

RICARDO MIGUEL BRASIL SOARES

Relatório de Trabalho Final de Curso

Processamento e análise de grandes volumes de dados

Orientador: Professor Rui Ribeiro

Unidade Curricular: Trabalho Final de Curso

Universidade Lusófona de Humanidades e Tecnologias

Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação

Lisboa

2014

Índice

Resumo	4
Abstract	5
Capítulo I - Introdução	6
Capítulo II - Enquadramento Teórico	8
2.1 - Internet of Things	9
2.2 - Big Data	11
2.3 - Hadoop	12
2.3.1 - História	12
2.3.2 - O que é ?	13
2.3.3 - Arquitectura	15
2.4 - Spark	19
2.4.1 - Resilient Distributed Datasets	20
2.4.2 - Módulos	21
Capítulo III – Método	22
3.1 - Prova de conceito I	25
3.2 - Prova de conceito II	26
Capítulo IV – Resultados	27
Capítulo V - Conclusões e trabalho futuro	31
Bibliografia	33
Anexos	34
Apêndices	37
Manual técnico	51
Glossário	57

Índice de Figuras e Gráficos

Figura 2.3.1 – Cluster Hadoop.....	13
Figura 2.3.2 – Arquitetura HDFS.....	16
Figura 2.3.3 – Replicação de blocos.....	17
Figura 2.3.4 – Fluxo MapReduce.....	18
Figura 2.4.1 – RDD Lineage.....	20
Figura 2.4.2 – Componentes Apache Spark.....	21
Figura 3.1 – Fluxo de análise.....	23
Figura 3.2 – Cluster de demonstração.....	24
Gráfico 4.1 – Velocidade de processamento.....	28

Resumo

Com a proliferação das tecnologias e comunicações *wireless*, e a capacidade que estas têm de comunicar entre si e de gerar informação, torna-se essencial a existência de ferramentas que permitam gerir a crescente quantidade de informação gerada.

Com esta ideia, a realização deste trabalho passa pela investigação e implementação de algumas dessas tecnologias. Os critérios utilizados na escolha das ferramentas foram a sua atualidade bem como a adoção por parte das empresas líderes nesta área. As plataformas escolhidas para o desenvolvimento deste projeto foram então o Apache Hadoop e o Apache Spark.

Ao longo deste trabalho serão implementadas ambas as tecnologias num *cluster* de demonstração e serão exploradas as suas funcionalidades através da análise de dois tipos de ficheiros de log.

Pôde-se verificar que ambas as plataformas demonstram características essenciais na gestão de grandes volumes de dados como a versatilidade, escalabilidade, performance, entre outras. Este trabalho é ainda suscetível de continuação pois não foram investigadas todas as potencialidades das plataformas Hadoop e Spark e apenas foi posto em prática num contexto de desenvolvimento.

Palavras-Chave: Internet of Things, Big Data, Hadoop, Spark.

Abstract

With the technological proliferation and evolution of the wireless communications and their ability to communicate with each other and generate data, it becomes necessary to have tools that are capable of manage the continuously growing amounts of generated data.

With this in mind, the development of this paper goes through the investigation and implementation of some of those tools. The criteria for choosing which technologies to use were based on their actuality as well as their adoption by the leading organizations in this field. So, the chosen platforms for this project development were Apache Hadoop and Apache Spark.

During this project both platforms will be implemented in a demonstration cluster and features will be explored through the analysis of two types of log files.

As a result, both platforms demonstrated crucial characteristics in the management of big data like versatility, scalability, performance, among others. This project is susceptible of being continued as not every feature of Hadoop and Spark was investigated and it was only tested in a development context.

Keywords: Internet of Things, Big Data, Hadoop, Spark.

Capítulo I - Introdução

Estima-se que a cada dois dias é gerada uma quantidade de informação igual à que existia desde a criação da civilização até ao ano 2003 [1]. A evolução tecnológica e da própria Internet contribuem em grande parte para o constante aumento da informação bem como a velocidade com que esta é gerada. Tecnologias como a *cloud*, dispositivos móveis, redes sociais ou a Internet of Things obrigam a que se encontrem formas de armazenar e processar as quantidades de dados gerados [2]. Os sistemas de gestão de base de dados e as ferramentas de processamento normalmente utilizadas não estão preparados para gerir estes novos fluxos de informação, que para além da sua dimensão, estão muitas vezes numa forma não-estruturada.

Pretende-se então com a elaboração deste trabalho tomar conhecimento acerca das tecnologias existentes que consigam gerir, armazenar e processar o que se pode chamar de Big Data. Em particular, irão ser exploradas as plataformas Apache Hadoop e Apache Spark. A escolha destas tecnologias deve-se à sua atualidade bem como ao desempenho e capacidade que oferecem na gestão de grandes volumes de dados. A plataforma Hadoop será utilizada como armazenamento distribuído aproveitando as suas características de escalabilidade e tolerância a falhas enquanto a plataforma Spark será responsável pelo processamento permitindo a execução de análises interativas sobre os dados ou o processamento em tempo-real.

Estas tecnologias serão implementadas através de duas provas de conceito. A primeira incide na análise de logs gerados por pórticos de cobrança de portagens os quais registam o correto funcionamento ou as falhas detetadas nos pórticos. É objetivo desta análise perceber quais os tipos de erros mais comuns bem como os pórticos mais afetados. A segunda prova de conceito é baseada na análise de logs gerados por um servidor web a partir dos quais se pretende distinguir os vários tipos de erros e verificar se existe algum padrão com base na frequência com que ocorrem.

Capítulo II - Enquadramento Teórico

2.1 - Internet of Things

Internet of Things é um paradigma que tem vindo a ter um foco crescente onde dispositivos eletrónicos, pessoas ou objetos comunicam entre si, ou com um serviço, através de uma rede.

Tecnologias como a identificação por radiofrequência (RFID), comunicação *wireless*, localização em tempo real e redes de sensores estão cada vez mais presentes tornando a Internet of Things numa realidade.

O termo Things refere-se a qualquer objeto que comunique com outro objeto ou com um serviço. Os objetos com que interagimos todos os dias serão capazes de comunicar entre si trazendo possibilidades que até agora nos são desconhecidas. Indiscutivelmente, o ponto-chave da Internet of Things é o impacto que esta terá no nosso quotidiano. Do ponto de vista de um uso pessoal, os efeitos óbvios da IoT serão notados tanto no aspeto profissional como doméstico. Neste contexto, domótica, e-health, ensino especializado são alguns exemplos de possíveis aplicações nas quais a IoT irá desempenhar um papel importante num futuro próximo. Numa perspectiva de negócio, a IoT irá ter um impacto em áreas como a automação, logística, gestão de negócios e/ou processos, transporte de pessoas e bens.

Prevê-se que em 2025 a maior parte dos objetos que utilizamos irão estar de alguma forma ligados à Internet. Criar-se-ão inúmeras possibilidades partindo da ideia que a procura do público associada à evolução tecnológica poderá levar a uma propagação generalizada da IoT que pode, tal como a Internet, contribuir para o desenvolvimento económico e social. As ameaças associadas a uma adoção generalizada são também um dos fatores a ter em conta uma vez que a partir do momento em que os objetos do dia-a-dia se tornem centros de informação, os perigos existentes hoje na Internet irão estar disseminados a uma escala global.

Alguns exemplos de possibilidades que a Internet of Things nos pode trazer são: câmaras montadas em prateleiras de supermercado que detetam os focos de atenção dos consumidores em tempo real, sensores embutidos em máquinas de produção que monitorizam o estado atual das peças enviando relatórios e avisando aquando da necessidade de manutenção, um implante que monitoriza o ritmo cardíaco e deteta a mais pequena alteração permitindo a alguém antecipar um problema cardíaco, sensores instalados em parqueamentos para automóveis que informam os condutores da existência de lugares disponíveis, um prédio com sensores de iluminação e temperatura que gerem as luzes e os ares condicionados de uma forma eficiente.

A Internet of Things tem o potencial de mudar as nossas vidas afetando-nos não só a nível pessoal mas também na indústria, no sector energético, nos bens de consumo, na saúde, nos transportes, ou seja, todas as áreas no nosso quotidiano.

2.2 - Big Data

A sociedade em que vivemos faz um uso cada vez mais intensivo das tecnologias e como resultado, as organizações produzem e armazenam grandes volumes de dados. Gerir e aproveitar a informação produzida representa um desafio e é chave para o sucesso organizacional. Soluções que analisem a informação são importantes pois permitem obter conhecimento não só sobre a informação gerada internamente mas também acerca de informação disponível na Internet.

A informação está a mudar a forma como as empresas trabalham e está a criar uma cultura em que esta é cada vez mais valorizada. O conhecimento obtido através da sua análise permite tomar melhores decisões, criar um maior envolvimento com o cliente e capitalizar novas formas de rendimento.

O termo Big Data refere-se a dados gerados em grande quantidade, a grande velocidade e com grande variedade que exigem formas inovadoras, de custo eficiente, que permitam processar os dados de forma a obter uma visão e um conhecimento preciso sobre a informação bem como a tomar decisões mais acertadas com base nos dados analisados.

Apesar da recente popularidade que o conceito Big Data obteve, armazenar e analisar grandes volumes de informação representa ainda um esforço e um desafio. O aumento contínuo da quantidade de informação gerada cria novos desafios na área das Tecnologias de Informação: escalabilidade – a necessidade de armazenar toda a informação implica ter uma infraestrutura flexível e que seja facilmente ampliável; performance – a velocidade a que a informação é gerada obriga a que seja processada de uma forma rápida e eficiente; diversidade – a informação chega nas mais variadas formas e é necessário adaptar a tecnologia aos dados e não o inverso; utilização – armazenar toda a informação é também um desafio no aspeto em que é importante saber identificar a utilidade da informação e como usá-la.

