tabelas são separadas em unidades mais pequenas, *tablets*, sofrendo uma partição dos dados. Além disso, para que haja maior disponibilidade dos dados, os dados são replicados pelas várias máquinas do *cluster* (Kudu, 2017; Lipcon et al., 2015).

3.3. Processamento de Dados em Tempo Real

Com vista à concretização de um BDW em tempo real, revela-se importante a exploração de tecnologias que permitam obter o tempo real, que permitam que não haja latência. Num contexto de *Big Data*, o Hadoop oferece várias soluções que permitem o armazenamento e processamento de grandes quantidades de dados de forma bastante eficiente. Contudo, num contexto de tempo real, o ecossistema do Hadoop, por processar dados em *batch*, oferece, de forma geral, alguma latência, apresentando limitações para o processamento em tempo real. Neste sentido, várias soluções começam a emergir de forma a dar resposta às necessidades de tempo real, tendo sido introduzidos componentes no ambiente Hadoop para a redução da latência e processamento de dados em *streaming* (X. Liu et al., 2014).

Para a concretização do tempo real desejado, terá que existir recolha de dados em *streaming* e processamento de dados, em poucos segundos. São necessárias um conjunto de tecnologias, que se encontram representadas na Figura 11, nomeadamente tecnologias de recolha de dados em *streaming*, tecnologias de processamento de dados em tempo real, tecnologias de armazenamento de dados (apresentadas na secção anterior, nomeadamente 3.2) e tecnologias de análise de dados, de possível integração na arquitetura desta dissertação.

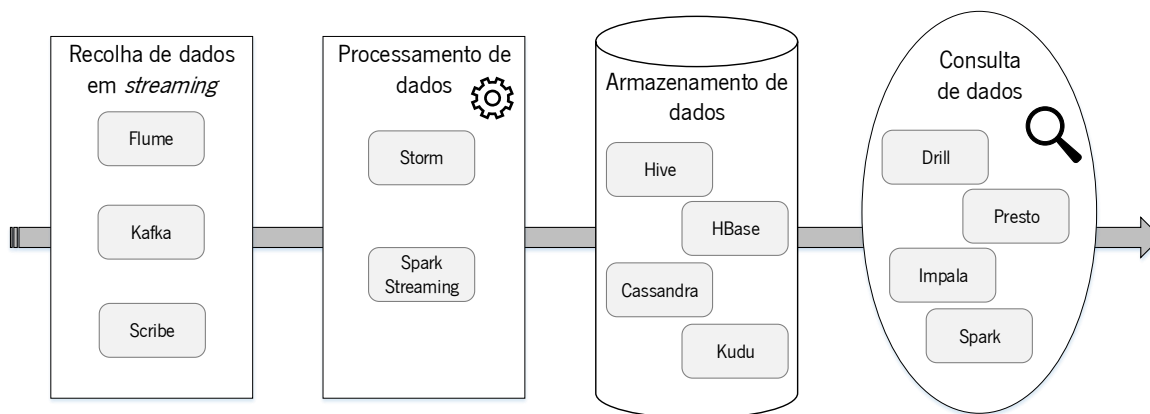


Figura 11 - Tecnologias de possível integração na arquitetura

Como tecnologia de recolha de dados em *streaming*, podem ser utilizadas as seguintes:

- **Flume**¹⁹ - Flume é um sistema confiável, distribuído e disponível, usado para recolher, agregar e transferir grandes quantidades de dados para o HDFS de forma eficiente. Os dados podem ser recolhidos de várias fontes, sendo que uma unidade de dados corresponde a um evento. Na sua topologia usa vários agentes, no qual cada agente possui três componentes, nomeadamente: *source* (responsável por recolher os dados em *streaming* como eventos), *sink* (responsável por escrever os eventos, tipicamente para o HDFS) e *channel* (que faz a ligação entre os outros dois componentes) (X. Liu et al., 2014). Este sistema foi desenhado para que pudessem ser inseridas grandes quantidades de dados com base nos eventos dos dados (White, 2012).
- **Kafka** - Kafka é um sistema para publicar e subscrever mensagens em tempo real, capaz de recolher e entregar grandes volumes de dados a uma baixa latência. Esta tecnologia é distribuída, escalável e possui uma taxa de transferência elevada, sendo uma solução usada pelo LinkedIn. Permite manter as várias mensagens organizadas, associadas a uma categoria, de acordo com a ordem com que foram publicadas. Fornece também uma grande durabilidade e tolerância a falhas, estando armazenadas as mensagens no disco, replicadas pelo *cluster*. Kafka possui aplicações *producers* (que publicam mensagens) e *consumers* (que consomem as mensagens) e poderá ser usado para processamento, monitorização e carregamento em tempo real. É a solução ideal para situações em que os utilizadores precisam de processar os dados e analisá-los em tempo real (Kreps, Narkhede, & Rao, 2011; X. Liu et al., 2014; Philip Chen & Zhang, 2014).

¹⁹ <http://flume.apache.org/>

- Scribe - Scribe é uma tecnologia de distribuição de mensagens, responsável por recolher, agregar e entregar os dados, com pequenos segundos de latência, nos sistemas em *batch* e em tempo real. É responsável por agregar mensagens e é robusto à rede e às falhas dos nós do *cluster*. Os dados são organizados em categorias e são armazenados no HDFS para ser garantida a sua durabilidade. Esta tecnologia é usada no Facebook (G. J. Chen et al., 2016; Kreps et al., 2011; X. Liu et al., 2014).

Como tecnologias de processamento de dados em tempo real podem ser utilizadas as seguintes:

- Spark Streaming - Spark Streaming é uma tecnologia que afirma ser escalável, disponível e tolerante a falhas. Esta tecnologia atua num *cluster* que possui um conjunto de nós, executando tarefas paralelas pelos nós no processamento de dados. Os dados em *streaming* são recolhidos e divididos em lotes mais pequenos para serem processados pelo Spark, para gerar o fluxo final de dados. Esta tecnologia é capaz de ler dados do HDFS, Flume, Kafka e Twitter, processá-los e armazená-los em sistemas de ficheiros e bases de dados (Fan et al., 2015; X. Liu et al., 2014; Spark, 2017).
- Storm - Storm é uma das *frameworks* mais populares para o processamento em tempo real, com baixa latência e mecanismos que garantem que tudo é processado. Possui um sistema de processamento distribuído e com tolerância a falhas e possui a capacidade de processar grandes volumes de dados a uma grande velocidade, em memória (dos Anjos et al., 2015; X. Liu et al., 2014; Philip Chen & Zhang, 2014). Segundo X. Liu et al. (2014), atualmente, muitas empresas usam o Storm combinado com o HBase, criando assim uma arquitetura em tempo real capaz de suportar o processamento de dados em *streaming*. O Storm é usado para integrar os dados em *streaming* continuamente e o HBase é usado como armazenamento e para a execução de consultas.

Como tecnologias de processamento de dados em tempo real podem ser utilizadas as seguintes:

- Drill - Drill é um sistema para análise interativa de dados, desenhado para responder a consultas com baixa latência. Caracteriza-se pela sua flexibilidade a nível de linguagem de *querying*, formatos dos dados e fontes de dados. É capaz de processar dados em *batch* ou em *streaming*, sendo que o armazenamento no HDFS e o paradigma MapReduce são usados para a análise em *batch* (Hausenblas & Nadeau, 2013).

- Impala - Impala é um sistema de análise em tempo real, que é escalável e interativo na execução de consultas e afirma possuir baixa latência. Esta tecnologia combina a tão conhecida linguagem SQL com multi-utilizadores, caracterizando-se pela sua flexibilidade e escalabilidade. Impala tem sido mais usado para análise exploratória dos dados, representando um mecanismo de execução de consultas distribuído que funciona por cima do HDFS, integrado no ecossistema do Hadoop (Kornacker et al., 2015; X. Liu et al., 2014).
- Presto - Presto é um sistema para análise de dados de forma interativa, podendo combinar na análise de dados uma variedade de fontes de dados. A característica deste sistema, que cativa mais a atenção, é o facto de este apresentar tempos de resposta muito pequenos, sendo possível uma análise aos dados de forma muito rápida (Presto, 2017).
- Spark - Spark é uma plataforma *Big Data* que permite resolver a falha existente entre o aumento do volume de dados e o processamento, com bom desempenho. Esta tecnologia afirma ser rápida, fácil de usar, que poderá ser executada em ambientes muito diferentes e a sua linguagem combina SQL, o que é uma vantagem dado a linguagem SQL ser tão conhecida. O *cluster* Spark possui um conjunto de nós, que são escaláveis de acordo com o volume de dados, responsáveis pelo carregamento dos dados em memória, por alocar as várias tarefas computacionais e evitar a replicação de dados. O Spark processa em *batch* e em *streaming*. Além disso, este afirma que a sua velocidade pode ser 100 vezes mais rápida que o MapReduce em memória e 10 vezes mais rápida em disco (Fan et al., 2015; X. Liu et al., 2014; Spark, 2017).

Contudo, qualquer que seja a solução, esta deverá ser fácil de usar, apresentar um elevado desempenho, tolerância a falhas e elevada escalabilidade. Uma solução em tempo real poderá fornecer informação valiosa à medida que os próprios eventos estão a acontecer (G. J. Chen et al., 2016). O processamento e análise em tempo real permite ao utilizador uma ação imediata em resposta aos eventos dos dados (X. Liu et al., 2014).

Este capítulo de exploração de tecnologias permite o conhecimento daquelas que poderão ser as tecnologias que virão a integrar a solução desenvolvida. Cada uma das tecnologias apresentará um papel importante na concretização de BDW em tempo real, sendo que estas tecnologias poderão ou não integrar o ecossistema do Hadoop. Revela-se necessária a existência de uma ou mais tecnologias para recolha de dados em *streaming*, processamento de dados, armazenamento de dados e análise de dados. A existência de tecnologias que permitem a concretização do tempo real pretendido motiva para a investigação no sentido de encontrar aquelas que oferecem o melhor desempenho.

4. *BIG DATA WAREHOUSES* EM TEMPO REAL

Para a concretização de um BDW em tempo real, é necessária a exploração dos conceitos envolvidos, tecnologias de possível aplicação e a proposta da arquitetura que permita a concretização deste tipo de repositórios. Para que a solução final seja concretizada, as tecnologias que compõe a arquitetura deverão ser as que obtêm o melhor desempenho.

Dado o facto de esta dissertação se enquadrar no projeto de investigação “P30 - Business Intelligence Platform for Data Integration” entre a Universidade do Minho e a BOSCH, algumas das escolhas tecnológicas e decisões tomadas no decorrer da dissertação justificam-se pelo trabalho a ser desenvolvido no projeto de investigação. Esta dissertação pretende dar contributos que possam ser integrados futuramente no projeto de investigação. Depois de o trabalho experimental testar os vários cenários de teste e da arquitetura estar finalizada, esta poderá ser implementada e validada num contexto organizacional real, neste caso, na BOSCH.

Neste sentido, o trabalho experimental desta dissertação começa pela definição da estrutura dos fluxos de dados e a proposta de arquitetura. São definidos os principais aspetos a ser discutidos na concretização, bem como o conjunto de dados, os cenários de teste e as tecnologias a utilizar, para que a solução corresponda ao esperado e que permita a análise em tempo real. A concretização da arquitetura analisa um conjunto de variáveis de forma a encontrar a melhor solução para o problema em questão.

4.1. Fluxos de Dados

A arquitetura a propor nesta dissertação para a concretização de um sistema de *Data Warehousing* em tempo real para contextos de *Big Data* deve integrar fontes de dados em tempo real (*streaming* de dados) e dados históricos (dados em *batch*), dados esses que são recolhidos e integrados na análise de dados. Os dados recolhidos em *streaming* são armazenados num repositório que permite o acesso rápido aos dados, para que o tempo real desejado seja verificado. Estes dados serão movidos periodicamente para o armazenamento de dados históricos, onde se encontram armazenados os dados que são recebidos por processos em *batch*. Este processo de armazenamento de dados implica o processamento e a preparação dos dados para que estes possam ser analisados no futuro. Estes fluxos, apresentados na Figura 12, permitem que a proposta de infraestrutura seja concretizada.

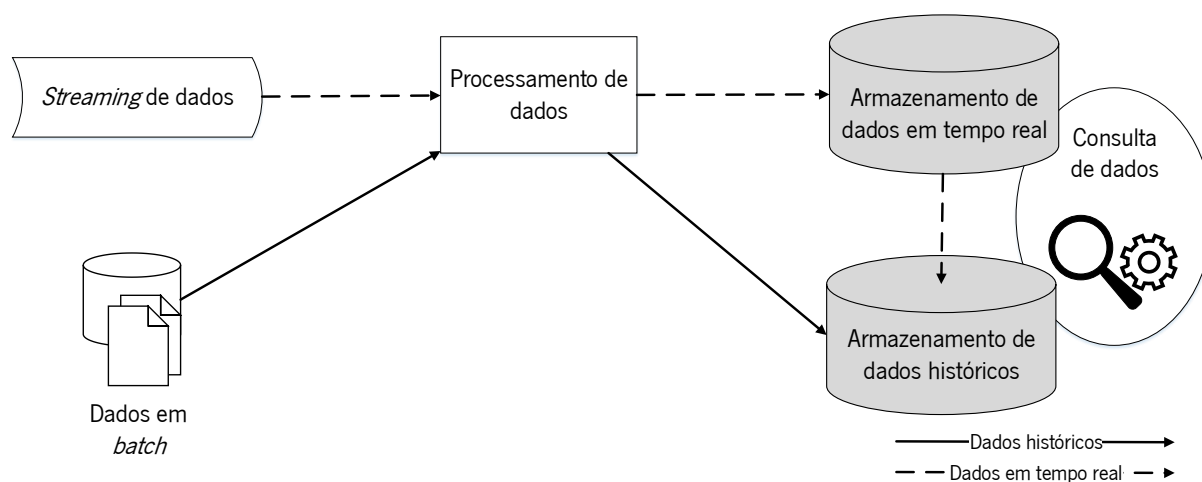


Figura 12 - Fluxo de dados em *streaming* e em *batch*

A criação de duas áreas de armazenamento permite que sejam obtidas as características de armazenamento desejadas, dado que se pretende que os dados em tempo real estejam armazenados num repositório que permita a rápida leitura, escrita e atualização, e os dados históricos estejam armazenados num repositório que permita uma análise detalhada e com bom desempenho. Desta forma, a análise detalhada aos dados históricos e a baixa latência de acesso aos dados em tempo real será garantida. Além disso, um aspeto importante é que a recolha de dados seja efetuada através de uma tecnologia que permita a transferência de dados de forma muito rápida, permitindo que os dados possam ser analisados pouco tempo depois de serem gerados. Quando é feita a consulta de dados, são lidos os dados em tempo real e, se necessário, os dados históricos, podendo existir uma junção dos seus resultados. No entanto, poderão existir consultas a apenas um dos repositórios.

Esta proposta para a integração dos fluxos de dados recolhe contributos das várias abordagens referidas anteriormente, na secção 2.3.2, integrando os diversos trabalhos e propondo um processo que se distingue por ser capaz de rapidamente recolher e armazenar dados de forma a que a análise de dados possa ser feita em poucos segundos, com os dados dos eventos que acabaram de acontecer. No caso desta dissertação, como se enquadra no projeto de investigação “P30 - Business Intelligence Platform for Data Integration” desenvolvido entre a Universidade do Minho e a BOSCH, o contexto de tempo real é cerca de 5 a 10 segundos. Este *near real-time* é o tempo desde a recolha dos dados das suas fontes de dados até à consulta destes, considerando o processamento, bem como a transformação dos dados.

4.2. Arquitetura para o Processamento de Dados

Devido a esta dissertação se enquadrar no projeto de investigação “P30 - Business Intelligence Platform for Data Integration” desenvolvido entre a Universidade do Minho e a BOSCH, a proposta de arquitetura desta dissertação segue as opções de desenho consideradas no projeto de investigação. Neste sentido, a arquitetura definida no projeto de investigação é adaptada aos objetivos desta dissertação, incluindo os requisitos que permitem obter o tempo real, tendo o seu desenho como base a metodologia utilizada no projeto de investigação. Na Figura 13 apresenta-se a respetiva arquitetura que possui como principais componentes as fontes de dados, o processo de ETL, o armazenamento de dados, a análise de dados e a segurança, administração e monitorização do sistema. Entre estes componentes são estabelecidos relacionamentos, de acordo com os dados que fluem entre eles. Para os componentes implementados são destacadas as tecnologias utilizadas, pelo que as mesmas serão apresentadas posteriormente nesta secção.

Nesta arquitetura, as fontes de dados são principalmente dados em *batch*, através de ficheiros e bases de dados, e dados em *streaming*, através de uma ou mais aplicações que disponibilizem este tipo de dados. Além destas, podem também ser consideradas, num contexto organizacional, ERP, emails e *web services*. O processo de ETL²⁰ consiste na extração dos dados, na transformação e no armazenamento destes no DW, sendo que os dados em tempo real e históricos são armazenados em repositórios diferentes. Para a análise de dados podem ser utilizados vários mecanismos, havendo a preocupação em que a consulta de dados seja feita rapidamente. A segurança, administração e monitorização não serão o foco desta dissertação, contudo, é importante incluí-las dado o contexto em que esta arquitetura se insere.

²⁰ Como referido anteriormente, na secção 2.1.1, o processo ETL em *Big Data* é normalmente referido como processo ELT. Contudo, de uma forma genérica, quando se fala deste processo na componente experimental desta dissertação refere-se ao processo ETL.

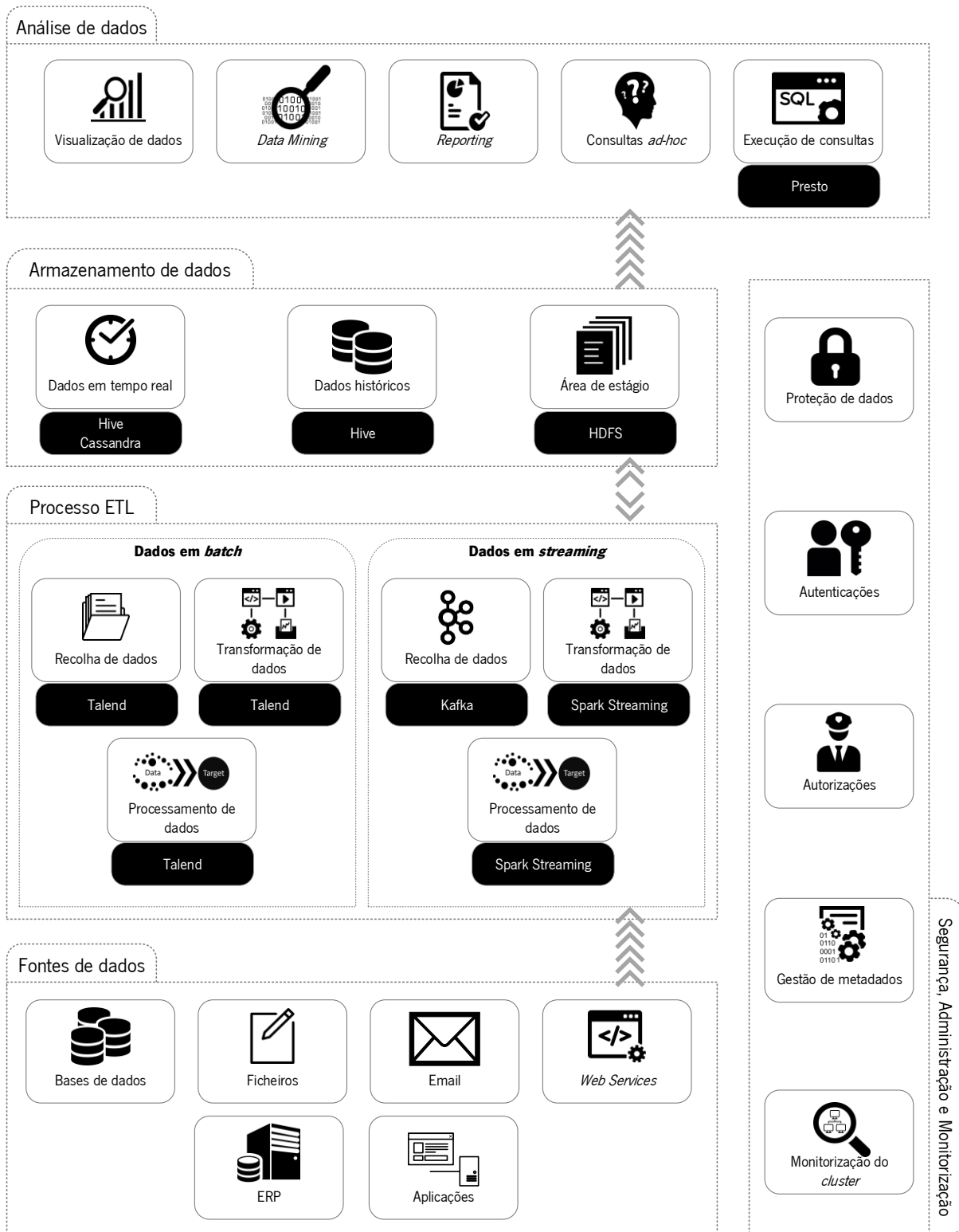


Figura 13 Arquitetura de processamento de dados. Adaptada de M. Y. Santos et al (2017)

Na instanciação da arquitetura encontram-se definidas as tecnologias que suportam todo o processo desde a recolha até à análise de dados. A tecnologia de recolha de dados em *streaming* utilizada neste trabalho é o Kafka, pela sua capacidade de enviar mensagens em tempo real, o que permite a recolha dos dados em tempo real que são gerados por uma aplicação. Estes dados, assim que

recolhidos, são processados recorrendo à tecnologia Spark Streaming, sendo definido o modelo de dados, os tipos de atributos e efetuada a limpeza, integração e transformação dos dados. Este processo deverá ocorrer rapidamente para que os dados possam ser armazenados na área de dados em tempo real, para consulta. A utilização do Spark Streaming permite o processamento dos dados em tempo real em tarefas paralelas pelos vários nós do *cluster*. As tecnologias para o armazenamento de dados em tempo real podem ser o Hive ou Cassandra, pelo que serão efetuados testes de forma a averiguar qual destas se adequa melhor ao contexto deste trabalho, enquanto que para o armazenamento de dados históricos é utilizado Hive. A tecnologia Hive destaca-se pela sua capacidade analítica, bem como na consulta e gestão de grandes quantidades de dados, e o Cassandra pela baixa latência de acesso aos dados. Periodicamente, em qualquer umas das tecnologias, os dados do armazenamento em tempo real são movidos para o armazenamento de dados históricos, o que poderá ocorrer, por exemplo, uma vez por dia.

A extração dos dados em *batch* ocorre através do Talend²¹, funcionando também como ferramenta de integração de dados. É efetuado o processo de limpeza, tratamento e integração e é, ainda, definido o modelo de dados e o tipo dos atributos. Quando concluído este processo, os dados são armazenados no HDFS, como área de estágio, e depois movidos para a área de dados históricos no Hive, para futura consulta de dados.

A consulta de dados poderá ser realizada através das tecnologias Presto, Spark e Drill, contudo, com base nos resultados obtidos no *benchmark* M. Y. Santos, Costa, et al. (2017), apenas será utilizada a tecnologia Presto, pois é a que apresenta mais rapidez na resposta às consultas de dados. As consultas podem ser efetuadas ao armazenamento de dados em tempo real e de dados históricos, havendo junção dos resultados, ou consulta a apenas uma das áreas de armazenamento.

As tecnologias para o processo de ETL, armazenamento de dados e análise de dados foram selecionadas com base na exploração das tecnologias para a concretização de um BDW e com base no trabalho desenvolvido no projeto de investigação. Para alguns dos casos a tecnologia mais adequada ainda não foi selecionada, pelo que irá depender dos resultados dos testes efetuados em cada contexto, sendo selecionada a que obtiver o melhor desempenho.

²¹ <https://www.talend.com/>

4.3. Concretização da Arquitetura

Apresentada a arquitetura de processamento de dados utilizada e definidos os fluxos dos dados para a concretização de um BDW em tempo real, é iniciada a concretização e avaliação da arquitetura proposta com as tecnologias instanciadas nesta, procurando garantir que a solução final apresente o melhor desempenho. De acordo com os objetivos e resultados pretendidos nesta dissertação, o trabalho experimental deverá permitir a concretização do BDW em tempo real, sendo este implementado e validado, para permitir que sejam retiradas conclusões.

Para a concretização e avaliação da arquitetura proposta é utilizado um caso de demonstração que utiliza como fonte de dados, em tempo real, dados que são obtidos da rede social Twitter. O objetivo da utilização deste conjunto de dados é a utilização de dados de um contexto real. De acordo com a velocidade e o volume de dados definido, são criados alguns cenários de recolha de dados avaliando o desempenho do sistema nos vários contextos. Cada um dos *tweets* é assim extraído através de uma aplicação, processado e armazenado para mais tarde ser analisado. No processamento dos dados é recebida cada uma das mensagens extraída e é efetuado um conjunto de transformações aos dados. Assim que os dados se encontram transformados, são armazenados na área de armazenamento de dados em tempo real. Para isso, são criados dois cenários de armazenamento em tempo real, um com tecnologia Hive e outro com a tecnologia Cassandra, pelo que serão efetuados testes de forma a perceber qual se revela mais adequado neste contexto. Juntamente com esta questão, é estudada a melhor forma de organização dos dados nas respetivas tecnologias bem como o seu comportamento com o aumento do volume de dados.

Além disso, é utilizada como fonte de dados os dados em *batch* através de ficheiros suplementares que poderão adicionar valor aos restantes dados. Estes dados são carregados, transformados e armazenados diretamente no armazenamento de dados históricos, para onde são movidos também, com uma determinada periodicidade, os dados em tempo real armazenados. A criação de cenários de teste tem como objetivo a avaliação das tecnologias mais adequadas para o propósito em questão. Os cenários de teste integram as tecnologias para armazenamento de dados, sejam dados em tempo real ou dados históricos, a tecnologia para recolha de dados, a tecnologia para processamento de dados e a tecnologia para a análise de dados, estando esquematizados na Figura 14.

Em qualquer um dos casos, os dados encontram-se desnormalizados numa só tabela, seja qual for a tecnologia de armazenamento de dados. O objetivo de os dados se encontrarem desnormalizados é que não haja a necessidade da junção de dados entre tabelas. Além disso, o facto de estes dados não estarem agregados permite obter mais detalhe na análise aos dados e que seja possível responder a

qualquer questão, porque as questões que se desejam responder agora podem não ser as questões que se pretendem ver respondidas no futuro. Por esta razão, são armazenados dados detalhados e não dados agregados.

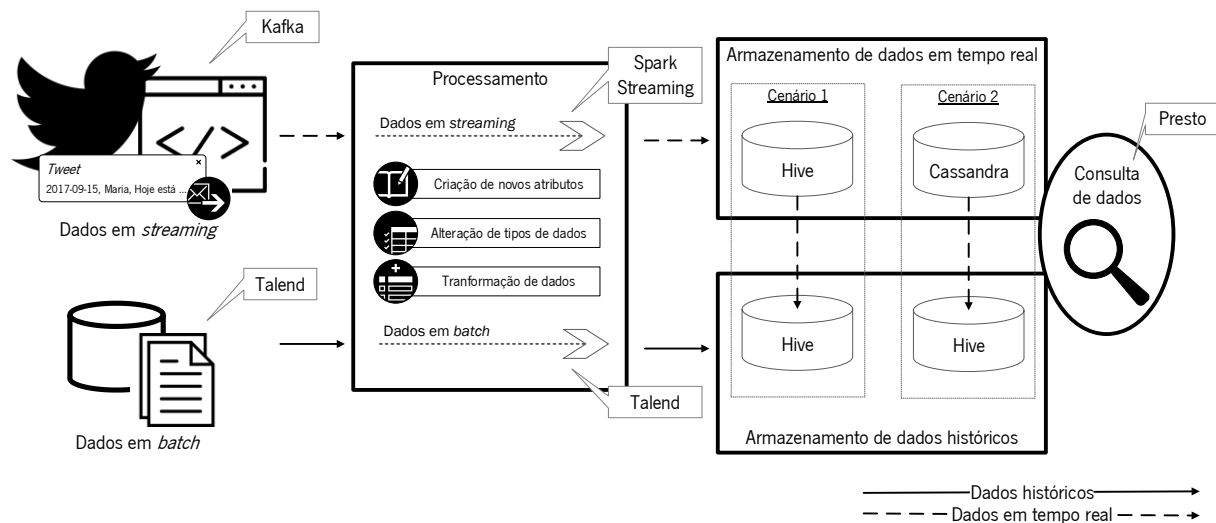


Figura 14 - Cenários de tecnologias para testes

O desempenho das tecnologias de armazenamento de dados selecionadas será avaliado, sendo avaliados os tempos que demoram a efetuar as suas tarefas. Com este *benchmark* de diferentes tecnologias, diferentes volumes e diferentes velocidades, será possível conhecer o desempenho do sistema. De seguida, apresenta-se a infraestrutura onde serão realizados todos os testes desta dissertação e é depois detalhada a componente experimental deste trabalho.

