Este artigo está originalmente publicado nos Proceedings do evento onde foi apresentado:

Eds. Fernandes, Paula O.; Nunes, Alcina; Lopes, Isabel Maria; Pereira, João Paulo; Teixeira, João Paulo; Leite, Joaquim; Alves, Jorge; Ribeiro, Nuno A.; Moutinho, Nuno; Raposo, Mário Lino Barata; Ferreira, João José de Matos; Alves, Helena Maria Batista; Leal Millán, Antonio; Barroso Castro, Carmen; Navarro García, Antonio (2020). XXX Jornadas Luso-Espanholas de Gestão Científica. Bragança: Instituto Politécnico. ISBN 978-972-745-279-8

http://hdl.handle.net/10198/22373

IMPLEMENTAÇÃO DE UM LABORATÓRIO DE *BIG DATA* PARA PROCESSAMENTO DE DADOS EM *BATCH* E *STREAMING*

Manuel António Nunes Dias, manuel.dias@estudantes.esce.ips.pt, Escola Superior de Ciências Empresariais do Instituto Politécnico de Setúbal
Vítor Barbosa, vitor.barbosa@esce.ips.pt, Departamento de Sistemas de Informação, Escola Superior de Ciências Empresariais do Instituto Politécnico de Setúbal
Hernâni Mourão, hernani.mourao@esce.ips.pt, Departamento de Sistemas de Informação, Escola Superior de Ciências Empresariais do Instituto Politécnico de Setúbal

RESUMO: *Big Data* é uma área que pretende proporcionar capacidade de processamento dos dados, face ao crescimento exponencial de informação gerada de dia para dia, através de novas tecnologias para recolha, transformação, processamento e análise de dados provenientes de diversas fontes e em diversos formatos. Os desafios do *Big Data* são significativos, daí terem surgido diversas tecnologias num curto espaço de tempo, o que torna também desafiante a entrada nesta área de estudo/investigação. Este artigo apresenta um projeto de implementação de um laboratório de *Big Data*, para processamento de dados históricos e em movimento (*streaming*), cujo propósito é permitir a utilização/exploração das tecnologias associadas em atividades de ensino e investigação. São apresentadas as tecnologias, a arquitetura implementada e testes de processamento de dados realizados para validação da correta configuração e funcionamento do laboratório.

PALAVRAS-CHAVE: Big Data, Fast Data, Hadoop, Kafka.

ABSTRACT: Big Data is a field that aims to provide data processing capacity, facing the exponential growth of information generated daily, through new technologies for collecting, transforming, processing and analysing data from various sources and in various formats. The challenges of Big Data are significant, so many technologies have emerged in a short time, making the entry into this area of study / research challenging as well. This paper presents a project for the implementation of a big data laboratory for processing historical and data in motion (streaming), whose purpose is to allow the use / exploitation of associated technologies in teaching and research activities. The technologies, the implemented architecture and data processing tests performed to validate the correct configuration and operation of the laboratory are presented.

KEYWORDS: Big Data, Fast Data, Hadoop, Kafka.

1. INTRODUÇÃO

O crescimento do volume de dados e o interesse das organizações na obtenção de informação confiável, resultante tanto de análise descritiva como preditiva, que lhes proporcione uma tomada de decisão em tempo oportuno, veio alavancar a implementação de infraestruturas que permitem o processamento de *Big Data* (Erl, Khattak, & Buhler, 2016).

Numa era digital, com milhões de dispositivos interligados e produtores de grandes quantidades de dados, tornou-se necessário criar soluções que permitam efetuar a recolha, processamento, armazenamento e análise de grandes volumes de dados, tanto históricos como em tempo real, sendo um desafio para diversos intervenientes, desde empresas clientes, fornecedores de serviços ou mesmo a academia (Gandomi & Haider, 2015).

Em contexto académico, e de modo a implementar um laboratório que permitisse explorar as tecnologias e realizar trabalhos de investigação com *Big Data*, foi sugerido esse desafio a um aluno finalista de licenciatura como o seu projeto organizacional aplicado (equivalente ao estágio).

O desafio de estudar e implementar uma arquitetura de permita extrair, processar e armazenar elevados volumes dados, oriundos de diversas fontes, tanto em fluxos (*streaming*) como em lotes (*batch*), mantendo a sua integridade e possibilitando uma análise rápida e eficaz, é motivação suficiente para a realização de um projeto cujo objetivo era explorar as tecnologias existentes para processamento de *Big Data*, particularmente em *open-source*, a seleção das tecnologias necessárias à implementação de uma infraestrutura com esse propósito, a concretização dessa infraestrutura num "Laboratório de *Big Data*" e validação do seu correto funcionamento de modo a permitir que, no futuro, o mesmo possa ser usado para ensino e investigação.

A panóplia de *frameworks* de *Big Data* existentes e a complexidade, tanto da escolha das ferramentas adequadas como da sua implementação, requereram um processo de investigação exigente, de forma a proporcionar uma solução exequível e o mais próximo possível da realidade, mesmo tratando-se de ambiente laboratorial, tendo em conta o cumprimento dos objetivos, dos resultados esperados e as limitações dos recursos existentes.

Dada a atualidade do tema e as amplas possibilidades de investigação e ensino com o resultado deste projeto académico, entende-se que o mesmo deve ser divulgado, através da sua descrição neste artigo, de modo a difundir o conhecimento recolhido e possibilitar, de um modo mais fácil, a sua replicação.

Este artigo inclui uma secção (dois) de enquadramento teórico com a apresentação dos tópicos essenciais à compreensão do projeto desenvolvido, a secção três apresentada a metodologia adotada na execução do projeto de investigação, a secção quatro descreve a arquitetura implementada e as tecnologias utilizadas na mesma e a secção cinco mostra como a arquitetura implementada foi testada nos dois tipos de processamento. O artigo termina com algumas conclusões e perspetivas de exploração do laboratório implementado.

2. ENQUADRAMENTO TEÓRICO

Neste capítulo apresentam-se os conceitos e definições teóricas necessários para enquadrar e contextualizar o projeto desenvolvido. Pretende-se efetuar uma sistematização do estado da arte das temáticas *Big Data* e *Fast Data* e dos respetivos ecossistemas, assim como das ferramentas *open-source* a serem utilizadas na *framework* adotada.

2.1 BIG DATA

Segundo (Marr, 2015, p. 9) "O *Big Data* está no centro da revolução inteligente". A ideia básica por detrás de frase "*Big Data*", é que atualmente tudo o que se faz, deixa cada vez mais um rasto digital, que pode ser analisado para tornar a sociedade mais inteligente. O *Big Data* surge como um campo dedicado à análise, processamento e armazenamento de grandes volumes de dados, de diferentes formatos e com origem em diversas fontes (Erl, Khattak, & Buhler, 2016).

A importância do *Big Data* advém da possibilidade que esta solução oferece às organizações de poderem recolher, armazenar, gerir e manipular grandes volumes de dados, na velocidade certa e no momento certo, para obter os *insights* corretos para o apoio à decisão na gestão das suas atividades (Hurwitz, et al., 2013). O processamento e análise de dados em soluções de *Big Data* pode levar à obtenção de grandes beneficios para as organizações nomeadamente: Otimização operacional e redução de custos, inteligência para a sua atividade, identificação de novos mercados, previsões mais precisas, deteção de falhas e fraudes, registos mais detalhados, descobertas cientificas e melhoria na tomada de decisão (Erl, Khattak, & Buhler, 2016).

Numa metodologia tradicional de *business intelligence* (BI), os dados das organizações são armazenados num servidor central e geralmente analisados em modo *off-line*. Os dados transacionais (OLTP - *online transaction processing*), na sua maioria organizados em bases de dados estruturadas, são transferidos para um "armazém de dados" *Data Warehouse* (Sawant & Shah, 2014). Este embiente desnormalizado baseado em bases de dados dimensionais permite a gestão, armazenamento e análise *offline*, com ferramentas analiticas (OLAP - *Online Analytical Processing*) (Caldeira, 2012).

Uma solução de *Big Data* é muito diferente de uma solução tradicional de BI, uma vez que os dados são retidos em sistemas de arquivos distribuídos, em vez de um servidor central. As funções de processamento analítico podem ser efetuadas diretamente a partir dos dados, sejam eles estruturados ou não estruturados,

tanto em tempo real como *offline*. As tecnologias usadas são maciçamente de processamento paralelo (Sawant & Shah, 2014).

De acordo com Wampler (2016) uma arquitura de *Big Data*, na sua essência requer três componentes: um mecanismo de armazenamento disponivel e escalável como uma base de dados ou um sistema de ficheiros distribuido, um sistema de computação distribuida para processamento e consulta de dados em escala, e ferramentas para gerir os recursos e os serviços usados para implementar esses sistemas. Atualmente existem diversos modelos arquiteturais de *Big Data* que permitem extrair, armazenar, processar e analisar grandes volumes dados, a grande velocidade e com origem numa grande variedade de fontes, tanto em lotes (*Batch*) como em fluxos (*Streaming*) em tempo real.

