Universidade de Évora
Escola de Ciências e Tecnologia
Departamento de Informática

Engenharia Informática

Análise e Detecção Inteligente de Anomalias

Josué D. A. Duarte Delgado

Orientador: *Prof. Luís Rato*

Évora, Outubro 2011

Universidade de Évora
Escola de Ciências e Tecnologia
Departamento de Informática

Engenharia Informática

Análise e Detecção Inteligente de Anomalias

Josué D. A. Duarte Delgado

Orientador: *Prof. Luís Rato*

Évora, Outubro 2011

Agradecimentos

Em primeiro lugar agradeço aos meus pais, porque começou neles a formação da pessoa que sou hoje.

À Eveline, por todo o amor e suporte, aos meus tios, José e Raquel, aos meus primos e D. Marcelina, à restante família em Cabo Verde, um muito obrigado por todo o carinho e ajuda.

Aos meus amigos, Vicente, Adenilda, Flávio, Nuno, Luís, Josué e Eloise Reis e muitos mais que não poderei enumerar aqui, um muito obrigado por me suportarem durante este período.

Um agradacimento muito especial ao Artur Romão e ao Miguel Reis, pelo tempo passado juntos e pela constante troca de conhecimento.

Um agradecimento especial a Deus pelo trabalho feito na minha vida e pela oportunidade que me deu de fazer parte da Sua família.

Sumário

Este trabalho tem como objectivo a construção de um modelo de representação de eventos e análise de padrões. O trabalho apresentado usa uma base com grandes volumes de dados, detectando-se, de forma dinâmica, situações anómalas, i.e., situações com eventos cuja representação seja desenquadrada dos padrões mais frequentes.

Para se atingir este objectivo usam-se técnicas de mineração de dados - classificação e aprendizagem - para catalogação dos eventos, sendo feito um mapeamento num espaço de características previamente definido.

Neste trabalho irá ser utilizada informação sobre dados reais de negócio de uma empresa. Com esta informação pretende-se, após implementação do modelo, ter um sistema de notificações para auxiliar num processo de decisão no negócio. Em particular, pretende-se dotar o sistema de meios para detectar anomalias, em tempo real, gerando alarmes sobre as mesmas.

Intelligent Detection and Analysis of Anomalies

This project has the goal to build a representation model of events and pattern analysis. The project uses a base of large volumes of data, so it can detect, in a dynamic way, anomalies, namely situations with events whose representation is not framed in the regular patterns.

To achieve this goal we use data mining techniques - classification and learning - to categorize events, making a mapping in a space with predefined features.

In this project it will be used information about real business data of a company. With this information we intend to, after applying the model, have a notification system to help in the business decisions. Particularly, we intend to detect anomalies in real-time, generating alarms when they occur.

Conteúdo

1 Introdução **1**
- 1.1 Enquadramento . . . 1
- 1.2 Saphety . . . 1
- 1.3 Objectivos . . . 2
- 1.4 Principais contribuições . . . 3
- 1.5 Estrutura . . . 3

2 Estado da Arte **5**
- 2.1 SGBDS . . . 5
- 2.2 NoSQL . . . 6
- 2.3 DataWarehouse e Business Intelligence . . . 6
- 2.4 Outras plataformas . . . 7
- 2.5 Trabalho relacionado . . . 8

3 Levantamento de Requisitos **9**
- 3.1 Análise dos dados . . . 9
  - 3.1.1 Sistemas de Messaging . . . 10
  - 3.1.2 Factura Electrónica . . . 13
- 3.2 Requisitos de negócio . . . 16

4 Ambiente de desenvolvimento e tecnologias adoptadas **17**
- 4.1 Tecnologias adoptadas . . . 17
- 4.2 Ambiente de Desenvolvimento . . . 19
- 4.3 Criação de sistema de computação distribuída . . . 20
  - 4.3.1 Servidores . . . 20
  - 4.3.2 Arquitectura . . . 20
- 4.4 Criação de plataforma analítica . . . 26
  - 4.4.1 Mahout . . . 26

5 Testes e apresentação de resultados **29**

6 Conclusões e Trabalho Futuro **35**
- 6.1 Conclusão . . . 35

6.2 Trabalho Futuro 36

Lista de Figuras

3.1 Número de transacções por mês 11
3.2 Volume de informação gerada por mês 12
3.3 Integration Scenario 13
3.4 Diagrama ER EInvoice 15

4.1 Cluster Hadoop 20
4.2 Serviços nos servidores 21
4.3 NameNode Details 22
4.4 Hadoop Map/Reduce Administration 24
4.5 Hadoop Job Details 25

5.1 Resultados de benchmark 33

Capítulo 1

Introdução

1.1 Enquadramento

Com o crescente volume de dados de negócios, transacções diversas, informação externa de interesse, registos locais de serviços (logs, qualidade de serviço interno e de fornecedores), torna-se essencial poder extrair informação útil e relevante para a operação de uma empresa.

De forma a poder trazer mais valor ao serviço prestado aos clientes, surge a necessidade de estender os serviços e a monitorização dos sistemas para além da observação da disponibilidade dos serviços, entrando assim na análise de fluxos de informação e consistência dos diversos dados trocados.

O objectivo deste trabalho é dotar as empresas dos recursos necessários à análise de dados e gerar notificações para quando a qualidade dos dados não é a esperada. Desta forma, as empresas poderão ganhar vantagem competitiva relativamente aos concorrentes e oferecerem serviços de enorme utilidade aos clientes.

Por esse motivo, este trabalho visa preencher uma lacuna encontrada no tratamento da informação nas empresas.

É objectivo deste trabalho disponibilizar um sistema capaz de tirar o máximo partido do hardware, utilizando mecanismos de distribuição de tarefas. Devido à dimensão dos dados usados, hoje quase um *terabyte* e com estimativa de crescimento para as dezenas, este desafio só pode ser enfrentado através de processamento distribuído.

1.2 Saphety

Para realização deste trabalho, utilizaram-se os dados gerados da operação da Saphety Level Trusted Services, S.A., para criação de protótipos e de uma plataforma de processa-

mento.

A Saphety é uma empresa do grupo SONAECOM que presta um conjunto de serviços de valor acrescentado na área dos *trusted services*[1], tais como Certificação Digital e troca segura de mensagens B2B (Bussiness to Bussines).

Como prestadora de serviços que envolvem dados, é crucial estar provida de ferramentas que possam ajudar na análise e na prevenção de anomalias que possam comprometer o normal funcionamento do serviço a clientes.

Neste cenário, surge a proposta de estudo dos diversos componentes de negócio e materialização dos mesmo num projecto onde a ciência e o negócio podem ser conjugados de maneira a trazer benefícios para ambos os campos.

Este trabalho incide num dos componentes dos serviços prestados: Factura Electrónica. Com o crescer do negócio e do mercado da Factura Electrónica, a Saphety teve de adaptar os sistemas e ter uma plataforma que permitisse acompanhar o enorme volume de dados associados a este serviço.

Para ganhar esta vantagem competitiva, como é objectivo descrito no enquadramento, era necessário que este trabalho fosse a base de um amplo projecto de integração, onde, depois de analisadas, fossem incluídos os dados dos serviços de factura electrónica, contratos electrónicos, transacções diversas e respectivos registos. Assim poder-se-á conjugar diversos dados, como por exemplo um cruzamento entre dados de indisponibilidade de serviços, prestados por fornecedores externos, com a qualidade do serviço prestado aos clientes.

1.3 Objectivos

Os objectivos do trabalho realizado são os seguintes:

- A modelização de um vasto conjunto de dados, com origem em registos transaccionais, em contextos específicos de comércio electrónico, de forma a poderem ser detectados padrões e efectuar análises predictivas sobre os mesmos;
- A construção de repositórios com capacidade para armazenar os dados transaccionais, de acordo com os modelos referidos, adaptados a requisitos de altos débitos de acesso;
- A implementação de aplicações que permitam efectuar as análises referidas acima, de forma a detectarem padrões de comportamento anómalos em conjuntos muito vastos de registos eventos transaccionais, e gerar notificações sobre a ocorrência actual ou futura de problemas potenciais decorrentes dessas anomalias.

[1] Serviços que asseguram a troca segura de informação por via electrónica, como a certificação digital, Factura Electrónica (armazenamento e respectivos workflows de validação), Contratos Electrónicos, Contratação Pública, entre outros.

1.4 Principais contribuições

Dados os objectivos traçados e os resultados atingidos, as principais contribuições desta tese são:

- Modelos de dados transaccionais, orientados para contextos de grandes volumes de dados, passíveis de serem analisados segundo técnicas de *business intelligence*;
- Aplicações para análise de grandes volumes de dados, suportadas em tecnologias emergentes orientadas para este tipo de cenários, permitindo detectar anomalias (reais ou potenciais) e gerar notificações sobre as mesmas, permitindo a tomada de decisões e acções preventivas de forma a minimizar os impactos dessas anomalias.

Um contexto específico de aplicabilidade dos resultados deste trabalho é em áreas de gestão de operações de sistemas de comércio electrónico, onde são gerados muitos eventos, e nos quais a detecção precoce de anomalias aumenta as probabilidades de actuação sobre as mesmas, oferecendo uma maior possibilidade de melhoria dos serviços prestados.

1.5 Estrutura

Este trabalho está dividido em:

Capítulo 1 - **Introdução,** contém os pressupostos da realização do trabalho.

Capítulo 2 - **Estado da arte,** descreve o que existe de tecnologia de suporte e trabalho relacionado.

