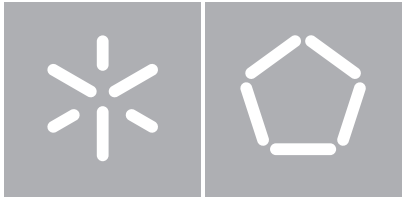
Universidade do Minho
Escola de Engenharia

Carla Flôr Costa Regadas

Dashboarding sobre aplicações Big Data

Outubro de 2015



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Carla Flôr Costa Regadas

Dashboarding sobre aplicações Big Data

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor Doutor Orlando Manuel de Oliveira Belo

Outubro de 2015

Dashboarding sobre aplicações Big Data

Carla Regadas

Dissertação apresentada à Universidade do Minho para obtenção do grau de Mestre em Engenharia Informática, elaborada sob orientação do Professor Doutor Orlando Manuel de Oliveira Belo.

2015

Dedico todo este trabalho aos meus pais e ao meu irmão que sempre me apoiaram ao longo da vida, com um sorriso e força necessária para enfrentar os meus momentos mais difíceis.

Agradecimentos

No final deste trabalho não posso esquecer e expressar o meu agradecimento a todos que contribuíram de algum modo para a concretização desta dissertação, nomeadamente:

Ao meu orientador Professor Doutor Orlando Belo, pela sua dedicação e disponibilidade com que sempre me recebeu, pelas sugestões que indicou, pelos seus ensinamentos e pelo seu incondicional apoio;

Aos professores, pela imprescindível orientação fornecida ao longo destes anos, sendo um suporte neste percurso académico;

Aos amigos que me apoiaram e me deram ânimo ao longo desta minha vida académica, com especial atenção, ao Miguel Veiga que tanto pela sua amizade, como pelo seu incondicional apoio, esteve sempre presente e disponível para me ajudar a ultrapassar as dificuldades encontradas ao longo da realização desta dissertação;

À empresa Blip, que disponibilizou todos os meios e condições propicias para o desenvolvimento desta dissertação;

Aos meus pais e ao meu irmão Filipe Regadas, pelo carinho, incentivo e o apoio incondicional;

Por fim, a todos aqueles que diretamente ou indiretamente me apoiaram nesta jornada, o meu *muito obrigada*.

Resumo

Dashboarding sobre aplicações Big Data

Atualmente, informação é um elemento crucial para a sobrevivência de uma organização na era altamente competitiva em que se vive. As plataformas desenvolvidas, integram cada vez mais com um maior numero de serviços (redes sociais, serviços de musica, serviços de localização, etc) com o simples objetivo de atingir massa critica (numero de utilizadores) suficiente para assegurarem a sustentabilidade. De forma a conseguirem responder as várias necessidades ou mudanças no mercado, existe a necessidade de armazenar o máximo de informação possível para posterior análise. Dado que atualmente, uma plataforma poderá ter um número de utilizadores bastante elevado (milhões), o volume de dados armazenado poderá ser na ordem dos *PetaBytes*.

A análise de informação num volume de dados desta grandeza requer o uso de plataformas *Big Data*, que abordam o armazenamento e análise dos dados de uma forma não convencional. Este tipo de plataformas confere assim às empresas uma maior capacidade de processamento e obtenção do respetivo valor de toda a informação recolhida, operando de forma mais eficiente e rentável. Desta forma permite asegurar à empresa um aumento da receita e uma diminuição dos custos, verificando-se consecutivamente o aumento da produtividade. Contudo, um dos aspetos negativos averiguados nos ultimos tempos, envolve a falta de eficiencia na análise dos dados encontram armazenados após a fase de processamento, existindo assim a necessidade de complementar essa análise através da implementação de uma interface gráfica dos dados manipulados, *dashboards*, de forma a que o valor contido nos dados seja mais percetível e legível.

Estudos revelam que o cérebro humano consegue processar 60,000 vezes mais rápido imagens do que texto e na realidade verifica-se que após o surgimento de ferramentas de pesquisa de dados com base na visualização, adquiriu-se uma oportunidade na construção de novos dados por simples incorporação de ferramentas interativas, tornando possível uma análise mais específica e uma maior organização gráfica para posteriores análises e apresentações perante o mercado. Atualmente,

existem algumas ferramentas que já possibilitam este tipo de abordagem gráfica, contudo, na maioria delas verifica-se a inexistência de processamento das ferramentas em tempo real.

Nesta dissertação irá ser explorado este conceito de *dashboarding* em tempo real sobre aplicações *Big Data*, com o complemento de um caso de estudo prático que consistirá no desenvolvimento de uma solução *Big Data* numa plataforma de aquisição, transformação e difusão de *feeds* desenvolvido pela empresa *Blip*.

Palavras-Chave: Informação, Análise, Big Data, Dashboards, Visualização, Feeds.

Abstract

Dashboarding on Big Data applications

Reaching the limits of a human's ability to respond, tools are built to process vast amounts of data available to decision makers, enabling them to analyze, present, and react to events as they occur.
While Apache Hadoop and other technologies are emerging to support back-end concerns such as storage and processing, visualization-based data discovery tools tend to focus firstly on the front end side of big data—on helping companies understand business data in a more efficient way.
Visualization-based data discovery tools provide an immense opportunity, not only to manage the growing volume, variety, and velocity of new and existing data but also to turn that data into value.
Studies show that the brain processes images 60,000x faster than text. This is why one of the steps in the analytics stage of any big data solution should be to improve the visual representation of the results obtained during analysis.
In Datameer, the visualization is tied to your analysis, so any time the data changes, the visualization will automatically update with the newest results.
Dashboards provide a snapshot view of your key business data for quick and easy data analysis / visualization.
A dashboard consists of embedded interactive reports (any of charts, pivot tables, summary views, tabular views, data tables and query tables) along with any formatted text.
One of the primary challenges in visualizing movement in public transportation networks is handling the amount of data. This becomes particularly difficult when the visualization should be available in real-time and for arbitrary spatial parts of the dataset, for example to provide an interactive live map.
In this dissertation, we present an approach that combines static schedule data and real-time delay information into vehicle trajectories that are treated as piecewise linear functions from the time domain to the projected map-plane, allowing for fast interpolation. We introduce a highly efficient server that accepts temporal and spatial boundaries as an input and returns trajectory parts within

these limits. We discuss the implementation details, show that our approach scales well and demonstrate that it is able to handle public transit datasets of entire countries. As a client, we introduce a thin vector-layer for map services that can display thousands of (time-lapsed) vehicle-movements at the same time. This dissertation also briefly outlines an approach to extract vehicle trajectories from geospatial datasets.

Keywords: Information, Big Data, Apache Hadoop, Visualization, Dashboards, Datameer.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

BI	Business Intelligence
DW	Data Warehousing
RDMS	Relational Database Management System
IT	Tecnologias de Informação
IO	Inteligência Operacional
EPSEM	Equal Probability Selection Element Method
SRS	Simple Random Sampling
CAP	Consistency, Availability and tolerance to network Partitions
ML	Machine Learning

Capítulo 1

Introdução

1.1 Contexto e Motivação

Com o avançar do tempo, o mundo das redes sociais, da computação em *cloud* ou dos websites entre outros, alcançaram um novo patamar na rotina diária das pessoas, produzindo em grande escala, uma maior diversidade de conjuntos de dados. Como consequência, surgiu um aumento progressivo do volume de dados, estruturados ou não, provenientes dos ficheiros de registo de eventos, de redes sociais, etc. Contudo este aumento provocou um elevado decréscimo na eficiência das tarefas de processamento e análise de dados no mercado de negócios.

Muitas soluções têm sido então criadas de forma a controlar esta súbita explosão do volume de dados que nos rodeia na atualidade, nomeadamente, os relacionados com as ditas soluções *Big Data*. Porém, muitos clientes que recorreram a este tipo de soluções, verificaram a existência de uma fase de análise de dados complexa e pouco acessível a qualquer pessoa. O surgimento de plataformas na área de visualização dos dados, aparecem, assim, como uma possível resposta a este problema, permitindo a monitorização dos dados através de imagens e gráficos, e facilitação de todo o processo de análise desses dados. A ideia de *dashboarding* em tempo real sobre aplicações *Big Data*, encontra-se atualmente em crescimento, dada as suas capacidades em fornecer maior eficiência e utilidade em questões de análise e por ser visivelmente mais prático ao utilizador.

Cada vez mais no mundo de negócio, há uma maior necessidade de agir rapidamente em conformidade com as alterações colocadas pelo mercado. Desta forma, o surgimento de métodos que possibilitam atualizações constantes ao longo de todo o processo, desde a recolha até ao processamento e, de seguida, à ilustração do conjunto de dados por diversos tipos de

representações gráficas, torna-se essencial e contribui para a diminuição do desperdício de recursos, fornecendo ao mesmo tempo uma maior utilidade a este de tecnologia.

1.2 Objetivos

O objetivo principal deste trabalho consiste na análise de conhecimento já existente sobre *dashboards*, na compreensão e otimização da representação dos dados em tempo real e na revisão de plataformas já desenvolvidas e implementadas no mercado, que abordam o conceito de *dashboarding* em soluções de *Big Data*. A possibilidade de compreender todo o procedimento que é realizado nos dias de hoje e deparar com limitações neste tipo de soluções torna-se mais elevada, contribuindo assim para o melhoramento e arranque de uma nova solução, que alcance e ilustre dados em tempo real. Ao mesmo tempo, este estudo é acompanhado de um caso prático, que consiste no desenvolvimento de uma solução *Big Data* numa plataforma de *feeds* desenvolvida pela empresa *Blip* e na respetiva construção de uma interface que ilustre uma combinação de ferramentas *dashboarding* que permita garantir uma maior visibilidade do negócio, face às necessidades da empresa.

1.3 Estrutura do documento

Esta dissertação encontra-se divida da seguinte forma: no **capítulo 1** é abordado uma pequena introdução sobre a grande necessidade existente no mercado na atualidade e o porquê da relevância de um painel de *dashboards* sobre soluções *Big Data* implementadas. É ainda abordado os motivos que levaram ao surgimento desta dissertação e o que se pretende alcançar com o desenvolvimento desta; no **capítulo 2** é feito um levantamento do estado de arte que serve de base para esta dissertação, nomeadamente, o conceito de Inteligência Operacional e os benefícios que pode fornecer às empresas que se baseiem nesta teoria. A visualização dos dados é outro grande foco desta dissertação e neste capítulo foi referida a sua importância, o seu estado na atualidade, elaborada uma comparação de plataformas já implementadas no mercado e as falhas que ainda existem e que podem vir a ser melhoradas;

no **capítulo 3** são mencionados os conceitos recolhidos sobre *sampling* e os seus métodos, assim como estes serão abordados neste trabalho e as vantagens obtidas por estes, tais como a possível otimização do tempo para a representação da informação; no **capitulo 4** aborda-se o caso prático desenvolvido no decorrer desta dissertação, sendo uma possível solução para muitos dos casos de *Big Data* onde ainda se verificam algumas falhas na fase de análise, sendo este o problema e a necessidade de muitas empresas nos dias de hoje; por fim, no **capítulo 5** conclui-se aspetos retirados de todo este trabalho, mencionando alguns pontos que podem vir a ser desenvolvidos num trabalho futuro.

Capítulo 2

Estado da Arte

2.1 Introdução

Atualmente, o número de empresas que atribuem à informação um papel crucial para a sua sobrevivência no mercado tem vindo a crescer e alguns estudos provam que este tipo de empresas atingem um melhor desempenho que as restantes, por exemplo, o caso da **Google** e da **Amazon**, tal como é ilustrado no gráfico apresentado na Figura 2.1.

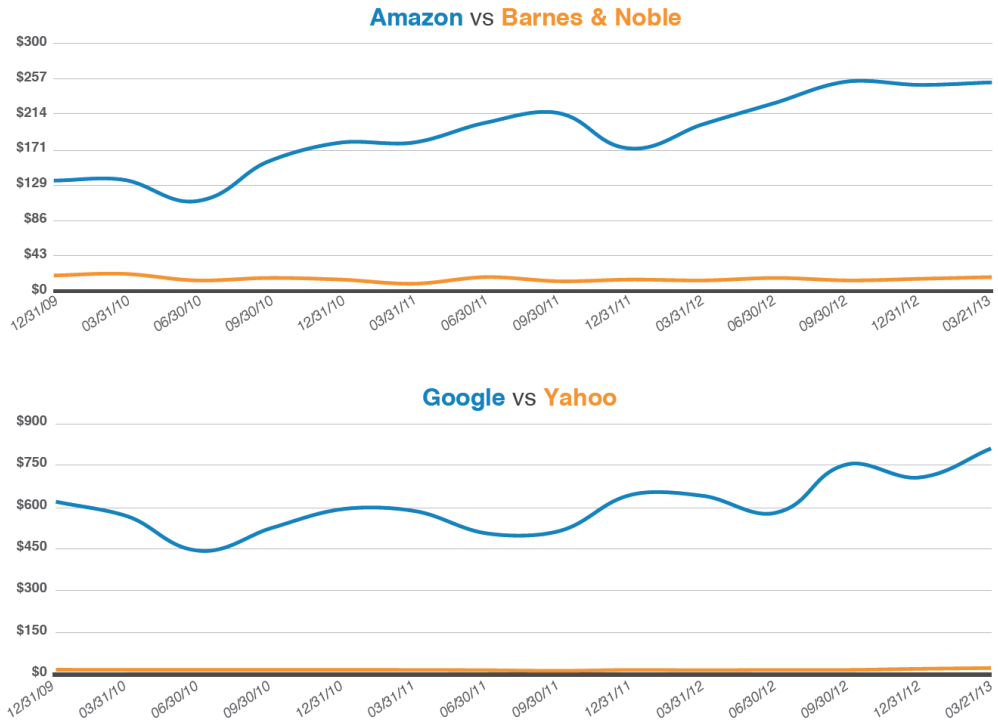


Figura 2.1: Desenvolvimento das empresas com base nos dados até o ano de 2013, realizado pela Datameer

Todo este desempenho provém do crescimento progressivo do uso das redes sociais, *websites*, entre outras fontes de dados, que produzem diariamente uma enorme quantidade de informação. Ou seja, o termo "informação" nos dias de hoje passa a ser considerado um ponto focal para a sobrevivência de uma empresa que se encontre neste tipo de mercado.

Antes da introdução do conceito *Big Data*, toda esta informação era processada e alocada em bases de dados tradicionais (RDBMS), recorrendo às ferramentas de *Business Intelligence (BI)* e *Data Warehousing (DW)*. Como consequência, muitas empresas depararam-se com um período de horas de processamento demasiado extenso, sobrecarga da base de dados e na maioria a inexistência de qualquer tipo de processamento por parte das ferramentas de BI e DW, que não suportam quer dados semiestruturados como não estruturados. Desta forma, as tarefas associadas aos processos usados na gestão desses dados tornam-se mais complexas, perdendo níveis de eficiência no suporte aos dados adquiridos e o tempo de análise torna-se extenso e demorado, não respondendo assim às necessidades de um mercado que se encontra em constante mudança, gerando prejuízos às empresas. Como consequência, a procura por soluções *Big Data* aumentou exponencialmente, sendo considerado o grande "*buzz*" da atualidade. Através do uso deste tipo de soluções, as empresas passam a ser capazes de recolher, processar e analisar eficientemente os seus dados. Contudo, à medida que o número de dados não estruturados (*raw data*) das redes sociais, ficheiros de registos, *websites*, continua a crescer, o mesmo se irá verificar com a necessidade das empresas na procura de soluções capazes de monitorizar e analisar em tempo real, para que deste modo possam extrair com mais eficácia o valor do negócio e usá-lo a seu favor.

Em resposta a tal necessidade, a área de *Inteligência Operacional* (IO) foi desenvolvida com o intuito de garantir o suporte na visibilidade, com uma perspetiva mais simples e prática, dos dados processados, eventos e operações à medida que são coletados para a base de dados.

> *"Contar uma história através de dados é o surgimento de uma forma de arte que mostra que os dados não correspondem apenas a valores em escala nem a análises variadas, mas sobre a transformação do que é intrinsecamente contraditório numa narrativa que outros se possam relacionar. Sem a visualização, os dados eram apenas um conjunto de factos, mas com a visualização passam a ter a habilidade de inspirar e transformar a forma como as pessoas vêem o mundo à sua volta." (Batstone)*

A visualização de dados é de facto, uma forma mais transparente, intuitiva e contextual para visualizar a informação sem haver necessidade de se recorrer apenas aos tradicionais valores estatísticos. Em cenários nos quais as decisões devem ser tomadas praticamente de forma instantânea, o uso deste tipo de ferramentas passa a ser essencial por deixar de depender tanto das análises "*offline*" realizadas através de algoritmos pré-definidos e interligar mais com a opinião humana. Por outras palavras, a incorporação de inúmeros volumes de dados processados numa solução *Big Data*, num conjunto instantâneo de diversos tipos de representação gráfica, causa um melhoramento na forma como as empresas agem e vêm o mundo do negócio.

2.2 Inteligência Operacional

Com o avanço do mundo das tecnologias e dos sistemas de *Tecnologias de Informação* (IT), a necessidade de existir uma evolução gradual perante novas necessidades, que eventualmente acabam por surgir devido a esse avanço, torna-se elevada. Como consequência, verifica-se o surgimento de uma nova geração de sistemas de IT, na qual a captura de informação relativa ao que vai ocorrendo no sistema torna-se insuficiente, surgindo assim no mercado, soluções de Inteligência Operacional.