4.3.1. Infraestrutura Utilizada

A infraestrutura utilizada para a realização deste trabalho consistiu num *cluster* Hadoop, com a versão 2.6.0 do Hortonworks Data Platform (HDP), com cinco nós, um deles como *NameNode* HDFS e gestor de recursos Yarn e os quatro restantes como *DataNode* HDFS e nós gestores do Yarn. A nível de *hardware*, cada nó inclui: Intel core i5, quad core, 3.1GHz-3.3 GHz; 32 GB de 1333MHz DDR3 *Random Access Memory* (RAM), com 24GB disponível para processamento de consultas; Samsung 850 EVO 500GB *Solid State Drive* (SSD) com até 540 MB/s de velocidade de leitura e até 520 MB/s de velocidade de escrita; e 1 *gigabit Ethernet*. O sistema operativo instalado em cada nó é *CentOS 7* com sistema de ficheiros XFS. Além disso, o Cassandra está instalado nos quatro nós onde se encontra o *DataNode* HDFS e o Presto está em todos os nós, pelo que o gestor é onde se encontra o *NameNode* HDFS. De referir que algumas das tecnologias utilizadas foram escolhidas tendo em conta que foi utilizado um *cluster* Hadoop, que oferece um conjunto base de tecnologias.

4.3.2. Componente Experimental

O processo desde a recolha de dados até ao armazenamento possui dois pontos de início nomeadamente os dados que são recolhidos do Twitter e os dados que são carregados em *batch* através de ficheiros suplementares. Os dados extraídos do Twitter possuem um conjunto de atributos acerca de cada um dos *tweets*, seja relativamente ao próprio *tweet* (como a sua data de criação, o seu texto, a língua em que foi escrito, o dispositivo usado), ao utilizador que escreveu o *tweet*, entre outros. Estes dados são recolhidos através de uma aplicação em Java, que é responsável por captar os *tweets* que são produzidos a cada segundo, tendo sido criada uma *stream* de *tweets*. Os dados a ser captados representam informação variada, não existindo qualquer filtro na informação extraída, pelo que foram selecionados alguns atributos representando uma amostra do contexto inserido. Esta aplicação capta de forma rápida os dados, na sua forma original, pelo que a forma como estes dados são recolhidos obriga posteriormente ao enriquecimento dos mesmos. O número de *tweets* a ser extraído, por segundo, é geralmente constante. Para o envio das mensagens de Twitter extraídas é criado um tópico Kafka, previamente, para onde são enviadas cada uma das mensagens extraídas. O tópico Kafka permitirá o envio das mensagens e depois a receção destas para processamento no Spark Streaming, funcionando como um canal de comunicação. Cada uma das mensagens extraída do Twitter é então enviada utilizando a tecnologia Kafka para ser futuramente processada, como se apresenta na Figura 15.

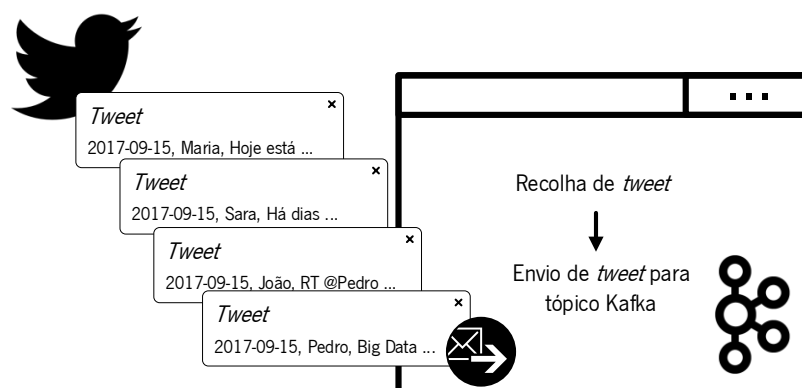


Figura 15 - Recolha de dados do Twitter

À medida que os *tweets* são enviados, são recebidos pelo consumidor de mensagens Kafka, para serem efetuadas as devidas transformações aos atributos de cada *tweet*, bem como criados novos atributos e alterados os tipos de dados, preparando-os para serem armazenados. Este processo decorre através de uma aplicação em Spark Streaming, na qual é criado um *micro-batch* com alguns *tweets*, pelo que assim que os dados se encontram transformados, são posteriormente inseridos no Cassandra ou no Hive. Para o armazenamento desses dados é criada uma tabela na respetiva tecnologia. O que distingue a aplicação Spark Streaming nos diferentes cenários é o local de armazenamento, sendo usado

um conector adequado e propriedades específicas de acordo com a tecnologia de armazenamento. Na Figura 16 encontram-se esquematizados estes passos do processo.

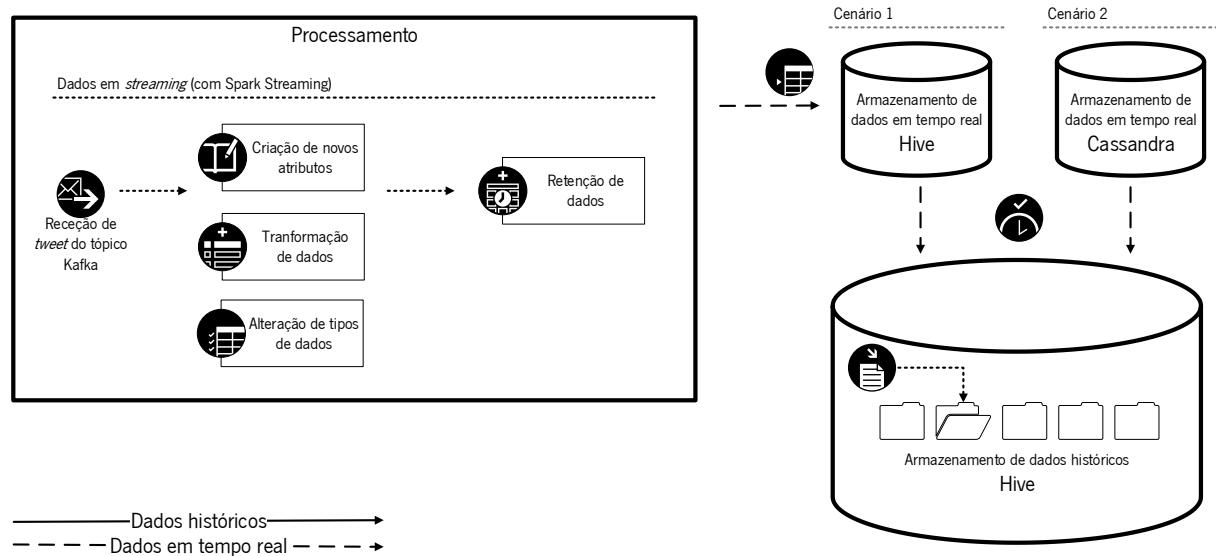


Figura 16 - Processamento e Armazenamento de dados em tempo real

De referir que seria de esperar que os dados que fluem em tempo real, antes de serem armazenados no Hive ou no Cassandra, fossem armazenados numa área de estágio, nomeadamente no HDFS, como cópia dos dados recolhidos. Contudo, devido aos testes efetuados, verificou-se que esta operação atrasaria a inserção dos dados no armazenamento de dados em tempo real. Desta forma, o armazenamento dos dados, que são recolhidos em tempo real, no HDFS não foi considerado.

Para os dados em *batch*, é carregado um ficheiro com dados, efetuadas as devidas transformações, pelo que são também criados novos atributos e alterados os tipos de dados. Assim, são preparados os dados para serem inseridos numa tabela Hive, diretamente no armazenamento de dados históricos, como se apresenta na Figura 17. De realçar que para os dados serem inseridos numa tabela Hive são primeiramente inseridos numa área de estágio, no HDFS, e depois movidos para a respetiva tabela.

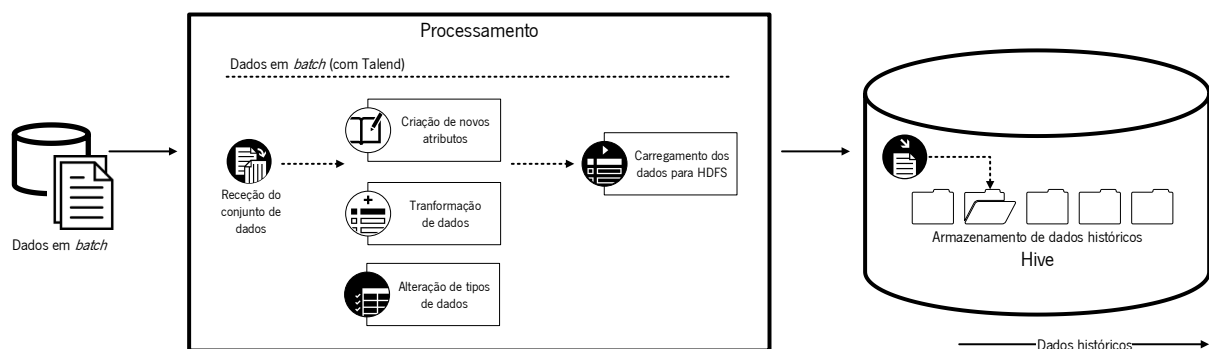


Figura 17 - Recolha, Processamento e Armazenamento de dados em *batch*

Este processo de ETL para os dados em *batch* recorre ao Talend como ferramenta de integração, não estando estabelecida nenhuma periodicidade específica para este processo ocorrer, sucedendo quando desejado. Na Figura 18 apresenta-se esquematizado este processo na ferramenta Talend. Este processo pode ser parametrizado atendendo às necessidades analíticas.



Figura 18 - Recolha, Processamento e Armazenamento de dados em *batch* no Talend

Este capítulo permite integrar a revisão de literatura efetuada e a exploração das tecnologias na concretização de *Big Data Warehouse* em tempo real desta dissertação. Cada um dos componentes do BDW em tempo real possui um papel importante de forma a garantir que a recolha, processamento e armazenamento de dados seja possível, havendo uma procura em responder aos requisitos em tempo real. O estudo do comportamento destas tecnologias motiva o desenvolvimento da componente experimental, com um caso de demonstração que permite analisar o desempenho desta infraestrutura proposta.

5. CASO DE DEMONSTRAÇÃO

De forma a implementar o caso de demonstração apresentado anteriormente para a componente experimental deste trabalho, foram analisados vários aspetos e questões tecnológicas que surgem neste ambiente, de forma a permitir retirar um conjunto de conclusões bem como recomendações a ter em conta na concretização de BDW em tempo real. A preocupação deste trabalho foca-se então em quatro principais pontos:

- Compreender o comportamento da tecnologia Kafka, bem como a influência do tamanho das mensagens enviadas pelo Kafka;
- Compreender o comportamento da tecnologia Spark Streaming, para o processamento e tratamento de dados;
- Compreender o comportamento da tecnologia Cassandra com o armazenamento de dados em tempo real à medida que aumenta o volume de dados;
- Compreender o comportamento da tecnologia Hive como armazenamento de dados em tempo real e com dados históricos;
- Avaliar o desempenho das consultas numa arquitetura de BDW em tempo real, com a tecnologia Presto.

Estes aspetos guiam o desenvolvimento da dissertação, nomeadamente todo o processo desde a recolha de dados, processamento e armazenamento. Para ser compreendido o comportamento das tecnologias foram feitos vários testes, criando-se assim cenários diferentes para conhecer as tecnologias e retirar conclusões da sua concretização. De seguida são debatidos cada um destes aspetos, pelo que o foco do caso de demonstração é apenas nos dados recolhidos em *streaming*, dado que é através destes que poderá ser concretizado o tempo real desejado e obtidas respostas atempadas às consultas de dados com dados atuais. Destaca-se que o foco não é colocado no contexto do negócio, que influenciaria requisitos funcionais e não funcionais da solução a implementar, mas numa solução genérica que pode ser adaptada a qualquer organização ou contexto de negócio.

5.1. Recolha de Dados

O conjunto de dados recolhidos do Twitter é vasto, tendo existido uma seleção de alguns atributos, representando o contexto inserido. Para a recolha destes dados foi desenvolvida uma aplicação em Java, na qual foi criada uma *stream* de *tweets*. Esta *stream* de *tweets* funciona como um conector

ao Twitter, estando constantemente a extrair *tweets*. Apesar de esta dissertação estar relacionada com o projeto de investigação P30, optou-se pela utilização deste conjunto de dados primeiramente, pois representa uma fonte de dados em *streaming*, pelo que na instanciação desta arquitetura no projeto de investigação, as fontes de dados serão substituídas por, por exemplo, dados reais das máquinas de produção na BOSCH.

O objetivo da utilização de dados do Twitter é a utilização de dados reais, com velocidade, variedade e volume de dados real. Contudo, devido ao conector ao Twitter apresentar limites na taxa de transferência, surgiu a necessidade de replicar os *tweets* que são extraídos, de forma a garantir uma maior velocidade. Desta forma, o cenário criado implica a produção de uma média de cerca de 1500 a 2500 mensagens por segundo.

Cada uma das mensagens recolhida do Twitter é enviada para um tópico Kafka, previamente criado, como canal de comunicação para a troca de mensagens entre o Kafka e o Spark Streaming. Da forma como a integração do Kafka com o Spark Streaming está desenvolvida, as mensagens podem ser enviadas através de um *batch* de várias mensagens ou pode ser feito *streaming* em que é enviada uma mensagem de cada vez. Comparando o desempenho das mensagens que são enviadas por *batch* e as mensagens a ser enviadas por *streaming*, as mensagens em *batch* acabam por ser enviadas mais rápido, porque é um conjunto de mensagens que são enviadas de uma só vez, contudo estas já não serão tão atualizadas, como referido na documentação do Spark Streaming²². Dado o contexto desta dissertação, o que faz sentido é o envio de mensagens à medida que estão disponíveis, em *streaming*, de forma a ser garantido que cada mensagem é enviada e recebida imediatamente depois de ser recolhida. Além disso, o Kafka assenta em dois aspetos, a latência e a taxa de transferência, procurando-se o equilíbrio entre eles. Neste trabalho, a preocupação recai em ambos os aspetos, tanto com o tempo que a mensagem demora desde que é enviada até quando é consumida pelo Spark Streaming, nomeadamente a latência, assim como a sua taxa de transferência, ou seja, quantas mensagens são enviadas por segundo, como referido na documentação do Kafka²³.

No sentido de se perceber o comportamento do Kafka e tentar tornar este processo o mais rápido possível, é analisado o desempenho do Kafka com diferentes tamanhos de mensagens. Cada mensagem que é enviada pelo Kafka possui um determinado tamanho, pelo que o interesse é conhecer a sua influência no desempenho do Kafka, ou seja, analisar quantas mensagens o Kafka é capaz de enviar e quantas mensagens o Spark Streaming é capaz de receber por segundo. Desta forma, encontra-

²² <http://spark.apache.org/docs/latest/streaming-kafka-0-10-integration.html>

²³ <https://kafka.apache.org/0101/documentation.html>

se ilustrado na Figura 19 este processo, pelo que são efetuados testes com mensagens de diferentes tamanhos, nomeadamente de 1, 10, 100, 1000 e 10000 *bytes*, tendo em conta que, em média, cada mensagem do Twitter possui 1500 *bytes*. Em cada um dos testes é registado o número médio de mensagens recebidas pelo Spark Streaming, por segundo, sendo que este teste foi realizado em cerca de 5 minutos de dados, apresentando-se os seus resultados no Gráfico 1.

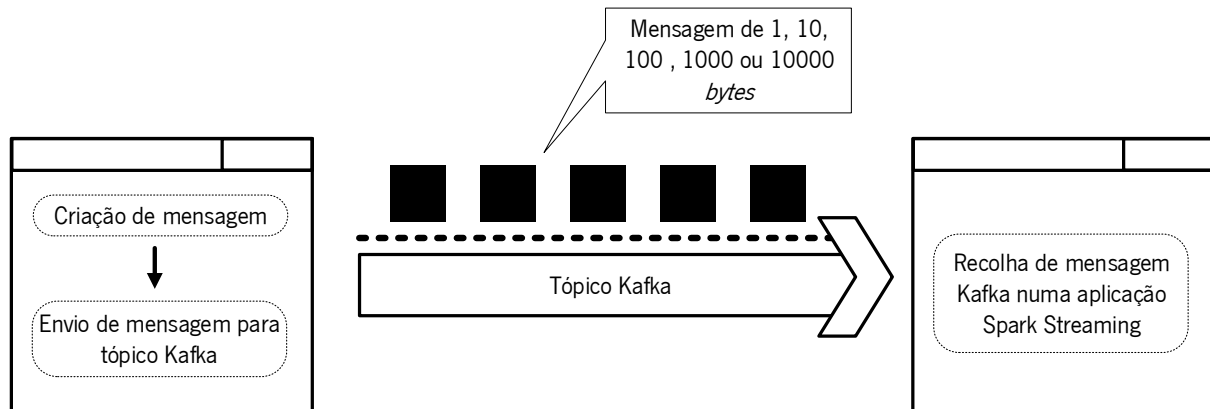


Figura 19 - Processo do envio de mensagens através do Kafka para o Spark Streaming

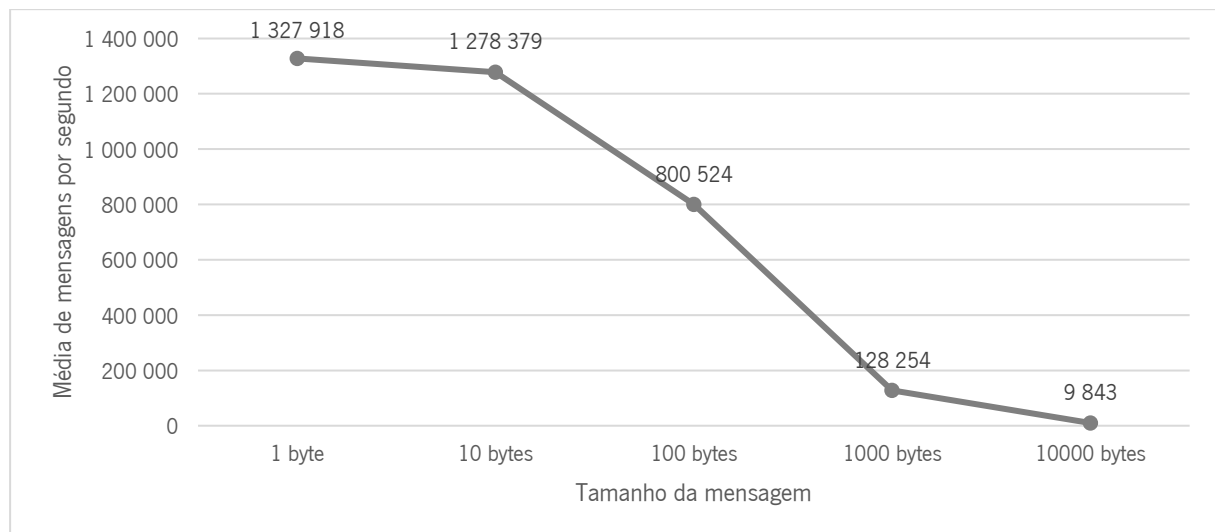


Gráfico 1 - Média de mensagens consumidas pelo Kafka

Pelos resultados obtidos, no Gráfico 1, pode-se afirmar que, como seria de esperar, quanto maior o tamanho das mensagens geradas, em *bytes*, menor é o número de mensagens enviadas do Kafka para o Spark Streaming, por segundo. Porém estes resultados são apresentados sem a influência da conexão ao Twitter, para recolha de *tweets* em *streaming*, pelo que é novamente efetuado este teste. O envio de mensagens de diferentes tamanhos, mas com a recolha como mensagens do Twitter apresenta-se ilustrado na Figura 20, pelo que os resultados deste teste se encontram representados no Gráfico 2.

Com os resultados apresentados no Gráfico 2, pode-se afirmar que com a utilização do mecanismo de recolha de dados do Twitter, o comportamento é bem diferente ao observado no Gráfico

1. Desta forma, o mecanismo de recolha de dados do Twitter revela-se limitador, seja qual for o tamanho da mensagem, pelo que não é pelo tamanho das mensagens que será recolhido um maior número de mensagens do Twitter.

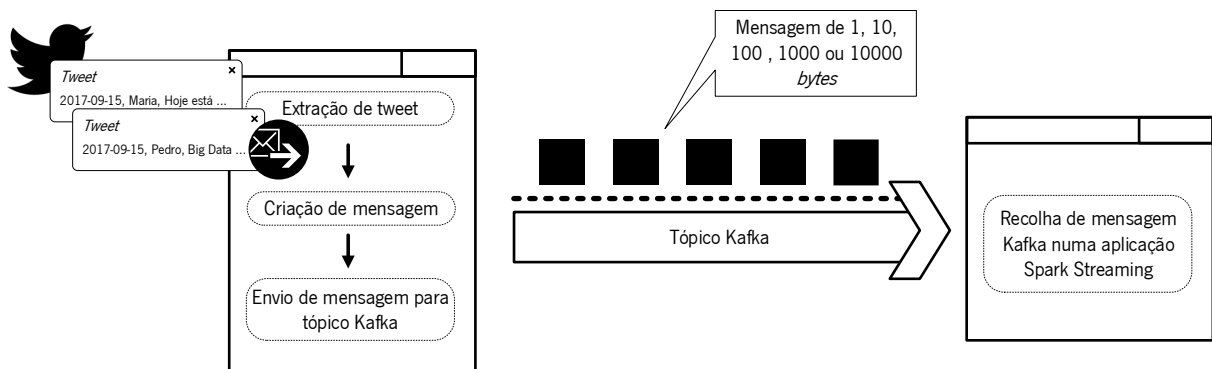


Figura 20 - Processo do envio de mensagens através do Kafka para o Spark Streaming com mecanismo do Twitter

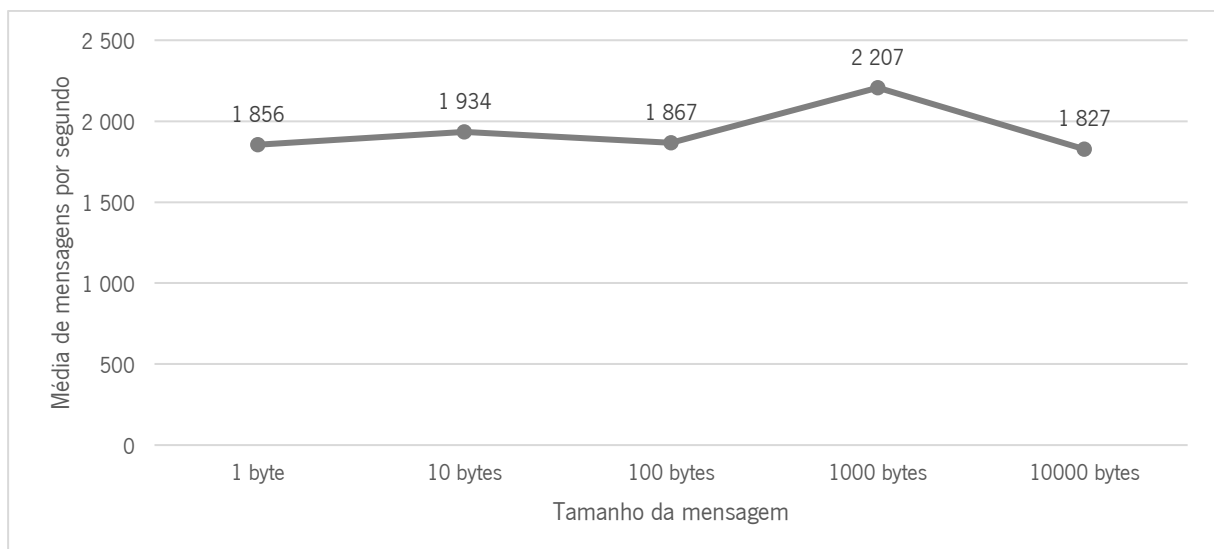


Gráfico 2 - Média de mensagens consumidas pelo Kafka com mecanismo Twitter

De referir que o Kafka possui uma propriedade relacionada com o tamanho do *batch*, nomeadamente *batch.size*, que apresenta o valor de 16384 *bytes*. Contudo, como neste contexto as mensagens estão a ser enviadas por *streaming*, cada envio possui apenas uma mensagem. Segundo a documentação do Kafka²⁴, num cenário de envio de mensagens em *batch*, quando é atingido este tamanho, não são enviadas mais mensagens neste conjunto de mensagens, pelo que, neste caso o *batch.size* representa o tamanho máximo de cada mensagem. Como em média cada mensagem do Twitter possui 1500 *bytes*, não existem muitas limitações com o tamanho de cada mensagem, pois este valor é inferior ao valor máximo do *batch.size*. Neste sentido, foram efetuados testes com um *batch.size*

²⁴ <https://kafka.apache.org/0101/documentation.html>

mais adequado ao tamanho das mensagens do Twitter, nomeadamente 4096 e 8092 *bytes*, contudo, em nenhum destes cenários se evidenciou benéfico comparativamente ao cenário normal de fluxo de dados do Twitter. Alguns testes foram realizados com outras propriedades, todavia os resultados obtidos com estas propriedades não se revelaram significativos. Porém, os testes realizados não invalidam um estudo mais pormenorizado das propriedades do Kafka com o objetivo de perceber a melhoria do desempenho no envio das mensagens.

O objetivo destes testes era tentar perceber se deveria haver um reajuste das propriedades do Kafka, de forma a garantir que as mensagens são enviadas e recebidas o mais rapidamente possível. Contudo, face ao contexto em que está inserido este trabalho e aos resultados obtidos, considera-se que não poderão ser feitas melhorias significativas. Os resultados obtidos não apresentam melhorias para permitir o fluxo de dados mais rapidamente nem com maior volume. Desta forma, neste contexto, o tamanho da mensagem pode ser maior ou menor, possuir mais ou menos atributos, e não é isso que irá aumentar ou diminuir o número de mensagens a serem enviadas por segundo. Porém, dependendo do contexto de trabalho, a otimização do desempenho da tecnologia do Kafka no envio de mensagens poderá ser estudada com mais pormenor, através da configuração das várias propriedades.

5.2. Processamento de Dados

A tecnologia Spark Streaming é responsável pelo processamento dos dados em tempo real, pelo que se espera que execute as suas tarefas com a maior rapidez para não atrasar o fluxo dos dados até ao armazenamento. Esta tecnologia permite o processamento de dados em *micro-batch*, sendo acumulado um conjunto de dados para serem processados pelo Spark Streaming. Este *micro-batch* é definido aquando da criação da aplicação, pelo que se deve atribuir um valor que seja superior ao tempo de processamento, para não haver atraso nas tarefas e todos os dados serem processados mal é criado o seu *micro-batch*.

De acordo com isto, para o caso em questão, foi definido um *micro-batch* de 5 segundos, visto que as tarefas de processamento têm normalmente uma duração até 3 segundos, como se apresenta na Figura 21. Além disso, com este *micro-batch* de 5 segundos, são consumidas cerca de 1500 a 2500 mensagem por segundo, ou seja, uma variação de 8000 a 13000 mensagens por 5 segundos, considerando que há variações ao longo do dia.