Capítulo 3 - **Desenvolvimento,** contém a descrição sobre a implementação feita.

Capítulo 5 - **Resultados,** contém a análise dos resultados obtidos e métricas de performance.

Capítulo 6 - **Conclusão,** contém a análise aos objectivos do trabalho.

Capítulo 2

Estado da Arte

2.1 SGBDS

Os SGBDs (sistemas de gestão de base de dados), têm acompanhado a evolução da indústria de TI, adaptando-se e dando resposta a um número crescente de necessidades das empresas para gestão da informação.

Estes sistemas estão cada vez mais direccionados a tratar informação estruturada. Para serem tidos em conta em qualquer projecto, oferecem funções de integridade da informação, permissões de segurança, sistema de registo de alterações efectuadas e diversas outras funções direccionadas ao tratamento da informação.

Quando se começa a escalar um sistema baseado em base de dados, começa-se a ter os primeiros problemas com o volume de informação e aparecem as opções de crescimento:

1. **Escalar Verticalmente** - Substitui-se hardware menos potente por hardware com mais capacidade. Normalmente chega para muitos dos cenários em que o crescimento não é muito acentuado e não há necessidade constante de processar dados não estruturados, semi-estruturados e estruturados[18, p29][16, p30].

2. **Escalar Horizontalmente** - Aqui o desafio é maior, visto o objectivo ser que muitos servidores possam responder aos pedidos. Para isso tem-se adoptado mecanismos para a partição de dados por diversos grupos de servidores, em que cada grupo tem um *master* e diversos *slaves*. É preciso a criação de camadas lógicas de maneira a guardar a informação de forma particionada e as aplicações terão que ser adaptadas para saber em que grupo de servidores está a informação que que se pretende.

   Este mecanismo é utilizado em diversos sítios web para poderem crescer, por exemplo no Twitter, Facebook, Typepad, Yahoo!, e Google[16].

2.2 NoSQL

A pressão para que a escalabilidade das bases de dados seja feita horizontalmente levou a indústria e a comunidade académica a desenvolver novos modelos. Um deles é o chamado *NoSQL* ("Not only SQL"), um termo genérico para uma classe de bases de dados não-relacionais. Uma característica distintiva destas bases de dados relaciona-se com o facto de não terem como objectivo fornecer garantias ACID[1].

Como exemplos deste tipo de bases de dados temos a BigTable da Google[7], Apache Cassandra e Apache HBase[21, p13, p17]. Todas estas tecnologias destinam-se a ser usadas em ambientes caracterizados por grandes volumes de dados, que necessitam de processamento intensivo, sobretudo para fins analíticos[21, p9].

2.3 DataWarehouse e Business Intelligence

Os sistemas de DataWarehouse são normalmente baseados em SGBDs. O objectivo é serem carregados com informação operacional (bases de dados), obedecendo a estruturas previamente definidas para se poder, através de componentes de *reporting* e *dashboarding*, mostrar a informação que se pretende já tratada.

Foram analisados alguns fornecedores de soluções de implementação de DataWarehouse opensource (JasperSoft e Pentahoo) e alguns dos principais *players* neste mercado.

JasperSoft e Pentahoo Estes dois fornecedores oferecem as soluções de BI, *Business Intelligence* mais conhecidas no mundo OpenSource. Suportam as funcionalidades de um tradicional sistema de BI e disponibilizam ferramentas de *ETL*, *Data Analysis* e *Reporting*, tornando-se assim sérios concorrentes para os tradicionais grandes fornecedores de BI, para o mercado de pequenas e médias empresas, em que o preço é um factor determinante.[11]

Outros (SAP Bussiness Objects, Oracle Hyperion, IBM Cognos, Microsoft BI) Estes fornecedores estão direccionados a médias e grandes empresas. A aposta é tal que o lote de funcionalidades é grande. O objectivo é oferecer um ambiente integrado onde, segundo o artigo "Magic Quadrant BI Plataforms" publicado pela Gartner[10], estão incluídos:

- Integração - Infraestrutura de BI, Gestão de Metadados, Ambiente de desenvolvimento e plataforma de Workflow e colaboração
- Disponibilização de informação - Relatórios, *Dashboards*, *Ad Hoc Query* e integração com ferramentas de escritório (e.g Microsoft Office).

A aposta neste tipo de soluções é limitada pelo preço de aquisição e também porque com equipas bem preparadas e ferramentas de BI opensource consegue-se responder às necessidades de informação das pequenas e médias empresas.

Existem muitas regras e conjuntos de boas práticas para se construir um DataWarehouse[3], mas na maior parte dos casos os sistemas de tratamento de informação podem constituir um entrave, dado que por estarem baseados em bases de dados estruturadas, ter que definir a granulidade da informação, as dimensões que se quer e as tabelas de factos aumenta a complexidade para o que se quer neste trabalho.

2.4 Outras plataformas

Devido às limitações encontradas nos sistemas analisados anteriormente, quer de ordem económica quer de ordem técnica, foi necessário procurar outro tipo de solução que fosse de encontro às necessidades tanto de processamento como de escalabilidade e investimento monetário.

Neste contexto foi analisada a plataforma de processamento Hadoop (`http://hadoop.apache.org`), criada inicialmente pelo Yahoo! Inc. e mantida nos últimos anos pela Apache Foundation. O interesse nesta plataforma surgiu pelos diversos artigos[13, 15] que demostram como é usada para tarefas de processamento de informação, bases de dados escaláveis e distribuídas, *datamining* e *machine learning* e coordenação de serviços para aplicações distribuídas.

Esta plataforma não é para substituir as habituais plataformas de operação, mas sim para o processamento e tratamento de informação, suportando dados estruturados, semi-estruturados e não estruturados, com capacidade de escala até aos *petabytes* devido à arquitectura distribuída.

Podemos tomar como exemplo para uma implementação deste tipo, a arquitectura implementada pela Rackspace (empresa que presta serviços de alojamento web e de e-mail). Os sistemas deles geram cerca de 150GB de *logs* de processamento da plataforma de e-mail e precisavam de dados analíticos resultantes do tratamento da informação gerada. Por isso, criaram um *cluster* com 15 nós de dados, com 500GB cada um e utilizaram o Hadoop/-MapReduce para um sistema de *tracking* de mensagens e para terem também os dados analíticos consolidados por cada dia de operação[18, p538].

Outro exemplo, este de grande proporção, pode ser encontrado na implementação de Hadoop feita pelo Facebook. Devido a problemas de performance e escalabilidade encontrados na implementação de *data warehouse* utilizando tecnologia da Oracle[18, p506], foi criado um *cluster* Hadoop. Esta infra-estrutura criada cresceu até cerca de 2400 cores e 9 TB de memória ram, processa 10 TB de dados por dia e armazena mais de 2 PB de dados[18, p507].

2.5 Trabalho relacionado

Muitas empresas e investigadores têm respondido ao desafio de integração de todos os dados da operação informática, oferecendo soluções para tratamento de dados e visualização.

Splunk Esta empresa fornece uma aplicação onde se poderão concentrar todos os dados das aplicações, registos de tratamento de informação, dados dos servidores, indexando assim toda esta informação[19]. Os dados recolhidos são apresentados em forma de *dashboards* e relatórios, depois de serem processados pela componente analítica.

Esta aplicação está muito mais vocacionada para tarefas *real time* e não se conhecem casos de utilização onde a dimensão da informação seja relevante para se poder comparar com sistemas implementadas recorrendo ao *Hadoop*.

ParaAccel Os sistemas de *datawarehouse*, a sua complexidade, as limitações dos SGBDs na capacidade e rapidez, são os pontos fortes que a ParaAccel quer revolucionar[20] através do seu produto: *ParaAceel Analytic Database*.

A arquitectura e o modo de funcionamento deste sistema são muito semelhantes ao Hadoop [20, p6]. Existem *leadernodes* que funcionam da mesma forma que os *namenode* e *jobtracker* no Hadoop. Estes são os responsáveis por toda a coordenação dos outros nós do cluster e são a única interface para os sistemas clientes.

A parte analítica e o tratamento de informação não estruturada, são lacunas encontradas neste sistema.

Capítulo 3

Levantamento de Requisitos

3.1 Análise dos dados

A Saphety, como prestadora de serviços online, gera e trata muitos dados a cada instante. Estes dados são guardados em sistemas diversos. Por existir esta dispersão, a necessidade da centralização da gestão dos dados é enorme, de forma a que tenham significado, passando assim a ser informação útil para o negócio. Contudo, além dos dados que ficam centralizados, existem as transacções e respectivos registos gerados, os dados de acesso, erros de processamento por parte dos clientes e erros de sistema. Tudo isto é informação que não deve ser descuidada e sim tratada como um todo, mesmo que o âmbito deste trabalho seja muito mais restrito devido à dimensão que um projecto desta natureza pode tomar.

Desta forma começou-se por dividir e tratar, a informação gerada, em grupos:

- Dados guardados em bases de dados, que correspondem à informação de clientes e aprovisionamento geral dos serviços;
- Dados resultantes do processamento das aplicações e das diversas transacções;
- Dados resultantes dos acessos ou tentativas de acesso aos diversos serviços expostos para fora (Internet);
- Dados resultantes da monitorização da disponibilidade dos serviços e recursos utilizados (por exemplo largura de banda ocupada, carga dos sistemas, entre outros indicadores).
- Dados externos à organização, mas que podem ser mais valias na percepção da evolução do negócio (e.g. relatórios financeiros, dados estatísticos de observatórios do ramo, etc).