A Inteligência Operacional define-se pela capacidade de resposta à seguinte questão: "O que está a acontecer exatamente neste preciso momento no sistema?" (CITO Research, 2013), ou seja, permite compreender melhor o estado atual dos sistemas e suas infraestruturas tecnológicas, em tempo real, sendo deste modo facilitado o tempo de ação e possibilitar a tomada de decisões de forma mais rápida e concisa. Segundo pesquisas por parte da organização *CITO Research* (empresa responsável por uma rede de pesquisas e análises no mundo da tecnologia), a IO define-se como um principio de organização, importante para sistemas e métodos capazes de mover um negocio em tempo real e ganhar conhecimento sobre velocidade e agilidade (CITO Research, 2013).

Por outras palavras, trata-se de uma nova abordagem de análise que fornece visibilidade aos processos desenvolvidos no ramo do mundo empresarial, a eventos e operações à medida que vão sendo executadas. Através do recurso à IO, as empresas passam a conseguir agir com base na informação, a executar atualizações constantes na manutenção dos *dashboards*, a incentivar os seus clientes a ajustar os mecanismos usados para o processamento dos dados e até mesmo a prevenir eventuais corrupções ao sistema (Russom, 2013).

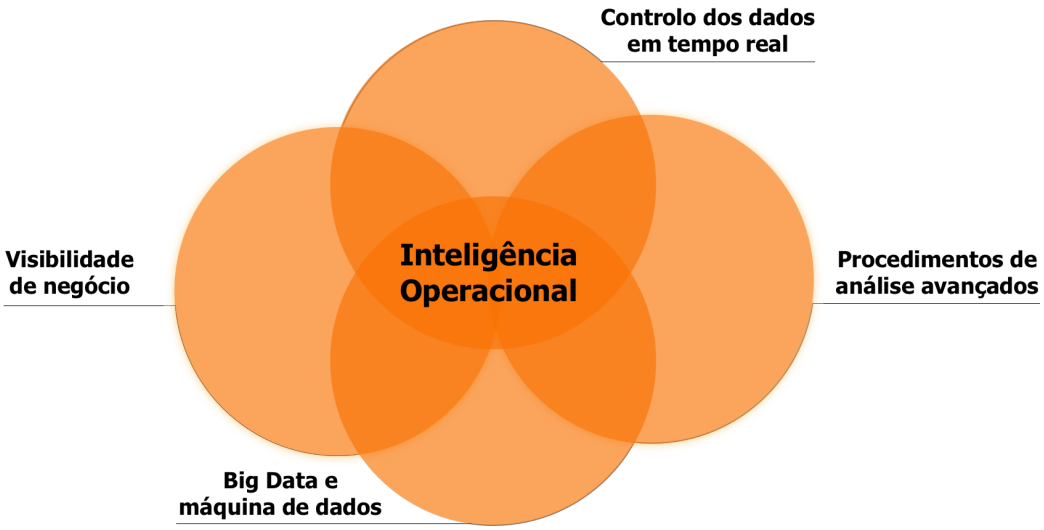


Figura 2.2: Definição de Inteligência Operacional (Russom, 2013)

A IO recorre, assim, a um conjunto de métodos e tecnologias que permitem fornecer maior visibilidade ao negócio e estabelece suporte a um conjunto de diversas tecnologias essenciais para a resolução de problemas que abordem a manipulação de conjuntos de dados na ordem dos *PetaBytes*, tais como máquinas de dados, sensores de dados, eventos provenientes de *streams* (canal que agrega um fluxo de dados em tempo real) e outras formas de *streaming* de dados, e, mais recentemente, soluções *Big Data*.

Como se pode observar na Figura 2.2, o conjunto de métodos e tecnologias que definem IO, designam-se por (Russom, 2013):

- **Controlo dos dados em tempo real**: suporte à captura e processamento dos dados quase em tempo real, ou seja, não em questão de horas, mas sim em questão de segundos ou até mesmo milissegundos, de diversas fontes, como por exemplo, dados recolhidos de *streams* ou mensagens em *queue*, entre outros tipos.
- **Procedimentos de análise avançados**: permitem estabelecer ligação entre eventos relacionados e criar assim novos padrões. Desta forma, verifica-se o aumento do conhecimento e a facilidade da descoberta de possíveis problemas e de oportunidades que mereçam uma atenção imediata.
- **Big Data e máquina de dados**: processa e analisa diariamente inúmeros *TeraBytes* de conjuntos de dados e dezenas de *TeraBytes* a *PetaBytes* em dados depositados no histórico, alcançando todos os tipos de dados, desde os relacionais aos de texto processado por linguagem humana (*raw data*) com ênfase em tempo real sobre máquinas de dados. Devido a este suporte, esta área tem evoluído consideravelmente e, nos dias de hoje, uma solução *Big Data* é bastante utilizada para análise, não só de dados estruturados em formato *raw*, como também de dados semiestruturados sem qualquer tipo de estrutura. Um bom exemplo disso, são as redes sociais como o *Facebook* e o *Twitter* que já possuem no seu sistema este tipo de soluções. Através da integração deste tipo de soluções nas tecnologias de base de dados tradicionais, consegue-se alcançar nos dias de hoje, uma abordagem mais eficaz, completa e vital na análise dos dados obtidos por estas tecnologias. Isto porque em várias situações, não é possível encontrar o valor por *byte* da informação até se aplicar esforço e dinheiro em armazenamento, mas para isso é necessário haver garantias acerca do valor da informação antes do investimento. Desta forma conseguem-se utilizar dados relevantes para o aumento da eficiência, não só dos sistemas IT como também de outras partes do negócio. Entretanto, novas ferramentas foram surgindo como resposta a esta nova "necessidade computacional", sendo sujeitas a constantes atualizações à medida que todo o conhecimento relativo a soluções Big Data progride, visto se tratar ainda de uma tecnologia recente. Um exemplo deste tipo de ferramenta é o da plataforma open-source *Hadoop*, cujo

desenvolvimento é apoiado pela fundação *Apache*, e que suporta o armazenamento e processamento distribuído de *Big Data* em clusters de acessível hardware (Zikopoulos, Eaton, deRoos, Deutsch, & Lapis, 2012). Por outro lado, o suporte às diversas máquinas de dados que existem torna possível o alcance a novas fontes de informação, extraindo dados retidos nas atividades e comportamentos dos clientes, utilizadores, em transações, servidores, API's, dispositivos móveis, etc.

- **Visibilidade de negócio**: fornece uma visão mais completa das entidades de negócio, como também sobre dados obtidos quer em tempo real ou de forma intermitente (*batch*), apresentando assim os dados de forma mais simplificada e compreensível dentro dos termos do negócio.

Durante anos, o grande desafio de muitas empresas, passou pela dificuldade em captar, armazenar, correlacionar, analisar e visualizar todo esse conjunto de dados na ordem dos *PetaBytes* de modo eficiente, visto que a maioria dessas empresas continuava a restringir-se aos sistemas tradicionais e tecnologias BI para tomarem decisões. Emerge assim uma grande necessidade de se ter dados em tempo real ou armazenados em histórico ao longo de um amplo conjunto de geradores de eventos e aplicações. Atualmente, já existem inúmeras formas de agregar e compreender inúmeras formas de *streams* de dados através de métodos capazes de escalar com uma proporcionalidade direta com o crescimento dos dados. Consequentemente, surge uma nova geração de ferramentas de suporte na fase de análise, proporcionando a integração de conjuntos de gráficos para uma representação de dados mais simples e eficiente.

2.3 Visualização dos dados

A maioria das soluções *Big Data* já implementadas no mercado possuem baixo suporte na representação dos dados por conjuntos de *dashboards*. Isto deve-se ao facto de a maioria das empresas terem focado o seu desenvolvimento no uso de ferramentas de BI e armazenamentos em RDBMS, recorrendo a uma análise e sucessiva tomada de decisão com base em valores estatísticos. Os métodos convencionais de visualização dos dados restringem-se, fundamentalmente, a **Mapas**, **Gráficos**, **Charts**, **Diagramas** e **Tabelas**. Vejamos, então, cada um destes métodos:

- **Mapas**. Uma abordagem que corresponde ao tipo de representação mais comum na visualização de dados provenientes de soluções *Big Data*, por serem bastante escaláveis e permitirem que o sistema evolua para áreas mais específicas em conjunto com o utilizador. No fundo, os mapas acabam por ser um suporte ao utilizador com um número bastante limitado de classes inicial. Há medida que o utilizador manipula a visualização, o número de classes difere com base na área abrangida. Em questões de *design*, é necessário a execução de programas como o *XDATA* que recorre ao processamento de símbolos para a representação dos dados no mapa (Chris, Xinle, William, & Jihoon, 2014).

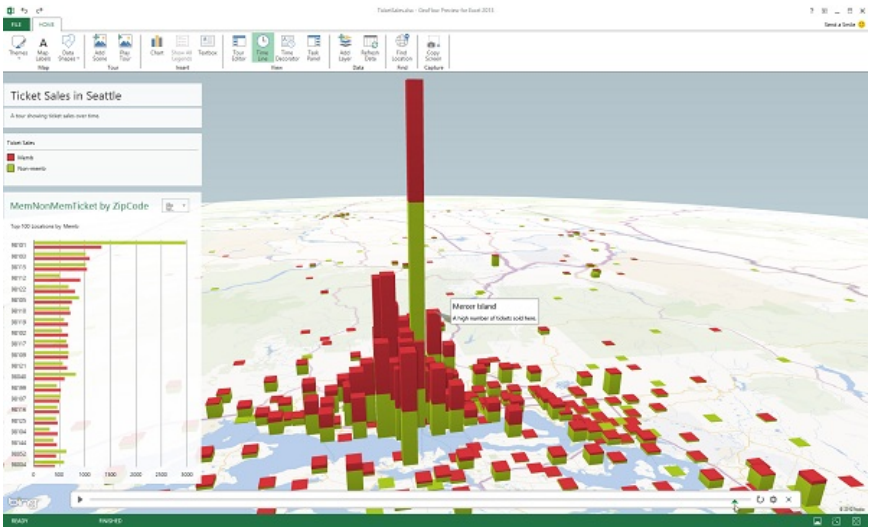


Figura 2.3: Exemplo de Mapa na representação da estatística dos saldos em Seattle

- **Gráficos e Charts**. Este tipo de abordagem acaba por ser utilizado na representação de um mesmo tipo de distribuição de informação, mas existem ligeiras diferenças nas suas características. Nos gráficos verifica-se sempre uma interpretação mais complexa e, de certa forma, que ilustra uma maior informação à audiência. Porém, não raras as vezes tal não é alcançado devido à sua complexidade. Com *charts*, designados por gráficos de pizza, já é possível comunicar de forma mais eficiente, por se basearem mais na agregação de dados e na elaboração de resumos de informação, exigindo assim interpretações mais simples (Chris, Xinle, William, & Jihoon, 2014).

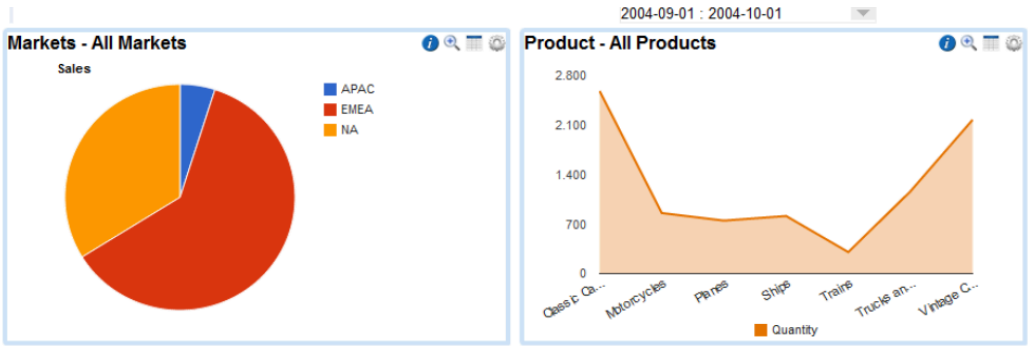


Figura 2.4: Exemplo de gráficos, transpondo informação mais detalhada

- **Tabelas**. O uso de tabelas é essencial num painel de *dashboards*, uma vez que permite aos utilizadores analisar a informação com maior detalhe, estando esta organizada por atributos, permitindo revelar dados extras que por outros tipos de representação seria impossível (Chris, Xinle, William, & Jihoon, 2014). Com o uso de tabelas, torna-se também possível agregar um outro conjunto de *dashboards* que podem interagir com cada elemento da tabela, revelando mais informação ao utilizador.

Com o aumento do uso de soluções *Big Data*, muitas empresas começaram a aperceber-se da importância dos *dashboards* na obtenção de uma visão completa da informação, na pesquisa de novos valores e na observação de possíveis alterações sobre um dado mercado. Como consequência, uma nova era de *dashboards* emergiu, uma vez que os convencionais métodos de visualização dos dados não se aplicam a este tipo de soluções. Surge assim uma extensão nas já conhecidas formas de representação de dados, sendo estas visualmente mais apelativas e intuitivas, mais flexíveis e com mais funcionalidades, permitindo assim uma maior interação com os utilizadores.

Figura 2.5: Exemplos de visualização de dados de diferentes épocas, sendo a zona cinza referente a uma época mais antiga

2.3.1 Plataformas no mercado

Com o aumento progressivo da informação gerada pela continua expansão de inúmeras fontes de dados e o aumento no número de empresas que recorrem a soluções *Big Data* para implementarem seu negócio, verifica-se o crescimento de plataformas de visualização de dados no mercado. A aposta neste tipo de plataformas na área da visualização, surge como consequência das novas necessidades criadas pelo avanço do mercado, nomeadamente, da importância de uma empresa em conseguir agir com maior rapidez e precisão mediante as súbitas alterações do mercado. Assim, uma empresa consegue pelo menos manter uma maior estabilidade económica ou no melhor caso conseguir se destacar da concorrência e alcançar um aumento dos seus valores. Contudo, sendo ainda um novo mercado para investimento, muitas das plataformas já desenvolvidas possuem algumas lacunas em questões de *rendering*, segurança, elevada latência no processamento, entre outros. Na Tabela 2.1, apresenta-se com maior detalhe a informação recolhida sobre algumas das plataformas de BI de maior destaque, salientando-se os pontos positivos e os pontos mais críticos, levantados pela maioria dos seus utilizadores.

Plataformas	Pontos Positivos	Pontos Negativos	Processamento em Tempo Real?
Datameer	Plataforma de análise que escala com as necessidades do seu utilizador.	Fraca representação dos dados e não possui uma interface prática para interagir com o utilizador.	Não, o processamento de imagens e gráficos é efetuado entre horas ou dias.
Splunk Enterprise	Fácil de usar e de monitorizar todo o tipo de dados; Escalável.	Torna-se bastante dispendioso à medida que o volume de dados aumenta.	Sim.

Tableau	Gera rapidamente diferentes tipos de representação (suporta ainda métodos de análise tradicionais) para os diversos tipos de dados; possui boas características para visualizar informação que possua elementos geoespaciais.	Problemas de segurança: (exemplo); Não se trata de uma plataforma ideal quando se pretende efetuar um resumo de um conjunto grande de dados (>1 milhão de linhas de tabela), tornando a resolução de imagem bastante lenta.	Não, o processamento demora alguns minutos.
ZoomData	Analisar todo o tipo de dados; Resolve a redundância da informação; Os utilizadores podem estabelecer ligação, visualizar e interagir facilmente com os dados em browsers e em dispositivos móveis;		Na maioria não, isto é, apenas permite uma proximidade de visualização dos dados em tempo real em duas das suas ligações a plataformas de processamento de dados.
Zoho Reports	Possui uma interface user friendly; Plataforma com suporte a ferramentas inteligência de negócio, online e mobile;	Não consegue escalar para as necessidades das grandes empresas;	Não.
MicroStrategy	Garante níveis de segurança (autenticação, autorização, entre outros); suporta diversos estilos de Inteligência de negócio; permite a reutilização dos dados;	Para muitos utilizadores, não se trata de uma plataforma fácil de usar na exploração e análise dos dados; Falta de eficiência na visualização dos dados e fraca interface.	Não, apenas possui processamento em tempo real na área de inteligência de negócios.

Tabela 2.1: Comparação entre algumas das plataformas atualmente existentes no mercado

Numa pequena observação, fica-se com a ideia de que a maioria deste tipo de plataformas restringe-se a um ambiente específico e verifica-se um nível mais baixo de interatividade com o utilizador, tendo uma interface pouco prática e mais complexa do que aquilo que era esperado pela comunidade abrangida. Com a observação da última coluna, consegue-se verificar que a maioria destas plataformas não executa o processamento de imagens em tempo real como muitas afirmam, distorcendo o verdadeiro significado do conceito de "real-time". De facto, o fator tempo é bastante influenciado, quer pelo hardware utilizado, quer pelas ferramentas utilizadas no desenvolvimento do processamento, como também pelos métodos de seleção de *datasets* para processamento das imagens/gráficos abordados.

2.4 Vantagens

Além das vantagens que já existem nos métodos convencionais de visualização, isto é, o promover da visualização inteligente da informação, a fácil detecção de falhas existentes nos processos de análise e de monitorização dos dados, existem outras potenciais vantagens em usar IO, tais como:

- **Melhor comunicação**. Uma das vantagens no recurso a este tipo de métodos, passa pela facilidade da comunicação entre utilizadores, ou seja, pela partilha de uma linguagem capaz de simplificar problemas complexos e permitir uma ligação maior entre vários tipos de utilizadores. O fortalecimento da ligação entre utilizadores deve-se essencialmente pela representação de uma mesma "história", onde qualquer elemento possa, num curto espaço de tempo, atingir melhores níveis de compreensão e intervir mais para o crescimento da empresa.