As primeiras três características referenciadas por alguns autores (Sawant & Shah, 2014; Hurwitz, Nugent, Halper, & Kaufman, 2013) como fundamentais do *Big Data*, são o volume, a velocidade e a variedade, no entanto (Erl, Khattak, & Buhler, 2016) atribuem-lhe também a veracidade e o valor.

a) Volume - O primeiro desafio a considerar na utilização do *Big Data* é a grande quantidade de dados recolhidos, provenientes de diversas fontes como redes sociais, aplicações de saúde e bem-estar, transações online, sensores, GPS, etc. Estes dados por si só, podem não fazer sentido, no entanto quando analisados com ferramentas adequadas, é possível compará-los, traçar perfis, determinar tendências e estimar previsões (Erl, Khattak, & Buhler, 2016). Segundo (Sawant & Shah, 2014), para que uma solução de dados seja considerada como Big Data, o volume deve estar pelo menos na faixa de 30 a 50 *terabytes* (TBs).

b) Velocidade - Em ambientes de *Big Data*, os dados podem ser extraídos a grandes velocidades e em grandes volumes, o que permite acumular muitos dados em curtos períodos. Do ponto de vista de uma organização, a velocidade dos dados traduz-se no tempo necessário para que os dados sejam processados quando entram no perímetro da empresa. Lidar com a rápida entrada de dados exige a projeção de soluções de processamento de dados altamente elásticas, disponíveis e com a capacidade de armazenamento correspondente. Dependendo da fonte de dados, a velocidade nem sempre é alta. Por exemplo, as imagens digitalizadas por ressonância magnética não são geradas com tanta frequência como as entradas de logs de um servidor de web com tráfego elevado (Erl, Khattak, & Buhler, 2016).

c) Variedade – A variedade de dados refere-se aos vários formatos e tipos de dados que precisam ser suportados pelas soluções Big Data. Esta multiplicidade de formatos traz desafios para as organizações em termos de integração, transformação, processamento e armazenamento de dados, sejam eles, estruturados, semiestruturados, não estruturados ou metadados (Erl, Khattak, & Buhler, 2016). As fontes de dados podem ser de duas categorias: As que são geradas por máquinas ou computadores e os que resultam da interação do ser humano com estes (Hurwitz, Nugent, Halper, & Kaufman, 2013).

d) Veracidade – Segundo Erl, Khattak e Buhler, (2016) a veracidade refere-se à qualidade ou fidelidade dos dados. Os dados que entram em ambientes de *Big Data* precisam de uma avaliação qualitativa, o que pode levar a atividades de processamento transformação e limpeza, para resolver dados inválidos e remover ruídos (dados que não podem ser convertidos em informação).

e) Valor – Erl, Khattak e Buhler, (2016) definem valor, como a utilidade dos dados para a organização. Os autores afirmam ainda que o valor está intuitivamente relacionado com a veracidade, uma vez que quanto maior for a fidelidade dos dados, mais valor detém para o negócio. O valor e o tempo estão inversamente relacionados, ou seja, quanto mais tempo levar a transformação de dados em informação significativa, menos valor têm para o negócio. Os resultados obsoletos inibem a qualidade e a velocidade da tomada de decisão.

2.2 PROCESSAMENTO DE DADOS EM CONTEXTO DE BIG DATA

De acordo com Krishnan (2013) o processamento de dados pode ser definido como a recolha, tratamento, gestão e geração de informação para o consumidor final. O mesmo autor refere que existem dois estilos *standards* de processamento de dados, um baseado numa arquitetura centralizada num único computador, com grande capacidade de memória, processamento e armazenamento, outro baseado numa arquitetura

distribuída entre regiões geográficas ou *datacenters.* Um exemplo desta arquitetura é a de cluster na qual os dados são processados pelos vários computadores (nós), distribuindo as tarefas entre eles.
Existem vantagens e desvantagens em cada uma das arquiteturas, no entanto, para ecossistemas de processamento de *Big Data*, a mais adequada é a que se baseia em modelos distribuídos, uma vez que permite a conceção de sistemas escaláveis e com processamento paralelo, que reduzem a latência (Krishnan, 2013).

Para que a projeção de uma arquitetura de *Big Data* seja eficiente, torna-se necessário conhecer bem cada um dos quatro estágios de processamento de dados: recolha, carregamento, transformação e extração. Com base neste fluxo de dados, a infraestrutura de processamento de *Big Data* tem como requisitos fundamentais: ser linearmente escalável, de alto rendimento, permitir o processamento para vários cenários do negócio, tolerante a falhas, auto recuperação em caso de falha, elevado grau de processamento paralelo e baseado em sistemas de ficheiros distribuídos (Krishnan, 2013).

Segundo Du (2015), o processamento de dados pode ser efetuado nas seguintes formas:

a) Em lotes (*Batch*) – Nesta forma de processamento os dados são lidos da fonte, processados e armazenados num sistema ficheiros partilhado e distribuído. A *framework open-source* Hadoop[1], da Apache, é a mais utilizada e permite o processamento de *Big Data* em clusters de computadores de forma distribuída, confiável, escalável e tolerante a falhas, através de um sistema de ficheiros distribuídos, designado de Hadoop Distributed File System (HDFS). O acesso aos dados é efetuado através de um sistema de processamento paralelo MapReduce apoiado numa estrutura de gestão de tarefas e recursos do *cluster* YARN (Yet Another Resource Negotiator) (Du, 2015).

b) Tempo Real (*Real-Time*) – No processamento em tempo real, os dados são processados no momento em que são recolhidos. Este processo utiliza um formato de armazenamento colunar com uma estrutura de índice rápido e algoritmos de agregação escalonáveis, que permitem calcular os resultados das consultas em sequências paralelas em vez de lotes. A computação *in-memory* (utilização da memória RAM - *Random Access Memory*) permite soluções de processamento em tempo real que oferecem uma largura de banda de 10 gigabytes/segundo, em comparação com 520 megabytes/segundo dos discos SSD. O motor de processamento analítico de código aberto Apache Spark[2], permite computação em memória e pode ser facilmente integrado com o Hadoop (Du, 2015).

c) Fluxos *(Streaming*) – O processamento em fluxos é um processo de transmissão contínua que permite publicar os dados e obter resultados à medida que os eventos acontecem (Du, 2015). Uma das plataformas de *streaming* mais populares é da Apache Kafka[3], que permite publicar, processar e armazenar fluxos de registos à medida que ocorrem, de forma durável e tolerante a falhas (Apache Kafka, 2017).

2.3 ARQUITETURA DE *BIG DATA* TRADICIONAL

Existem diversas *frameworks* para *Big Data* no mercado, no entanto, o Hadoop (*framework* de código aberto para computação distribuída confiável, escalável e de baixo custo), estabeleceu-se como a principal arquitetura utilizada pelas organizações, para soluções de tratamento e análise de grandes volumes de dados em *batch*.

Segundo Sawant & Shah, (2014), os componentes essenciais de uma arquitetura de *Big Data* baseada em Hadoop, estão representadas na figura 2, onde são apresentadas as diversas camadas que a constituem, nomeadamente: *Data Sources, Ingestion Layer, Security Layer, Monitoring Layer, Hadoop Infrastructure Layer, Distributed (Hadoop) Storage Layer, Hadoop Platform Management Layer, Analytics Engines e Visualization Layer*.

A *framework* Hadoop possibilita o armazenamento de elevados volumes de dados de forma relativamente *low-cost* comparativamente a outras soluções. No núcleo dessa capacidade de armazenamento está o sistema distribuído de ficheiros (HDFS), que permite um armazenamento distribuído de grandes volumes de dados (*terabytes* ou *pentabytes*) em diversas máquinas (*nodes*) de um *cluster*. Este sistema de arquivos confere um

[1] http://hadoop.apache.org/.

[2] https://spark.apache.org/.

[3] http://kafka.apache.org/.

elevado grau de redundância e capacidade de processamento paralelo de alta velocidade (Sawant & Shah, 2014).