Toda esta informação teve de ser analisada como um todo, visto que há muita informação que poderá ser cruzada e analisada no futuro, através dos fluxos de informação. Isto permite sugerir uma plataforma e tecnologias que permitam processar elevados volumes de

dados.

3.1.1 Sistemas de Messaging

Os sistemas de messaging são o canal de transmissão de todos os dados dos serviços prestados e funcionam como base de toda plataforma EDI (*Electronic Data Interchange*)[1]. Há diversos sistemas presentes e formas de comunicação, por isso fazemos uma pequena descrição de cada um[4, 17]:

AS1 Especificação de transporte de dados de forma segura e fiável, que se assenta sobre SMTP e utiliza assinatura digital e cifra entre o emissor e receptor para garantir a segurança da informação (confidencialidade, não repúdio, etc).

AS2 Especificação de transporte de dados de forma segura, à semelhança do AS1, mas que funciona sobre *http* ou *https*.

Web Services A implementação de Web Services internas é baseada na norma WSDL (*Web Services Description Language*)[2]. Utilizada para integrações directas com clientes.

SFTP De *SSH File Transfer Protocol*, serve para trocas de ficheiros de dimensões não comportáveis para os protocolos anteriormente descritos.

eFAX Plataforma criada internamente na Saphety, que conjuga tradicionais linhas de fax com um dos protocolos anteriormente descritos. Desta forma consegue-se "unir" empresas que já utilizam as tecnologias de comunicação segura, que foram descritas acima, com empresas que ainda usam equipamentos de fax na sua operação logística.

Não sendo o foco do trabalho, foi necessário compreender a dimensão e o volume de dados trocados utilizando estes protocolos. Cada serviço da empresa, baseada nos protocolos acima, guarda informação estruturada em bases de dados, contendo remetente, destinatário, datas de envio/recepção e avisos de recepção. Além desta informação, toda a informação do processamento é guardada de forma semi-estruturada em ficheiros de registos, em cada ponto de passagem dos dados.

A figura 3.1 mostra a evolução do número de transacções respeitantes ao tráfego e documentos de factura electrónica agrupados por mês. Têm crescido de forma sustentada e estão na ordem dos oito milhões de transacções mensais. Contudo o plano de crescimento da empresa prevê que a curto prazo esse número seja dez vezes mais, passando então a oitenta milhões de transacções mensais.

Devido a estas previsões de crescimento, é cada vez mais importante que a plataforma proposta tenha capacidade de processamento no dia-a-dia e seja desenhada para crescer e não ter problemas de escala. O espaço necessário para dados também é fundamental, visto

[1] Troca segura de informação estruturada entre sistemas/aplicações de forma automática ou com a mínima intervenção externa

[2] Especificação da W3C baseada em XML que descreve serviços de rede para troca de mensagens ou procedimentos entre *endpoints*.

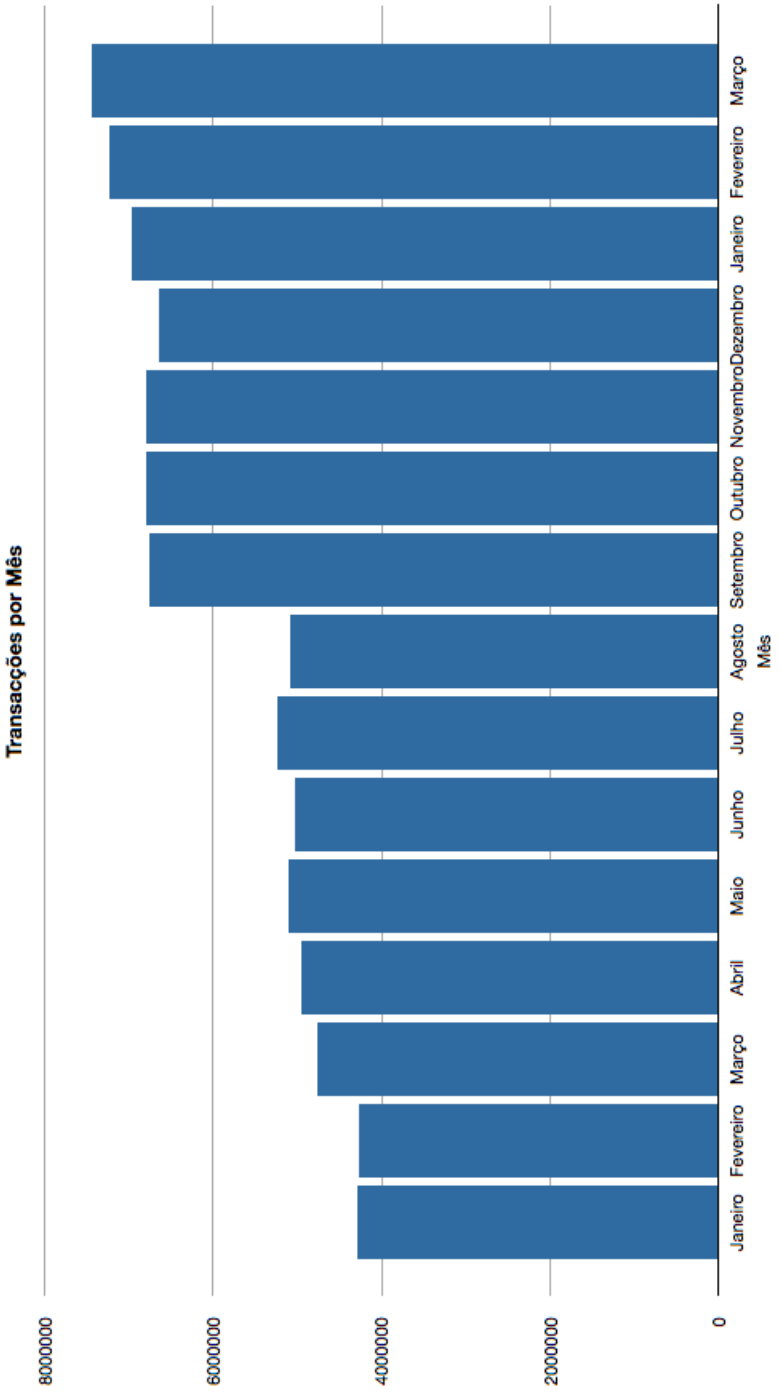


Figura 3.1: Número de transacções por mês

que as quase oito milhões de transacções geram à volta de oitocentos *Gygabytes* por mês, como se pode ver no gráfico 3.2, que mostra a evolução do volume de informação gerada, agrupada por mês.

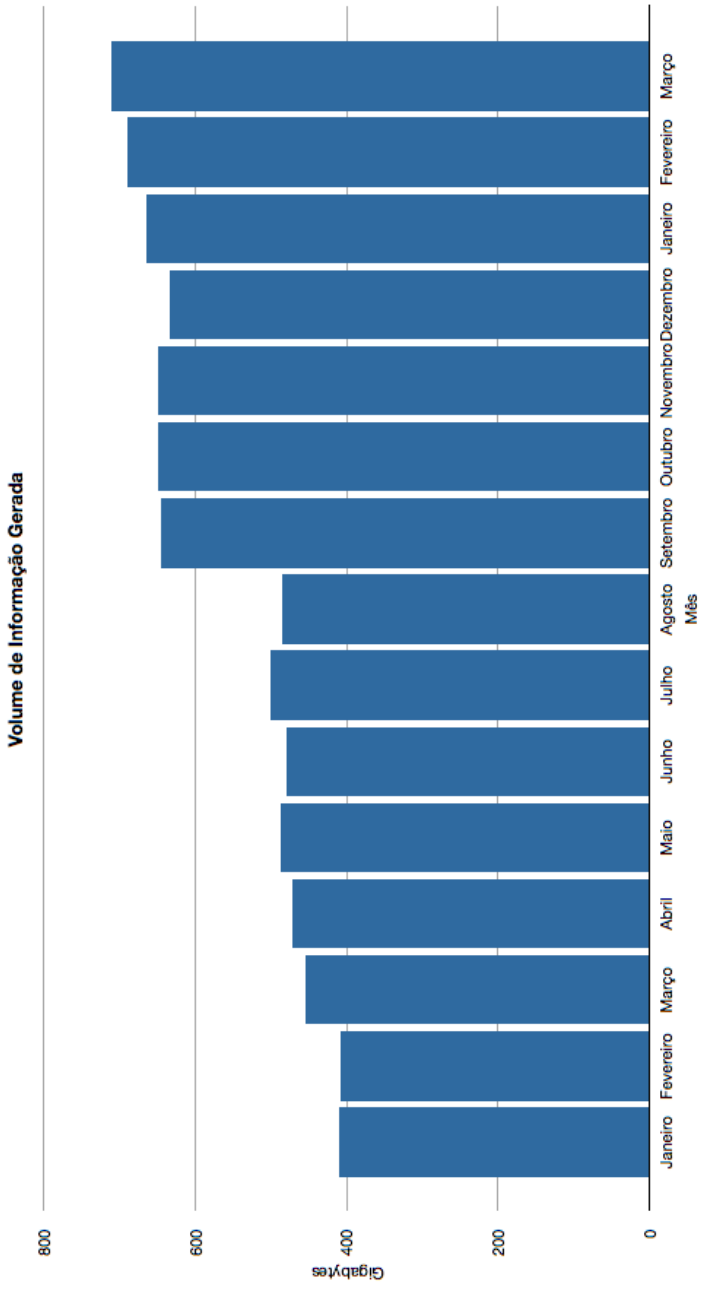


Figura 3.2: Volume de informação gerada por mês

3.1.2 Factura Electrónica

A factura electrónica é um dos serviços *core* que compõem a oferta de serviços da Saphety. Sendo este um serviço de vital importância, vamos debruçar-nos um pouco mais no detalhe e especificação do funcionamento. No entanto, por razões de confidencialidade (visto tratar-se de informação muito sensível da empresa), a informação apresentada foi objecto de algumas modificações para ocultar detalhes e assim salvaguardar informação confidencial.