Todavia, tem que ser analisado cada contexto e adequar o *micro-batch* ao cenário em que está inserido. Pelo facto de este ser um contexto em tempo real, mais rapidamente se tentaria concretizar um *micro-batch* de 1 segundo, contudo, este tempo não é suficiente para o processamento dos dados, o que iria levar a atrasos na inserção e a uma lista de tarefas pendentes para serem executadas pelo

Spark Streaming. Estando esta dissertação inserida num contexto de BDW, e não numa aplicação de detecção de erros ou falhas, o foco não é tanto neste *micro-batch* de 1 segundo, mas sim em que todos os dados sejam transformados/tratados a tempo para que os dados fluam normalmente e que haja preparação e enriquecimento dos dados para análise.

Batch Time	Input Size	Scheduling Delay ⁽¹⁾	Processing Time ⁽²⁾	Total Delay ⁽³⁾	Output Ops: Succeeded/Total
2017/09/24 13:15:40	13150 records	0 ms	3 s	3 s	1/1
2017/09/24 13:15:35	12400 records	0 ms	2 s	2 s	1/1
2017/09/24 13:15:30	12543 records	0 ms	3 s	3 s	1/1
2017/09/24 13:15:25	13657 records	1 ms	3 s	3 s	1/1
2017/09/24 13:15:20	12800 records	0 ms	3 s	3 s	1/1
2017/09/24 13:15:15	13600 records	0 ms	3 s	3 s	1/1
2017/09/24 13:15:10	13800 records	0 ms	3 s	3 s	1/1
2017/09/24 13:15:05	12350 records	0 ms	2 s	2 s	1/1
2017/09/24 13:15:00	12250 records	0 ms	3 s	3 s	1/1
2017/09/24 13:14:55	12450 records	0 ms	3 s	3 s	1/1
2017/09/24 13:14:50	12437 records	0 ms	3 s	3 s	1/1
2017/09/24 13:14:45	13263 records	0 ms	3 s	3 s	1/1
2017/09/24 13:14:40	12350 records	0 ms	3 s	3 s	1/1
2017/09/24 13:14:35	12600 records	1 ms	3 s	3 s	1/1

Figura 21 - Processamento de dados na aplicação Spark Streaming

De forma a exemplificar o problema de o *micro-batch* não se adequar ao tempo de processamento, na Figura 22 apresenta-se ilustrado um caso em que o *micro-batch* é de 5 segundos e o processamento demora 7 segundos. Neste caso, como o tempo de processamento é superior ao tempo de cada *micro-batch*, a execução das tarefas fica comprometida, acumulando tarefas em fila de espera, pelo que chega, neste caso, a um atraso de 22 minutos.

2017/09/08 09:47:15	6200 records	-	-	0/1	queued
2017/09/08 09:47:10	8650 records	-	-	0/1	queued
2017/09/08 09:47:05	6750 records	-	-	0/1	queued
2017/09/08 09:47:00	7650 records	-	-	0/1	queued
2017/09/08 09:46:55	7800 records	-	-	0/1	queued
2017/09/08 09:46:50	7150 records	-	-	0/1	queued
2017/09/08 09:46:45	7550 records	-	-	0/1	queued
2017/09/08 09:46:40	8800 records	-	-	0/1	queued
2017/09/08 09:46:35	9550 records	-	-	0/1	queued
2017/09/08 09:46:30	8450 records	-	-	0/1	queued
2017/09/08 09:46:25	8966 records	-	-	0/1	queued
2017/09/08 09:46:20	7484 records	-	-	0/1	queued
2017/09/08 09:46:15	7350 records	-	-	0/1	queued
2017/09/08 09:46:10	6850 records	22 min	-	0/1	processing

Completed Batches (last 1000 out of 2080)

Batch Time	Input Size	Scheduling Delay ⁽¹⁾	Processing Time ⁽²⁾	Total Delay ⁽³⁾	Output Ops: Succeeded/Total
2017/09/08 09:46:05	9350 records	22 min	7 s	22 min	1/1
2017/09/08 09:46:00	7889 records	22 min	7 s	22 min	1/1
2017/09/08 09:45:55	7935 records	22 min	7 s	22 min	1/1
2017/09/08 09:45:50	7887 records	22 min	7 s	22 min	1/1
2017/09/08 09:45:45	7339 records	22 min	7 s	22 min	1/1
2017/09/08 09:45:40	7350 records	22 min	7 s	22 min	1/1
2017/09/08 09:45:35	8300 records	22 min	7 s	22 min	1/1
2017/09/08 09:45:30	8350 records	22 min	7 s	22 min	1/1

Figura 22 - Fila de espera na aplicação Spark Streaming

Além disto, enquanto está a ser processado o *micro-batch* do Spark Streaming, podem existir consultas aos dados, bem como acessos dos vários utilizadores, pelo que estes podem atrasar o processamento. Normalmente, num cenário em que o *micro-batch* definido se adequa ao tempo de processamento, como se encontra ilustrado na Figura 23, quando os dados são consultados, a diferença entre o tempo de processamento e o tempo do *micro-batch* é benéfica para o possível atraso provocado pela consulta de dados.

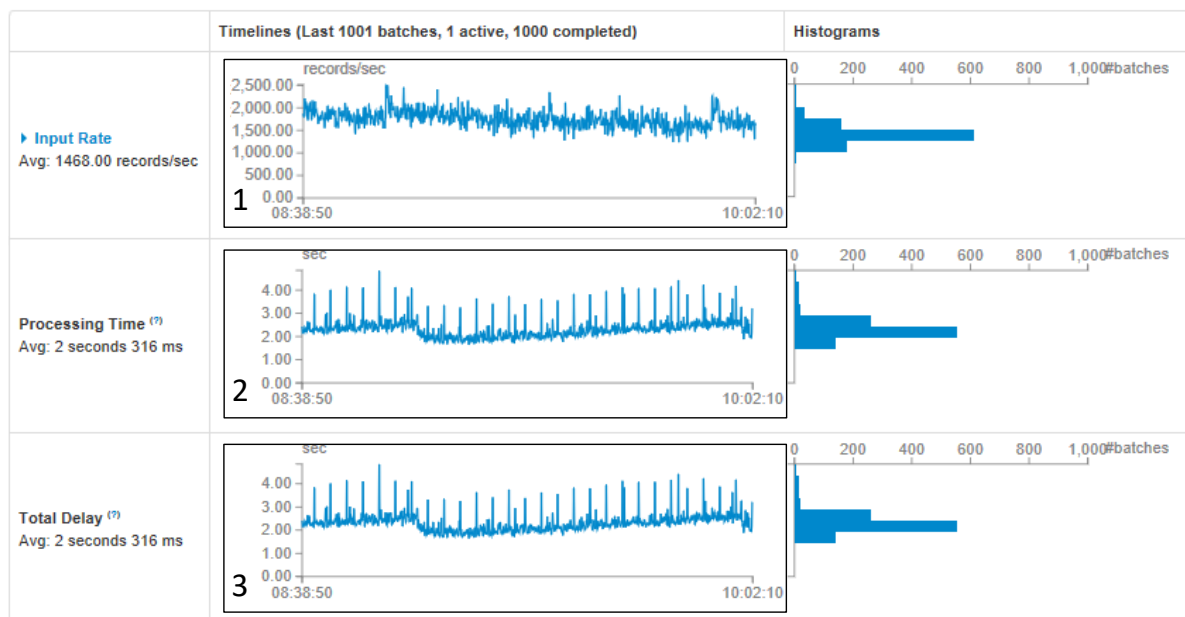


Figura 23 - Consulta de dados na aplicação Spark Streaming

Na Figura 23, no gráfico 1, encontra-se representado o fluxo de dados, pelo que como já referido este apresenta uma variação de 1500 a 2500 registos por segundo, existindo alguns picos no fluxo de dados. No gráfico 2, encontra-se representado o tempo de processamento, pelo que neste caso varia até 5 segundos, o que representa um cenário ideal, visto que não supera o tempo do *micro-batch*, não provocando assim o atraso da inserção de dados. Por fim, no gráfico 3, encontra-se representado o atraso total, que corresponde à soma do tempo de espera, que neste caso não existe, e do tempo de processamento, sendo que como neste caso a média é inferior ao tempo do *micro-batch*, não são acumuladas tarefas em espera. Este cenário representa o ideal para garantir que o tempo real possa ser concretizado no BDW.

5.3. Armazenamento de Dados

O armazenamento dos dados, sejam dados em tempo real ou dados históricos, é feito em tabelas suportadas pelas respetivas tecnologias, em formato ORC. Tanto para o armazenamento em tempo real como o histórico não foi definido nenhum modelo de dados, pelo que os dados se encontram desnormalizados e sem agregações. Este cenário é também verificado nos dados que são transmitidos do armazenamento em tempo real para o histórico, não sofrendo qualquer tipo de agregação. Assim, tanto o armazenamento dos dados nas respetivas áreas como a passagem dos dados do armazenamento em tempo real para o armazenamento histórico, são mais rápidas, devido a não existir agregação dos dados. Contudo, esta opção prende-se principalmente com o volume de dados e com a duração da dissertação, pelo que não invalida a possibilidade de se apresentarem modelos para os dados, bem

como até serem diferentes para dados em tempo real e históricos. Em muitos contextos organizacionais nem se revela necessária a mesma granularidade para dados em tempo real e dados históricos, podendo ser realizadas agregações nos dados, ou até a criação de modelos em estrela, havendo neste caso a penalização pelo processamento de *joins*, como referido por Floratou, Minhas, & Özcan (2014).

Visto que ambas as tecnologias de armazenamento possuem especificidades que afirmam trazer melhorias no desempenho, há que ter em conta essas especificidades. Neste sentido, para ambas as tecnologias foram avaliadas as vantagens de serem aplicadas essas especificidades, sendo analisados os resultados das consultas de forma a compreender se se justifica a sua utilização.

Deste modo, de seguida são debatidos alguns resultados dos testes efetuados de forma a serem retiradas algumas conclusões sobre o tipo de armazenamento que se adequa a cada situação. Como aqui o objetivo não é avaliar as ferramentas de consulta de dados, é apenas usado o Presto pois este revela possuir um melhor desempenho, como referido por M. Y. Santos, Costa, et al. (2017).

Para o armazenamento em tempo real, os testes foram realizados com dados a fluir em *streaming*, pelo que o fluxo não é exatamente o mesmo em todas as horas, contudo não existem picos nos dados. O fluxo de dados representa uma média de 1500 a 2500 registos por segundo e uma variação de 8000 a 13000 registos por 5 segundos, considerando que há variações ao longo do dia. As variações ao longo do dia devem-se ao facto de os dados serem extraídos do Twitter em momentos diferentes, traduzindo-se em testes com um volume de dados um pouco diferente, contudo semelhantes e até proporcionais ao longo das várias horas. De 1 em 1 hora, num período de 8 horas, é assim questionado o armazenamento dos dados com um conjunto de consultas, sendo que são usadas as mesmas consultas para o Hive e para o Cassandra.

Para o armazenamento de dados históricos, os testes foram realizados com dados que migraram do armazenamento de dados em tempo real para o histórico, pelo que é avaliado o seu desempenho com diferentes volumes de dados. As consultas efetuadas nos vários testes representam uma amostra das questões que poderiam ser colocadas aos dados, tendo em vista o interesse analítico, bem como a variedade e complexidade de operadores. De seguida, na Tabela 1, apresenta-se a lista de consultas utilizadas neste caso de demonstração.

Tabela 1 - Consultas aos dados

Consulta	Designação	Instrução SQL
Consulta 1 (C1)	Verificação dos <i>tweets</i> existentes	<code>select count(*) from table</code>
Consulta 2 (C2)	Verificação dos <i>tweets</i> existentes apenas na hora anterior	<code>select count(*) from table where tweet_date_hour='\$hora_anterior'</code>

Consulta	Designação	Instrução SQL
Consulta 3 (C3)	As cinco línguas mais verificadas nos <i>tweets</i> da hora anterior	<code>select lang, count(*) from table where tweet_date_hour='\$hora_anterior' group by lang order by number_of_tweets desc limit 5</code>
Consulta 4 (C4)	Verificação dos <i>tweets</i> existentes em cada hora	<code>select tweet_date_hour, count(*) from table group by tweet_date_hour order by tweet_date_hour asc</code>
Consulta 5 (C5)	Verificação das cinco horas com mais <i>tweets</i>	<code>select tweet_date_hour, count(*) as number_of_tweets from table group by tweet_date_hour order by number_of_tweets desc limit 5</code>
Consulta 6 (C6)	As cinco línguas mais verificadas nos <i>tweets</i>	<code>select lang, count(*) as number_of_tweets from table group by lang order by number_of_tweets desc limit 5</code>
Consulta 7 (C7)	A média de <i>tweets</i> por hora	<code>select avg(contador) as tweets_hora from (select count(*) as contador from table group by tweet_date_hour) as b</code>
Consulta 8 (C8)	Verificação das cinco datas com mais <i>tweets</i>	<code>select tweet_date, count(*) as number_of_tweets from table group by tweet_date order by number_of_tweets desc limit 5</code>
Consulta 9 (C9)	As cinco horas onde são verificados mais <i>tweets</i>	<code>select tweet_date_hour, count(*) from historical2 group by tweet_date_hour order by count(*) desc limit 5</code>
Consulta 10 (C10)	Verificação dos <i>tweets</i> existentes cuja língua utilizada é o inglês	<code>select count(*) from table where lang='en'</code>
Consulta 11 (C11)	Verificação dos <i>tweets</i> existentes num determinado dia	<code>select count(*) from table where tweet_date='\$data'</code>
Consulta 12 (C12)	Verificação dos <i>tweets</i> existentes para cada minuto numa determinada data	<code>select tweet_date_minute, count(*) from table where tweet_date='\$data' group by tweet_date_minute order by tweet_date_minute asc</code>
Consulta 13 (C13)	Verificação dos <i>tweets</i> existentes para cada minuto numa determinada hora	<code>select tweet_date_minute, count(*) from table where tweet_date_hour='\$hora' group by tweet_date_minute order by tweet_date_minute asc</code>
Consulta 14 (C14)	Os cinco utilizadores com mais <i>tweets</i> entre um intervalo de datas	<code>select distinct user_id, count(*) from (select * from table union all select * from table2) where createdat <= now() and createdat >= '\$data' group by user_id order by count(*) desc limit</code>
Consulta 15 (C15)	Os cinco dispositivos mais verificados	<code>select source, count(*) as contador from (select * from table union all select * from table2) group by source order by contador desc limit 5</code>
Consulta 16 (C16)	As cinco datas em que se verificam um maior número de <i>tweets</i>	<code>select tweet_date, count(*) as contador from (select * from table union all select * from table2) group by tweet_date order by contador desc limit 5</code>

De referir que, o armazenamento de dados em tempo real chega a um ponto em que o seu desempenho começa a piorar com o aumento do volume de dados, razão pela qual existe a necessidade de os dados serem movidos para o armazenamento de dados histórico. Nesta secção é analisado o comportamento das tecnologias com o aumento do volume de dados em vários cenários, bem como o ganho ao nível de desempenho, assim que esses dados são movidos para o histórico.

5.3.1. Hive no Armazenamento de Dados em Tempo Real

A tecnologia Hive organiza os seus dados em tabelas nas quais podem ser criadas partições, que são uma forma de subdividir os dados da tabela, otimizando o seu desempenho na análise de dados. Assim como referido por Costa, Costa, & Santos (2017), Rutherglen, Wampler, & Capriolo, (2012), Thusoo et al. (2009, 2010) e White (2012), a organização dos dados em partições traz benefícios no acesso mais rápido aos dados, otimizando a consulta de dados. A criação de partições permite organizar os dados em pastas, de acordo com os atributos de particionamento, estando nessa pasta apenas os registos correspondentes. Além disso, nas tabelas ou partições podem ser criados *buckets* para organizar os dados, contudo, como estes são mais apropriados para quando se pretende trabalhar com amostras de dados, como referido por Rutherglen et al. (2012), considerou-se que não se adequavam ao trabalho desta dissertação.

Neste sentido, optou-se então pela organização dos dados em duas tabelas, uma sem particionamento e outra com particionamento, sendo recolhidos, processados e armazenados dados em *streaming* para serem testadas as duas áreas de armazenamento. Neste caso, o atributo de particionamento é a hora dos eventos, porque como o armazenamento é em tempo real só se espera ter dados daquele dia, de menos de 24 horas. Na Figura 24 é apresentado como o sistema de ficheiros, o HDFS, organiza os dados na tabela particionada, nomeadamente numa espécie de pastas, cada uma referente a uma determinada hora.


Name >	Size >	Last Modified >	Owner >	Group >	Permission
					
<input type="checkbox"/> tweet_date_hour=15	--	2017-09-15 16:00	lid4	hdfs	drwxrwxrwx
<input type="checkbox"/> tweet_date_hour=16	--	2017-09-15 17:00	lid4	hdfs	drwxrwxrwx
<input type="checkbox"/> tweet_date_hour=17	--	2017-09-15 18:00	lid4	hdfs	drwxrwxrwx
<input type="checkbox"/> tweet_date_hour=18	--	2017-09-15 19:00	lid4	hdfs	drwxrwxrwx
<input type="checkbox"/> tweet_date_hour=19	--	2017-09-15 20:00	lid4	hdfs	drwxrwxrwx

Figura 24 - Sistema de ficheiros da tabela particionada

As consultas de dados definidas pretendem avaliar o desempenho das duas áreas de armazenamento, sendo definidas as mesmas consultas para as duas tabelas, com base nos atributos existentes na amostra de dados. Tendo em conta que a criação de partições nos dados é favorecida quando as consultas apresentam no operador “where” o atributo pelo qual a tabela foi particionada, foram definidas consultas com e sem este operador de forma a conhecer o desempenho da tecnologia. Numa tabela particionada, quando existe este operador, há um filtro das partições em que é necessário ler os dados, não havendo assim a necessidade de percorrer todos os registos, que é o que acontece numa tabela não particionada.

De referir que todos os testes foram realizados em tabelas externas, não geridas pelo Hive, e com particionamento dinâmico. Apesar de não existir grande diferença entre as tabelas geridas pelo Hive e tabelas externas, as tabelas externas não têm que manter as estatísticas automaticamente, pelo que pode ajudar na redução do tempo de inserção dos dados (Rutherglen et al., 2012; White, 2012).

Para avaliar o desempenho da tecnologia Hive, é analisado o seu comportamento numa tabela particionada (CP) à hora do evento e outra tabela não particionada (SP), ambas alimentadas com dados recolhidos em *streaming* em simultâneo com a realização destes testes. Para a comparação do desempenho das tabelas são analisados os resultados de algumas consultas, de forma a perceber qual a mais adequada para o contexto deste trabalho, pelo que de seguida são debatidos os resultados obtidos.

No Gráfico 3 encontra-se representada a consulta 1 (C1), que é uma verificação dos registos existentes na tabela que está a ser alimentada por *streaming*, pelo que tanto na tabela particionada como na tabela não particionada, com esta consulta, é necessário percorrer todos os registos da tabela. Isto deve-se ao facto de esta consulta não possuir nenhum operador “where”, pelo que têm que ser analisados todos os registos, estando a tabela organizada ou não em partições.

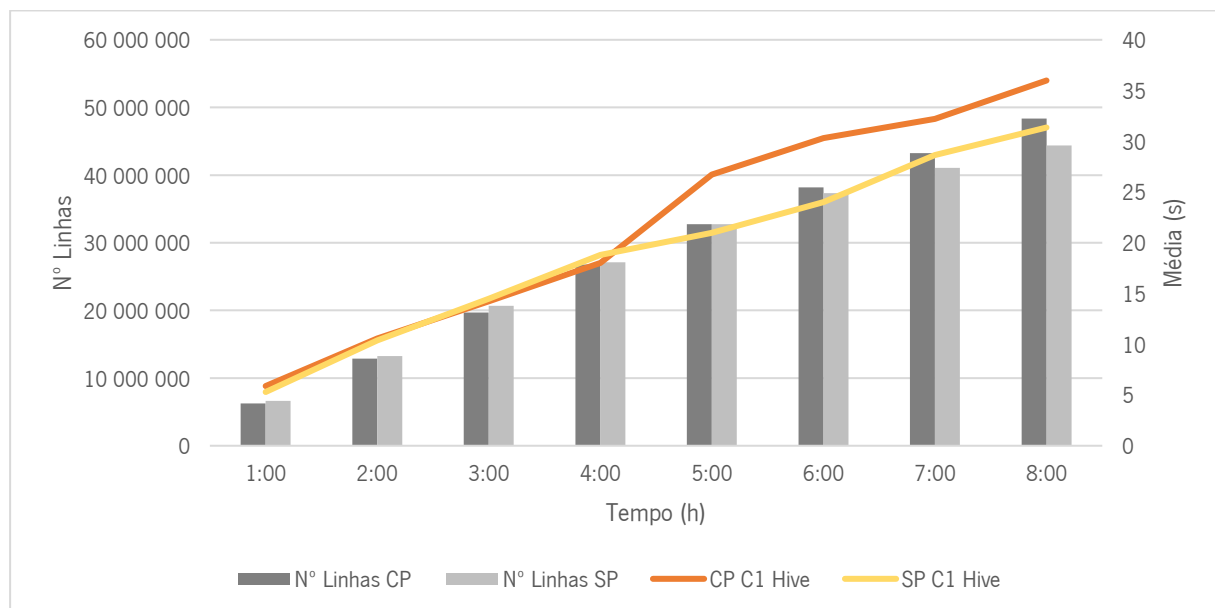


Gráfico 3 - Resultados da Consulta 1 (C1) para a tabela particionada (CP) e não particionada (SP)

Como se pode observar, o número de linhas armazenadas na tabela particionada e não particionada, nas primeiras horas, não é exatamente igual, algo que se deve às variações do fluxo de dados em *streaming*. Quando, ao fim de 5 a 6 horas do início da recolha, processamento e armazenamento de dados, a tabela não particionada começa a registar um menor número de linhas, as consultas de dados são mais rápidas. Quando o volume de dados armazenado é menor, numa consulta

como esta, são menos dados para ler, revelando-se inevitavelmente em consultas com melhor desempenho. O facto de o volume de dados ser menor deve-se à dificuldade na inserção de dados, acumulando alguns conjuntos de linhas de dados, e atrasando a sua inserção. Como referido por White (2012), no Hive, na inserção de registos são criados muitos ficheiros de pequena dimensão e, antes de estes serem inseridos, existe um conjunto de operações de verificação, de forma a averiguar se o ficheiro já existe bem como se existe permissão para o criar. Esta situação evidencia-se na tabela não particionada, pelo que estas operações de verificação são dificultadas ao fim de algumas horas de recolha, processamento e armazenamento de dados por existirem mais ficheiros para ser efetuada a verificação. Na Figura 25 é apresentado o estado do sistema de ficheiros, podendo verificar-se a existência de muitos ficheiros de pequena dimensão. Uma solução para este problema, de os ficheiros apresentarem pequena dimensão, é o aumento do *micro-batch* do Spark Streaming, o que permite que o tamanho dos ficheiros seja superior. Na tabela particionada, como só se encontram apenas os registos daquela hora, esta dificuldade não se verifica, pois, o conjunto de ficheiros é só referente à respetiva hora, existindo um conjunto de dados muito mais pequeno em cada uma das partições.

Name >	Size >	Last Modified >	Owner >	Group >	Permission
↳					
↳ part-00002	12.2 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00000	12.9 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00000_copy_2	45.8 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00000_copy_3	44.0 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00000_copy_4	40.3 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00000_copy_5	44.9 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00000_copy_6	41.2 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx
↳ part-00001	13.2 kB	2017-10-12 17:38	lid4	hdfs	-rwxrwxrwx

Figura 25 - Ficheiros de pequena dimensão no sistema de ficheiros

Num teste como este, o que acontece é que, como representado na Figura 26, as tarefas de inserção de dados, ao fim de algumas horas de fluxo de dados, são acumuladas em espera. Devido a existirem mais ficheiros para serem verificados, o processamento de dados aumenta de 3 segundos para até 10 segundos, transmitindo-se num valor superior ao tamanho do *micro-batch*, não existindo a capacidade de processar as tarefas mal estas estão disponíveis, provocando inevitavelmente o acumular de tarefas em espera. Como se apresenta na Figura 26, o atraso acumulado vai neste caso em 1,9 horas, pelo que, por exemplo, às 09h14 estão a ser inseridos os dados das 07h20, sendo que à medida que o tempo passa, maior será o atraso acumulado.

Active Batches (1376)

Batch Time	Input Size	Scheduling Delay ^(h)	Processing Time ^(s)	Output Ops: Succeeded/Total	Status
2017/10/06 09:14:50	8300 records	-	-	0/1	queued
2017/10/06 09:14:45	7142 records	-	-	0/1	queued
2017/10/06 09:14:40	6831 records	-	-	0/1	queued
2017/10/06 09:14:35	7827 records	-	-	0/1	queued
2017/10/06 09:14:30	6650 records	-	-	0/1	queued
2017/10/06 09:14:25	6850 records	-	-	0/1	queued
2017/10/06 09:14:20	7250 records	-	-	0/1	queued
2017/10/06 09:14:15	6900 records	-	-	0/1	queued
2017/10/06 09:14:10	7900 records	-	-	0/1	queued
● ● ●					
2017/10/06 07:20:25	7350 records	-	-	0/1	queued
2017/10/06 07:20:20	8450 records	-	-	0/1	queued
2017/10/06 07:20:15	7100 records	1.9 h	-	0/1	processing

Completed Batches (last 1000 out of 5303)

Batch Time	Input Size	Scheduling Delay ^(h)	Processing Time ^(s)	Total Delay ^(h)	Output Ops: Succeeded/Total
2017/10/06 07:20:10	8450 records	1.9 h	10 s	1.9 h	1/1
2017/10/06 07:20:05	8350 records	1.9 h	9 s	1.9 h	1/1
2017/10/06 07:20:00	7450 records	1.9 h	9 s	1.9 h	1/1
2017/10/06 07:19:55	7400 records	1.9 h	10 s	1.9 h	1/1
2017/10/06 07:19:50	6850 records	1.9 h	9 s	1.9 h	1/1
2017/10/06 07:19:45	7150 records	1.9 h	9 s	1.9 h	1/1
2017/10/06 07:19:40	7100 records	1.9 h	9 s	1.9 h	1/1

Figura 26 - Dificuldade na inserção de dados na tabela não particionada do Hive

Numa organização, quando se deparam com problemas como este, pode haver investimentos de *hardware*, no sentido de resolver o problema. Neste caso de demonstração, o que poderia ser feito era a concatenação desses ficheiros, sendo realizada essa operação com uma determinada periodicidade ou antes da inserção dos dados. Porém, esta operação iria atrasar o processamento dos dados que acabam de ser recolhidos, na medida em que atrasaria o seu armazenamento.

Todavia, num contexto organizacional, muitas das consultas desejadas são vistas numa janela temporal, havendo uma filtragem temporal dos dados a serem consultados. É nestes contextos que o particionamento evidencia as suas potencialidades, pela existência do operador “where” com o atributo de particionamento nas consultas. No Gráfico 4 apresentam-se os resultados da consulta 2 (C2), que é uma verificação dos registos existentes apenas na última hora, e no Gráfico 5 da consulta 3 (C3), que representa as cinco línguas mais verificadas nos *tweets* na última hora.

Pelo que se pode observar, tanto no Gráfico 4 como Gráfico 5, o desempenho das consultas é melhor na tabela particionada do que na tabela não particionada. O conjunto de dados a ser percorrido pela C2 e C3 na tabela particionada é menor do que na tabela não particionada, pelo que são percorridos apenas os registos da partição da hora anterior, e não todos os registos existentes na tabela. Com o aumento do volume de dados, as consultas à tabela particionada mantêm o tempo necessário, visto que estão apenas a aceder a uma partição e essas possuem um volume de dados mais ou menos uniforme. Por outro lado, na tabela não particionada, com o aumento do volume de dados, o tempo de resposta das consultas aumenta, pois esta é obrigada a percorrer muitos mais registos para apresentar o resultado. Além disso, como na C1, ao fim de 5 a 6 horas o número de registos começa a ser menor.