- **Melhor desempenho na fase de elaboração de decisões.** Membros de diferentes graus numa determinada empresa, passam a compreender melhor as relações estabelecidas entre os dados, debater entre si e agir com maior rapidez perante as situações que vão surgindo. Deste modo, torna-se possível tomar as devidas decisões

sobre onde focar essencialmente a pesquisa, verificando-se como consequência, um aumento da produtividade (Russom, 2013). Na área empresarial, tempo acaba por ser dinheiro, e, numa empresa que se baseie pelos seus dados, se surgir algum incidente, uma reação lenta a esta alteração pode tornar-se numa descida em questões económicas. As ferramentas de IO, impedem que situações dessas ocorram, devido à sua velocidade de processamento, resolvendo os incidentes de forma mais rápida.

- **Flexibilidade e um design mais intuitivo**. A visualização dos dados necessita de ser flexível e intuitiva no design para que dessa forma os utilizadores consigam retirar vantagens da interface e averiguar melhor as soluções que respondam com maior eficácia às necessidades, estando assim o painel em constante mudança (Chris, Xinle, William, & Jihoon, 2014).
- **Complemento de *BI/DW* pela execução de operações em "verdadeiro" tempo real**. A prática tradicional de *BI* envolve a transição de *queries* de dados, sendo estes eventos recentes e o desempenho por dados estruturados, captados em algumas horas ou minutos. Ou seja, mesmo que os dados retidos numa base de dados, não contenham um valor tão elevado em relação aos recentes dados coletados, acaba na mesma por se deixar de lado uma parte da "imagem" dos dados armazenados (Russom, 2013). O que não acontece com o recorrer à IO, em que se visualizam todos os dados a partir do momento em que entram no sistema.

O facto de se lidar com os dados em tempo real, acaba por trazer consigo mais aspetos positivos e as empresas atingem melhores resultados ao combinarem a sua prática e experiência em situações do mercado com toda a informação de qualidade recolhida pelas ferramentas de análise utilizadas.

2.5 Desafios

À medida que se verifica um aumento do tamanho de uma solução *Big Data* e da diversidade dos seus dados, a monitorização dos métodos de visualização de dados torna-se cada vez mais complexa devido, essencialmente, aos seguintes aspetos:

- **Gestão dos dados**. O facto da visualização dos dados ser feita em proximidade a um tempo real, leva a que as empresas consigam monitorizar métricas chaves para o negócio de modo eficiente e responder aos desafios de modo imediato. No caso de uma medição de tendências a longo prazo, é necessário recorrer aos dados armazenados no histórico. Atualmente, com grandes quantidades de pontos de informação úteis a serem gerados a cada minuto, é necessário alcançar um *dashboard* que processe em tempo real (ou próximo de tempo real), permitindo deste modo às empresas, visualizar os dados e maximizar o impacto, tornando-se assim numa ferramenta extremamente poderosa (Nyland, 2013).
- **Erros e incertezas**. Lidar com dados provenientes de diversas fontes como *social media*, aumenta a probabilidade do surgimento de erros e incertezas, apesar destas últimas terem usualmente uma probabilidade quase de 50% em todos os processos de análise, de qualquer tipo de dados. Por exemplo, amostras, transformação ou a filtragem de dados podem induzir o surgimento de erros e de inconsistências na visualização de dados (Liu, Cui, Wu, & Liu, 2014-12). Ultrapassar este desafio, passa pelo entendimento dos métodos de visualização existentes, de forma a atribuir o mais adequado a cada aplicação.
- **Ruído visual**. Quando diversos conjuntos de dados possuem inúmeras relações com outros conjuntos, cria-se uma rede bastante extensa e densa de dados devido à sobreposição de pontos. Visualmente, isto torna a fase de *rendering* mais complexa, sendo mais difícil repartir os dados e formar novos grupos de interesse (Wang, Wang, & Alexander, 2015).

- **Perda de informação.** Um dos grandes desafios encontrados, em praticamente todo o tipo de abordagens que envolva grande quantidade de dados, é a existência de uma probabilidade elevada na perda de dados, desde o momento que são captados até à sua representação. Praticamente, torna-se complexo evitar esse tipo de falhas, visto que, por exemplo, para se obter um painel de *dashboards* conciso e legível, é importante que os dados passem por uma fase de filtragem e *sampling*, excluindo dados que não se enquadrem (Wang, Wang, & Alexander, 2015).

Além de tudo o que acabámos de referir, encontram-se outros desafios, como o mau desempenho por parte dos algoritmos de visualização utilizados. Por motivos de má seleção, tendo em conta as condições de análise que uma empresa tenha no momento, estes podem influenciar os resultados obtidos pelos *dashboards*. Para além disso, os próprios desafios encontrados numa solução *Big Data* relacionam-se como base de todo o sistema envolvido e podem, consequentemente, influenciar o desempenho de um painel de *dashboards*. Alguns exemplos desses desafios são:

- **Diversidade numa *stream Big Data* em tempo real**. A análise em tempo real, correlaciona-se com falhas que possam emergir nas inúmeras fontes de *Big Data* de uma *stream*, como por exemplo, nos servidores web, máquinas de dados, entre outros.
- **Elevada contagem de mensagens pequenas**. Existe uma possibilidade de se criarem pequenas mensagens. O procedimento inicia-se na captação de cada evento de uma *stream*, separando por eventos de interesse, retirando de seguida o ruído contido nos dados. Após a filtragem dos dados, efetuam-se as correções com outras *streams* e base de dados, reagindo a alguns eventos em tempo real e, por fim, armazena-se os eventos provenientes de análises *offline*. No decorrer deste procedimento algumas falhas podem acontecer. Por exemplo, a atribuição errada de um dos algoritmos de filtragem a um conjunto de dados, que consequentemente origina pequenos blocos de dados sem interesse para a fase de análise devido à falta de enquadramento.

- **Segurança ou privacidade**. Caso esta propriedade falhe, os dados facilmente são manipulados por terceiros, e como consequência de os dados serem corrompidos, a visualização desses dados irá acabar por deixar de fazer sentido ou conter qualquer valor importante no contexto de posteriores análises.

Muitos dos desafios mencionados, tais como o ruído visual, a diversidade de dados numa *stream*, a gestão de dados, os erros e as incertezas, podem no, melhor dos casos, serem evitados ou acontecer que, pelo menos, as probabilidades de ocorrência de alguns desses sejam reduzidas através do recurso adequado de técnicas de redução de dados, como *sampling, filtragem, clustering, PCA ou multidimensional scaling*. Embora muitas dessas técnicas tenham obtido sucesso, nenhuma é considerada "perfeita". Ainda hoje se verificam algumas dificuldades em estabelecer um equilíbrio entre um nível elevado de uma visão geral sobre os dados e um nível baixo com a informação em modo mais detalhado. Alguns dos processos de pesquisas levados a caso abordaram a importância da interatividade entre o painel de *dashboards* e os seus respetivos utilizadores, de forma a que este contribua com o seu conhecimento no projeto e responda com maior eficácia às suas necessidades.

Capítulo 3

Sampling

3.1 Contextualização

Quando o conceito "visualização" começa a ganhar impacto e passa a ser o foco de inúmeras discussões realizadas globalmente - visto que a principal preocupação da atualidade, passa por encontrar a melhor forma de se processarem os dados, sem que se verifique um aumento dos custos - torna-se inevitável o surgimento de alguns problemas, quer na análise como na representação dos inúmeros conjuntos de dados recolhidos, em segundos, nas diversas fontes e formatos disponíveis. Assim, numa primeira observação, costuma-se averiguar que à medida que os dados são recolhidos torna-se incontornável a necessidade de se tolerar valores elevados de latência relativamente à fase de processamento, não havendo garantias de que a informação seja devidamente representada, de forma explicita e compreensível, para uma posterior análise. Por forma a contornar este problema, muitas empresas recorrem a estratégias de redução analítica ou dimensional da informação antes de se iniciar a fase de análise e de visualização. Este recurso exprime-se na redução da complexidade dos dados através de modelos matemáticos, mantendo-se as características essenciais dos dados inalteradas.

Ao longo do tempo, novos métodos de redução foram estudados e desenvolvidos, de forma a ser possível atingir uma resposta diretamente proporcional ao aglomerado de dados que vão sendo recolhidos da fonte e tentar alcançar ao mesmo tempo, uma melhoria na eficiência da fase de processamento. *Sketching* e *Sampling* são dois exemplos deste tipo de estratégias.

O sketching é uma técnica que agrega a informação de forma aleatória, como resumos de *stream* consoante a sua chegada em pequenos blocos, designados por "sketches" para a fase de *rendering*. Através de uma pequena porção de memória é criado um "esboço" do

gráfico/imagem, que em modo continuo, rápido e compacto, são realizadas atualizações à medida que novos blocos são recolhidos. Embora seja uma técnica bastante utilizada por algumas plataformas, como *ZoomData*, não será um foco nesta dissertação.

Quanto à segunda técnica referida, o *sampling*, esta assenta num processo de seleção de subconjuntos compostos por unidades de uma população pré-estabelecida, como parâmetro de trabalho, sendo esta um conjunto de objetos ou de dados que representam um universo de estudo, a ser posteriormente analisado. No decorrer deste tipo de abordagem, averigua-se a preocupação na absorção de uma pequena "parte" que represente de forma aproximada, a informação de um "todo", ou seja, para uma dada população é retirado um conjunto de dados relativamente mais pequeno, que seja capaz de representar toda essa população em futuras medições. Apenas existem duas formas de abordar métodos de *sampling*: a ***probability sample***, que assenta a sua base na matemática probabilística e a ***non-probability sample***, que se baseia na probabilidade de selecionar uma amostra que é totalmente desconhecida, não sendo possível formar amostras representativas da população em estudo.

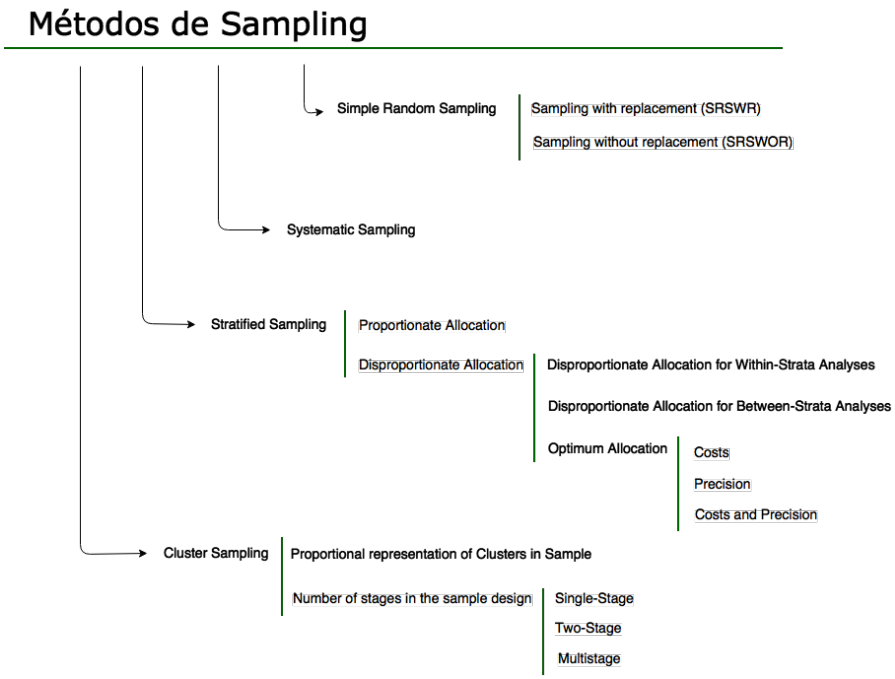


Figure 3.1: Conjunto de métodos de sampling.

Atualmente existe um conjunto de métodos com vários subtipos, que divergem mediante os objetivos que se pretende alcançar através do recurso a este tipo de estratégia de seleção (figura 3.1). Em cada observação, especificamente, ou em cada caso de estudo abordado, efetua-se uma medição de uma ou várias propriedades que permitem realizar a distinção entre objetos independentes ou individuais e uma seleção adequada do método de *sampling* a ser implementado. Estas são propriedades que, na maioria dos casos, se restringem ao tamanho de uma amostra e ao grau de confiança nos resultados pretendidos. Isto é, em termos de probabilidade, é a margem de erro que se pretende alcançar. Sendo as soluções *Big Data* que envolvem milhares de dados recolhidos em segundos, ou até mesmo em milissegundos, a "base de estudo" desta dissertação, e considerando um dos objetivos atingir, emerge da possibilidade em analisar toda essa informação nas mesmas "porções" de tempo. Como tal, torna-se essencial o procedimento de técnicas de *sampling*, com princípios matemáticos sólidos para maior eficiência do processo de análise de resultados.

3.2 Métodos de Sampling

Atualmente existem inúmeros métodos desenvolvidos que permitem a abordagem do conceito de *sampling* em diversos casos de estudo, nomeadamente: *Simple Random Sampling*, *Systematic Sampling*, *Stratified Sampling*, *Cluster Sampling*. Cada um destes métodos apresenta diferentes características e objetivos, diferenciando-se de todos os outros, o que faz com que cada um deles tenham as suas próprias especificidades de aplicação.

3.2.1 Simple Random Sampling

A aplicação deste método dá origem à formação de subconjuntos de dados que se baseiam numa propriedade igualmente distribuída, conhecido como EPSEM (Equal Probability Selection Element Method). Contudo, não corresponde a um dos métodos mais utilizados por falta de eficiência na representação de uma população, contribuindo, desta forma, para o surgimento de erros de *sampling*. Este facto deve-se à abordagem aleatória realizada no processo de

seleção de elementos, contribuindo, deste modo, para a existência de uma possibilidade em adquirir, como resultado, um conjunto de dados que não seja capaz de refletir o conjunto de características da população em causa.

Neste método a abordagem pode ser feita por *sampling* com recolocação do elemento selecionado para amostra na população ou por *sampling* sem recolocação de elementos, removendo todo o elemento da população à medida que seja selecionado. Desta forma, *sampling* sem recolocação torna-se mais eficiente por não permitir que elementos voltem a ser novamente selecionados, verificando-se assim a produção de amostras distintas e mais representativas (Daniel, 2012). Uma das grandes vantagens de SRS, passa pela sua simplicidade na análise dos resultados e na redução existente na construção de padrões, atingindo deste modo, tempos de processamento mais baixos e, consequentemente, uma maior rapidez no acesso aos resultados. Analisemos um pequeno exemplo:

> *Perante uma situação de mudanças de uma família para uma nova residência, um dos membros pretende selecionar um conjunto de livros para ler nos primeiros tempos, enquanto não transporta o resto da sua coleção. Decide, assim, levar apenas 4 livros de duas categorias diferentes. Este método atua neste caso com eficácia ao selecionar de forma aleatória dois livros de dois conjuntos diferentes, correspondendo cada um a uma lista de livros de uma determinada categoria (Figura 3.2).*

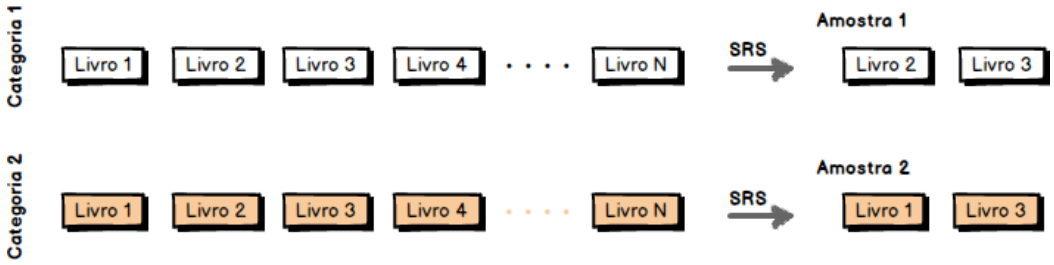


Figura 3.2: Esquema de um exemplo de abordagem do método SRS

3.2.2 Systematic Sampling

Este método baseia-se num processo de análise de uma população pela formação de um esquema ordenado, no qual este é representado através de subconjuntos de dados por forma a identificar padrões e tendências. Desta forma, pode-se averiguar a criação de um ponto inicial de uma seleção com intervalos regulares e fixos, sendo parcialmente aleatório (importante que o ponto se encontre entre o primeiro e o K elemento do subconjunto selecionado) e servindo apenas de referência ou como marco para o procedimento da fase de seleção, que passa a ser a cada K elemento, sendo atribuído ao K o tamanho de uma dada população ou amostra. Em termos de eficiência, durante a fase de estratificação induzida do método, esta acaba por ser bastante influenciada, tanto pelo algoritmo de seleção do ponto inicial como pela variável do esquema formado a partir da população. Isto se for possível verificar uma correlação com a variável de estudo. Em suma, trata-se de um método bastante abordado em casos nos quais a população é logicamente homogénea e se verifiquem conjuntos de dados com dimensões maiores relativamente a SRS, ganhando, desta forma, mais eficiência no processo de análise como um todo.

Muitas vezes este tipo de método confunde-se com a SRS, por causa de todos os elementos terem uma mesma probabilidade. Porém difere num pormenor de que todos os subconjuntos, embora tenham o mesmo tamanho, possuem probabilidades de seleção diferentes entre eles. Caso o intervalo de *sampling* pré-definido, abranger padrões existentes na população, eventualmente será gerada uma rutura no processo de aleatoriedade durante a seleção dos elementos, originando amostras, não uniformemente distribuídas, sobre a população em causa. De forma semelhante ao caso anterior, analisemos também um pequeno exemplo:

> *Perante uma situação de mudanças de uma família para uma nova residência, um dos membros pretende selecionar um conjunto de livros para ler nos primeiros tempos enquanto não transporta o resto da sua coleção. Este sabe que pretende levar k:4 livros da categoria terror, mas encontra-se indeciso e por fim decide*

organizar os livros por datas de aquisição, limitando a escolha entre os anos 1990 e 2000.