Quando se fala de factura electrónica, fala-se de um amplo projecto de desmaterialização de documentos, onde constam ordens de compra, mensagens de status, facturas, notas de débito e crédito, entre outros. A imagem 3.3, mostra o fluxo de informação e a relação entre quem compra e quem vende.

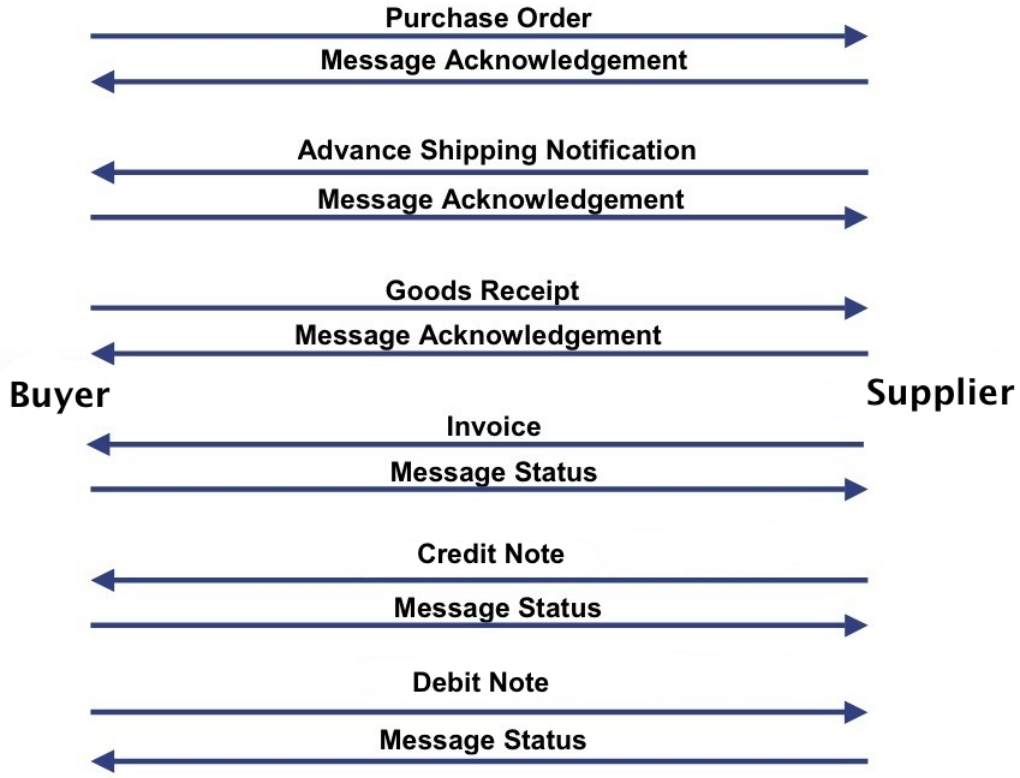


Figura 3.3: Integration Scenario

Como se pode ver na imagem, todos os documentos transaccionados geram um novo documento de estado de integração para o remetente inicial. Além destas mensagens, são gerados diversos registos pelo sistemas que vão processando estes documentos.

Este conjunto de documentos é constituído por:

- **Ordem de compra** Documento emitido por um comprador ao vendedor, indicando quantidades e preços de produtos ou serviços a adquirir.
- **Guia de remessa** Documento emitido quando são enviados os produtos encomendados, indicando as quantidades expedidas.
- **Factura** Documento comercial que representa os valores a serem cobrados por uma determinada transacção (venda).
- **Notas de débito e de crédito** São documentos comerciais emitidos com o intuito de acertar entre o comprador e vendador, valores não cobrados ou cobrados de forma errada, pode ser emitida para compensar a devolução de produtos, entre outros.

Todos estes documentos trocados no âmbito deste serviços são estruturados e têm formatos próprios aceites. O Edifact[3], xCBL[4] e IDOC[5] são alguns dos formatos aceites e que, depois de validados, são transformados em formatos internos da empresa e inseridos em base de dados. A imagem 3.4 contém uma estrutura de mapeamento de dados presentes nos documentos para uma base de dados.

[3]United Nations/Electronic Data Interchange For Administration, Commerce and Transport (UN/EDIFACT) - formato de texto para troca de documentos de negócio

[4]XML Common Business Library - Framework XML para criação de documentos de negócio

[5]IDoc é um documento criado pela SAP e assemelha-se a um XML utilizado para transferir documentos de negócio

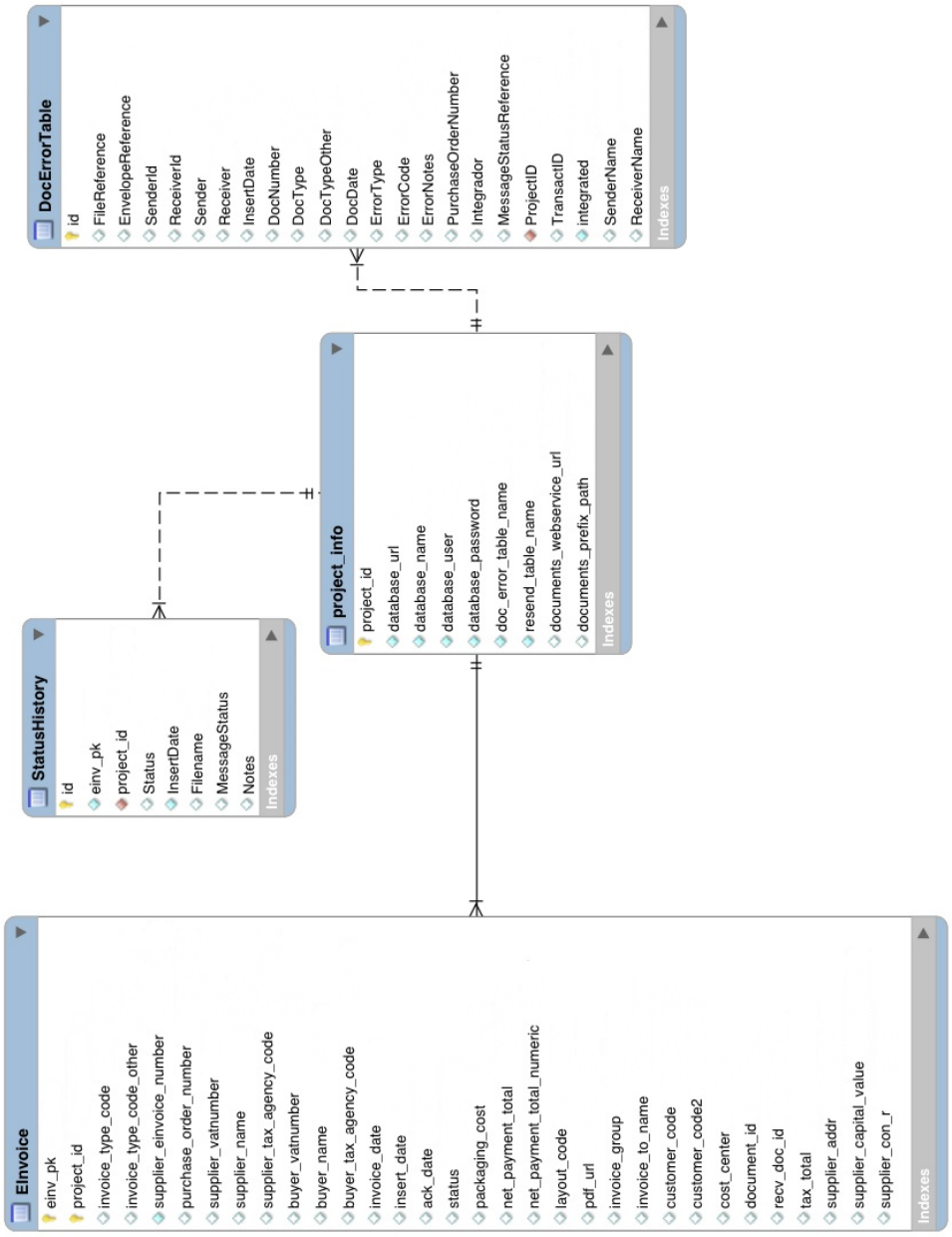


Figura 3.4: Diagrama ER EInvoice

3.2 Requisitos de negócio

Como foi referido anteriormente no documento, é requisito obrigatório existir uma plataforma de análise de informação de negócio, que terá que permitir ter no futuro toda a informação processada, analisada, gerando no final eventos e relatórios.

Devido ao crescimento obtido e aos planos de crescimento da empresa, a solução final terá de oferecer condições de crescimento horizontal, possibilitando assim um crescimento faseado da capacidade de processamento, acompanhando desta forma o crescimento do negócio.

O crescimento horizontal é factor importante devido a ser a política implementada na empresa e que envolve baixos custos em hardware.

Além do crescimento horizontal, a plataforma deverá permitir tratar dados não estruturados e semi estruturados juntamente com a informação guardada em bases de dados. Desta forma poder-se-á cruzar diversos tipos de informação e dados.