2.3 - Hadoop

2.3.1 - História

O Hadoop nasceu em 2004 com o nome Nutch Distributed Filesystem (NFDS), uma implementação open-source com base num *paper* publicado pela Google acerca do seu sistema de ficheiros distribuído, GFS (Google File System) [3] com o objetivo inicial de ser um motor de busca. Ainda em 2004, a Google introduziu ao mundo o MapReduce num novo paper e em 2005 o projeto Nutch já contava com uma implementação funcional do MapReduce.

As possibilidades que o NDFS e o MapReduce permitiam iam além do objetivo inicial do projeto o que levou à formação de um novo projeto em 2006, desta vez já com o nome de Hadoop.

Em 2008, o Hadoop torna-se num projeto top-level da Apache confirmando o seu sucesso e apoiado por uma ativa comunidade. Nesta altura o Hadoop já era utilizado por grandes empresas como Yahoo!, Facebook, New York Times, entre outras.

2.3.2 - O que é?

O software Apache Hadoop é uma framework que permite o armazenamento e processamento distribuído de grandes volumes de dados através de clusters de computadores. Foi desenvolvido para ser altamente escalável e oferecer um elevado nível de fiabilidade – ao invés de depender do hardware para oferecer alta-disponibilidade tem a capacidade de detetar e gerir falhas na camada aplicacional [4].

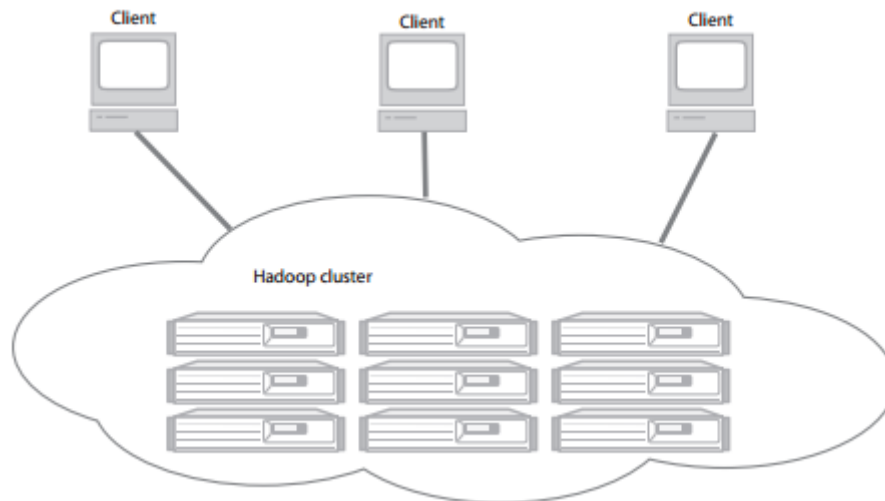


Figura 2.3.1 - Cluster Hadoop

O projeto Apache Hadoop inclui os seguintes módulos:

Hadoop Common: contém a estrutura principal do Hadoop bem como várias aplicações e bibliotecas que dão suporte aos restantes módulos. É neste subprojecto que estão as interfaces que permitem a integração com outros sistemas de ficheiros, como o Amazon S3.

Hadoop Distributed File System (HDFS): Sistema de ficheiros distribuído que permite a leitura e escrita de grandes volumes de dados.

Hadoop YARN: Framework para agendamento de tarefas e gestão de recursos do *cluster*.

Hadoop MapReduce: Modelo de programação desenvolvido para o processamento de grandes volumes de dados.

O Hadoop distingue-se por algumas características tais como acessibilidade – capacidade de correr em *hardware* de baixo custo ou em serviços presentes na *cloud* como o Amazon's Elastic Compute Cloud, robustez – desenvolvido com o pressuposto de existirem falhas frequentes de hardware, pelo que é importante gerir facilmente essas falhas, escalabilidade – permite facilmente adicionar máquinas a um cluster de modo a acompanhar o aumento do volume de dados, simplicidade – proporciona aos utilizadores desenvolverem rapidamente código através da abstração do processamento paralelo.

2.3.3 - Arquitetura

HDFS

O Hadoop Distributed File System (HDFS) é um sistema de ficheiros desenvolvido para correr em *hardware* de baixo custo. Tem várias semelhanças com os sistemas de ficheiros distribuídos conhecidos, no entanto, as diferenças que o distinguem são significativas.

NameNode e DataNode

O HDFS é construído com base numa arquitetura *master / slave*. Um *cluster* HDFS é composto por um único NameNode que gere o *namespace* do sistema e regula os acessos aos ficheiros pelas aplicações clientes. Além do NameNode existem vários DataNodes que armazenam os dados. Internamente, os ficheiros são divididos em um ou mais blocos que são por sua vez guardados em diversos DataNodes.

O NameNode executa operações do *namespace* do sistema de ficheiros como abrir, fechar e renomear ficheiros ou pastas. É o NameNode quem também determina o mapeamento dos blocos para os DataNodes.

Os DataNodes são responsáveis por realizar os pedidos de leitura e escrita feitos pelas aplicações cliente. Realizam também operações de criação, remoção e replicação de blocos.

Uma arquitetura típica é composta por uma máquina dedicada onde corre o NameNode e cada uma das restantes máquinas do *cluster* irá correr uma instância de um DataNode.

A existência de um único NameNode simplifica a arquitetura do sistema na medida em que o NameNode representa um repositório de meta-dados do HDFS. Além disso, o sistema foi desenhado de forma a que os dados a serem armazenados nunca passem pelo NameNode [5].

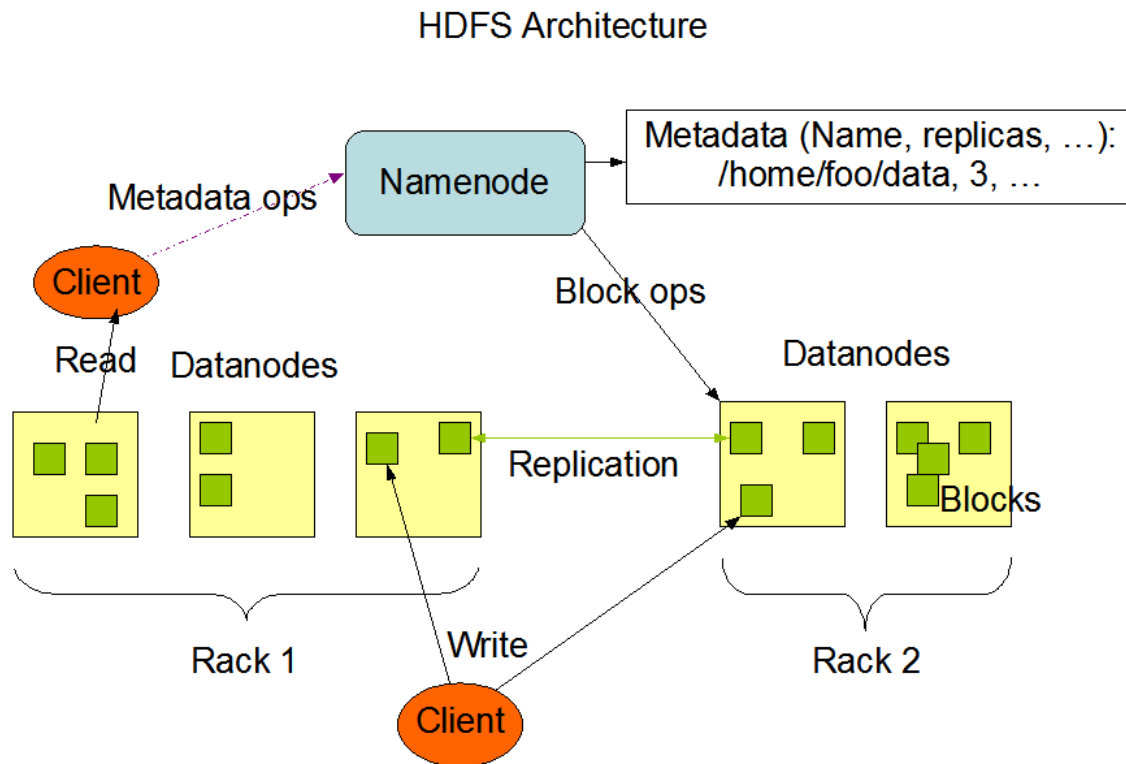


Figura 2.3.2 - Arquitetura HDFS

Sistema de ficheiros

O HDFS suporta uma organização hierárquica tradicional. Uma aplicação cliente pode criar pastas e aí armazenar ficheiros ou outras pastas. A hierarquia do *namespace* é semelhante a outros sistemas de ficheiros existentes; permite a criação e remoção de ficheiros, mover ficheiros de uma pasta para outra ou renomear um ficheiro.

O NameNode é responsável por manter o *namespace* do sistema de ficheiros. Qualquer alteração ao *namespace* do sistema de ficheiros ou às suas propriedades é registada pelo NameNode.

O HDFS permite às aplicações especificarem o número de réplicas a serem criadas para um ficheiro. O número de réplicas de um ficheiro é chamado *replication factor* e é também registado pelo NameNode [5].

Replicação

O HDFS foi desenvolvido para armazenar ficheiros de grandes dimensões divididos por várias máquinas num *cluster*. Cada ficheiro é guardado como uma sequência de blocos todos do mesmo tamanho, exceto o último.

O tamanho de cada bloco e o seu fator de replicação são configuráveis pela aplicação que os cria. O fator de replicação é especificado aquando da criação do ficheiro mas pode ser alterado posteriormente.

O NameNode é quem toma todas as decisões relacionadas com a replicação de blocos. Periodicamente recebe um *heartbeat* e um *blockreport* de cada um dos DataNodes conseguindo assim saber se um DataNode está a funcionar corretamente, através do *heartbeat*, e obter uma lista de todos os blocos existentes num DataNode, através do *blockreport* [5].

Block Replication

Namenode (Filename, numReplicas, block-ids, ...)
 /users/sameerp/data/part-0, r:2, {1,3}, ...
 /users/sameerp/data/part-1, r:3, {2,4,5}, ...

