FACULDADES INTEGRADAS DE CARATINGA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL ANTONIO DINIZ

HADOOP PARA PROCESSAMENTO DE DADOS EM LARGA ESCALA

CARATINGA, 2012

Gabriel Antonio Diniz

HADOOP PARA PROCESSAMENTO DE DADOS EM LARGA ESCALA

Monografia apresentada à Faculdade de Ciência da Computação das Faculdades Integradas de Caratinga como exigência parcial da disciplina de Trabalho de Conclusão de Curso sob orientação do professor Glauber Luis da Silva Costa.

CARATINGA, 2012

Gabriel Antonio Diniz

HADOOP PARA PROCESSAMENTO DE DADOS EM LARGA ESCALA

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação das Faculdades Integradas de Caratinga como requisito para obtenção do grau de Bacharel.

_____

Prof. Glauber Luis da Silva Costa

Faculdades Integradas de Caratinga

_____

Prof. Msc Fabrícia Pires Souza Tiola

Faculdades Integradas de Caratinga

_____

Prof. Msc Jacson Rodrigues C. Silva

Faculdades Integradas de Caratinga

CARATINGA, 2012

DEDICATORIA

Dedico esse trabalho aos meus pais, por terem me dado dom da vida, e a meus irmãos que são os primeiros, e melhores amigos que tive na vida.

AGRADECIMENTO

Agradeço primeiramente Deus pela oportunidade oferecida, e pela força dada para conseguir chegar até aqui. Agradeço também a meus 7 companheiros de batalha Alessandro, Harrison, Luiz, Garcia, Brum, Wadson, e Weslley, amizades feitas para durar uma vida toda. Sem vocês lutando ao meu lado eu não estaria aqui hoje.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

RESUMO

Com o crescimento da Web 2.0, pode-se observar uma quantidade cada vez maior de dados circulando pela *Internet*. Comparados com os sistemas presentes na *Web* no início da década de 2000, as redes sociais, mecanismos de busca, portais de notícias, entre outros serviços *Web* da atualidade, possuem um volume de tráfego e armazenamento de dados enorme. As grandes bases de dados desses sistemas são conhecidas como *Big Data*.

Nesse contexto, os modelos de armazenamento de dados existentes não atendem a demanda desses grandes fluxos de informação. Para resolver esse problema, foram desenvolvidos soluções específicas para lidar com *Big Data*. O objetivo desse trabalho é analisar algumas dessas soluções. Usou-se o modelo de processamento distribuído de dados *MapReduce* desenvolvido pela Google, através da implementação de código fonte aberto desse modelo: o Hadoop.

Foram analisados dados a respeito das olimpíadas de Londres 2012 provenientes do Twitter, que é a ferramenta de *microblogs* mais utilizada na atualidade. A partir desses dados pode-se obter informações do tipo: tópicos mais comentados, usuários mais influentes, períodos onde mais se comentou determinados assuntos entre outras análises.

Assim esse trabalho apresenta a possibilidade do uso do Hadoop para análise de dados provenientes de redes sociais, gerando dados estatísticos sobre a repercussão de determinado assunto nesses meios de comunicação.

1. INTRODUÇÃO

Atualmente, com o crescimento da necessidade de se processar informação em quantidades cada vez maiores, algumas empresas investiram em novas tecnologias de armazenamento e processamento de dados. Google, Yahoo!, Facebook, entre outras gigantes da Internet se depararam com grandes volumes de informação a serem gerenciadas, processadas e disponibilizadas a seus usuários, e os modelos de armazenamento tradicionais já não atendiam mais essa demanda (BRITO, 2010).

Uma das soluções encontradas foi o modelo de programação chamado *MapReduce*, desenvolvido pela Google. Esse modelo possibilita se trabalhar com processamento paralelo de dados, abstraindo todas as rotinas que cuidam do paralelismo do processamento, deixando para o desenvolvedor cuidar apenas da lógica da tarefa propriamente dita. A forma como o *MapReduce* faz isso será abordada nas próximas seções desse trabalho.

Baseado no *MapReduce* a Google otimizou o processamento de dados em vários de seus serviços chegando a processar mais de 20 *petabytes*[1] de dados por dia nos seus *data centers* (DEAN & GHEMAWAT 2008). A implementação do *MapReduce* da Google não possui código fonte aberto, e não está disponível para o público em geral, porém com base na publicação de Jeffrey Dean e Sanjay Ghemawat sobre o modelo (DEAN & GHEMAWAT, 2004), outras implementações *open source* do *MapReduce* foram surgindo e ganhando força na comunidade de software livre.

A mais difundida dessas implementações é o Hadoop, que foi desenvolvido em 2004 por Doug Cutting e recebeu apoio de empresas como Yahoo!, e também da comunidade acadêmica. O Hadoop implementa o modelo *MapReduce* de forma análoga a do Google, tendo como objetivo facilitar o trabalho do desenvolvedor. Desde janeiro de 2008 o Hadoop é mantido pela *Apache Software Foudation* (WHITE, 2010).

As duas implementações do modelo *MapReduce* foram construídas de forma que

1 1 Petabyte equivale a 1024 Terabytes (http://dictionary.com/browse/petabyte)

seja possível usar hardware de baixo poder de processamento, e consequentemente de custo baixo. Apenas aumentando o número de máquinas que executam as tarefas, garante-se que o desempenho do conjunto como um todo também aumente. Utilizando maior quantidade de máquinas, pode-se ter uma maior distribuição dos dados, assim se uma máquina falhar, diminui-se a possibilidade de perda de dados, e diminui também o impacto no desempenho geral do conjunto (TAYLOR 2010; DEAN & GHEMAWAT 2008).

Tanto o *MapReduce* do Google quanto o Hadoop são construídos sobre um sistema de arquivos distribuído, que cuida da parte de replicação dos dados e da tolerância a falhas. Na implementação do Google ele é chamado de GFS (*Google File System*), e no Hadoop de HDFS (*Hadoop Distributed File System*).

Baseado no projeto Hadoop, foi criado o SGBD não relacional Hbase, classificado como um banco de dados orientado a família de colunas (DE DIANA e GEROSA, 2010; TAYLOR, 2010). Hbase também é mantido pela Apache (WHITE, 2010).

Assim como o Hbase, existem outros bancos de dados baseados no *MapReduce* como o projeto que deu origem a ele – o Google BigTable que é construído sobre o GFS – e o HadoopDB que é uma forma híbrida do Hadoop com o banco de dados relacional PostgreSQL (TAYLOR, 2010). Existem também outras alternativas que não são baseadas no Hadoop, mas são baseadas no conceito do Google BigTable, como os projetos HyperTable e Cassandra. E existem ainda outros bancos de dados não relacionais que utilizam outros modelos de armazenamento como o Redis, MongoDB, CouchDB, Neo4J, Voldemort, Dynamo, entre muitos outros (DE DIANA e GEROSA, 2010).

Diante desse cenário as quais novas tecnologias surgem com o propósito de resolver novos problemas, enxerga-se a necessidade de estudar as várias soluções criadas a partir dessas tecnologias aplicadas em problemas dos mais variados tipos, a fim de entender quais soluções são mais apropriadas a quais tipos de problemas, contribuindo tanto com a comunidade acadêmica, que poderá usar dessas soluções em pesquisas futuras, quanto para o mercado, que poderá aproveitar as experiências e resultados obtidos nesses estudos no desenvolvimento dos seus projetos.

Nesse trabalho, apresenta-se o uso do modelo *MapReduce* para análise de dados provenientes das redes sociais, através de sua implementação *open source,* o Hadoop.

Com essa análise é possivel extrair informações de uma base de dados não estruturada composta por dados coletados do Twitter a partir das publicações dos usuários do serviço a respeito das olimpíadas de Londres 2012. Busca-se extrair informações como: *hashtags* mais utilizadas, temas mais comentados, usuários mais populares (dentro do tema) entre outras. A escolha das olimpíadas se deve ao fato da grande repercussão desse evento esportivo na mídia mundial e consequentemente no Twitter.

Para isso, foi utilizado um *cluster*[2] em proporções reduzidas (10 nós), onde será configurado o Hadoop e o Hbase e serão armazenados os dados coletados, para assim poder processar os dados e encontrar as informações citadas acima. Para auxiliar na configuração do *cluster*, foi utilizado uma distribuição do Hadoop chamada CDH (Cloudera's Distribution Hadoop). Essa distribuição contém uma ferramenta chamada *Cloudera Manager* que facilita o processo de instalação, configuração e gerenciamento do *cluster*.

2 Cluster: uma série de coisas da mesma espécie, crescendo ou mantidas juntas (http://dictionary.com/browse/cluster). Na computação entende-se por *cluster* um conjunto de computadores ligados em rede trabalhando como se fossem uma única máquina de grande porte. Cada computador dentro do conjunto é chamado de nó (http://www.webopedia.com/TERM/C/cluster.html)

2. REFERENCIAL TEÓRICO

Nas seções a seguir serão abordados os principais assuntos pertinentes ao presente trabalho, fazendo uma contextualização e conceituação dos tópicos mais importantes dentro de cada assunto.

2.1 Big data

Pesquisas recentes mostram que o volume de dados armazenados nas empresas de grande porte e também o volume de dados recebidos pelas famílias norte-americanas estão atingindo níveis cada vez maiores. Segundo Xavier (2012) em 2010 as empresas americanas com mais de mil funcionários armazenavam em média 200 *terabytes* de dados, chegando ao patamar de um *petabyte* em alguns setores. E de acordo com a revista The Economist (2010) em 2008 as famílias americanas receberam em média cerca de 34 *gigabytes* de informação por pessoa a cada dia. Uma pesquisa da empresa Cisco estima que em 2013 o tráfego anual da Internet será de 667 *exabytes*[3] (THE ECONOMIST, 2010). O International Data Corporation (IDC, 2008) revelou que o tamanho de todo o universo de dados digitais em 2007 foi calculado como aproximadamente 281 *exabytes* e estimou que em 2011 esse patamar chegasse a 1,800 *exabytes*.

Diante dessa realidade de crescimento do volume de armazenamento e do tráfego de dados na Internet, surge o conceito de Big Data que pode ser definido de várias maneiras. Para o Gartner Goasduff, o termo está envolvido com três fatores: Volume, Variedade e Velocidade (PETTEY e GOASDUFF, 2011):

- Volume: está ligado ao tamanho da massa de dados, e a quantidade de tráfego envolvido.
- Variedade: refere-se ao grande número de diferentes fontes e formatos de

3 Exabyte: 1 *exabyte* equivale a 1024 *petabytes* (http://dictionary.com/browse/exabyte)

dados, já que os dados úteis são originários de dados tabulados, hierárquicos, documentos, e-mails, vídeo, imagens, transações financeiras, etc.

- Velocidade: envolve a velocidade que os dados são produzidos e a necessidade de processá-los de acordo com a demanda.

Collett (2011) complementa, afirmando que não são necessários os três fatores para se considerar que uma aplicação esteja lidando com Big Data, dependendo do caso, menores quantidades de dados que necessitam ser processados instantaneamente, ou grandes massas de dados em formatos não estruturados que precisam apenas ser gerenciados, já a caracterizam como tal. E ainda finaliza definindo as soluções Big Data como "a mineração de enormes volumes de dados estruturados e não estruturados de informações úteis, usando ferramentas não-tradicionais" (COLLETT, 2011). Taurion (2011) segue a mesma linha de raciocínio, defendendo que o conceito de Big Data é "juntar grandes volumes de informações estruturadas e não estruturadas para que as empresas tomem decisões baseadas em fatos". Outros autores são mais específicos, segundo Sevilla (2011), "Big Data é necessário quando o volume de dados cresceu tanto que a área de Tecnologia da Informação (TI) não consegue retornar informações básicas em menos de um minuto".

Para Jacobs (2009), um dos fatores que tem tornado o volume de dados realmente muito grande é o fato das empresas estarem cada vez mais gravando a repetição de informações simples através do tempo e/ou espaço como, por exemplo, operadoras de celular armazenando a posição dos seus celulares a cada 15 segundos, ou serviços de Internet que armazenam cada ação de cada um de seus usuários durante o uso dos seus serviços.

Para Xavier (2012), os principais desafios nas soluções Big Data estão em conseguir otimizar a utilização recursos tanto se tratando de software ou de hardware. Nesse sentido, utilização de *clusters* formados por hardware de baixo custo para o processamento paralelo no lugar de uma única máquina robusta tem se mostrado melhor na relação custo/benefício (JACOBS, 2009; DEAN & GHEMAWAT 2008). Quanto ao software, novos conceitos de armazenamento de dados fugindo a algumas restrições e regras clássicas do modelo relacional ganharam força e várias novas soluções de bancos de

dados surgiram caracterizando o início do movimento NoSQL (BRITO, 2010). Essas soluções serão abordadas no tópico 2.2.

As próximas seções apresentam uma explicação mais detalhada desse novo conceito de armazenamento e gerenciamento de dados denominado modelo não-relacional ou NoSQL, a sua comparação com o modelo relacional clássico e ainda uma explicação sobre vários diferentes conceitos de armazenamento dentro do modelo não-relacional.

2.2 Bancos de dados relacionais e não-relacionais

Durante muitos anos o modelo relacional foi amplamente utilizado para armazenamento de dados nos mais variados tipos de sistemas, sendo o modelo base dos SGBD (Sistemas de Gerencimento de Bancos de Dados) que ainda são líderes de mercado, como MySQL, SQL Server, Oracle, e PostgreSQL. Porém, com o crescimento cada vez maior do volume de dados de alguns sistemas, principalmente dos serviços disponibilizados na *Internet*, o modelo relacional passou a não atender totalmente às necessidades desses sistemas, e assim surgiram novos conceitos de armazenamento de dados. Os SGBD criados seguindo esses novos modelos são chamados de bancos de dados não relacionais, também conhecidos como NoSQL (BRITO, 2010).

Algumas características em comum agrupam diversos SGBD com os mais variados modelos de armazenamento na categoria de não-relacional ou NoSQL. Essas caracteristicas são: não-relacional, distribuído, de código aberto, escalável horizontalmente, ausência de esquema ou esquema flexível, suporte à replicação nativo e acesso via APIs simples (DE DIANA e GEROSA, 2010). Essas são as características mais comuns nos bancos de dados NoSQL, mas não é regra que um banco de dados precise seguir todas elas para ser considerado como tal (BRITO, 2010).