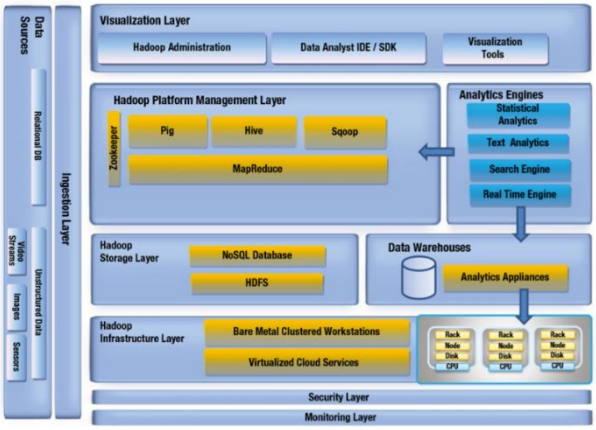


Figura 1: Arquitetura de *Big Data*
Fonte: (Sawant & Shah, 2014, p. 10)

O HDFS possui uma arquitetura *Master/Slave* conforme se pode observar na Figura 2. Um *cluster* HDFS consiste num NameNode (servidor *master*) que efetua a gestão do sistema de arquivos, determina o mapeamento de blocos para os DataNodes e regula o acesso a arquivos por clientes, o qual pode ter réplicas de modo a tolerar falhas. Os DataNodes são responsáveis por atender solicitações de leitura e gravação dos clientes do sistema de arquivos (Apache Hadoop, 2019).

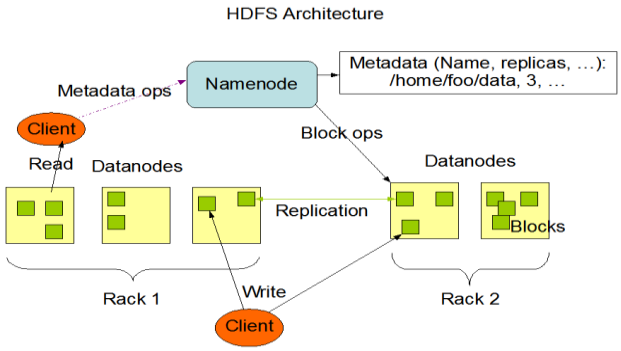


Figura 2: Arquitetura HDFS
Fonte: HDFS Architecture Guide (Apache Hadoop, 2019)

No HDFS, o armazenamento de cada arquivo é efetuado numa sequência de blocos, em que todos os blocos, exceto o último, são do mesmo tamanho (128Mb por omissão). Os blocos são replicados pelos *Datanodes* para permitir tolerância a falhas. Tanto o tamanho como o fator de replicação são configuráveis (Apache Hadoop, 2019).

Esta estrutura de dados distribuída, é compatível com bases de dados não relacionais e altamente escaláveis como a NoSQL (*Not only SQL*). Esta base de dados é tolerante a falhas projetada para guardar dados semi-estruturados e não estruturados (Erl, Khattak, & Buhler, 2016).

Sendo o armazenamento e processamento realizado de forma distribuída, como referido anteriormente, a infraestrutura física reflete essa realidade, ou seja, os dados são armazenados fisicamente em computadores diferentes, ligados em rede. O Hadoop YARN (*Yet Another Resource Negotiator*) e o HDFS têm a capacidade de gerir a camada de infraestrutura num ambiente virtualizado, tanto a nível local como na *cloud* ou através de computação distribuída de servidores numa rede de alta velocidade (Sawant & Shah, 2014).

O YARN[4] funciona como "sistema operativo" que tem a capacidade de gerir os recursos da infraestrutura fazendo o agendamento e monitorização das tarefas, através de um *ResourceManager*, que efetua a

[4] http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

negociação entre os clientes e os *NodeManagers*, responsáveis pela gestão da execução dos pedidos em cada nodo. (Apache Hadoop, 2019).

A camada *Management* fornece ferramentas e linguagens que permitem a execução de consultas às bases de dados *NoSQL* ou diretamente ao HDFS. Segundo os autores Sawant & Shah, (2014) o paradigma de execução mais comum no Hadoop é o MapReduce[5]**.** A função *map* é utilizada para processar a unidade de informação distribuída pelos nodos gerando os resultados na forma (chave, valor) a serem enviados para o *reduce.* A função *reduce* é utilizada para agregar os dados pelas chaves dos dados recebidos. Este tipo de gestão confere ao sistema uma elevada resiliência.

2.4 ARQUITETURA DE *STREAMING* (*FAST DATA*)

O crescimento da economia digital veio criar a necessidade de os gestores das organizações tomarem decisões cada vez mais rápidas, baseadas em informação fidedigna e de preferência em tempo quase real. Este processo de transformação digital veio aumentar a variedade, volume e velocidade dos dados, o que permitiu alavancar a evolução das arquiteturas clássicas de processamento em *batch*, baseadas na *framework* Hadoop, para arquiteturas que permitem processar dados em *streaming* capturados em tempo real.

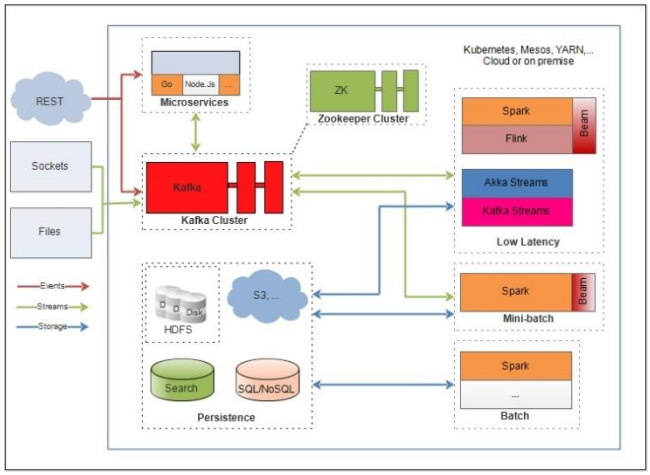


Figura 3: Fast data (streaming) architecture
Fonte: Adaptado de (Wampler, 2018, p. 10)

Segundo Estrada (2018), o sucesso de uma arquitetura de *Fast Data* depende de três requisitos fundamentais: Aquisição e carregamento de dados confiáveis e atualizados; flexibilidade na consulta e armazenamento; ferramentas de análise sofisticadas. Os sistemas modernos de *Fast Data* são compostos por quatro etapas de transformação: aquisição, armazenamento, processamento e análise, apresentação e visualização (Estrada, 2018).

Na Figura 3, está representada a arquitetura de processamento em tempo real, elaborada por Wampler (2018), que enumera as várias fases do processamento de dados em fluxos e pretende cobrir a maioria dos recursos essenciais para desenvolver sistemas de *streaming*.

2.4.1 Cluster Kafka

O Kafka[6] é um *software open-source* que permite construir *pipelines* em tempo real, de dados e aplicações de *streaming* com origem em diversas fontes, de forma distribuida, replicada e tolerante a falhas. Ou seja, funciona como um sistema de leitura, escrita, processamento, armazenamento distribuido e replicado de mensagens ou eventos em tempo real (Apache Kafka, 2017). Um cluster Kafka é a espinha dorsal da arquitetura de *streaming* e geralmente têm *hardware* dedicado, de forma a permitir o máximo de escalabilidade no carregamento dos fluxos de dados e o aumento do desempenho no seu processamento (Wampler, 2018).

[5] http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

[6] http://kafka.apache.org/.

Como se pode verificar na Figura 4, onde se representa um cenário típico de agregação e análise de *Big Data* suportado pelo sistema de mensagens Apache Kafka, um cluster pode ser constituido por um ou mais *brokers* (servidores), que podem ter um ou mais processos de streaming em execução (Garg, 2015).

O ZooKeeper serve como interface de coordenação entre o Kafka Cluster e os Consumers. O *broker* usa o Zookeeper para obter informações sobre o seu estado e configuração e o consumer usa-o para rastrear a localização das mensagens (Garg, 2015).

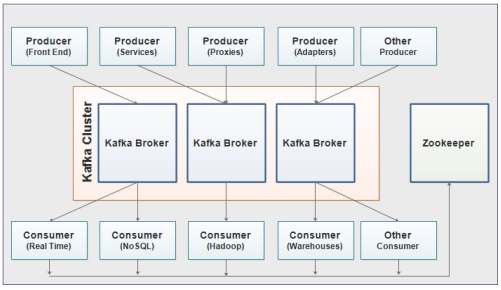


Figura 4: Cluster Kafka
Fonte: Adaptado de Garg (2015)

O *cluster* Kafka armazena fluxos de registos em categorias denominadas tópicos, cada registo é guardado no sistema com uma chave, valor, data e hora (*timestamp*). Os tópicos podem ser divididos em partições e réplicas configuráveis de acordo com o número de *brokers* (Apache Kafka, 2017).

A Figura 5, mostra os cinco componentes de um cluster Kafka: Zookeeper, Broker, Topic, Producer e Consumer. De acordo com a informação oficial (Apache Kafka, 2017), este é constituido por quatro APIs (*Application Programming Interface)* principais:

- *Producer* - Os *producers* publicam mensagens nos tópicos escolhidos e são responsáveis por selecionar os registos a atribuir a cada partição dentro do tópico. Existem diversos tipos de *producers*, desde *logs* provenientes de aplicações web, serviços de rastreio de chamadas, *proxies* de análise, entre outros (Garg, 2015).