Datanodes

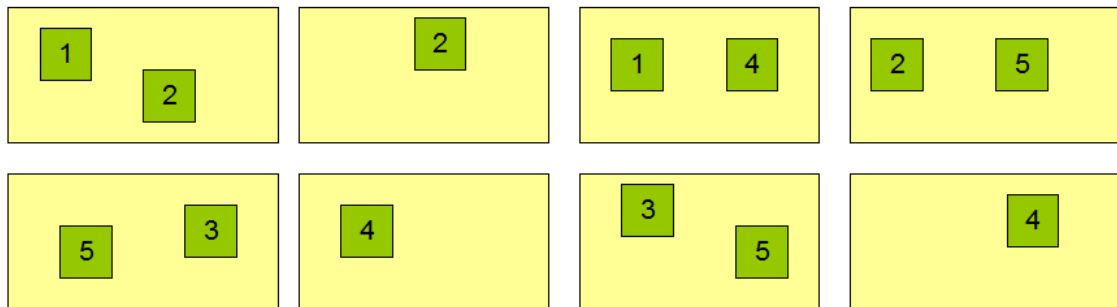


Figura 2.3.3 - Replicação de blocos

MapReduce

O MapReduce é um modelo de programação baseado no conceito de programação paralela desenvolvido para o processamento de grandes volumes de dados. Com base na noção de programação paralela, o processamento é dividido em n sub-processamentos que são executados concorrentemente. Este modelo tem as suas origens nas linguagens de programação funcional, como o Lisp. [6]

Uma tarefa MapReduce divide normalmente o *input* em blocos mais pequenos que são processados paralelamente pela função *map*. O *output* obtido pela função *map* é ordenado e transformado em *input* da função *reduce*.

O paralelismo no MapReduce é obtido através de um processo de *split / sort / merge / join* que pode ser descrito da seguinte forma:

- É iniciado um programa sob a forma de uma tarefa MapReduce que utiliza como *input* o conjunto de dados definido.
- O *input* é decomposto em blocos e cada bloco ou conjunto de blocos é enviado para um ou mais nós do cluster para ser processado pela função *map*. Este processo conclui a fase *split*.
- Quando um nó recebe os dados processa-os com a função *map* e gera um conjunto de pares [chave, valor]. O conjunto de pares gerado é então ordenado e enviado para a função *reduce*.
- A função *reduce* agrega todos os pares com a mesma chave e processa-os de acordo com o programado pelo utilizador. Este processo é chamado *merge*.
- Finalmente após se juntarem os *outputs* das funções *reduce* (fase *join*) o processamento termina.

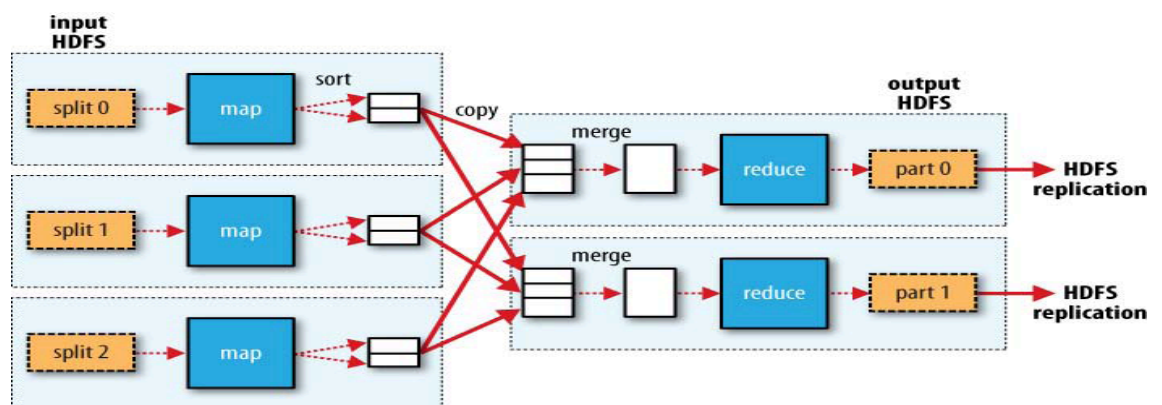


Figura 2.3.4 - Fluxo MapReduce

2.4 - Spark

O projeto Spark começou como um projeto de investigação na Universidade de Berkeley em 2009 com o objeto de criar um modelo de programação mais amplo que o MapReduce mantendo, no entanto, a alta tolerância a falhas. O modelo MapReduce é ineficiente para alguns tipos de aplicações analíticas comuns como algoritmos iterativos, *data mining interativo*, *stream processing*, entre outros [7].

A razão pela qual o MapReduce se torna menos eficiente nas aplicações mencionadas deve-se ao facto de ser baseado num fluxo acíclico: uma aplicação tem de executar várias tarefas, cada uma das quais lê os dados a partir de um dispositivo de armazenamento (sistema de ficheiros distribuído) e volta a escrevê-los de volta para o mesmo armazenamento o que envolve um custo grande em operações *read/write*.

Pode dizer-se que é uma evolução do modelo de programação MapReduce na medida em que se adequa a mais tipos de processamento. Sendo a velocidade um aspeto importante no processamento de grandes quantidades de dados, o Spark utiliza como um dos seus principais recursos o processamento em memória [8].

De uma maneira geral, o Spark é desenvolvido para acumular funções que antes requeriam vários sistemas distribuídos e que requerem um tipo de processamento interativo e em tempo real.

2.4.1 - Resilient Distributed Datasets

Um RDD é um conjunto de registos distribuídos por varias máquinas, de modo *read-only*, que só pode ser criado através de operações determinísticas em dados presentes num dispositivo de armazenamento ou em outros RDD [9].

Os RDD suportam dois tipos de operações: as transformações, que criam um *dataset* a partir de um já existente e as ações, que retornam um valor para a aplicação depois de processarem um RDD. Por exemplo, uma transformação *map* percorre cada registo de um *dataset*, aplica-lhe uma função e retorna um novo RDD. Por outro lado, a ação *reduce* agrega todos os elementos de um RDD através de uma função e retorna o resultado para a aplicação [10].

Todas as transformações no Spark são *lazy* na medida em que não são executadas de imediato. São memorizadas e só são processadas quando requeridas por uma ação.

RDD – Persistência

Uma das características mais importantes dos RDD é a persistência (ou *caching*) de um *dataset* em memória entre operações. Quando um RDD é guardado em memória, cada máquina do *cluster* guarda uma parte desse RDD que irá processar em memória e reutilizar em outras operações sobre esse RDD. Isto permite que futuras operações sejam executadas rapidamente. O *caching* de RDDs em memória é um fator essencial para a utilização de algoritmos iterativos.

Um RDD pode ser armazenado em memória através das operações *.persist()* ou *.cache()*. Os RDD guardados em memória são tolerantes a falhas – se alguma parte do RDD for corrompida, esta é automaticamente recriada através das transformações que a geraram [10].

RDD – Tolerância a falhas

É possível recuperar um RDD corrompido, ou parte de um RDD corrompido, através do mecanismo de *lineage* (linhagem) que é uma característica dos RDD e do Apache Spark. Cada vez que um RDD é gerado, é guardado um registo das transformações através das quais foi gerado. Desta forma, ao existir uma falha num RDD, a transformação ou transformações anteriores são reexecutadas de forma a recuperar o RDD corrompido [9].

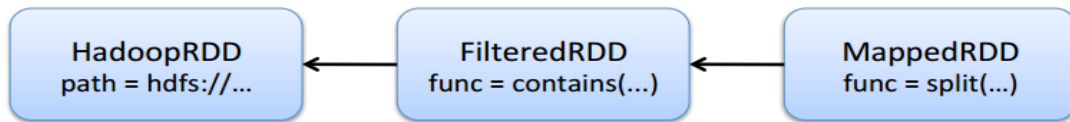


Figura 2.4.1 – RDD lineage

2.4.2 - Módulos

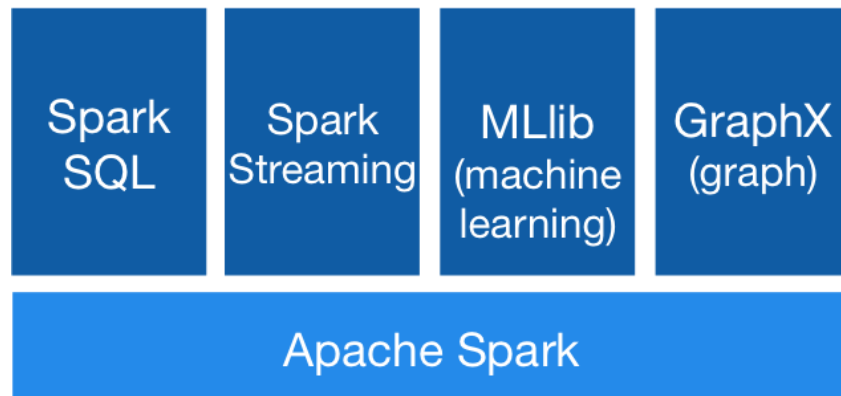


Figura 2.4.2 - Componentes Apache Spark

Spark Core

Contém as funcionalidades básicas do Spark, incluindo componentes para o agendamento de tarefas, gestão de memória, recuperação de falhas, interação com sistemas de armazenamento, entre outros.

Spark SQL

Proporciona uma interação com o Spark através de SQL. O Spark SQL permite representar tabelas como RDDs bem como transformar *queries* SQL em operações Spark.

Spark Streaming

Componente do projeto que permite processar *streams* contínuos de dados como ficheiros log gerados por servidores web, *feeds* de mensagens, entre outros.

MLib

Biblioteca que contém vários algoritmos de *machine learning* como classificação binária, regressão, filtragem colaborativa.

GraphX

Biblioteca que fornece uma API para manipulação de grafos (ex. grafo do círculos de amigos numa rede social) e processamento paralelo de grafos.