Uma característica essencial para um banco de dados ser considerado NoSQL é não seguir totalmente o padrão ACID do modelo relacional. Esse padrão define que uma transação deve ser Atômica, Consistente, Isolada e Durável. Isso quer dizer que o SGBD

deve se encarregar do controle de concorrência entre as transações, da consistência dos dados entre transações diferentes, e da consolidação de cada transação depois de seu término (STRAUCH, 2011).

As transações ACID tornam o trabalho de desenvolver aplicações muito mais simples, pois garantem a consistência e disponibilidade dos dados, porém fazem surgir dificuldades no particionamento dos dados, e na sua distribuição. Assim os bancos de dados não-relacionais tem que optar entre disponibilidade ou consistência. Dada a natureza das aplicações que utilizam esses bancos de dados, eles costumam optar por disponibilidade, e passam a tolerar alguma inconsistência temporária (DE DIANA e GEROSA, 2010). Assim é uma característica muito comum entre os NoSQL a consistência eventual dos dados, onde operações de inserção ou atualização tendem a ter um pequeno atraso em relação a leitura. Isso faz, por exemplo, com que uma operação de leitura de um dado que tenha acabado de ser atualizado retorne seu valor antigo durante um pequeno intervalo de tempo (BRITO, 2010).

Dentro do universo dos bancos de dados NoSQL vários modelos de armazenamento foram propostos. Dentre eles, alguns autores (DE DIANA e GEROSA, 2010; STRAUCH, 2011; LÓSCIO, DE OLIVEIRA e PONTES, 2011; BRITO, 2010) destacam como os principais: chave-valor, orientado a documentos, orientado a colunas e orientado a grafos. Nas próximas seções cada modelo será explicado detalhadamente e serão apresentados exemplos de implementação de cada modelo.

2.2.1 Modelo chave-valor (*key-value*)

É considerado o modelo mais simples, devido sua estrutura de armazenamento se basear em apenas duas informações: uma chave que identifica um registro e o valor correspondente a essa chave. Esse modelo é também conhecido como Tabela *Hash* distribuída, pois os dados são armazenados e localizados através de chaves (*hash*), utilizando o conceito de mapa ou dicionário de dados (DE DIANA e GEROSA, 2010).

O armazenamento de chave-valor não é uma novidade, já existem soluções implementando esse conceito há algum tempo, como por exemplo os produtos da linha Bekerley da Oracle (STRAUCH, 2011). Uma das principais implementações desse modelo é o DynamoDB (LÓSCIO, DE OLIVEIRA e PONTES, 2011). Outras opções são os bancos: Project Voldemort, Memcache, Redis, RIAK, Scalaris e Tokyo Cabinet.

2.2.1.1 DynamoDB

O DynamoDB é um dos bancos de dados usados em alguns dos vários serviços fornecidos pela Amazon em sua estrutura de milhares de servidores espalhados pelo mundo. Uma das suas principais características é a forma com que ele faz o balanceamento de carga entre os nós de um cluster, mesmo se os servidores envolvidos possuírem capacidades diferentes de processamento, cada um dos nós possui a mesma responsabilidade. Isso é garantido pelo fato de cada servidor possuir vários nós virtuais, onde cada nó virtual dentro do *cluster* possui uma capacidade de processamento fixa. Isso garante a divisão do trabalho de forma homogênia entre os nós envolvidos nas tarefas (STRAUCH, 2011).

2.2.1.2 Project Voldemort

Foi criado incialmente para o serviço de rede social LinkedIn (STRAUCH, 2011), mas atualmente possui licença de código aberto. Segundo Cattell (2010), os dados são particionados de forma automática através do conceito de *Consistent Hashing* onde os dados são replicados num número de vezes (definido pelo administrador do sistema) sendo que cada parte replicada fica em um servidor diferente. As atualizações são assíncronas, ou seja, não ocorrem no mesmo momento que solicitada, e nem ao mesmo tempo em todas as

replicações de uma mesma informação, fazendo com que o sistema não garanta a consistência dos dados. Quando novos servidores são adicionados ou removidos, o sistema automaticamente se adapta. Além disso, o gerenciador também detecta e recupera automaticamente falhas nos nós.

2.2.2 Bancos de dados orientado a documentos

Os documentos armazenados nos bancos de dados desse modelo são coleções de atributos e valores, sendo que um atributo pode conter mais de um valor. Bancos de dados orientados a documentos geralmente não possuem esquema, podendo cada documento armazenar atributos diferentes dos outros. Isso torna ideal para armazenamento de dados semi-estruturados, onde os dados não seguem uma mesma estrutura, mas pode-se identificar um padrão estrutural nos mesmos (STRAUCH, 2011).

Geralmente bancos de dados orientados a documentos armazenam dados em um formato que permite encapsular um documento dentro de outro, assim o valor referente a um determinado atributo de um documento pode ser outro documento. Isso leva a duplicação de dados, que por sua vez pode levar a problemas de consistência causados por exemplo por falhas na atualização ou deleção dos mesmos.

Os SGBD relacionais buscam a normalização dos dados para evitar esse problema, porém no modelo não relacional, outros fatores como o particionamento das tabelas entre vários nós são considerados mais importantes do que a normalização, então considerando que se perde a integridade referencial no particionamento do banco de dados, a falta da normalização e a consequente duplicação de dados não são encarados como um problema (BRITO, 2010).

Por outro lado, a duplicação de dados pode facilitar a distribuição do sistema, já que pode-se diminuir a quantidade de nós a serem consultados quando os dados relacionados estiverem próximos (Se os dados estiverem todos embutidos, a consulta pode ser feita usando apenas um nó). Esse fato aponta para uma característica presente não só nos bancos

de dados orientados a documentos, mas em bancos de dados NoSQL em geral, que é a criação de modelos de dados que favoreçam o seu uso para leitura (DE DIANA e GEROSA, 2010). Bancos de dados populares nessa categoria são o MongoDB e o CouchDB apresenrados no decorrer desse texto

2.2.2.1 CouchDB

Desenvolvido na linguagem Erlang, é mantido pela Apache desde 2008. A estrutura básica de armazenamento é um documento constituído de campos e valores. É possível fazer referências entre documentos, porém não há nenhuma garantia de consistência nem integridade referecial (BRITO, 2010).

A arquitetura do CouchDB foi projetada de forma que não exista o conceito de mestre e escravo, cada nó desempenha o mesmo papel no conjunto. Assim que um servidor localiza o outro na rede, inicia-se o processo automático de replicação. Outro diferencial esta no fato de que as consultas podem ser paralelizadas entre diversos nós utilizando o mecanismo de Map/Reduce (DE DIANA e GEROSA, 2010).

2.2.2.2 MongoDB

Cattell (2010) define o MongoDB como sendo similar ao CouchDB, ressaltando que as operações de leitura e escrita são atômicas. O formato de armazenameto dos dados é o BSON3, que é muito semelhante ao JSON[4] . Tem suporte ao armazenamento de dados binários podendo ser usado para guardar arquivos de áudio e vídeo.

O fato de suportar consultas parecidas com SQL, ter um controle parcial de integridade referencial e a modelagem de dados ser feita de forma análoga ao modelo

4 "JavaScript Object Notation", é um formato para intercâmbio de dados computacionais

relacional (utilizando o conceito de tabelas e colunas), faz do MongoDB uma boa opção para migração entre aplicações que usam o modelo relacional para NoSQL (TIWARI, 2011).

Entre as empresas que se destacam na utilização do MongoDB está a Foursquare que o utiliza nos seu serviços de check-in[5]. Os motivos principais para adoção do MongoDB no Foursquare foram o modelo flexível dos dados (semi-estruturado) e o ganho de performance na leitura e escrita dos dados comparando com ferramentas que seguem outros modelos de armazenamento (TIWARI, 2011).

Apesar de não existir uma linguagem nativa para consulta aos dados, possui bibliotecas clientes em várias linguagens, entre elas: C, C#, Java, JavaScript, Perl, PHP, Python e Ruby.

2.2.3 Bancos de dados orientado a grafos

O modelo de grafos possui três elementos básicos: os vértices dos grafos (nós), as arestas (relacionamentos) e os atributos (propriedades dos nós e dos relacionamentos) (LÓSCIO, DE OLIVEIRA e PONTES, 2011). Os dados são representados por grafos dirigidos, onde cada nó está relacionando com vários outros nós através das arestas.

Para De Diana e Gerosa (2010), este padrão está ligado diretamente ao modelo de grafos, onde os dados são representados como grafos dirigidos ou estruturas que generalizam a noção de grafos. O padrão suporta restrições através de integridade referencial.

Segundo Angles e Gutierrez (2010), o uso do modelo orientado a grafos é mais interessante quando informações sobre a relação e conectividade entre os dados é mais importante, ou tão importante quanto, os dados propriamente ditos. Os principais exemplos dessa categoria são o Neo4j e o OrientDB.

5 https://foursquare.com/about/

2.2.3.1 Neo4j

Segundo Vicknair, et al. (2010), o Neo4j foi desenvolvido em 2005 na linguagem Java e é mantido desde então pela empresa Neo Technology sob licença open source para uso não comercial. O mecanismo de busca do Neo4j foi construído utilizando o Lucene, um mecanismo de busca textual de alta performance desenvolvido e mantido pela Apache.

Um diferencial do Neo4J é que existe a garantia de parte das propriedades ACID. Outra vantagem está na economia de processamento e baixo tempo de resposta das informações relacionadas, isto devido ao processo de busca em profundidade a partir de um vértice específico (VICKNAIR et al., 2010).

2.2.3.2 OrientDB

Possui características tanto de orientado a documentos quanto de orientado a grafos. Assim como o Neo4j, tem suporte a operações ACID e pode trabalhar tanto com dados estruturados ou semi-estruturados. Devido ao fato de suportar consultas SQL, facilita a migração de programadores acostumados com o modelo relacional, isso tem feito seu uso se tornar bastante popular (BRITO, 2010).

2.2.4 Bancos de dados orientado a colunas

Segundo Lóscio, De Oliveira e Pontes (2011), nesse modelo, a indexação dos dados é feita através de tres informações: linha, coluna e *timestamp*. Linha e coluna são identificadas por chaves e o *timestamp* permite diferenciar as várias versões de um mesmo dado. Também existe o conceito de família de colunas (*column family*) que é utilizado para

agrupar colunas de mesmo tipo.

Os principais bancos nessa categoria são: BigTable, Hypertable, HBase e Cassandra. Nas próximas seções serão apresentadas suas principais características.

2.2.4.1 BigTable

O BigTable da Google, criado em 2004, foi uma das primeiras implementações de um sistema não relacional com o objetivo de promover maior escalabilidade e disponibilidade (BRITO, 2010). Ele foi construído sobre a plataforma Google File System (GFS) e é utilizado em mais de 60 produtos da empresa entre eles: Google Analytics, Orkut e Google Earth (PADHY, PATRA e SATAPATHY, 2011).

Para Orend (2010), o BigTable possui um conceito semelhante ao modelo de armazenamento chave-valor, com a diferença de mapear linha, coluna e *timestamp*, ao invés de somente uma chave com um valor.

Segundo Strauch (2011), os dados são gravados em unidades de armazenamento chamados de *tablets*. Cada *tablet* é a partição de uma tabela em um conjunto de linhas. O *timestamp* é utilizado para diferenciar as versões de uma célula de valor, além disso, também é utilizado para ordenar as alterações dos valores, mantendo o valor mais atual como sendo o primeiro a ser lido. O GFS é um sistema de arquivos distribuído que é utilizado para a persistência dos arquivos e dos logs. Segundo Orend (2010), o particionamento é realizado relacionando um *tablet* a um único nó. Um servidor principal cuida de mapear a relação entre o *tablet* e o nó onde ele está armazenado. Dessa forma, a replicação dos dados não é feita pelo BigTable e sim pelo GFS. Por não existir replicação nativa, não existe o problema na consistência dos dados, pois esse processo é garantido pelo GFS de forma transparente ao BigTable.

O BigTable não suporta junções, nem integridade referencial, assim as relações entre as tabelas devem ser tratadas pela aplicação. A API é fornecida na linguagem C++.

2.2.4.2 Cassandra

O Cassandra foi escrito em Java por um engenheiro do Facebook e um engenheiro do projeto Dynamo inicialmente para atender apenas as necessidades do próprio Facebook (CATTELL, 2010). Porém, em 2008 foi licenciado pela Apache e tornou-se código aberto sendo mantido pela Apache Foundation, mas ainda recebendo colaboração dos desenvolvedores do facebook (BRITO, 2010).

A detecção de falhas no sistema e o processo de recuperação dessas falhas é feia de forma automatizada, assim como a detecção de novos nós no *cluster*. Um dos pontos fracos do sistema é o sistema de concorrência, que não possui travas durante o acesso aos dados e as réplicas não são atualizadas de forma síncrona, o que pode gerar inconsistência nos dados (CATTELL, 2010).

Ao acessar ou modificar os dados de uma linha, idependente da quantidade de colunas que se deseja recuperar ou gravar, esse acesso é feito de forma atômica por linha, ou seja, sempre envolve toda a linha no processo. Quando uma requisição é feita, o sistema identifica qual nó possui a informação e faz a requisição a ele. Para otimizar a performance, o sistema armazena índices de colunas e família de colunas para evitar leitura desnecessária no disco durate a recuperação da informação (LÓSCIO, DE OLIVEIRA e PONTES, 2011).

A API do Cassandra está disponível em Java, Ruby Phynton, C#, Thrift entre outras (BRITO, 2010).

2.2.4.3 HBase

É um projeto da Apache escrito em Java e que foi patenteado logo após o BigTable. Sua principal diferença está em utilizar a técnica de Map/Reduce, sob o sistema de arquivo do Hadoop (HDFS), ao invés do GFS. A caso de sucesso na utilização do HBase é no

sistema de mensagens em tempo real construído para o Facebook em 2010 (OREND, 2010). Uma explicação mais detalhada sobre esse SGBD será feita no decorrer da seção 2.4 onde é feita uma análise do ecossistema Hadoop.