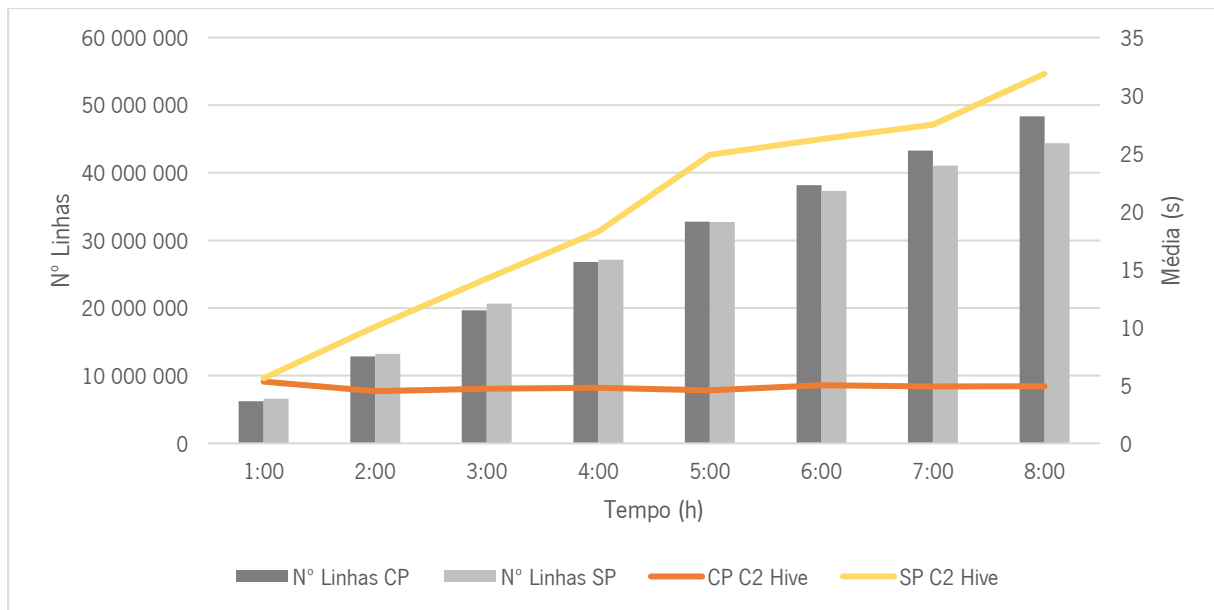


Gráfico 4 - Resultados da Consulta 2 (C2) para a tabela particionada (CP) e não particionada (SP)

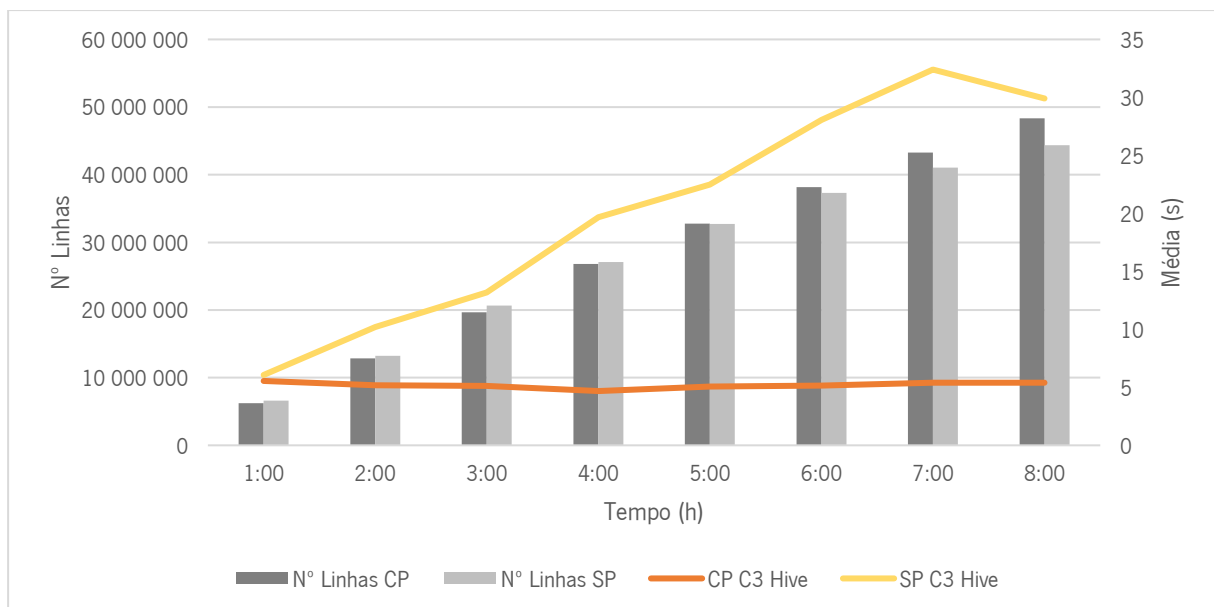


Gráfico 5 - Resultados da Consulta 3 (C3) para a tabela particionada (CP) e não particionada (SP)

De seguida, apresentam-se outras consultas, nomeadamente a consulta 4 (C4), que é a verificação de quantos registos existem em cada hora, a consulta 5 (C5), que é uma verificação das cinco horas com mais registos, a consulta 6 (C6), que é a verificação das cinco línguas mais utilizadas nos *tweets*, a consulta 7 (C7), que é a média de *tweets* por hora, e a consulta 8 (C8) que é a verificação das cinco datas com mais registos. Nestas consultas, com os tempos de processamento apresentados na Tabela 2, não existe no operador “where” o atributo de particionamento, pelo que as partições não se encontram favorecidas. Os resultados apresentados representam um comportamento bastante

semelhante entre a tabela particionada e não particionada, contudo a tabela particionada encontra-se ligeiramente prejudicada em termos de tempo de processamento.

Tabela 2 - Resultados das consultas 4, 5, 6, 7 e 8 na tabela particionada (CP) e tabela sem partições (SP) no Hive

Hora	Nº Linhas CP	CP C4 Hive (s)	CP C5 Hive (s)	CP C6 Hive (s)	CP C7 Hive (s)	CP C8 Hive (s)
01:00	6249238	5,021	5,779	5,326	5,246	5,127
02:00	12873525	10,021	9,836	10,326	9,928	9,751
03:00	19668338	15,062	14,790	16,201	13,916	16,190
04:00	26797143	19,368	19,205	21,038	19,434	20,108
05:00	32767975	23,711	24,805	25,667	24,150	25,427
06:00	38164450	29,838	30,530	29,790	28,743	31,367
07:00	43264900	33,961	34,861	33,558	35,985	33,792
08:00	48346425	37,821	39,512	38,167	40,283	39,497
Hora	Nº Linhas SP	SP C4 Hive (s)	SP C5 Hive (s)	SP C6 Hive (s)	SP C7 Hive (s)	SP C8 Hive (s)
01:00	6624185	5,465	5,327	5,513	5,451	5,622
02:00	13234724	10,523	9,527	11,798	9,590	9,712
03:00	20667007	14,642	15,537	14,425	15,623	14,014
04:00	27118538	17,998	20,694	20,250	20,324	18,314
05:00	32747794	24,741	25,005	23,309	21,110	22,461
06:00	37335494	24,992	26,765	25,547	26,001	27,060
07:00	41069351	29,865	29,125	29,444	28,490	30,680
08:00	44364001	34,173	35,045	32,145	32,886	31,359

Olhando para os vários resultados apresentados nas consultas efetuadas com o Presto ao repositório Hive, é possível afirmar que quando as consultas evidenciam as especificidades do particionamento, os resultados são mais positivos com a tabela particionada, mas quando as consultas não evidenciam as especificidades do particionamento, os resultados são mais positivos na tabela não particionada, apesar de nestes casos a diferença para os resultados da tabela particionada não ser muito significativa. De referir que esta tecnologia, quando questionada pelo Presto, não revela significante atraso na inserção dos dados na tabela particionada. Na tabela não particionada, como referido acima, muitas das suas tarefas já se encontram em fila de espera, mas devido a outras razões, pelo que a consulta aos dados poderá provocar um ligeiro atraso.

Desta forma, considera-se que a tabela quando particionada apresenta melhores resultados, porque apesar de em todas as consultas esta não ser necessariamente a que possui o melhor desempenho, quando são filtrados os dados possui um melhor comportamento. O cenário habitual numa organização, principalmente num contexto em que o interesse é conhecer os eventos que acabaram de acontecer para a tomada de decisão, a grande necessidade poderá nem ser a consulta de todos os dados

do armazenamento de dados em tempo real, mas sim apenas os das horas mais recentes. Além disso, o facto de a tabela não particionada, ao fim de algumas horas, não concretizar o tempo real, é um fator que não permite a execução do BDW em tempo real. Deste modo, dependendo do contexto do negócio, a tabela deve ser particionada por um ou mais atributos, pois desta forma as suas consultas poderão ser otimizadas.

De referir que os testes realizados com a tecnologia Hive foram inicialmente efetuados com o formato de ficheiros texto (em TXT), contudo, como também referido por Floratou, Minhas, & Özcan (2014), com o formato ORC é otimizada a análise de dados, pelo que os testes foram repetidos para apresentarem melhores resultados. De forma a ser compreendida a diferença em termos de desempenho da tecnologia Hive, com armazenamento de dados no formato TXT e ORC, no Gráfico 6 e Gráfico 7 apresenta-se a C1 (apresentada anteriormente no Gráfico 3) e a C2 (apresentada anteriormente no Gráfico 4), respetivamente.

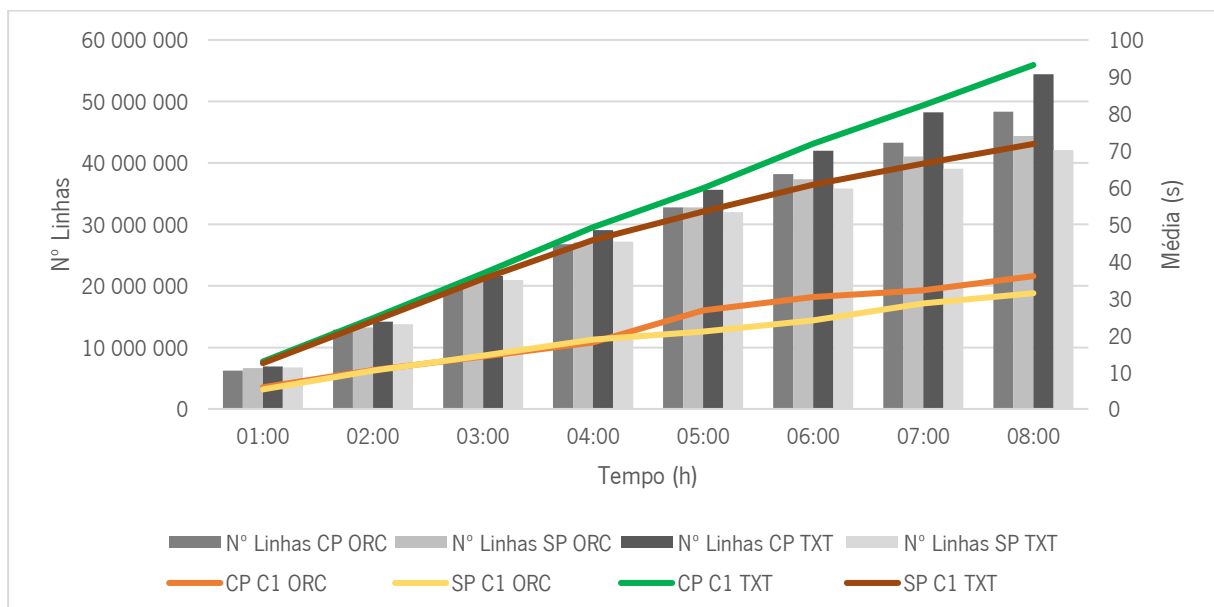


Gráfico 6 - Resultados da Consulta 1 (C1) para a tabela particionada (CP) e não particionada (SP) com formato ORC e TXT

Como se pode observar, no Gráfico 6, os resultados da C1 no armazenamento de dados com formato ORC são melhores do que os apresentados no formato TXT. Pelo facto de o formato ORC estar otimizado para a análise de dados, este apresenta melhores resultados, pelo que se revela o mais adequado para ser utilizado.

Os resultados da C2, apresentados no Gráfico 7, revelam novamente que o formato ORC é o mais adequado já que apresenta resultados com melhor desempenho. Na C2 está evidenciada a grande diferença de desempenho entre a tabela particionada e não particionada, pelo que os melhores resultados são apresentados na tabela particionada, enquanto que na C1, como são percorridos todos

os registos existentes na tabela, as partições não apresentam os melhores resultados, como já referido acima aquando a discussão de resultados da C1 e C2 (no Gráfico 3 e Gráfico 4 respetivamente). O facto de o teste com ORC ter menos ou mais linhas do que o teste com TXT deve-se ao fluxo dos dados do momento em que foi realizado o teste, pelo que apenas nos testes às tabelas sem particionamento é notável uma diminuição a partir das 5 a 6 horas, contudo isto deve-se ao acumular de tarefas em fila de espera para a inserção de dados, como referido acima nesta secção. Desta forma, em qualquer um dos casos, é possível afirmar que o formato ORC é mais adequado do que TXT devido a apresentar melhor desempenho nas consultas.

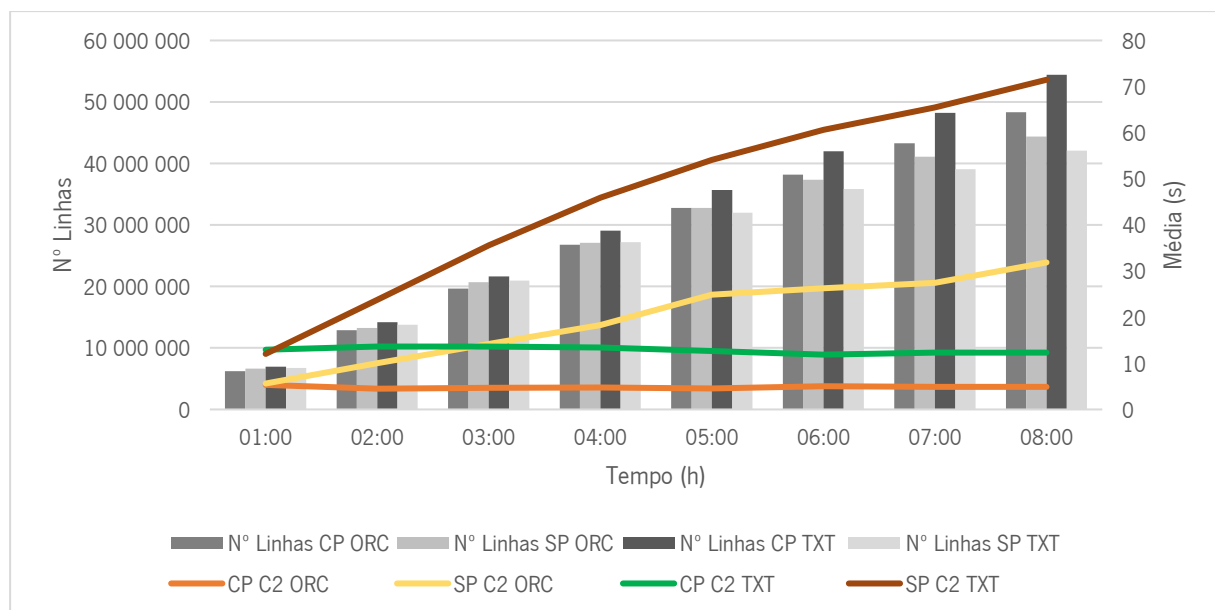


Gráfico 7 - Resultados da Consulta 2 (C2) para a tabela particionada (CP) e não particionada (SP) com formato ORC e TXT

5.3.2. Cassandra no Armazenamento de Dados em Tempo Real

Com a tecnologia Cassandra, aquando da definição da tabela para armazenar os dados é necessário definir uma chave primária, que é o identificador da linha. Como referido por Chebotko, Kashlev, & Lu (2015) e Hewitt & Carpenter (2016), a chave primária é composta pela chave de partição e chave de *clustering*. Estando o Cassandra distribuído pelos vários nós do *cluster*, a chave de partição determina em que nó os dados são armazenados e a chave de *clustering*, que é opcional, determina a ordem pela qual os dados são armazenados em cada nó. Neste sentido, de forma a perceber o comportamento da tabela com a utilização de *clustering*, foi criada uma tabela apenas com chave de partição e outra com chave de partição e chave de *clustering*. Optou-se então pela utilização do identificador da linha, que consiste na concatenação de um número aleatório e o *timestamp* da data do *tweet*, como chave da partição e a hora do evento como chave de *clustering*, visto que nesta tabela apenas estarão armazenados dados de um dia.

As consultas aos dados foram realizadas em duas tabelas, nomeadamente uma tabela com chave de *clustering* (CC) e outra tabela sem chave de *clustering* (SC). De seguida, apresentam-se os resultados obtidos enquanto estão a ser recolhidos, processados e armazenados dados em *streaming*.

O Gráfico 8 representa a consulta 1 (C1), que é a verificação de todos os registos da tabela, evidenciando que, de forma geral, as tabelas apresentam um desempenho semelhante. Aliás, a pequena diferença no número de linhas e consequentemente nos seus resultados deve-se principalmente ao fluxo de dados em *streaming* não ser exatamente o mesmo nos dois testes, pelo que quando o volume de dados é menor, a consulta será efetivamente mais rápida, pois existirão menos registos para serem percorridos. Desta forma, numa consulta como esta, a diferença entre com e sem *clustering* não é significativa, não estando evidenciado o benefício da existência da chave de *clustering*.

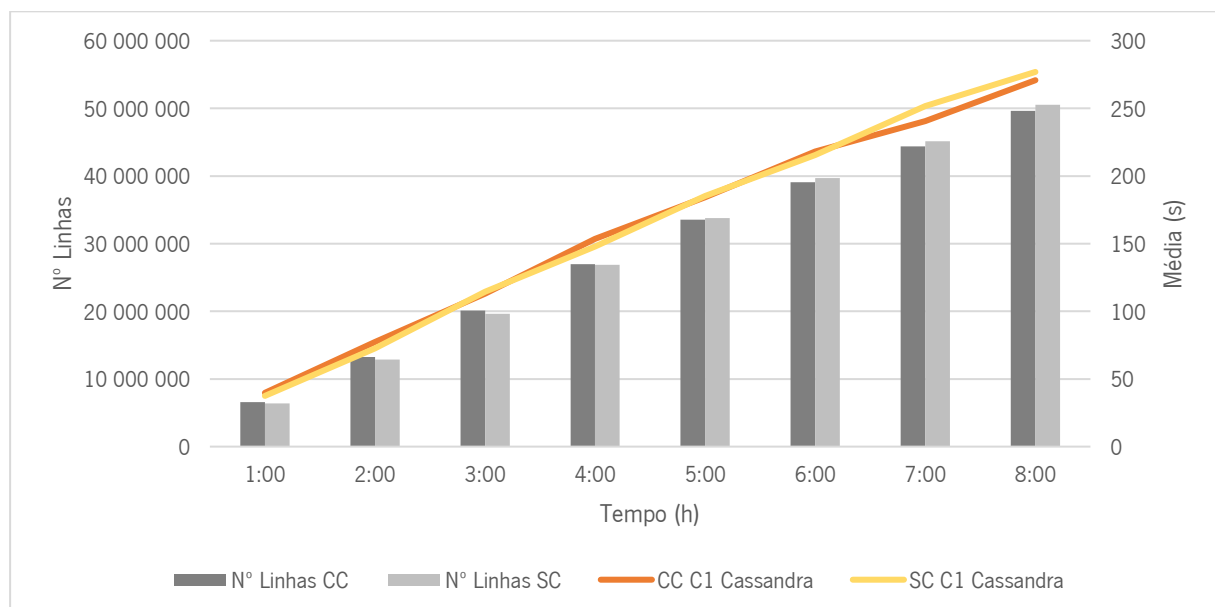


Gráfico 8 - Resultados da Consulta 1 (C1) para a tabela com chave de *clustering* (CC) e sem chave de *clustering* (SC)

De forma a tentar perceber em que cenários é que a tabela com chave de *clustering* poderia trazer vantagens, efetuou-se a consulta 2 (C2), que é uma verificação dos registos existentes apenas na última hora, e a consulta 3 (C3), que são as cinco línguas mais verificadas na última hora. Ambas as consultas agrupam e filtram os dados pela chave de *clustering*. De seguida, apresentam-se os gráficos das consultas 2 e 3, nomeadamente o Gráfico 9 e o Gráfico 10.

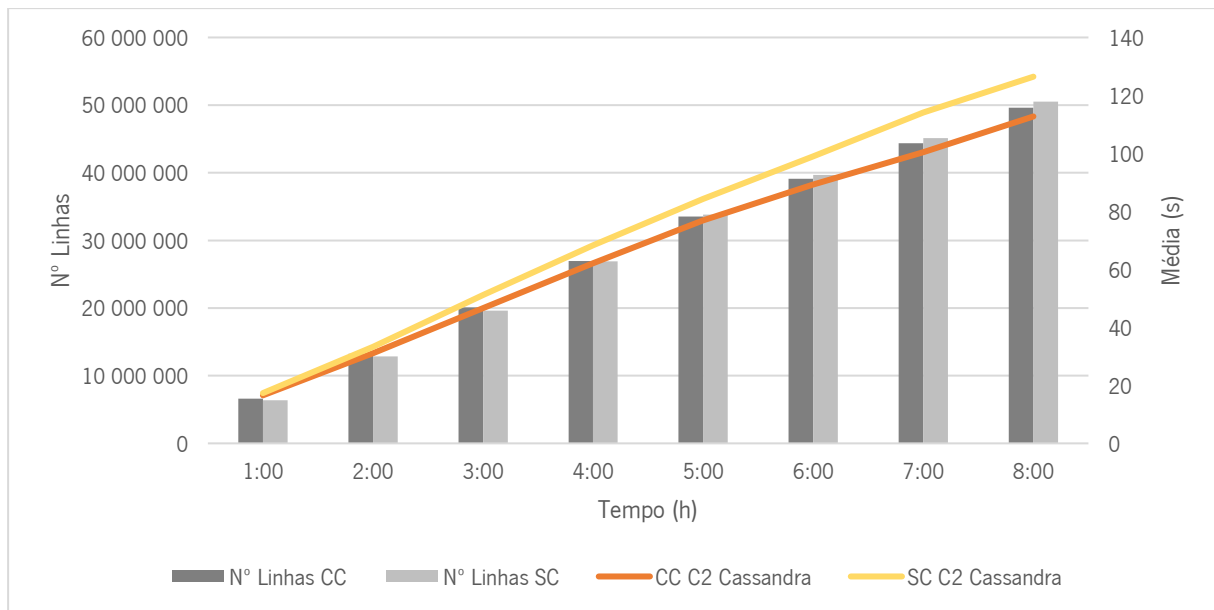


Gráfico 9 - Resultados da Consulta 2 (C2) para a tabela com chave de *clustering* (CC) e sem chave de *clustering* (SC)

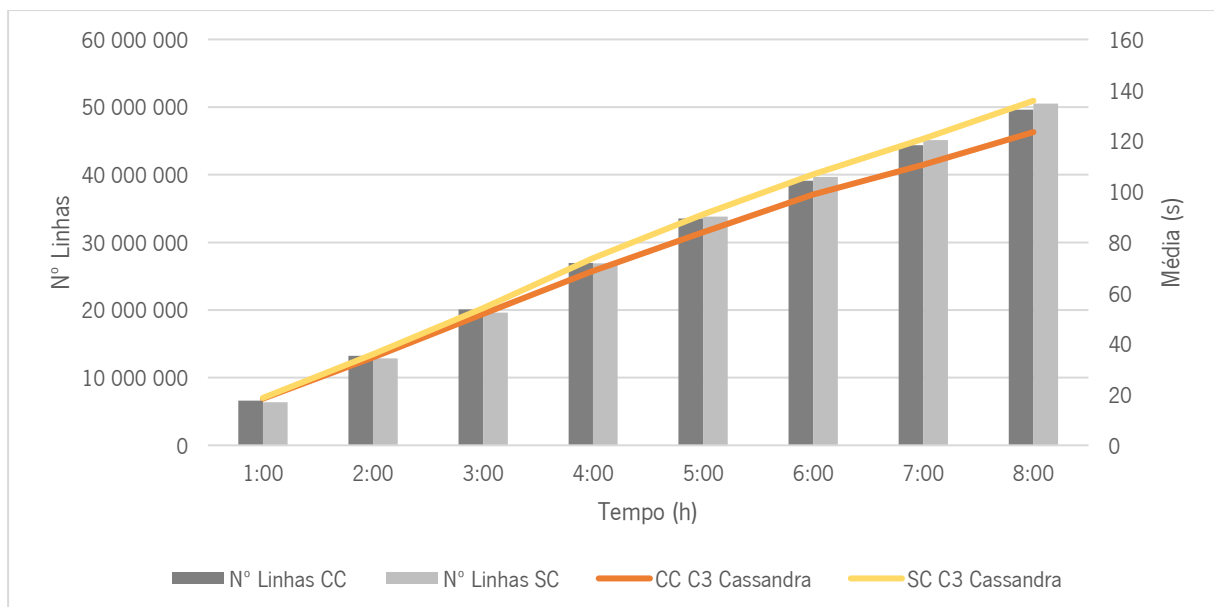


Gráfico 10 - Resultados da Consulta 3 (C3) para a tabela com chave de *clustering* (CC) e sem chave de *clustering* (SC)

Pelos resultados obtidos, no Gráfico 9 e Gráfico 10, é possível afirmar que os resultados não foram otimizados na tabela com chave de *clustering*, para nenhuma das consultas. Novamente, em ambas as consultas, verifica-se que o número de linhas nas duas tabelas não é igual, mas isto deve-se ao facto de o fluxo não ser exatamente o mesmo. O facto da tabela com chave de *clustering* não evidenciar melhores resultados deve-se ao conector do Presto ao repositório Cassandra percorrer, em qualquer uma destas consultas, todos os registos da tabela, pelo que, para isso não acontecer a consulta terá que filtrar pela chave de partição apenas ou pela chave de partição e pela chave de *clustering*. Esta informação foi obtida como conhecimento prático, pelo que isto se verifica na versão do Presto instalada,

nomeadamente versão 0.180, o que não invalida que no futuro hajam alterações. Contudo, dado o contexto deste caso de demonstração, não faz sentido a chave de partição ser filtrada pois esta é o identificador da tabela, ou seja, esta identifica inequivocamente cada uma das linhas²⁵. Quanto à chave de *clustering*, o atributo já foi filtrado na C2 e C3, mas estas não apresentaram melhores resultados, devido a esta apenas apresentar melhores resultados quando filtrada juntamente com a chave de partição. Porém, como exemplo para se perceber o impacto destas chaves nas pesquisas, foi realizada uma consulta com uma chave de partição específica, nomeadamente um só valor identificador da linha, pelo que se obtém como resultado menos de 1 segundo de resposta, o que é um tempo de execução da consulta muito menor comparativamente aos tempos das consultas 1, 2 e 3, considerando o volume de dados armazenado (consulta efetuada ao fim de 8 horas de dados). Contudo, o resultado desta consulta com uma chave de partição específica é apenas uma linha de entre todos os registos da tabela. Como este tipo de consultas não se encaixa no contexto deste caso de demonstração, é possível afirmar que, neste caso, é indiferente a utilização de chave de *clustering*, pois não existe melhoria no desempenho com a sua utilização.

Além das consultas apresentadas, na Tabela 3 encontram-se os resultados das consultas 4, 5, 6, 7 e 8, também realizadas para o Hive, tal como se apresenta na secção anterior. Comparando os resultados das consultas é possível afirmar que os tempos de resposta na tabela sem chave de *clustering* não diferem muitos dos da tabela com chave de *clustering*.