Com o recurso a este método a seleção é efetuada através do método SRS, considerando como ponto inicial, o Livro 3. A partir desse ponto é então efetuado a seleção de um novo elemento a cada k elemento, ou seja, num intervalo de k-1 unidades (Figura 3.3).

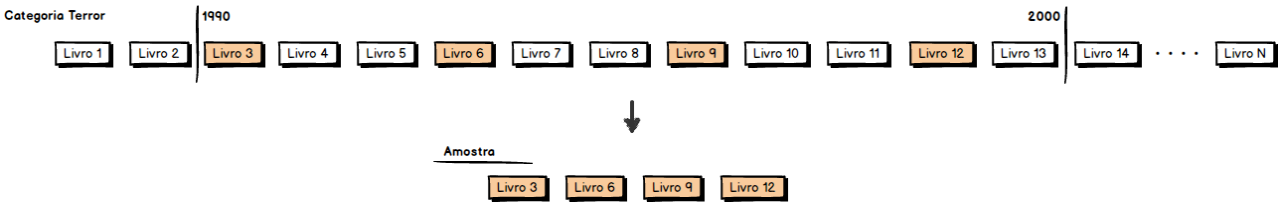


Figura 3.3: Esquema de um exemplo de abordagem do método systematic sampling.

3.2.3 Stratified Sampling

Esta técnica é uma extensão do método SRS que, perante uma população, efetua uma agregação de conjuntos com características distintas e independentes em conjuntos homogéneos designados por *strata*. Em cada *strata* é posteriormente aplicado um processo de seleção individual (recurso ao método SRS ou *systematic*), dando origem aos *stratum*. A obtenção da amostra final resulta de uma nova seleção dos elementos de cada *stratum*. Cada *strata* é visto como uma subpopulação resultante de uma seleção aleatória de elementos, mantendo-se na mesma representação da população, mas tornando possível a progressão de precisão nas estimativas elaboradas sobre toda a população, através da inclusão de uma maior precisão da informação nas amostras formadas. As variáveis de estratificação são características que influenciam a formação dos subconjuntos. Uma variável pode ser aplicada a mais do que uma estratificação. Neste processo deve existir algum controle sobre a criação

de variáveis, caso contrário, muitas delas poderão entrar em conflito e, deste modo, cancelarem o seu efeito no procedimento.
Em termos de especificação dos tamanhos das amostras, existe uma margem de erro na definição do tamanho de cada *strata*, uma vez que tal pode originar limitações no procedimento da criação de um *stratum*. Quanto ao tamanho de cada *stratum*, este é estabelecido por alocação proporcional ao tamanho de cada *strata* (significa que a cada *strata* vai ser aplicada na mesma porção no procedimento de simplificação), para que desta forma a amostra resultante efetue um "match" com a população em causa.

Existem duas formas de se abordar este método, com base na proporcionalidade do tamanho dos conjuntos formados no decorrer do seu procedimento: a estratificação proporcional e a estratificação não proporcional. Basicamente, o que diferencia estes dois tipos é a teoria de proporções aplicável ao número de elementos que compõem as amostras obtidas em cada *strata*, igualmente com as mesmas porções no número de elementos de cada *strata* na representação da população no total. O recurso a este método passa por efetuar uma estimativa referente aos parâmetros da população em análise, que, no caso da estratificação não proporcional, deve existir variáveis de peso na composição da população para se tornar possível. Contudo, em certos casos, torna-se necessário efetuar uma análise mais detalhada relativamente a pequenos conjuntos de amostras, nomeadamente *stratums*, e fazer comparações entre *stratas*. Neste caso, o mais indicado é abordar a estratificação de forma não proporcional. Na estratificação não proporcional não é aplicado um procedimento EPSEM, ou seja, os elementos da população não possuem a mesma probabilidade de serem incluídos na amostra e, desta forma, um *strata* gera diversas frações de *sampling* (as várias amostras que se vai obtendo ao longo do método) e, assim, uma mesma fração de *sampling* acaba por não ser aplicável a cada *stratum*.

Com base na solução que se pretende alcançar através da alocação implementada, este tipo de abordagem sem recurso à proporcionalidade, divide-se em três subtipos: a alocação entre *stratas* e entre elementos de um mesmo *strata*, e a otimização da alocação em questões de otimização de custos e precisão. No caso de se efetuar uma alocação entre elementos de um

mesmo *strata*, o procedimento de análise requer um aumento do tamanho da amostra obtida pela estratificação proporcional, por sobreposição de amostras, passando assim a ser considerada uma abordagem desproporcional. Quanto ao caso da alocação entre *stratas*, a quantidade de elementos a ser selecionada para cada categoria deve ser o suficiente para que a análise a ser feita tenha o mínimo de informação necessária. Desta forma, por vezes, é feita a maximização do tamanho da amostra de cada *stratum*, podendo ocorrer um balanceamento na alocação à medida que o método é processado. Também pode ocorrer a formação de amostras com a mesma quantidade de elementos para permitir que uma comparação possa ser realizada de forma mais equilibrada.

Por fim, com o recurso à via de otimização de alocação, pode-se atingir níveis de precisão muito mais elevados que aqueles que podem ser obtidos na estratificação proporcional, e de acordo com o tipo de análise pretendida pode ser definido o tamanho da amostra de diferentes *stratas*. Com base nos aspetos intitulados pela análise, custos e precisão, as frações de *sampling* sofrem constantes alterações com base nos custos e na variabilidade verificada nos *stratas*. Por exemplo, quanto maior forem os custos de um conjunto de dados de um *stratum*, mais baixo será o tamanho da amostra selecionada (Daniel, 2012). Este tipo de abordagem, em comparação com a estratificação proporcional, é mais útil quando se pretende uma análise mais detalhada ou quando um *strata* varia em termos de custos na recolha dos dados e na variabilidade das variáveis de interesse. Um dos pontos positivos na prática deste método, consiste na possibilidade de se aplicar diferentes tipos de abordagens de *sampling* em cada *strata*. Assim, este método permite uma melhor análise em cada subconjunto da população, devido ao facto de cada *strata* ser considerada como uma população individual. Por outro lado, mesmo que o uso deste método se adeque a um caso de estudo, pode-se verificar um aumento dos custos e da complexidade na sua implementação, assim como na escolha do tamanho das amostras de cada *strata* para ir de encontro com o tamanho total N da amostra previamente especificado.

Por forma a alcançar uma maior eficiência no seu desempenho deste método, deve ser possível verificar-se um determinadp conjunto de condições, tais como a variabilidade dos

elementos nos *strata* e nos *stratum*. As variáveis também devem ser definidas com base nas subpopulações criadas, sendo estas fortemente correlacionadas com a variável dependente pretendida. Em questões de variabilidade, nos *strata* deve-se maximizar a diferença e nos *stratum* deve ser feita uma minimização da diferença de elementos. A identificação e respetiva implementação de cada *strata* tem de ser cuidadosamente bem planeada, uma vez que contribui para o aumento dos custos e da complexidade da seleção de amostras, o que consequentemente provocará um aumento da complexidade no calculo de estimativas da população.

A seleção do número de elementos de cada *stratum* é efetuada de forma aleatória, e deve ser realizada uma seleção de pelo menos um elemento de cada *stratum,* para a sua representação na amostra, e outra, de dois elementos, para o cálculo da margem de erro das estimativas calculadas a partir dos dados recolhidos. Em casos em que se averigúe um elevado número de *stratas* ou em que o tamanho de cada *stratum* obtido pela estratificação de *stratas* seja bastante reduzido, tem-se a necessidade de constituir amostras potencialmente maiores em comparação aos restantes métodos, não ultrapassando, na maioria dos casos, o tamanho das amostras geradas pelo método SRS. A grande desvantagem deste método reside quando se está perante uma baixa variabilidade de elementos de uma população, não sendo assim possível efetuar a partição em subconjuntos disjuntos. Tomemos em consideração um pequeno exemplo para análise:

> *Perante uma situação de mudanças de uma família para uma nova residência, um dos membros pretende selecionar um conjunto de livros para ler nos primeiros tempos enquanto não transporta o resto da sua coleção. Sabendo que pretende levar apenas 4 livros onde se verifique a presença de duas categorias diferentes para os primeiros dias, elege apenas dois dos seus autores favoritos para a sua escolha.*

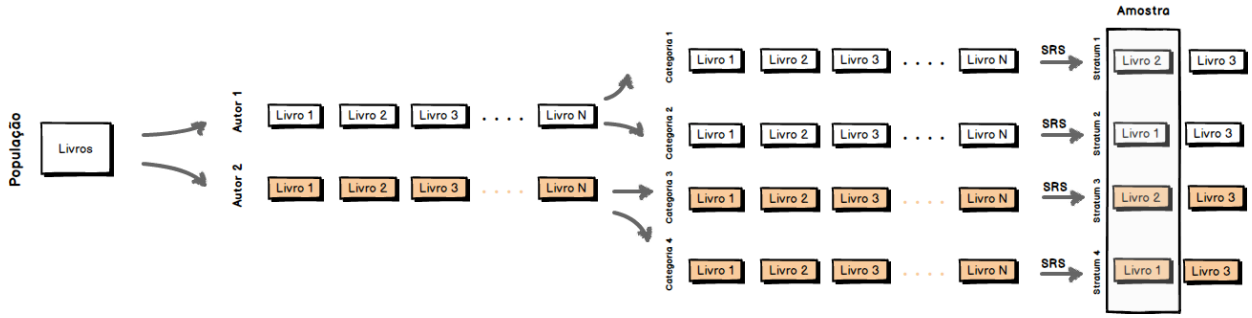


Figura 3.4: Esquema de um exemplo de abordagem do método de estratificação proporcional

Através deste método, são criados dois subconjuntos da população inicial com base no atributo pré-definido, neste caso os dois autores. A partir desses subconjuntos é feita uma nova divisão com base nas categorias dos livros, no qual é aplicado o método SRS para efetuar a seleção aleatória de dois elementos de cada categoria. Como resultado obtém-se uma lista de livros de cada categoria, nomeadamente *stratas*, da qual é retirado um elemento de cada para a formação de uma amostra.

3.2.4 Cluster Sampling

Neste método efetua-se uma divisão da população por parâmetros, tais como, geográficos, temporais, etc., em subconjuntos da população designados por *clusters*. A formação de amostras passa pela seleção de clusters através do método SRS e não pela recolha de elementos de cada subconjunto, tal como ocorre no método de estratificação. Cada cluster deve representar a heterogeneidade e coerência da população como um todo e manter-se homogéneo relativamente aos restantes clusters. Em caso de falha não haverá qualquer problema em termos de perda de informação, basta ser feita uma nova seleção de clusters. A variabilidade nas amostras que este método fornece em relação aos restantes métodos, é relativamente influenciada pelo grau de diversidade existente entre clusters, comparada com a variação local entre clusters. Para que seja atingido o mesmo nível de exatidão e manter a verificação de custos mais baixa em relação à do método SRS é necessário criar amostras com tamanhos maiores em relação ao método SRS.

Existem dois tipos de abordagens de *clustering*, uma que se baseia no número de estados do design da amostra e a outra que se baseia na representação proporcional dos *clusters* na amostra no total. Na abordagem Base no número de estados, um estado designa-se por um passo realizado no processo de *sampling*, de onde é obtida uma amostra. Existem 3 tipos de design de amostras que se diferenciam mediante o número de passos efetuados durante a sua formação, nomeadamente:

- **Single-stage**: recolha de amostra num único passo, na qual os clusters e respetivos elementos são selecionados de forma aleatória.

- **Two-stage**: difere da *single-stage* na parte final (o mesmo ocorre em *multistage*), na qual a recolha da amostra, após a seleção de um ou vários clusters, se obtém por aleatoriedade dos elementos de cada cluster selecionado, formando-se assim um conjunto de subconjuntos. Por norma, a não ser que os clusters sejam homogéneos, este tipo é mais vantajoso que o *single-stage*.

- **Multistage**: utilizado perante situações de maior diversidade (por exemplo, áreas geograficamente maiores) que necessitem de um design mais complexo. Este envolve a repetição de dois passos básicos, nomeadamente, listagem e *sampling*. Em cada estado podem ser aplicados diferentes métodos de *sampling* (SRS, estratificação e sistemática) e em que o tamanho dos *clusters* é progressivamente mais pequeno.

Na abordagem Base na representação de clusters, verificam-se casos de amostras obtidas a partir de clusters, que partilham o mesmo tamanho e desta forma o design de amostras baseia-se num modelo EPSEM, onde a probabilidade de seleção é igual para todos os elementos. Em casos nos quais o tamanho das amostras se revele irregular, para que seja definido um modelo EPSEM, torna-se necessário abordar probabilidades proporcionais ao tamanho de cada amostra (PPS).

Assim, pode-se verificar que quanto maior o número de estados numa estrutura de cluster, maior a possibilidade de ocorrência de erros no processo de *sampling*. Isto porque, à medida que o número de estados aumenta, a seleção de PSUs passa para outros estados, alcançando unidades de *sampling* mais homogéneas. Este problema pode eventualmente ser minimizado, caso uma amostra contenha mais PSUs que SSUs ou mais SSUs que TSUs, etc. Tipicamente, este método é utilizado em casos em que se torne demasiado dispendioso listar todos os elementos da população ou, simplesmente, quando não se possui total conhecimento de toda a população. Nesses casos, deixa de fazer sentido a criação desse tipo de lista para o estudo em causa, porque a pesquisa de um elemento numa lista, em comparação com a localização de um elemento num cluster, conduz a custos relativamente mais elevados e a uma redução na eficiência, uma vez que não é garantido que o elemento extraído por SRS, contenha a informação procurada (Stephen, 2003). A grande desvantagem que pode surgir com o procedimento deste método é a obtenção de amostras que não representem a população, devido à extração em grupos de clusters que não revelam a existência de homogeneidade entre eles. Por outro lado, verificando-se as condições exigidas, a formação de amostras acaba por ser feita de forma mais eficiente através da simples escolha de *clusters*, facilitando a recolha de informação.

O Clustering torna-se bastante útil em situações nas quais a listagem de uma dada população se torne impraticável e dispendioso a qualquer processo de compilação sobre essa lista, por elevado número de elementos individuais de uma população. Ponderando a criação de uma lista, os custos associados às operações básicas, como pesquisa, são relativamente mais elevados do que simplesmente obter a informação pretendida através de um *cluster* que possua elementos idênticos. Um dos aspetos positivos deste método, é que, durante o processo de seleção, a possibilidade de ocorrência de *crossover* é praticamente nula, verificando-se desta forma a inexistência de elementos a pertencerem a mais do que um *cluster*. Por outro lado, é possível formar-se amostras de *clusters,* não homogéneos entre eles, perdendo-se desta forma a representação da população em análise. Ou seja, obtém-se valores de precisão e exatidão mais baixos. Vejamos o seguinte exemplo:

Perante uma situação de mudanças de uma família para uma nova residência em Dundee. Um dos membros dessa família encontrava-se a realizar um estudo sobre a influência da antiga Escócia na atualidade em inúmeras zonas da Escócia. Por forma a reduzir custos e tempo, decide ilustrar um conjunto de amostras correspondentes às zonas mais a norte da Escócia (Figura 3.5).

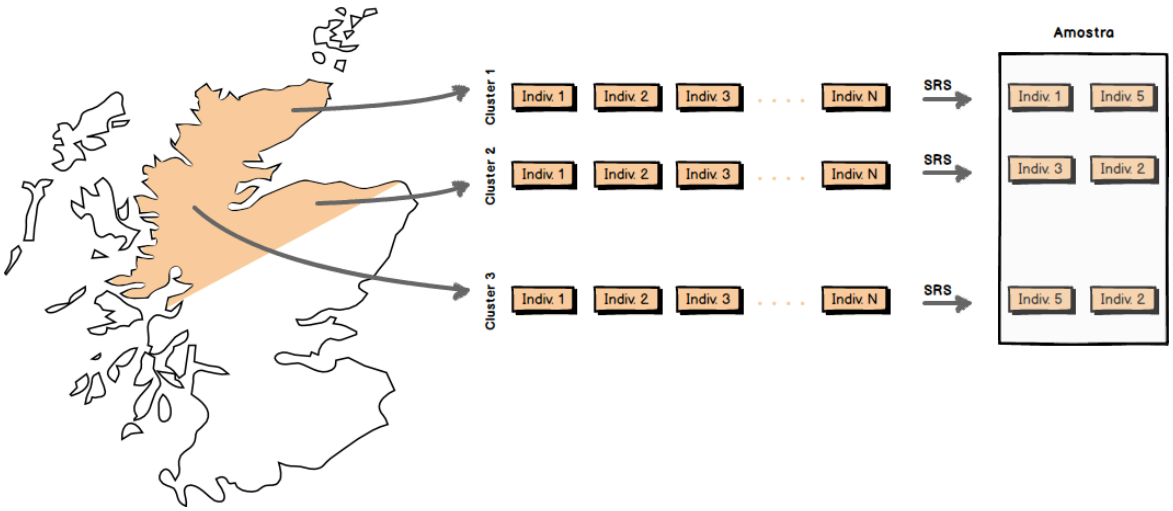


Figura 3.5: Esquema de um exemplo de abordagem de multistage com clustering e SRS

Com base na informação adquirida, e de forma a clarificar melhor as ideias sobre os métodos estudados, foram elaboradas duas tabelas com focos diferentes, nomeadamente: a primeira tabela (Tabela 3.1) foca-se nas características mais relevantes de cada método, enquanto que na segunda tabela (Tabela 3.2) já se verifica os pontos de maior destaque que acentua as diferenças entre eles, contribuindo assim para uma atribuição do método a um determinado caso de forma mais eficiente. A abordagem destes métodos, no caso prático desta dissertação, é feita de duas formas distintas. Na primeira abordagem verifica-se a perspetiva de um filtro, que há medida que novos dados vão sendo recolhidos de uma *stream* e introduzidos num canal de transmissão de mensagens, é aplicado um método de *sampling* que melhor se adeque ao agregado de dados em causa. Na segunda abordagem tem-se as imergentes *queries* que

eventualmente vão surgindo à medida que se pretenda obter informação mais especifica através da interatividade com o painel de *dashboards*.