2.3 MapReduce

Os bancos de dados NoSQL resolveram grande parte do problema de armazenamento de grandes volumes de dados ao se lidar com *big data*, isso se deve a algumas características comuns entre eles, a maioria é: distribuído, escalável, não possui esquema ou possui esquema flexível, e suporte nativo a replicação (DE DIANA & GEROSA, 2010). Porém, quando além do armazenamento e gerenciamento de grandes bases de dados, existe a necessidade de constante processamento e análise desses dados, outros conceitos surgiram para tornar essa tarefa mais simples e rápida. O principal conceito nesse sentido é o *MapReduce*.

MapReduce é um modelo desenvolvido pela Google para processamento distribuído de grandes quantidades de dados. Ele permite que todo o trabalho seja feito a partir de duas funções básicas definidas pelo usuário: *map* e *reduce*, e se encarrega de abstrair toda parte de paralelização do processamento, balanceamento de carga entre as máquinas e tolerância a falhas na leitura ou escrita dos dados, como falhas em software e ate mesmo no hardware (DEAN & GHEMAWAT 2004).

Map é responsável por gerar uma série intermediária de pares chave/valor a partir de um conjunto de dados de entrada. *Reduce* é responsável por juntar todos os valores intermediários associados a uma mesma chave intermediária gerando a solução final (DEAN & GHEMAWAT 2004). Para ilustrar o funcionamento do modelo vamos demonstrar uma rotina de contagem de palavras repetidas em um texto usando o modelo *MapReduce.* Considere o seguinte arquivo de texto simples como entrada:

Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet

Quadro 2.1: Arquivo entrada.txt

Supondo que a tarefa seja divida em duas partes, cada subtarefa fica responsável por metade do arquivo:

Lorem ipsum dolor sit amet Lorem ipsum

Quadro 2.2: Primeira metade do arquivo

dolor sit amet Lorem ipsum dolor sit amet

Quadro 2.3: Segunda metade do arquivo

A função *Map* nesse caso contaria o número de palavras repetidas em cada metade do arquivo, retornando os seguintes conjuntos intermediários:

Lorem, 2 ipsum, 2 dolor,1 sit, 1 amet, 1

Quadro 2.4: contagem da primeira metade

Lorem, 1 ipsum, 1 dolor, 2 sit, 2 amet, 2

Quadro 2.5: contagem da segunda metade

Os conjuntos intermediários são então enviados para a funçao *Reduce*, que une as informações vindas das duas tarefas e retorna o resultado final:

Lorem {2 ,1}
ipsum {2, 1}
dolor {1, 2 }
sit {1, 2 }
amet {1, 2 }

Quadro 2.6: União dos resultados intermediários

Lorem, 3
ipsum, 3
dolor, 3
sit, 3
amet, 3

Quadro 2.7: Resultado final da contagem

Seguindo esse modelo, é possível abstrair uma enorme gama de problemas relativos ao processamento de dados brutos, transformando-os em informação útil.

A implementação do modelo *MapReduce* criada pela Google é proprietária, e seu código não está disponível ao público. Porém, existem implementações *open source* do modelo, a mais popular delas é o Hadoop, um projeto mantido pela Apache Software Foudation (LEE et al. 2011).

2.4 Ecossistema Hadoop

As soluções “Big Data” encontram na computação o meio para criar implementações capazes de gerenciar e analisar bases de dados de proporções cada vez maiores, realizar cálculos complexos em cima delas, identificar padrões e, a partir dos resultados obtidos, serem capazes de oferecer melhorias nos seus serviços, ou disponibilizar serviços personalisados aos seus usuários. Porém, quase sempre esbarram no poder computacional das máquinas atuais. Mesmo com a evolução no poder de processamento dos recursos computacionais atuais, eles não conseguem acompanhar o crescimento da necessidade de se gerenciar e analisar *big data* (GOLDMAN et. al, 2012).

Devido a grande demanda por capacidade computacional e por restrições físicas das arquiteturas convencionais, a computação paralela e distribuída desponta como alternativa para contornar os maiores desafios computacionais. Assim, esse modelo de computação vem desempenhando papel fundamental na mineração de informações úteis nas bases de dados de aplicações *big data*. Geralmente essa computação é realizada em *clusters* onde um aglomerado de computadores comuns trabalhando em conjunto, conseguem um alto poder de processamento a um custo final consideravelmente mais baixo (DEAN & GHEMAWAT 2008).

Apesar do fato de a computação distribuída acenar como uma das soluções mais viáveis para resolver os problemas encontrados ao se trabalhar com *big data*, essa ferramenta não é vista como de uso confortável pela maioria dos desenvolvedores. Dividir tarefas complexas em várias tarefas menores e então executar cada uma delas em uma unidade de processamento diferente de forma paralea e concorrente não é uma procedimento trivial. Além disso, se for mal dimensionado, o tamanho e a divisão das tarefas poderão comprometer consideravelmente o desempenho da aplicação. E para complicar ainda mais, o desenvovedor precisa: analisar a interdependência dos dados da aplicação; desenvolver um algoritmo de balanceamento de carga e de escalonamento para as tarefas garantindo a otimização do uso dos recursos computacionais; e, garantir a recuperação ou a não interrupção de uma tarefa no caso de alguma falha em uma parte do processo (LEE et al., 2011).

Foi nesse contexto que foi desenvolvido o Apache Hadoop, um framework para o processamento de grandes quantidades de dados em *clusters*. A ideia de promover soluções para os desafios encontrados ao se trabalhar com sistemas distribuídos em um único framework é o ponto central do projeto Hadoop (WHITE, 2010). No ambiente Hadoop, problemas como disponibilidade dos dados, falhas de hardware, escalabilidade da aplicação e balanceamento de carga são tratados de forma transparente ao usuário. Além disso, seu sistema de armazenamento de dados, e seu modelo de execução de tarefas o torna consideravelmente mais eficiente que outras soluções similares (GOLDMAN et. al, 2012). Atualmente, além do sucesso no mundo empresarial, o Apache Hadoop também tem obtido crescente apoio da comunidade acadêmica, sendo usado nas mais diversas áreas

de pesquisa como ferramenta de processamento de grandes bases de dados (RAO & REDDY, 2011; TAYLOR, 2010; GOLDMAN et. al, 2012).

2.4.1 Histórico de desenvolvimento do Hadoop

Desde sua origem até hoje, o Hadoop apresentou uma rápida evolução. Goldman e parceiros (2012) fizeram um levantamento detalhado do histórico da ferramenta. A seguir um resumo de alguns dos fatos marcantes levantados nesse trabalho que nortearam a evolução do Hadoop, como é hoje conhecido.

- Fevereiro de 2003: Jeffrey Dean e Sanjay Ghemawat, dois funcionários da Google, desenvolvem a tecnologia *MapReduce*, para otimizar a indexação e catalogação dos dados sobre as páginas Web e suas ligações.
- Outubro de 2003: Sanjay Ghemawat junto com Howard Gobioff e Shun-Tak Leung (dois outros funcionarios da Google), publicam um artigo intitulado The Google File System (GHEMAWAT, GOBIOFF, & LEUNG, 2003) que descreve o chamado Google File System (GFS ou ainda GoogleFS).
- Dezembro de 2004: Sanjay Ghemawat e Jeffrey Dean, publicam o artigo Simplified Data Processing on Large Clusters (DEAN & GHEMAWAT, 2004), agora sobre o modelo de programação *MapReduce*, desenvolvido por eles em 2003.
- Dezembro de 2005: o consultor de software Douglas Cutting divulga uma implementação *open source* do *MapReduce* e do sistema de arquivos distribuído com base nos artigos do GFS e do *MapReduce* publicados pelos funcionários da Google.
- Fevereiro de 2006: a empresa Yahoo! contrata Cutting e investe para que ele desse andamento a seu projeto de código aberto. Nesse mesmo ano, o projeto recebe o nome de Hadoop - nome do elefante amarelo de pelúcia do filho de Cutting.

- Abril de 2007: Yahoo! anuncia ter executado com sucesso uma aplicação Hadoop em um aglomerado de 1.000 máquinas. Também nessa época, o Yahoo! passa a ser o maior contribuinte no desenvolvimento do framework.
- Janeiro de 2008: o Apache Hadoop, que já se encontrava na versão 0.15.2, tem lançamento constante de versões com correção de erros e novas funcionalidades e, nesse mês, se transforma em um dos principais projetos da Apache.
- Julho de 2008: uma aplicação Hadoop em um dos *clusters* do Yahoo! quebra o recorde mundial de velocidade de processamento na ordenação de 1 terabyte de dados. O aglomerado era composto de 910 máquinas e executou a ordenação em 209 segundos, superando o recorde anterior que era de 297 segundos.
- Dezembro de 2011: seis anos após seu lançamento, o Apache Hadoop disponibiliza sua versão 1.0.0. versão com maior confiabilidade e estabilidade em escalonamentos.
- Maio de 2012: Apache faz o lançamento da primeira versão da série 2.0 do Hadoop.

Na Figura 2.1 (GOLDMAN et. al, 2012) é apresentado um cronograma no formato linha do tempo do lançamento de todas as versões disponibilizadas do Hadoop.

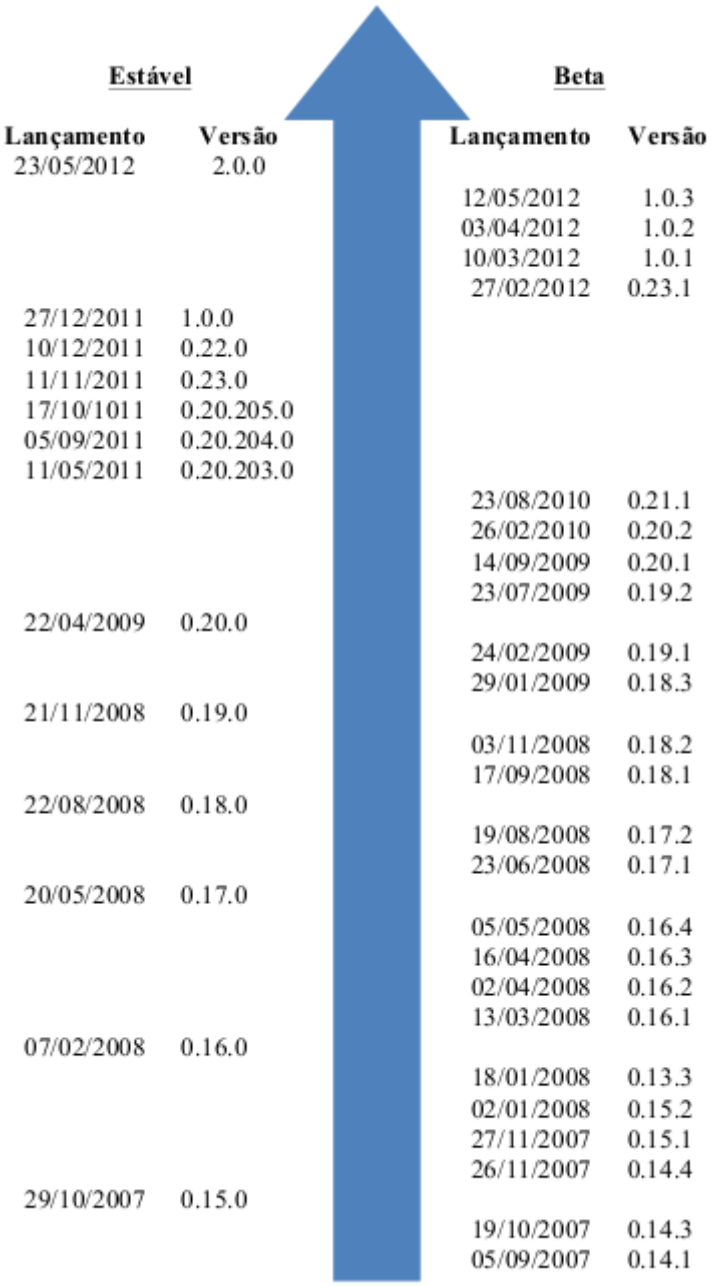

Figura 2.1: Lançamento das versões do Apache Hadoop

2.4.2 Quem utiliza o Hadoop

Uma prova da eficiência do Hadoop está na grande quantidade de importantes empresas que o utilizam nos seus parques computacionais para os mais variados fins. Na tabela 2.1 é apresentada uma compilação das principais empresas com seus respectivos números de parque computacional e atividade em que o Hadoop está empregado (as

informações foram retiradas do site oficial do projeto Hadoop[6] onde há uma lista extensa de outras empresas):

Empresa	***Parque***	***Atividade***
Adobe	~80 nós	Processamento de dados internos e de redes sociais
E-Bay	~532 nós	Otimização de buscas
Facebook	~1400 nós	Análise de logs
Last.fm	~64 nós	Análise de logs, análise de perfil de usuário, outros
LinkedIn	~1900 nós	Análise e busca na similaridade de perfis de usuários
The New York Times	Externo (Amazon)	Conversão de imagens, armazenamento de jornais digitais.
Twitter	Não divulgado	Armazenamento de mensagens, processamento de informações
Yahoo!	~40.000 nós	Processamento de buscas, recomendações de publicidade, testes de escalabilidade.

Tabela 2.1: Empresas que utilizam Hadoop

Apesar dos números não serem exatos, é possível visualizar o tamanho da infra-estrutura em que essas empresas utilizam o Hadoop, e também a diversidade de áreas onde ele vem sendo usado.

2.4.3 Vantagens no uso do Hadoop

Atualmente o Apache Hadoop é considerado uma das melhores ferramentas para processamento de grandes quantidade de dados. Entre os benefícios que seu uso pode trazer, Taylor (2010) destaca os seguintes:

- Código aberto: Todo software livre de sucesso tras consigo uma série de

6 http://wiki.apache.org/hadoop/PoweredBy

valores agregados como comunidade ativa de usuários e desenvolvedores – no caso do projeto Hadoop essa comunidade conta com a participação de várias empresas importantes, como as citadas no item 2.4.2. Um maior número de desenvolvedores implica em uma capacidade maior de encontrar falhas e corrigi-las mais rapidamente.