- *Consumer* – Os *consumers* permitem que as aplicações subcrevam um ou mais tópicos e processem os fluxos de registos recebidos dos *producers* (Apache Kafka, 2017). Existem diversos tipos de *consumers: Off-line*, que consomem e armazenam mensagens no Hadoop ou em *Data Warehouses* tradicionais para análise posterior; *Near real-time*, que consomem e armazenam mensagens em bases de dados NoSQL como HBase ou Cassandra[7] para análise em tempo quase real; *Real-time*, como o Spark ou Storm[8], que filtram mensagens em memória e criam eventos de alerta em tempo real (Garg, 2015).

- *Streams* – Esta API permite que uma aplicação atue como processador de fluxos, ou seja, consome um fluxo de um ou mais tópicos e produz um fluxo de saída para um ou mais tópicos, transformando efetivamente os fluxos de entrada em fluxos de saída.

- *Connector* - API que permite criar e executar *producers* ou *consumers* reutilizáveis, que conectam tópicos do Kafka a aplicações ou sistemas de bases dados. Por exemplo, um conector para uma base de dados relacional pode capturar todas as alterações numa tabela.

[7] http://cassandra.apache.org/.
[8] http://storm.apache.org/.

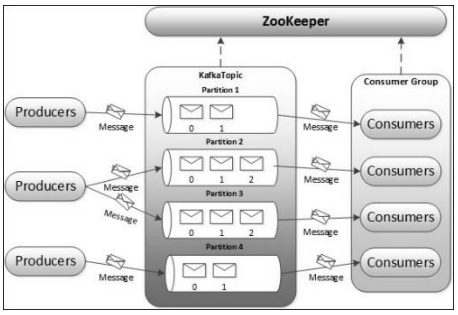


Figura 5: Componentes de um cluster Kafka
Fonte: (Garg, 2015)

2.4.2 REST e Microservices

O REST (*Representational State Transfer*) atua como um canal de comunicação sincrona ou assincrona entre o Kafka e os microserviços, através de *websockets* (tecnologia que permite a comunicação bidirecional por canais full-duplex em TCP (*Transmission Control Protocol*) (Wampler, 2018).

Em infraesturas grandes, a utilização de microserviços permite decompor as funções em tarefas relativamente mais simples, que podem ser geridas por pequenas equipas mais especializadas (Dunning & Friedman, 2016).

2.4.3 Persistência

O armazenamento dos fluxos de dados é garantida através do Kafka Connect que permite mover grandes volumes de dados muito rápidamente, para as bases de dados ou sistemas de ficheiros alocados à infraestrutura. Por exemplo NoSQL/SQL, HDFS, AWS S3[9] (Amazon Web Services) entre outras (Wampler, 2018).

Num cluster Kafka, a configuração padrão para a retenção das mensagens de um tópico em cada um dos brokers, são sete dias ou até que atinja um valor configurável, definido em número de bytes (exemplo. 1Gb). Quando o limite é atingido, as mensagens expiram e são excluidas por ordem de chegada (Narkhede, Shapira, & Palino, 2017).

2.4.4 Processamento e análise de *Streaming*

Nas arquiteturas de processamento de streaming existem três formas de processamento:

a) Baixa latência - Através da ingestão de dados do Kafka para motores de processamento de *streaming* como o Flink[10] e o Spark[11] que fornecem tipos de análise semelhantes e podem funcionar como "corredores" para fluxos de dados definidos com o Apache Beam[12]. O Akka Streams[13] e o Kafka Streams[14] que fornecem menor latência e menor sobrecarga, mas são menos orientados para a criação de serviços analíticos e mais para a criação de microserviços gerais sobre dados de fluxo contínuo (Wampler, 2018).

b) *Mini-batch* - O modelo *mini-batch* do Apache Spark Streaming[15] pode ser usado quando são toleradas latências um pouco mais longas (100 milissegundos ou intervalos mais longos), uma vez que esta janela de tempo extra é valiosa para cálculos mais complexos, como por exemplo, aperfeiçoamento de algoritmos de *Machine Learnig* e processamento de gráficos. No entanto, o Spark Streaming está a ser

9 https://aws.amazon.com/pt/.
10 https://flink.apache.org/.
11 https://spark.apache.org/.
12 https://beam.apache.org/.
13 https://akka.io/.
14 https://kafka.apache.org/documentation/streams/.
15 https://spark.apache.org/docs/latest/streaming-programming-guide.html.

substituido pelo Structured Streaming,[16] uma vez que este já atingiu um estado de maturidade suficiente para ser implementado. (Wampler, 2018).

c) *Batch* - Com a associação do Spark a um sistema de armazenamento persistente, como o HDFS ou Base de Dados, é possível realizar processamento em lotes e análise interativa. Isto significa que a arquitetura representada é suficientemente flexível para suportar cenários de análise tradicionais (Wampler, 2018).

O modelo de arquitetura de *Fast Data* descrito anteriormente pode ser implemento em clusters Mesos[17] ou Hadoop / YARN, bem como em ambientes em nuvem, como AWS , Google Cloud Environment ou Microsoft Azure (Wampler, 2018).

2.5 BATCH VS STREAMING

De acordo com Wampler, (2018, p. 12) os três componentes essenciais de uma arquitetura em *batch* (Hadoop) são, o HDFS para armazenamento, o MapReduce e/ou o Spark para processos de computação e o YARN para controlo, enquanto na arquitetura de *streaming* o Kafka é utilizado para armazenamento, o Spark, Flink, Akka Streams e Kafka Streams são usados para computação, e o controlo pode ser feito pelo Kerberos, Mesos ou pelo YARN (com algumas limitações).

Embora a arquitetura *Fast Data* possa armazenar *datasets* em *petabytes* (PB), uma tarefa de *streaming* funciona normalmente em *megabytes* (MB) ou no máximo *terabytes* (TB). No entanto, o tempo de processamento e execução de tarefas em *streaming* pode ir de microssegundos a minutos em vez das horas necessárias para processamento em *batch* (Wampler, 2018).

Os mecanismos de *streaming* com latências muito baixas como o Kafka, permitem funcionalidades similares às do *MapReduce* realizado em ambientes Hadoop, com a diferença que podem processar *terabytes* em microssegundos em vez das altas latências dos sistemas de *batch* (Narkhede, Shapira, & Palino, 2017).

Uma arquitetura de *Big Data* deve configurar as duas soluções de processamento, uma vez que nem todos os processos exigem uma resposta imediata. O processamento em *batch* tem grande eficiência, é altamente escalável, de baixo custo e processa dados em repouso, enquanto o *streaming* permite dar respostas imediatas à medida que os eventos acontecem continuamente nas organizações, melhorando a capacidade de resposta a situações como deteção de fraude, ajustamento de preços em tempo real, alertas de situações clínicas entre outras (Narkhede, Shapira, & Palino, 2017).

3. METODOLOGIA DE INVESTIGAÇÃO

O trabalho de investigação, conceção e desenvolvimento deste projeto, foi baseado numa metodologia aplicada a sistemas de informação, designada de *Design Science Research Methodology*, proposta por Peffers et al. (2007). Segundo os autores, esta metodologia de investigação e desenvolvimento, incorpora princípios, práticas e procedimentos que permitem conduzir os investigadores durante as seis fases do processo, indicadas na Figura 6, e fornecer um modelo mental para a apresentação dos resultados.

16 https://spark.apache.org/docs/2.2.0/structured-streaming-programming-guide.html.

17 http://mesos.apache.org/.

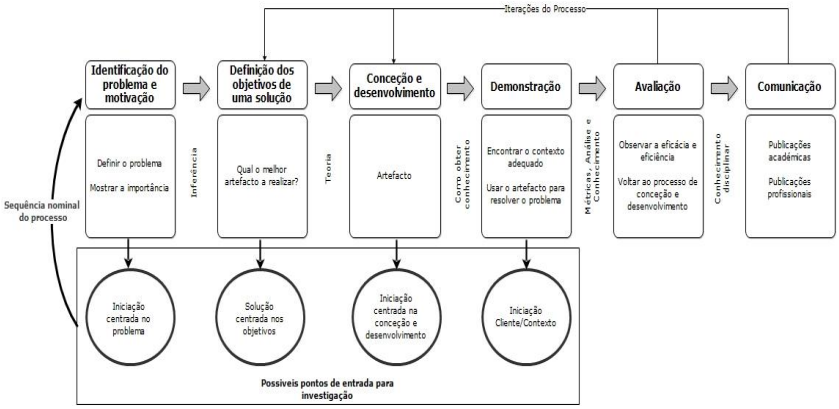


Figura 6: *Design Science Research Methodology Process Model*
Fonte: Adaptado de Peffers et al. (2007, p. 54)

a) Identificação do problema e motivação

Nesta fase, a estratégia de investigação adotada deve ser centrada no problema, de forma a ser definida uma pesquisa específica para a solução a implementar. A relevância da implementação de uma *framework* no laboratório que permita recolher, tratar e processar grandes volumes de dados, eventualmente em tempo real, assumiu-se como o problema a abordar, tendo em conta o tempo e recursos disponíveis para realizar o projeto.