Métodos	Tamanho da amostra	Tipo de seleção	Objetivo
Simple Random	Fixo	Amostra obtida pela seleção de elementos aleatórios diretamente da população	-
Systematic	Fixo	De forma aleatória é selecionado um elemento da frame (ponto inicial) sendo posteriormente estabelecido um intervalo regular com os valores das próximas posições da frame(ordenada) a serem extraídas	Criação de um conjunto de amostras uniformemente distribuídas sobre a população
Stratified	Todos os subconjuntos ficam com o mesmo tamanho, para que desta forma seja garantido no final do processo uma frame que represente uma pequena parte da população pretendida	Amostra composta por elementos de cada subconjunto (strata)	Aumento da precisão da amostra e representação da população; Estimar parâmetros da população.
Cluster	Definida inicialmente pelo utilizador (clusters com maior tamanho tem maior probabilidade em serem escolhidos)	Amostra composta por clusters (inclui todos os elementos dos clusters selecionados)	Redução de custos; Aumento de eficiência de processamento.

Tabela 3.1: Especificação de algumas das características de métodos de Sampling

Métodos	Pontos Positivos	Pontos Negativos
Simple Random	Fácil compreensão e comunicação; Tendência para produzir amostras mais representativas à população.	Surgimento de erros de sampling e menor precisão; Caso a população seja muito dispersa, verificam-se custos elevados; Pode não ser possível listar todos os elementos da população a partir da amostra obtida; Embora seja tecnicamente válida, considera-se pouco eficiente em casos cuja a dimensão da amostra é relativamente elevada.
Systematic	Fácil de se implementar; Aplicável essencialmente se a população em causa é logicamente homogénea.	A sua eficiência é bastante influenciada pelo algoritmo de seleção do ponto inicial; Pode não ser possível listar todos os elementos da população.
Stratified	Possível interferir em cada stratum e efetuar comparações entre stratas; Obtenção de amostras mais representativas (representação da população transmitida ao longo do processamento)	A identificação das variáveis e formação dos subconjuntos(stratas) pode tornar-se numa tarefa difícil e complexa.
Cluster	Possível efetuar a recolha de amostras num só passo, pelo facto de existir possibilidades em se obter todas as características num só cluster; Não há necessidade de se definir frames de sampling para todos os elementos/clusters selecionados.	Surgimento de erros de sampling, caso não se verifique a existência de clusters homogéneos entre si (menor o nº de clusters, maior o erro); Maior complexidade no processamento e análise dos dados.

Tabela 3.2: Resumo de algumas diferenças entre os métodos de Sampling

3.3 Vantagens e Desvantagens

Com o recurso aos métodos de *sampling*, a representação dos dados através de um painel de *dashboards* torna-se mais eficiente, uma vez que ocorre uma seleção de dados por um conjunto de atributos pré-estabelecidos, resultando na formação de conjuntos de dados mais simples e com um tamanho mais baixo, sem deixar de representar de modo geral, a população em estudo. Segundo W. Edwards Deming, um dos contribuidores no desenvolvimento dos métodos de *sampling*, "*Sampling* não se trata de uma questão de uma situação secundária referente a uma cobertura parcial ou total. *Sampling* corresponde a uma ciência, arte de controlar e medir a relevância de informação útil recolhida por meios estatísticos através da teoria da probabilidade." (Deming, 1950, 2). O facto de se tratarem de processos de seleção probabilística, estes métodos permitem uma recolha de dados mais rápida e, consequentemente, as estimativas são publicadas num espaço de tempo mais curto. Além disso, providenciam informação de confiança sobre a população em estudo com um menor custo em relação a outras abordagens, como é o caso dos "censos" (Lohr).

Qualquer estudo que envolva uma fase de *sampling*, requer um maior cuidado na pré-fase do processamento ao se efetuar medições de variáveis e à sua associação entre o método e o conjunto de dados a ser analisado. Caso contrário, quando não é feita uma seleção de elementos adequada ao conjunto de dados para a criação de amostras aguardadas, podem-se verificar erros do tipo *sistemático (non-sampling)* aquando de uma má estimativa dos parâmetros correspondentes a uma dada população. Deste modo, antes de se proceder à implementação destes métodos, torna-se essencial adquirir conhecimento sobre a população em estudo, definir com rigor os objetivos pretendidos e averiguar possíveis situações que podem ser propicias à ocorrência de erros na tentativa de prevenção da maioria.

As maiores causas de erros num estudo derivam de falhas na cobertura, resposta e organização da coleção de dados. Os erros podem surgir de diversas fontes, podendo ser definidos em duas categorias: erro **sistemático** (*non-sampling*) e erro **não sistemático** (*sampling*). Os erros do tipo sistemático, também conhecidos como os erros não derivados dos métodos de *sampling,*

logo considerados como *non-sampling*, surgem como resultado de decisões que criem tendências na formação de amostras e podem provocar desvios nos valores pretendidos para análise, tais como:

- **Erros na fase de seleção**, que ocorrem a partir da existência de um desequilíbrio, do ponto de vista probabilístico, no qual um elemento tem maior probabilidade de ser selecionado em relação aos restantes elementos da população. Uma forma de se evitar este tipo de erro é usar amostras com tamanhos maiores, para que todos os elementos tenham a mesma probabilidade de serem escolhidos.
- **Erros por ausência de resposta**, que ocorrem quando a informação recolhida não se encontra correta ou está simplesmente incompleta, devido à falta de cooperação das fontes. Perante um caso deste tipo, não existe muito controlo sobre as amostras construídas. A única solução é melhorar o processo de recolha dos dados.
- **Erros na *frame* da amostra**, que surgem quando um determinado subconjunto da população, originado de forma errada, é utilizado na formação da amostra.
- **Erros do tipo não sistemático**, que emergem no decorrer dos métodos, sendo provocado por variações no número ou na representatividade das amostras produzidas. Este tipo de erro pode ser controlado pelo aumento do tamanho das amostras, com o aumento do cuidado do design de amostra e através de um maior número de conjuntos de dados que respondam à necessidade da recolha de elementos para a formação da amostra pretendida (qualtrics).

Por outro lado, em casos como aquele que é abordado nesta dissertação, onde o tratamento de um volume de dados na ordem dos *PetaBytes,* é considerado a base de um dos problemas que consome nos dias de hoje imenso tempo e torna-se bastante dispendioso. Como forma de minimizar este problema, recorre-se a métodos de *sampling* que garantem, como vimos, imensas vantagens. De referir, por exemplo: a redução dos custos e uma maior precisão, ou a otimização no tempo esperado na face de análise dos dados, atuando sobre o mercado de forma mais rápida.

Capítulo 4

O Caso Prático

Concluída a fase de investigação, inicia-se agora a abordagem à implementação de um caso prático. Esta é, pois, uma oportunidade para se aplicar todos os conceitos e ideias que foram especificadas ao longo desta dissertação. O foco do caso de estudo selecionado centra-se na elaboração de uma solução capaz de responder às necessidades solicitadas pela plataforma de suporte a *feeds* do mercado de apostas da Betfair, desenvolvida por uma das equipas da Blip e na implementação de um sistema capaz de ilustrar em tempo real, através de um conjunto de *dashboards*, conjuntos de dados na ordem dos *PetaBytes,* recolhidos por fontes de dados ativas 24h a gerar a cada milissegundo. Ambos os motivos conduzem ao objetivo principal desta dissertação, ou seja, à otimização de análise dos dados a partir da implementação de um painel de *dashboards* interativos, que possibilite a visualização da informação em tempo real, à medida que novos dados vão sendo recolhidos, processados e atualizados.

No decorrer deste capitulo verificar-se-á toda a linha de pensamento efetuada na elaboração do sistema. Assim, iniciamos com a organização de objetivos, estruturação e modelação do problema e a aquisição de conhecimento de inúmeros conceitos de elementos chave deste trabalho, ou seja a fase da pré-análise do problema - pré-processamento. De seguida são mencionadas de forma sucinta e comparativa, as ferramentas abordadas, quer na fase de armazenamento como de processamento. Infelizmente algumas das ferramentas acabaram por não serem implementadas na solução desenvolvida por se encontrarem fora do âmbito definido para esta dissertação, embora sejam mencionadas devido ao seu valor neste tipo de soluções. Por exemplo, a biblioteca *MLearning* que suporta técnicas de ML (*Machine Learning)* para o enriquecimento de dados armazenados sobre *Spark*[1], uma ferramenta bastante utilizada na fase de processamento de dados que será mencionada mais tarde neste capítulo. Por fim, é

[1] http://spark.apache.org/docs/latest/

mencionado na seção de visualização e análise dos resultados deste capitulo, toda análise elaborada com base nos resultados que foram obtidos pela solução desenvolvida, e na secção aplicabilidade industrial é feita uma abordagem de modo semelhante ao caso desenvolvido, mas numa via mais industrial sobre uma possível solução para a plataforma de feeds desenvolvida pela empresa Blip.

4.1 Pré-Processamento

Após o planeamento da solução a ser implementada na plataforma, foi abordada a hipótese de se recorrer a outro caso com um nível de complexidade inferior, por forma a facilitar todo o procedimento e ser possível alcançar os objetivos de forma mais rápida sendo de seguida exequível aplicar o "esboço" desenhado na plataforma de *feeds* já mencionada. O caso definido foca-se na utilização da *stream* de dados de uma das redes sociais mais utilizadas no momento, o *Twitter*. A ideia passa por captar todos os *tweets* que vão sendo publicados por utilizadores e disponibilizados numa *stream*, com uma margem de captura de duas semanas a partir do momento em que a ligação à *stream* é estabelecida, e aplicar sobre estes, métodos capazes de os processar em segundos ou milissegundos sem provocar um aumento na latência, tornando desta forma possível uma otimização na fase de análise e no tempo de ação de resposta perante alterações inesperadas. Para se iniciar o processo de desenvolvimento desses métodos é necessário efetuar um mapeamento sobre os dados que irão ser processados, averiguar os campos mais relevantes para o negócio e que permitem elaborar análises objetivas e especificas, de forma a ser possível a criação de gráficos simples e percetíveis para uma leitura rápida.

4.1.1 Modelação dos dados

Numa primeira abordagem, verifica-se que um *tweet* encontra-se dividido em 2 partes, nomeadamente:

- Campos básicos de um *tweet* elaborados pelo autor, como a mensagem, data em que foi criado ("*created_at*") e id;
- Campos referentes à conta do autor, destacando-se o id, o nome utilizado por este na rede ("*screen_name*"), a sua localização ("l*ocation*"), quantidade de *followers* ("*followers_count*") e amigos ("*friends_count*") e a data em que a sua conta foi criada ("*created_at*").

Caso se esteja perante um *tweet* de resposta a um outro *tweet*, surge uma outra parte que menciona todos os campos do *tweet* original[2]. Todos esses campos destacados, são importantes quer para a fase de armazenamento na organização dos dados, como para a fase de processamento e no fim, campos como id e data, tornam-se fatores essenciais, caso se pretenda efetuar testes de validação.

4.1.2 Arquitetura

Antes da solução ser implementada, é muito importante agregar todas as ideias que foram surgindo no decorrer da fase de investigação, de uma forma organizada, num desenho de arquitetura dos vários componentes que a compõem. Na figura 4.1 apresenta-se o esquema geral do caso abordado.

[2] Em anexo, apresenta-se também um exemplo de um *tweet* recolhido pela *stream – (Anexo 1)*

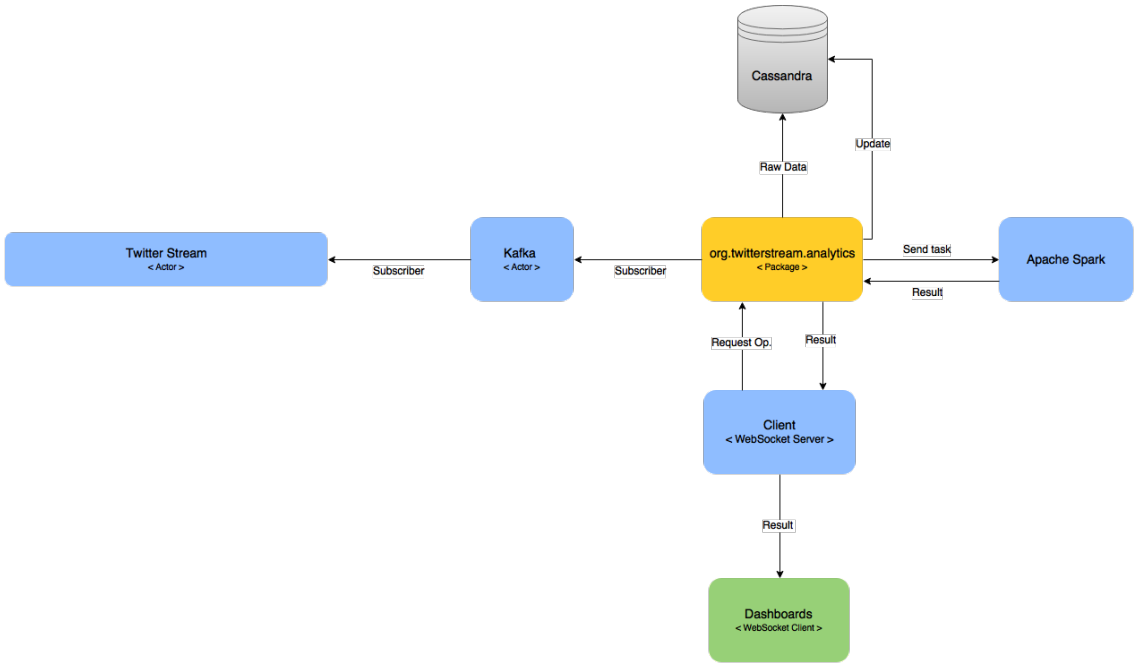


Figura 4.1: Arquitetura de processamento de tweets em tempo real

O fluxo dos dados desenvolve-se em 3 fases distintas, nomeadamente a fase inicial, correspondente a fase de suporte à recolha dos dados, a fase composta por um conjunto de componentes responsáveis pelo armazenamento e processamento dos dados obtidos na fase anterior e, por fim, a fase de análise, que permite através de um serviço web fazer a ilustração de um conjunto de *dashboards* que transponham os dados obtidos ao longo de todo este processo de modo a ser possível efetuar-se uma fácil e rápida leitura sobre o estado do mercado abrangido.

Na fase inicial, podemos identificar a presença de um objeto, *TwitterStream*, que se comporta segundo o padrão *Publisher/Subscriber*. A função deste objeto é fazer a recolha de todos os dados obtidos da *stream* e a sua consequente disponibilização a futuros *subscribers* (utilizadores que submetem tarefas ao seu serviço). Um desses *subscribers* trata-se de um objeto que é responsável por reencaminhar todos os *tweets* recebidos pelo Kafka, um sistema de mensagens distribuído. Cada *queue* do Kafka preserva a ordem de escrita das mensagens e

está particionada pelos vários nodos do cluster, bem como as suas partições que estão igualmente replicadas pelos diferentes nodos. Este modelo permite uma maior escalabilidade da arquitetura do sistema, bem como uma maior tolerância a falhas. A componente *Analytics* é responsável por consumir todas as mensagens publicadas no Kafka e por toda a fase de armazenamento e processamento dos dados. Ao longo dessa fase é importante garantir que o fluxo de dados não crie problemas de latência, isto é, garantir que, à medida que os dados são escritos na base de dados e posteriormente atualizados, não se verifique um atraso de resposta de um pedido efetuado pelo cliente por inexistência dos dados. Caso contrário, embora a fase de processamento seja efetuada de modo bastante rápido, acrescentar-se-ia um intervalo de tempo de espera, por forma a permitir a obtenção da resposta pretendida pelo cliente, impedindo, desta forma, um dos objetivos desta dissertação no que diz respeito à representação dos dados em tempo real.

No decorrer da fase de armazenamento, todos os dados que sejam recolhidos no Kafka, ainda em formato *raw,* ou obtidos como resultado de uma tarefa processada pelo *Spark*, são guardados e atualizados de forma ordenada numa base de dados *Cassandra*. Esses dados são depois recolhidos à medida que um cliente solicite um pedido. O objeto que se comporta como cliente encontra-se implementado na ferramenta *Play*, recorrendo a servidores *WebSockets* de forma a permitir estabelecer uma simples comunicação com um serviço *http*. Desta forma, quando um cliente efetua um pedido, este é transformado numa tarefa do *Spark*, sendo executada em segundos. O resultado deste pedido é guardado em paralelo na base de dados e encaminhado para o cliente que irá disponibilizar os dados, para passando estes por uma fase de renderização e ilustrados num painel sobre a forma de gráficos.