- Economia: Por ser software livre não há o custo de pagamento de licenças de software. Outra economia está no fato dele ser projetado para trabalhar com hardware comum, que é potencialmente mais barato. E além disso, pode-se ainda optar por contratar um dos serviços de computação em núvem disponíveis, onde pode-se executar aplicações Hadoop alugando um *cluster* virtual, ou pagando por tempo de processamento utilizado.
- Robustez e confiabilidade: Como o Hadoop foi projetado para trabalhar com hardware comum, ele prevê possíveis falhas nesses equipamentos, e possui estratégias internas para recuperação dessas falhas, como replicação de dados e armazenamento de informações sobre a tarefas que estão sendo executadas. Isso garante a continuidade das tarefas mesmo ocorrendo alguma falha em uma parte do processo.
- Escalabilidade: Hadoop foi projetado para que o esforço para configurar um parque computacional com 1000 máquinas não seja muito maior do que em um com 20 máquinas (WHITE, 2010). O aumento da capacidade de armazenamento e processamento é limitado apenas pela quantidade e capacidade de armazenamento e processamento dos equipametos utilizados.
- Simplicidade: Hadoop auxilia o desenvolvedor retirando dele a responsabilidade sobre as complexas questões relativas à computação paralela (gerenciamento de falhas, escalonamento, balanceamento de carga, etc.). Todas as operações são abstraídas pelo desenvolvedor em apenas duas funções: uma de mapeamento (*Map*) e uma de junção (*Reduce*). Dessa forma o foco é mantido somente na abstração do problema ao modelo *MapReduce*.

2.4.4 Desvantagens no uso do Hadoop

Por se tratar de uma ferramenta recente e em evolução, algumas funcionalidades do Hadoop ainda não estão maduras o suficiente para atender todos as situações em que ele vem sendo empregado. Vale ressaltar que a cada nova versão melhorias significativas são implementados no framework. Para Goldman e parceiros (2012) duas situações são os principais pontos negativos atualmente no Hadoop.

- Arquitetura mestre-escravo com apenas um nó mestre: A centralização de tarefas pode causar empecilhos na escalabilidade, dando possibilidade ao surgimento de um gargalo no desempenho do conjunto. Esse também é um ponto crítico uma vez que uma falha no nó mestre compromete o funcionamento de um *cluster* inteiro.
- Dificuldade no gerenciamento do *cluster*: Esse é um problema enfrentado por qualquer aplicação de computação paralela. Depurar a execução de uma aplicação que executa em diversos nós não é uma tarefa trivial, e mesmo com o esforço na criação de projetos que trabalhem em conjunto com o Hadoop para auxiliar os desenvolvedores nessa questão, esse ainda é um ponto negativo, não só do Hadoop, mas de qualquer aplicação de computação paralela.

2.4.5 Casos em que não é viável o uso do Hadoop

Mesmo o Hadoop sendo atualmente uma das melhores ferramentas para processamento de dados, existem alguns casos que seu uso não é adequado. White (2010) destaca as seguintes questões:

- Problemas com grande dependência entre os dados: Se em uma sub-tarefa executada houver a necessidade de informações contidas nos dados que estão sendo processados por outra tarefa, haverá uma quebra na paralelização do processamento. Essa dependência fará com que o tempo total de execução seja

equivalente, ou maior do que se fosse executada de forma sequencial.

- Processamento de pequenas bases de dados ou de uma grande quantidade de arquivos de tamanho reduzido: Os custos na preparação para divisão das tarefas e comunicação entre os nós envolvidos farão com que o processamento distribuido não seja a melhor opção.
- Regras de negócio complexas: A complexidade do problema, ou o tamanho do fluxo de execução poderá tornar árdua ou inviável a tarefa de abstrair o processo nas duas funções do modelo *MapReduce*.

2.4.6 Componentes do projeto Hadoop

A seguir serão explicados os componentes e subprojetos do Hadoop que foram utilizados na análise dos dados coletados para esse trabalho. O projeto Hadoop é subdividido em 3 projetos principais:

- Hadoop Common: é a estrutura base da aplicação, é ela que da suporte e une os demais subprojetos.
- Hadoop MapReduce: cuida da parte de controle e execução das tarefas de processamento distribuído que seguem o modelo *MapReduce.*
- Hadoop Distributed File System (HDFS): é um sistema de arquivos distribuído projetado e construído para cuidar do armazenamento e transmissão dos grandes conjuntos de dados gerenciados pelo Hadoop. Possui mecanismos que o torna um sistema altamente tolerante a falhas.

2.4.6.1 Outros subprojetos

Além dos componentes principais, outros projetos foram incorporados ao

ecossistema Hadoop, a fim de facilitar tarefas que por ser repetitivas custariam muito tempo ou se tornariam complexas de mais, e também projetos que oferecem novas funcionalidades ao sistema. Dentre os principais podemos destacar:

- Chukwa[7]: sistema especialista em coleta e análise de logs em sistemas de larga escala. Utiliza o HDFS para armazenar os arquivos e o *MapReduce* para geração de relatórios. Possui como vantagem um conjunto auxiliar de ferramentas, que promete melhorar a visualização, monitoramento e análise dos logs.
- Hbase[8]: banco de dados criado pela empresa Power Set em 2007, tendo posteriormente se tornando um projeto da Apache Software Foundation. Considerado uma versão de código aberto do banco de dados BigTable, criada pela Google, é um banco de dados distribuído e escalável que dá suporte ao armazenamento estruturado e otimizado para grandes tabelas.
- Hive[9]: desenvolvido pela equipe de funcionários do Facebook, tendo se tornado um projeto de código aberto em agosto de 2008. Sua principal funcionalidade é em conjunto com o Hbase fornecer uma infraestrutura que permita utilizar o HiveQL, uma linguagem de consulta similar a SQL, bem como demais conceitos de dados relacionais tais como tabelas, colunas e linhas, para facilitar as análises complexas feitas nos dados não relacionais de uma aplicação Hadoop.
- ZooKeeper[10]: criado pelo Yahoo! em 2007 com o objetivo de fornecer um serviço de coordenação para aplicações distribuídas de alto desempenho, que provê meios para facilitar as seguintes tarefas: configuração de nós, sincronização de processos distribuídos e grupos de serviço.

A seção 2.5 apresenta os conceitos relacionados ao Twiter, que foi a fonte de coleta dos dados usados nesse trabalho

7 Chukwa: http://incubator.apache.org/chukwa
8 Hbase: http://hbase.apache.org
9 Hive: http://hive.apache.org
10 ZooKeeper: http://zookeeper.apache.org

2.5 O Twitter

O Twitter é um serviço de *microblogging* que permite que seus usuários publiquem mensagens de texto limitadas a 140 caracteres. Lançado em outubro de 2006, o microblog, se destaca pela velocidade com que as informações a respeito de acontecimentos importantes são divulgados, superando muitas vezes a mídia tradicional (PRIMO, 2008). O Twitter é considerado uma rede social devido a forma com que é feita a interação entre os usuários. Apesar das características similares a dos blogs, o microblog se diferencia pelas mensagens de tamanho limitado, e as atualizações são feitas com maior frequência (TRÄSEL, 2008). As publicações, também conhecidas como tweets, podem ser enviadas por computadores, dispositivos móveis, utilizando tanto serviços oferecidos pelo próprio Twitter, como também aplicativos desenvolvidos por terceiros.

O objetivo inicial do Twitter pode ser resumido pela pergunta: “What’s happening?”, que em português significa: “O que está acontecendo?”. A pergunta aparece na página inicial do serviço e incentiva o usuário a narrar fatos que estão ocorrendo no momento.

Dados oficiais divulgados em fevereiro de 2011 pela consultora britânica Kathryn Corrick, mostraram que o Twitter já possuía 200 milhões de usuários cadastrados (AGUIARI, 2011). O número foi confirmado pela vice-presidente de estratégia internacional do microblog, Katie Stanton, durante uma conferência realizada em 2012 em Nova York. Outras pesquisas estimam que esse número já chegava em 500 milhões de usuários cadastrados em fevereiro de 2012 (TWITTER...2012). Porém, esses dados não foram confirmados pela companhia até o momento da escrita deste trabalho. Dados disponíveis no site oficial[11] do serviço mostram o número de usuários ativos e o número de *tweets* médio por dia. Em novembro de 2012 esse número era de 140 milhões de usuários ativos e 340 milhões de *tweets* por dia.

Além de publicar conteúdo, os usuários do serviço podem usufruir de outras funcionalidades fornecidas. É possível se cadastrar para receber todas as publicações de

11 https://business.twitter.com/en/basics/what-is-twitter/

um determinado usuário, ação chamada de *follow;* responder postagens de outros usuários, o chamado *reply;* repostar um *tweet* de um usuário, o *retweet*, entre outras funcionalidades. Uma prática muito utilizada pelos usuários do Twitter é a *tagging*. A ação consiste em criar etiquetas ou para determinados temas de forma a organizá-los, facilitando a procura por atualizações semelhantes. Para etiquetar uma mensagem, utiliza-se o sinal de sustenido "#", em frente a uma ou mais palavras-chave. Essas palavras são conhecidas entre os usuários do Twitter como *hashtags*.

Os temas mais comentados no Twitter aparecem em uma lista chamada *Trending Topics*. Os *Trending Topics* são atualizados em tempo real, várias vezes durante o dia refletindo os temas mais comentados no momento. O ranking, composto por 10 temas, que podem ser referentes aos assuntos mais comentados em nível mundial, por país, ou por região (nem todos os países, e regiões estão disponíveis), informa ao usuário quais são os assuntos mais relevantes naquele momento, e como está a repercussão deles no microblog.

A escolha do Twitter como fonte de dados para esse trabalho se deve ao fato de ser atualmente uma das ferramentas mais populares da Internet, unindo isso à facilidade da coleta dos dados através de sua API.

2.5.1 Twitter API

Para possibilitar a intereção dos seus serviços com serviços de terceiros, o Twitter mantém uma API[12] (Application Program Interface) que oferece um meio de acesso a todas as funcionalidades do serviço e todas as informações públicas dos usuários. A partir dessa API, o Twitter permite que aplicativos desenvolvidos por terceiros gerenciem as contas dos seus usuários, podendo interagir com outros usuários, publicar, pesquisar, e visualizar dados (ZAGO, 2009).

A API do Twitter funciona através do protocolo HTTP e por meio de requisições GET e POST ela possibilita o recebimento e o envio de dados para o serviço. Para garantir

12 http://dev.twitter.com

a segurança dos dados dos usuários e o sigilo dos mesmos, no uso de funcionalidades que alterem dados das contas, ou que possam ver informações sigilosas, a API exige que o usuário dono da conta dê permissão explícita ao aplicativo antes que ele tenha acesso à mesma (TWITTER, 2012).

3. METODOLOGIA

Nos tópicos a seguir serão abordados os assuntos relacionados às etapas executadas durante a pesquisa. Na seção 3.1 há uma explicação de como é o ambiente de coleta e processamento dos dados, na 3.2 fala-se sobre a ferramenta de coleta de dados e na 3.3 é discutido o uso das ferramentas Hadoop e Hbase na análise dos dados coletados.

3.1 Configuração do ambiente de análise

Uma das maiores vantagens do modelo *MapReduce* é sua escalabilidade, que permite que várias máquinas trabalhem em conjunto em uma mesma tarefa, dividindo essa tarefa em partes menores. Isso permite que seja usado hardware de baixo custo em operações antes possíveis apenas com máquinas robustas e com custo elevado. De acordo com Dean & Ghemawat, em 2008, cada *cluster* dos *data centers* do Google era composto por aproximadamente 1800 máquinas, cada uma com dois processadores Intel Xeon de 2GHz, 4GB de memória RAM, e dois discos IDE de 160GB(Dean & Ghemawat 2008). Esses dados demonstram que não é necessário usar hardware com alto poder de processamento, pois o *MapReduce* se encarrega de dividir a tarefa em partes menores, cada uma processada em uma máquina diferente.

Nesse trabalho, foi usada uma infraestrutura reduzida, com aproximadamente 10 máquinas, onde foram armazenados e processados os dados coletados do *Twitter*. Em cada máquina foi instalada uma instância do Hadoop e do Hbase, possibilitando o processamento distribuído dos dados.

Na próxima seção será explicado como se deu a instalação e configuração do ambiente de análise dos dados.

3.1.1 Configuração e instalação do Hadoop/Hbase

Por ser uma ferramenta livre e de código fonte aberto, existem no mercado várias distribuições diferentes do Hadoop. A maior parte dessas distribuições tem como componente alguma ferramenta que auxilia na instalação e configuração de todos os aplicativos necessários para o funcionamento e gerenciamento de um *cluster* para processamento de dados.

Esse tipo de ferramenta é extremamente útil, pois a tarefa de instalar e configurar separadamente em cada nó todos os softwares necessários, além de ser um trabalho custoso que pode consumir um tempo considerável dependendo das dimensões do *cluster*, ainda pode estar sujeito a erros, o que pode tomar ainda mais tempo. Por esse motivo, foram criadas ferramentas que automatizam a instalação de todos os aplicativos em todos os nó, efetuando também a configuração padronizada, e permitindo o gerenciamento centralizado do *cluster*.

Alguns exemplos de distribuições do Hadoop presentes no mercado incluem: Apache Hadoop[13] (distribuição padrão da Apache Foundation), Amazon Elastic MapReduce[14], HortonWorks[15], GreenPlum HD[16] e MapR[17]. Nesse trabalho foi utilizado a distribuição Cloudera CDH devido a sua ampla documentação, interface traduzida para o português brasileiro e a sua grande comunidade de usuários.

A seguir serão mostrados os passos para instalação e configuração da distribuição Cloudera CDH do Hadoop.

3.1.2 Processo de instalação do Coudera CDH

O Cloudera CDH é composto por dois componentes básicos: *Cloudera Manager* e

13 http://hadoop.apache.org/
14 http://aws.amazon.com/pt/elasticmapreduce/
15 http://hortonworks.com/
16 http://www.greenplum.com/products/greenplum-hd
17 http://www.mapr.com/

o *Cloudera Manager Agent.* O primeiro possui uma interface *Web* por onde é possível gerenciar todo o *cluster*, e o segundo fica instalado em cada um dos nós enviando informações como integridade do nó, recursos disponíveis, serviços em execução entre outros.