b) **Definição dos objetivos para a solução**

O processo de definição de objetivos foi iniciado com a realização de um plano de atividades para os três meses de duração do projeto. Foi nesta fase que foram iniciadas pesquisas na internet, em *sites* e repositórios científicos, sobre as temáticas de *Big Data* e *Fast Data*, onde foi obtida toda a informação bibliográfica e técnica, necessária para sustentar a escolha de uma solução arquitetural que permitisse a resolução do problema.

c) **Conceção e desenvolvimento**

Com base na identificação do estado da arte, resultante da fase anterior, foi criado um repositório das arquiteturas, ecossistemas e plataformas de *Big Data e Fast Data* mais usadas pelos fornecedores de soluções, baseadas em ferramentas de *open-source*. Decorrente dessa informação, foram propostas duas soluções para a implementação de uma *framework* de *Big Data* no laboratório, optando-se pela possibilidade de testar as duas soluções em coexistência. A infraestrutura resultante da implementação, conjuga tecnologias de ambas as soluções e será descrita no próximo capítulo "Implementação".

d) **Demonstração**

Nesta fase, Peffers et al. (2007) indiciam que a demonstração da infraestrutura deve ser efetuada com um estudo de caso de uma determinada atividade para experimentar, simular e resolver um problema concreto, validando assim, a arquitetura proposta. Será demonstrada a utilização da infraestrutura criada no processamento de dados em *batch* e em *streaming* de modo a, essencialmente, validar que a mesma se encontra funcional e demonstrar contextos onde a mesma pode ser utilizada.

e) **Avaliação**

Relativamente à avaliação, os autores referem que os resultados reais da implementação devem ser comparados com outras soluções, para a resolução do mesmo problema, de forma a medir a eficácia e eficiência do sistema. É nesta fase que se devem fazer as alterações necessárias para melhorar o desempenho do sistema. No presente projeto, não foi possível proceder à avaliação comparativa de desempenho do sistema, não sendo, por enquanto, uma prioridade, pois o objetivo é disponibilizar o laboratório para fins

pedagógicos e de investigação, podendo este, futuramente, ser ajustado para a otimização do seu desempenho em problemas concretos.

f) **Comu\nicação**

Nesta fase deve ser efetuada a divulgação do projeto e realçar a importância, *design* e eficiência, aos investigadores e outros públicos relevantes. A comunicação deste projeto foi efetuada internamente, foi também descrito no relatório final do projeto do aluno que o realizou e está a ser feita através da sua apresentação neste artigo.

4. IMPLEMENTAÇÃO

Neste capítulo será apresentado o projeto de criação de uma infraestrutura de Big Data, em contexto académico e com recursos limitados, que permita, no futuro, a sua utilização para novos trabalhos de investigação que necessitem deste tipo de tecnologia. Ao projeto foi atribuída a designação "Implementação de um Laboratório de *Big Data* para processamento de dados em *batch* e *streaming*".

Decorrente de reuniões realizadas com docentes do Departamento de Sistemas de Informação, decidiu-se iniciar a criação de uma unidade laboratorial de *Big Data*, que permita a recolha, tratamento, armazenamento e análise de grandes volumes de dados, provenientes de diversas fontes, a diferentes velocidades e mantendo a sua integridade.
O propósito da implementação do laboratório é a criação de valor académico no apoio aos projetos de investigação da comunidade científica e no desenvolvimento curricular dos alunos, proporcionando o acesso a ferramentas de código aberto, recentes e eficazes no tratamento analítico de grandes volumes de dados.

Objetivando uma solução arquitetural para o projeto, foi feito um trabalho de investigação sobre o estado da arte das soluções e ferramentas de código aberto, na área de *Big Data*, que permitam garantir a implementação uma infraestrutura escalável, que se adeque aos meios técnicos existentes. Parte do resultado desse estudo foi incluído no enquadramento teórico deste artigo.

Como foi referenciado, na fase metodológica da definição dos objetivos, da realização do processo de investigação e análise de soluções, resultou a identificação de dois tipos de arquiteturas de *Big Data* (em *Batch* e *Streaming*), apontando uma possível solução arquitetural única, escolhida com base nas tecnologias de *open-source* mais utilizadas para os dois tipos de processamento, e que se representa na Figura 7. A solução apresentada permite o processamento em *Batch* e *Streaming* e é suportada nas tecnologias Hadoop, Kafka e Spark.

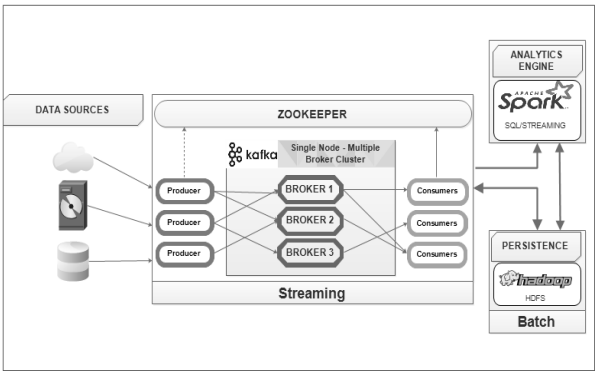


Figura 7: Arquitetura de *Big Data*

No modelo proposto, os dados oriundos de diversas fontes são recolhidos e processados em *streaming* num cluster Kafka, implementado num único nó com três *brokers*. Os dados são carregados por *producers* e armazenados de forma distribuída e replicada em tópicos, que podem ser subscritos e processados em tempo real por um ou mais *consumers*. O Zookeeper atua como coordenador entre os diversos componentes do cluster. O Spark é usado para processamento analítico *in-memory* de *streaming* ou *batch* e utiliza o HDFS para armazenamento persistente. O *cluster* Hadoop permite o armazenamento de grandes volumes de dados de forma distribuída, redundante e tolerante a falhas, podendo ser usado por diversas ferramentas que

utilizam o paradigma *MapReduce*. Trata-se, portanto, de uma *framework* que permite escalabilidade e à qual podem ser associadas outras tecnologias de processamento, armazenamento e análise de dados.

4.1 FRAMEWORK DE BIG DATA IMPLEMENTADA

Com se pode verificar na Figura 8, a *framework* adotada foi desenhada com base em dois clusters, um Kafka e outro Hadoop.

4.1.1 Cluster Kafka

O *cluster* Kafka *Single-Node* é constituído por três *brokers* e por um coordenador Zookeeper, todos virtualizados num único computador. Na mesma máquina, foi instalado um servidor virtual de KSQL (ferramenta incluída na plataforma Confluent) usado como *consumer* para realização de consultas de *streaming* e *producer* de tópicos efetuados sobre essas consultas.

4.1.2 Cluster Hadoop

O cluster Hadoop é constituído por um nodo *Master* e três *Slaves*, todos instalados em máquinas físicas diferentes. No nodo Master foi instalado um servidor virtual com o software analítico Apache Spark, que pode operar como *consumer* do *cluster* Kafka, para tarefas de análise e armazenamento de fluxos de dados no HDFS.
A framework implementada permite recolher, processar, armazenar e analisar dados em *streaming* e *batch*, podendo ainda configurar outro tipo de ferramentas, ou bases de dados, que eventualmente venham a ser necessárias para a realização de projetos concretos.

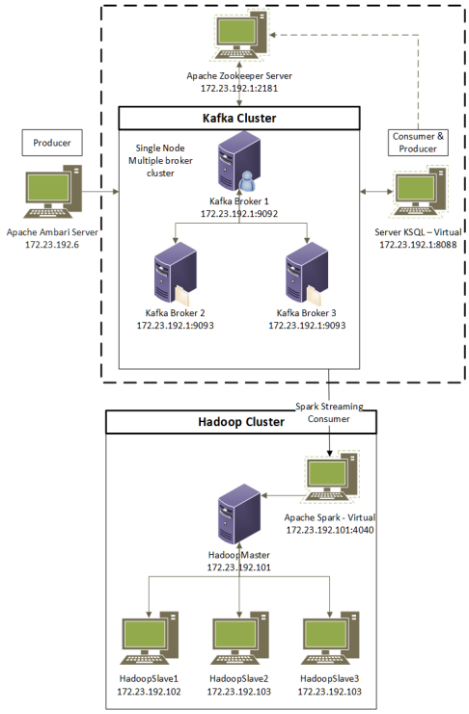


Figura 8: *Framework* de *Big Data* implementada

4.2 INSTALAÇÃO E CONFIGURAÇÃO DOS COMPONENTES

A instalação e configuração dos componentes de uma *framework* de Big Data requerem elevados conhecimentos de algumas linguagens e sistemas operativos. No caso da infraestrutura implementada sugere-se como fundamental deter conhecimentos de operação e configuração de sistemas operativos Linux, redes e linguagens como Java, Python, Scala, SQL, KSQL e Pig latin.