Capítulo III – Método

O sistema a implementar será composto por quatro componentes principais:

1. Plataforma Apache Hadoop
2. Plataforma Apache Spark
3. Base de dados MySQL
4. Interface Web

Numa primeira fase, os dados serão armazenados no sistema de ficheiros distribuído do Hadoop (HDFS) aproveitando a sua replicação de blocos tornando assim o armazenamento dos dados tolerante a falhas. O componente responsável pelo processamento dos dados irá ser o Spark devido a sua rapidez bem como à capacidade de recuperar blocos de dados que possam ter sido corrompidos durante um processamento. Após os dados terem sido processados serão armazenados, através de um processo ETL, numa base de dados MySQL de modo a poderem ser posteriormente visualizados num interface Web.

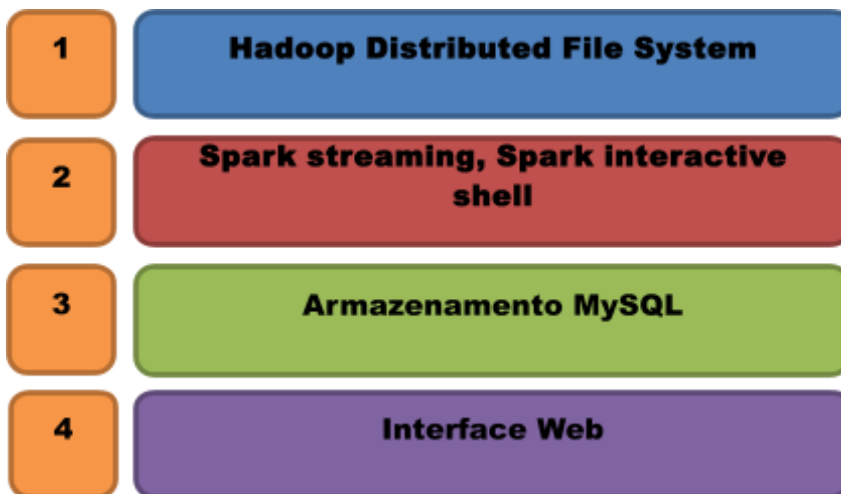


Figura 3.1 – Fluxo de análise

De modo a desenvolver as provas de conceito propostas, e dada a limitação de poder ter um *cluster* com várias máquinas o sistema será implementado num *cluster* de demonstração de apenas duas máquinas. A primeira desempenhará um papel de *master* / *slave* em ambas as plataformas enquanto a segunda será apenas *slave*. O *cluster* é composto então da seguinte forma:

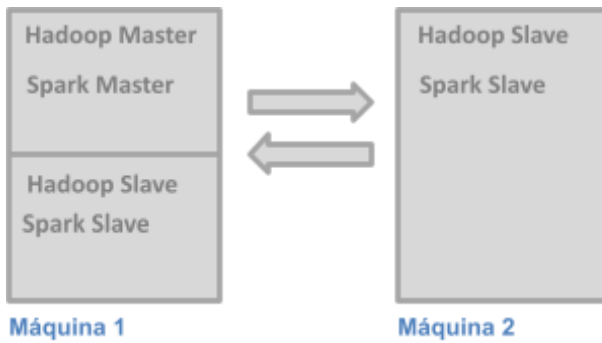


Figura 3.2 – Cluster de demonstração

3.1 - Prova de conceito I

Esta prova de conceito tem como objetivo a análise de ficheiros de log gerados por pórticos de cobrança de portagens (cf. Anexo I). Pretende-se com esta análise conseguir perceber quais os tipos de erros mais comuns bem como os pórticos que mais são afetados.

Uma vez que os logs estão constantemente a ser gerados será utilizado o modo Spark Streaming de modo a analisar os dados à medida que vão chegando e a possibilitar a visualização em tempo real através do interface Web.

A aplicação desenvolvida monitoriza uma pasta do HDFS (cf. Apêndice I, linha 48) e sempre que existe um ficheiro novo, processa-o. Neste caso específico, a aplicação filtra as linhas do ficheiro log descartando as que não contêm qualquer mensagem de erro (cf. Apêndice I, linha 49). Após esta filtragem, a aplicação percorre os registos do RDD gerado que contém apenas as linhas com mensagens de erro e guarda-os numa base de dados MySQL (cf. Apêndice I, linha 51 a 67).

A partir do momento em que existam registos na base de dados MySQL, os mesmos poderão ser consultados na interface Web onde também se poderá observar algumas estatísticas sobre os mesmos (cf. Apêndice II).

Através do interface web podem ser consultados os registos que estão na base de dados. Por defeito mostra os 20 registos mais recentes (cf. Apêndice X). A partir deste interface é possível filtrar os registos criados entre duas datas (cf. Apêndice XII) bem como filtrar os resultados a partir de *keywords*, por exemplo escrevendo 4108 na *searchbox* da coluna Hostname apenas serão apresentados os resultados cujos Hostname contenham a expressão 4108 (cf. Apêndice XIII) ou escrevendo 00:01 na *searchbox* da coluna Timestamp apenas irá apresentar os erros ocorridos à meia-noite e um minuto (cf. Apêndice XIV). São ainda mostradas algumas estatísticas como a quantidade de erros reportados por cada pórtico, a quantidade de vezes que um dado erro foi reportado e quais os componentes mais afetados calculadas através de consultas SQL à base de dados (cf. Apêndices VII, VIII e IX).

3.2 - Prova de conceito II

Pretende-se com a segunda prova de conceito explorar a capacidade de análise interativa da plataforma Spark através da execução de *queries ad-hoc*. Para esta prova de conceito serão utilizados logs de um servidor web. Ao contrário da primeira prova de conceito, onde os logs já tinham sido estruturados de forma a serem mais facilmente analisados, estes estão numa forma menos estruturada, tal como gerados pela aplicação (cf. Anexo2).

Através desta análise interativa é possível obter rapidamente algumas informações como contagem de linhas, visualizar uma amostra do *dataset*, filtrar o *dataset* de acordo com o que se pretende, entre outras.

Para esta prova de conceito irá ser carregado um ficheiro *.txt* localizado no disco local da máquina (cf. Apêndice IX, linha 1) sobre o qual se pode à partida efetuar a ação *.count()* (retorna o número de linhas do ficheiro) ou *.take(n)* (retorna uma amostra do *dataset*) onde *n* representa o número de linhas da amostra.

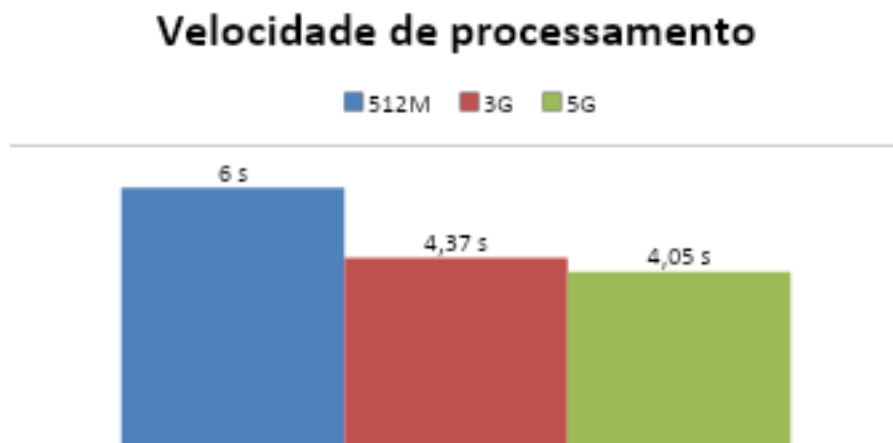
Fazendo também uso das transformações *.filter()* e *.map()* será possível perceber quais os erros registados no ficheiro para um determinado tempo (cf. Apêndice IX, linha 2 a 5).

Capítulo IV – Resultados

Os resultados obtidos através do desenvolvimento deste projeto foram satisfatórios. Ambas as plataformas utilizadas mostraram ser adequadas ao que se propunha.

Na análise dos logs de pórticos de cobrança de portagens foi possível desenvolver um sistema completo, de cariz empresarial, e ficou demonstrado a capacidade do Apache Hadoop e do Apache Spark.

Os logs analisados têm em média 140000 linhas e demoram cerca de 4 segundos a serem processados. Verificou-se, conforme o gráfico abaixo, que a quantidade de memória assignada a cada máquina tem uma influência significativa na velocidade do processamento.



Tomando como exemplo o log de dia 21/06/2014, que tem 138235 linhas, o RDD gerado após as transformações *filter*, *map* e *split* contém 1435 registos correspondentes às linhas de erro presentes no log. Obtiveram-se as seguintes estatísticas a partir dos registos que contêm erros:

- Os únicos pórticos que reportaram erros durante este dia foram o 0407 (7 erros), o 4108 (6 erros), o 4120 (6 erros) e o 2539 (1 erro)
- Apenas foram reportados erros de nível WARN o que significa que os pórticos devem ser verificados e os problemas corrigidos.
- 19 dos 20 erros reportados devem-se ao modem do Sistema de Identificação Eletrónica (EIS) estar com dificuldades na receção de sinal. Apenas um dos erros reportados se deve a uma avaria na UPS do Multilane Controller (MLC).

A realização desta prova de conceito e consequente análise dos dados demonstra ser benéfica e ter potencialidade de ser implementada fora de um ambiente de desenvolvimento pois

permite com relativa facilidade analisar ficheiros de log gerados periodicamente e com grandes dimensões. Tendo em conta o exemplo mostrado, ao analisar estatisticamente a afetação de pórticos, os erros mais comuns ou os componentes com mais problemas facilmente reconhecem-se alguns padrões como por exemplo a receção de sinal dos modems do sistema de identificação eletrónica ser um problema recorrente e transversal aos vários pórticos. Pôde-se através desta prova de conceito observar a capacidade de processar dados em modo *stream* (processamento em tempo-real) bem como a velocidade de processamento através do armazenamento de dados em memória da plataforma Spark. Destaca-se ainda a versatilidade que esta ferramenta apresenta, devido às suas APIs, que permitiram facilmente a programação da aplicação.