Através do *Cloudera Manager* é possível configurar todos os nós do *cluster* de uma só vez, conforme apresentado nas imagens a seguir.

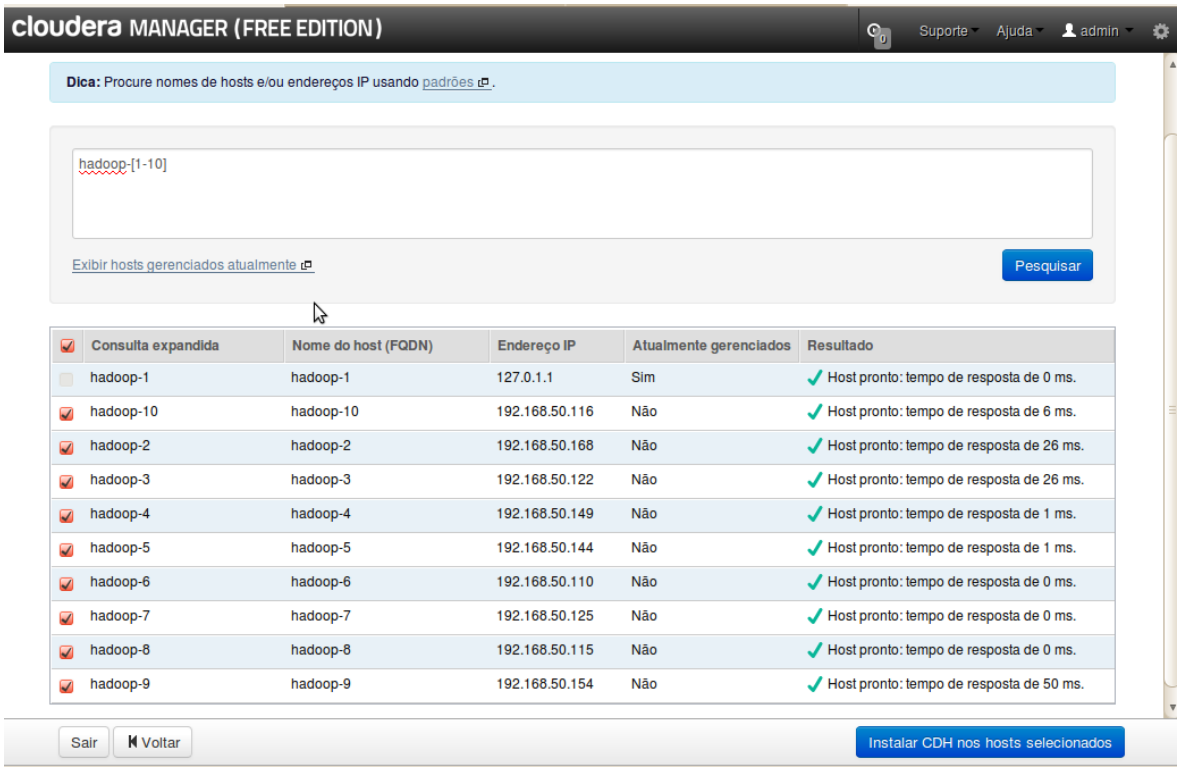

Figura 3.1: Especificando quais hosts serão configurados

A Figura 3.1 mostra a interface onde se especifica quais *hosts* serão configurados pelo *Cloudera Manager*. É possível usar faixas de numeração conforme mostrado no campo superior na imagem (*hadoop-[1-10]*). Ao clicar em 'Pesquisar' a ferramenta lista todos os hosts disponíveis.

Após selecionar os hosts e clicar em 'Instalar CDH nos hosts selecionados' é apresentada a etapa mostrada pela Figura 3.2 onde é selecionada a versão do CDH a ser instalada. Nesse trabalho utilizou-se a versão 4.1.0, que é a mais recente.

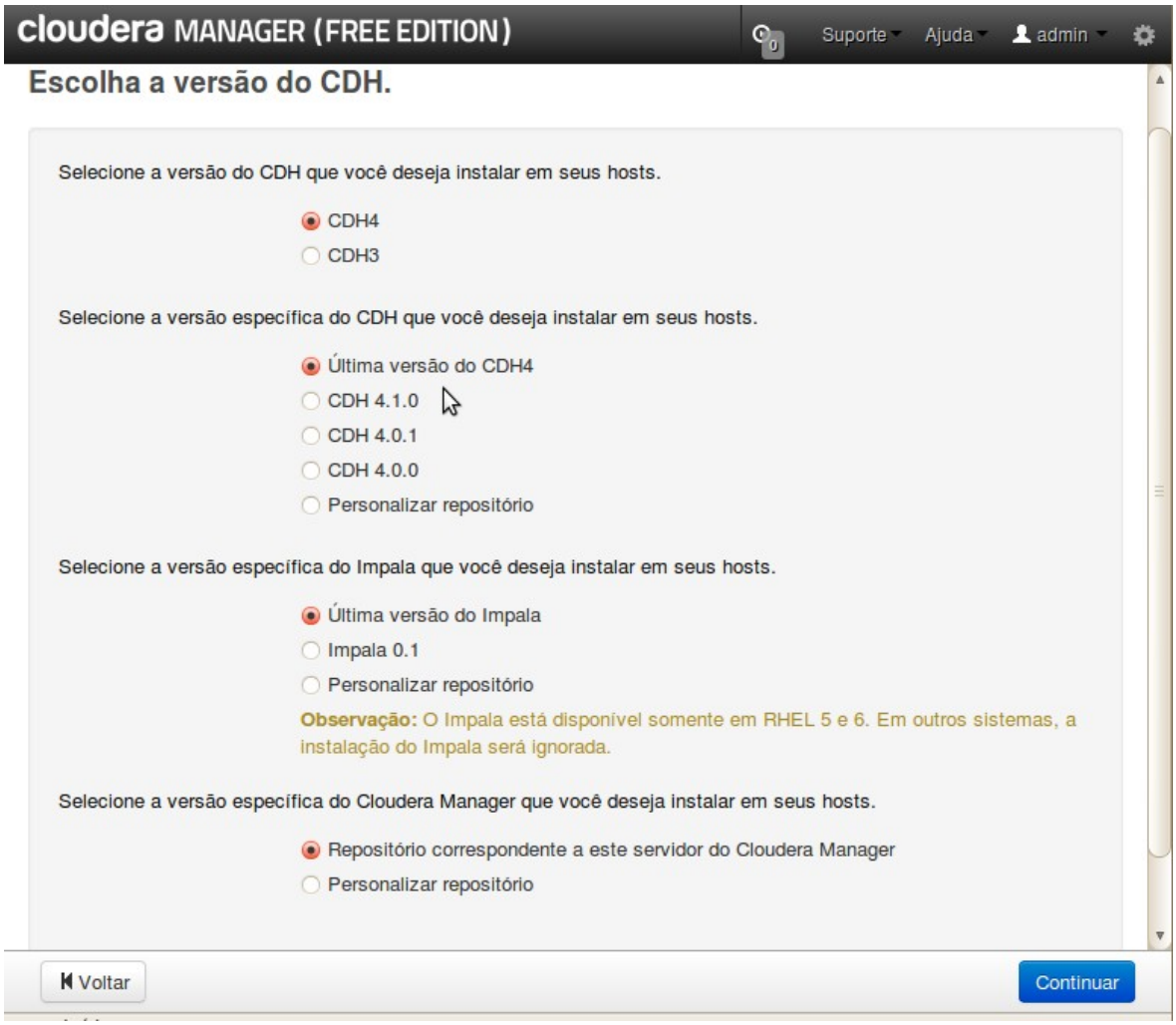

Figura 3.2: Escolhendo a versão do CDH a ser instalada

Clicando em ‘Continuar’ segue-se para a etapa onde são informadas as credenciais para acesso remoto aos nós conforme mostrado na Figura 3.3. Se todos os hosts forem configurados com a mesma senha para o usuario *root*, basta informar essa senha e clicar em ‘Iniciar Instalação’.

A partir desse momento, o *Cloudera Manager* acessa todos os hosts, instala todas as ferramentas necessárias, e faz todas as configurações para que o *cluster* possa funcionar. A Figura 3.4 mostra essa etapa do processo.

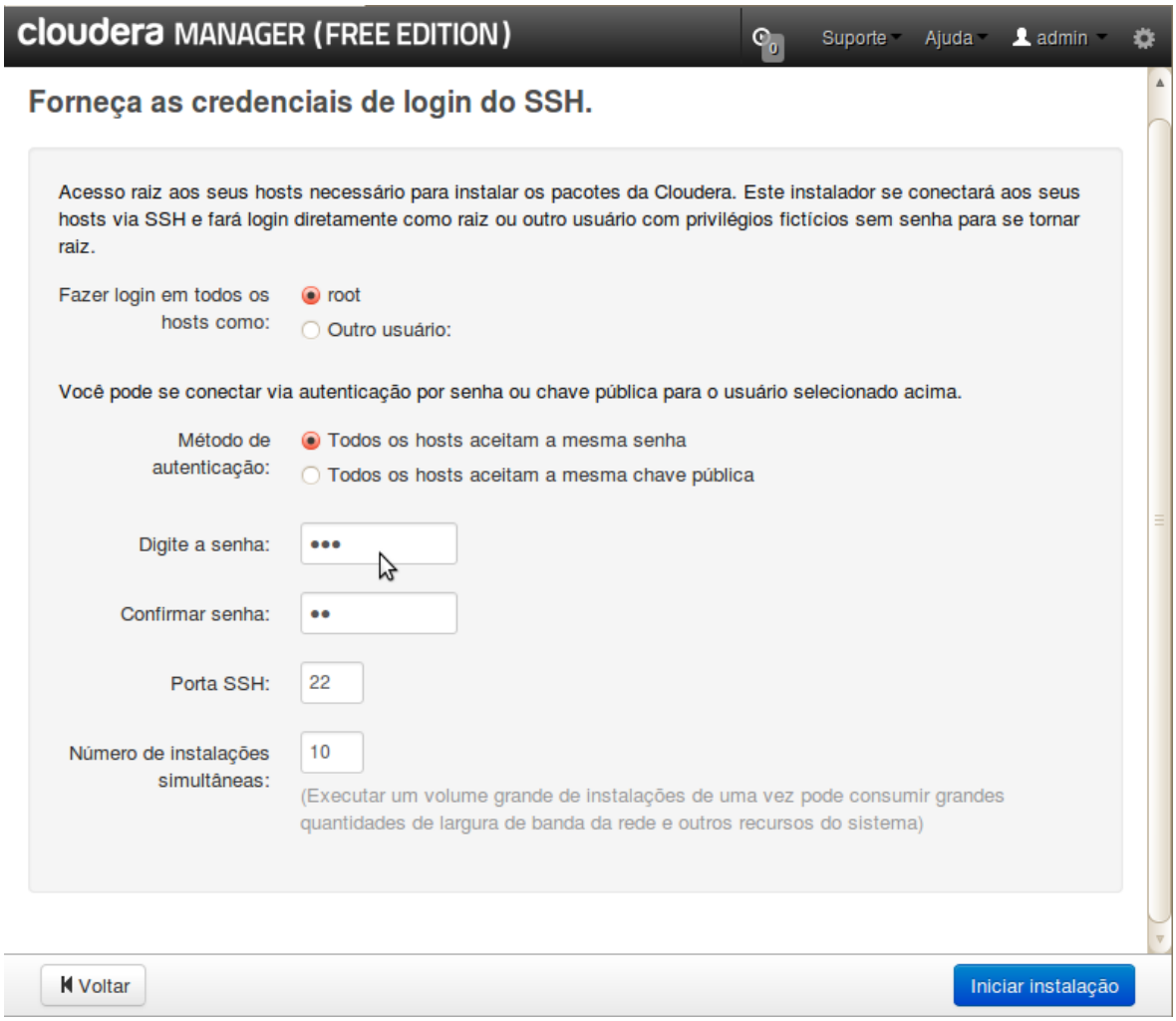

Figura 3.3: Fornecendo as credenciais de acesso

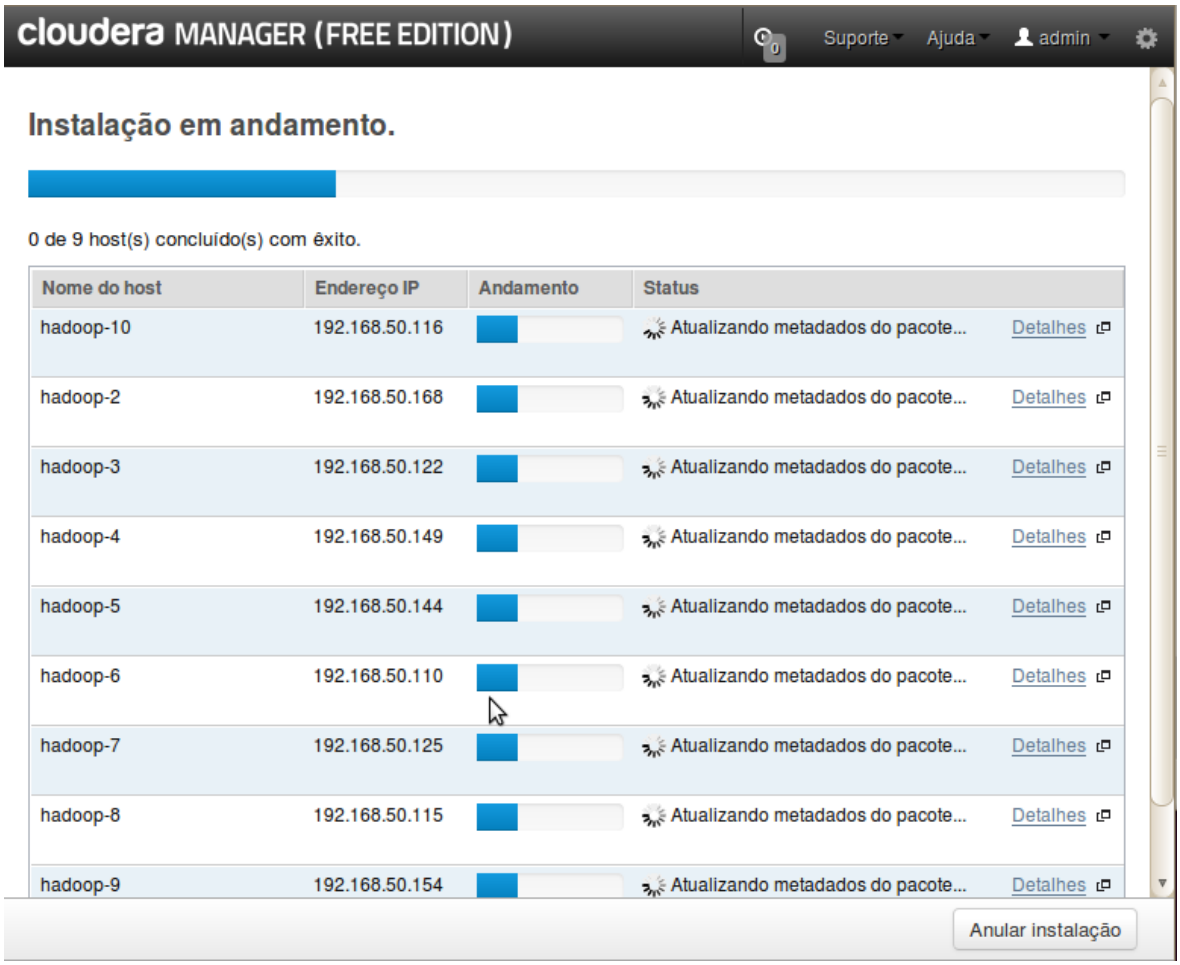

Figura 3.4: Cloudera Manager durante a configuração dos hosts

Após a conclusão da instalação em todos os hosts, o Cloudera Manager realiza uma verificação em cada nó para encontrar possíveis falhas na instalação, ou na comunicação entre as ferramentas de gerenciamento coforme mostrado na Figura 3.5.