Nas alíneas seguintes descrevem-se os processos de instalação/configuração do software necessário à implementação da infraestrutura:

a) Sistemas Operativos

O sistema operativo instalado nos seis computadores utilizados foi o Ubuntu desktop[18], (cinco com a versão 18.04LTS e um com a versão 16.04LTS por exigência de compatibilidade com a plataforma Apache Ambari).

b) Componentes do ecossistema Kafka

O Apache Kafka foi instalado num único servidor *Single Node-Multiple Brokers* (computador com CPU Intel Core I7-7700 3.60GHz – 8 cores e 64Gb RAM*)*. Os componentes instalados foram o Java, Zookeeper e Apache Kafka. Foi instalado o Kafka Tool que permite visualizar, gerir e navegar pelos tópicos armazenados no cluster.
No mesmo servidor foi ainda instalado o Server-KSQL integrado na plataforma *open-source* da Confluent (Confluent, Inc, 2018).

c) Componentes do ecossistema Hadoop

O cluster Hadoop MultiNode é constituído por quatro computadores (CPU Intel Core I7-7700 3.60GHz – 8 cores 16Gb RAM) onde foram instalados/configurados o Apache Hadoop e o Java.
No Hadoop Master foi instalado o software Apache Pig[19] para desenvolvimento/teste de aplicações MapReduce no cluster Hadoop.

d) Apache Spark
O software de processamento analítico Spark foi instalado no servidor físico do Hadoop Master.
e) Plataforma Ambari

A Apache Ambari é uma plataforma baseada no Web Browser que permite simplificar a instalação, gestão e monitorização de clusters Hadoop. Foi instalada num computador com CPU Intel Core I7-7700 3.60GHz – 8 cores 64Gb RAM, recorrendo às instruções dos sites da Apache[20]. Ainda não foi usada neste projeto.

Neste servidor foram também instalados dois geradores de dados em *streaming* para carregar os tópicos do *cluster* Kafka: O Apache JMeter[21] e o Json Data Generator[22].

5. AVALIAÇÃO

No âmbito deste projeto, a avaliação pretende apenas demonstrar o funcionamento da arquitetura proposta, de modo a validar a correta instalação e configuração dos seus componentes e, paralelamente, demonstrar o modo de funcionamento dos dois tipos de processamento que se pretende executar.

5.1 TESTES DE PROCESSAMENTO EM *STREAMING*

De modo a testar e analisar o processamento de dados em *streaming* (processados em tempo real) foi criado um cenário simulado. Pretendeu-se simular a recolha de eventos de admissões em urgências hospitalares, sendo gerados dados aleatórios que representam utentes que são admitidos e que variam em género, idade, especialidade em que são afetos, e também algumas variáveis aleatórias para representar o tempo de triagem, tempo de consulta e alta hospitalar. A área da saúde é um contexto onde o processamento em tempo-real pode fazer a diferença (Baljak, Ljubovic, Michel, Montgomery, & Salaway, 2018).

18 https://www.ubuntu.com.
19 http://pig.apache.org/docs/r0.17.0/start.html.
20 https://cwiki.apache.org/confluence/display/AMBARI/Ambari+User+Guides.
21 http://jmeter.apache.org/index.html.
22 https://github.com/acesinc/json-data-generator.

A simulação criada está esquematizada na Figura 9. Foi utilizado o Json Data Generator como *producer* dos "eventos" das urgências, sendo possível lançar diversos processos e com *timings* diferentes na frequência com que cada evento é gerado de modo a simular variações no fluxo de procura da urgência. As mensagens dos geradores são enviadas para o cluster Kafka, sendo registadas num tópico ("Fluxo Saúde") com três partições (as partições permitem distribuir as mensagens pelos vários *brockers*, sendo normalmente usadas as chaves para decidir em que partição uma mensagem vai ser guardada). Foi usado o KSQL para ler o *stream* "StreamOrigem" (que permite ler as mensagens do tópico "Fluxo Saúde") e criar um novo *stream* "Fluxo por Consulta" e um tópico com o mesmo nome onde as mensagens foram separadas pela especialidade em que os utentes foram atendidos, sendo este tópico distribuído por quatro partições, em que cada especialidade está em apenas uma partição (não foi configurada a replicação das mensagens). Para ler os *streams* de dados foi utilizado o Spark que assumiu o papel de *consumer* e que permitiu fazer alguns tipos de análises simples aos dados obtidos e guardar os mesmos no cluster Hadoop para futuro processamento em *batch*.

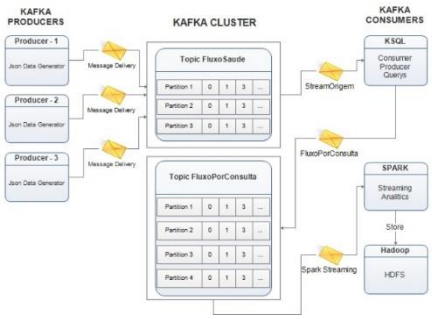


Figura 9: Processamento em *Streaming*

Nos processos de produção e transformação realizados no *cluster* Kafka, os dados são armazenados em cada partição de um tópico durante um período de sete dias. Podem ser usados pelos *consumers* à medida que vão sendo processados em *streaming* ou quando necessário.

A Figura 10 mostra as estatísticas de uma simulação de processamento em *streaming* efetuado pelo Spark durante o período de 1 hora e 19 minutos, onde realizou 951 *batches* completos. O script definido no Spark foi configurado para consultar o *stream* de dados a cada 5 segundos (1 batch corresponde ao processamento das mensagens que chegam ao Kafka nos últimos 5 segundos).

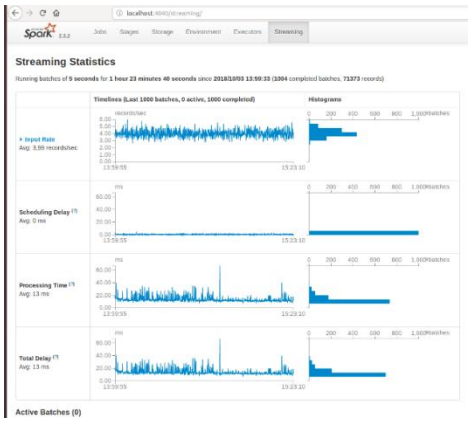


Figura 10: Spark - Processamento de *streaming.*

Os fluxos de dados recebidos pelo Spark Streaming, podem ser tratados pelas ferramentas incluídas no Spark ou por outras ferramentas analíticas que permitam a conectividade com este. No cenário implementado, como os fluxos são armazenados para persistência no cluster Hadoop, podem também ser tratados a partir de ferramentas compatíveis com o HDFS.

5.2 TESTES DE PROCESSAMENTO EM *BATCH*

Os testes de processamento em *batch* foram iniciados com a criação de ficheiros de grande dimensão para distribuição no sistema de ficheiros HDFS.

As interfaces Web do HadoopMaster permitem fazer a gestão do cluster Hadoop e de todas as tarefas relacionadas com a distribuição HDFS e com operações de *MapReduce*. Os gestores de recursos e tarefas são visíveis nas portas 8088 e 50070, enquanto o histórico está acessível na porta 19888.

A Figura 11 mostra um exemplo da *Browse Directory* onde é possível verificar as propriedades das pastas e ficheiros do HDFS, assim como a forma como estão distribuídos, partidos e replicados pelos nodos do cluster.

No exemplo da Figura 11, os três primeiros ficheiros representam *inputs*, um dos quais com 16.79Gb (designado por grande15G) que contêm o texto de um livro replicado 15000 vezes. Este ficheiro foi usado para testes de MapReduce, diretamente no Hadoop e através da utilização do software Pig.

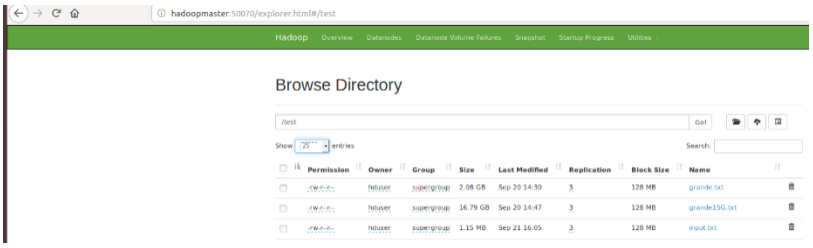


Figura 11: Hadoop *Browse Directory*

A Figura 12, representa um quadro onde é possível visualizar três processos de *mapreduce* de contagem de palavras, dois terminados e um em execução (34,9%). No processo em execução é possível verificar os recursos atribuídos à operação pelo HadoopMaster, como o número de nodos ativos, os containers utilizados, a memória e o número de *cores* atribuídos. O processo em execução refere-se ao ficheiro anteriormente designado como grande15G.