Para que o fluxo apresentado ocorra sem falhas, é necessário que o conjunto de ferramentas implementadas nesta estrutura, seja tolerante a falhas para que, deste modo, se possa minimizar os riscos. Contudo, mesmo assim, continua a ser essencial possuir uma ferramenta que seja capaz de gerir todo o sistema. Para este caso, escolhemos o Zookeeper, um serviço de gestão distribuído, que possibilita aos sistemas distribuídos agirem como sistemas funcionais, facilitando o uso e a integração de qualquer sistema que necessite deste tipo de funcionalidade.

Além disso, esta ferramenta permite averiguar grandes volumes de transações feitas por clientes, através da previsão e resolução de potenciais problemas antes da sua ocorrência, eliminando desta forma grande número de falhas e riscos. Desta forma, é possível obter-se um maior controlo dos custos e dos processos de *streamline* para agilidade operacional.

Para a concretização desta solução, as ferramentas Spark, Cassandra e Play foram fundamentais. O Spark é uma ferramenta de processamento bastante utilizada na atualidade em aplicações de processamento de dados e disponibiliza uma grande quantidade de funcionalidades. Além disso, é fácil de aprender e de utilizar. Por se tratar de um serviço confiável, o *Cassandra* como uma base de dados distribuída, aborda um modelo *Key/Value e* DHT (distributed hash table), e por fim, a Framework de aplicações Web, *Play*, que permite escalabilidade no desenvolvimento de aplicações reativas e interativas de forma simples. Mais à frente, estas ferramentas serão abordadas com um pouco mais de detalhe.

Num sistema distribuído, por vezes, lidar com o fator concorrência nem sempre se torna numa tarefa simples, principalmente quando existe uma maior necessidade em se garantir toda a manutenção, processamento e sincronização sem qualquer falha, perante um número significativamente elevado de *cores* de um processador. Uma via, face a essas necessidades, pode ser alcançada não por um sistema de *threads* mas por um sistema de atores, como por exemplo o *AKKA*. Este sistema baseia-se em transições assíncronas de mensagens e de estados imutáveis, o que permite fornecer a um sistema um nível de abstração mais elevado em relação ao sistema de *threads*. Desta forma, os seus utilizadores não precisam de se focar em detalhes de baixo nível, mas sim tratar da escrita lógica do negócio. Por fim, referir algo não menos relevante. Este sistema segue o modelo *Erlang*, um modelo de tolerância a falhas, visível através da forma como os atores se organizam num sistema com uma hierarquia de supervisores, em que cada componente é monitorizado por outro e reiniciado no caso de falha (Bonér, 2011).

4.2 Armazenamento

Atualmente, o uso da informação assume um papel relevante no progresso e evolução das empresas. Cada vez mais, estas acolhem nos seus sistemas um volume de dados cada vez maior, com taxas de crescimento impressionantes em alguns casos. Assim, é vulgar detetar situações nas quais a capacidade de resposta no armazenamento de dados de grandes dimensões torna-se bastante ineficiente. Os sistemas relacionais enfrentam hoje muitas destas situações. Este tipo de base de dados foca-se essencialmente no estabelecimento de relacionamentos entre inúmeras tabelas através de um conjunto de atributos definidos com base na informação obtida pelos dados recolhidos e nos objetivos pretendidos para a posterior análise. Contudo, em soluções que exijam uma grande escalabilidade, debatemo-nos frequentemente com problemas de eficiência em determinadas operações de manutenção chave. Por exemplo, o facto de não se fornecer suporte a dados em formato *raw*, a presença de limitações em questões de escalabilidade das tabelas, ou a dificuldade em implementar simples *queries* através do SQL, são apenas alguns desses exemplos (Zaki, 2014). As dificuldades reveladas pelos sistemas relacionais na capacidade de armazenar grandes quantidades de dados, pode ser gradualmente resolvida através da abordagem de soluções mais dispendiosas, como a implementação de novos clusters. Porém, este tipo de abordagem não garante que, após um súbito aumento dos dados, exista uma resposta rápida e eficiente, podendo afetar o desempenho e contribuir para um decréscimo dos lucros (existência de uma proporcionalidade direta entre os dados e os custos) de uma empresa. Hoje, este tipo de problemas já pode ser contornado. Novos tipos de sistemas de base de dados emergiram, sistemas tipicamente não relacionais, vindo dar atenuar este conjunto de limitações em soluções com uma infraestrutura de grande escalabilidade, permitindo deste modo gerir e analisar os dados sem a preocupação de custos de hardware. Sistemas de gestão de bases de dados como o *Redis*, o *Cassandra* ou *HBase* são apenas alguns desses sistemas que são capazes de oferecer condições para este tipo de soluções. Vejamos algumas das características agregadas por este tipo de sistemas:

- O **HBase** consiste numa base de dados distribuída, implementada sobre o sistema *Hadoop* e *Zookeeper* e modelada pelo conceito de *BigTable* introduzido pela Google, no qual vários conjuntos de dados são repartidos por múltiplas gerações usando apenas a memória para assegurar a última geração criada. A sua arquitetura é composta por tabelas de hash orientadas por coluna e ordenadas por linha, o que permite um acesso aleatório e em tempo real por operações de escrita e leitura. Alem disso, recorre também ao uso do sistema de ficheiros da *Hadoop*, *HDFS*, para garantir tolerância a falhas e a operação de replicação, garantindo escalabilidade linear e modelar e sendo estritamente consistente no acesso aos dados. *HDFS* corresponde ao sistema primário de armazenamento distribuído do *Hadoop*, no qual as várias réplicas geradas de blocos de dados são distribuídas pelos diversos nodos de *clusters*, o que garante tolerância a falhas e uma maior rapidez no processamento. Este não se encontra otimizado no acesso por leituras aleatórias e verifica-se a existência de latência na rede entre o servidor de base de dados e o servidor do ficheiro, que corresponde ao nodo de dados do *Hadoop*. Em suma, trata-se de uma ferramenta bastante utilizada sempre que existe uma necessidade de se efetuar operações de escrita em aplicações mais pesadas e de providenciar acessos aleatórios dos dados de forma rápida (Doble, 2014).
- O **Redis** trata-se de uma base de dados localizada maioritariamente em memória, com persistência de dados no modelo *Key/Value*. De acordo com os casos onde é aplicada, parte dos dados pode persistir temporariamente em disco, com operações de escrita efetuadas de forma assíncrona. Caso contrário, por omissão, os dados são mantidos em memória. Uma das fortes características do *Redis*, é o suporte à tolerância a falhas através da criação de uma árvore de replicação por master-slave, isto é, em cada nodo master os dados podem ser replicados por inúmeros nodos slave, que por sua vez, cada nodo slave pode se comportar como um nodo master, originando um fluxo de replicação ao longo desta estrutura. Contudo, este tipo de replicação é "non-blocking" quer em master como slave, ou seja, o master pode em paralelo, continuar a lidar com *queries* enquanto um ou mais slaves encontram-se a comunicar de forma síncrona, e o slave pode responder às *queries* com a ultima versão dos dados durante essa sincronização. Através deste modelo de replicação, múltiplos nodos slaves podem responder a *queries*

recorrendo somente a operações de leitura, resultando numa arquitetura bastante escalável (Rabl, Sadoghi, & Jacobsen, 2012) . Em casos de rápida alteração dos dados e previsibilidade do tamanho das tabelas, este tipo de base de dados é bastante utilizado, concentrando a maioria dos dados em memória e deste modo as operações de escrita e leitura tornam-se bastante rápidas e eficientes.

- O **Apache Cassandra** é um produto da segunda geração de base de dados distribuídas que utiliza o modelo *Key/Value*, aplicando o modelo *DHT* (Distributed hash tables) às partições dos seus dados para simplificar operações de escrita através de uma função hash, nomeadamente SHA-1 (Shankar). O Cassandra agrega um esquema de replicação múltipla, incluindo o armazenamento de réplicas em servidores vizinhos, permitindo ao utilizador escolher o nível de consistência que melhor se adequa ao caso em prática. Desta forma, contribui para uma maior escalabilidade caso surja uma inclinação para a troca de consistência de dados. Quando é executada uma operação de escrita, esta é automaticamente registada num log num dos nodos de um quorum que faça suporte às réplicas, sendo os dados replicados ao longo dos nodos localizados em memória. Ou seja, caso se verifique uma falha do sistema, existe uma pequena percentagem de perda dos dados. Mas, por outro lado, garante um bom nível de segurança dos dados devido à entropia preservada por uma rede de valores de hash que permite uma verificação mais eficiente e segura dos dados. A sua arquitetura é uma mistura do modelo BigTable da Google com o Dynamo da empresa Amazon (cada nodo do *cluster* possui a mesma função, assegurando desta forma a inexistência de pontos de falhas, o que não acontece em HBase) e o layout é orientado a colunas, ou seja, todos os dados são guardados por coluna em disco. A grande vantagem é que se torna bastante eficiente no processo de armazenamento de dados esparsos e permite em disco uma distribuição em bloco dos dados armazenados. Como desvantagem, nem todas as colunas são acedidas e acaba por ocupar espaço de disco desnecessário, que em soluções deste tipo, acaba por afetar um pouco o desempenho (Gokavarapu). Possui duas fases no processo de armazenamento, nomeadamente a fase de repartição dos dados por chave em diferentes servidores e a fase de replicação dos dados em diferentes servidores. Verifica-se também a existência de uma grande margem em questões de elasticidade nas operações de

leitura e escrita, porque ambas as operações aumentam de forma linear à medida que novas máquinas são adicionadas, sem quebra de tempo ou existência de interrupções nas aplicações abrangidas (Rabl, Sadoghi, & Jacobsen, 2012). Em resumo, é bastante utilizada em situações de transações de processamento em tempo real e interatividade de informação, resultando em casos onde a necessidade em se efetuar operações de escrita é maior, permitindo o armazenamento de uma quantidade de dados demasiado grande para um servidor, mas sem abdicar de uma interface simples.

Existe de facto uma diversidade bastante grande de soluções a nível de armazenamento, como tal, torna-se essencial averiguar com base nos recursos disponíveis e nos objetivos, a solução capaz de responder às necessidades solicitadas. Desta forma, segue-se a seguinte tabela, que salienta os aspetos mais relevantes das bases de dados já mencionadas:

	Cassandra	**Redis**	**HBase**
Teorema CAP	AP (mas é possivel balancear o grau de todas as propriedades)	C (Garantindo a propriedade "A" através da biblioteca *Redis Sentinel* e propriedade "P" por *Redis Cluster*)	CP (alguns dados podem não estar disponiveis, mas o resto mantem-se consistente)
Localização dos Dados	Disco mas com registo de logs em Memória	Memória	Disco e Memória
Arquitetura	BigTable, Google; Dynamo, Amazon; Modelo Key/Value	Modelo Key/Value	BigTable, Google
Desempenho Operacional	Valores mais elevados em operações de escrita	Valores semelhantes nas operações de escrita e leitura	Valores mais elevados em operações de escrita
Melhor Caso	Situações onde a necessidade de executar operações de escrita é mais elevada em relação às operações de leitura	Situações de rápida alteração dos dados e previsibilidade do tamanho das tabelas	Situações de acesso(leitura/escrita) aos dados de forma aleatória e em tempo-real

Tabela 4.1: Pontos relevantes de uma base de dados para fins comparativos

Muitas das comparações feitas entre sistemas distribuídos e sistemas de base de dados recorrem ao teorema CAP (*Consistency, Availability and tolerance to network Partitions*) que surge após *Eric Brewer* ter introduzido a ideia de existência de fundamentos de comparação entre os termos: consistência, disponibilidade e tolerância a partições. O teorema baseia-se numa medida de valores a atribuir, mas através de propriedades necessárias para um sistema distribuído, não sendo atribuídas mais do que duas propriedades. Em suma, "consistência" refere-se à persistência dos dados após a execução de uma dada operação, "disponibilidade" significa que se algum nodo se perder no *cluster*, o sistema distribuído continua operacional e "tolerância a partições" significa que se os nodos do *cluster* se dividirem em dois grupos que já não conseguem estabelecer comunicação devido a uma falha de rede, o sistema mantem-se mais uma vez operacional (Gilbert & Lynch).

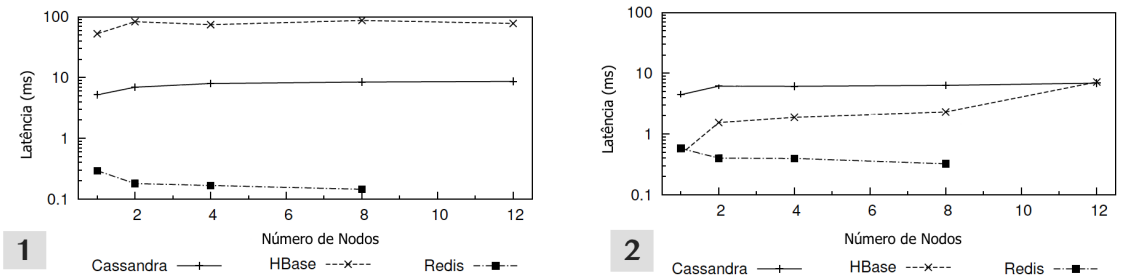


Figura 4.2: Desempenho de base de dados medido por valores de latência, numa escala logarítmica

Recorrendo a um conjunto de resultados obtidos de duas experiências elaboradas pela Universidade de Toronto (Rabl, Sadoghi, & Jacobsen, 2012) , é possível analisar em situações extremas, isto é, uso de 90% quer em operações de leitura (gráfico 1) como de escrita (gráfico 2), como as bases de dados se comportam em questões de tempo de resposta fase à solicitação de um pedido operacional. No primeiro gráfico ilustra o resultado de uma experiência onde foram efetuadas operações de leitura intensivas. Numa primeira análise, verifica-se um padrão semelhante entre Cassandra e HBase, com uma subida inicial dos valores de latência, sendo depois constantes à medida que o número de nodos aumenta. Visto que a maioria dos dados se encontra em memória, o acesso a estes é mais rápido, logo o *Redis* é sem dúvida a

base de dados com melhores resultados a nível de latência, e HBase a solução com valores mais altos de latência, visto que sempre que é feita uma operação de leitura, efetua um acesso ao disco e recorre à memória caso a informação pretendida se encontre em falta. Já no caso do segundo gráfico, o Cassandra mantem-se constante, mas verifica-se uma subida gradual dos valores em HBase, alcançando a mesma latência de Cassandra num cluster com 12 nodos.

Com base nos objetivos delineados para o caso prático, a base de dados que melhor se enquadra é Cassandra, uma vez que existe uma maior necessidade em se recorrer a operações de escrita em relação às operações de leitura e de manter grande quantidade de dados em histórico, garantindo assim escalabilidade, tolerância a falhas e consistência dos dados. HBase tornou-se numa forte opção, mas acabou por ser rejeitada devido às operações de leitura de atingirem valores de latência bem mais elevados, enquanto que em Cassandra, não só é mais simples de se utilizar e de estabelecer a lógica de *queries* como consegue se manter constante à medida que se escala.

4.3 Processamento

Nesta fase aplicámos um conjunto de tarefas específicas a vários conjuntos de dados formados com base nos objetivos delineados para cada uma dessas tarefas e, sucessivamente, sujeitámos esses dados ao processamento a uma das ferramentas mais utilizadas no mercado dentro da área de soluções em grande escala, nomeadamente, o *Apache Spark*. Um dos motivos que conduziu ao uso desta ferramenta, assentou na facilidade que esta ferramenta disponibiliza para conjugar vários *workloads* como aplicações de *batch*, streaming e interatividade de *queries*, numa só máquina. Desta forma, é possível lidar com vários tipos de processamento de dados sem grandes custos, reduzindo-se o tempo de manutenção, uma vez que só se está a utilizar uma única ferramenta. Esta ferramenta pode ser assim utilizada como uma forma de melhorar o desempenho do MapReduce (processamento em paralelo, implementado pela *Hadoop*), conseguindo alcançar uma maior velocidade de processamento. A sua utilização está muito facilitada devido à enorme variedade de API's que esta ferramenta

suporta. Além disso, consegue correr sobre diversos gestores de clusters, tais como o Yarn (desenvolvido para Hadoop), Mesos (gere um conjunto de nodos do cluster e *frameworks,* incluindo Hadoop*,* que suportam um agregado de API's com recursos para operações de manutenção) e o modo Standalone que se trata do gestor do Spark por *default*.

Embora seja bastante abordado nos últimos tempos o Hadoop tem revelado algumas falhas, nomeadamente em questões de latência na execução de operações por MapReduce. Uma das vantagens do Spark, é a de não necessitar que o sistema recorra ao Hadoop, alcançando desta forma resultados de forma mais rápida e escondendo, ao mesmo tempo, a complexidade do sistema distribuído, da comunicação de redes e tolerância a falhas. Todo o trabalho que é concebido pelo Spark expressa-se sobe a forma de criação, transformação ou invocação de operações de RDDs, por forma a obter-se o resultado pretendido. Basicamente, uma *Resilient distributed dataset (RDD)*, consiste numa unidade de abstração distribuída em memória das operações de execução de tarefas, isto é, uma coleção imutável e distribuída de objetos, na qual o processamento é efetuado em memória por inúmeros *clusters*, o que garante uma maior tolerância a faltas. Cada um é dividido em inúmeras partições que podem ser processadas em diferentes nodos de um cluster (*workers*), sendo posteriormente processados em paralelo. Caso um *worker* falhe, os dados perdidos durante o procedimento são novamente processados através da sua replicação e caso a máquina falhe, os dados são completamente perdidos, a não ser que estejam a ser guardados numa base de dados como o Cassandra.