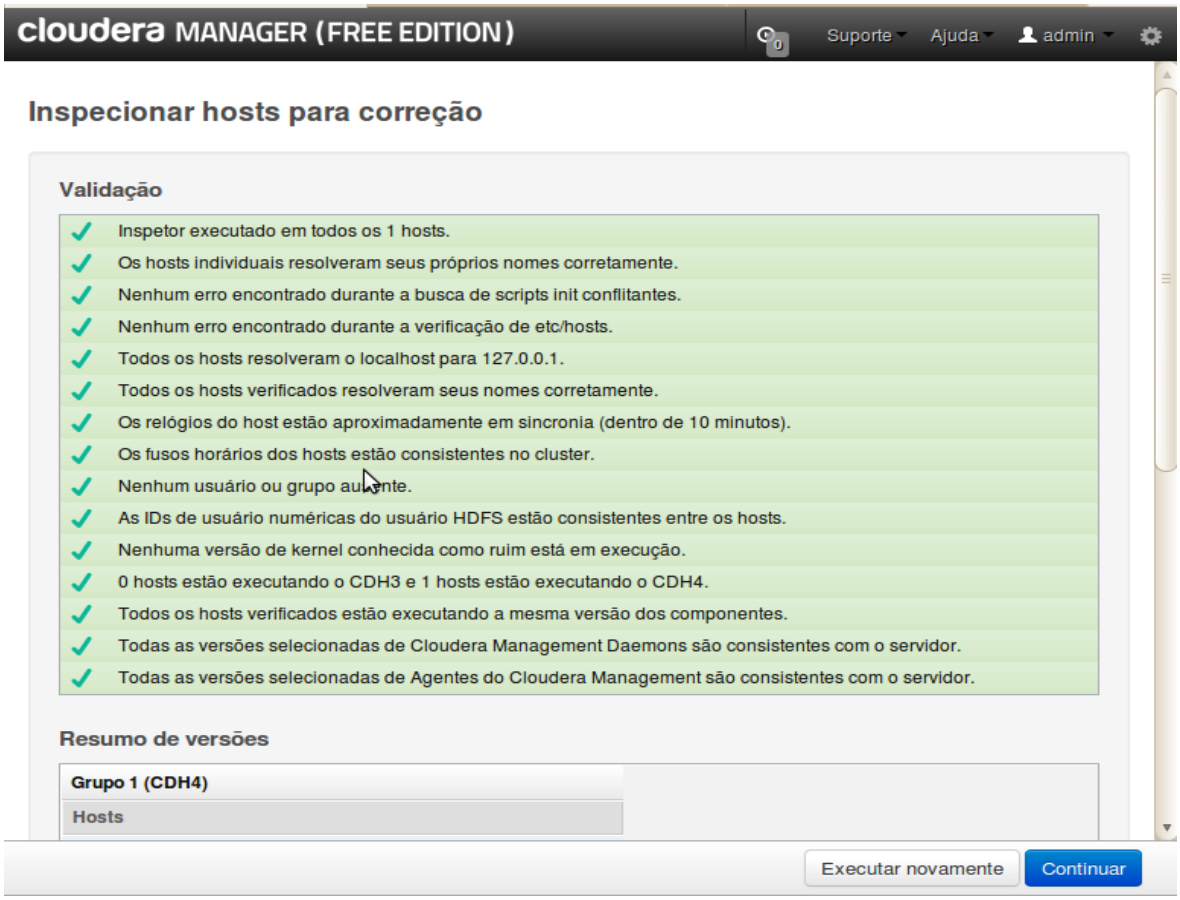

Figura 3.5: Verificação da integridade do cluster

A Figura 3.6 mostra a tela onde pode-se visualizar todos os nós gerenciados, a versão instalada em cada um, e o estado atual de integridade dos mesmos.

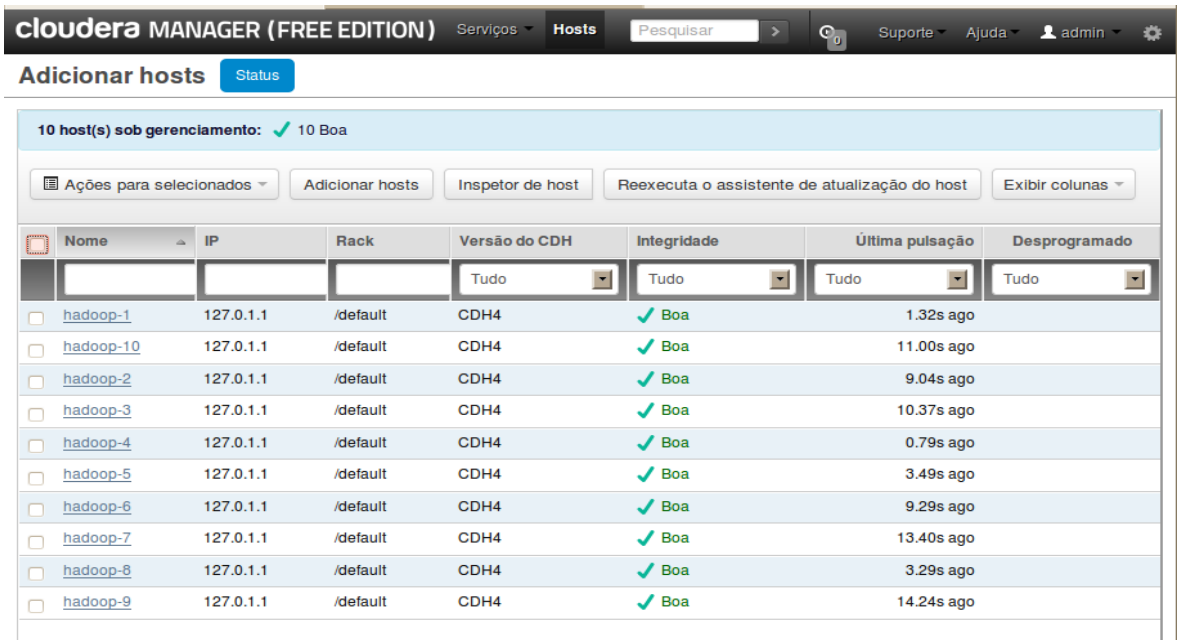

Figura 3.6: Lista de hosts gerenciados

3.2 Ferramenta de coleta de dados

Utilizando a API do Twitter, foi construída uma ferramenta capaz de recuperar uma grande parte dos *tweets* dos usuários de todo o mundo contendo uma ou mais palavras chaves entre um grupo de termos predefinidos. Esses termos foram principalmente derivações da palavra “Olimpíada” e suas traduções para diversos idiomas, e também termos referentes ao nome da cidade sede (Londres). Para a recuperação dos dados foi usada a funcionalidade de pesquisa da API do Twitter, onde se pode procurar em tempo real, *tweets que* contenham os termos indicados em uma busca (TWITTER 2012).

Através desse mecanismo de pesquisa, foram recuperadas as postagens referentes às olimpíadas de Londres 2012, procurando pelos seguintes termos listados na Tabela 4.1.

Idioma	***Termos para busca***					
Português	Olimpíadas	Olímpicas	Olímpica	Olímpico	Londres 2012	Olímpicos
Inglês	Olympiads	Olympics	Olympic		London 2012	
Alemão	Olympiade	Olympischen				
Francês	Jeux Olympiques	Olympiques	Olympique			
Espanhol	Juegos Olimpicos	olimpiadas				
Italiano	Olimpiade	Olimpiadi				

Tabela 3.1: Termos usados nas pesquisas

Fazendo as mesmas pesquisas repetidamente, em intervalos de tempo controlados, é possível recuperar a maioria das postagens relacionadas aos termos pesquisados. Nos sub-tópicos a seguir será explicado mais detalhadamente como funcionam as rotinas de coleta de dados.

3.2.1 O ambiente de armazenamento temporário

Devido a data marcada para o inicio das olimpíadas de Londres 2012 estar muito próxima a do inicio da execução desse trabalho, não foi possível utilizar a ferramenta de busca para armazenar os dados diretamente no *cluster* configurado, pois sua instalação e configuração se deu após a realização dos jogos olímpicos. Por essa razão, os dados foram armazenados na forma de texto simples, em uma base de dados relacional (MySQL), e no mesmo formato disponibilizado pela API do Twitter. Após o ambiente final de armazenamento ser configurado, esses dados foram migrados para ele.

Como esses dados não foram armazenados nesse banco de dados relacional na forma estruturada, aliado ao tamanho dessa base de dados, e também ao formato de armazenamento, não foi viável realizar nenhuma operação de leitura ou busca na mesma,

sendo apenas uma solução temporária encontrada para o armazenamento dos dados que foram sendo coletados.

Mesmo com esse ambiente temporário sendo usado apenas para escrita, foram enfrentadas dificuldades em lidar com bases de dados não estruturados em SGBD tradicionais o que demonstrou a necessidade do uso de outras ferramentas, não baseadas no modelo relacional para trabalhar com dados dessa natureza.

Apenas a título de comparação, durante o processo de coleta, após uma queda de energia elétrica, o MySQL demorou cerca de um dia para analisar e reparar as tabelas do banco de dados. No ambiente configurado com o Hadoop esse problema não afetaria todo o conjunto, pois o sistema de arquivos HDFS tem um mecanismo de recuperação de falhas eficiente mesmo quando o volume de dados é muito grande, característica que MySQL não demonstrou.

3.2.2 O script de coleta de dados

A ferramenta desenvolvida para coleta das informações que os usuários do Twitter postavam no serviço a respeito das olimpíadas de Londres, consiste em um programa escrito na linguagem PHP que utiliza uma base de dados MySQL para configuração da busca, e armazenamento temporário dos dados obtidos.

Na configuração da busca, define-se quais são as palavras chaves que vão ser utilizadas nas buscas feitas através da API do Twitter. Uma vez definidas essas palavras chaves, o programa começa as consultas ao *Twitter*. Essas consultas são realizadas em intervalos de tempo dinâmicos, variando de acordo com a quantidade de resultados que a busca retornar. Dessa forma, se uma palavra-chave X que retornar um número de resultados duas vezes menor que uma palavra-chave Y, a freqüência de consultas da palavra X será também duas vezes menor. Se com o decorrer do tempo esses números se alterarem e as duas passarem a ter o mesmo número de resultados, a freqüência de pesquisa

das suas será a mesma, pois a ferramenta se encarrega de atualizar as configurações que definem a freqüência que cada palavra-chave deve ser pesquisada. Essa atualização é feita a cada busca realizada de acordo com o aumento ou diminuição do número de resultados de cada busca. A ferramenta possui esse comportamento devido algumas restrições existentes no uso da API do Twitter. Essas restrições e a solução utilizada para contorna-las são descritas na seção 3.2.3.

3.2.3 Limitações da API

O *Twitter* impõe limitações ao uso de sua API, com a finalidade de evitar usos abusivos da mesma. A principal limitação é quanto a quantidade de busca permitida por intervalo de tempo. Assim, se o sistema do *twitter* detecta que há um grande número de requisições vindos de um mesmo cliente, ele passa a monitorar a quantidade de acessos desse cliente à API e pára de responder caso ultrapasse os limites pré-determinados.

Por padrão, cada busca em tempo real retorna no máximo os 15 últimos resultados encontrados. Assim, se a busca retornar exatamente 15 resultados, existe a possibilidade de a API não ter enviado todos os resultados mais recentes, pois se existirem 20 resultados recentes, ela mesmo assim retornaria apenas 15.

Por esses dois motivos, o controle de frequência das buscas de cada palavra-chave se torna mais importante, já que se a frequência é alta demais, o número de buscas pode ultrapassar os limites estabelecidos pela API, e assim a ferramenta fica sem resposta por alguns minutos. Por outro lado, se a frequência da busca for muito baixa, o programa não é capaz de obter todos os dados que poderia, diminuindo sua capacidade de coleta.

Frente a esse impasse a solução encontrada após vários testes foi uma fórmula que atualiza as configurações de cada palavra-chave dependendo do número de resultados da última busca, aumentando a frequência caso o numero de resultados seja próximo de 15, diminuindo caso seja próximo de zero e mantendo estável caso fique próximo a um número mediano. Esse número é configurável de acordo com a necessidade.