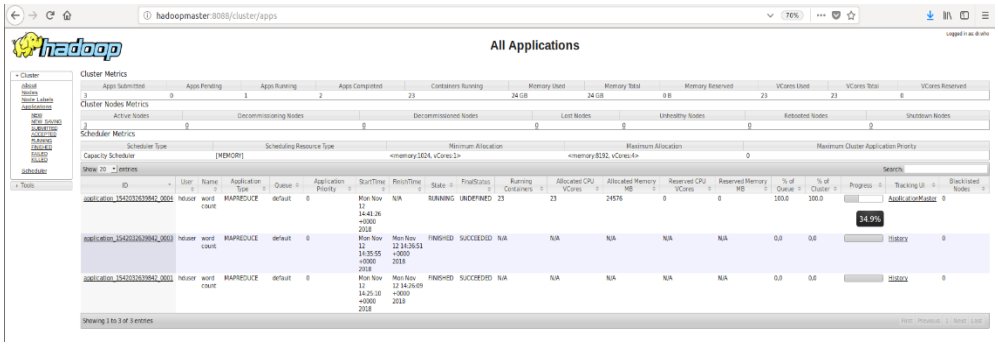


Figura 12: Processos de *MapReduce* no Hadoop

Na Figura 13, representa-se a informação histórica dos processos onde é possível verificar se foram concluídos com sucesso e retirar mais algumas informações úteis. Para o ficheiro anteriormente referido (jobs 004, 005), o tempo de execução com um ou dois *reducers* foi respetivamente (4,58m e 4,55m) e o número de *maps* efetuados de 135 em cada um dos processos.

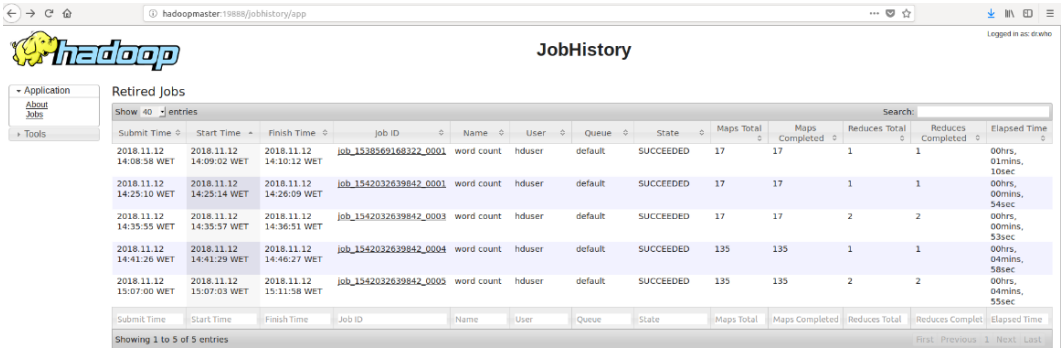


Figura 13: Histórico do processo de MapReduce

A plataforma de software Pig, foi utilizada na realização de alguns processos de *mapreduce* com os mesmos ficheiros usados no Hadoop MapReduce. Na execução do Pig em modo local, ou seja, usando um ficheiro de 2Gb armazenado no sistema operativo Linux, verificou-se que o tempo de execução foi o dobro do tempo de quando o processo foi efetuado recorrendo ao sistema de ficheiros distribuído HDFS. Na comparação entre a execução da aplicação *WordCoun*t implementada com o Pig ou diretamente no Hadoop (python ou java), não se encontraram grandes diferenças de tempo de execução. O Pig, pelo facto de usar uma linguagem de alto nível, que permite criação de funções para consultas mais elaboradas, afigura-se uma ferramenta muito útil no tratamento de grandes volumes de dados em sistemas de computação paralela, especialmente para os menos experientes em programação.

5.3 ANÁLISE E VISUALIZAÇÃO DE DADOS

O tratamento analítico dos resultados obtidos nos processos de *mapreduce* realizados no Hadoop e das consultas de *streaming*, efetuadas no cluster Kafka e processadas pelo Spark Streaming, descritos nos pontos anteriores, foi realizado com o software Tableau Desktop.[23]

O painel da Figura 14, representa o resultado do teste executado "em batch" com a aplicação *WordCount* num ficheiro (grande15G) com cerca de 2.9 Biliões de palavras no total, realizado em *mapreduce* no Hadoop. O gráfico de dispersão mostra o aglomerado de palavras encontradas, destacando as que aparecem em maior número, enquanto o gráfico Top10, representa numericamente as dez palavras mais encontradas no documento. Como não foi feito qualquer tipo de pré ou pós processamento, o top 10 é composto por "*stop words*" que normalmente são removidas por ser esperado que sejam as mais frequentes.

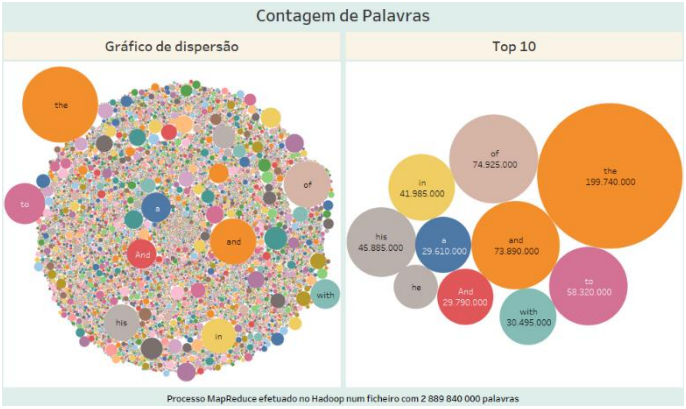


Figura 14: Painel de contagem de palavras

Na simulação de processamento em *streaming*, foram processados 105RDDs (RDD - *resilient distributed dataset*), correspondendo cada RDD ao conjunto de eventos recebidos em cada *time-box* de 5 segundos, pelo Spark Streaming, tendo sido recebidos 15752 registos com dados de saúde obtidos do tópico "FluxoPorConsulta" do cluster Kafka.

O gráfico da Figura 15, mostra a variação do número de registos recebidos por cada RDD, que variam entre 30 e 250. Esta flutuação foi configurada no código do gerador de registos de forma a simular uma situação o mais próxima possível da realidade (não no número, mas na variação ao longo do tempo).

[23] https://www.tableau.com/.

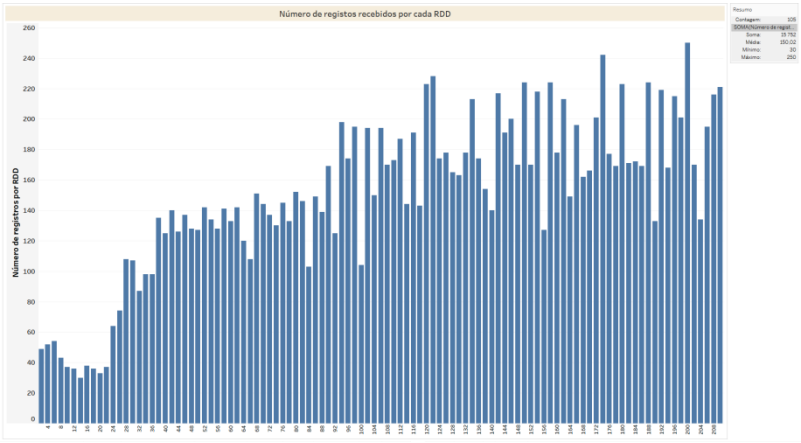


Figura 15: Número de registos por RDD

Numa ótica de demonstração de exemplos de análise efetiva dos dados relativos a saúde, recebidos e processados pelo Spark Streaming, foram efetuados dois painéis que permitem traduzir graficamente alguma informação referente ao primeiro semestre de 2018, que poderia ser útil para a gestão de uma unidade de saúde. O painel apresentado na Figura 16 é constituído por dois gráficos que indicam os tempos médios de espera por consulta dos utentes, desde o momento em que dão entrada no sistema até ao final do ato médico. No primeiro gráfico pode verificar-se que o tempo médio de espera do utente em todas as especialidades, antes de ser consultado é de 56,9min, variando entre um mínimo 47min em Ginecologia e um máximo de 63min em Cardiologia. No segundo gráfico pode verificar-se que o tempo médio de atendimento médico em todas as especialidades é de 18,8min que varia entre um mínimo de 17min em cardiologia e um máximo de 26min em Pediatria. Os dois gráficos permitem realizar filtros por mês e por tipo de especialidade.