Na implementação desenvolvida o gestor de clusters utilizado é o que se encontra por *default* no Spark, o *Standalone*, um gestor simples que, para além da configuração de um cluster, permite gerir os diversos nodos associados e averiguar os vários parâmetros envolvidos, como tempos e tarefas executadas por cada nodo através de uma interface Web. Desta forma, é possível agregar um conjunto de dados relevantes para análises que se possam realizar (Zaharia, Das, Li, Hunter, Shenker, & Stoica, 2013). Em questões de algoritmos de processamento dos dados, os métodos de sampling abordados foram os seguintes:

- **Estratificação não proporcional**. Este método é aplicado no algoritmo para obtenção de um conjunto das K hashtags mais utilizadas nas últimas duas semanas do *twitter*. Aqui, parte-se de um agregado de *tweet*s que sofrem uma filtragem por hashtags e que são posteriormente, armazenados numa das tabelas da base de dados juntamente com um *count* (número de vezes que a mesma hashtag surge num *tweet*), que dá origem a um strata. Este strata é depois ordenado por ordem decrescente para que depois seja feita a seleção de K elementos. Neste método verifica-se a inexistência da proporcionalidade probabilística, pelo facto de existirem elementos do strata que possuem uma maior probabilidade em serem selecionados em relação aos restantes, nomeadamente todos os elementos que possuem um valor de *count* superior.
- **Clustering**. Este método é abordado num algoritmo desenvolvido para a obtenção de um conjunto das K localizações com maior número de *tweets* e as correspondentes K hashtags mais utilizadas para uma determinada localização. Ou seja, com base na população inicial, que agrega todo um conjunto de *tweets* das duas últimas semanas do *twiiter*, gerou-se um conjunto de clusters. Cada cluster passou a corresponder a um conjunto de *tweets* publicados de uma determinada região, sendo esta a localização especificada pelo utilizador no acesso à respetiva conta no *twitter*. Depois, estes *clusters* são novamente repartidos, dando origem a um novo conjunto de *clusters*. Cada um destes novos *clusters* irá representar um conjunto de K hashtags reconhecidas por um fator de utilização maior na respetiva região.

Muitos outros algoritmos, que não foram abordados nesta dissertação, podem contudo ser aplicados neste mesmo contexto, tendo apenas como fator de influencia, o planeamento estabelecido para os vários objetivos a serem cumpridos em termos do projeto e do tipo de informação que se pretenda adquirir para uma futura análise.

4.4 Visualização e Análise de resultados

Com base na análise do desempenho dos algoritmos implementados nas fases já referidas, foi possível ilustrar o seguinte conjunto de resultados obtidos (Figura 4.3).

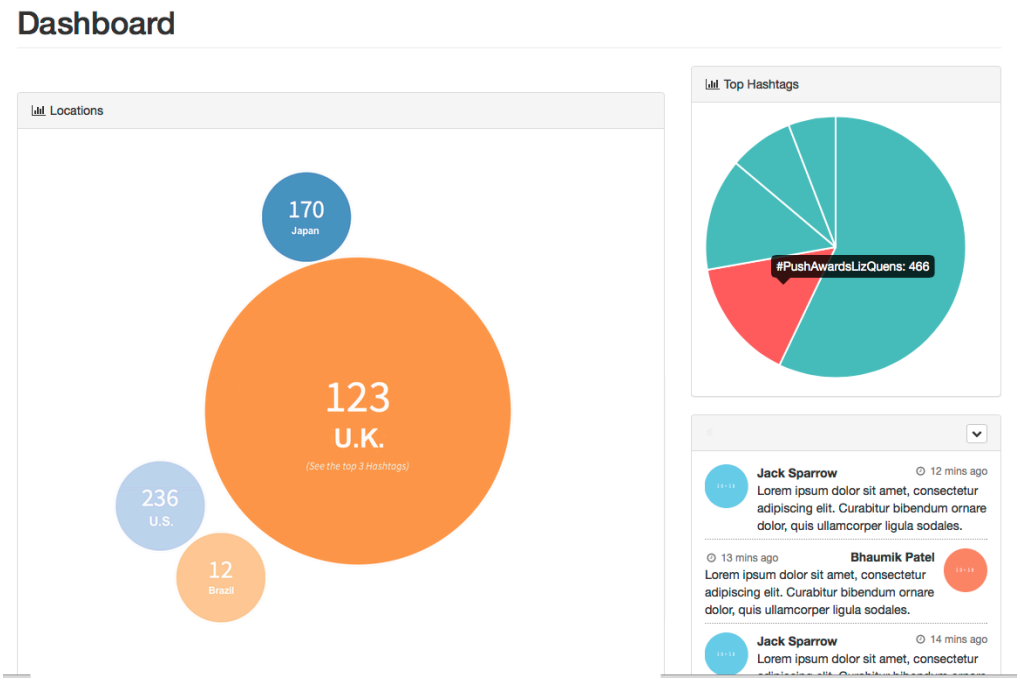
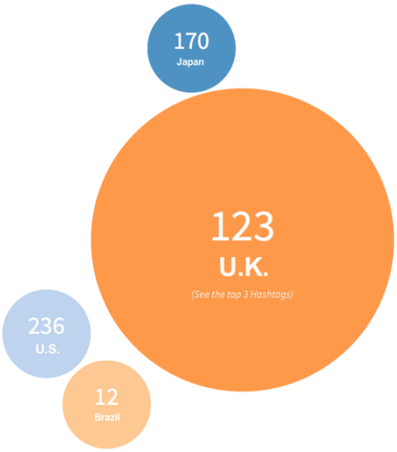


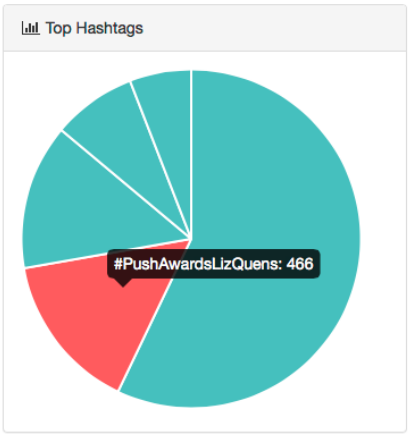
Figura 4.3: Painel de dashboards composto por conjuntos de dados recolhidos em tempo real da stream

Neste painel podemos encontrar dois gráficos, com objetivos diferentes, sendo um deles referente ao top 5 de hashtags mais abordadas nos últimos dias, e o outro correspondendo às regiões com mais *tweets* nos últimos dias e às 3 hashtags mais utilizadas, respetivamente. Para fazer a implementação destes gráficos, recorreu-se às bibliotecas de d3.js[3] e charts.js[4].

[3] http://d3js.org

[4] https://github.com/nnnick/Chart.js





Ao se recorrer à utilização de *WebSockets* no estabelecimento de uma ligação com um serviço *Http*, todos os gráficos do painel referido são atualizados à medida que novos conjuntos de dados são processados e renderizados, não se deixando, porém, de se poder atribuir um intervalo de tempo no caso de se estar de acordo com os objetivos pré-estabelecidos. A implementação de *dashboards* interativos revelou-nos uma desvantagem clara da sua implementação, que, basicamente, requere um grande espaço de memória para suportar todo o processamento efetuado em milissegundos.

Na Figura 4.4 é ilustrado parte dos logs recolhidos durante a execução de uma das tarefas a serem executadas pela ferramenta Spark.

```
-------------------------------------------
Time: 1446315966400 ms
-------------------------------------------
#webhosting
#domains
#cheaphosting
```

Figura 4.4: Exemplo de um conjunto de dados processados pelo Spark

4.5 Aplicabilidade Industrial

O mercado de apostas é extremamente competitivo, não só entre clientes, mas especialmente entre as empresas que desenvolvem soluções de software para este tipo de mercado. Em Portugal, a Blip é a única empresa que desenvolve software em grande escala neste domínio, uma vez que se trata de um dos diversos polos da Betfair, uma das grandes empresas de apostas a nível internacional. Nos últimos anos o nível de exigência sobre este tipo de empresas tem sofrido um crescimento bastante elevado, pelo facto de se tratar de um mercado cada vez mais competitivo e sobre constantes alterações. Assim, surgiu a necessidade de se alcançar soluções que permitem a estas empresas otimizar a sua análise de resultados e responder com eficácia, e num espaço de tempo bastante reduzido, a todas essas alterações impostas pelo mercado.

Uma das equipas de desenvolvimento da empresa Blip, encontra-se a desenvolver uma plataforma para recolher e processar feeds produzias pela empresa, que, basicamente, consistem num conjunto de dados gerados por outras fontes/equipas de suporte, referentes a todos os acontecimentos ocorridos a cada instante num evento de um determinado mercado.

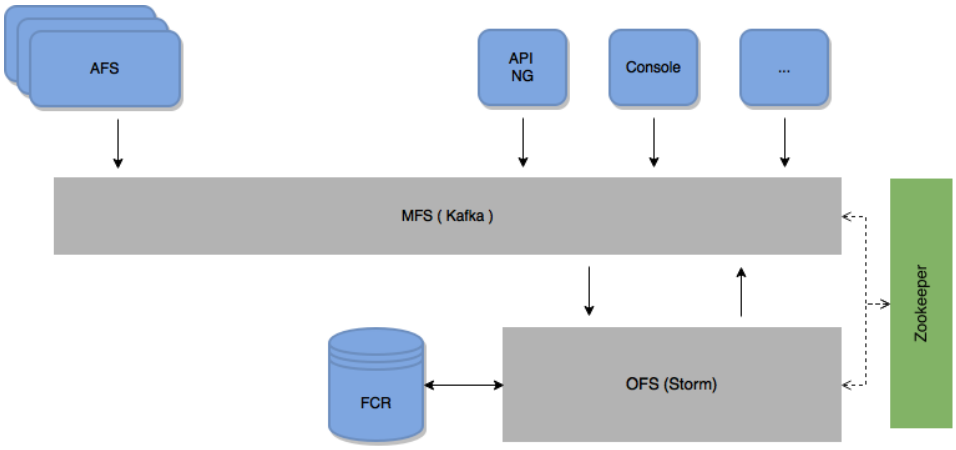


Figura 4.5: Plataforma de feeds desenvolvida pela Blip

Num curto espaço de tempo, várias tecnologias como *Storm*, *Kafka* e *Redis* são utilizadas no momento da criação e gestão de um fluxo de *feeds.* Desta forma, tornou-se possível obter um conjunto de vantagens fundamentais no arranque ao crescimento da empresa, tais como: um acesso mais rápido aos dados em histórico, garantir maior ordem entre os dados relacionados, gerir as transformações de modo dinâmico, ou garantir a escalabilidade dos sistemas e a sua tolerância a falhas, entre outras.

Contudo, verificou-se que tal não era o suficiente. Era possível obter-se, ainda, um maior proveito desta plataforma, alcançando assim novas funcionalidades, que permitem à empresa obter não só um ganho mais elevado, como um maior peso perante outras empresas que se encontrem neste tipo de mercado.

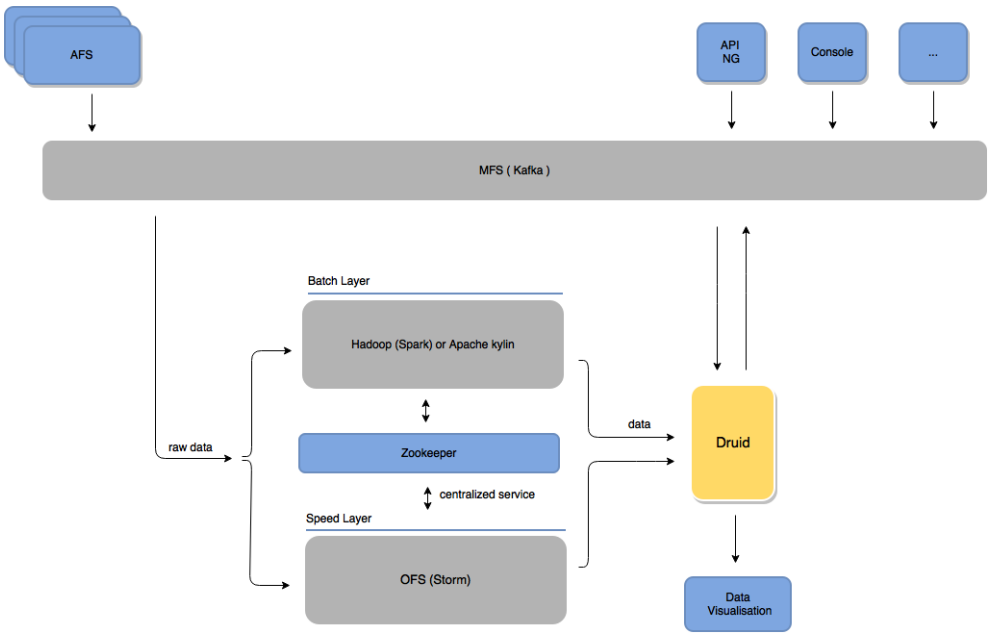


Figura 4.6: Solução criada para a nova plataforma de feeds

Com a introdução de uma nova camada para *batch processing*, obtivemos algumas vantagens, tais como:

- despender de algum tempo no processamento de conjuntos de dados para o enriquecimento da informação obtida por estes;
- armazenar em grande escala tanto os dados que provêm das inúmeras fontes, *raw data*, como os dados pré-processados para histórico;
- implementar uma nova fase de renderização dos dados para formação de um conjunto de gráficos em tempo real para futuras análises.

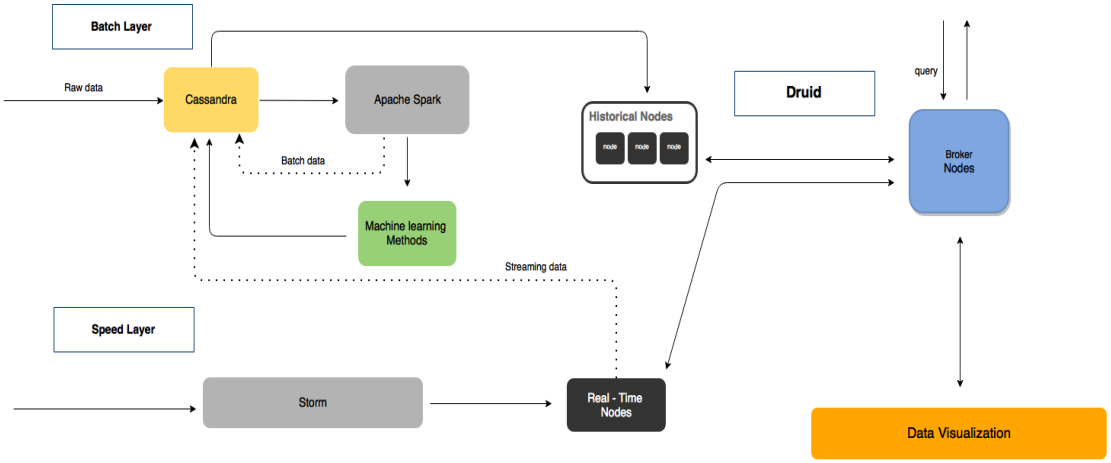


Figura 4.7: Representação detalha da estrutura da plataforma de feeds com a introdução da nova camada de batch processing

De forma mais detalha, o fluxo dos dados na nova arquitetura está organizado de acordo com o que a figura 4.7 apresenta. O fluxo inicia-se com a chegada dos diversos conjuntos de dados, em milissegundos, que são dispersos em duas pipelines: a *batch layer* e a *speed layer*. A *speed layer* inicia-se com a recepção dos dados provenientes do Kafka, *raw data,* no Storm, por forma a serem processados e reencaminhados para uma espécie de cache criada pelo sistema *Druid*, que é usualmente designada por *Real-Time Nodes*. Estes encapsulam toda a funcionalidade,

criando periodicamente segmentos que representem esses dados e disponibilizam de imediato esses segmentos no sistema, de forma a que se possa proceder à execução de queries provenientes dos *Broker Nodes.* Ao mesmo tempo, estes dados são encaminhados para a base de dados Cassandra, que aqui atua como um sistema histórico para futuras pesquisas. Quanto ao Druid, este consiste num sistema distribuído que permite o acesso em tempo real a grandes conjuntos de dados, garantindo assim resposta a possíveis *queries* provenientes de inúmeras fontes, como ocorre neste caso, por parte de equipas de API NG, Console, etc. Na *batch layer*, os dados provenientes do Kafka são diretamente armazenados na base de dados para a criação de histórico de dados não processados, sendo de seguida, eventualmente, encaminhados para o Spark para serem processados. A partir desta fase os dados que resultam desse processo seguem duas vias distintas: são novamente armazenados ou entram noutro género de processamento, desta vez efetuado por algoritmos de ML para o enriquecimento dos dados. Nesta camada, o sistema de resposta a queries é feito através de mais um dos componentes do Druid, designados por *Historical Nodes*, que com base nas queries solicitadas agrega temporariamente conjuntos de dados da base de dados e reenvia-os para o Broker Nodes, que trata de encaminhar a informação ao cliente. Desta forma, o processo de pesquisa torna-se mais rápido, visto que para dois clientes distintos, mas com objetivos idênticos, para um mesmo conjunto de dados, o processo de pesquisa para o segundo cliente concentra-se apenas num dos Historical Nodes. Isto porque a informação já foi agregada nos nodos por causa do primeiro cliente. Assim, é efetuado um retorno praticamente imediato da informação para o segundo cliente. As queries dos clientes podem ser transformadas em pedidos que são depois introduzidos no Kafka, por fontes terceiras ou simplesmente em pedidos provenientes de um servidor WebSocket para recolha de informação que é essencial na geração ou atualização de um determinado conjunto de gráficos.