Capítulo 4

Ambiente de desenvolvimento e tecnologias adoptadas

4.1 Tecnologias adoptadas

A análise feita aos serviços, natureza dos dados gerados e requisitos de negócio levou a que fossem feitas algumas escolhas tecnológicas que respondessem aos desafios, não só deste trabalho, mas da empresa.

Na base da plataforma temos o sistema operativo e o sistema de storage. Foi escolhida a distribuição CentOS Linux 5.5, que é um clone da distribuição comercial RedHat Enterprise Linux (RHEL). Se a empresa quiser suporte comercial e planos de manutenção, assegurados pela RedHat, a mudança será simples visto o CentOS ser baseado em RHEL.

Para o ambiente de processamento da informação, foi escolhida a framework Apache Hadoop e usada a distribuição disponibilizada pela Cloudera, Inc. Com esta framework, utilizou-se diversos componentes que respondem às necessidades de crescimento e de escala:

- **HDFS** Hadoop Distributed File System, que é um sistema de ficheiros escalável, com garantias de integridade de informação, redundância e elevada rapidez de leitura para operações de processamento. Esta rapidez é garantida através de leitura e escrita sequencial em disco, eliminando os acessos aleatórios, que é um dos conhecidos factores que motivam a perda de performance. Este sistema de ficheiros está na base de todos os componentes do Hadoop, possibilitando a estes componentes a capacidade de processarem desde Gigabytes até Petabytes de informação em paralelo.
- **MapReduce**[6] é um modelo de programação e uma *framework* que permite escrever aplicações que processam largos volumes de dados. Esta *framework* assegura, além da execução em paralelo, mecanismos que lidam com a falha dos servidores e dos processos, reiniciando as tarefas. Garante também os mecanismos de comunicação entre os nós do *cluster*.[6]

- **Zookeeper** Serviço centralizado para gestão de configurações, nomes, disponibilizando mecanismos de sincronização distribuída. Peça de elevada importância para a coordenação de tarefas entre os nós do cluster e serviços que precisam executar tarefas em paralelo.
- **Pig** Linguagem de alto nível para criar programas de análise de dados, juntamente com uma infra-estrutura para testar os programas. O Pig pode executar estas tarefas em paralelo, permitindo assim tratar grandes volumes de dados.

  A vantagem de usar o Pig é ser uma linguagem de alto nível e aumentar a eficiência temporal. O código produzido é 5% do tamanho da versão produzida utilizando Java/-MapReduce e demora apenas 5% do tempo a ser produzido[12]. Contudo, as tarefas demoram mais tempo a serem executadas no Hadoop se forem feitas em Pig[12], por isso se forem tarefas a serem repetidas, o melhor é usar Pig para modelar os dados e testar conceitos, mas depois é necessário fazer o programa em Java para tirar mais proveito dos recursos à disposição.
- **Mahout** (`http://mahout.apache.org`) é uma biblioteca de algoritmos de *machine learning*, com especial ênfase na escalabilidade e utilização em ambientes cujo o volume a ser tratado é considerável. Esta biblioteca foi pensada desde o início para a plataforma *Hadoop* e assim poder usufruir da capacidade do MapReduce para execução de tarefas em paralelo. Todo o trabalho efectuado pelas diversas empresas envolvidas é mantido pela Apache Foundation, centralizando todo o esforço de desenvolvimento.

MapReduce

Devido ao MapReduce desempenhar um papel central em todo o sistema, é necessário compreender o modelo de programação. Para tirar partido deste modelo introduzido, não é necessário ter conhecimentos e experiência em sistemas distribuídos. Há dois passos principais no desenvolvimento de uma aplicação, que têm que ser seguidos: *map* e *reduce*,

O primeiro passo, *map*, recebe como *input* um conjunto de elementos, sob a forma de chave/valor e retorna uma lista de valores intermédios, também sob a forma de chave valor:

$Map(k1, v1) \rightarrow list(k2, v2)$

O segundo passo junta todos os valores intermédios que dizem respeito à mesma chave, para apresentar o resultado final:

$Reduce(k2, list(\text{v2})) \rightarrow list(v2)$

Tal como podemos verificar no artigo original do *MapReduce*[6], o algoritmo 1 é uma implementação simples do algoritmo para um contador de palavras.

A função map recebe o conteúdo do documento, separa em $tokens$ e emite para cada $token\ w$ o valor $(w, 1)$. A função $reduce$ por sua vez, soma os valores emitidos para cada

Algorithm 1 map and $reduce$ examples

Procedure: map(String key, String $value$):
  // key: document name
  // value: document contents
  for each word w in $value$ **do**
    $EmitIntermediate(w, 1)$;
  end for each

Procedure: $reduce$(String key, Iterator $values$):
  // key: a word
  // values: a list of counts
  $int\ result = 0$;
  for each v in $values$ **do**
    $result = result + ParseInt(v)$;
    $Emit(AsString(result))$;
  end for each

palavra.

Pode-se utilizar o algoritmo 1 descrito acima, para se contar palavras num conjunto de documentos de grande dimensão, para efectuar pesquisas de forma distribuída, contagem de frequência de acesso nas aplicações, entre outros.

4.2 Ambiente de Desenvolvimento

Para se poder desenvolver e testar os conceitos a aplicar, decidiu-se implementar e usar um ambiente de desenvolvimento baseado na virtualização de servidores, utilizando para este propósito o Linux 2.6, KVM(*Kernel-based Virtual Machine*) e LVM(*Logical Volume Management*).

KVM É um sistema de virtualização nativo para Linux que suporta a virtualização por hardware (extensões Intel VT ou AMD-V). Esta tecnologia consegue valores de performance muito próximos à utilização dos Sistemas Operativos directamente no hardware virtualizado. Está-se a tornar o padrão na virtualização OpenSource e é utilizada por grandes empresas como a RedHat no fornecimento de soluções a clientes, o que garante a estabilidade e continuidade desta tecnologia. Todos estes factores conduziram a que esta fosse a plataforma escolhida para a realização deste trabalho.

LVM É uma tecnologia que cria uma abstracção acima dos discos presentes no sistema, de forma a aumentar a flexibilidade. O uso desta tecnologia, aliada à virtualização, vem permitir de forma rápida alocar storage para máquinas virtuais, bem como redimensionar o espaço em storage alocado de forma a satisfazer as necessidades de crescimento. Esta tecnologia permite também a possibilidade de se tirar *snapshot* a uma partição,

que é uma cópia dos dados num determinado instante e que permite tarefas de backup, recuperação de dados e replicação.

Foram utilizadas as linguagens de programação Java 6, Perl 5.10 e Apache Pig para desenvolver algoritmos, ferramentas de ETL (*Extract Transform Load*)[1].

4.3 Criação de sistema de computação distribuída

A criação do sistema foi dividido em diversas partes: instalação e configuração de servidores, instalação do Apache Hadoop, criação dos nós de dados, criação dos nós de computação e configuração e uso do sistema de coordenação Zookeeper.

4.3.1 Servidores

Foram usados três servidores para serem os nós de computação, numa arquitectura *master/slave*. Um dos nós, demoninado *master*, é o responsável pela coordenação, distribuição de tarefas e apresentação de resultados, enquando que os outros, os *slave*, executam tarefas e respondem ao *master*. A imagem 4.1 ilustra a arquitectura de servidores adoptada.

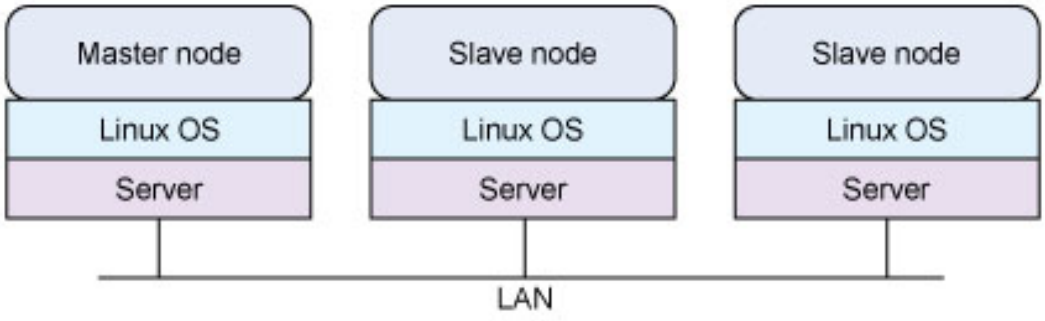


Figura 4.1: Cluster Hadoop

4.3.2 Arquitectura

Temos os seguintes componentes a funcionar nesta solução:

- **HDFS** Temos em primeiro lugar o cluster HDFS, responsável pela gestão dos dados.
- **MapReduce** - responsável por toda a coordenação das tarefas, recuperação automática e distribuição pelos nós do *cluster*.

[1]Utilizados para extrair os dados dos sistemas e importá-los para o cluster. Antes da importação é feita a normalização (transformação) dos dados.

.

HDFS

O cluster de HDFS é composto por um *NameNode* e diversos *DataNodes*. O *NameNode* servidor aplicacional é o responsável pela gestão de nomes e regula também o acesso aos ficheiros pelos sistemas clientes. É o *NameNode* que gere o processo de abrir, fechar e renomear ficheiros.

O *DataNode* é, por sua vez, um sistema aplicacional localizado em cada *slave*, responsável por gerir o acesso ao *storage* de cada nó, criando, apagando e replicando os ficheiros, após instrução do *NameNode*.