A fórmula é a seguinte:

$X = (R - A)^2 / 100$

se $R < A$

$F1 = F - (F * X)$

se $R > A$

$F1 = F + (F * X)$

se $R = A$

$F1 = F$

Onde:

R = Número de resultados da última busca

A = Número de resultados médio desejado

F = frequência anterior

F1 = frequência atualizada

X = Coeficiente de atualização

Quadro 3.1: Fórmula para atualização da frequência de busca de um determinado termo

O coeficiente X é encontrado através do quadrado da diferença entre o número de resultados desejados e o número de resultados obtidos, e então o valor da frequência é aumentado ou diminuído de acordo com a necessidade. Dessa forma quanto maior essa diferença maior sera a mudança na frequência de busca da palavra-chave em questão.

3.3 Análise dos dados

A ferramenta de coleta descrita na seção 3.2 foi utilizada no periodo de 19/04/2012 até 30/08/2012 conseguindo coletar aproximadamente 51 milhões de *tweets*, totalizando aproximadamente 56 *gigabytes* de dados.

Para o processo de análise dos dados coletados, foi utilizada a infra-estrutura dos laboratórios de informática das Faculdades Integradas de Caratinga, onde foram instaladas e configuradas todas as ferramentas necessárias conforme descrito no tópico 3.1. O *cluster* configurado foi composto por 10 nós sendo um nó mestre e 9 escravos. Todas as máquinas possuiam a seguinte configuração de hardware:

- Processador: Intel Dual Core 2.0Ghz
- Memória RAM: 2GB
- Armazenamento: HD Sata 160GB
- Rede: Ethernet 100 MB/s

Para armazenar os dados foi criada uma base de dados no Hbase contendo apenas uma tabela que armazena todas as informações. Nessa tabela, cada registro representava um *tweet.* A tabela 3.2 descreve a estrutura do banco de dados

Campo	***Descrição***
Dia	O dia do mês em que a publicação foi feita
Mês	O mês que a publicação foi feita
Hashtags	Array com a(s) *hashtag(s)* presentes na publicação
Idioma	Idioma do usuário que publicou
Fonte	Aplicativo utilizado para realizar a publicação
Para	Usuário para a qual a publicação se destinou (*mentions*)
Text	O texto da publicação

Tabela 3.2: Estrutura do banco de dados

Após a migração dos dados do ambiente temporário de armazenamento para o *cluster* onde foi configurado o ambiente para processamento com o Hbase como base de

dados e Hadoop para realizar o processamento, foi possível obter as seguintes informações coletadas a partir dos dados brutos:

- Aplicativos utilizados: Analisar quais foram os 10 principais aplicativos usados para realizar as postagens sobre as olimpíadas.
- Idioma: Analisar quais foram os 10 idiomas com maior fluxo de informação.
- Fluxo diário: Analisar o fluxo de informação diário relativo às olimpíadas no Twitter durante o período de coleta dos dados.
- Usuários mais populares: Analisar quais usuários foram mais mencionados nas publicações.
- Análise das *hashtags:* Realizando o cruzamento de informações foi possível obter as seguintes análises:
  - *Hashtags* mais utilizadas no geral
  - *Hashtags* mais utilizadas por período de tempo
  - *Hashtags* mais utilizadas por idioma

Para se extrair essas informações da base de dados criada foi utilizada a ferramenta Hive, que permite realizar consultas ao Hbase com sintaxe semelhante a linguagem SQL, transformando essas consultas em rotinas *MapReduce* executadas pelo Hadoop. No capítulo 4serão apresentados e discutidos os resultados dessas análises.

4. ANÁLISE DE RESULTADOS

Nos tópicos a seguir serão apresentados os resultado obitidos com a análise dos dados coletados. Cada tópico seguinte abordará o resultado de uma das análises descritas na seção 3.3. O universo de dados considerado abrange apenas os dados que foram coletados, e não o Twitter como um todo.

4.1 Aplicativos mais utilizados pelos usuários

Para se encontrar os aplicativos mais utilizados foi criada a seguinte consulta:

```
SELECT fonte, count(*)
FROM tweet
GROUP BY fonte
ORDER BY count(*) DESC
```

Gerando os resultados da Figura 4.1

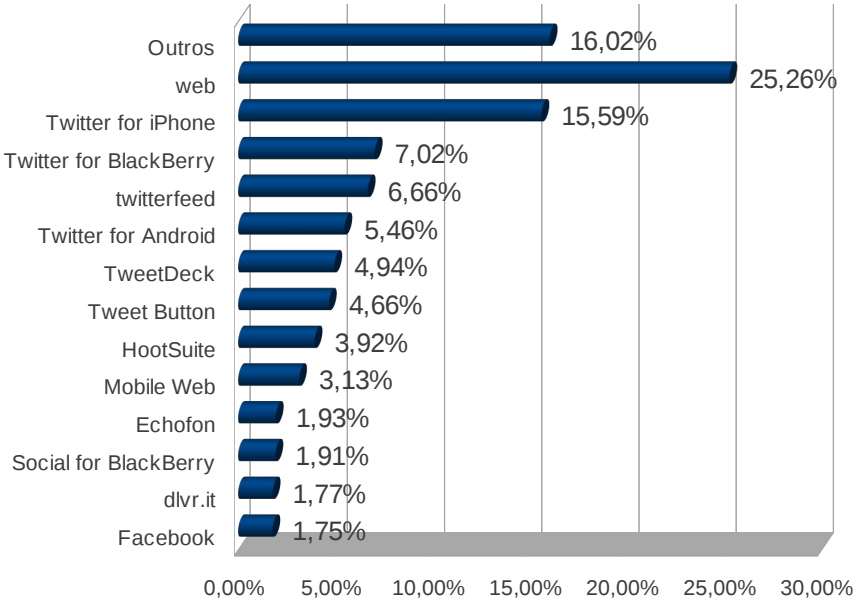

Figura 4.1: Aplicativos mais utilizados

Aproximadamente 17 mil aplicativos diferentes foram utilizados pelo menos uma vez para publicar algum conteúdo, por esse motivo apenas aplicativos com mais de 1% de participação foram apresentados no gráfico. O gráfico 4.1 nos mostra que 25% dos *tweets* foi publicado a partir da interface padrão do serviço (*web*). Outra informação relevante é que pelo menos 30% dos *tweets* foram publicados através de dispositivos móveis – esse número pode ser maior, visto que entre os 16,02% dos *tweets* publicados por aplicativos que não estão relacionados no grafico, há também publicações desse tipo.

Essa consulta demorou 43,7 segundos para ser executada, retornando 17.211 registros.

4.2 Idiomas

Para se encontrar os idomas mais utilizados foi criada a seguinte consulta:

```
SELECT idioma, count(*)
FROM tweet
GROUP BY idioma
ORDER BY count(*) DESC
```

Gerando os resultados da figura 4.2

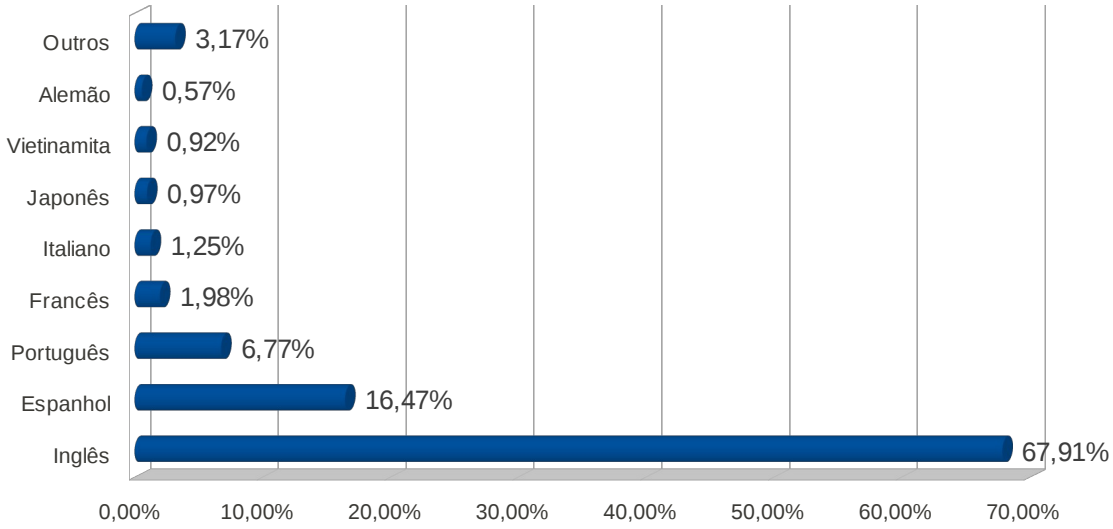

Figura 4.2: Principais idiomas

Foram encontrados *tweets* de usuários de 55 idiomas diferentes sendo que quase 68% eram da língua inglesa com participação um pouco menos significativas de usuários de língua espanhola (16,47%) e potuguesa (6,77%). Aparecem ainda outros idiomas, porém com participação bem menos relevantes.

Essa consulta retornou 55 resultados e demorou 32,5 segundos para ser executada.

4.3 Fluxo diário de *tweets*

Para se encontrar o número de *tweets* publicados por diafoi criada a seguinte consulta:

```
SELECT mês,dia,
count(*)
FROM tweet
GROUP BY mês,dia
ORDER BY mês,dia
```

Gerando os resultados da figura 4.3

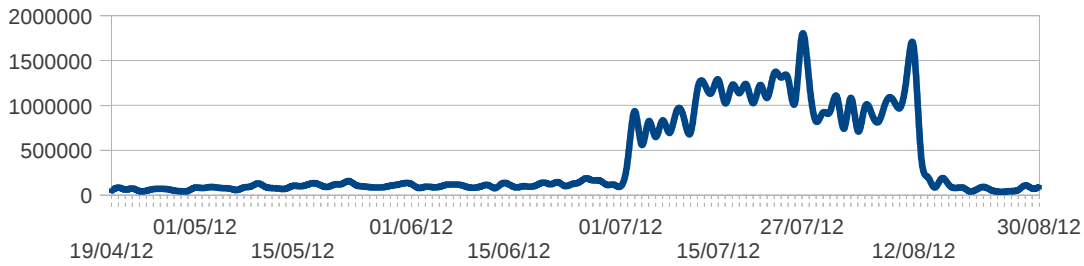

Figura 4.3: Fluxo diário

O gráfico mostra que a maior parte dos *tweets* foram publicados entre os dias 01/07/2012 e 12/08/2012. Esse período compreende os dias que antecederam o início dos jogos até o dia do encerramento. Podemos observar também dois picos, um no dia 27/07 e outro no dia 12/08, nessas datas ocorreram respectivamente as cerimônias de abertura e encerramento, eventos que tiveram grande destaque na mídia mundial.

Essa consulta demorou 44,7 segundos para ser executada, retornando 103 registros.

4.4 Usuários mais populares

Para encontrar os usuários mais populares foi criada a seguinte consulta:

```
SELECT para, count(*)
FROM tweet
GROUP BY para
ORDER BY count(*) DESC
```

Gerando os resultados da figura

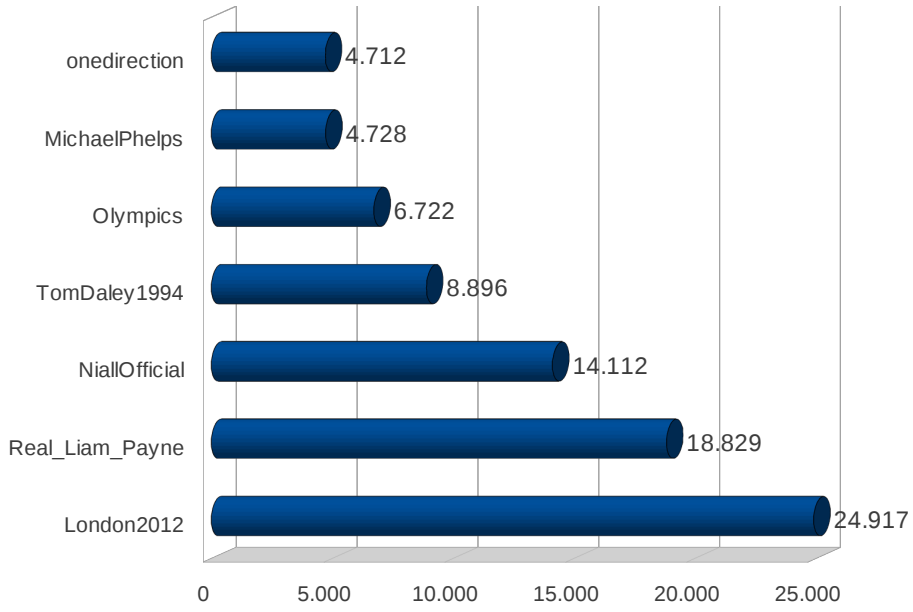

Figura 4.4: Usuários mais populares

O quesito considerado para se eleger os usuários mais populares foi o número de *tweets* que os usuários publicaram diretamente a outros, ou seja, apenas *mentions*. Portanto, os *tweets* de resposta (*reply*) e os *retweets* não entraram na contagem, pois essas informações não são fornecidas pela API.

Os 7 usuários mais populares segundo esse quesito foram:

- **London2012:** Perfil oficial dos jogos olímpicos Londres 2012
- **Real_Liam_Payne:** Músico integrante da banda inglesa One Direction
- **NiallOfficial:** Músico integrante da banda inglesa One Direction
- **TomDaley1994:** Atleta britânico dos saltos ornamentais
- **Olympics:** Perfil oficial do Olympic Movement
- **MichaelPhelps:** Atleta norte-americado da natação
- **onedirection:** perfil oficial da banda *teen* inglesa

Essa consulta demorou 25,7 segundos para ser executada, retornando 7 registros.

4.5 Análise das *hashtags*

Cada *tweet* pode conter mais de uma *hashtag*, e como na estrutura do banco de dados as *hashtags* estão armazenadas em um array, foi criado uma tabela temporaria onde cada registro era uma *hashtag* única. A tabela armazenava além da *hashtag*, o idioma do *tweet* relacionado e o mês da publicação, para que seja possível analisar a as *hashtags* mais utilizadas por idioma e por periodo de tempo (mês).