Os logs da segunda prova de conceito, por estarem numa forma menos estruturada, foram analisados através da consola de forma a perceber que tipos de transformações se podem aplicar para obter alguma informação. Analisando o log relativo ao dia 02/08/2014 pode-se observar que os seguintes métodos da aplicação geraram exceções:

```
VMS_ECLR.ECLR.SetMessage(VMSMessageInfo
VMS_ECLR.ECLR.SendCommand(Byte[])
VMS_AESYS.AESYS.GetAlarms()
VMS_MicroProcessador.MicroProcessador.GetAlarms()
ProviderMock.MockVMS.SendCommand(Byte[])
GenericVMS.VMS.SignalRCommand(IHubProxy
Microsoft.AspNet.SignalR.Client.Hubs.HubProxy.Invoke[T](String
VMS_ECLR.ECLR.GetAlarms()
System.DateTime.DateToTicks(Int32
VMS_MicroProcessador.MicroProcessador.SetDateTime()
VMS_MicroProcessador.MicroProcessador.SendCommand(Byte[])
Microsoft.AspNet.SignalR.Client.Hubs.HubProxy.Invoke[TResult,TProgress](String
VMS_AESYS.AESYS.SendCommand(Byte[])
System.DateTime..ctor(Int32
VMS_AESYS.AESYS.SetDateTime()
ProviderMock.MockVMS.SetMessage(VMSMessageInfo
VMS_AESYS.AESYS.SetMessage(VMSMessageInfo
Microsoft.AspNet.SignalR.Client.Connection.Send(String
VMS_MicroProcessador.MicroProcessador.SetMessage(VMSMessageInfo
VMS_ECLR.ECLR.VMSClear()
GenericVMS.VMS.keepAliveTimer_Tick(Object
```

Verifica-se também, que para um tempo específico, por exemplo 23:50, os módulos que retornaram erro foram:

```
(23:50:25,984,SignalRCommand:)
(23:50:42,949,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)
(23:50:05,238,keepAliveTimer_Tick:)
(23:50:19,876,SignalRCommand:)
(23:50:57,946,SignalRCommand:)
(23:50:53,291,keepAliveTimer_Tick:SignalRCommand:)
(23:50:06,409,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)
(23:50:07,581,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)
(23:50:33,341,SignalRCommand:)
(23:50:17,111,keepAliveTimer_Tick:)
(23:50:55,931,SetMessage:)
(23:50:24,469,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)
(23:50:25,156,SetMessage:)
```

```
(23:50:58,024,SignalRCommand:)  
(23:50:00,707,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)  
(23:50:59,023,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)  
(23:50:18,330,keepAliveTimer_Tick:SignalRCommand:)  
(23:50:27,577,SignalRCommand:)  
(23:50:18,548,SignalRCommand:)  
(23:50:17,127,SignalRCommand:SignalRCommand:)  
(23:50:31,342,SignalRCommand:)  
(23:50:58,305,SignalRCommand:)  
(23:50:32,904,SetMessage:)  
(23:50:14,893,SignalRCommand:)  
(23:50:03,316,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)  
(23:50:45,636,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)  
(23:50:21,985,SignalRCommand:)  
(23:50:24,344,SignalRCommand:)  
(23:50:47,558,keepAliveTimer_Tick:SignalRCommand:)  
(23:50:04,706,SetMessage:)  
(23:50:21,751,keepAliveTimer_Tick:SignalRCommand:SignalRCommand:)  
(23:50:26,702,SignalRCommand:)  
(23:50:53,634,SignalRCommand:)  
(23:50:51,260,SignalRCommand:)
```

Na utilização da consola interativa da plataforma Apache Spark ficou presente a facilidade que oferece na necessidade de analisar um *dataset* através de *queries ad-hoc* e a possibilidade de executá-las de forma distribuída. A análise interativa adequa-se a uma primeira fase de análise podendo depois evoluir para uma aplicação *standalone*.

Capítulo V - Conclusões e trabalho futuro

Este trabalho permite concluir que cada vez mais, ferramentas como as utilizadas, se irão tornar comuns no mundo empresarial como são hoje em dia as ferramentas de gestão de bases de dados relacionais. A Internet of Things e a Big Data, além de conceitos abrangentes, são cada vez mais uma realidade. Os volumes de informação gerados aumentam a cada dia e com eles a necessidade de armazená-los e processá-los.

É cada vez mais uma realidade à qual as empresas e as pessoas terão que se adaptar e encarar como uma evolução tecnológica. Este é um paradigma que vai requerer habituação pois para além das vantagens óbvias a nível tecnológico, terá também um impacto social.

A realização deste trabalho é considerada uma mais-valia na medida em que permitiu tomar conhecimento acerca de tecnologias atuais utilizadas pelas maiores empresas do mundo. Embora de forma não muito aprofundada, exploraram-se as tecnologias de acordo com o necessário e obteve-se sucesso na sua implementação e utilização.

Ficam por explorar alguns aspetos do Apache Spark como o Spark SQL que permite analisar dados através de instruções SQL, a funcionalidade de Machine Learning oferecida pelo módulo MLlib e o GraphX que permite a criação e processamento de grafos.

Sendo este um trabalho que pretende implementar tecnologias que possam gerir grandes volumes de dados, verificaram-se como limitações o não existir uma amostra suficientemente grande para explorar todas as capacidades das plataformas Apache Spark e Apache Hadoop e também a falta de uma infraestrutura de maiores dimensões.

Como trabalho futuro e na continuação deste projeto, seria interessante desenvolvê-lo numa escala maior. Aplicá-lo ao mundo empresarial seria proveitoso de forma a poder explorar todas as potencialidades das plataformas utilizadas. Com o conhecimento obtido será possível dar continuidade a este trabalho e aprofundar a investigação feita.

Bibliografia

- [1] E. Schmidt, “Techonomy Conference,” Agosto 2010
- [2] M. Barlow, “Will Big Data Make IT Infrastructure Sexy Again?,” Março 2014
- [3] S. Ghemawat, H. Gobioff, S. Leung, “The Google File System,” Outubro 2003,
<http://static.googleusercontent.com/media/research.google.com/pt-PT//archive/gfs-sosp2003.pdf>
- [4] The Apache Software Foundation, “What is Apache Hadoop,” Agosto 2014,
<http://hadoop.apache.org/>
- [5] The Apache Software Foundation, “HDFS Architecture,” Junho 2014,
<http://hadoop.apache.org/docs>
- [6] Dominique A. Heger, “Hadoop Design, Architecture & MapReduce Performance,”
<http://www.datanubes.com/mediac/HadoopArchPerfDHT.pdf>
- [7] The Apache Software Foundation, “Project History,”
<https://spark.apache.org/community.html>
- [8] H. Karau, A. Kowinski, M. Zaharia, “Learning Spark – Lightning-fast Data Analysis,”
Junho 2014
- [9] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker,
I. Stoica, “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster
Computing,” http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf
- [10] The Apache Software Foundation, “Spark Programming Guide,”
<http://spark.apache.org/docs/>