O processo de criação do cluster de *HDFS* está ilustrado na figura 4.2. O *NameNode* encontra-se a funcionar no *master* e os *datanode* nos *slave*. À medida que houver mais necessidade de armazenamento em disco e de poder de processamento, adiciona-se *datanodes* noutros *slaves*.

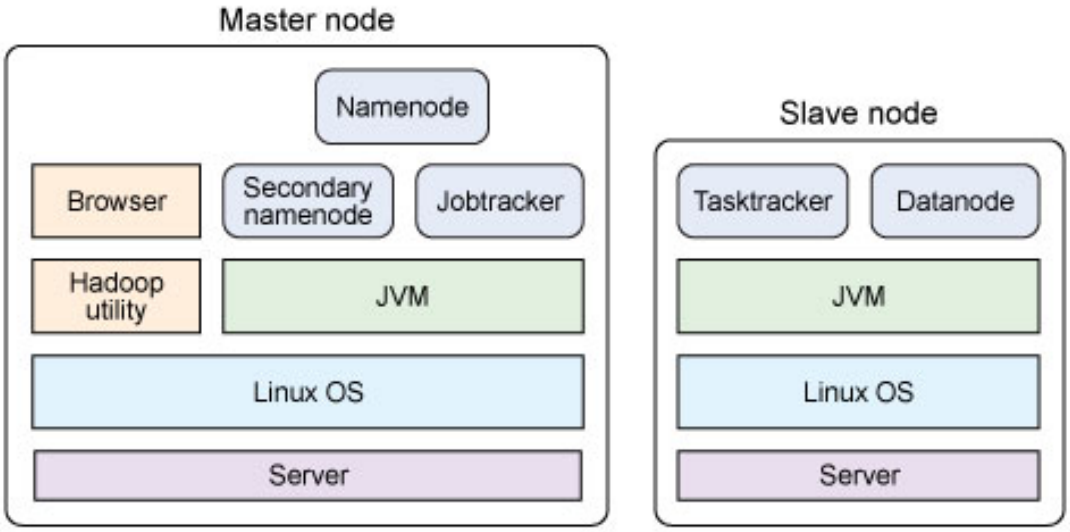


Figura 4.2: Serviços nos servidores

O sistema de monitorização do cluster, mas precisamente o *NameNode*, pode ser encontrado na imagem 4.3. Esta imagem é um espelho do actual estado do cluster onde se pode ver o número de nós, espaço ocupado, blocos replicados (para redundância).

Há também a funcionalidade de ver os registos e os ficheiros armazenados em disco.

NameNode 'master:54310'

Started:	Tue Apr 19 12:26:26 UTC 2011
Version:	0.20.2-cdh3u0, r81256ad0f2e4ab2bd34b04f53d25a6c23686dd14
Compiled:	Fri Mar 25 20:07:24 EDT 2011 by root
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

135 files and directories, 282 blocks = 417 total. Heap Size is 27.59 MB / 966.69 MB (2%)

Configured Capacity	:	137.8 GB
DFS Used	:	26.86 GB
Non DFS Used	:	7.42 GB
DFS Remaining	:	103.53 GB
DFS Used%	:	19.49 %
DFS Remaining%	:	75.13 %
Live Nodes	:	2
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	33

NameNode Storage:

Storage Directory	Type	State
/hadoop/dfs/name	IMAGE_AND_EDITS	Active

Figura 4.3: NameNode Details

MapReduce

Devido à filosofia do Hadoop ter o poder de computação junto dos dados, como foi anteriormente explicado, o MapReduce foi colocado junto aos nós de dados (*NameNode* e *DataNode*). O cluster de MapReduce é composto por um *JobTracker* e por diversos *TaskTracker*.

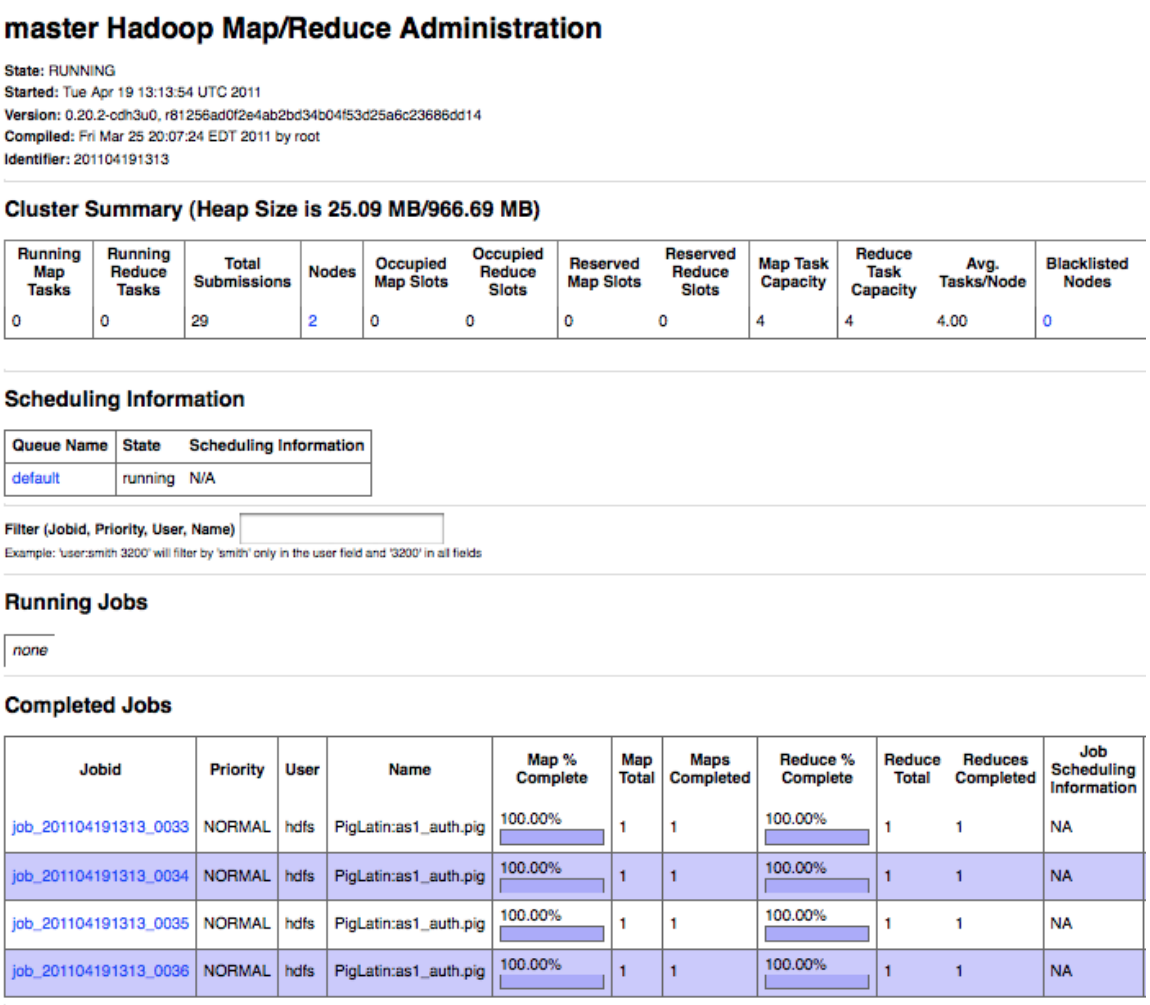
O *JobTracker* é o único ponto de onde se pode dar início à execução de aplicações *MapReduce*. É o responsável pela divisão da tarefa principal em subtarefas, atribuindo estas aos diversos *JobTrackers*, de acordo com o número de nós de computação disponíveis e os dados a serem utilizados.

O *TaskTracker* recebe as tarefas do *JobTracker* e os dados do *DataNode*, para assim poder fazer a computação que lhe foi atribuída e devolver o resultado ao *JobTracker*.

Tanto os componentes do *MapReduce* como também os do *HDFS* encontram-se nos mesmos nós de computação, tal como ilustrado na imagem 4.2.

A imagem 4.4 mostra a página de administração do *MapReduce*. Nesta página está sumarizada informação útil sobre as tarefas que são colocadas em execução, tarefas completadas, nós de computação activos, etc. Disponibiliza um panorama completo sobre o cluster, ajudando assim no processo de administração.