Tabela 3 - Resultados das consultas 4, 5, 6, 7 e 8 na tabela com chave de *clustering* (CC) e tabela sem chave de *clustering* (SC) no Cassandra

Hora	Nº Linhas CC	CC C4 Cassandra (s)	CC C5 Cassandra (s)	CC C6 Cassandra (s)	CC C7 Cassandra (s)	CC C8 Cassandra (s)
01:00	6595821	16,937	16,635	18,288	16,903	18,045
02:00	13230315	31,439	32,512	35,047	32,601	34,946
03:00	20103095	47,542	49,220	52,296	49,491	51,611
04:00	26947993	64,661	66,643	69,589	65,541	69,893
05:00	33525692	79,148	79,369	85,053	79,977	84,283
06:00	39098811	93,223	93,667	99,277	92,972	98,830
07:00	44365732	104,883	106,240	113,283	108,837	112,489
08:00	49625850	118,612	118,972	127,657	120,939	127,025

²⁵ Face a esta situação, poderia ser criada uma outra chave de partição, ou até ser mais do que um atributo a chave de partição. Contudo, nesta dissertação, devido à limitação da taxa de transferência de *tweets* (como referido na secção 5.1), os *tweets* extraídos são replicados, pelo que para a identificação de cada uma das linhas da tabela tem que ser usado um atributo criado como identificador e não pode ser usado o identificador natural do *tweet* ou outros atributos do *tweet*. Desta forma, tem que ser necessariamente criado um atributo como identificador da linha, para garantir que, na inserção de dados, todas as linhas são armazenadas no Cassandra.

Hora	Nº Linhas SC	SC C4 Cassandra (s)	SC C5 Cassandra (s)	SC C6 Cassandra (s)	SC C7 Cassandra (s)	SC C8 Cassandra (s)
01:00	6364603	17,921	17,646	18,109	17,729	17,480
02:00	12867015	33,746	34,550	35,086	35,187	35,282
03:00	19627897	51,918	52,710	53,316	52,970	52,286
04:00	26892926	70,901	71,713	72,813	72,553	71,519
05:00	33803095	86,868	89,379	89,707	89,522	89,547
06:00	39700243	102,384	103,128	105,244	105,074	104,513
07:00	45125812	118,137	120,064	119,559	120,167	118,989
08:00	50529723	131,719	134,442	133,959	134,675	131,847

Desta forma, observando os resultados apresentados nas várias consultas, é possível afirmar que não se mostra benéfica a utilização da chave de *clustering*, pelo que não existe grande diferença da sua, ou não, utilização, dado o contexto desta dissertação. Neste contexto, a utilização da tabela, apenas com chave de partição, representa uma melhor solução.

De referir que, nesta tecnologia, tanto na tabela com chave de *clustering* como na tabela sem chave de *clustering*, a inserção de dados foi dificultada, contudo, por motivos diferentes do comportamento da tecnologia Hive. No Cassandra, ao contrário do Hive, não existe atraso no processo de escrita pelo facto de existir uma operação de verificação para a escrita dos dados, mas sim pelo facto de, quando esta tecnologia está a ser questionada pelo Presto, as operações de escrita revelam atraso, causando atraso na inserção de dados. O facto de esta tecnologia não apresentar resultados tão rápidos para as consultas como o Hive, leva a que algumas tarefas fiquem em espera no Spark Streaming, atrasando a inserção destes dados no Cassandra. Quando existem consultas aos dados, situação que se encontra ilustrada na Figura 27, o atraso na inserção de dados aumenta até, neste caso, 16 minutos. Contudo, como se pode observar nesta figura, esta situação é compensada nos segundos seguintes, em que o atraso reduz até ao momento em que já não se verifica qualquer atraso.

Além disso, o tempo de processamento, aumenta até 12 segundos, voltando mais tarde ao tempo normal de processamento de cerca de 1 segundo. Estes atrasos, quando existem consultas aos dados, provocam tarefas em espera, como se apresenta na Figura 28, na qual os picos no primeiro gráfico representam os momentos em que a consulta de dados é iniciada e provoca atraso nas tarefas, e no segundo gráfico encontra-se representado o aumento do tempo de processamento das tarefas provocado pela realização de consultas aos dados.

Batch Time	Input Size	Scheduling Delay ^(?)	Processing Time ^(?)	Total Delay ^(?)	Output Ops: Succeeded/Total
2017/10/11 08:59:55	5950 records	1 ms	1 s	1 s	1/1
2017/10/11 08:59:50	7550 records	0 ms	1 s	1 s	1/1
2017/10/11 08:59:45	7050 records	0 ms	1 s	1 s	1/1
2017/10/11 08:59:40	7500 records	0 ms	1 s	1 s	1/1
2017/10/11 08:59:35	6950 records	0 ms	1 s	1 s	1/1
2017/10/11 08:59:30	6200 records	0 ms	1.0 s	1.0 s	1/1
● ● ●					
2017/10/11 08:42:25	7100 records	23 s	1.0 s	24 s	1/1
2017/10/11 08:42:20	7250 records	27 s	1.0 s	28 s	1/1
2017/10/11 08:42:15	6800 records	31 s	1.0 s	32 s	1/1
2017/10/11 08:42:10	7400 records	34 s	1 s	36 s	1/1
2017/10/11 08:42:05	8000 records	38 s	1 s	39 s	1/1
2017/10/11 08:42:00	6300 records	42 s	0.9 s	43 s	1/1
● ● ●					
2017/10/11 08:21:15	9100 records	16 min	10 s	16 min	1/1
2017/10/11 08:21:10	11850 records	16 min	12 s	16 min	1/1
2017/10/11 08:21:05	4250 records	16 min	5 s	16 min	1/1
2017/10/11 08:21:00	7700 records	16 min	8 s	16 min	1/1
2017/10/11 08:20:55	6750 records	16 min	9 s	16 min	1/1
2017/10/11 08:20:50	7300 records	15 min	8 s	16 min	1/1
2017/10/11 08:20:45	5850 records	15 min	7 s	16 min	1/1
● ● ●					
2017/10/11 08:00:30	8350 records	44 s	13 s	57 s	1/1
2017/10/11 08:00:25	8435 records	34 s	16 s	49 s	1/1
2017/10/11 08:00:20	7700 records	22 s	17 s	39 s	1/1
2017/10/11 08:00:15	7000 records	15 s	12 s	27 s	1/1
2017/10/11 08:00:10	9950 records	8 s	11 s	20 s	1/1
2017/10/11 08:00:05	11450 records	0 ms	13 s	13 s	1/1
2017/10/11 08:00:00	7050 records	0 ms	1 s	1 s	1/1
2017/10/11 07:59:55	6750 records	0 ms	1.0 s	1.0 s	1/1
2017/10/11 07:59:50	7000 records	0 ms	1 s	1 s	1/1
2017/10/11 07:59:45	7900 records	0 ms	1 s	1 s	1/1

Figura 27 - Atraso no Spark Streaming na inserção de dados no Cassandra

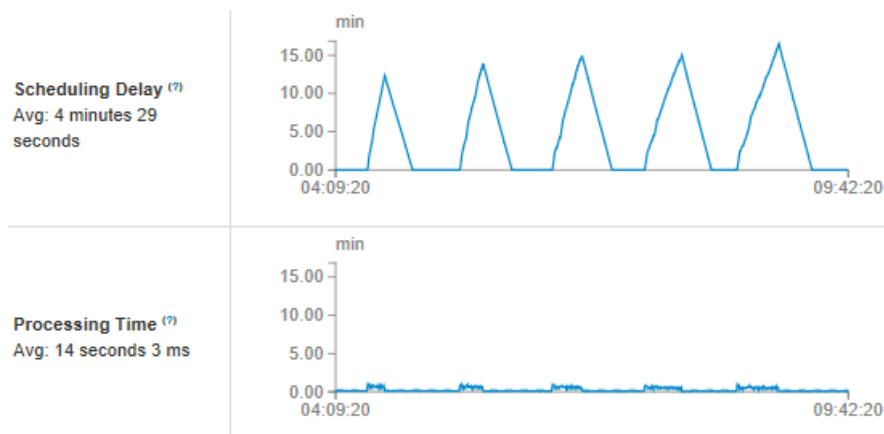


Figura 28 - Espera no Spark Streaming

5.3.3. Hive *versus* Cassandra no Armazenamento de Dados em Tempo Real

A comparação da tecnologia Hive e Cassandra é relevante para perceber a que melhor se enquadra como área de armazenamento de dados em tempo real. Neste sentido, é utilizada uma tabela Hive particionada e uma tabela Cassandra sem chave de *clustering*, devido a estas representarem uma melhor opção, como referido nas seções anteriores, nomeadamente 5.3.1 e 5.3.2. A comparação do desempenho das tecnologias permite conhecer o seu comportamento ao longo do tempo, à medida que aumenta o volume de dados.

Para isso, são comparados os resultados das mesmas consultas efetuadas às duas áreas de armazenamento, de forma a conhecer o seu comportamento, sendo que, de seguida, são debatidos os resultados obtidos.

No Gráfico 11 apresenta-se uma comparação dos resultados obtidos na consulta 1 (C1) na tabela Hive e Cassandra, pelo que a tabela particionada do Hive apresenta melhores resultados.

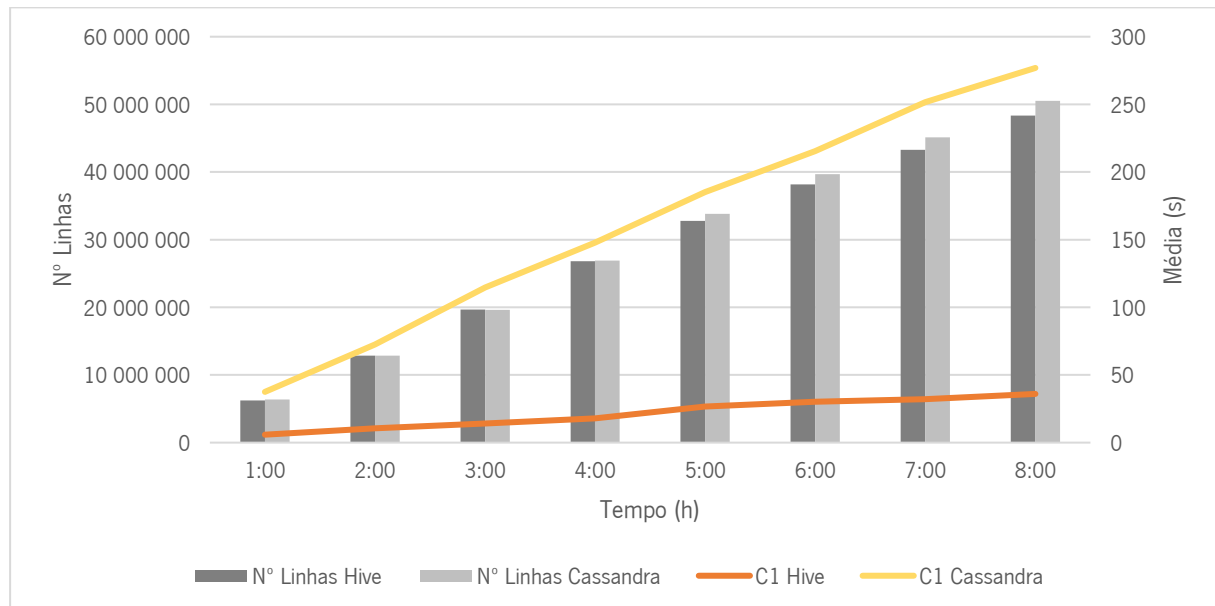


Gráfico 11 - Resultados da Consulta 1 (C1) para tabela Hive e Cassandra

Como se observa nos resultados obtidos, no Gráfico 11, o desempenho do Hive é muito superior ao do Cassandra, pelo que analisando os resultados ao longo do tempo, é possível afirmar que o Hive apresenta uma melhor capacidade de lidar com grandes volumes de dados. Com o aumento do volume de dados, o Cassandra apresenta cada vez menos rapidez na resposta a consultas de dados, motivando a necessidade de os dados serem movidos para outra área de armazenamento que possibilite uma análise rápida aos dados. De referir que a diferença no número de linhas entre o Cassandra e o Hive se deve, como já referido no 5.3, às variações do fluxo de dados do momento em que são extraídos os dados do Twitter.

Na consulta 2 (C2), representada no Gráfico 12, a diferença entre o desempenho do Hive e do Cassandra é mais acentuada, principalmente pelo facto de o Hive apresentar um desempenho semelhante ao longo do tempo. Comparativamente à C1, o comportamento do Hive é constante ao longo do tempo, o que se deve ao facto de esta consulta ser uma contagem dos registos da última hora, pelo que o Hive apenas percorre os registos da partição da hora anterior e o Cassandra percorre todos os registos. Contudo, como referido na 5.3.2, isto deve-se ao facto de, no contexto deste trabalho, não se conseguir utilizar aquelas que se afirmam ser as potencialidades do Cassandra que lhe permitem um melhor desempenho, o que não invalida de melhores resultados do Cassandra noutra contexto em que as suas especificidades se evidenciem. Apesar de o desempenho do Cassandra ser inferior ao do Hive, é de realçar que o tempo máximo desta consulta do Cassandra é menor do que os resultados

apresentados no Gráfico 11, na consulta 1, pelo que se pode afirmar que consultas que necessitam de fazer a contagem de todos os registos se verificam mais custosas para o Cassandra.

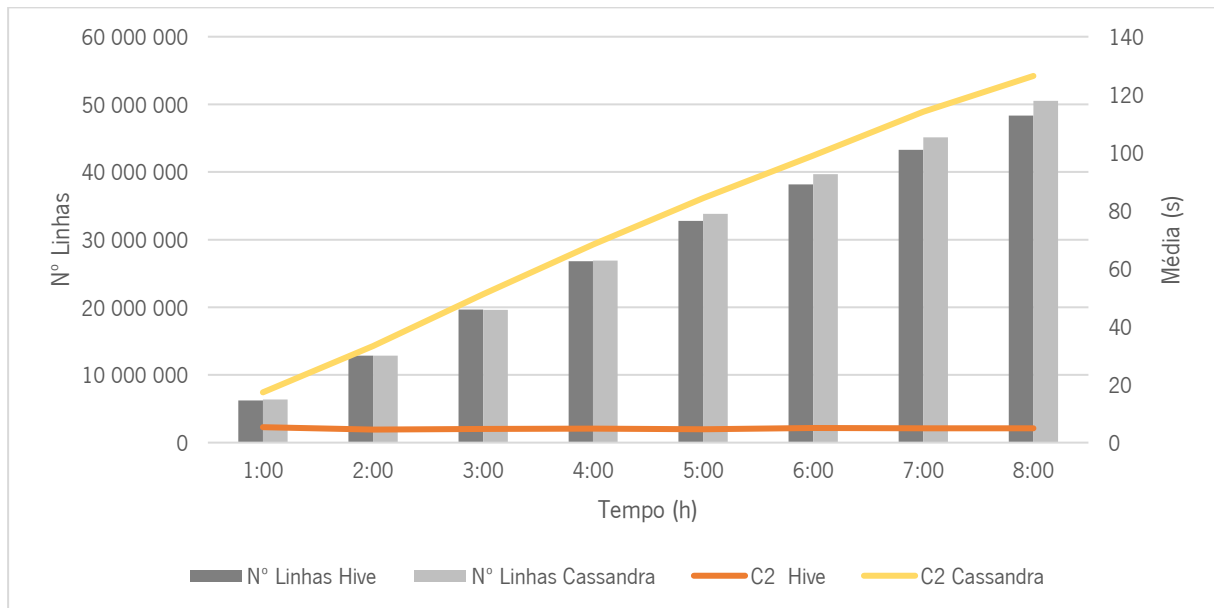


Gráfico 12 - Resultados da Consulta 2 (C2) para tabela Hive e Cassandra

A consulta 3 (C3), cujos resultados se encontram no Gráfico 13, apresenta um comportamento muito semelhante ao da consulta 2 apresentado no Gráfico 12, o que se deve a ambas as consultas serem filtradas pela hora do evento. Nesta consulta o Hive evidencia novamente o seu desempenho pela sua organização em partições por hora, sendo uma consulta que otimiza os resultados do Hive, desempenho esse que o Cassandra não consegue alcançar.

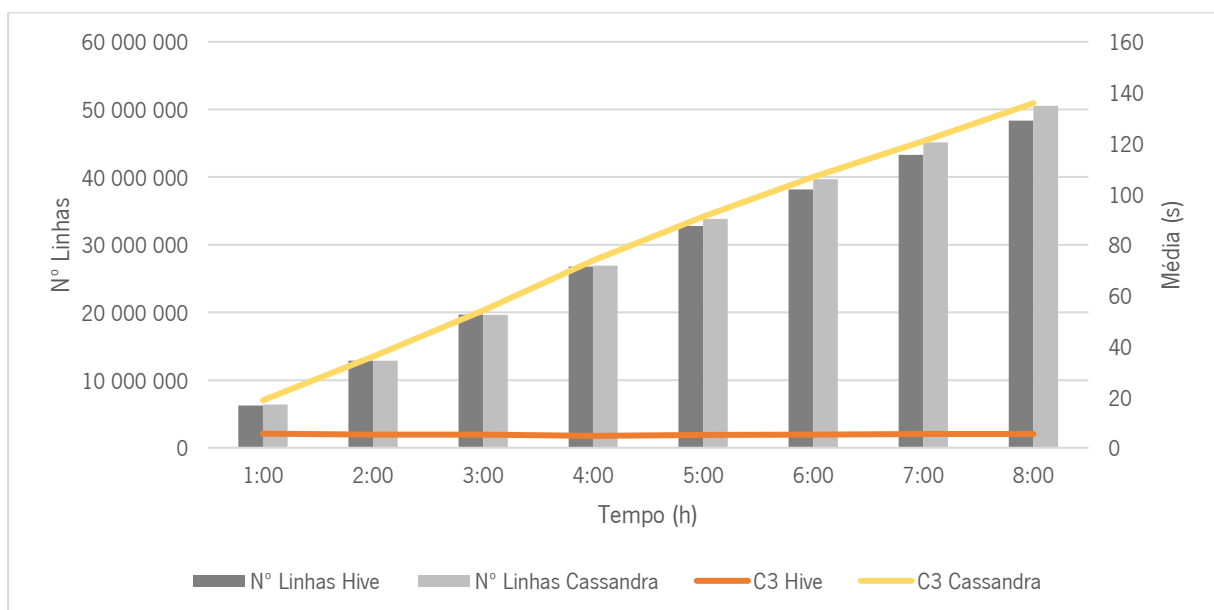


Gráfico 13 - Resultados da Consulta 3 (C3) para tabela Hive e Cassandra

O comportamento do Cassandra face ao Hive, na consulta 4, apresentada no Gráfico 14, é semelhante aos apresentados nas anteriores consultas desta secção, pelo que mesmo quando a consulta não evidencia as partições do Hive este apresenta um melhor desempenho comparativamente ao Cassandra.

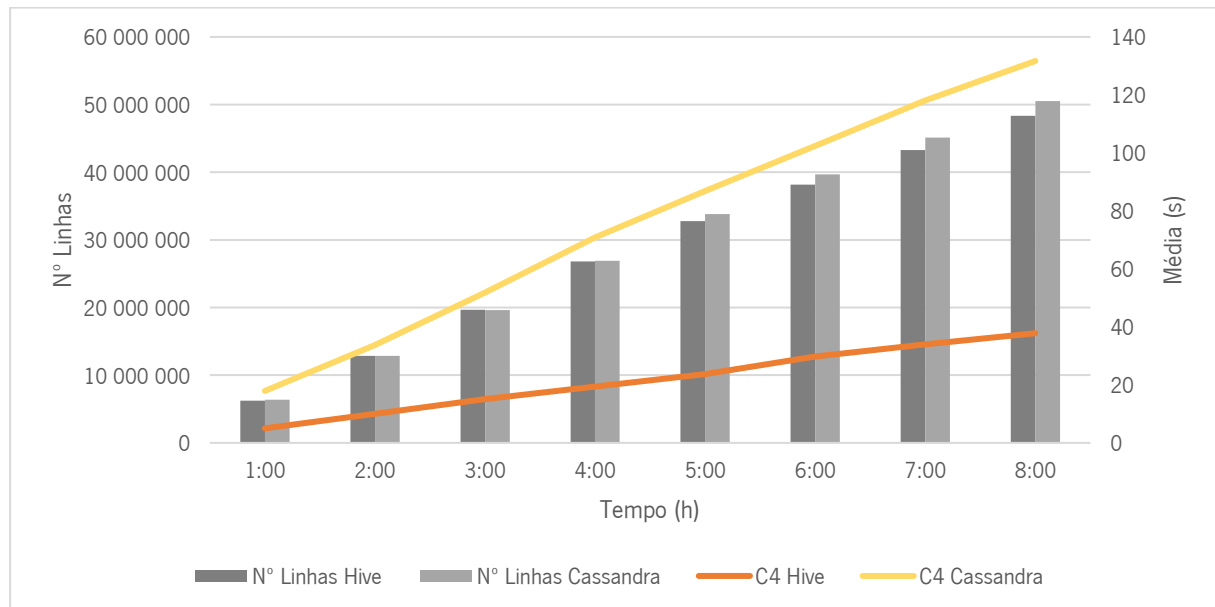


Gráfico 14 - Resultados da Consulta 4 (C4) para tabela Hive e Cassandra

Comparando o desempenho das duas tecnologias para armazenamento de dados em tempo real, o Hive destaca-se com melhor desempenho em todas as consultas. Apesar de as principais características do Hive não serem a rápida leitura e escrita, este, neste contexto, apresenta um bom comportamento, destacando-se pelas suas capacidades na análise de dados. Além disso, considerando que o esperado é que uma consulta não ultrapasse, de uma forma geral, os 60 segundos, no contexto deste trabalho, os dados armazenados no Cassandra teriam que ser mais rapidamente movidos para um armazenamento histórico, de forma a não prejudicar a análise em tempo real aos dados. O aumento do volume de dados, mais rapidamente no caso do Cassandra, causa a necessidade de os dados serem movidos devido ao volume deixar de ser suportável para que a análise atempada seja garantida. Porém, esta questão do tempo máximo esperado por uma consulta, dependerá de cada organização, de acordo também com as consultas que se pretendem aos dados em tempo real, tendo que ser desenhada a solução para cada caso, de forma a estar o mais otimizada possível.

Contudo, se no cenário em questão, se estivesse a lidar com questões de atualização de dados, acredita-se que o Cassandra poderia ter melhor desempenho, pois uma das suas principais características é a rápida atualização dos dados já armazenados. Além disso, comparando o tempo de

processamento do Cassandra com Hive, o Cassandra apresenta cerca de 1 segundo e o Hive cerca de 3 segundos de tempo de processamento, pelo que, num contexto em que se desejasse acesso rápido aos dados que acabam de ser gerados, sem grande volume de dados, o Cassandra poderia ser uma solução. Contudo, neste caso, há que ter em conta que as consultas aos dados serão mais lentas, pelo que o ideal é, dependendo do negócio, exista um equilíbrio entre o que é desejado e o que as tecnologias oferecem ou até a escolha de várias tecnologias para contextos diferentes, interligando os seus dados.

De forma a avaliar o desempenho da tecnologia Hive e Cassandra noutros cenários, é testado o comportamento destas tecnologias com um *micro-batch* do Spark Streaming de duração diferente bem como com um volume de dados diferente, para conhecer o comportamento destas tecnologias. De seguida apresentam-se os resultados para estes testes. De referir que a diferença no volume de dados nos vários testes se deve a estes terem sido realizados em momentos diferentes, pelo que o fluxo de dados apresenta uma variação.

Para a análise do impacto de um *micro-batch* do Spark Streaming diferente no comportamento das tecnologias, foi definida a duração de 20 segundos, em vez de 5 segundos, como nos testes apresentados até ao momento. O Hive com o *micro-batch* de 20 segundos, Gráfico 15, apresenta um melhor desempenho nas várias consultas em relação ao desempenho do *micro-batch* de 5 segundos, Gráfico 16. Assim, é possível afirmar que, com a tecnologia Hive, quanto menor for o *micro-batch* do Spark Streaming, pior será o desempenho da tecnologia nas consultas.

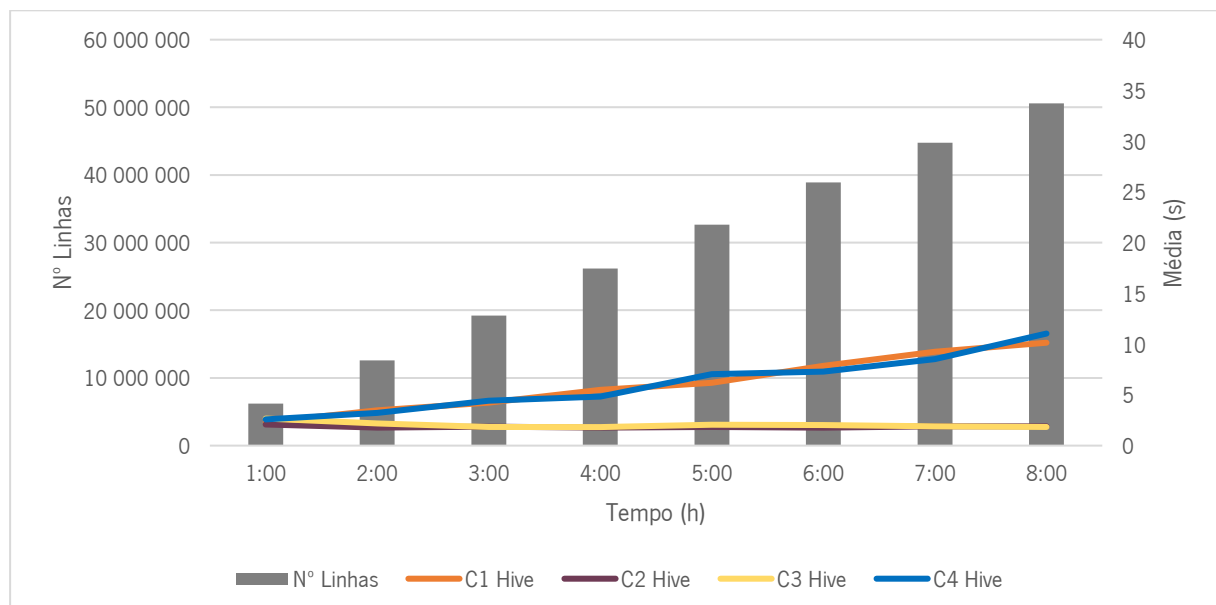


Gráfico 15 - Resultados Hive com *micro-batch* de 20 segundos

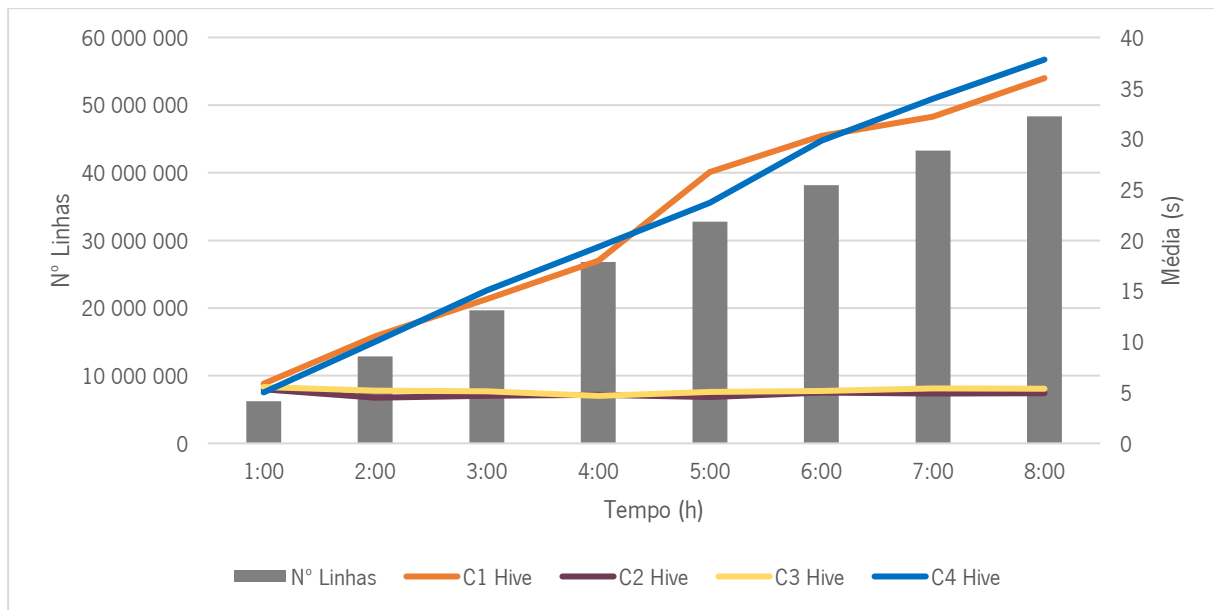


Gráfico 16 - Resultados Hive com *micro-batch* de 5 segundos

Esta diferença de desempenho, apresentada no Gráfico 15 e Gráfico 16, deve-se ao facto de, quando o *micro-batch* apresenta uma duração maior, existir menos sobrecarga na escrita, ou seja, é inserida uma quantidade de dados maior de uma só vez, representando um número inferior de inserções, nomeadamente menos ficheiros criados. O facto de na área de armazenamento de dados em tempo real existirem menos ficheiros, leva a que os dados estejam mais concatenados, o que agilizará a consulta de dados, de acordo com o funcionamento do Hive. Deste modo, sugere-se a necessidade de os dados serem movidos, ao fim de algumas horas, para o armazenamento de dados históricos, pois neste, mesmo estando armazenado um grande volume de dados, os dados encontram-se ainda mais concatenados, permitindo consultas de melhor desempenho a grandes quantidades de dados.