Anexos

Anexo I - Exemplo de log gerado por pósito de cobrança de portagens

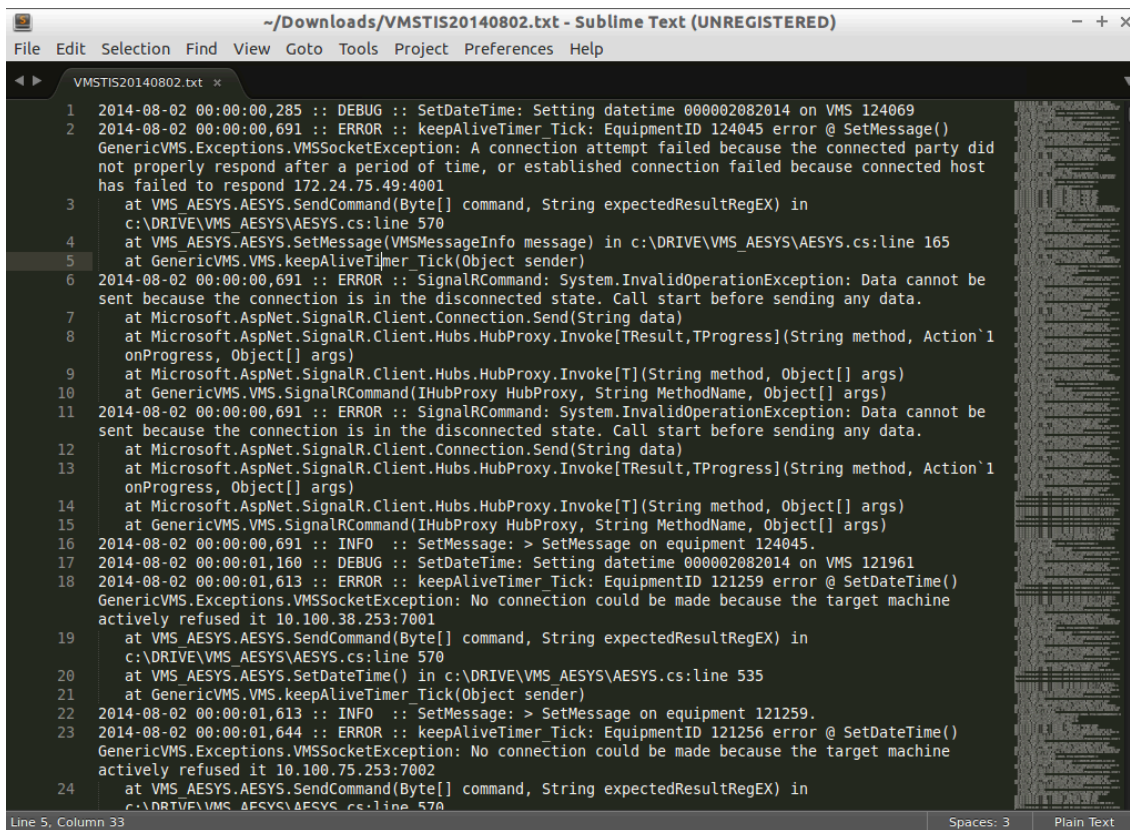
```

~/Documents/mon20140621.log - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

mon20140621.log x
77449 2014-06-21 01:47:01.502 pt02x4101mlc010 4101 RLOG000
77450 2014-06-21 01:47:03.750 pt02x4202mlc010 4202 RLOG000
77451 2014-06-21 01:47:03.953 pt02x4106mlc010 4106 RLOG000
77452 2014-06-21 01:47:06.707 pt02x0407mlc010 0407 RLOG002 WARN pt02x0407tx011 ""WARN RX-01: Low signal""
77453 2014-06-21 01:47:07.678 pt02x4108mlc010 4108 RLOG000
77454 2014-06-21 01:47:10.561 pt02x4107mlc010 4107 RLOG000
77455 2014-06-21 01:47:13.057 pt02x4120mlc010 4120 RLOG000
77456 2014-06-21 01:47:14.630 pt02x4119mlc010 4119 RLOG000
77457 2014-06-21 01:47:17.802 pt02x4201mlc010 4201 RLOG000
77458 2014-06-21 01:47:17.809 pt02x4110mlc010 4110 RLOG000
77459 2014-06-21 01:47:28.479 pt02x0408mlc010 0408 RLOG000
77460 2014-06-21 01:47:32.690 pt02x4208mlc010 4208 RLOG000
77461 2014-06-21 01:47:37.806 pt02x4207mlc010 4207 RLOG002 WARN pt02x4207tx011 ""WARN RX-11: Low signal""
77462 2014-06-21 01:47:38.029 pt02x4113mlc010 4113 RLOG000
77463 2014-06-21 01:47:38.058 pt02x4109mlc010 4109 RLOG000
77464 2014-06-21 01:47:38.837 pt02x4116mlc010 4116 RLOG000
77465 2014-06-21 01:47:39.504 pt02x4212mlc010 4212 RLOG000
77466 2014-06-21 01:47:40.586 pt02x4102mlc010 4102 RLOG000
77467 2014-06-21 01:47:42.868 pt02x4211mlc010 4211 RLOG000
77468 2014-06-21 01:47:44.233 pt02x0410mlc010 0410 RLOG000
77469 2014-06-21 01:47:56.656 pt02x4115mlc010 4115 RLOG000
77470 2014-06-21 01:47:58.377 pt02x4105mlc010 4105 RLOG000
77471 2014-06-21 01:47:59.250 pt02x0409mlc010 0409 RLOG000
77472 2014-06-21 01:48:01.207 pt02x4114mlc010 4114 RLOG000
77473 2014-06-21 01:48:01.503 pt02x4101mlc010 4101 RLOG000
77474 2014-06-21 01:48:03.750 pt02x4202mlc010 4202 RLOG000
77475 2014-06-21 01:48:03.953 pt02x4106mlc010 4106 RLOG000
77476 2014-06-21 01:48:06.708 pt02x0407mlc010 0407 RLOG002 WARN pt02x0407tx011 ""WARN RX-01: Low signal""
77477 2014-06-21 01:48:07.678 pt02x4108mlc010 4108 RLOG000
77478 2014-06-21 01:48:10.562 pt02x4107mlc010 4107 RLOG000
77479 2014-06-21 01:48:13.058 pt02x4120mlc010 4120 RLOG000
77480 2014-06-21 01:48:14.631 pt02x4119mlc010 4119 RLOG000
77481 2014-06-21 01:48:17.802 pt02x4201mlc010 4201 RLOG000
77482 2014-06-21 01:48:17.812 pt02x4110mlc010 4110 RLOG000
77483 2014-06-21 01:48:28.482 pt02x0408mlc010 0408 RLOG000
77484 2014-06-21 01:48:32.691 pt02x4208mlc010 4208 RLOG000
77485 2014-06-21 01:48:37.810 pt02x4207mlc010 4207 RLOG002 WARN pt02x4207tx011 ""WARN RX-11: Low signal""
77486 2014-06-21 01:48:38.030 pt02x4113mlc010 4113 RLOG000
Line 77468, Column 20
Tab Size: 4 Plain Text

```

Anexo II - Exemplo de log gerado por servidor web



```
--/Downloads/VMSTIS20140802.txt - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

VMSTIS20140802.txt x
1 2014-08-02 00:00:00,285 :: DEBUG :: SetDateTime: Setting datetime 000002082014 on VMS 124069
2 2014-08-02 00:00:00,691 :: ERROR :: keepAliveTimer Tick: EquipmentID 124045 error @ SetMessage()
   GenericVMS.Exceptions.VMSSocketException: A connection attempt failed because the connected party did
   not properly respond after a period of time, or established connection failed because connected host
   has failed to respond 172.24.75.49:4001
3   at VMS_AESYS.AESYS.SendCommand(Byte[] command, String expectedResultRegEX) in
   c:\DRIVE\VMS_AESYS\AESYS.cs:line 570
4   at VMS_AESYS.AESYS.SetMessage(VMSMessageInfo message) in c:\DRIVE\VMS_AESYS\AESYS.cs:line 165
5   at GenericVMS.VMS.keepAliveTimer.Tick(Object sender)
6 2014-08-02 00:00:00,691 :: ERROR :: SignalRCommand: System.InvalidOperationException: Data cannot be
   sent because the connection is in the disconnected state. Call start before sending any data.
7   at Microsoft.AspNet.SignalR.Client.Connection.Send(String data)
8   at Microsoft.AspNet.SignalR.Client.Hubs.HubProxy.Invoke[TResult,TProgress](String method, Action`1
   onProgress, Object[] args)
9   at Microsoft.AspNet.SignalR.Client.Hubs.HubProxy.Invoke[T](String method, Object[] args)
10  at GenericVMS.VMS.SignalRCommand(IHubProxy HubProxy, String MethodName, Object[] args)
11 2014-08-02 00:00:00,691 :: ERROR :: SignalRCommand: System.InvalidOperationException: Data cannot be
   sent because the connection is in the disconnected state. Call start before sending any data.
12  at Microsoft.AspNet.SignalR.Client.Connection.Send(String data)
13  at Microsoft.AspNet.SignalR.Client.Hubs.HubProxy.Invoke[TResult,TProgress](String method, Action`1
   onProgress, Object[] args)
14  at Microsoft.AspNet.SignalR.Client.Hubs.HubProxy.Invoke[T](String method, Object[] args)
15  at GenericVMS.VMS.SignalRCommand(IHubProxy HubProxy, String MethodName, Object[] args)
16 2014-08-02 00:00:00,691 :: INFO :: SetMessage: > SetMessage on equipment 124045.
17 2014-08-02 00:00:01,160 :: DEBUG :: SetDateTime: Setting datetime 000002082014 on VMS 121961
18 2014-08-02 00:00:01,613 :: ERROR :: keepAliveTimer Tick: EquipmentID 121259 error @ SetDateTime()
   GenericVMS.Exceptions.VMSSocketException: No connection could be made because the target machine
   actively refused it 10.100.38.253:7001
19  at VMS_AESYS.AESYS.SendCommand(Byte[] command, String expectedResultRegEX) in
   c:\DRIVE\VMS_AESYS\AESYS.cs:line 570
20  at VMS_AESYS.AESYS.SetDateTime() in c:\DRIVE\VMS_AESYS\AESYS.cs:line 535
21  at GenericVMS.VMS.keepAliveTimer.Tick(Object sender)
22 2014-08-02 00:00:01,613 :: INFO :: SetMessage: > SetMessage on equipment 121259.
23 2014-08-02 00:00:01,644 :: ERROR :: keepAliveTimer Tick: EquipmentID 121256 error @ SetDateTime()
   GenericVMS.Exceptions.VMSSocketException: No connection could be made because the target machine
   actively refused it 10.100.75.253:7002
24  at VMS_AESYS.AESYS.SendCommand(Byte[] command, String expectedResultRegEX) in
   c:\DRIVE\VMS_AESYS\AESYS.cs:line 570

Line 5, Column 33 Spaces: 3 Plain Text
```

Apêndices

Apêndice III - qfree.scala

```

01 import org.apache.spark.SparkContext
02 import org.apache.spark.SparkContext._
03 import org.apache.spark.SparkConf
04 import org.apache.spark.streaming.{Seconds, StreamingContext}
05 import org.apache.spark.streaming.StreamingContext._
06 import org.apache.spark.Logging
07 import org.apache.log4j.{Level, Logger}
08 import java.sql.{Connection, DriverManager, ResultSet};
09
10 object qfree {
11
12   def stripAll(s: String/*, bad: String*/): String = {
13     var bad : String = " :\""
14     @scala.annotation.tailrec def start(n: Int): String =
15       if (n == s.length) ""
16       else if (bad.indexOf(s.charAt(n)) < 0) end(n, s.length)
17       else start(1 + n)
18     @scala.annotation.tailrec def end(a: Int, n: Int): String =
19       if (n <= a) s.substring(a, n)
20       else if (bad.indexOf(s.charAt(n - 1)) < 0) s.substring(a, n)
21       else end(a, n - 1)
22     start(0)
23   }
24
25   def main(args: Array[String]) {
26     if (args.length < 1) {
27       System.err.println("Usage: qfree <directory>")
28       System.exit(1)
29     }
30
31     //mysql
32     val str = "jdbc:mysql://192.168.1.7:3306/spark?user=root&password=root"
33     Class.forName("com.mysql.jdbc.Driver").newInstance
34     val con = DriverManager.getConnection(str)
35     val statement = con.createStatement(ResultSet.TYPE_FORWARD_ONLY,
36     ResultSet.CONCUR_UPDATABLE)
37
38     //loglevel
39     StreamingExamples.setStreamingLogLevels()
40
41     //sparkconf
42     val sparkConf = new SparkConf().setAppName("HdWordCount")
43
44     //streamingcontext
45     val ssc = new StreamingContext(sparkConf, Seconds(2))
46
47     // Create the FileInputDStream on the directory and use the
48     // stream to count words in new files created
49     val lines = ssc.textFileStream(args(0))
50     val warns = lines.filter(line => line.contains("WARN"))
51     val split = warns.map(_.split(' '))
52     split.foreachRDD(rdd => {
53       val col = rdd.collect()
54       col.foreach(s => {
55         val prep = con.prepareStatement("INSERT INTO qfree
56         (data,timestamp,hostname,cpId,msgId,subsysstate,subsyshostname,compstate,complabel,compstatedesc
57         ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?) ")
58         prep.setString(1, stripAll(s(0)))
59         prep.setString(2, stripAll(s(1)))
60         prep.setString(3, stripAll(s(2)))
61         prep.setString(4, stripAll(s(3)))
62         prep.setString(5, stripAll(s(4)))
63         prep.setString(6, stripAll(s(5)))
64         prep.setString(7, stripAll(s(6)))
65         prep.setString(8, stripAll(s(7)))
66         prep.setString(9, stripAll(s(8)))
67         prep.setString(10, stripAll(s(9))+ " "+stripAll(s(10)))
68         prep.executeUpdate
69       })
70     })
71     ssc.start()