Através desta solução, é possível alcançar valores de latência mais baixos na fase de processamento de queries. Consequentemente, os gráficos podem ser produzidos de uma forma mais rápida, sendo assim possível progredir com maior conhecimento na fase de análise, reagindo deste modo, com maior eficácia e num curto espaço de tempo, perante alterações do mercado abrangido.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões

Com o surgimento das aplicações *Big Data*, o conceito de *dashboarding*, ganhou um novo estatuto na atualidade, sendo cada vez mais empregado em inúmeras fontes de informação e aplicações. Contudo, hoje ainda se verifica que em muitas empresas que desenvolvem *software* com base na informação, o recurso a painéis de *dashboard* apenas permite a utilização de 50% dos seus potenciais. Ou seja, embora se verifique um maior recurso a este tipo de análise por visualização de conjuntos de dados pré-processados, muitas empresas ainda não tiveram oportunidade de obter um maior aproveitamento desta iniciativa e de todos os aspetos tecnológicos associados. Isto deve-se maioritariamente ao facto de ainda existirem muitas empresas que tentam lidar com problemas de *Big Data* e, ao mesmo tempo, a averiguarem o surgimento de uma nova necessidade para conseguirem progredir ou, no mínimo, manter um rendimento sustentável. Essa necessidade surge devido à presença de valores de latência elevados ao longo de todo o processamento dos dados, quando se requer uma ação que vá de encontro com o ritmo do crescimento do mercado abrangido sem se verificar rupturas no sistema.

O alcance das soluções *Big Data* que suportem um conjunto de *dashboards*, gerados num curto espaço de tempo, como em milissegundos, trás inúmeras vantagens a qualquer empresa em termos de desempenho, quer este seja funcional ou analítico. Nesta dissertação realizou-se um estudo inicial sobre o estado atual de muitas empresas que se debatem com a manipulação de dados na ordem dos *PetaBytes* e com a incapacidade de recolher informação útil provocada por processos de análise ineficientes. Em suma, neste trabalho mostrou-se que a ideia de *Big Data* e de *dashboarding* nascem como resposta a uma maior necessidade de tratar e analisar

grandes volumes de dados e que se complementam em função de um objetivo comum: eficiência na ação de resposta perante as indeterminâveis e constantes alterações do mercado. Surge desta forma, o conceito de IO, como um principio de organização, importante para qualquer sistema e conjunto de métodos que sejam capazes de mover um negócio em tempo real e de adquirir, ao mesmo tempo, um conhecimento sobre velocidade e agilidade em processos de tratamento de dados. Deste modo relevámos um conjunto de tecnologias, vantagens e propriedades garantidas por todo este novo conceito, que os métodos tradicionais de BI usualmente não agregam, como também evidenciámos os vários desafios na sua implementação que ainda hoje um grande número de empresas enfrenta. Inúmeras empresas, como a *Zoomdata*, a *Datameer*, a *Splunk* ou a *Enterprise*, são mencionadas nesta dissertação como exemplos de empresas que surgem nos dias de hoje com grande influência neste domínio, uma vez que apostam no desenvolvimento de software para a área da visualização de dados e que, ainda assim, apresentam nas suas soluções algumas lacunas em questões de *rendering*, latência, segurança, entre outras. A maior parte desses problemas têm como principal causa a incapacidade demonstrada pelos atuais mecanismos e processos de recolha e tratamento dos dados, que só por si provoca um aumento dos valores de latência e a atribuição errada quer de métodos de redução analítica como do tipo de gráficos gerados.

Por fim, após a referência a todo este estudo, necessário para suportar os vários aspetos discutidos nesta dissertação, abordámos o desenvolvimento de um caso prático que foi escolhido especialmente para esta dissertação. O caso selecionado é uma prova clara de que, atualmente, existem inúmeras ferramentas capazes de suportar soluções capazes de manipular e de trabalhar enormes quantidades de dados, não ficando apenas pela fase de armazenamento, mas também alcançando resultados positivos noutras áreas de trabalho, quer seja ao nível da fase de processamento como ao nível da fase de visualização dos dados.

5.2 Trabalho Futuro

Relativamente ao caso prático apresentado, podemos dizer que este foi apenas de um ponto de partida para a implementação de um sistema distribuído de recolha e tratamento de dados, que pode ainda ser otimizado no futuro, através de um aperfeiçoamento de inúmeros aspetos que se considerem relevantes quer para o bom funcionamento do sistema como para o enriquecimento dos dados que consequentemente têm uma grande influência na fase de análise executada pelo utilizador. Como exemplo, tem-se a possível otimização dos valores de latência obtidos ao longo de todas as fases, desde a recolha até à renderização dos dados para a geração de gráficos, cuja a sua diminuição permitirá alcançar um novo conjunto de vantagens ao utilizador, nomeadamente, passar a ter um poder de resposta mais rápido e eficiente mediante a evolução do mercado e as alterações que consequentemente agrega.

De facto, seria interessante, e relevante para este tipo de soluções, efetuar-se a implementação de algumas das ferramentas já mencionadas ao longo desta dissertação, tais como, as bibliotecas de ML para o enriquecimento do conhecimento contido nos dados, o sistema distribuído *Druid* para lidar de forma mais eficiente com um conjunto de *queries*, entre outras. Caso a solução fosse desenvolvida num âmbito industrial, seria ainda mais interessante averiguar a capacidade de resposta por parte desse conjunto de ferramentas mediante o crescimento dos dados e com base num painel de *dashboard*, verificar o tipo de análise mais viável e rentável para a empresa, efetuando assim um ajustamento ao conjunto de gráficos para uma representação mais eficiente. Ainda assim, um dashboard poderia gradualmente sofrer otimizações com a introdução de gráficos cada vez mais dinâmicos e que permitissem uma maior ligação entre eles, ou seja, uma solução que permitisse fortalecer a comunicação entre os inúmeros gráficos existentes dentro de um painel. Desta forma, com a recepção, processamento e visualização do conjunto de dados disponível, em milissegundos (*near real-time*), o utilizador acabaria por conseguir ter, com maior facilidade, uma percepção não só do estado atual da empresa e dos pontos visivelmente mais afetados pelas alterações do mercado, como também prever possíveis pontos que possam gerar futuras vantagens à empresa ou, ao contrário, causar algum tipo de problema.

Bibliografia

Batstone, G. (s.d.). (NGRAIN) Obtido de http://www.ngrain.com/3-reasons-why-visualization-is-the-biggest-v-for-big-data/

Bonér, J. (12 de 5 de 2011). *Why-akka.* Obtido de Typesafe : http://www.typesafe.com/blog/why-akka

Chris, G., Xinle, H., William, B., & Jihoon, K. (2014). *Data Visualization for Big Data.* Parsons Institute for Information Mapping, New York.

CITO Research. (2013). *Operational Intelligence: What it is and Why you need it now.* Paper.
Daniel, J. (2012). Choosing the type of probability sampling. Em J. Daniel, *Sampling Essentials: Practical Guidelines for Making Sampling Choices.*

Doble, W. (2014). *Comparing the Use of Amazon DynamoDB and Apache HBase for NoSQL.*

Gilbert, S., & Lynch, N. A. *Perspectives on the CAP Theorem.* National University of Singapore.

Gokavarapu, H. *Exploring Cassandra and HBase with BigTable Model.* Indiana University Bloomington, Department of Computer Science.

Liu, S., Cui, W., Wu, Y., & Liu, M. (2014-12). A survey on information visualization: recent advances and challenges. *The Visual Computer , 30* (12), 1373-1393 .
Sampling and Nonsampling Errors. Em S. L. Lohr, *Sampling: Design and Analysis* (pp. 15-6).

Nyland, J. (25 de Novembro de 2013). *Streaming Data: What a Real-Time Dash Does and Doesn't Tell You.* Obtido de Digital Marketing Blog: http://blogs.adobe.com/digitalmarketing/analytics/streaming-data-what-a-real-time-dash-does-and-doesnt-tell-you
qualtrics. (s.d.). Obtido de https://www.qualtrics.com/wp-content/uploads/2013/05/Sampling.pdf

Rabl, T., Sadoghi, M., & Jacobsen, H.-A. (2012). Solving Big Data Challenges for Enterprise Application Performance Management. *The 38th International Conference on Very Large Data Bases. 5.* Istanbul: VLDB Endowment.

Russom, P. (2013). *Operational Intelligence: Real-Time Business Analytics from Big Data.* TDWI, The Data Warehousing Institute.

Shankar, V. *Cassandra DHT-based storage system.* Indiana University Bloomington.

Splunk. (s.d.). *Machine Data*. Obtido de Splunk: https://www.splunk.com/en_us/resources/machine-data.html

Stephen, F. E. (2003). *Notes on Cluster Sampling for Statistics 36-303: Sampling, Surveys and Society.* Carnegie Mellon University, Statistics.

Wang, L., Wang, G., & Alexander, C. A. (2015). Big Data and Visualization: Methods, Challenges and Technology Progress. *Digital Technologies , 1* (1), 33-38.
What Is Machine Data? (s.d.). Obtido de https://www.splunk.com/en_us/resources/machine-data.html

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2013). Discretized Streams: Fault-Tolerant Streaming Computation at Scale. *SOSP '13 ACM SIGOPS 24th Symposium on Operating Systems Principles.*

Zaki, A. K. (2014). NoSQL Databases: New Millennium Database for Big Data, Big Users, Cloud Computing and its Security Challenges. *IJRET: International Journal of Research in Engineering and Technology , 3* (3).

Zikopoulos, P. C., Eaton, C., deRoos, D., Deutsch, T., & Lapis, G. (2012). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data.* United States: McGraw-Hill Osborne Media.

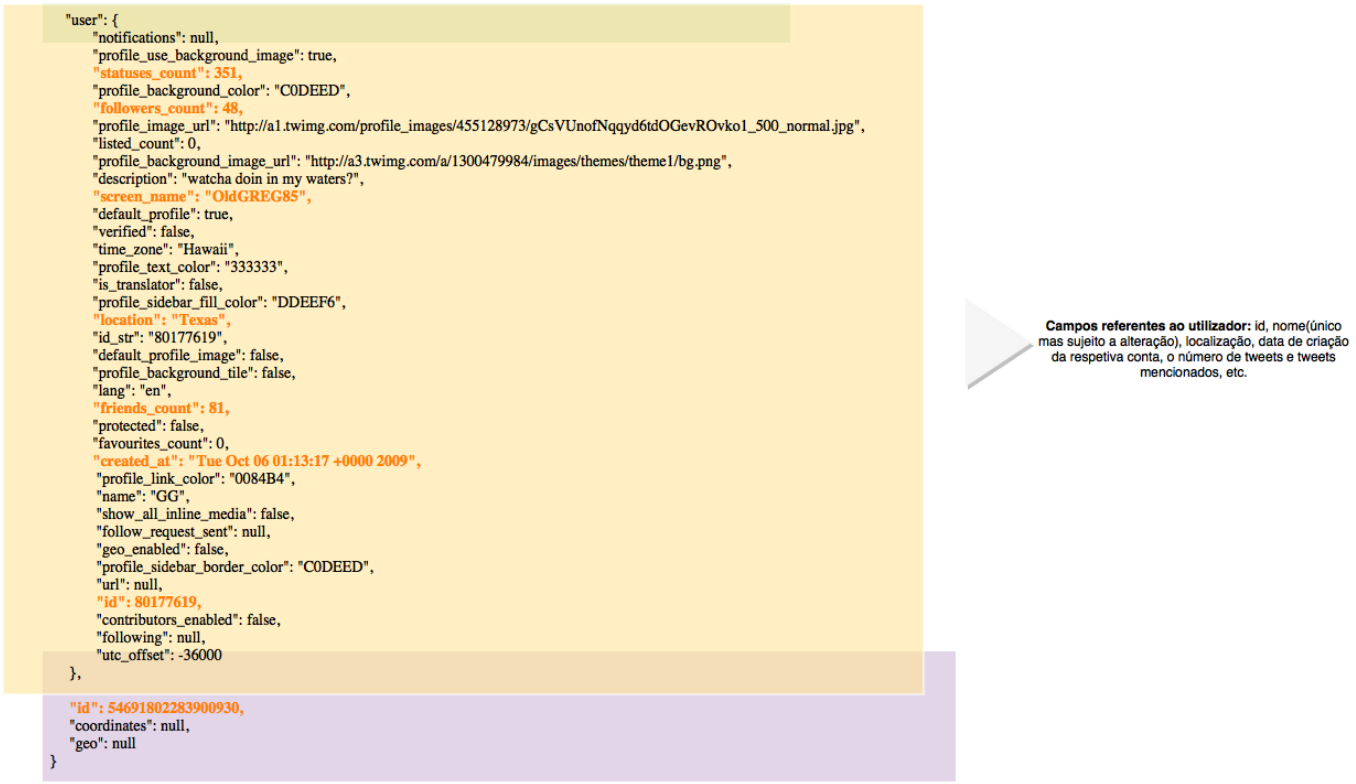
Anexos e Apêndices


```
{ "text": "RT @PostGradProblem: In preparation ... analyzing my fantasy baseball team during ...",
  "truncated": true,
  "in_reply_to_user_id": null,
  "in_reply_to_status_id": null,
  "favorited": false,
  "source": "<a href=\"http://twitter.com/\" rel=\"nofollow\">Twitter for iPhone</a>",
  "in_reply_to_screen_name": null,
  "in_reply_to_status_id_str": null,
  "id_str": "5469180228390092 8",
  "entities": {
        "user_mentions": [
            {  "indices": [3,19],
               "screen_name": "PostGradProblem",
               "id_str": "271572434",
               "name": "PostGradProblems",
               "id": 271572434   }
         ],
         "urls": [ ],
        "hashtags": [ ]
    },
  "contributors": null,
  "retweeted": false,
  "in_reply_to_user_id_str": null,
  "place": null,
  "retweet_count": 4,
  "created_at": "Sun Apr 03 23:48:36 +0000 2011",
```


Campos base de um *Tweet*: id, mensagem, data, lista de entidades como as Hashtags definidas pelo utilizador num url ou no campo da descrição, número de vezes que este tweet foi referenciado...etc


```
  "retweeted_status": {
        "text": "In preparation....analyzing my fantasy baseball team during company time. #PGP",
        "truncated": false,
        "in_reply_to_user_id": null,
        "in_reply_to_status_id": null,
        "favorited": false,
        "source": "<a href=\"http://www.hootsuite.com\" rel=\"nofollow\">HootSuite</a>",
        "in_reply_to_screen_name": null,
        "in_reply_to_status_id_str": null,
        "id_str": "54640519019642881",
        "entities": {
            "media": [
                { "id": 76360760611180544,
                  "id_str": "76360760611180544",
                  "media_url": "http://p.twimg.com/AQ9JtQsCEAA7dEN.jpg",
                  "media_url_https": "https://p.twimg.com/AQ9JtQsCEAA7dEN.jpg",
                  "url": "http://t.co/qbJx26r",
                  "display_url": "pic.twitter.com/qbJx26r",
                  "expanded_url": "http://twitter.com/twitter/status/76360760606986241/photo/1",
                  "sizes": {
                     "large": { "w": 700, "resize": "fit",  "h": 466 },
                     "medium": { "w": 600, "resize": "fit", "h": 399 },
                     "small": { "w": 340, "resize": "fit", "h": 226 },
                     "thumb": { "w": 150, "resize": "crop", "h": 150 }
                   },
                    "type": "photo",
                    "indices": [ 34, 53 ]  }
            ],
            "user_mentions": [ ],
            "urls": [ ],
            "hashtags": [
                 {  "text": "PGP",
                    "indices": [ 130, 134 ]
                 },
                 {  "text": "LaLa",
                    "indices": [ 130, 134 ]
                 }
            ]
        },
        "contributors": null,
        "retweeted": false,
        "in_reply_to_user_id_str": null,
        "place": null,
        "retweet_count": 4,
        "created_at": "Sun Apr 03 20:24:49 +0000 2011",
        "user": {
            "notifications": null,
            "profile_use_background_image": true,
            "statuses_count": 31,
            "profile_background_color": "C0DEED",
            "followers_count": 3066,
            "profile_image_url": "http://a2.twimg.com/profile_images/1285770264/PGP_normal.jpg",
            "listed_count": 6,
            "profile_background_image_url": "http://a3.twimg.com/a/1301071706/images/themes/theme1/bg.png",
            "description": "",
            "screen_name": "PostGradProblem",
            "default_profile": true,
            "verified": false,
            "time_zone": null,
            "profile_text_color": "333333",
            "is_translator": false,
            "profile_sidebar_fill_color": "DDEEF6",
            "location": "",
            "id_str": "271572434",
            "default_profile_image": false,
            "profile_background_tile": false,
            "lang": "en",
            "friends_count": 21,
            "protected": false,
            "favourites_count": 0,
            "created_at": "Thu Mar 24 19:45:44 +0000 2011",
            "profile_link_color": "0084B4",
            "name": "PostGradProblems",
            "show_all_inline_media": false,
            "follow_request_sent": null,
            "geo_enabled": false,
            "profile_sidebar_border_color": "C0DEED",
            "url": null,
            "id": 271572434,
            "contributors_enabled": false,
            "following": null,
            "utc_offset": null
        },
        "id": 54640519019642880,
        "coordinates": null,
        "geo": null
    },
```


Campo referente a um outro *Tweet que é mencionado no Tweet original*: Composto pelos campos básicos referentes ao Tweet fonte, mas com alguns detalhes extra, dado o processo de referênciamento



Anexo 1 - Exemplo de um tweet recolhido da stream