O desempenho do Cassandra no *micro-batch* de 20 segundos, que se apresenta no Gráfico 17, é semelhante ao *micro-batch* de 5 segundos, apresentado no Gráfico 18. Desta forma, a duração do *micro-batch* não influencia o desempenho da tecnologia, pelo que deve ser analisado o contexto do negócio e o tempo de processamento para a definição do *micro-batch* do Spark Streaming. No *micro-batch* de 20 segundos é também verificada a situação de atraso quando os dados são questionados pelo Presto, cenário já verificado no *micro-batch* de 5 segundos, como apresentado na secção anterior, nomeadamente 5.3.2.

Na Figura 29 apresenta-se ilustrado o atraso provocado pelas consultas de dados, causando um atraso até 17 minutos e atingindo o tempo de processamento os 52 segundos. Contudo, esta situação é de alguma forma compensada nos segundos seguintes, até voltar novamente de 4 a 5 segundos de tempo de processamento e sem atraso nas tarefas.

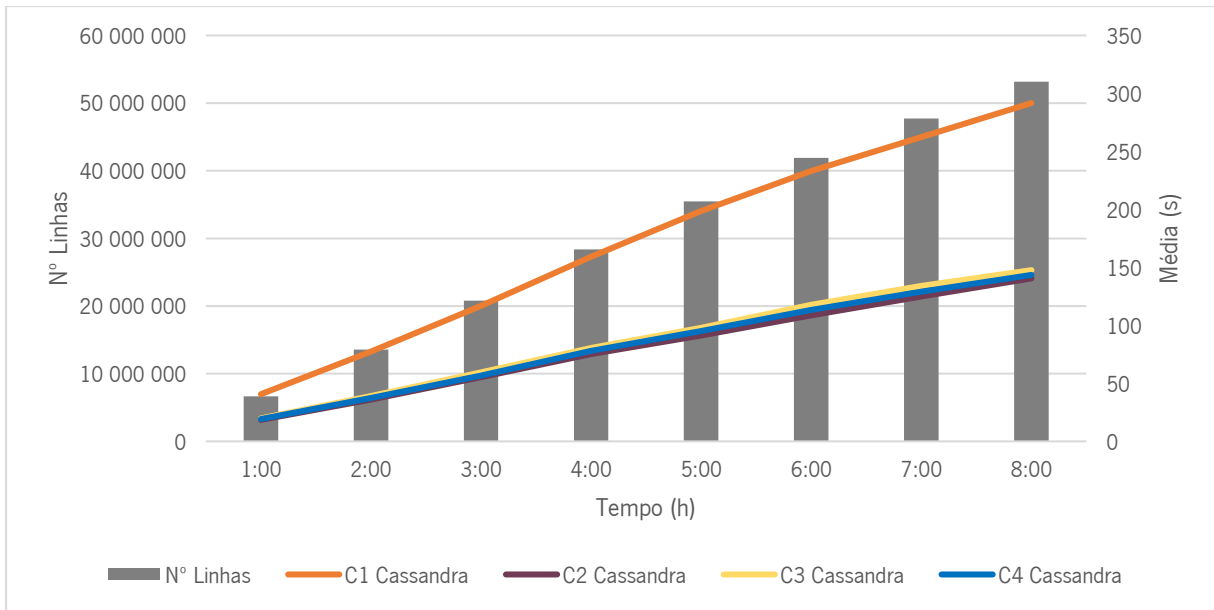


Gráfico 17 - Resultados Cassandra com *micro-batch* 20 segundos

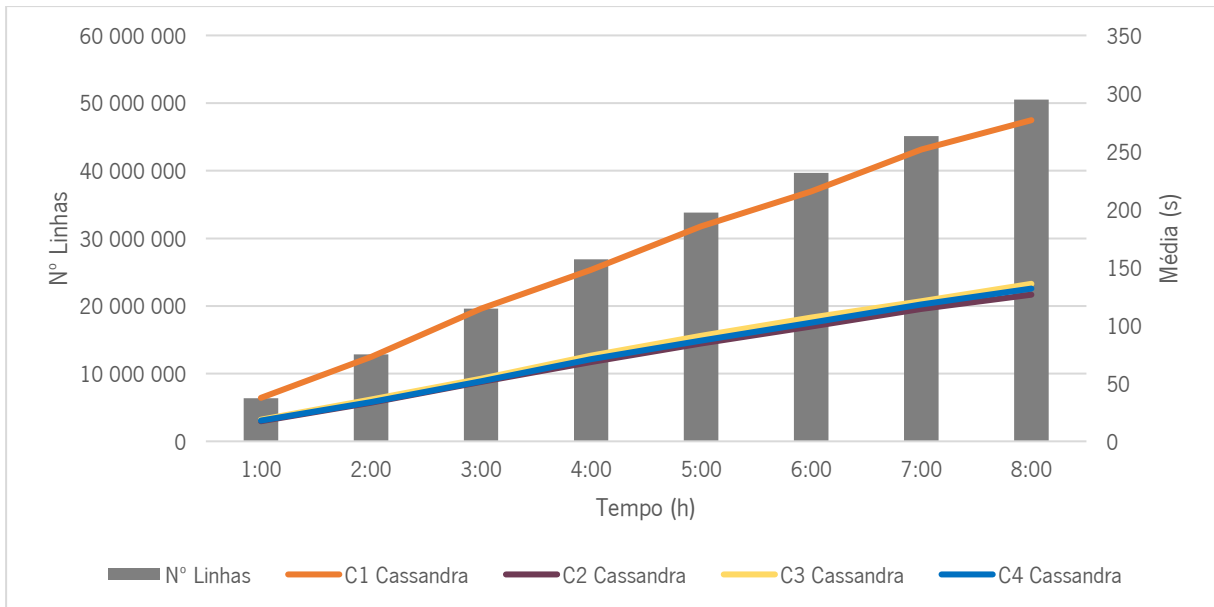


Gráfico 18 - Resultados Cassandra com *micro-batch* de 5 segundos

Batch Time	Input Size	Scheduling Delay ⁽¹⁾	Processing Time ⁽²⁾	Total Delay ⁽³⁾	Output Ops: Succeeded/Total
2017/10/09 09:42:20	31371 records	0 ms	5 s	5 s	1/1
2017/10/09 09:42:00	31163 records	0 ms	5 s	5 s	1/1
2017/10/09 09:41:40	30600 records	0 ms	5 s	5 s	1/1
2017/10/09 09:41:20	31483 records	0 ms	5 s	5 s	1/1
2017/10/09 09:41:00	28217 records	0 ms	4 s	4 s	1/1
● ● ● ●					
2017/10/09 09:17:00	27900 records	1.4 min	4 s	1.5 min	1/1
2017/10/09 09:16:40	29850 records	1.7 min	5 s	1.8 min	1/1
2017/10/09 09:16:20	33100 records	1.9 min	5 s	2.0 min	1/1
2017/10/09 09:16:00	29150 records	2.2 min	5 s	2.3 min	1/1
2017/10/09 09:15:40	29600 records	2.4 min	4 s	2.5 min	1/1
2017/10/09 09:15:20	34200 records	2.7 min	5 s	2.8 min	1/1
2017/10/09 09:15:00	29800 records	2.9 min	4 s	3.0 min	1/1
2017/10/09 09:14:40	29650 records	3.2 min	4 s	3.3 min	1/1
2017/10/09 09:14:20	32400 records	3.5 min	5 s	3.5 min	1/1
● ● ● ●					
2017/10/09 09:00:00	28428 records	14 min	4 s	14 min	1/1
2017/10/09 08:59:40	29750 records	15 min	4 s	15 min	1/1
2017/10/09 08:59:20	30050 records	15 min	4 s	15 min	1/1
2017/10/09 08:59:00	27450 records	15 min	4 s	15 min	1/1
2017/10/09 08:58:40	28900 records	15 min	4 s	16 min	1/1
2017/10/09 08:58:20	29075 records	16 min	4 s	16 min	1/1
2017/10/09 08:58:00	28325 records	16 min	4 s	16 min	1/1
2017/10/09 08:57:40	25950 records	16 min	3 s	16 min	1/1
2017/10/09 08:57:20	29850 records	17 min	5 s	17 min	1/1
2017/10/09 08:57:00	28150 records	16 min	29 s	17 min	1/1
2017/10/09 08:56:40	29300 records	16 min	34 s	17 min	1/1
2017/10/09 08:56:20	31300 records	16 min	32 s	16 min	1/1
2017/10/09 08:56:00	29000 records	16 min	35 s	16 min	1/1
2017/10/09 08:55:40	28150 records	16 min	23 s	16 min	1/1
● ● ● ●					
2017/10/09 08:31:40	31650 records	2.0 min	45 s	2.8 min	1/1
2017/10/09 08:31:20	32500 records	1.6 min	44 s	2.4 min	1/1
2017/10/09 08:31:00	34000 records	1.1 min	52 s	2.0 min	1/1
2017/10/09 08:30:40	31950 records	41 s	45 s	1.4 min	1/1
2017/10/09 08:30:20	33350 records	7 s	54 s	1.0 min	1/1
2017/10/09 08:30:00	29700 records	0 ms	27 s	27 s	1/1
2017/10/09 08:29:40	28650 records	0 ms	5 s	5 s	1/1
2017/10/09 08:29:20	30900 records	0 ms	6 s	6 s	1/1

Consulta aos dados

Figura 29 - Atraso no Spark Streaming na inserção de dados no Cassandra no *micro-batch* de 20 segundos

Nos resultados obtidos para as duas tecnologias encontram-se situações muito distintas, pelo que o comportamento do Hive com o aumento da duração do *micro-batch* permitiu estes melhores resultados, porém, no caso do Cassandra, esta situação não se revelou muito benéfica. Devido às tarefas de escrita não estarem tão sobrecarregadas, esperava-se que em ambos os casos se obtivessem melhores resultados.

Tendo em vista a análise do comportamento das tecnologias com um volume diferente, efetua-se um teste para cada uma das tecnologias com dados a fluir em *streaming*, mas com o dobro do volume dos testes anteriormente realizados, ou seja, cerca de 3000 mensagens a fluir por segundo, e outro teste com metade do volume dos testes anteriormente realizados, nomeadamente cerca de 800 mensagens a fluir por segundo. Desta forma, é averiguado o comportamento com o dobro do volume (V2) apresentado nos restantes testes (V1) assim como com metade do volume (V3), de forma a perceber se a infraestrutura proposta está preparada para um volume de dados superior ou até inferior.

No caso do Hive, apresentando-se os resultados no Gráfico 19, comparando o volume dos restantes testes (V1) com o dobro do volume (V2), nas consultas 1 e 2, é visível que o tempo necessário para responder às consultas é mais ou menos o mesmo, mesmo quando o volume de dados é superior.

Sendo assim, através destes resultados, é possível verificar a capacidade do Hive de lidar com grandes volumes de dados, pelo que, seja qual for o volume (V1 ou V2), este mantém o seu desempenho.

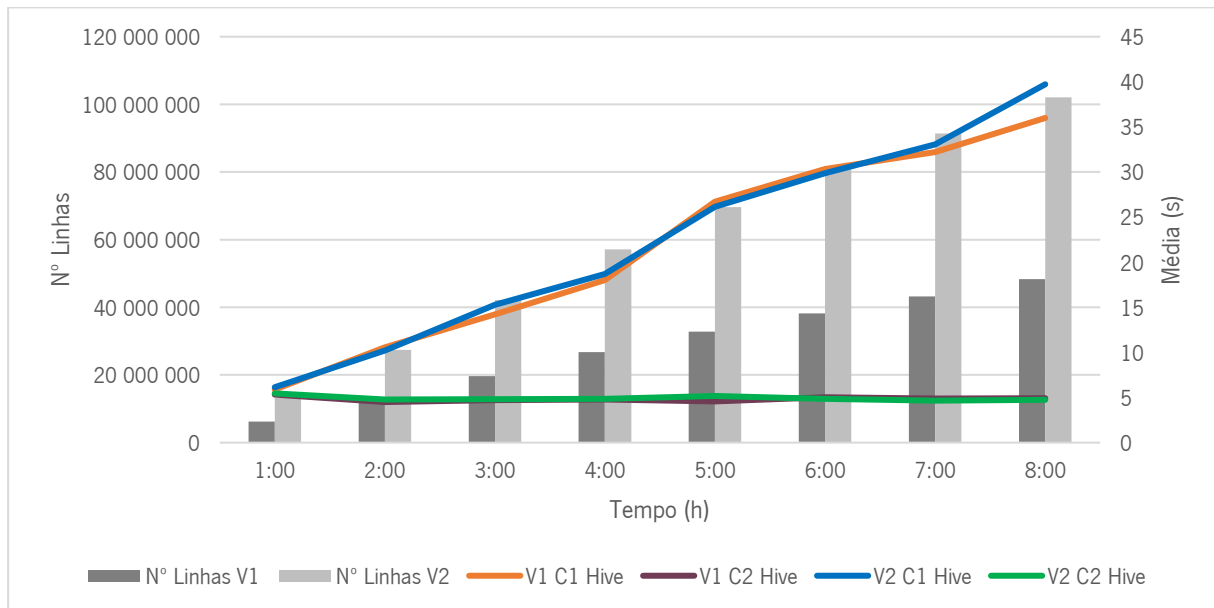


Gráfico 19 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V2) no Hive

O facto de o comportamento do Hive se manter deve-se a, com um volume de dados superior, nomeadamente V2, serem, no processo de recolha, processamento e armazenamento, gerados um número de ficheiros semelhante no armazenamento de dados em tempo real. Estes ficheiros, como se pode observar na Figura 30, serão ficheiros de maior dimensão, pois possuem uma maior quantidade de registos.

Name >	Size >	Last Modified >	Owner >	Group >	Permission
part-00002	34.0 kB	2017-10-12 14:10	lid4	hdfs	-rwxrwxrwx
part-00000	33.5 kB	2017-10-12 14:10	lid4	hdfs	-rwxrwxrwx
part-00000_copy_10	50.6 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_11	45.9 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_12	52.0 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_13	50.1 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_14	48.0 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx

Ficheiros de V1

Name >	Size >	Last Modified >	Owner >	Group >	Permission
part-00002_copy_1	67.5 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_10	72.0 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_11	62.2 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_2	63.2 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_3	72.0 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_4	68.4 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_5	73.4 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx
part-00002_copy_6	76.1 kB	2017-10-12 14:36	lid4	hdfs	-rwxrwxrwx

Ficheiros de V2

Figura 30 - Tamanho dos ficheiros de dois volumes de dados diferentes no Hive

Desta forma, nas consultas efetuadas pelo Presto à tabela do Hive, as dificuldades são as mesmas apresentadas para V1, pelo que o tempo de resposta às consultas é muito semelhante. Assim, o tempo de resposta às consultas está não só relacionado com o volume de dados, mas, neste caso, principalmente relacionado com o número de ficheiros criados no armazenamento do Hive, ou seja, com o processo de armazenamento de dados, que neste caso está intimamente relacionado com o *micro-batch* do Spark Streaming.

O comportamento da tecnologia Cassandra, apresentado no Gráfico 20, distingue-se do apresentado pelo Hive para o dobro do volume (V2). O Cassandra, com um maior volume de dados, aumenta, de forma geral, o tempo de resposta das consultas. O facto é que a tecnologia não possui capacidade para lidar com grandes volumes de dados, pelo que, ao fim de algumas horas, o tempo de resposta à consulta 1, por exemplo, é de 300 segundos, o que leva a que não sejam esperados melhores resultados por parte desta tecnologia, com um volume de dados como este, nomeadamente V2.

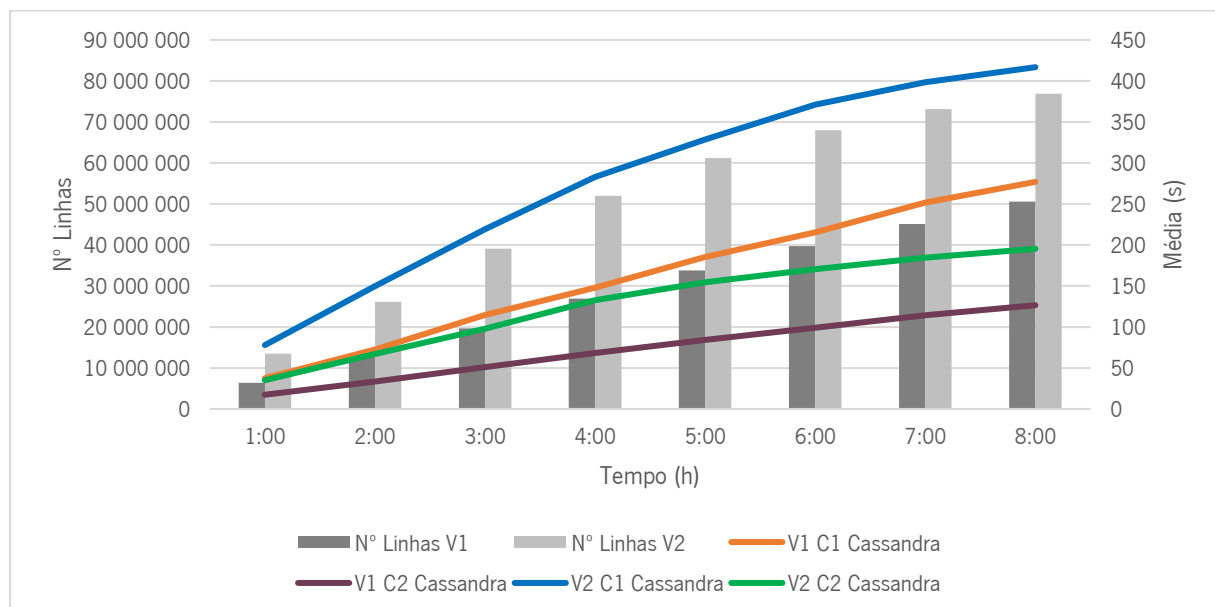


Gráfico 20 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V2) no Cassandra

De referir que, durante o teste com o volume V2, verificou-se um grande atraso nas tarefas do Cassandra, maior do que o registado no V1, referido na secção 5.3.2. Como representado na Figura 31, o atraso na inserção de dados no Spark Streaming chegou a 3,5 horas, não se concretizando assim o fluxo dos dados em tempo real, sendo que, neste caso este valor de atraso não é compensado nos segundos seguintes, pelo que, ao fim de algumas horas de fluxo de dados, este atraso aumenta cada vez mais. Além do atraso, também é verificado o aumento do tempo de processamento, devido a estarem a ser efetuadas consultas ao mesmo tempo que os dados são inseridos.

Batch Time	Input Size	Scheduling Delay ⁽¹⁾	Processing Time ⁽¹⁾	Output Ops: Succeeded/Total	Status
2017/10/12 09:10:30	15859 records	-	-	0/1	queued
2017/10/12 09:10:25	16841 records	-	-	0/1	queued
2017/10/12 09:10:20	18204 records	-	-	0/1	queued
2017/10/12 09:10:15	16896 records	-	-	0/1	queued
2017/10/12 09:10:10	18200 records	-	-	0/1	queued
● ● ● ●					
2017/10/12 05:40:20	17900 records	-	-	0/1	queued
2017/10/12 05:40:15	16500 records	3.5 h	-	0/1	processing

Completed Batches (last 1000 out of 4460)

Batch Time	Input Size	Scheduling Delay ⁽¹⁾	Processing Time ⁽¹⁾	Total Delay ⁽¹⁾	Output Ops: Succeeded/Total
2017/10/12 05:40:10	18700 records	3.5 h	18 s	3.5 h	1/1
2017/10/12 05:40:05	19760 records	3.5 h	20 s	3.5 h	1/1
2017/10/12 05:40:00	15840 records	3.5 h	17 s	3.5 h	1/1
2017/10/12 05:39:55	19000 records	3.5 h	20 s	3.5 h	1/1
● ● ● ●					
2017/10/12 05:07:40	17200 records	2.1 h	4 s	2.1 h	1/1
2017/10/12 05:07:35	18400 records	2.1 h	21 s	2.1 h	1/1
2017/10/12 05:07:30	22400 records	2.1 h	23 s	2.1 h	1/1
2017/10/12 05:07:25	18600 records	2.1 h	18 s	2.1 h	1/1
2017/10/12 05:07:20	23040 records	2.1 h	31 s	2.1 h	1/1
● ● ● ●					
2017/10/12 04:39:35	19200 records	1.1 h	22 s	1.1 h	1/1
2017/10/12 04:39:30	21900 records	1.1 h	27 s	1.1 h	1/1
2017/10/12 04:39:25	16100 records	1.1 h	18 s	1.1 h	1/1
2017/10/12 04:39:20	19900 records	1.1 h	22 s	1.1 h	1/1
2017/10/12 04:39:15	22900 records	1.1 h	22 s	1.1 h	1/1
2017/10/12 04:39:10	21600 records	1.0 h	24 s	1.1 h	1/1
● ● ● ●					
2017/10/12 04:23:05	19000 records	58 min	3 s	58 min	1/1
2017/10/12 04:23:00	17900 records	58 min	2 s	58 min	1/1
2017/10/12 04:22:55	19500 records	58 min	3 s	58 min	1/1
2017/10/12 04:22:50	19100 records	58 min	3 s	58 min	1/1
2017/10/12 04:22:45	19900 records	58 min	3 s	58 min	1/1
2017/10/12 04:22:40	19800 records	58 min	3 s	58 min	1/1

Figura 31 - Atraso no Spark Streaming na inserção de dados no Cassandra

Para um menor volume de dados, metade do volume dos testes anteriormente realizados, nomeadamente V3, no caso do Hive, representado no Gráfico 21, o seu comportamento é mais ou menos o mesmo quando o volume de dados é superior. Esta situação apresentada no Hive, encontrada também nos resultados das consultas com V2, deve-se ao facto de o tempo para responder às consultas estar principalmente relacionado com o número de ficheiros no armazenamento de dados em tempo real.

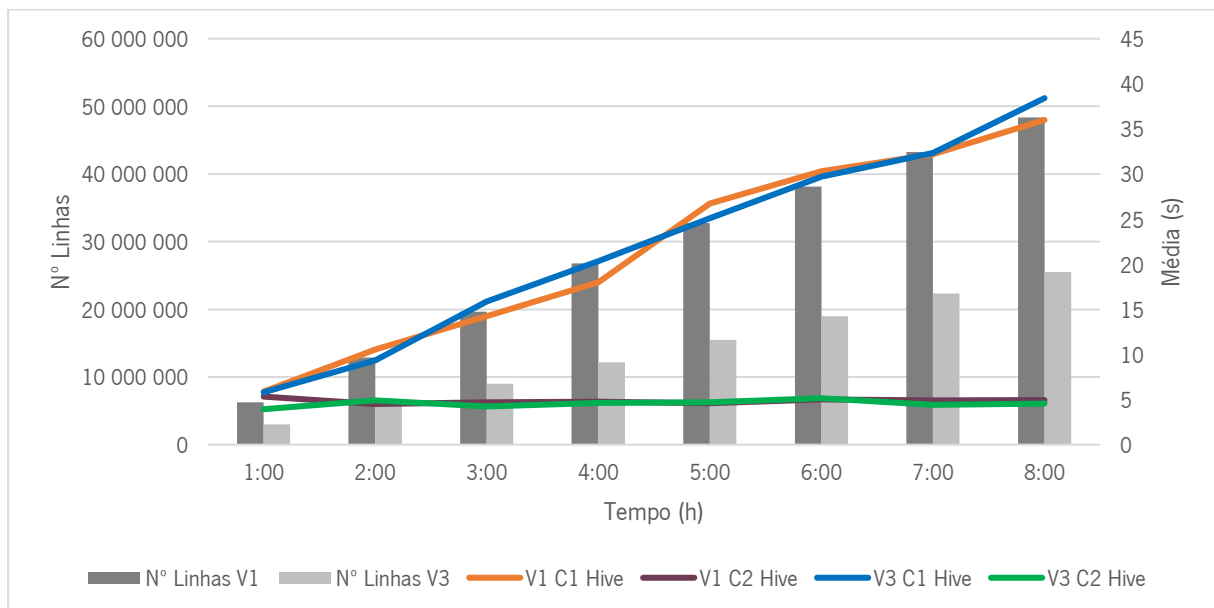


Gráfico 21 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V3) no Hive

De referir que, com menor volume de dados, poderia ser ajustada a duração do *micro-batch* do Spark Streaming, pois, como apresentado na Figura 32, o tempo de processamento é de cerca de 2 segundos, o que se revela inferior a 5 segundos. Contudo, quanto menor for a duração do *micro-batch* do Spark Streaming, no caso do Hive, pior será o desempenho das suas consultas, como já referido no Gráfico 15 e Gráfico 16, pelo que, de acordo com o contexto deve ser balanceada a duração necessária.

Completed Batches (last 1000 out of 8544)

Batch Time	Input Size	Scheduling Delay ^(s)	Processing Time ^(s)	Total Delay ^(s)	Output Ops: Succeeded/Total
2017/10/15 10:20:25	3825 records	0 ms	2 s	2 s	1/1
2017/10/15 10:20:20	4653 records	0 ms	2 s	2 s	1/1
2017/10/15 10:20:15	4247 records	0 ms	2 s	2 s	1/1
2017/10/15 10:20:10	4875 records	1 ms	2 s	2 s	1/1
2017/10/15 10:20:05	4075 records	1 ms	2 s	2 s	1/1
2017/10/15 10:20:00	4900 records	0 ms	2 s	2 s	1/1
2017/10/15 10:19:55	3950 records	0 ms	2 s	2 s	1/1
2017/10/15 10:19:50	4425 records	1 ms	2 s	2 s	1/1
2017/10/15 10:19:45	4975 records	0 ms	2 s	2 s	1/1

Figura 32 - Tempo de processamento no Spark Streaming com volume de dados V3 na tecnologia Hive

Comparando os resultados de V1, V2 e V3 é possível afirmar que o Hive mantém o seu comportamento seja qual for o volume de dados, evidenciando a sua capacidade de lidar com grandes volumes de dados. Além disso, de acordo com o desempenho apresentado por esta tecnologia, o tempo de resposta às consultas está relacionado principalmente com o número de ficheiros criados na inserção de dados e armazenados no armazenamento de dados em tempo real e não necessariamente com o volume de dados que flui em *streaming* nem o tamanho dos ficheiros. A melhoria destes resultados passa pelo aumento do *micro-batch* do Spark Streaming, de forma a serem criados um número menor de ficheiros na inserção de dados, ou a transferência destes dados para outro armazenamento com alguma periodicidade.