Em todas as análises, as *hashtags* que mais apareceram foram as de nome relacionado aos jogos olimpicos. “London2012”, “Olympics” e variações desses nomes em alguns idiomas ficaram sempre nas primeiras colocações. Para evitar repetir esses termos em todos os resultados eles não aparecerão nas análises seguintes, onde serão mostrados apenas os demais termos.

4.5.1 *Hashtags* por período

A tabela 4.1 apresenta as cinco *hashtags* mais utilizadas em cada mês. O mês de abril não foi incluido pelo fato da coleta ter abrangido apenas alguns dias do mês.

Mês	***Hashtags***				
Maio	OlympicTorch	FlyMeToLondon	Instants	ThatOnePersonInSchool	News
Junho	London	OlympicTorch	News	RejectedOlympicEvents	Jo2012
Julho	OpeningCeremony	OlympicCeremony	TeamUSA	OlympicTorch	TeamGB
Agosto	TeamGB	TeamUSA	Paralympics	BBC2012	Gold

Tabela 4.1: Hashtags mais utilizadas por mês

4.5.2 *Hashtags* por idioma

A tabela 4.2 mostra as cinco *hashtags* mais usadas nos *tweets* postados por usuários dos cinco idiomas mais utilizados de acordo com o gráfico 4.2

Idioma	***Hashtags***				
Ingles	OpeningCeremony	TeamGB	OlympicTorch	OlympicCeremony	TeamUSA
Espanhol	Jjoo	México	Cuba	Fb	Venezuela
Português	1dNews	Brasil	TerraLondres2012	OlympicCeremony	Uol
Francês	Ftvjo	Basket	Handball	Natation	Judo
Italiano	OlympicCeremony	SkyOlimpiadi	OpeningCeremony	Volley	Italia

Tabela 4.2: Hashtags mais utilizadas por idioma

4.5.3 Análise geral *Hashtags*

A tabela 4.3 mostras as dez *hashtags* mais utilizadas em todo o período analisado.

1º	OpeningCeremony
2º	OlympicCeremony
3º	TeamGB
4º	TeamUSA
5º	OlympicTorch
6º	Jjoo
7º	Usa
8º	News
9º	RejectedOlympicEvents
10º	BBC2012

Tabela 4.3: Hashtags mais utilizadas

4.5.4 Análise geral

Ao se analisar os resultados obtidos com as consultas anteriores, algumas informações relevantes podem ser levantadas:

- Dispositivos móveis: o número de publicações feitas a partir de celulares, smartphones, e tablets superou a interface padrão do Twitter, e comparando as plataformas móveis com as plataformas convencionais, elas ficam quase em igualdade de participação no numero total de publicações.
- Idiomas: Apesar da variedade de idiomas encontrados, o inglês domina com mais de dois terços do total de publicações.
- Fluxo de informação: Além dos destaques nos dois picos de quantidade de informação (abertura e encerramento dos jogos) podemos destacar também um considerável aumento de fluxo a partir do dia 01/07/2012, chegando o volume nos dias que antecederam os jogos ser maior do que durante as competições.
- Usuários mais populares: Analisando os perfis dos usuários mais populares podemos perceber que com exceção dos perfis oficiais dos jogos e do atleta norte-americano Michael Phelps, todos os outros nomes são celebridades da música ou dos esportes que tem grande influência entre os adolecentes (a banda inglesa One Direction e o atleta Tom Daley), mostrando a grande participação dessa fatia da população nos dados coletados.

5. CONCLUSÃO

O crescimento da demanda por ferramentas capazes de gerenciar e analisar bases de dados com tamanhos e níveis de complexidade cada vez maiores fez com que as grandes corporações do setor de serviços para Internet buscassem soluções alternativas que não estivessem ligadas ao uso de hardware de alto desempenho, devido o alto custo para aquisição e manutenção dos mesmos. A solução encontrada por essas companhias foi o uso da computação distribuida, ou computação paralela. Nesse cenário as tecnologias existentes, como os SGBD relacionais, não eram capazes de atender sua necessidade, levando essas empresas a investir em um novo conceito de armazenamento e processamento de dados.

Assim surgiram uma série de soluções capazes de contornar os problemas existentes no armazenamento e processamento de grandes massas de dados, as chamadas soluções *big data*. Entre essas soluções apresentadas, nesse trabalho optou-se por utilizar o Hadoop que vem se destacando no cenário corporativo como ferramenta para extração de informação a partir de grandes massas de dados brutos.

A partir de uma massa de dados coletada do Twitter, o Hadoop foi capaz de dar suporte para realização de diversas análises diferentes, utilizando de forma transparente de recursos computacionais distribuídos nos nós de um *cluster*, cuidando do particionamento dos dados, divisão das tarefas, balanceamento de carga, e recuperação de falhas.

Quando comparado o ambiente onde foi feita a análise dos dados (SGBD não relacional) com o ambiente temporário (SGBD relacional) o modelo não relacional se mostrou muito mais eficiente ao lidar com grandes bases de dados não estruturadas. A forma não estruturada com que os dados foram armazenados no MySQL torna inviável uma comparação direta entre o desempenho das duas abordagens, porém, vale ressaltar que o Hadoop foi projetado para trabalhar justamente nessas condições: grandes volumes de dados em formato não estruturado, ao passo que o MySQL por seguir o modelo relacional se sai melhor com dados estruturados.

Considerando o conjunto de ferramentas do ecossistema Hadoop que auxiliam na configuração e no gerenciamento de um *cluster*, suas funcionalidades que permitem a abstração na distribuição dos dados, escalonamento de tarefas e recuperação de falhas e a possibilidade do uso de hardware de baixo custo para tarefas que exigem grande poder de processamento e capacidade de armazenamento, pode-se concluir que o Hadoop é uma opção viável quando precisamos lidar com grandes bases de dados não estruturadas ou semi-estruturadas, como no caso estudado nesse trabalho. Por outro lado, quando utilizado para lidar com pequenas quantidades de dados, quando existe uma forte interdependência entre os dados e quando é necessário manter a consistêcia em tempo real dos dados, a literatura nos mostra que o ecossistema Hadoop não é a solução mais viável para esses casos.

6. TRABALHOS FUTUROS

Possíveis melhorias na metodologia podem contribuir ainda mais para demonstrar as diversas possibilidades de uso do Hadoop. Trabalhos futuros podem ser realizados utilizando ferramentas que trabalhem com processamento de linguagem natural em conjunto com o hadoop para se extrair mais informações relevantes das publicações no Twitter. Pode-se também utilizar de ferramentas de BI (Bussiness Inteligence) para se obter estatísticas mais elabordas. Já existem ferramentas disponíveis no mercado para atender essas duas situações, no caso do processamento de linguagem natural existe o projeto Apache Lucene[18], para o uso de BI existe o PentahoBigdata[19]. Ambos possuem integração nativa com o Hadoop.

Outra possibilidade de trabalho futuro é a construção de um sistema de monitoramento de redes sociais. Nesse sistema o usuário entraria com os termos de busca desejados e o periodo de tempo da coleta. E então, após o periodo de coleta o sistema forneceria relatórios a respeito da repercussão no Twitter a respeito dos termos pesquisados.

18 http://lucene.apache.org
19 http://www.pentahobigdata.com/

7. REFERÊNCIAS

AGUIARI V., Twitter atinge 200 milhões de usuários. **INFO Online**. 2 maio 2011. Disponível em: http://info.abril.com.br/noticias/Internet/twitter-atinge-200-milhoes-de-usuarios-02052011-23.shl. Acesso em: 17 nov 2012.

ANGLES, R. E GUTIERREZ, C. Survey of Graph Database Models. **ACM Computing Surveys**, 40(1):1–39, 2008.

BRITO, R. W. Bancos de Dados NoSQL x SGBDs Relacionais:Análise Comparativa. **InfoBrasil**, 2010.

CATTELL, R., Scalable SQL and NoSQL data stores, **ACM SIGMOD Record**, v.39 n.4, 2010.

COLLETT, S. Big Data: é um grande negócio? **Computerworld**, 02 set. 2011. Disponível em: <http://computerworld.uol.com.br/tecnologia/2011/09/02/big-data-e-um-grande-negocio/>. Acesso em: 22 out. 2012.

DE DIANA, M., GEROSA, M. A. NoSQL na Web 2.0: Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0. In: **Workshop de Teses e Dissertações de Bancos de Dados do Simpósio Brasileiro de Bancos de Dados WTDBD2010**.; 2010

DEAN, J. & GHEMAWAT, S., 2008. MapReduce. **Communications of the ACM**, 51(1), p.107.

DEAN, J. & GHEMAWAT, S., 2004 MapReduce: Simplified data processing on large clusters. **OSDI '04: 6th Symposium on Operating Systems Design and Implementation.**, p.137-149.

GHEMAWAT, S., GOBIOFF, H., & LEUNG, S.-T. The Google file system. (C. Roisin, E. V. Munson, & C. Vanoirbeek, Eds.) ACM SIGOPS Operating Systems Review, 37(5), 29, 2003.

GOLDMAN, A. et al., **Apache Hadoop**: Conceitos Teóricos e Práticos, Evolução e

Novas Possibilidades. **XXXII Congresso Da Sociedade Brasileira De Computação**. 2012.

IDC. The Diverse and Exploding Digital Universe, 2008. Disponível em: <http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf>. Acesso em: 15 se 2012.

JACOBS, A. The Pathologies of Big Data. **Communications of the ACM - A Blind Person's Interaction with Technology**, Nova Iorque, v. 52, n. 8, p. 36-44, ago. 2009.

LEE, K., LEE, Y., CHOI, H., CHUNG, Y. D., & MOON, B. **Parallel Data Processing with MapReduce**: A Survey. **Architecture**, 40(4), 11–20, 2011.

LÓSCIO, B. F.; DE OLIVEIRA, H. R.; PONTES, J. C. D. S. NoSQL no desenvolvimento de aplicações Web colaborativas. **Simpósio Brasileiro de Sistemas Colaborativos**, Paraty, ago. 2011.

OREND, K. Analysis and Classification of NoSQL Databases and Evaluation of their Abilityto Replace an Object-relational Persistence Layer. **Technical University Munich, Faculty of Informatics**, Munich, 2010.

PADHY, R. P.; PATRA, M. R.; SATAPATHY, S. C. **RDBMS to NoSQL**: Reviewing Some Next-Generation Non-Relational Databases. I**nternational Journal of AdvancedEngineering Sciences and Technologies**, v. 11, n. 1, p. 15-30, set. 2011.

PETTEY, C.; GOASDUFF, L. Gartner. Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data. Gartner, 2011. Disponível em: <http://www.gartner.com/it/page.jsp?id=1731916>. Acesso em: 27 out 2012.

PRIMO, Alex. A cobertura e o debate público sobre os casos Madeleine e Isabella: encadeamento midiático de blogs, Twitter e mídia massiva. **Galáxia**, São Paulo, v. 16, 2008. Disponível em: http://www6.ufrgs.br/limc/PDFs/caso_Isabella_e_Madeleine.pdf. Acesso em: 5 nov 2012.

RAO, B. T., & L.S.S.REDDY, D. Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments, 34(9), 28–32, 2010.

SEVILLA, M. What is Big Data? **Capgemini**, 8 nov. 2011. Disponível em: <http://www.capgemini.com/technology-blog/2011/11/what-is-big-data/>. Acesso em: 20 out. 2012.

STRAUCH, C. NoSQL Databases. **iiWAS '11 Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services**, Nova Iorque, dez. 2011. 278-283.

TAURION, C. Big Data: a nova fronteira da inovação. **iMasters**, 2011. Disponível em: <http://imasters.com.br/artigo/22095/tendencias/big-data-a-nova-fronteira-da-inovacao>. Acesso em: 27 set. 2012.

TAYLOR, R. C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. **BMC Bioinformatics**. 2010;11(Suppl 12):S1.

THE ECONOMIST. All too much. **The Economist**, 25 fev. 2010. Disponível em: <http://www.economist.com/node/15557421>. Acesso em: 15 out. 2012.

THE ECONOMIST. Data, data everywhere. **The Economist**, 25t fev. 2010. Disponível em: <http://www.economist.com/node/15557443>. Acesso em: 15 out. 2012.

TIWARI, S. **Professional NoSQL**. [S.l.]: John Wiley, 2011. Disponível em: <http://books.google.com.br/books?id=tv5iO9MnObUC>.

TRÄSEL, M. O uso do microblog como ferramenta de interação da imprensa televisiva com o público. In: ENCONTRO NACIONAL DE PESQUISADORES EM JORNALISMO, 6., São Bernardo do Campo, SP. **Anais...** São Bernardo do Campo: SBPJOR, 2008.

Twitter chega aos 500 milhões de usuários, diz site. **Folha de São Paulo**. 23 fevereiro 2012. Disponível em: http://www1.folha.uol.com.br/tec/1052381-twitter-chega-aos-500-milhoes-de-usuarios-diz-site.shtml. Acesso em: 17 nov 2012.

TWITTER. Documentação API Twitter. 2012. Available at: dev.twitter.com. Accessed April 18, 2012.

VICKNAIR, C. et al. A Comparison of a Graph Database and a Relational Database. **ACM SE '10 Proceedings of the 48th Annual Southeast Regional Conference**, Oxford, abr. 2010. Disponível em: <http://dl.acm.org/citation.cfm?

id=1900008.1900067>

WHITE, T. **Hadoop**: The definitive guide. 2nd Editio. 2010:625.

XAVIER, W. **O desafio da TI**: como armazenar e processar tantas informações no cenário atual. **Olhar Digital**, 2012. Disponível em: <http://olhardigital.uol.com.br/negocios/digital_news/noticias/o-desafio-da-ti-como-armazenar-e-processar-tantas-informacoes-no-cenario-atual>. Acesso em: 26 set. 2012.

ZAGO, G. O Twitter como suporte para produção e difusão de conteúdos jornalísticos. **Ciberlegenda**. 2009;(6):1-16.