Nesta página podemos ver as tarefas em execução ou concluídas, podendo igualmente ver o detalhe de execução da tarefa, ilustrada na figura 4.5. Estes detalhes servem para se poder controlar onde a tarefa demora mais tempo a executar, tempos gastos por cada uma das sub tarefas, o total das subtarefas e volume de dados transaccionados, entre outros.



master Hadoop Map/Reduce Administration

State: RUNNING
Started: Tue Apr 19 13:13:54 UTC 2011
Version: 0.20.2-cdh3u0, r81256ad0f2e4ab2bd34b04f53d25a6c23686dd14
Compiled: Fri Mar 25 20:07:24 EDT 2011 by root
Identifier: 201104191313

Cluster Summary (Heap Size is 25.09 MB/966.69 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	29	2	0	0	0	0	4	4	4.00	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_201104191313_0033	NORMAL	hdfs	PigLatin:as1_auth.pig	100.00%	1	1	100.00%	1	1	NA
job_201104191313_0034	NORMAL	hdfs	PigLatin:as1_auth.pig	100.00%	1	1	100.00%	1	1	NA
job_201104191313_0035	NORMAL	hdfs	PigLatin:as1_auth.pig	100.00%	1	1	100.00%	1	1	NA
job_201104191313_0036	NORMAL	hdfs	PigLatin:as1_auth.pig	100.00%	1	1	100.00%	1	1	NA

Figura 4.4: Hadoop Map/Reduce Administration

Hadoop Job job_201104191313_0033 on History Viewer

User: hdfs
JobName: PigLatin:as1_auth.pig
JobConf: hdfs://master:54310/tmp/hadoop-mapred/mapred/staging/hdfs/.staging/job_201104191313_0033/job.xml
Job-ACLs: All users are allowed
Submitted At: 27-May-2011 08:40:20
Launched At: 27-May-2011 08:40:20 (0sec)
Finished At: 27-May-2011 08:40:41 (20sec)
Status: SUCCESS
Analyse This Job

Kind	Total Tasks(successful+failed+killed)	Successful tasks	Failed tasks	Killed tasks	Start Time	Finish Time
Setup	1	1	0	0	27-May-2011 08:40:20	27-May-2011 08:40:23
Map	1	1	0	0	27-May-2011 08:40:23	27-May-2011 08:40:29
Reduce	1	1	0	0	27-May-2011 08:40:30	27-May-2011 08:40:39
Cleanup	1	1	0	0	27-May-2011 08:40:40	27-May-2011 08:40:41

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	SLOTS_MILLIS_MAPS	0	0	9,868
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Launched map tasks	0	0	1
	Data-local map tasks	0	0	1
	SLOTS_MILLIS_REDUCES	0	0	9,887
FileSystemCounters	FILE_BYTES_READ	0	16,320	16,320
	HDFS_BYTES_READ	11,378,661	0	11,378,661
	FILE_BYTES_WRITTEN	184,506	184,406	368,912
	HDFS_BYTES_WRITTEN	0	13,332	13,332
Map-Reduce Framework	Reduce input groups	0	426	426
	Combine output records	426	0	426
	Map input records	216,735	0	216,735
	Reduce shuffle bytes	0	0	0
	Reduce output records	0	426	426
	Spilled Records	426	426	852
	Map output bytes	8,127,256	0	8,127,256

Figura 4.5: Hadoop Job Details

4.4 Criação de plataforma analítica

A plataforma analítica é algo que tem que ser desenvolvida e integrada nas ferramentas de análise e *reporting* da empresa. Contudo, por questões de testes e para se fazer prova da solução, desenvolveu-se as aplicações necessárias a tirar proveito do *MapReduce*, gerando relatórios e notificações a partir da junção de diferentes fontes de dados.

Para esta tarefa e resultante da análise dos dados, foram utilizadas as características tempo e tipo de informação na elaboração de relatórios.

Toda esta parte do trabalho foi feita recorrendo às linguagens de programação Java e Pig. A linguagem Pig, serve principalmente na modelação de dados. O código abaixo mostra a parte final do tratamento da informação que é gerada através da extracção de base de dados e ficheiros de logs, onde o Pig é utilizado para agrupar pela dimensão dia, gerando assim um relatório de actividade:

```
- - Load log file with login and acess times to each user/mailbox
raw = LOAD 'auth_log/auth.log-*' USING PigStorage('\t')
      AS (time, user, mailbox);

- - pass all username to lowercase
clean = FOREACH raw GENERATE time, org.uevora.mestrado.utils.ToLower(user)
        AS user, mailbox;

- - extract the hour from the time field
daily = FOREACH clean GENERATE org.uevora.mestrado.utils.ExtractDateDay(time)
AS day, user, mailbox;

- - grouped by day
daily_grouped = GROUP daily BY day;
daily_frequency = FOREACH daily_grouped GENERATE flatten($0), COUNT($1)
                  AS count;
daily_ordered = ORDER daily_frequency by count;
uniq_frequency = GROUP daily_frequency BY group::user;
```

4.4.1 Mahout

Na continuação da análise dos dados, foi introduzido o Mahout na plataforma e adicionado aos nós de processamento (*Datanode*). O Mahout veio ajudar a ter uma melhor percepção dos dados, complementando o código e os algoritmos criados.

Foram analisados os algoritmos presentes nesta biblioteca e a atenção recaiu principalmente sobre o FPG (*Frequent Pattern Growth*). Este algoritmo foi implementado para que pudesse ser executado de forma paralela, conhecido também como *Parallel FP-Growth* [9], e assim poder tirar partido da plataforma e cluster de Hadoop.

No ambiente criado, foi utilizado o algoritmo para ajudar a gerar as classes com os dados de acesso diário e semanal, utilizando a informação resultante do cruzamento das mailboxes com os logs das aplicações do *cluster* de *frontend*, tal como demonstrado na tabela 5.3.

Este algoritmo, pertence ao conjunto de algoritmos de regras de associação, tal como o Apriori. Estes algoritmos são utilizados para identificar itens que ocorrem de forma conjunta num determinado conjunto de dados[9]. A sua aplicação sobre os dados retorna um conjunto de itens (*itemset*) que ocorrem de forma frequente e o seu suporte (métrica explicada abaixo).

Seja $I = \{a_1, a_2, a_3, ..., a_m\}$ um conjunto de itens, temos que a base de dados de transacções, DB, constituída por um conjunto de sub-conjuntos de I, pode ser definida como $DB = \{T_1, T_2, ..., T_n\}$ onde cada $T_i \subset I (1 \leq i \leq n)$ é uma transacção[9, 5].

O suporte $supp(A)$ de um padrão, sendo $A \subset I$, é uma medida utilizada para definir as ocorrências mais significativas, porque é a contagem do número de vezes que um determinado *itemset* ocorre no conjunto de dados analisados. O padrão A passa a ser frequente se $supp(A) \geq \xi$, sendo ξ o mínimo de ocorrências aceite[9, 5].

O algoritmo FPG aceita como parâmetros a base de dados DB e o suporte ξ. No seu funcionamento, percorre a base de dados por duas vezes. Na primeira vez, gera uma lista de itens frequentes (F-List[2]) ordenados pela sua frequência. Na segunda vez, transforma a base de dados numa FP-Tree[3]. Após estar nesta árvore, o algoritmo começa a percorrer a FP-Tree para cada item cujo suporte seja superior a ξ e a construir a sua FP-Tree condicional. O processo de mineração do algoritmo é feita de forma recursiva na FP-Tree[9, 5]

[2]F-List denominação dada à lista ordenada por ordem decrescente de frequência dos itens presentes na base de dados

[3]É uma árvore de prefixos, onde é guardada informação crucial sobre os padrões

Capítulo 5

Testes e apresentação de resultados

O resultado esperado, de ter uma plataforma que pudesse centralizar a informação e gerar notificações, foi alcançado e está pronto para se começar a usar em larga escala.

Toda a informação gerada pela empresa, quando centralizada, e neste caso, esta plataforma permite que se possam cruzar dados de diferentes fontes. Ao utilizar as diversas fontes, os dados de negócio e de operação informática podem ser conjugados. Pode também incluir-se dados externos à organização com os dados de serviço.

A tabela 5.1 mostra um relatório resultante do cruzamentos de dados do serviço de *messaging*, mais precisamente mailboxes do serviço AS1, com dados de acesso aos servidores. A dimensão temporal utilizada é o dia, visto que ajuda a perceber a evolução diária e permite reportar em caso de falhas. O valor máximo indicado para normal operação é de 4000 transacções por segundo ou 5760000 por dia (valor calculado pela Saphety para o serviço em questão), o que demonstra que o serviço ainda tem muita margem de crescimento.

De qualquer forma, a tabela 5.1 contém os dados agrupados e não espelha a utilização de cada cliente. Por isso, podemos ter outro relatório, ver tabela 5.2, onde vemos, para um determinado dia, a mesma informação agrupada por dia e por cliente. O valor de referência para interligação dos clientes à Saphety, é de 120 transacções de acesso por hora, que dá 2880 acessos diários.

Neste caso são gerados alertas de serviço para os clientes presentes na tabela 5.2 e que estão a exceder aquele valor, o que pode interferir com o serviço prestado.

Tabela 5.1: Tabela de tentativas de acesso

Dia	**Acessos**
2010-10-01	262970
2010-10-02	268791
2010-10-03	247431
2010-10-04	256462
2010-10-05	252957
2010-10-06	261615
2010-10-07	264124
2010-10-08	260894
2010-10-09	253568
2010-10-10	250475
2010-10-11	316759
2010-10-12	319634
2010-10-13	326705
2010-10-14	318400
2010-10-15	304003
2010-10-16	277686
2010-10-17	260787
2010-10-18	296282
2010-10-19	329290
2010-10-20	315623
2010-10-21	304675
2010-10-22	317383
2010-10-23	302993
2010-10-24	299455
2010-10-25	307208
2010-10-26	322759
2010-10-27	329746
2010-10-28	333735
2010-10-29	335364
2010-10-30	311876

Tabela 5.2: Detalhe por cliente

2010-10-21	1838	cliente1
2010-10-21	1940	cliente2
2010-10-21	2160	cliente3
2010-10-21	2183	cliente4
2010-10-21	3281	cliente5
2010-10-21	3830	cliente6
2010-10-21	4629	cliente7
2010-10-21	5177	cliente8
2010-10-21	6498	cliente9
2010-10-21	7930	cliente10
2010-10-21	9360	cliente11
2010-10-21	13816	cliente12
2010-10-21	42674	cliente13
2010-10-21	56744	cliente14
2010-10-21	67680	cliente15

Para geração de alertas de possíveis problemas para as plataformas dos clientes, utilizou-se também a tabela 5.3, que foi gerada a partir do cruzamento de dados das mailboxes com todos os dados de acesso. Esta tabela contém o histórico de acesso, bem como a classe (colunas min e max) a que pertencem os dados. A classe foi obtida a partir da média do histórico de acesso, sem nenhum incidente, para se poder ter o valor frequente.