```

```
70     ssc.awaitTermination()  
71     con.close  
72 }  
73 }
```


Apêndice IV - qfree.html

```

001 <link rel="stylesheet" type="text/css"
href="https://cdn.datatables.net/1.10.2/css/jquery.dataTables.css">
002 <script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>
003 <script src="https://cdn.datatables.net/1.10.2/js/jquery.dataTables.min.js"></script>
004 <script src="jquery.dataTables.columnFilter.js"></script>
005 <script src="https://code.jquery.com/ui/1.11.1/jquery-ui.js"></script>
006
007 <link rel="stylesheet"
href="http://code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css">
008 <link rel="stylesheet" href="css/tables.css">
009
010 <style type="text/css">
011     tfoot input {
012         width: 100%;
013         padding: 3px;
014         box-sizing: border-box;
015     }
016     .dataTables_filter, .dataTables_info {
017         display: none;
018     }
019     table.dataTable thead th{
020         font-size: x-small;
021     }
022     table.dataTable tbody td{
023         font-size: x-small;
024     }
025     div.ui-datepicker{
026         font-size: 11px;
027     }
028     #container{
029         background-color: #FFEBEB;
030     }
031     #table1 input{
032         width:50px;
033     }
034     #table1 thead tr th {
035         text-align:left;
036         padding-left: 10px;
037     }
038     #dateInterval input{
039         width:100px;
040     }
041 </style>
042
043 <form id='dateInterval'>
044     <legend>Date interval:</legend>
045     <input type='text' id='ini'><br>
046     <input type='text' id='fim'>
047     <input type='submit' class="pure-button pure-button-primary" id='sub'>
048 </form>
049
050 <script>
051     $('#dateInterval').submit(function(){
052         updateFiltered(document.getElementById('ini').value,document.getElementById('fim').value);
053         return false;
054     });
055
056     $(function(){
057         $('#ini').datepicker({
058             dateFormat: 'yy-mm-dd',
059             inline: true,
060             showOtherMonths: true
061         });
062         $('#fim').datepicker({
063             dateFormat: 'yy-mm-dd',
064             inline: true,
065             showOtherMonths: true
066         });
067     });
068 </script>
069

```

```

070 <div id='table1'>
071 <table id="example" class="display" cellspacing="0" cellpadding="0">
072   <thead>
073     <tr>
074       <th></th>
075       <th></th>
076       <th></th>
077       <th></th>
078       <th></th>
079       <th></th>
080       <th></th>
081       <th></th>
082       <th></th>
083       <th></th>
084     </tr>
085     <tr>
086       <th>Data</th>
087       <th>Timestamp</th>
088       <th>Hostname</th>
089       <th>Charging Point</th>
090       <th>Message ID</th>
091       <th>Subsys State</th>
092       <th>Subsys Hostname</th>
093       <th>Comp State</th>
094       <th>Comp Label</th>
095       <th>Comp State Desc</th>
096     </tr>
097   </thead>
098
099   <tbody id='tbodyid'>
100 </tbody>
101 </table>
102 </div>
103 <br>
104 <div id='container'>
105   <div id='scroll'>
106     <table style='display:inline-block' class='filt' id="filtered">
107       <thead>
108         <tr>
109           <th>Charging Point</th>
110           <th># of errors</th>
111         </tr>
112       </thead>
113       <tbody id='filteredTbodyId'>
114 </tbody>
115     </table>
116     <table style='display:inline-block' class='filt' id="filtered2">
117       <thead>
118         <tr>
119           <th>Error</th>
120           <th># of occurrences</th>
121         </tr>
122       </thead>
123       <tbody id='filteredMsgId'>
124 </tbody>
125     </table>
126     <table style='display:inline-block' class='filt' id="filtered3">
127       <thead>
128         <tr>
129           <th>Affected Component</th>
130           <th># of times</th>
131         </tr>
132       </thead>
133       <tbody id='filteredCompLabel'>
134 </tbody>
135     </table>
136   </div>
137 </div>
138
139 <script>
140   function updateFiltered(ini,fim){
141     $.ajax({
142       type: "POST",
143       url: "test.php",

```

```

144         data: 'data='+ini+' '+fim,
145         dataType: 'json',
146         cache: false,
147         success: function(records) {
148             $('#example').dataTable().fnDestroy();
149             $('#tbodyid').empty();
150             $.each(records, function(i, item) {
151                 $('#tr>').html("<td>" + records[i].data + "</td><td>" +
records[i].timestamp + "</td><td>" + records[i].hostname + "</td><td>" + records[i].cpId +
"</td><td>" + records[i].msgId + "</td><td>" + records[i].subsysstate + "</td><td>" +
records[i].subsyshostname + "</td><td>" + records[i].compstate + "</td><td>" +
records[i].complabel + "</td><td>" + records[i].compstatedesc + "</td>").appendTo('#example');
152             });
153             filter()
154             contents()
155         }
156     });
157 }
158
159 function update() {
160     $.ajax({
161         type: "POST",
162         url: "test.php",
163         dataType: 'json',
164         cache: false,
165         success: function(records) {
166             $.each(records, function(i, item) {
167                 $('#tr>').html("<td>" + records[i].data + "</td><td>" +
records[i].timestamp + "</td><td>" + records[i].hostname + "</td><td>" + records[i].cpId +
"</td><td>" + records[i].msgId + "</td><td>" + records[i].subsysstate + "</td><td>" +
records[i].subsyshostname + "</td><td>" + records[i].compstate + "</td><td>" +
records[i].complabel + "</td><td>" + records[i].compstatedesc + "</td>").appendTo('#example');
168             });
169             filter()
170             contents()
171         }
172     });
173 }
174 function filter() {
175
176     var table = $('#example').dataTable({
177         'scrollY': '300px',
178         'scrollCollapse': true,
179         'bPaginate': false
180     }).columnFilter({
181         sPlaceholder: "head:before",
182         aoColumns: [
183             { type: "text" },
184             { type: "text" },
185             { type: "text" },
186             { type: "text" },
187             { type: "text" },
188             { type: "text" },
189             { type: "text" },
190             { type: "text" },
191             { type: "text" },
192             { type: "text" }
193         ]
194     });
195 }
196 function contents(){
197     var arr = new Array();
198     $('#example tbody tr').each(function(row,tr){
199         arr[row]=$ (tr).find('td:eq(1)').text();
200     });
201     var json = JSON.stringify(arr);
202     $.ajax({
203         type: 'POST',
204         url: 'test2.php',
205         data: 'data='+json,
206         success: function(msg){
207             $.ajax({
208                 type: 'POST',
209                 url: 'test3.php',

```

```
210         dataType: 'json',
211         cache: false,
212         success: function(records) {
213             $('#filteredTbodyId').empty();
214             $.each(records, function(i, item) {
215                 $('<tr>').html("<td>" + records[i].cpId + "</td><td>" +
records[i].cnt + "</td>").appendTo('#filtered');
216             });
217         }
218     });
219     $.ajax({
220         type: 'POST',
221         url: 'test4.php',
222         dataType: 'json',
223         cache: false,
224         success: function(records) {
225             $('#filteredMsgId').empty();
226             $.each(records, function(i, item) {
227                 $('<tr>').html("<td>" + records[i].msgId + "</td><td>" +
records[i].cnt + "</td>").appendTo('#filtered2');
228             });
229         }
230     });
231     $.ajax({
232         type: 'POST',
233         url: 'test5.php',
234         dataType: 'json',
235         cache: false,
236         success: function(records) {
237             $('#filteredComplabel').empty();
238             $.each(records, function(i, item) {
239                 $('<tr>').html("<td>" + records[i].complabel + "</td><td>" +
records[i].cnt + "</td>").appendTo('#filtered3');
240             });
241         }
242     });
243 }
244 })
245 }
246 update()
247 </script>
```

Apêndice V - getdata.php

```
01 <?php
02 $pdo = new PDO('mysql:host=localhost;dbname=spark', 'root', 'root');
03 if (count($_POST) > 0) {
04     $data      = stripslashes($_POST['data']);
05     $date      = explode(" ", $data);
06     $select    = 'select *';
07     $from      = ' from qfree';
08     $where     = " where data between '" . $date[0] . "' and '" . $date[1] . "'";
09     $sql       = $select . $from . $where;
10     $statement = $pdo->prepare($sql);
11     $statement->execute();
12     $results   = $statement->fetchAll(PDO::FETCH_ASSOC);
13     $json      = json_encode($results);
14     echo ($json);
15 } else {
16     $select    = 'select *';
17     $from      = ' from qfree limit 20';
18     $where     = null;
19     $sql       = $select . $from . $where;
20     $statement = $pdo->prepare($sql);
21     $statement->execute();
22     $results   = $statement->fetchAll(PDO::FETCH_ASSOC);
23     $json      = json_encode($results);
24     echo ($json);
25 }
26 ?>
```