No caso do Cassandra, para metade do volume de dados, nomeadamente V3, o tempo de resposta às consultas apresenta melhor desempenho, como representado no Gráfico 22. Contudo, ao contrário do verificado no V2, com este volume de dados, esta tecnologia não apresenta atrasos na consulta de dados. Como se pode observar pela Figura 33, quando o Cassandra é questionado ao mesmo tempo que estão a ser inseridos dados, o seu tempo de processamento aumenta, assim como o atraso para a inserção de dados. Apesar deste atraso, ao fim de alguns minutos, este diminui voltando ao tempo de processamento habitual de menos de 1 segundo e não apresentando praticamente espera para a inserção de dados. De referir que neste cenário poderia ser ajustada a duração do *micro-batch* do Spark Streaming, pois o tempo de processamento de menos de 1 segundo é substancialmente inferior aos 5 segundos, devido ao Cassandra apresentar concorrência na escrita. Ao contrário da tecnologia Hive, com o aumento da duração do *micro-batch* do Spark Streaming, o desempenho do Cassandra é semelhante, pelo que poderia ser adequado ao contexto em questão, requerendo uma avaliação do possível atraso na inserção de dados.

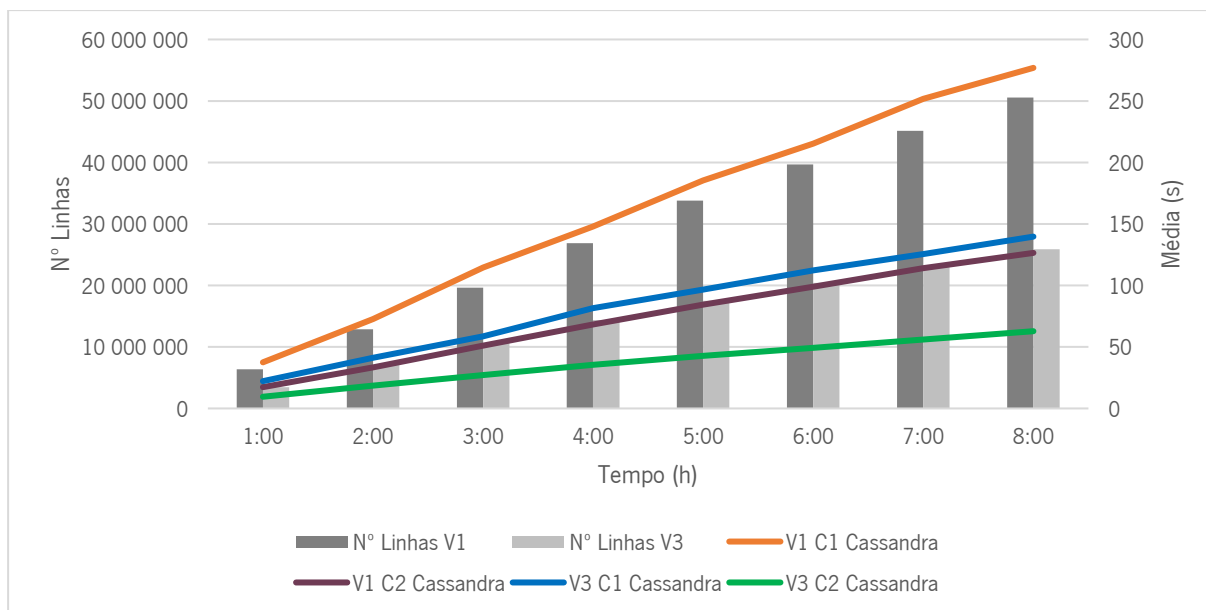


Gráfico 22 - Resultados da Consulta 1 (C1) e 2 (C2) para dois volumes de dados diferentes (V1 e V3) no Cassandra

2017/10/17 09:21:20	4348 records	0 ms	0,7 s	0,7 s	1/1
2017/10/17 09:21:15	3702 records	0 ms	0,6 s	0,6 s	1/1
2017/10/17 09:21:10	4325 records	0 ms	0,8 s	0,8 s	1/1
2017/10/17 09:21:05	3700 records	0 ms	0,6 s	0,6 s	1/1
2017/10/17 09:21:00	3650 records	1 ms	0,5 s	0,5 s	1/1
2017/10/17 09:20:55	3450 records	0 ms	0,5 s	0,5 s	1/1
2017/10/17 09:20:50	3600 records	1 ms	0,6 s	0,6 s	1/1
...					
2017/10/17 09:01:25	4500 records	36 s	9 s	44 s	1/1
2017/10/17 09:01:20	4100 records	34 s	7 s	41 s	1/1
2017/10/17 09:01:15	5350 records	29 s	10 s	39 s	1/1
2017/10/17 09:01:10	4725 records	26 s	9 s	34 s	1/1
2017/10/17 09:01:05	4350 records	24 s	7 s	31 s	1/1
2017/10/17 09:01:00	4450 records	21 s	8 s	29 s	1/1
2017/10/17 09:00:55	4775 records	18 s	8 s	26 s	1/1
2017/10/17 09:00:50	4550 records	15 s	8 s	23 s	1/1
2017/10/17 09:00:45	4675 records	13 s	7 s	20 s	1/1
2017/10/17 09:00:40	5075 records	11 s	7 s	18 s	1/1
2017/10/17 09:00:35	4300 records	8 s	7 s	16 s	1/1
2017/10/17 09:00:30	4275 records	7 s	7 s	13 s	1/1
2017/10/17 09:00:25	3625 records	7 s	5 s	12 s	1/1
2017/10/17 09:00:20	4850 records	5 s	6 s	12 s	1/1
2017/10/17 09:00:15	4475 records	3 s	7 s	10 s	1/1
2017/10/17 09:00:10	4375 records	2 s	6 s	8 s	1/1
2017/10/17 09:00:05	5800 records	0 ms	7 s	7 s	1/1
2017/10/17 09:00:00	3275 records	0 ms	1 s	1 s	1/1
2017/10/17 08:59:55	3275 records	1 ms	0,5 s	0,5 s	1/1
2017/10/17 08:59:50	3375 records	0 ms	0,6 s	0,6 s	1/1
2017/10/17 08:59:45	3275 records	0 ms	0,5 s	0,5 s	1/1
2017/10/17 08:59:40	3400 records	0 ms	0,9 s	0,9 s	1/1
2017/10/17 08:59:35	3525 records	0 ms	0,5 s	0,5 s	1/1

Figura 33 - Comportamento da tecnologia Cassandra no processamento no Spark Streaming com volume de dados V3

Fazendo um balanceamento do que é pretendido nesta dissertação e o comportamento da tecnologia Hive e Cassandra, é possível afirmar que a tecnologia Hive satisfaz melhor os requisitos analíticos, dado o contexto da dissertação assim como o projeto de investigação em que se encontra enquadrado.

5.3.4. Hive no Armazenamento de Dados Históricos

Para o armazenamento dos dados históricos é utilizada a tecnologia Hive, devido principalmente a esta possuir a capacidade de lidar com grandes quantidades de dados. A necessidade de os dados serem movidos para o histórico é justificada pelo facto de estes dados terem de ser utilizados em análises com características históricas, mantendo o desempenho na execução das consultas.

À medida que o tempo avança, ou seja, que o volume de dados no armazenamento de dados em tempo real aumenta, como observado nas secções anteriores, o comportamento da tecnologia começa a apresentar perdas de desempenho. Neste sentido, os dados devem ser movidos com uma determinada periodicidade do armazenamento de dados em tempo real para o histórico, pelo que, dada a forma como os dados se encontram organizados no armazenamento de dados históricos, mesmo que apresente grande volume de dados, comparativamente ao armazenamento em tempo real, justifica-se a transferência dos dados. Há que ter em conta que, uma organização, quando se depara com problemas de desempenho provavelmente a preocupação é o investimento em *hardware*, contudo, o que se pretende também perceber com este trabalho, é a necessidade de os dados serem movidos para uma área de armazenamento que possua o histórico dos dados, representando esta uma solução que possa satisfazer melhor as necessidades de qualquer organização. Além disso, devido à área de armazenamento de dados em histórico não estar a ser alimentada em tempo real, revelar-se-á mais eficiente, existindo menos necessidade de processamento por parte das tecnologias. De realçar que o tempo que os dados devem permanecer no armazenamento de dados em tempo real antes de serem movidos para o repositório histórico deve ser estudado de acordo com cada contexto, dependendo das consultas desejadas, das tecnologias em questão, bem como do volume de dados.

A área de armazenamento de dados históricos representa uma solução para os todos os dados da organização, pelo que, comparando os ficheiros armazenados no histórico com os armazenados no repositório de dados em tempo real, os ficheiros do histórico apresentam uma maior dimensão, bem como a quantidade de ficheiros é inferior, como representado na Figura 34, pois a informação encontra-se concatenada. Além disso, os ficheiros armazenados no histórico estão otimizados para o acesso sequencial do Hadoop, visto que são de dimensão igual ou superior ao tamanho do bloco. Ainda, o esquema de particionamento está otimizado para as questões analíticas mais frequentes pelo que está particionado pela data do evento. Dependendo do contexto do negócio, o esquema de particionamento deverá ser adequado ao que a organização procura, pelo que, como caso de exemplo, para as vendas dos últimos 3 meses, o seu esquema de particionamento poderá ser, por exemplo, ao mês.

Name >	Size >	Last Modified >	Owner >	Group >	Permission
<div style="border: 1px solid black; padding: 2px; display: inline-block;">Ficheiros no armazenamento de dados históricos</div>					
000014_0	24.2 MB	2017-10-05 14:29	lid4	hdfs	-rwxrwxrwx
000000_0	24.9 MB	2017-10-05 14:27	lid4	hdfs	-rwxrwxrwx
000002_0	25.0 MB	2017-10-05 14:28	lid4	hdfs	-rwxrwxrwx
000003_0	25.2 MB	2017-10-05 14:28	lid4	hdfs	-rwxrwxrwx
000004_0	25.3 MB	2017-10-05 14:27	lid4	hdfs	-rwxrwxrwx
000005_0	25.4 MB	2017-10-05 14:28	lid4	hdfs	-rwxrwxrwx
000006_0	25.5 MB	2017-10-05 14:28	lid4	hdfs	-rwxrwxrwx
000007_0					
<div style="border: 1px solid black; padding: 2px; display: inline-block;">Ficheiros no armazenamento de dados em tempo real</div>					
part-00002	34.0 kB	2017-10-12 14:10	lid4	hdfs	-rwxrwxrwx
part-00000	33.5 kB	2017-10-12 14:10	lid4	hdfs	-rwxrwxrwx
part-00000_copy_10	50.6 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_11	45.9 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_12	52.0 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_13	50.1 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_14	48.0 kB	2017-10-12 14:11	lid4	hdfs	-rwxrwxrwx
part-00000_copy_15	50.9 kB	2017-10-12 14:12	lid4	hdfs	-rwxrwxrwx
part-00000_copy_16	52.3 kB	2017-10-12 14:12	lid4	hdfs	-rwxrwxrwx
part-00000_copy_17	53.0 kB	2017-10-12 14:12	lid4	hdfs	-rwxrwxrwx

Figura 34 - Diferença da dimensão de ficheiros nas áreas de armazenamento de dados

De seguida, no Gráfico 23, apresentam-se alguns resultados das consultas efetuadas de forma a ser compreendido o desempenho do Hive como armazenamento de dados históricos, com diferentes volumes de dados. Apresentam-se as consultas 1 (C1), 2 (C2), 3 (C3) e 4 (C4) já apresentadas nas secções anteriores, bem como a consulta 9 (C9), que questiona as cinco horas onde são verificados mais *tweets*, a consulta 10 (C10), que é a verificação do número de registos cuja língua utilizada é o inglês, a consulta 11 (C11), que é uma verificação do número de registos num determinado dia, a consulta 12 (C12), que é a verificação do número de *tweets* em cada minuto para uma determinada data, e a consulta 13 (C13), que é a verificação do número de *tweets* em cada minuto para uma determinada hora.

Pelo que se pode observar pelos resultados apresentados e comparando com os resultados das consultas no Hive e Cassandra, ao fim de 4 e 8 horas de dados a fluir em *streaming*, na Tabela 4, apresentados anteriormente nas secções 5.3.1 e 5.3.2, o armazenamento de dados históricos apresenta melhor desempenho, considerando o volume de dados em questão.

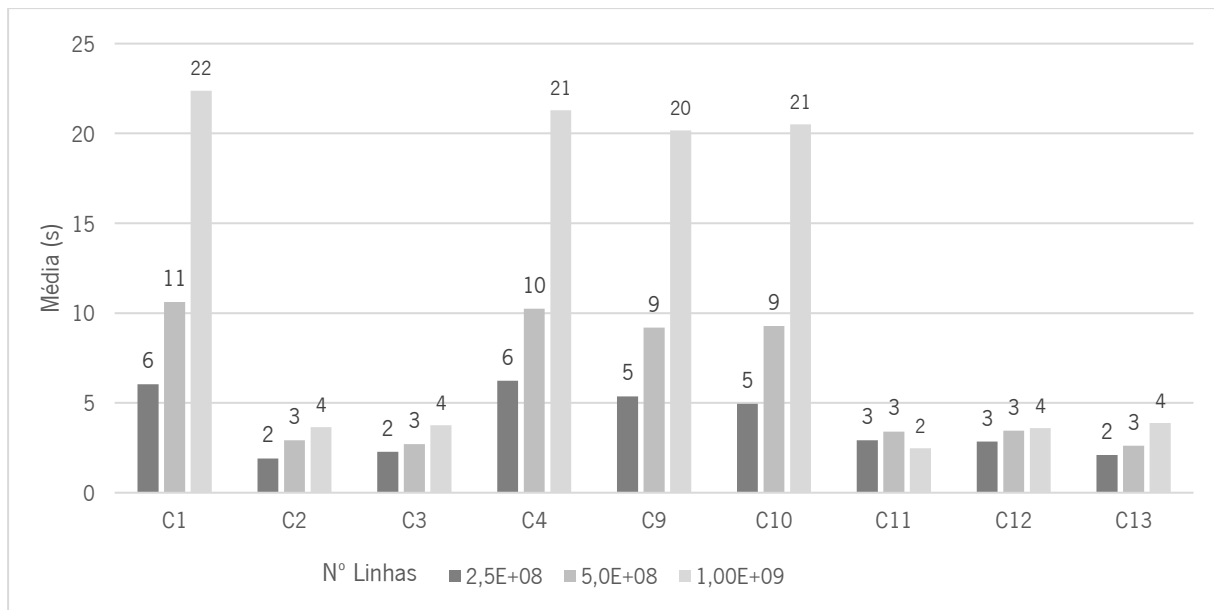


Gráfico 23 - Resultados da consulta C1, C2, C3, C4, C9, C10, C11, C12 e C13 na tabela Hive do histórico²⁶

Tabela 4 - Resultados da consulta 1 (C1), 2 (C2), 3 (C3) e 4 (C4) no Hive e Cassandra

Hora	Nº Linhas Hive	C1 Hive (s)	C2 Hive (s)	C3 Hive (s)	C4 Hive (s)
4:00	2,7E+07	18,011	4,782	4,681	19,368
8:00	4,8E+07	35,991	4,929	5,388	37,821
Hora	Nº Linhas Cassandra	C1 Cassandra (s)	C2 Cassandra (s)	C3 Cassandra (s)	C4 Cassandra (s)
4:00	2,7E+07	147,892	68,326	73,762	70,901
8:00	5,1E+07	276,910	126,467	135,820	131,719

Além disso, analisando os resultados das várias consultas, no Gráfico 23, é possível afirmar que, de forma geral, mesmo não estando estas consultas filtradas pelo atributo de particionamento, nomeadamente a data do *tweet*, estas apresentam um bom desempenho. A consulta C11 e C12 encontram-se filtradas pelo atributo de particionamento e apresentam evidenciado o benefício do particionamento da tabela. No entanto, é de realçar que a C2, C3 e C13 apresentam um tempo de resposta inferior ao das restantes consultas. Estas consultas, C2, C3 e C13, percorrem os dados de apenas uma determinada hora, contudo, percorrem as várias partições, ou seja, os dados armazenados correspondentes às várias datas.

Por conhecimento prático, é possível concluir que os dados no armazenamento de dados históricos, ao serem movidos de uma tabela particionada por um determinado atributo no

²⁶ A notação usada para a representação do número de linhas é a notação científica, pelo que por exemplo 2,5E+08 representa 250000000 linhas.

armazenamento de dados em tempo real, herdamos os dados da tabela, ou seja, os seus metadados. O facto da tabela particionada à data do evento no armazenamento de dados históricos herdar metadados da tabela particionada à hora do evento no armazenamento de dados em tempo real permite-lhe facilmente averiguar em que partições estão guardados os dados para responder às consultas de uma hora em questão, determinando mais rapidamente se existem dados armazenados referentes aquela hora e em que ficheiros se encontra.

Desta forma, os dados no armazenamento histórico, estando particionados pela data do evento, evidenciam as suas especificidades nas consultas que filtram determinadas datas (nomeadamente C11 e C12) mas também consultas que filtram pela hora do evento (nomeadamente C2, C3 e C13). Existe assim um reconhecimento por parte do Hive e Presto, chamado de *predicate pushdown* ao nível do ficheiro ORC, pelo que são mais facilmente obtidos dados estatísticos de cada ficheiro, de forma a perceber se o ficheiro está ou não incluído na consulta. Os dados estatísticos incluem valores como a contagem, máximo, mínimo, o que ajuda o Presto a descobrir se o ficheiro possui informação relevante ou não para a consulta em questão, pelo que cada ficheiro ORC criado tem contidas essas estatísticas devido a serem armazenadas no próprio ficheiro. O facto de ignorar parte dos ficheiros torna-se relevante pois permite ganhos de desempenho, como se observa pelos resultados da C2, C3 e C13. Apesar de a C3 e a C13, comparativamente à C2, não serem apenas a contagem dos registos dessa hora, mas existir um filtro nos dados ou agregação de resultados de outra forma, o desempenho destas mantém-se semelhante ao da C2. O mesmo acontece na C12, pelo que a C11 efetua a contagem dos registos, mas a C12 agrega os resultados de outra forma, contudo o desempenho de C11 e C12 é semelhante.

Comparando os resultados das consultas do repositório do histórico com a tecnologia Hive no armazenamento de dados em tempo real, em que a tabela se encontra particionada à hora do evento, mesmo estando os dados já organizados em partições, onde por cada inserção de dados é criado um ficheiro no Hive, a concatenação dos dados, a sua junção em ficheiros de maior dimensão, traz vantagens ao nível do seu desempenho.

Deste modo, mesmo existindo mais registos no armazenamento de dados históricos, o desempenho deste é comparativamente melhor ao do armazenamento de dados em tempo real, reforçando a ideia de ganhos no desempenho pelos dados serem movidos com uma determinada periodicidade para o histórico. Esta análise da penalização do armazenamento em tempo real para histórico permite perceber até que ponto as tecnologias são escolhas adequadas ou não atendendo ao volume de dados em questão.

Todavia, pensando num contexto organizacional, no armazenamento de dados históricos seriam efetuadas agregações nos dados, pelo que, apesar de no histórico a principal preocupação não ser necessariamente o tempo de resposta às consultas, mas sim obter informação histórica, o seu desempenho seria ainda melhor. Deste modo, é importante ter em conta qual o tempo de resposta máximo que se espera nas consultas ao armazenamento em tempo real e com base nisto definir a periodicidade, garantindo assim que as consultas ao armazenamento em tempo real obtêm respostas em tempo adequado.

De forma a reforçar a integração do armazenamento de dados em tempo real com o repositório do histórico, como exemplo, no Gráfico 24, apresentam-se os resultados das consultas que integram a tabela de dados históricos com a tabela de dados em tempo real no Hive, enquanto estão a ser recolhidos, processados e armazenados os dados a fluir em *streaming*. As consultas apresentadas são a consulta 14 (C14), que são os cinco utilizadores com mais *tweets* entre um intervalo de datas, a consulta 15 (C15), que são os cinco dispositivos mais utilizados, e a consulta 16 (C16) que são as cinco datas em que se verificam um maior número de *tweets*. Os cenários de testes são T1 com 500E+06 registos no armazenamento de dados históricos e 20E+06 registos no armazenamento de dados em tempo real, T2 com 500E+06 registos no histórico e 36E+06 registos em tempo real, T3 com 800E+06 registos no histórico e 25E+06 registos em tempo real, T4 com 800E+06 registos no histórico e 94E+06 registos em tempo real, T5 com 1080E+06 registos no histórico e 22E+06 registos em tempo real e T6 com 1080E+06 registos no histórico e 38E+06 registos em tempo real.

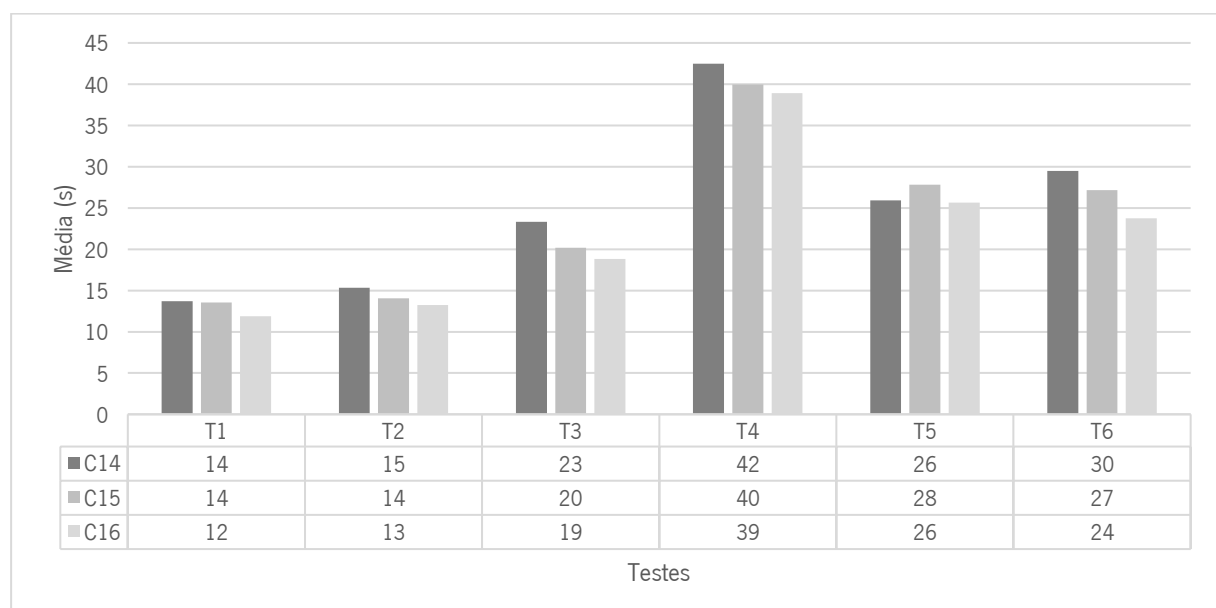


Gráfico 24 - Resultados das consultas C14, C15 e C16 com diferentes volumes de dados no armazenamento de dados em tempo real e histórico no Hive

O objetivo da realização destas consultas é a validação das capacidades da arquitetura proposta, pelo que, com já referido, podem ser efetuadas consultas apenas ao armazenamento de dados em tempo real, ao armazenamento de dados em histórico ou a ambos os repositórios. Observando os resultados apresentados no Gráfico 24 e, considerando os resultados obtidos nos restantes testes, é possível afirmar que estas tecnologias apresentam também um bom desempenho quando há junção de dados de dois repositórios diferentes. Além disso, quanto maior o volume de dados no armazenamento de dados em tempo real, pior o desempenho das consultas que integram os dados dos dois repositórios.

5.4. Síntese de Resultados

A possibilidade de os dados fluírem em tempo real é um grande desafio, considerando os trabalhos que têm vindo a ser desenvolvidos na área, representando um fator que maximiza a eficiência do negócio. Contudo, o objetivo desta dissertação ainda se revela maior por se procurar o equilíbrio entre a integração de novos dados e a consulta em tempo real, garantindo isto com o volume, a velocidade e a variedade dos dados.

De forma a garantir que o objetivo desta dissertação é cumprido, concretizando o *Big Data Warehouse* em tempo real, foram realizados um conjunto de testes na procura das tecnologias que oferecem um melhor desempenho, garantindo assim que, havendo recolha, processamento e armazenamento de dados, a análise seja rapidamente possível. Para isso, é necessário que os dados fluam em tempo real, ou seja, que as tecnologias apresentem rapidez nas suas tarefas.

A extração dos dados deverá partir de uma fonte de dados que ofereça uma grande quantidade de dados por segundo, pelo que o Kafka representa uma ferramenta capaz de lidar com a velocidade dos dados, podendo ser ainda otimizado para apresentar melhor desempenho.

Quanto à tecnologia de processamento de dados, nomeadamente Spark Streaming, é importante que o *micro-batch* definido seja adequado ao contexto de negócio, bem como ao tempo de processamento e ao volume de dados que fluem em tempo real. O facto de o *micro-batch* apresentar a duração adequada ao contexto, permitirá que os dados que acabam de ser recolhidos, sejam rapidamente processados, não atrasando a inserção de dados e garantindo que a análise aos dados possa ser feita rapidamente.

As tecnologias de armazenamento em tempo real revelam-se capazes de garantir o desejado tempo real, contudo, em condições diferentes. O Hive, com a sua capacidade de lidar com grandes volumes de dados, apresenta um bom desempenho seja qual for o volume de dados que é extraído em *streaming*. Contudo, é importante que as suas especificidades, ou seja, a criação de partições, adequadas ao contexto de negócio, sejam incluídas, porque as consultas que filtrem pelo atributo de

particionamento destacam-se muito pelo seu desempenho. O Cassandra, ao fim de algumas horas de dados, apresenta atrasos na inserção de dados devido ao facto de as consultas serem efetuadas ao mesmo tempo em que os dados são recolhidos, processados e armazenados, pelo que, dado o volume de dados, as consultas tornam-se mais lentas. Todavia, num volume de dados inferior, o seu comportamento é melhor, conseguindo um menor atraso na inserção de dados enquanto as consultas aos dados estão a ser realizadas. Contudo, dado o contexto desta dissertação, não foi possível a utilização das especificidades do Cassandra tendo em vista a melhoria do seu desempenho, nem existe atualização de dados, contextos em que esta tecnologia se poderia destacar. Desta forma, ambas as tecnologias de armazenamento de dados poderão representar uma solução, dependendo do contexto, tendo em conta que os dados permanecerão nestas tecnologias por um curto espaço de tempo. Como armazenamento de dados em tempo real, no caso desta dissertação, o Hive, com a tabela particionada à hora do evento, representa uma melhor opção pois permite a concretização do BDW em tempo real, garantindo que os dados fluem em tempo real. Porém, em ambas as tecnologias, com os resultados obtidos, verifica-se a necessidade de mover estes dados para um armazenamento de dados históricos, alinhado com a abordagem de processamento em tempo real de Kimball & Ross (2013), pois, com o aumento do volume de dados, estas tecnologias apresentam um decréscimo no seu desempenho. O repositório de dados históricos mostra que, mesmo tendo uma grande quantidade de dados, consegue-se ter consultas com bom desempenho. Além disso, se os dados permanecerem demasiado tempo no armazenamento em tempo real, o tempo real desejado será prejudicado.

De forma a sistematizar os resultados obtidos, e destacar os contextos em que as tecnologias testadas mais se diferenciam, a Tabela 5 resume os cenários testados e a tecnologia de armazenamento que obteve melhores resultados.

Conjugando os objetivos desta dissertação com os resultados obtidos é possível afirmar que a tecnologia Hive representa uma melhor solução dado o contexto desta dissertação. Contudo, existem cenários em que ambas as tecnologias se destacam, havendo a necessidade de estudar o contexto em questão de a forma a que os requisitos pretendidos sejam cumpridos.

Tabela 5 - Síntese de resultados obtidos

Testes	Hive	Cassandra
Sem especificidades na tecnologia	A tabela sem particionamento revela maior dificuldade na inserção de dados devido a operações de verificação para o armazenamento de dados, pelo que foi possível verificar um atraso de cerca de 1,9 horas.	A tabela sem chave de <i>clustering</i> possui atraso na inserção de dados até cerca de 16 minutos, havendo aumento no tempo de processamento, quando a inserção de dados é efetuada ao mesmo tempo que a consulta de dados.