A última linha da tabela é utilizada para geração de uma notificação, devido aos valores estarem distantes do habitual para a mailbox em questão e também porque havia informação na mailbox (resultante do cruzamento). Esta informação é muito útil para empresa detectar sistemas de clientes que não se comportam como o esperado, alertando assim os clientes de problemas de forma que estes possam processar a informação que esperam em tempo razoável.

Tabela 5.3: Histórico e classes de acesso

Mailbox	**Date**	**Acess**	**Min**	**Max**
cliente12	2011-05-25	14918	12,570.3	15,363.7
cliente12	2011-05-26	14875	12,570.3	15,363.7
cliente12	2011-05-27	14620	12,570.3	15,363.7
cliente12	2011-05-28	14620	12,570.3	15,363.7
cliente12	2011-05-29	375	12,570.3	15,363.7

Como a rede da Saphety é aberta, significando por isso que um cliente pode comunicar com outro qualquer cliente, temos um conjunto enorme de possibilidades de interacção. Além da frequência de comunicação, com o FPG conseguiu-se relacionar as comunicações dos clientes.

O *output* encontrado de seguida na tabela 5.4, é de um cliente (cliente20) que apenas comunicou com outro cliente, o padrão encontrado foi que as comunicações ocorrem de forma frequente entre a Quarta e a Sexta Feira.

Tabela 5.4: Caso 1

Key	**Value**	**suporte**
client20		
	cliente21, cliente20	393
	Wednesday, cliente21, cliente20	125
	Friday, cliente21, cliente20	95
	Thursday, cliente21, cliente20	86

Na tabela 5.5, encontramos um outro caso. O cliente30 comunicou com outros dois clientes apenas e distribuídos por dois dias. Podemos utilizar estes dados para se gerar um alerta se por exemplo, ao fim do dia de Quarta Feira não tivermos registos de comunicações entre eles.

Tabela 5.5: Caso 2

Key	**Value**	**suporte**
client30		
	cliente30	268
	cliente31, cliente30	152
	cliente32, cliente30	116
	Monday, cliente30	86
	Wednesday, cliente30	130
	Wednesday, cliente32, cliente30	68
	Wednesday, cliente21, cliente30	62

Outro tipo de resultados explorados, foi relativo à performance. O ambiente utilizado para desenvolvimento do modelo não tem as características finais de servidores em produção, visto ser limitado em termos de memória e velocidade de acesso aos discos, o que interfere na leitura e na escrita.

Mesmo assim, os tempos são aceitáveis visto que se conseguiu processar informação a uma velocidade de 37MB por segundo, o que dá 15h para tratar 2 TeraBytes de informação neste ambiente. A figura 5, mostra um dos quadros resumo de resultados de execução de tarefas sobre um conjunto de dados de 3.5GB (divididos entre input e output).

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	517,227
	Launched reduce tasks	0	0	4
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Launched map tasks	0	0	41
	Data-local map tasks	0	0	41
	SLOTS_MILLIS_REDUCES	0	0	370,729
FileSystemCounters	FILE_BYTES_READ	26,964	5,478	32,442
	HDFS_BYTES_READ	2,579,490,625	0	2,579,490,625
	FILE_BYTES_WRITTEN	6,724,422	507,882	7,232,304
	HDFS_BYTES_WRITTEN	0	3,726	3,726
Map-Reduce Framework	Reduce input groups	0	138	138
	Combine output records	518	0	518
	Map input records	44,855,547	0	44,855,547
	Reduce shuffle bytes	0	6,012	6,012
	Reduce output records	0	138	138
	Spilled Records	518	182	700
	Map output bytes	1,255,955,316	0	1,255,955,316
	SPLIT_RAW_BYTES	15,471	0	15,471
	Map output records	44,855,547	0	44,855,547
	Combine input records	44,855,883	0	44,855,883
	Reduce input records	0	182	182

Figura 5.1: Resultados de benchmark

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusão

Os principais objectivos propostos para este trabalho eram:

- Criar um modelo de dados transaccionais, orientados para contextos de grandes volumes de dados, para depois serem analisados segundo técnicas de *business intelligence*.
- Aplicações para análise de grandes volumes de dados, suportadas em tecnologias emergentes orientadas para este tipo de cenários e assim permitir detectar anomalias (reais ou potenciais).

Estes objectivos propostos para a realização deste trabalho foram cumpridos, visto que se conseguiu criar o protótipo para a análise de dados pretendida pela empresa e validar os conceitos envolvidos. Os resultados são interessantes visto que a plataforma poderá acrescentar valor e significado à crescente informação gerada e assim contribuir para melhorar a qualidade do serviço prestado pela empresa, aumentando a sua vantagem competitiva.

A escalabilidade de um sistema deste género é um factor importante para o tipo de serviço que a Saphety quer. As aspirações de crescimento exponencial da empresa, a heterogeneidade de dados e de diversas fontes, conduziram a esta plataforma criada. Esta solução permitirá crescer de forma faseada, bastando juntar nós ao *cluster* e configurando os conectores das aplicações.

Do lado da Saphety, o interesse e a aposta foram essenciais para a realização deste projecto e após validação pela sua equipa de operações, vai-se continuar o desenvolvimento deste protótipo, passando a serviço e envolvendo o que for necessário para se ter esta plataforma analítica funcional.

A aprendizagem obtida é algo de difícil medida. A variedade tecnológica trazida para projecto, os *cases studies* analisados e a leitura complementar de livros e artigos contribuíram para

aumentar o conhecimento e aumentar os desafios tecnológicos, visto querer-se fazer sempre melhor.

Adicionalmente, as perspectivas de trabalho futuro abertas pelos conhecimentos adquiridos nas áreas envolvidas, constituem um factor de motivação e interesse acrescidos.

6.2 Trabalho Futuro

Uma área em que se pretende desenvolver algum trabalho, usando como ponto de partida o que foi desenvolvido no âmbito desta tese, é a da Competitive Intelligence. Esta pode ser definida, de forma muito lata, como um conjunto de acções que envolve a definição, compilação, análise e distribuição de informação sobre produtos, clientes, concorrentes e aspectos relativos ao ambiente de negócios, necessárias para suportar a tomada de decisões estratégicas pela gestão de uma organização (tipicamente, uma empresa).

Em termos concretos, é necessário recolher e organizar informação externa a uma organização, de forma a poder-se detectar riscos e oportunidades de mercado, antes que os mesmos sejam do conhecimento público generalizado. O trabalho desenvolvido no âmbito desta tese pode ser estendido, recorrendo a metodologias ainda mais avançadas de mineração e análise de dados, de forma a proporcionar ferramentas que permitam obter este tipo de resultados.

Bibliografia

[1] Jim Gray and Andreas Reuter, *Distributed Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993.

[2] Craig S. Mullins, *Database Administration: The Complete Guide to Practices and Procedures*, Addison Wesley, 2002.

[3] Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)*, John Wiley & Sons, 2002.

[4] T. Harding, R. Drummond and C. Shih, *MIME-based Secure EDI*, IETF, 2002.

[5] Jiawei Han, Jian Pei, Yiwen Yin and Runying Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, 2004.

[6] Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, Google, Inc, 2004.

[7] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes and Robert E. Gruber, *Bigtable: A Distributed Storage System for Structured Data*, Google, Inc, 2006.

[8] Zbigniew Michalewicz, Martin Schmidt, Matthew Michalewicz and Constantin Chiriac, Adaptive Business Intelligence, Springer, 2006.

[9] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang and Edward Chang, *PFP: Parallel FP-Growth for Query Recommendation*, Google Beijing Research, Peking University, Beijing, 2008.

[10] James Richardson, Kurt Schlegel, Rita L. Sallam and Bill Hostmann, *Magic Quadrant for Business Intelligence Platforms*. Gartner, 2009.

[11] Roland Bouman and Jos van Dongen, *Pentaho®Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL®* Wiley Publishing, Inc, 2009.

[12] Kevin Weil, *Hadoop, Pig, and Twitter (NoSQL East 2009)*, Twitter, Inc, 2009.

[13] Jeff Bean, *Considerations for Hadoop and BI: http://www.cloudera.com/blog/2010/05/considerations-for-hadoop-and-bi*, Cloudera Inc, 2010, acedido em 05-05-2011.

[14] The Apache Foundation, *HDFS Architecture. Available at http://hadoop.apache.org/common/docs/current/hdfs_design.pdf*, The Apache Foundation, 2010, acedido em 01-02-2011.

[15] Ashish Thusoo, Zheng Shao, Suresh Anthony, Dhruba Borthakur, Namit Jain, Joydeep Sen Sarma, Raghotham Murthy and Hao Liu. *Data Warehousing and Analytics Infrastructure at Facebook*, Facebook, 2010.

[16] Chuck Lam, *Hadoop in Action* Manning Publications, 2010.

[17] K. Meadors, Ed., *Electronic Data Interchange - Internet Integration (EDIINT)* IETF, 2010.

[18] Tom White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc, Second Edition, 2011.

[19] Splunk, *IT Silos Are Hurting Your Company: Make Machine Data a Strategic Asset*. Splunk, 2011.

[20] ParAccel, *The ParAccel Analytic Database: A Technical Overview*. Splunk, 2011.

[21] Shashank Tiwari, *Professional NoSQL*, John Wiley & Sons, Inc, 2011.