Apêndice VI - filter.php

```
01 <?php
02 $pdo = new PDO('mysql:host=localhost;dbname=spark', 'root', 'root');
03 $json = '';
04 if (isset($_POST['data'])) {
05     $data = stripslashes($_POST['data']);
06     $data = json_decode($data, TRUE);
07     $str = implode("'", $data);
08     $str = "('$str')";
09     $sql = 'drop table if exists temp; create table temp select * from qfree where
timestamp in '.$str;
10     $statement = $pdo->prepare($sql);
11     $statement->execute();
12 }
13 ?>
```

Apêndice VII - cp.php

```
01 <?php
02 $pdo = new PDO('mysql:host=localhost;dbname=spark', 'root', 'root');
03
04 $sql = 'select cpId,count(*) as cnt from temp group by cpId order by cnt desc';
05 $statement = $pdo->prepare($sql);
06 $statement->execute();
07 $results = $statement->fetchAll(PDO::FETCH_ASSOC);
08 $json = json_encode($results);
09 echo ($json);
10 ?>
```

Apêndice VIII - msg.php

```
01 <?php
02 $pdo = new PDO('mysql:host=localhost;dbname=spark', 'root', 'root');
03
04 $sql      = 'select msgId,count(*) as cnt from temp group by msgId order by cnt desc';
05 $statement = $pdo->prepare($sql);
06 $statement->execute();
07 $results  = $statement->fetchAll(PDO::FETCH_ASSOC);
08 $json     = json_encode($results);
09 echo ($json);
10 ?>
```


Apêndice IX - comp.php

```
01 <?php
02 $pdo = new PDO('mysql:host=localhost;dbname=spark', 'root', 'root');
03
04 $sql = 'select complabel,count(*) as cnt from temp group by complabel order by cnt
desc';
05 $statement = $pdo->prepare($sql);
06 $statement->execute();
07 $results = $statement->fetchAll(PDO::FETCH_ASSOC);
08 $json = json_encode($results);
09 echo ($json);
10 ?>
```

Apêndice X - Interface Web

Mozilla Firefox

http://local...st/test.html

localhost/test.html

Submit Query

Data	Timestamp	Hostname	Charging Point	Message ID	Subsys State	Subsys Hostname	Comp State	Comp Label	Comp State Desc
2014-06-21	11:14:53.211	pt02x2539mlc010	2539	RLOG002	WARN	pt02x2539mlc010	WARN	UPS:No	external power
2014-06-21	00:00:06.612	pt02x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal
2014-06-21	00:00:07.558	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:00:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal
2014-06-21	00:01:06.613	pt02x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal
2014-06-21	00:01:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:01:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal
2014-06-21	00:02:06.613	pt02x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal
2014-06-21	00:02:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:02:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal

Charging Point	# of errors
0407	7
4108	6
4120	6
2539	1

Error	# of occurrences	Affected Component	# of times
RLOG002	20	RX-01	19
		UPS:No	1

Apêndice XI – Spark Shell

```
01 val log = sc.textFile("/home/.../VMSTIS20140802.txt");
02 val erros = log.filter(_.contains("ERROR"));
03 val data = erros.filter(_.contains("00:00:06"));
04 val split = data.map(_.split(" "));
05 val reduce = split.map(line=>(line(5),line(1))).reduceByKey(_+_,1).collect.foreach(println)
```

Apêndice XII – Interface Web, filtrar por data

Mozilla Firefox

http://local...st/test.html

localhost/test.html

Date interval:
2014-09-01

Submit Query

September 2014

Su	Mo	Tu	We	Th	Fr	Sa
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

stname	Charging Point	Message ID	Subsys State	Subsys Hostname	Comp State	Comp Label	Comp State Desc		
2x2539mlc010	2539	RLOG002	WARN	pt02x2539mlc010	WARN	UPS:No	external power		
2x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal		
2014-06-21	00:00:07.558	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:00:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal
2014-06-21	00:01:06.613	pt02x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal
2014-06-21	00:01:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:01:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal
2014-06-21	00:02:06.613	pt02x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal
2014-06-21	00:02:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:02:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal

Charging Point	# of errors
0407	7
4108	6
4120	6
2539	1

Error	# of occurrences
RLOG002	20

Affected Component	# of times
RX-01	19
UPS:No	1

Apêndice XIII – Interface Web, filtrar por Hostname

Mozilla Firefox

http://local...st/test.html

localhost/test.html

Date interval:

Data	Timestamp	Hostname	Charging Point	Message ID	Subsys State	Subsys Hostname	Comp State	Comp Label	Comp State Desc
2014-06-21	00:00:07.558	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:01:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:02:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:03:07.566	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:04:07.567	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:05:07.568	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal

Charging Point	# of errors
0407	7
4108	6
4120	6
2539	1

Error	# of occurrences	Affected Component	# of times
RLOG002	20	RX-01	19
		UPS:No	1

Apêndice XIV – Interface Web, filtrar por Timestamp

Mozilla Firefox

http://local...st/test.html

localhost/test.html

Date interval:

Submit Query

00:01

Data	Timestamp	Hostname	Charging Point	Message ID	Subsys State	Subsys Hostname	Comp State	Comp Label	Comp State Desc
2014-06-21	00:01:06.613	pt02x0407mlc010	0407	RLOG002	WARN	pt02x0407tx011	WARN	RX-01	Low signal
2014-06-21	00:01:07.563	pt02x4108mlc010	4108	RLOG002	WARN	pt02x4108tx011	WARN	RX-01	Low signal
2014-06-21	00:01:13.000	pt02x4120mlc010	4120	RLOG002	WARN	pt02x4120tx011	WARN	RX-01	Low signal

Charging Point	# of errors
0407	7
4108	6
4120	6
2539	1

Error	# of occurrences	Affected Component	# of times
RLOG002	20	RX-01	19
		UPS:No	1

Manual técnico

Instalação Hadoop

Requisitos: Java 1.6 ou mais recente, SSH

Extrair o conteúdo do ficheiro `hadoop-2.4.1.tar.gz` para uma pasta. Assume-se que a pasta é `/usr/local`.

Editar o ficheiro `/usr/local/hadoop-2.4.1/etc/hadoop/hadoop-env.sh` e adicionar a seguinte linha:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java
```

Editar o ficheiro `/usr/local/hadoop-2.4.1/etc/hadoop/core-site.xml` e adicionar a seguinte configuração:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Editar o ficheiro `/usr/local/hadoop-2.4.1/etc/hadoop/core-site.xml` e adicionar a seguinte configuração:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

Configurar o SSH para acesso sem *password*:

```
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

Editar o ficheiro `/usr/local/hadoop-2.4.1/etc/hadoop/slaves` e adicionar as máquinas que desempenharão o papel de slave no *cluster*. Ex:

```
slave1
192.168.1.2
slave3
```

Instalação Spark

Extrair o conteúdo do ficheiro spark-1.0.2-bin-hadoop2.tgz para uma pasta. Assume-se que a pasta é /usr/local.

Editar o ficheiro /usr/local/spark-1.0.2-bin-hadoop2/conf/slaves e adicionar as máquinas que desempenharão o papel de slave no cluster. Ex:

```
slave1
192.168.1.2
slave3
```

Editar o ficheiro /usr/local/spark-1.0.2-bin-hadoop2/conf/slaves e adicionar as seguintes linhas:

```
export STANDALONE_SPARK_MASTER_HOST=<fqdn da máquina que irá desempenhar
o papel de master no cluster>
export SPARK_MASTER_IP=<ip da máquina que irá desempenhar o papel de
master no cluster>
```

Iniciar *cluster* Hadoop

Formatar o sistema de ficheiros:

```
$ bin/hdfs namenode -format
```

Iniciar os serviços NameNode e DataNode:

```
$ sbin/start-dfs.sh
```

Iniciar os serviços ResourceManager e NodeManager:

```
$ sbin/start-yarn.sh
```

Aceder à interface Web do NameNode:

```
http://localhost:50070/
```

Iniciar *cluster* Spark

Iniciar o serviço master:

```
$ sbin/start-master.sh
```

Iniciar os serviços master e slave:

```
$ sbin/start-all.sh
```

Estabelecer ligação a partir de um slave ao master:

```
$ bin/spark-class org.apache.spark.deploy.worker.Worker spark://IP:PORT
```


Manipular HDFS

Listar pastas:

```
$ bin/hdfs dfs -ls /
```

Criar pasta:

```
$ bin/hdfs dfs -mkdir /pasta
```

Copiar ficheiro ou pasta para o HDFS:

```
$ bin/hdfs dfs -copyFromLocal /origem /destino
```

Remover ficheiro ou pasta do HDFS:

```
$ bin/hdfs dfs -rmr /pasta
```

Utilização Spark *shell*

Iniciar shell:

```
$ bin/spark-shell
```

Carregar um ficheiro de texto:

```
val file = sc.textFile("teste.txt");
```

Algumas acções e transformações:

```
textFile.count();
```

```
textFile.first();
```

```
textFile.filter(line => line.contains("teste")).count();
```

Glossário

Apache – Organização sem fins lucrativos criada para suportar os projectos Apache

Apache Hadoop – Plataforma open-source para armazenamento e processamento de dados de forma distribuída

Apache Spark – Plataforma open-source para análise de dados de forma distribuída

Internet of Things – Conceito utilizado para referir objectos que comunicam entre si utilizando a Internet

Big Data – Conceito utilizado para referir grandes volumes de dados

HDFS – Hadoop Distributed File System

MapReduce – Framework de processamento paralelo de dados

Cluster – Conjunto de computadores ligados entre si que trabalham em conjunto podendo ser vistos como um único sistema

NameNode – Responsável por manter a hierarquia de pasta e ficheiros bem como a localização dos dados num sistema de ficheiros HDFS

DataNode – Responsável por armazenar e processar os dados presentes num sistema de ficheiros HDFS

RDD – Resilient Distributed Dataset

Log – Ficheiro que guarda um registo de eventos