Testes	Hive	Cassandra
Com especificidades na tecnologia	A tabela particionada revela melhor desempenho em consultas que se apresentem filtradas pelo atributo de particionamento. Nas restantes consultas, a tabela sem particionamento e com particionamento apresentam um desempenho muito semelhante, contudo a tabela sem particionamento apresenta melhor desempenho.	A tabela com chave de <i>clustering</i> possui o atraso e aumento de tempo de processamento semelhante ao ocorrido na tabela sem chave de <i>clustering</i> . O desempenho das consultas é muito semelhante entre a tabela com e sem chave de <i>clustering</i> .
Hive <i>versus</i> Cassandra (tabela particionada <i>versus</i> tabela sem chave de <i>clustering</i>)	A tabela particionada Hive apresenta melhor desempenho nas consultas. O tempo de processamento é de cerca de 3 segundos.	A tabela sem chave de <i>clustering</i> apresenta pior desempenho nas consultas, criando a necessidade de os dados serem mais rapidamente movidos para o histórico. Contudo, esta tecnologia não se encontra num cenário de atualização de dados onde poderia evidenciar a sua capacidade. O tempo de processamento é de cerca de 1 segundo, apresentando rápida escrita de dados.
<i>Micro-batch</i> Spark Streaming 5 <i>versus</i> 20 segundos	A tecnologia Hive apresenta melhor desempenho com a duração superior do <i>micro-batch</i> (20 segundos), devido a serem criados menos ficheiros na inserção de dados, havendo menos operações de escrita. Isto revela que a tecnologia é sobrecarregada com a escrita de dados, pelo que o aumento do <i>micro-batch</i> representa um cenário benéfico.	O desempenho da tecnologia Cassandra é muito semelhante nas diferentes durações do <i>micro-batch</i> do Spark Streaming, pelo que a sobrecarga na escrita de dados não afeta o seu desempenho. Isto revela a capacidade de o <i>micro-batch</i> poder ser de duração ainda menor, dependendo do contexto (do tempo de processamento e do tempo de resposta às consultas esperado).
Volume de dados superior (V2) <i>versus</i> inferior (V3)	O volume de dados, sendo superior ou inferior, não tem grande impacto no desempenho das consultas efetuadas no Hive. Contudo, face à tecnologia Cassandra, este apresenta sempre melhor desempenho nas suas consultas.	Quanto maior o volume de dados, pior o desempenho das consultas no Cassandra, revelando-se em maior atraso na inserção de dados ao mesmo tempo que estão a ser efetuadas as consultas de dados. Quanto menor o volume de dados, melhor o desempenho das consultas, revelando-se num menor atraso e rápida recuperação para o processamento rápido e inserção de dados sem atraso.
Armazenamento de dados históricos	Os dados no histórico apresentam, comparativamente aos dados no armazenamento de dados em tempo real, melhor desempenho nos tempos de resposta das consultas. Em consultas a ambos os repositórios, quanto maior o volume de dados no armazenamento de	

Testes	Hive	Cassandra
	dados em tempo real, pior o desempenho das consultas, e quanto maior o volume de dados no armazenamento de dados históricos, mais adequados serão os tempos de resposta das consultas.	

6. CONCLUSÃO

Este documento apresenta os principais conceitos emergentes nesta área, explicando as principais características e desafios de *Big Data*, as abordagens e particularidades de um *Data Warehouse* bem como os aspetos diferenciadores e as abordagens de concretização de um *Big Data Warehouse*. A explicação de como o armazenamento de dados pode ser efetuado, as bases de dados existentes e os modelos de dados que podem ser utilizados no contexto de *Big Data*, permitem perceber como são organizados os dados e os repositórios em que estes podem ser guardados. A exploração dos requisitos e as principais características de tempo real permitem entender os aspetos diferenciadores, que permitem que uma organização possa ter acesso aos dados assim que eles são recolhidos.

A análise em tempo real tem-se tornado uma vantagem face aos concorrentes, aumentando a procura de soluções que o permitam concretizar. Os trabalhos desenvolvidos por outros autores nem sempre apresentam a componente tecnológica, mas as ideias que apresentam ajudam na proposta da solução desta dissertação, representando um bom ponto de partida para a proposta da arquitetura de *Big Data Warehouse* em tempo real desta dissertação.

A exploração das tecnologias permite o conhecimento do ambiente de *Big Data*, nomeadamente a análise das possíveis tecnologias que poderão integrar a solução pretendida. Representam uma base de soluções, mecanismos e tecnologias importantes para a proposta de um sistema em tempo real. Além disso, apesar de as tecnologias de forma geral oferecerem latência, já há soluções que permitem a redução da latência representando uma tentativa para que seja atingido o tempo real desejado. Cada vez mais estão a ser desenvolvidas novas tecnologias e tendo-as articuladas numa arquitetura, poderão ser obtidos resultados melhores do que os obtidos até ao momento. É neste sentido, que a procura de novas soluções e novas tecnologias motiva a investigação para resultados cada vez mais adequados às organizações.

Com base nos principais conceitos e particularidades de um *Big Data Warehouse* é definida uma proposta de arquitetura, que permite atingir os resultados esperados, pelo que esta representa uma evolução da *Lambda Architecture* considerando as suas três principais camadas que permitem o acesso aos dados em tempo real, tendo em conta a abordagem de processamento em tempo real de Kimball & Ross (2013). Realça-se que este trabalho se enquadra numa área que apresenta um problema que ainda não tem uma solução consistente na comunidade científica, pelo que demonstra a importância da sua realização. De forma a que a solução desenvolvida esteja preparada para os vários contextos que o *Big*

Data oferece, são definidos um conjunto de testes para que esta possa ser aprimorada e represente uma solução em que seja garantida a análise em tempo útil, visto que é este o grande desafio e obstáculo para a análise de dados em tempo real.

Com o desenvolvimento da componente experimental são efetuados um conjunto de testes de forma a estudar cada um dos cenários identificados e perceber o comportamento das tecnologias presentes na arquitetura proposta. Os testes realizados permitem perceber o comportamento da tecnologia de recolha de dados, da tecnologia de processamento de dados e, ainda, o comportamento das tecnologias de armazenamento de dados. O desempenho das tecnologias de armazenamento de dados é analisado, de forma a ser compreendido o desempenho destas com grande volume, velocidade e variedade de dados. Os resultados obtidos permitem tecer um conjunto de considerações, necessárias para o desenvolvimento de um *Big Data Warehouse* em tempo real numa organização.

Neste capítulo são assim resumidos os principais desenvolvimentos desta dissertação, bem como os resultados obtidos ao longo do desenvolvimento deste trabalho. Apresenta-se também a investigação que poderá ser feita futuramente neste âmbito, de forma a ver respondidas algumas questões que foram surgindo neste trabalho, completando o trabalho desenvolvido nesta temática.

6.1. Resultados Obtidos

Como apresentado nos objetivos definidos e resultados esperados, este trabalho permite a concretização do tempo real em *Data Warehouses* em contextos de *Big Data*, algo que ainda está pouco explorado e para o qual ainda não existem muitas soluções na comunidade científica que apresentem resultados conclusivos. Com esta dissertação é, assim, utilizada uma arquitetura e adaptada para permitir concretizar o tempo real pretendido pela concretização de um *Big Data Warehouse*, sendo obtidos resultados e tecidas considerações no trabalho experimental sobre a adequabilidade da mesma num contexto organizacional.

Considera-se que os resultados obtidos concretizam os objetivos pretendidos, destacando-se como principais resultados:

- Sistematização do estado de arte sobre a temática *Big Data Warehouses*;
- Análise das características de tempo real em *Big Data Warehouses*;
- Estudo dos trabalhos realizados na área, numa abordagem mais tradicional bem como em contexto de *Big Data*;
- Análise das possíveis tecnologias para a recolha de dados, o processamento e o armazenamento de dados;

- Arquitetura para o processamento de dados para a concretização de *Big Data Warehouses* em Tempo Real;
- Definição de testes e cenários para a validação da arquitetura proposta;
- Concretização e validação da arquitetura proposta com um caso de demonstração;
- Análise dos resultados obtidos e definição de um conjunto de boas práticas para a concretização de *Big Data Warehouses* em Tempo Real.

Desta forma, com esta dissertação é, assim, concretizado um *Big Data Warehouse* em tempo real permitindo uma análise com informação útil, processamento de dados em tempo real e o cruzamento dos dados em tempo real com dados históricos, para que as decisões sejam tomadas com conhecimento de padrões dos dados e informação atualizada do negócio. O facto de esta dissertação não ter tido foco no contexto de negócio, mas sim na construção de uma solução capaz de ser adaptada a qualquer organização, permite a utilização das suas considerações para a concretização do BDW em tempo real numa organização, oferecendo uma resposta rápida às consultas de dados com dados atuais.

Durante a realização deste trabalho foram encontradas algumas dificuldades relativamente a esta área ser nova, a existir pouca informação técnica, ao facto de as tecnologias estarem a evoluir, o que se revelaram também desafios que foram enfrentados e solucionados.

6.2. Trabalho Futuro

No âmbito da área de investigação desta dissertação, nomeadamente *Big Data Warehouses* com requisitos de tempo real, surgem várias questões de investigação, pelo que nem todas elas foram possíveis abordar nesta dissertação. De seguida apresentam-se questões que podem vir a ser desenvolvidas num trabalho futuro enquadrado nesta dissertação.

De acordo com o que estava planeado nesta dissertação, esta seria implementada no projeto de investigação, num contexto real, com dados reais da BOSCH, pelo que este será o próximo passo desta dissertação. Com base no trabalho desta dissertação e as considerações do seu desenvolvimento, este processo passará por adaptar os requisitos de tempo real mais especificamente aos dados do contexto real da BOSCH e aplicar a arquitetura proposta. Este processo trará desafios ao nível do conhecimento dos dados, da definição do modelo de dados adequado, da análise aos dados desejada, bem como a seleção da tecnologia adequada para o armazenamento de dados.

Outras questões a ter em conta nesta temática seria o estudo de outras tecnologias de recolha de dados, de processamento de dados, bem como de armazenamento de dados, de forma a conhecer mais cenários e ser capaz de tecer considerações acerca de cada uma das tecnologias estudadas e

perceber qual o conjunto de tecnologias que oferece maior rapidez em determinados contextos. Uma das tecnologias emergentes é o Kudu, tecnologia de armazenamento de dados, que afirma ser capaz de integrar as características do Hive e Cassandra. Além disso, seria também interessante analisar um contexto de dados em que existisse atualização de dados, de forma a conhecer o comportamento do Cassandra num cenário em que este afirma evidenciar as suas propriedades. Uma outra questão que não foi possível abordar nesta dissertação está associada às tabelas que suportam transações ACID, que permitem a inserção linha a linha no Hive, pois através destas seria possível a realização de atualizações de dados. Estas não foram consideradas nesta dissertação devido ao Presto, na versão utilizada, ainda não suportar a consulta destas tabelas Hive. Outro cenário a ter em conta nesta temática é o facto de se poder considerar o modelo em estrela no armazenamento de dados históricos, de forma a adequar o contexto do trabalho mais ao contexto das organizações.

REFERÊNCIAS BIBLIOGRÁFICAS

- Alekseev, A. A., Osipova, V. V., Ivanov, M. A., Klimentov, A., Grigorieva, N. V., & Nalamwar, H. S. (2016). Efficient Data Management Tools for the Heterogeneous Big Data Warehouse. *Physics of Particles and Nuclei Letters*, 13(5), 689–692. <https://doi.org/10.1134/S1547477116050022>
- Bruckner, R. M., List, B., & Schiefer, J. (2002). Striving towards Near Real-Time Data Integration for Data Warehouses. In Y. Kambayashi, W. Winiwarter, & M. Arikawa (Eds.), *Data Warehousing and Knowledge Discovery* (pp. 317–326). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-46145-0_31
- Cattell, R. (2011). Scalable SQL and NoSQL Data Stores. *ACM SIGMOD Record*, 39(4), 12–27. <https://doi.org/10.1145/1978915.1978919>
- Chandarana, P., & Vijayalakshmi, M. (2014). Big Data Analytics Frameworks. In *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)* (pp. 430–434). <https://doi.org/10.1109/CSCITA.2014.6839299>
- Chebotko, A., Kashlev, A., & Lu, S. (2015). A Big Data Modeling Methodology for Apache Cassandra. In *Proceedings of the 2015 IEEE International Congress on Big Data* (pp. 238–245). Washington, DC, USA: IEEE Computer Society. <https://doi.org/10.1109/BigDataCongress.2015.41>
- Chen, G. J., Wiener, J. L., Iyer, S., Jaiswal, A., Lei, R., Simha, N., ... Yilmaz, S. (2016). Realtime Data Processing at Facebook. In *Proceedings of the 2016 International Conference on Management of Data* (pp. 1087–1098). New York, NY, USA: ACM. <https://doi.org/10.1145/2882903.2904441>
- Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *Management Information Systems Quarterly*, 36(4), 1165–1188.
- Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, 19(2), 171–209. <https://doi.org/10.1007/s11036-013-0489-0>
- Chen, Y., Qin, X., Bian, H., Chen, J., Dong, Z., Du, X., ... Zhang, H. (2014). A Study of SQL-on-Hadoop Systems. In J. Zhan, R. Han, & C. Weng (Eds.), *Big Data Benchmarks, Performance Optimization, and Emerging Hardware: 4th and 5th Workshops, BPOE 2014, Salt Lake City, USA, March 1, 2014 and Hangzhou, China, September 5, 2014, Revised Selected Papers* (pp. 154–166). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-13021-7_12
- Chevalier, M., El Malki, M., Kopliku, A., Teste, O., & Tournier, R. (2015). Implementing Multidimensional Data Warehouses into NoSQL. In *ICEIS 2015 - 17th International Conference on Enterprise Information Systems* (Vol. 1, pp. 172–183). SciTePress.
- Clegg, D. (2015). Evolving Data Warehouse and BI Architectures: The Big Data Challenge. *Business Intelligence Journal*, 20(1), 19.
- Costa, E., Costa, C., & Santos, M. Y. (2017). Efficient Big Data Modelling and Organization for Hadoop Hive-Based Data Warehouses. In M. Themistocleous & V. Morabito (Eds.), *Information Systems: 14th European, Mediterranean, and Middle Eastern Conference, EMCIS 2017, Coimbra, Portugal, September 7-8, 2017, Proceedings* (pp. 3–16). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-65930-5_1
- Cuzzocrea, A., Song, I.-Y., & Davis, K. C. (2011). Analytics over Large-Scale Multidimensional Data: The Big Data Revolution! *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP*, 101–104. <https://doi.org/10.1145/2064676.2064695>
- Demchenko, Y., Grosso, P., de Laat, C., & Membrey, P. (2013). Addressing Big Data Issues in Scientific Data Infrastructure. In *2013 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 48–55). <https://doi.org/10.1109/CTS.2013.6567203>
- Di Tria, F., Lefons, E., & Tangorra, F. (2014a). Big Data Warehouse Automatic Design Methodology. In *Big Data Management, Technologies, and Applications* (pp. 115–149). IGI Global. <https://doi.org/10.4018/978-1-4666-4699-5.ch006>
- Di Tria, F., Lefons, E., & Tangorra, F. (2014b). Design Process for Big Data Warehouses. *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, 512–518.

- <https://doi.org/10.1109/DSAA.2014.7058120>
- Dijcks, J. P. (2013). Oracle: Big Data for the Enterprise. *Oracle White Paper*.
- dos Anjos, J. C. S., Assunção, M. D., Bez, J., Geyer, C., de Freitas, E. P., Carissimi, A., ... Pereira, R. (2015). SMART: An Application Framework for Real Time Big Data Analysis on Heterogeneous Cloud Environments. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)* (pp. 199–206). Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.29>
- Du, D. (2015). *Apache Hive Essentials*. Packt Publishing Ltd.
- Durham, E.-E. A., Rosen, A., & Harrison, R. W. (2014). A Model Architecture for Big Data Applications using Relational Databases. In *2014 IEEE International Conference on Big Data* (pp. 9–16). <https://doi.org/10.1109/BigData.2014.7004462>
- Fan, W., Zhao, D., & Wang, S. (2015). A Fast Statistics and Analysis Solution of Medical Service Big Data. *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, 9–12. <https://doi.org/doi.ieeecomputersociety.org/10.1109/ITME.2015.14>
- Farooq, F., & Sarwar, S. M. (2010). Real-time Data Warehousing For Business Intelligence. In *Proceedings of the 8th International Conference on Frontiers of Information Technology*. New York, NY, USA: ACM. <https://doi.org/10.1145/1943628.1943666>
- Floratos, A., Minhas, U. F., & Özcan, F. (2014). SQL-on-Hadoop: Full circle back to shared-nothing database architectures. In *Proceedings of the VLDB Endowment* (Vol. 7, pp. 1295–1306). Association for Computing Machinery.
- Freudenreich, T., Furtado, P., Koncilia, C., Thiele, M., Waas, F., & Wrembel, R. (2013). An On-Demand ELT Architecture for Real-Time BI. In M. Castellanos, U. Dayal, & E. A. Rundensteiner (Eds.), *Enabling Real-Time Business Intelligence* (pp. 50–59). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39872-8_4
- Geerts, G. L. (2011). A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems*, 12(2), 142–151. <https://doi.org/http://dx.doi.org/10.1016/j.accinf.2011.02.004>
- Golab, L., & Johnson, T. (2014). Data Stream Warehousing. In *International Conference on Data Engineering* (pp. 1290–1293). IEEE Computer Society. <https://doi.org/10.1109/ICDE.2014.6816763>
- Goss, R. G., & Veeramuthu, K. (2013). Heading Towards Big Data Building A Better Data Warehouse For More Data, More Speed, And More Users. In *ASMC (Advanced Semiconductor Manufacturing Conference)* (pp. 220–225). <https://doi.org/10.1109/ASMC.2013.6552808>
- Hadoop. (2017). Apache Hadoop. Retrieved January 1, 2017, from <http://hadoop.apache.org/>
- Hausenblas, M., & Nadeau, J. (2013). Apache Drill: Interactive Ad-Hoc Analysis at Scale. *Big Data*, 1(2), 100–104. <https://doi.org/10.1089/big.2013.0011>
- Hecht, R., & Jablonski, S. (2011). NoSQL Evaluation: A Use Case Oriented Survey. In *2011 International Conference on Cloud and Service Computing (CSC)* (pp. 336–341). <https://doi.org/10.1109/CSC.2011.6138544>
- Hewitt, E., & Carpenter, J. (2016). *Cassandra: The Definitive Guide* (2nd ed.). O'Reilly Media, Inc.
- Jacobs, A. (2009). The Pathologies of Big Data. *Communications of the ACM*, 52(8), 36–44. <https://doi.org/10.1145/1536616.1536632>
- Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013). Big Data: Issues and Challenges Moving Forward. In *2013 46th Hawaii International Conference on System Sciences* (pp. 995–1004). <https://doi.org/10.1109/HICSS.2013.645>
- Katal, A., Wazid, M., & Goudar, R. H. (2013). Big Data: Issues, Challenges, Tools and Good Practices. In *2013 Sixth International Conference on Contemporary Computing (IC3)* (pp. 404–409). <https://doi.org/10.1109/IC3.2013.6612229>
- Khan, M. A., Uddin, M. F., & Gupta, N. (2014). Seven V's of Big Data Understanding Big Data to extract Value. In *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1)* (pp. 1–5). <https://doi.org/10.1109/ASEEZone1.2014.6820689>
- Kimball, R., & Caserta, J. (2004). *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. John Wiley & Sons.

- Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. John Wiley & Sons.
- Kornacker, M., Behm, A., Bittorf, V., Bobrovitsky, T., Ching, C., Choi, A., ... Yoder, M. (2015). Impala: A modern open-source SQL engine for Hadoop. In *7th Biennial Conference on Innovative Data Systems Research (CIDR)* (Vol. 1).
- Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: a Distributed Messaging System for Log Processing. In *Proceedings of the NetDB*.
- Krishnan, K. (2013). *Data Warehousing in the Age of Big Data*. Elsevier Inc. <https://doi.org/10.1016/B978-0-12-405891-0.00001-5>
- Kudu. (2017). Apache Kudu. Retrieved January 1, 2017, from <https://kudu.apache.org/>
- Labrinidis, A., & Jagadish, H. V. (2012). Challenges and Opportunities with Big Data. *Proceedings of the VLDB Endowment*, 5(12), 2032–2033. <https://doi.org/10.14778/2367502.2367572>
- Laney, D. (2001). *3D Data Management: Controlling Data Volume, Velocity, and Variety*.
- Lebdaoui, I., Orhanou, G., & Elhajji, S. (2014). An Integration Adaptation for Real-Time Datawarehousing. *International Journal of Software Engineering and Its Applications*, 8(11), 115–128. <https://doi.org/10.14257/ijseia.2014.8.11.10>
- Li, X., & Mao, Y. (2015). Real-Time Data ETL Framework for Big Real-Time Data Analysis. In *2015 IEEE International Conference on Information and Automation* (pp. 1289–1294). Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/ICInfA.2015.7279485>
- Lipcon, T., Alves, D., Burkert, D., Cryans, J.-D., Dembo, A., Percy, M., ... others. (2015). Kudu: Storage for Fast Analytics on Fast Data. Technical report, Cloudera, Inc.
- Liu, X., Iftikhar, N., & Xie, X. (2014). Survey of Real-time Processing Systems for Big Data. In *18th International Database Engineering & Applications Symposium - IDEAS '14* (pp. 356–361). Association for Computing Machinery. <https://doi.org/10.1145/2628194.2628251>
- Liu, Z., Yang, P., & Zhang, L. (2013). A Sketch of Big Data Technologies. In *2013 Seventh International Conference on Internet Computing for Engineering and Science* (pp. 26–29). Washington, DC, USA: IEEE Computer Society. <https://doi.org/10.1109/ICICSE.2013.13>
- Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). *Big data: The next frontier for innovation, competition, and productivity*. McKinsey Global Institute. <https://doi.org/10.1080/01443610903114527>
- Mao, Y., Min, W., Wang, J., Jia, B., & Jie, Q. (2014). Dynamic mirror based real-time query contention solution for support big real-time data analysis. In *2nd International Conference on Information Technology and Electronic Commerce (ICITEC 2014)* (pp. 229–233). <https://doi.org/10.1109/ICITEC.2014.7105608>
- Marz, N., & Warren, J. (2015). *Big Data: Principles and best practices of scalable real-time data systems* (1st ed.). Greenwich, CT, USA: Manning Publications Co.
- Mathur, A., Sihag, A., Bagaria, E. G., & Rajawat, S. (2014). A New Perspective to Data Processing: Big Data. *2014 International Conference on Computing for Sustainable Global Development*, 110–114. <https://doi.org/10.1109/IndiaCom.2014.6828111>
- Mohamed, N., & Al-Jaroodi, J. (2014). Real-Time Big Data Analytics: Applications and Challenges. In *2014 International Conference on High Performance Computing Simulation (HPCS)* (pp. 305–310). <https://doi.org/10.1109/HPCSim.2014.6903700>
- Mohanty, S., Jagadeesh, M., & Srivatsa, H. (2013). *Big Data Imperatives: Enterprise Big Data Warehouse, BI Implementations and Analytics* (1st ed.). Berkely, CA, USA: Apress.
- Özcan, F., Tatbul, N., Abadi, D. J., Kornacker, M., Mohan, C., Ramasamy, K., & Wiener, J. (2014). Are We Experiencing a Big Data Bubble? In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (pp. 1407–1408). New York, NY, USA: ACM. <https://doi.org/10.1145/2588555.2618215>
- Peffer, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Philip Chen, C. L., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275, 314–347. <https://doi.org/10.1016/j.ins.2014.01.015>
- Pokorny, J. (2014). How to store and process big data: Are today's databases sufficient? *Lecture Notes in Computer*

- Science*, 8838, 5–10.
- Presto. (2017). Presto. Retrieved January 1, 2017, from <https://prestodb.io/>
- Russom, P. (2016). *Data Warehouse Modernization in the Age of Big Data Analytics. TDWI Best Practices Report*.
- Rutherglen, J., Wampler, D., & Capriolo, E. (2012). *Programming Hive* (1st ed.). O'Reilly Media, Inc.
- Sagioglu, S., & Sinanc, D. (2013). Big Data: A Review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 42–47). <https://doi.org/10.1109/CTS.2013.6567202>
- Santos, M. Y., & Costa, C. (2016). Data Models in NoSQL Databases for Big Data Contexts. In Y. Tan & Y. Shi (Eds.), *Data Mining and Big Data* (pp. 475–485). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-40973-3_48
- Santos, M. Y., Costa, C., Galvão, J., Andrade, C., Martinho, B. A., Vale Lima, F., & Costa, E. (2017). Evaluating SQL-on-Hadoop for Big Data Warehousing on Not-So-Good Hardware. In *Proceedings of the 21st International Database Engineering & Applications Symposium* (pp. 242–252). New York, NY, USA: ACM. <https://doi.org/10.1145/3105831.3105842>
- Santos, M. Y., Oliveira e Sá, J., Costa, C., Galvão, J., Andrade, C., Martinho, B., ... Costa, E. (2017). A Big Data Analytics Architecture for Industry 4.0. In *WorldCist'17 - 5th World Conference on Information Systems and Technologies*.
- Santos, R. J., & Bernardino, J. (2008). Real-time Data Warehouse Loading Methodology. In *Proceedings of the 2008 International Symposium on Database Engineering & Applications* (pp. 49–58). New York, NY, USA: ACM. <https://doi.org/10.1145/1451940.1451949>
- Santos, R. J., Bernardino, J., & Vieira, M. (2011). 24/7 Real-Time Data Warehousing: A Tool for Continuous Actionable Knowledge. In *2011 IEEE 35th Annual Computer Software and Applications Conference* (pp. 279–288). <https://doi.org/10.1109/COMPSAC.2011.44>
- Spark. (2017). Apache Spark.
- Thusoo, A., Sarma, J. Sen, Jain, N., Shao, Z., Chakka, P., Anthony, S., ... Murthy, R. (2009). Hive - A Warehousing Solution Over a Map-Reduce Framework. In *Proceedings of the VLDB Endowment* (Vol. 2, pp. 1626–1629).
- Thusoo, A., Sarma, J. Sen, Jain, N., Shao, Z., Chakka, P., Zhang, N., ... Murthy, R. (2010). Hive - A Petabyte Scale Data Warehouse Using Hadoop. In *International Conference on Data Engineering* (pp. 996–1005). <https://doi.org/10.1109/ICDE.2010.5447738>
- Ularu, E. G., Puican, F. C., Apostu, A., & Velicanu, M. (2012). Perspectives on Big Data and Big Data Analytics. *Database Systems Journal*, 3(4), 3–14.
- Vaisman, A., & Zimányi, E. (2012). Data warehouses: Next Challenges. *Lecture Notes in Business Information Processing*, 1–26. https://doi.org/10.1007/978-3-642-27358-2_1
- Waas, F., Wrembel, R., Freudenreich, T., Thiele, M., Koncilia, C., & Furtado, P. (2013). On-Demand ELT Architecture for Right-Time BI: Extending the Vision. *International Journal of Data Warehousing and Mining (IJDWM)*, 9(2), 21–38. <https://doi.org/10.4018/jdwm.2013040102>
- Ward, J. S., & Barker, A. (2013). Undefined By Data: A Survey of Big Data Definitions. *arXiv.org*, 2.
- White, T. (2012). *Hadoop: The Definitive Guide* (4th ed., Vol. 54). O'Reilly Media, Inc. <https://doi.org/citeulike-article-id:4882841>
- Zikopoulos, P. C., DeRoos, D., Parasuraman, K., Deutsch, T., Corrigan, D., & Giles, J. (2012). *Harness the Power of Big Data: The IBM Big Data Platform. McGraw Hill Professional*. <https://doi.org/10.1007/s13398-014-0173-7.2>
- Zikopoulos, P., Eaton, C., DeRoos, D., Deutsch, T., & Lapis, G. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data* (1st ed.). McGraw-Hill Osborne Media.