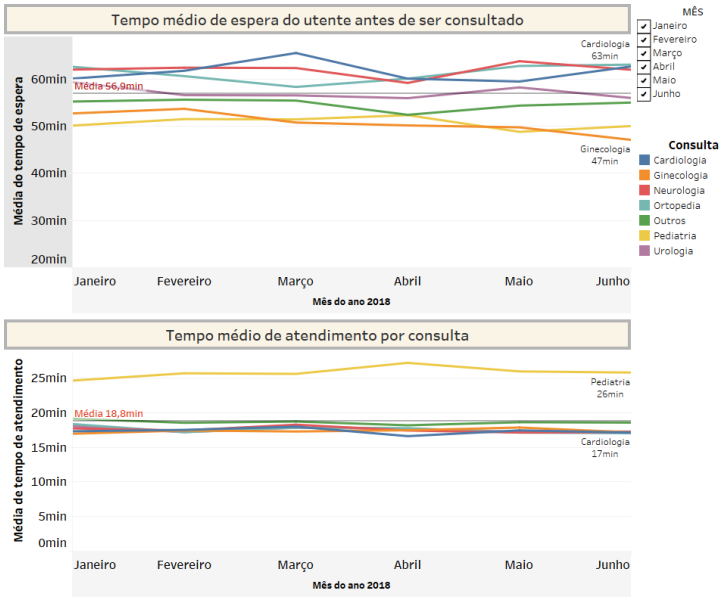
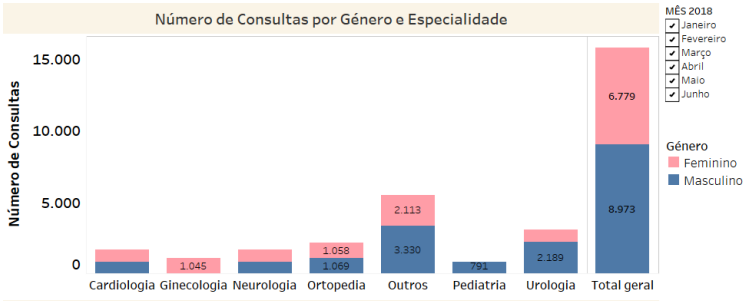


Figura 16: Tempos médios de espera e atendimento dos utentes de saúde

O painel apresentado na Figura 17 é constituído por dois gráficos que mostram o número de consultas por género e grupo etário realizadas em cada especialidade. O primeiro gráfico representa o número de consultas por género realizadas em cada especialidade, nos primeiros seis meses do ano de 2018. Este gráfico a segmentação dos dados por mês. O segundo gráfico representa o número de consultas por especialidade dentro de cada faixa etária, e permite filtros por mês e escalão etário. Cada um dos gráficos detêm ainda informação sobre os totais por cada género e escalão etário, respetivamente.



Consultas por Escalão Etário

Escalão	Cardiologia	Ginecologia	Neurologia	Ortopedia	Outros	Pediatria	Urologia	Total geral
0-2	240		252	302	598	264	241	1.897
3-10	221		256	304	597	280	236	1.894
11-17	221	192	194	310	1.078	247	521	2.763
18-29	239	214	232	297	778		503	2.263
30-49	230	215	266	301	836		529	2.377
50-65	250	216	215	316	773		540	2.310
>66	245	208	226	297	783		489	2.248

Figura 17: Número de Consultas por género, escalão etário e especialidade

A análise de dados depende essencialmente das questões que as organizações querem ver respondidas para melhorarem as suas operações diárias, ou para apoiarem as suas decisões de gestão. No processo analítico efetuado anteriormente, não se pretende responder a questões concretas das atividades de uma organização, mas sim mostrar a fase final do processo de implementação da infraestrutura criada para o tratamento de *Big Data*, tanto em *batch* como em *streaming*, que é a visualização da informação.

Os exemplos retratados nos painéis anteriores, são meramente indicativos das possibilidades oferecidas pelos processos analíticos, tanto em dados históricos (para processamento em *batch*) como que chegam às organizações em tempo real (*streaming*). No âmbito deste projeto exploratório, foram apenas testados dois cenários de modo a validar o correto funcionamento da infraestrutura implementada.
Sendo o objetivo final de qualquer processo de tratamento de dados, a obtenção, transformação e disponibilização de informação atempada e fidedigna paras as organizações. Pretende-se, demonstrar a possibilidade de execução de análise descritiva, de diagnóstico ou preditiva (não incluída neste trabalho), através de ferramentas que permitem a criação de relatórios e painéis interativos que retratam informação em tempo útil, tanto histórica, como à medida que vai sendo recebida pela organização.

6. CONCLUSÃO

Este artigo apresenta um projeto de implementação de um laboratório de *Big Data*, em contexto académico, que pretende ser um recurso valioso para o ensino/exploração das tecnologias e/ou métodos de processamento de *Big Data* com os alunos de licenciatura e mestrado da instituição de ensino. Pretende-se também que seja um recurso que permita expandir a investigação nesta área tão relevante para as organizações no contexto atual.

Este artigo descreve como foi implementado o laboratório, descrevendo a arquitetura concebida e as tecnologias adotadas. Para contextualizar o leitor, foi incluído um enquadramento teórico que permite caracterizar o contexto e os desafios do *Big Data* e distinguir processamento de dados históricos (*batch*) e de dados em movimento/fluxo (*streaming*), as respetivas tecnologias e como as mesmas funcionam. É descrita a solução implementada e apresentados dois cenários de avaliação da arquitetura, para processamento em *batch* e *streaming*, que mostram como o recurso pode ser aproveitado para projetos de *data analytics* nas duas vertentes. Foi criado um cenário simulado de processamento em *streaming*, por ser considerado o mais complexo, uma vez que envolve a ligação de diversas máquinas e tecnologias, de modo a demonstrar a capacidade de processar os dados em tempo-real, algo que é cada vez mais ambicionado pelas organizações.

O projeto que deu origem a este artigo pretendia apenas implementar a arquitetura e demonstrar a sua funcionalidade com o propósito de, no futuro, angariar projetos de *Big Data* (dimensionados para os recursos disponíveis) que esses sim permitam expandir o conhecimento das tecnologias, nomeadamente na otimização

das configurações para dar resposta a cada desafio e criar *pipelines* de processamento de dados, integração de outros algoritmos e/ou de análises concretas dos dados recebidos/armazenados.

REFERÊNCIAS

Apache Hadoop. (2019). *http://hadoop.apache.org/*. Obtido em 30 de outubro de 2019

Apache Kafka. (2017). *http://kafka.apache.org/*. Obtido em 30 de outubro de 2019

Baljak, V., Ljubovic, A., Michel, J., Montgomery, M., & Salaway, R. (2018). A scalable realtime analytics pipeline and storage architecture for physiological monitoring big data. , 9-10, 275-286. *Smart Health, 9-10*, 275-286. doi:doi: https://doi.org/10.1016/j.smhl.2018.07.013

Caldeira, C. P. (2012). *Data Warehousing - Conceitos e Modelos.* Lisboa: Edições Silabo, Lda.

Confluent, Inc. (2018). *https://www.confluent.io*. Obtido em 1 de novembro de 2019

Du, D. (2015). *Apache Hive Essentials.* Birmingham, UK: Packt Publishing Ltd.

Dunning, T., & Friedman, E. (2016). *Streaming Architecture - New Designs Using Apache Kafka and MapR Streams.* Sebastopol, CA.: O'Reilly Media, Inc.

Erl, T., Khattak, W., & Buhler, P. (2016). *Big Data Fundamentals - Concepts, Drivers &Techniques.* Indiana: Arcitura Education Inc.

Estrada, R. (2018). *From big data to fast data - Designing application architectures for real-time decisions.* Obtido em 01 de novembro de 2019, de https://www.oreilly.com/ideas/from-big-data-to-fast-data

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management, 35*(2), 137-144. doi:https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Garg, N. (2015). *Learning Apache Kafka Second Edition.* Birmingham B3 2PB, UK.: Packt Publishing Ltd.

Hurwitz, J., Nugent, A., Halper, F., & Kaufman, M. (2013). *Big Data For Dummies.* Hoboken, New Jersey: John Wiley & Sons, Inc.

Krishnan, K. (2013). *Data Warehousing in the Age of Big Data.* Waltham, USA: Elsevier Inc.

Marr, B. (2015). *Big data : using smart big data, analytics and metrics to make better decisions and improve performance.* Chichester,West Sussex, United Kingdom: JohnWiley & Sons Ltd,.

Narkhede, N., Shapira, G., & Palino, T. (2017). *Kafka: The Definitive Guide.* Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc.

Peffers, K., Tuunanen, T., Rothenberger , M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, Winter 2007–8, Vol. 24, No. 3, pp. 45–77.

Sawant, N., & Shah, H. (2014). *Big Data Application Architecture Q&A - A Problem-Solution Approach.* Apress.

Wampler, D. (2018). *Fast Data Architectures for Streaming Applications - Second Edition.* Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc.