UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DESENVOLVIMENTO DE SISTEMAS PARA INTERNET E DISPOSITIVOS MÓVEIS

ONEIDE LUIZ SCHNEIDER

COMPARATIVO DE PERFORMANCE DO SISTEMA GERENCIADOR DE BANCO DE DADOS MONGODB SOBRE CLUSTERES HETEROGÊNEOS

TRABALHO DE CONCLUSÃO

FCO. BELTÃO

2014

ONEIDE LUIZ SCHNEIDER

COMPARATIVO DE PERFORMANCE DO SISTEMA GERENCIADOR DE BANCO DE DADOS MONGODB SOBRE CLUSTERES HETEROGÊNEOS

Trabalho de conclusão apresentado como requisito parcial à obtenção do título de Especialista, da coordenação de licenciatura em informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Me. Michel Albonico
Co-Orientador: Prof. Me. Welton Costa

FCO. BELTÃO

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Francisco Beltrão

Diretoria de Francisco Beltrão
<Nome da Coordenação>
<Nome do Curso>

TERMO DE APROVAÇÃO

COMPARATIVO DE PERFORMANCE DO SISTEMA GERENCIADOR DE BANCO DE DADOS MONGODB SOBRE CLUSTERES HETEROGÊNEOS

por

ONEIDE LUIZ SCHNEIDER

Esta Monografia foi apresentada em _____ de _____ de _____ como requisito parcial para a obtenção do título de Especialista em < nome do curso >. O(a) candidato(a) foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho _____.

(Aprovado / Aprovado com Restrições / Reprovado)

_____
<Nome do orientador>
Prof.(a) Orientador(a)

_____
<Nome do membro titular>
Membro titular

_____
<Nome do membro titular>
Membro titular

Dedico este trabalho à minha família, namorada e amigos pelos momentos de ausência.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço ao meu orientador Prof. Me. Michel Albonico e Prof. Me. Welton Costa, pela sabedoria com que me guiaram nesta trajetória.

Aos meus colegas de sala.

A Secretaria do Curso, pela cooperação.

Enfim, a todos os que por algum motivo contribuíram para a realização deste trabalho.

“A única maneira de se fazer um excelente trabalho é amar o que você faz”.

Steve Jobs.

RESUMO

Schneider, Oneide Luiz. Comparativo de performance do sistemas gerenciador de banco de dados MongoDB sobre clusteres heterogêneos. 2014. Trabalho de conclusão da Especialização (Especialização para o Desenvolvimento de Sistemas para Internet e Dispositivos Móveis), Universidade Tecnológica Federal do Paraná. Francisco Beltrão 2014.

Os bancos de dados NoSQL (i.e., Not only SQL) tem ganhado vários adeptos devido sua escalabilidade e por seu alto desempenho comparado a bancos de dados relacionais. Visando presente trabalho possui como objetivo um experimento em banco de dados NoSQL, exclusivamente com o MongoDB, onde serão efetuados os experimentos de performance num cluster com vários computadores. Juntamente com algoritmos em Java testar a estrutura.

Palavras-chave: MongoDB 1. NoSQL 2. Java 3. Clusters 4.

ABSTRACT

Schneider, Oneide Luiz. Comparative performance of the database manager MongoDB data on heterogeneous clusters systems. 2014 Work completion of Expertise (Expertise for the Development of Systems for Internet and Mobile Devices), Federal Technological University of Paraná. Francisco Beltran in 2014.

The NoSQL (ie Not only SQL) data has gained many fans due to its scalability and its high performance compared to relational databases. Aiming this work aims a performance test in NoSQL database, MongoDB exclusively, where the performance tests in a cluster with multiple computers will be made. Together with test algorithms Java structure.

Keywords: MongoDB 1. NoSQL 2. Java 3. Clusters 4.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ACID	Atomicidade,Consistência,Isolamento e Durabilidade
BASE	*Basic Availability, Soft-state, Eventual consistency*
DAO	*Data Access Object*
DTO	*Data Transfer Object*
JAR	*Java Archive*
JSON	*Javascript Object Notation*
NOSQL	*Not Only SQL*
XML	*Extensible Markup Language*
API	*Application Programming Interface*
MIT	*Massachusetts Institute of Technology*
IDE	*Integrated Development Environment*
POJO	*Plain Old Java Objects*

SUMÁRIO

1 INTRODUÇÃO

A quantidade de informação nas organizações vem crescendo constantemente e os bancos de dados mais utilizados (e.g., bancos de dados relacionais) começam a não ser mais eficientes. Isso acontece porque eles normalmente baseiam-se em estruturas computacionais centralizadas, que permitem uma expansão limitada dos recursos computacionais (i.e., troca ou adição de processador, adição de memória). Com isso, as organizações precisam de uma estrutura mais escalável e dinâmica para armazenarem seus dados e processá-los. Segundo Sadalage (2013), o paradigma chamado de bancos de dados NoSQL vem ganhando destaque.

Os bancos de dados NoSQL são projetados sobre arquiteturas distribuídas, o que lhes permite processar grandes volumes de dados, com alta disponibilidade e escalabilidade. Atualmente existem vários Sistemas Gerenciadores de Banco de Dados (SGBD) não-relacionais, como os NoSQLs. Cada um deles possui conceitos e particularidades diferentes. Segundo Sadalage (2013), eles podem ser divididos em: chave/valor, documentos, colunas e de grafos. O MongoDB é um SGBD NoSQL que está na família dos orientados a documentos e de acordo com WHO (2014), o MongoDB é utilizado por um grande número de empresas.

1.1 JUSTIFICATIVA

Antes de uma organização optar por um banco de dados NoSQL específico, ela precisa se certificar de que ele tenha uma boa performance. Segundo Rover e Sun (1994), um sistema pode apresentar diferentes medidas de performance se executado sobre arquiteturas diferentes. Sendo assim, é importante que a performance de um banco de dados NoSQL seja verificada sobre mais do que uma arquitetura (e.g., clusteres com diferentes tipos de computadores). Pensando nisso, neste trabalho será verificada a performance do banco de dados NoSQL MongoDB sobre clusteres com diferentes configurações.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo principal é verificar a performance do MongoDB em clusters com configurações distintas.

1.2.2 Objetivo Específico

O presente trabalho tem como objetivos específicos:

- Criar clusteres com diferentes configurações para que o MongoDB seja executado;
- Executar o MongoDB com diferentes quantidades de computadores;
- Coletar as medidas de performance do MongoDB, considerando um algoritmo específico;
- Avaliar os resultados.

1.3 METODOLOGIA

Os experimentos para a verificação da performance do MongoDB segue uma sequência de passos rígida. Primeiramente, foi definido a estrutura do cluster, o formato e quantidade dos dados utilizados nos experimentos. Os dados utilizados foram coletados dos websites do Facebook e da Wikipedia.

Para poder ter um mais significativo foi definido que grande parte dos experimentos realizados em laboratorios com muitos computadores, pois neles temos os mais variados tipos, desde hardware ao sistema operacional. O resto dos experimentos seriam realizados em ambiente doméstico. Foram utilizados 10 computadores para os experimentos na primeira configuracao e 6 notebooks na segunda.

Diante disso, o projeto proposto trata-se de um experimento de performance de um cluster MongoDB. Neste projeto foram utilizadas diversas tecnologias desenvolvidas na linguagem de programação Java (i.e., linguagem de programação orientada a objeto) para efetuar os experimentos. Os dados usados foram extraidos-

coletados da API(e.g., Application Programming Interface) pública do Facebook (i.e., maior rede social do mundo), dados de uma base de XMLs (e.g., eXtensible Markup Language) – documentos fiscais eletrônicas, da empresa Microsys Sistemas Ltda[1] (i.,e., software house) e artigos acadêmicos da Wikipédia[2](e.g., a enciclopédia livre que todos podem editar).

Inicialmente foram definidas algumas diretivas básicas em relação aos dados a serem usados no desenvolvimento dos experimentos, relacionadas ao volume de dados. Esse relacionamento de refere a tipagem dos dados, como por exemplo dados do tipo texto, que o MongoDB trata de formas distintas dependendo o tamanho dos registros. Onde foi definido a forma primitiva, que são textos simples chegando ao maximo 16 megabytes por registro, pois se extrapolar esse tamanho deve-se seguir outro forma de armazenamento. Com base nestas regras foi possível testar o desempenho dos algoritmos criados para os experimentos.

[1] Site da empresa Microsys Sistemas Ltda - www.microsys.inf.br
[2] Site da Wikipedia - http://www.wikipedia.org/

1.4 ESTRUTURA DO TRABALHO

Este trabalho segue a seguinte estrutura: 1) Elaborar introdução e objetivos como a metodologia utiliza, 2) desenvolvimento teórico, 3) resultados e conclusões.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma fundamentação teórica de todos os elementos utilizados na redação do trabalho, focando tanto literatura em livros quanto sites e artigos encontrados na internet.

2.1 SISTEMAS DISTRIBUÍDOS

De acordo com COULOURIS (2013), sistema distribuídos é aquele onde objetos em computadores interligados em uma rede conversem e coordenam suas ações apenas passando informações. E além disso é referência à computação paralela e descentralizada, que é realizada por computadores conectados através de uma rede, cujo objetivo é concluir uma tarefa em comum.

TANENBAUM (2007) enfatiza que com a evolução das tecnologias utilizadas pelos computadores tem aumentado os sistemas distribuídos, por sua vez criados para atender a demanda das diferentes aplicações computacionais. Além disso essas aplicações vem se tornando cada vez mais complexas.

Segundo COULOURIS (2013), um dos principais motivos de utilizar sistemas distribuídos é o desejo se compartilhar recursos. Que por sua caracteriza bem o a divisão de tarefas em uma rede de computadores interligada, podendo ela estar no mesmo prédio ou até em outro continente.

O Google de acordo com COULOURIS (2013), é líder de mercado em pesquisas Web, onde utilizou uma complexa infraestrutura de sistema distribuído para líder com todos seus serviços com a pesquisa em seu site principal. O que se destaca nessa estrutura é uma grande número de computadores interligados em rede isso localizados em todos mundo, além disso um sistema de arquivos distribuídos que foi projeto para suportar grandes arquivos, e ainda suporte a gerenciamento de cálculos paralelos e distribuídos muito grandes.

2.2 NOSQL

De acordo com Alexandre Porcelli (2010), o NoSQL (*not only sql*) teve origem em 2009, onde teve o encontro promovido por Johan Oskarsson e 17 Eric Evans, que teve como objetivo discutir o crescente surgimento de soluções Open Source3 de armazenamento de dados distribuídos não relacionais. E em outubro de 2009, na conferência chamada “no:SQL (east)”, foi redefinido o uso do termo NoSQL para descrever a maneira para armazenamentos não relacionais.

Segundo SADALAGE (2013) o NoSQL não tem como objetivo de invalidar ou substituir totalmente os bandos de dados relacionais, mas sim é servir de alternativa e invalidar que o modelo relacional é como única solução correta ou válida. E ainda ao pensar em sistemas de gerenciamento de banco de dados, logo se pensa em tabelas e registros e a necessidade de ter tudo modelado antes da aplicação ser construída. SADALAGE (2013) enfatiza que no NoSQL, não necessariamente existem tabelas e registros, mas sim o armazenamento é feito de diferentes maneiras, dependendo do banco de dados NoSQL abordado.

Dentro banco de dados NoSQL temos vários tipos, que acordo com SADALAGE (2013) alguns deles são:

- Orientados a Documentos: documentos já é o principal conceito. O banco de dados armazena em formato XML, JSON, BSON, entre outros. Como exemplo desse grupo temos o MongoDB[http://www.mongodb.org/], CouchDB[http://couchdb.apache.org/].
- Chave-valor: um deposito chave-valor é uma tabela hash simples. São os mais simples de usar a partir da perspectiva de API. Os bancos de dados mais populares são Riak[http://basho.com/riak/], Voldemort[http://www.project-voldemort.com/voldemort/].
- Familia de Colunas: armazenam dados em família de colunas como linhas que tenham muitas colunas associadas. Como exemplo nesse grupo temos o famoso Cassandra[http://cassandra.apache.org/], HBase[http://hbase.apache.org/], os dois são projetos apache[http://www.apache.org/].

Com a alta escalabilidade e desempenho, não é possível dizer que NoSQL pode ser usado em todas as situações, pois muitos sistemas, como por exemplo sistemas de automação industrial e comercial, necessitam ter integridade entre os

dados, e como a grande parte dos bancos de dados NoSQL não trabalham com ACID (Atomicidade, Consistência, Isolamento e Durabilidade) o que torna inviável sua implementação em alguns casos. Normalmente são utilizados para salvar enormes quantidades de dados como logs, analises para relatórios gerenciais, processamento maciça de cálculos em grande volumes de dados.

2.3 MONGODB

Wilson (2013) aponta que quando se trata de bancos de dados NoSQL, é difícil ter algo melhor e fácil do que o MongoDB. Ainda Wilson (2013) diz que além da ótima documentação também possui uma ótima e ampla e prestativa comunidade, além de ser muito amigável com os desenvolvedores. Com isso tornando-se um atrativo para os iniciantes no mundo do NoSQL.

Segundo INTRODUCTION (2014), o MongoDB é um banco de dados que fornece um excelente desempenho, alta disponibilidade, escalabilidade. Por ser orientado a documentos é fácil o manuseio dos objetos via linguagem de programação. Além de um excelente serviço de replicação de dados. Quando ser fala em fácil escalabilidade no MongoDB o termo usado por ele é Sharding, que de acordo com Plugge (2010) é a divisão de coleções entre instancias dele.

De acordo com Sadalage (2013) o conceito principal de banco de dados de documentos são Documentos. Os bancos de dados armazenam documentos que podem ser JSON, BSON entre outros tipos. E são essencialmente o próximo nível do chave-valor, permitindo valores aninhados associados a cada chave. Abaixo temos na Figura 1 um representação de um documento.

Figura 1 - Um documento MongoDB. Fonte MONGODB: (2014).

Quando se fala em processamento distribuído Chodorow (2011) destaca que são muito difíceis, tanto para projetar bem e para continuar rodando. A medida que são adicionados servidores, isso gera um tipo dependência entre eles. Isso pode ocasionar problemas, que de acordo com Chodorow (2011) o MongoDB consegue tratar grande parte. Abaixo temos a Figura 2 onde temos a estrutura de um cluster Sharding MongoDB.

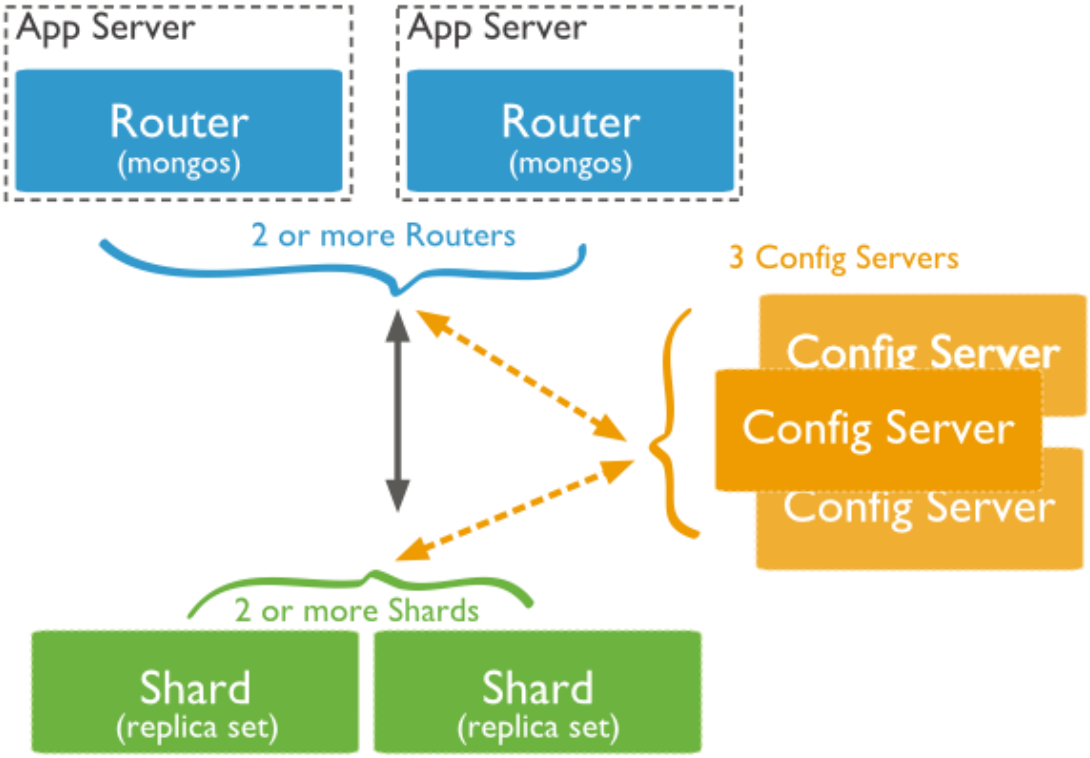


Figura 2- Diagrama de um cluster Sharding.Fonte: SHARDING (2014).

Segundo SHARDING (2014), o MongoDB possui uma ótima destruição de carga. Como sabemos quando um banco de dados começa a ficar muito grande e a quantidade de acessos aumentar, com uma única CPU pode ocasionar na interrupção do serviço. E consequentemente mais caro se torna isso ao adicionar mais processadores e memória. Com o MongoDB esse trabalho se torna mais fácil e menos custoso.

Por conta disso SHARDING (2014) destaca que para tratar desses problemas temos o escalonamento vertical e o Shading. O escalonamento vertical adiciona mais CPUs e quantidade de armazenamento para com isso conseguir mais processamento. Mas isso pode se tornar mais caro. O Sharding por sua vez ao

contrário, divide o conjunto de dados e distribui os dados em múltiplos servidores. Cada fragmento é um banco de dados independente, e juntos, os fragmentos compõem um único banco de dados lógico.

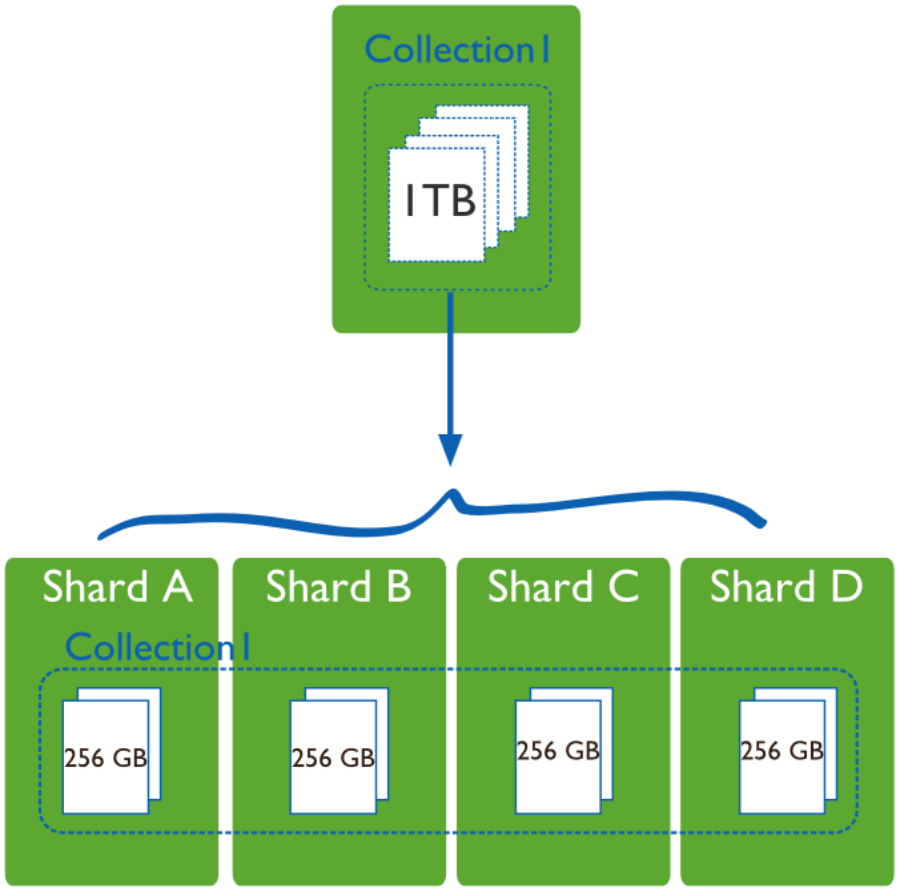


Figura 3 - Diagrama de uma grande coleção de dados distribuídos em quatro fragmentos. Fonte: SHARDING (2014)

Ainda cabe ressaltar que o MongoDB tem um sistema de replicação, para casos de falhas em um dos clusters o outro assume automaticamente. Neste particular MONGODB IN ACTION (2012, p. 33) registrou que :

> Conjuntos de réplicas consistem exatamente um nó primário e um ou mais nós secundários. Como o mestre-escravo replicação que você pode estar familiarizado com o de outros bancos de dados, de um conjunto de réplicas nó primário pode aceitar tanto lê e escreve, mas os nós secundários são somente leitura.

2.4 SPRING FRAMEWORK

De acordo com WEISSMANN (2013), Spring Framework é um framework voltado para o desenvolvimento de sistemas corporativos para a plataforma Java. Ele é baseado nos conceitos de inversão de controle e também injeção de dependência.

Segundo WEISSMANN (2013), o Spring Framework surgiu em 2004, e foi apresentado ao público no livro Expert One-To-One J2mEE Development Without WJB. (JOHNSON, 2004).

WEISSMANN (2013) enfatiza que não é difícil se maravilhar com seu módulos básicos, pois possuem técnicas incríveis. O Spring possui um na sua versão mais básica é formado por seis componentes: o contêiner de inversão de controle, tem suporte a AOP, instrumentação, acesso a dados/integração, suíte de testes e web. Na figura 4 podemos ver seus componentes básicos.

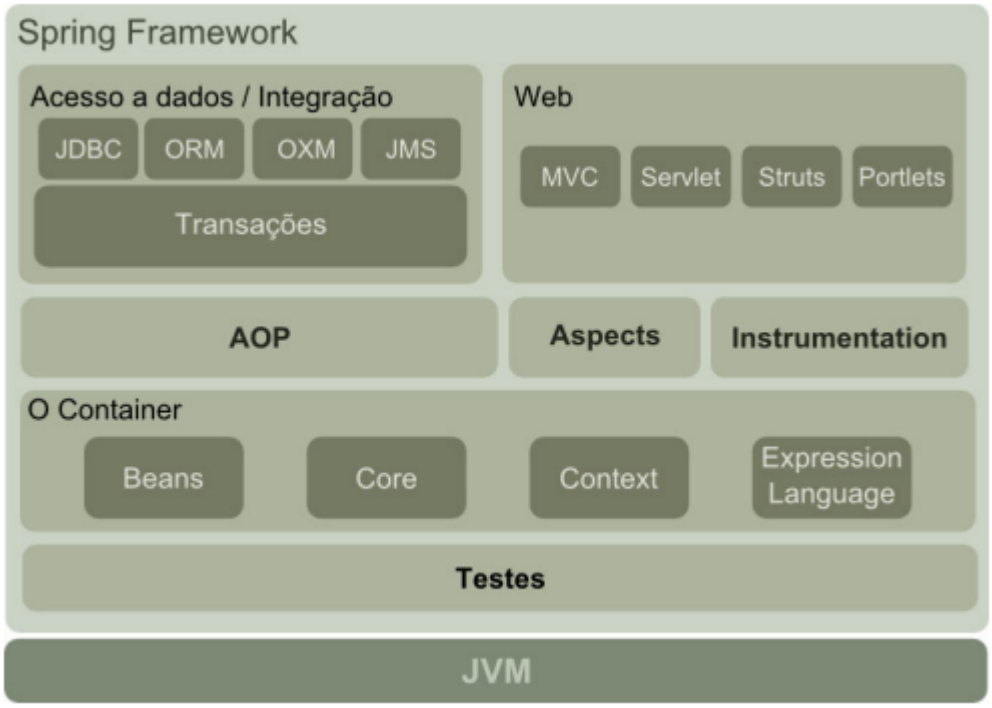


Figura 4 - Os componentes básicos do Spring.

2.5 Java

De acordo com Laura Lemay (1999), “Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems”. Se basou no C++, e foi feita com desígnio de ser mais simples e portável através das plataformas e sistemas operacionais. Ser independente de plataforma é umas das principais vantagens sobre outras linguagens de programação, essencialmente para sistemas que necessitam trabalhar em muitas plataformas diferentes.

2.5.1 Java 8

Silveira e Turini (2014) enfatizam que são praticamente 20 anos desde a primeira versão do Java e que em 2004 com o Java 5, houve mudanças significativas, como por exemplo generics, enums e anotações.

Da mesma form em 2014 temos o Java 8. Que por sua vez nos traz a mais variável de novas funcionalidades, como o Lambda e method references, além disso foram feitas mudanças na linguagem. Ainda cabe resaltar que a API *Collections*, recebeu um significativo *upgrade*, que por sua vez não teve alterações desde 1998. Algumas funcionalidades ainda não foram incluídas infelizmente, como uma melhoria aos *Garbage Collectors*, além da reificação dos generics.

Abaixo na figura 5 temos um exemplo de uso do java 8.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.exemplojava8;

import java.util.Arrays;
import java.util.List;
import java.util.function.Consumer;

/**
 *
 * @author Oneide
 */
public class NovaVersao {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        Usuario user1 = new Usuario("Oneide Luiz", 150);
        Usuario user2 = new Usuario("Oneide 123", 120);
        Usuario user3 = new Usuario("Oneide L S", 190);
        List<Usuario> usuarios = Arrays.asList(user1, user2, user3);

        usuarios.forEach(u -> System.out.println("Nome : "+  u.getNome() + " => Pontos :" + u.getPontos()));

    }
}
```

Figura 5 - Código usando Java 8. Fonte: Autor.

2.5.2 Restfb

Segundo RESTFB (2014), o RestFB é uma simples e flexível API Facebook Graph e Old API REST, cliente escrito em Java. Ele é um software livre disponibilizado sob os termos da licença MIT.

De acordo com RESTFB (2014) tem como principal objetivo: 1) API pública Minimal; 2) Extensibilidade máxima; 3) Robustez em face de freqüentes mudanças na API do Facebook; 3) Configuração orientada por metadados Simples; 4) Zero dependências. Na figura 6 temos um exemplo de uso.

```
public static void main(String[] args) {
    FacebookClient facebookClient = new DefaultFacebookClient("AQUI_VAI_SEU_TOKEN");
    Connection<User> meusAmigos = facebookClient.fetchConnection("me/friends", User.class);
    out.println("User = Oneide Luiz Schneider");
    out.println("->Quantidade de Amigos : " + meusAmigos.getData().size());

    for (List<User> m : meusAmigos) {
        for (User frnd : m) {
            Connection<User> mutualFriends = facebookClient.fetchConnection("me/mutualfriends/" + frnd.id, User.class);
            out.println("=========================================");
            out.println("Meus amigos em Comum com : " + frnd.name + " Total : " + mutualFriends.getData().size());
            out.println("=========================================");
            mutualFriends.getData().forEach(mf -> out.println("   id: " + mf.id + " nome:" + mf.name));
            out.println("=========================================");
        }
    }
}
```

Figura 6 - Exemplo de Uso do RESFB. Fonte: Autor.

2.5.3 Framework Map-Reduce

OVERVIEW (2014) destaca que o framework MapReduce é um paradigma de programação que permite a escalabilidade massiva através de aglomerado de servidores (e.g., milhares de nós). O termo MapReduce realmente se refere a duas tarefas diferentes e separadas. O primeiro é o trabalho map, o que leva um conjunto de dados e converte-lo em outro conjunto de dados, onde os elementos individuais são divididos em tuplas (e.g., pares de chave / valor). E reduce, a saída de um map como entrada e combina essas tuplas de dados em um conjunto menor de tuplas. Na figura 7 temos um exemplo de uso.

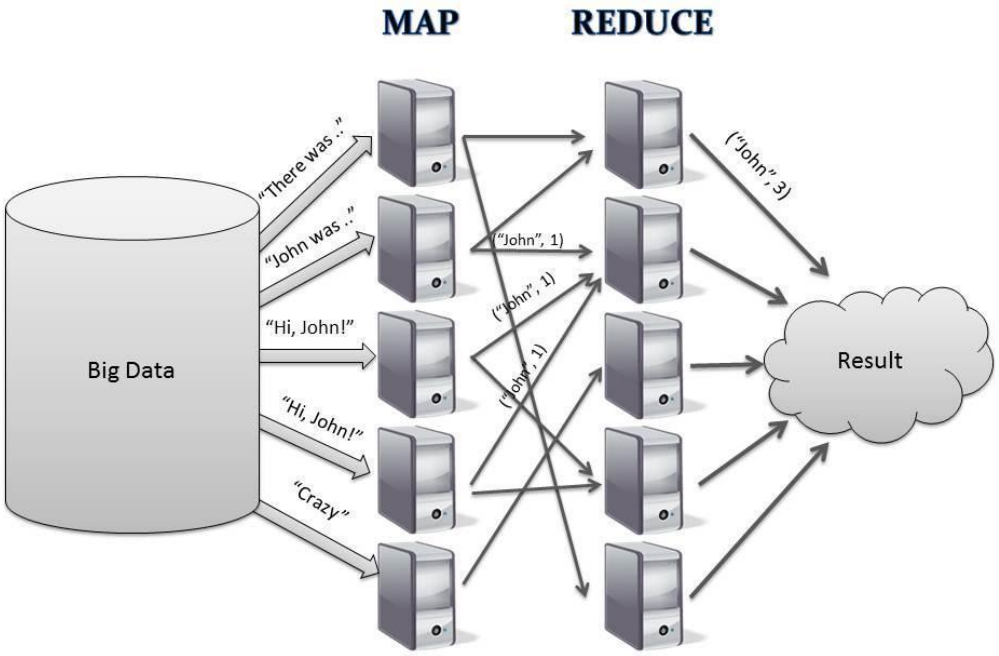


Figura 7 - A computação MapReduce. Fonte: MAPREDUCE 2014.

De acordo com MAPREDUCE (2014), em cima deste novo modelo de programação, as implementações do framework MapReduce mostram uma série de vantagens não-funcionais. Podem se adaptar a vários nós de computação, que são facilmente expandidas por novos nós. E são tolerantes a falhas, o que quer dizer que se um dos nós falhar os outros não são afetados. Por exemplo, as organizações normalmente tem dezenas ou centenas de computadores que são usados apenas em determinados momentos do dia e na maioria das vezes apenas para aplicações de escritório. Estes computadores geralmente mostram muita capacidade não utilizada, em termos de tempo de processamento e espaço em disco, consequentemente podem ser usados para esses fins.

2.5.4 Glassfish

De acordo com GLASSFISH (2014), é um servidor de aplicação *opensource* (e.i., código fonte aberto) cujo qual foi criado para desenvolvimento de aplicações baseadas na tecnologia Java, principalmente em Java EE.

O *GlassFish* foi usado como servidor de aplicação para o desenvolvimento dos algoritmos do MongoTest. Se justifica o uso dele pois já uma produto auxiliar do *Netbeans*(e.i., IDE de desenvolvimento), além disso de acordo com GLASSFISH (2014), é de fácil manuseio e amigável aos desenvolvedores, pois por padrão de instalação não adiciona uma senha, podendo assim configura-lo de forma rápida sem complicação.

2.5.5 Ferramenta Gephi

Segundo GEPHI (2014), a ferramenta *Gephi* é um software de código aberto para a visualização e análise gráficos. Ele usa um mecanismo de renderização 3D para exibir gráficos em tempo real e acelerar a sua exploração. Ele pode ser usado para explorar, analisar, especializar, filtro, manipular e exportar todos os tipos de gráficos.

3 METODOLOGIA

Esse capítulo tem por finalidade descrever o processo de desenvolvimento dos experimentos do cluster e dos algoritmos utilizados, alem da importancia do paradigma MapReduce quando se trata de processamento distribuído.

O desenvolvimento dessa pesquisa surgiu com a premissa do desenvolvimento de clusters em ambientes escalaveis, a dificuldade e difícil manutenção. Com isso o ambiente proposto deveria ser de facil manutenção, além de ter um bom desempenho em ambientes heterogêneos.

Com base nessa premissa obtivemos a ideia do uso do banco de dados NoSQL MongoDB. Que por sua vez é excelente e de fácil manutenção, por ter uma estrutura em formato de JSON[3] internamente, que facilita desenvolvedores criar softwares altamente escaláveis.

Como o MongoDB possui o MapReduce nativamente foi implementado alguns algoritmos para os experimentos. O MapReduce que foi proposto pelo Google em 2004, é um modelo de programação paralela para processamento largamente distribuído de grandes volumes de dados. Ele demonstrou ser adequado para trabalhar com problemas que podem ser particionados ou fragmentados em subproblemas. Isso porque podemos aplicar separadamente as funçõoes Map e Reduce a um conjunto de dados. Se os dados forem suficientemente grandes, podem ainda ser divididos para serem executadas em diversas funções Map ao mesmo tempo, em paralelo.

Ainda cabe resaltar que o MongoDB possui conexão nativa com os componentes do *Framework Hadoop*, que é altamente escalável, e é referência quando se fala em processamento distribuído. Além disso ainda possui driver nativo ao *Apache Mahout*, que possui algoritmos de aprendizagem de máquina altamente escaláveis.

[3] JSON (JavaScript Object Notation) - é uma formatação leve de troca de dados.

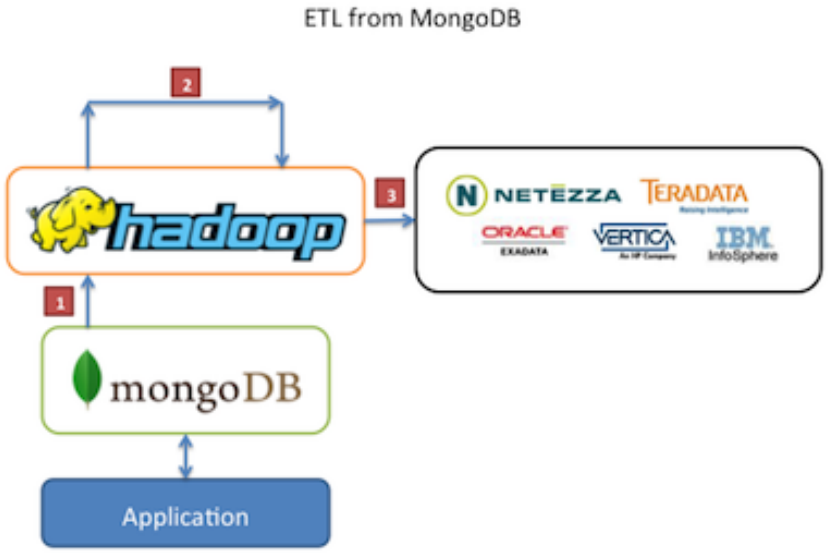


Figura 8 - Integração MongoDB Hadoop. Fonte: SHARDING (2014)

3.1 AMBIENTE DE DESENVOLVIMENTO

Como linguagem de programação foi utilizado JAVA, e o próprio *shell* do MongoDB que é similar ao JavaScript. O Netbeans IDE 8.0 foi utilizado como editor de codigo JAVA e compilador. O Netbeans é uma ferramenta *Opensource* para programadores, que permite escrever, compilar, depurar e instalar programas, alem disso de acordo com NETBEANS (2014), a IDE é completamente escrito em Java, mas suporta qualquer linguagem de programação. O NetBeans IDE é um produto livre, sem restrições à sua forma de utilização. Juntamente com o Netbeans IDE foram utilizadas diversas dependencias para execucao dos algoritmos e programas.

Os algoritmos e programas criados no projeto foram disponibilizados no GitHub[4]. Os mesmos podem ser encontrados, baixados e melhorados pela comunidade.

[4] Repositorio no GitHub nesse link https://github.com/oneidewarrior/utfprtcc .

3.2 AMBIENTE DE TESTES

Grande partes dos experimentos foram realizados em laboratórios com um grande numero de computadores. Ainda cabe resaltar que alguns experimentos cruciais foram realizados de forma mais simples com varios notebooks em ambiente doméstico.

Foram realizados 10 experimentos utilizando computadores com as seguintes configurações: 1) processadores Intel Core 2 duo, Core I5. Core I7, AMD Phenon X4 e X2, HD de 500 e 250 gigabytes de 5400rpm, rede /10 e /100, Sistemas Operacionais Windows XP, Vista, 7, 8, 8.1, Linux Ubuntu, Mint, Debian, juntamente com o Sistema Gerenciador de Banco de Dados MongoDB na sua versão 2.6. Algumas imagens relativas ao processo de implantação podem ser vistas no Apendice B. 2) processador Intel i5, 8 giga de memoria, HDs de 500 gigabytes de 5400rpm com sistemas operacionais Windows 7, Linux Debian, Linux Ubuntu. Algumas imagens relativas ao processo de implantação podem ser vistas no Apendice A.

3.3 PROJETO DE ALGORITMOS E PROGRAMAS

3.3.1 Api Facebook

Com o lançamento de Java 8 em 2014, algumas das novas funcionalidades ja foram usadas, como as expressões lambdas, que são comuns em muitas linguagens, em particular as que seguem o paradigma programação funcional, mas que recentemente vêm sendo introduzidas em linguagens de outros paradigmas. Com essa nova versão foi criado um algoritmo para efetuar download de dados do *facebook*, esse algoritmo foi elaborado juntamente com a biblioteca RestFB que é especifica para trabalhar com os serviços que o *facebook* fornece para os desenvolvedores.

O resultado deste projeto foi um grafo com todos os vínculos dos amigos do autor no *facebook*. O projeto esta disponível para toda comunidade no GitHub[5] , onde qualquer um pode acessar e adapta-lo para sua necessidade.

3.3.2 Xml Database Mongodb

Se tratando de consultas, o MongoDB possui uma facilidade, pois devido a forma como os objetos são salvos, e além disso, o Framework Spring foi utilizado como ferramenta de persistência dos dados, pois ele facilita essa criação, com métodos específicos para criação de *Querys MongoDB*.

```
public List<NfesPorUf> getNfesUf() {
    List<Ufs> ufs = getufs();
    List<NfesPorUf> listRestorno = new ArrayList<>();
    for (Ufs uf : ufs) {
        Query q = new Query();
        q.addCriteria(Criteria.where("emitUf").is(uf.getufNome()));
        q.fields().include("emitUf");
        Long nfes = mongoOperation.count(q, Nfe.class);
        listRestorno.add(new NfesPorUf(uf.getufNome(), nfes));
    }
    return listRestorno;
}
```

Figura 9 - Consultas com o Framework Spring. Fonte: Autor.

Apresentado na Figura 9, o método *getNfesUf()* tem como função retornar todas NFes(e.g., notas fiscais eletrônicas) por estado, a partir de um estado do tipo *string*, após é consultado a quantidade de NFes salvas no banco de dados.

Foi criado na parte servidor da aplicação, vários POJOS, os quais podem ser envolvidos com as anotações do *Spring*, e serão aproveitados exclusivamente nas classes de persistência dos algoritmos.

Como pode ser visto na Figura 10, a classe Nfe apresentada, tem a anotação *@Document* , consequentemente recebendo como valor a String "nfe", isto sugere que todos os objetos Nfe serão salvos em uma coleção de nome "nfe". Logo

[5] Link do repositorio https://github.com/oneidewarrior/FacebookRestFB-Example-FRE

abaixo, compreende-se a anotação *@Id* em um objeto de tipo *ObjectId*, necessário pelo Spring para suscitar automaticamente os ids do MongoDB. Ainda a anotação *@Indexed* , onde o Spring intende que esse campo é indexado. Tambem temos a anotação *@DBRef* que é importante pois ela dita qual é o relacionamento da propriedade com outra classe POJO no sistema.

```
import java.util.Date;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.index.Indexed;
import org.springframework.data.mongodb.core.mapping.DBRef;
import org.springframework.data.mongodb.core.mapping.Document;

@Document
public class Nfe implements Serializable {

    @Id
    private String id;
    private Integer nNf;
    @Indexed
    private String chave;
    private Date ideDEmi;
    private Date ideDSaiEnt;
    private Date ideHSaiEnt;
    private String emitCnpjCpf;
    private Double totalVNf;
    private byte[] xml;
    @DBRef
    private Usuario usuario;
    private Date dataArmazenamento = new Date();

    public String getId() {
        return id;
    }

    public String getChave() {
        return chave;
    }

    public void setChave(String chave) {
        this.chave = chave;
    }

    public Date getIdeDEmi() {
        return ideDEmi;
    }
```

Figura 10 - Nfe POJO. Fonte: Autor.

3.3.2.1 Connection Mongodb

Tendo os POJOs que são objetos básicos Java, devidamente anotados e praticados, o próximo passo para utilização de MongoDB e Spring nos algoritmos é a concepção de uma classe utilitária para ligação com o banco de dados. Esta classe foi criada com nome de “SpringMongoConfig”. Conforme a Figura 11 e 12 abaixo temos a configuração de uma base, além de um *singleton* com o nome “ConexaoMongo” para evitar múltiplas conexões desnecessárias com o banco de dados.

```
import com.mongodb.MongoClient;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.mongodb.MongoDbFactory;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.SimpleMongoDbFactory;

/**
 *
 * @author ONEIDE L S
 */
@Configuration
public class SpringMongoConfig {

    public @Bean
    MongoDbFactory mongoDbFactory() throws Exception {
        return new SimpleMongoDbFactory(
                new MongoClient("10.0.0.102", 27017), "backupnfe"
        );
    }

    public @Bean
    MongoTemplate mongoTemplate() throws Exception {
        MongoTemplate mongoTemplate = new MongoTemplate(mongoDbFactory());
        return mongoTemplate;
    }
}
```

Figura 11 - Configuração da conexão com o MongoDB. Fonte: Autor.

```java
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext
import org.springframework.data.mongodb.core.MongoOperations;

/**
 *
 * @author ONEIDE L S
 */
public class ConexaoMongo {

    private static MongoOperations mongoOperation = null;

    private ConexaoMongo() {
    }

    public synchronized static MongoOperations getInstance() {
        if (mongoOperation == null) {
            ApplicationContext ctx
                    = new AnnotationConfigApplicationContext(
                            SpringMongoConfig.class
                    );
            mongoOperation = (MongoOperations) ctx.getBean("mongoTemplate");
        }
        return mongoOperation;
    }
}
```

Figura 12 – Singeton da conexão com o MongoDB. Fonte: Autor.

3.3.3 Algoritmo Map-Reduce

O paradigma MapReduce foi implementado para os experimentos mais complexos. Pois com esse modelo de programação podemos avaliar o cluster como um todo e analisar possíveis falhas e a distribuição da carga de dados.

O MongoDB possui esse paradigma nativamente, com isso não é preciso de qualquer linguagem de programação auxiliar para executar tal algoritmo. Mas para poder tratar o resultado de forma mais dinâmica foi implementado esse algoritmo das duas formas: 1) via linguagem de programação, Java, como mostra a Figura 13; 2) direto no *Shell* do MongoDB, conforme Figura 14.

```
public class WordCountContadorUserPosts {

    private MongoOperations mongoOperation = ConexaoMongo.getInstance();

    public void contadorPostsUsuario() throws UnknownHostException {
        String map = "function() {\n"
                + "    var dados = this.caption;\n"
                + "    if (dados) {\n"
                + "        // maicusculo rápido para normalizar por suas exigências\n"
                + "        dados = dados.toLowerCase().split(\" \");\n"
                + "        // remove caracteres especiais        \n"
                + "        var r = dados;//.replace(/\\^|~|\\?|,|\\*|\\.|\\-/g, \"\");        \n"
                + "        for (var i = r.length - 1; i >= 0; i--) {\n"
                + "            if (r[i]) {      // certificar-se de que há algo\n"
                + "                emit(r[i], 1); // armazenar um 1 para cada palavra\n"
                + "            }\n"
                + "        }\n"
                + "    }\n"
                + "};";
        String reduce = "function( key, values ) {    \n"
                + "    var count = 0;    \n"
                + "    values.forEach(function(v) {            \n"
                + "        count +=v;    \n"
                + "    });\n"
                + "    return count;\n"
                + "};";
        System.out.println("Iniciou = " + Calendar.getInstance().getTime().toString());
        Mongo mongo = new Mongo("172.30.7.163", 27020);
        DB db = mongo.getDB("dadosFacebook");
        DBCollection colecaoPosts = db.getCollection("postsComplete");
        MapReduceCommand cmd = new MapReduceCommand(colecaoPosts, map,
                reduce, null, MapReduceCommand.OutputType.INLINE, null);
        MapReduceOutput out = colecaoPosts.mapReduce(cmd);
        System.out.println("terminou = " + Calendar.getInstance().getTime().toString());
    }

}
```

Figura 13 – Algoritmo MapReduce Java. Fonte: Autor.

```
MongoDB shell version: 2.6.1
connecting to: test
> var map = function() {
...     var dados = this.texto;
...     if (dados) {
...         // maicusculo rápido para normalizar por suas exigências
...         dados = dados.toLowerCase().split(" ");
...         // remove caracteres especiais
...         var r = dados;//.replace(/\^|~|\?|,|\*|\.|\-/g, "");
...         for (var i = r.length - 1; i >= 0; i--) {
...             if (r[i].length > 5) {      // certificar-se de que há algo
...                 emit(r[i], 1); // armazenar um 1 para cada palavra
...             }
...         }
...     }
... };
>
> var reduce = function(key, values) {
...     var count = 0;
...     values.forEach(function(v) {
...         count += v;
...     });
...     return count;
... };
>
> db.artigos.mapReduce(map, reduce,
...     {out: "word_count"},
...     { query: {$where: "(this.texto.length < 250)"}}
... );
```

Figura 14 – Algoritmo MapReduce Shell MongoDB. Fonte: Autor.

Conforme a Figura 15, ao verificar que as funções Map e Reduce são criadas antes e salvas em variáveis com o mesmo nome, apos isso elas são passadas como parâmetro dentro função nativa mapReduce do MongoDB. O resultado é salvo na coleção “word_count”, que por sua vez também é passada por parâmetro. E ainda é possível passar um filtro, tudo isso de forma simples. Analisando a Figura 15 podemos ver que tudo que passarmos por parâmetro ou qualquer consulta, tudo se refere a um *json.*

Para executar qualquer pesquisa ou qualquer algoritmo mais complexo para tratar os dados dentro do shell do MongoDB é so preciso conectar na instancia dele e executar o que precisar. A linguagem interpretada é similar ao JavaScript, que de acordo com GETTING (2014) é igual. Na Figura 15 temos um exemplo onde é comparada uma consulta a um banco de dados relacional ao NoSQL MongoDB.

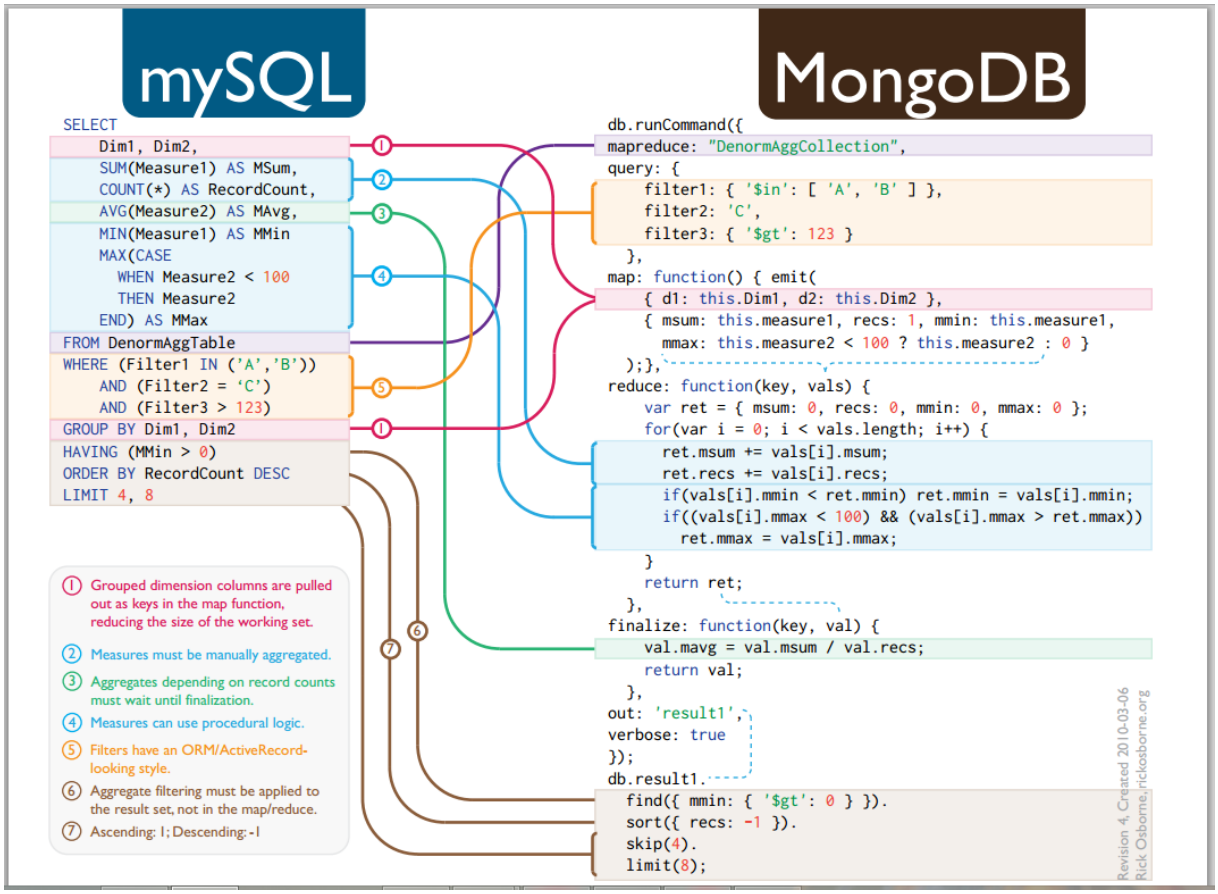


Figura 15 - Comparação SQL Relacional ao NoSQL. Fonte: TRANSLATE 2014.

3.3.4 Dados Wikipedia

Para um experimento com uma maior quantidade de dados, foi efetuado um download de artigos acadêmicos da Wikipedia. No qual o formato é em XML, onde foi preciso criar um novo algoritmo para tratar esses dados e salvá-los no MongoDB. Pois esses XML tem em torno de 45 gigabytes de tamanho, gerando uma base de 120 gigabytes no MongoDB.

Como esse arquivo é muito grande para abrir e consequentemente carregar ele, foi precisar fazer a leitura sob demanda. Onde o algoritmo criado carregava por partes e já retornava o texto. Exemplo do início do algoritmo na Figura 16. Ainda cabe ressaltar que esse algoritmo pode ser adaptado para qualquer situação que precise tratar arquivos grandes.

```
public class CarregarXML {

    public static void main(String[] args) throws Exception {

        XMLReader r = new XMLReader();
        r.addHandler("page", new NodeHandler() {
            Conexao con = new Conexao();
            @Override
            public void process(StructuredNode node) {
                try {
                    if (!node.isEmpty("revision/text")) {
                        String text = node.queryString("revision/text");
                        text = text.replaceAll("[^a-zZ-Z1-9 ]", "");
                        String resultado = text.substring(0,400);
                        con.insert(text);
                    }
                } catch (XPathExpressionException e) {
                    e.printStackTrace();
                } catch (UnknownHostException ex) {
                    Logger.getLogger(ExampleXML.class.getName()).log(Level.SEVE
                } catch (Exception ex) {
                    Logger.getLogger(ExampleXML.class.getName()).log(Level.SEVE
                }
            }
        });
        r.parse(new FileInputStream("D:/enwiki-20140203-pages-articles.xml"));
    }
}
```

Figura 16 – Algoritmo para carregar arquivos grandes. Fonte: Autor 2014.

4 APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Os resultados alcançados foram satisfatórios, a estrutura criada mostrou ser de fácil implantação, mas também mostrou que alguns cuidados devem ser tomados. Como a cada registro do tipo *String* dentro da coleção não poderá passar de 16 *megabytes,* para registros maiores deve-se usar o GridFS. Ainda cabe ressaltar que a cada nó adicionada o desempenho aumentou. Estrutura do trabalho pode ser vista na Figura 17.

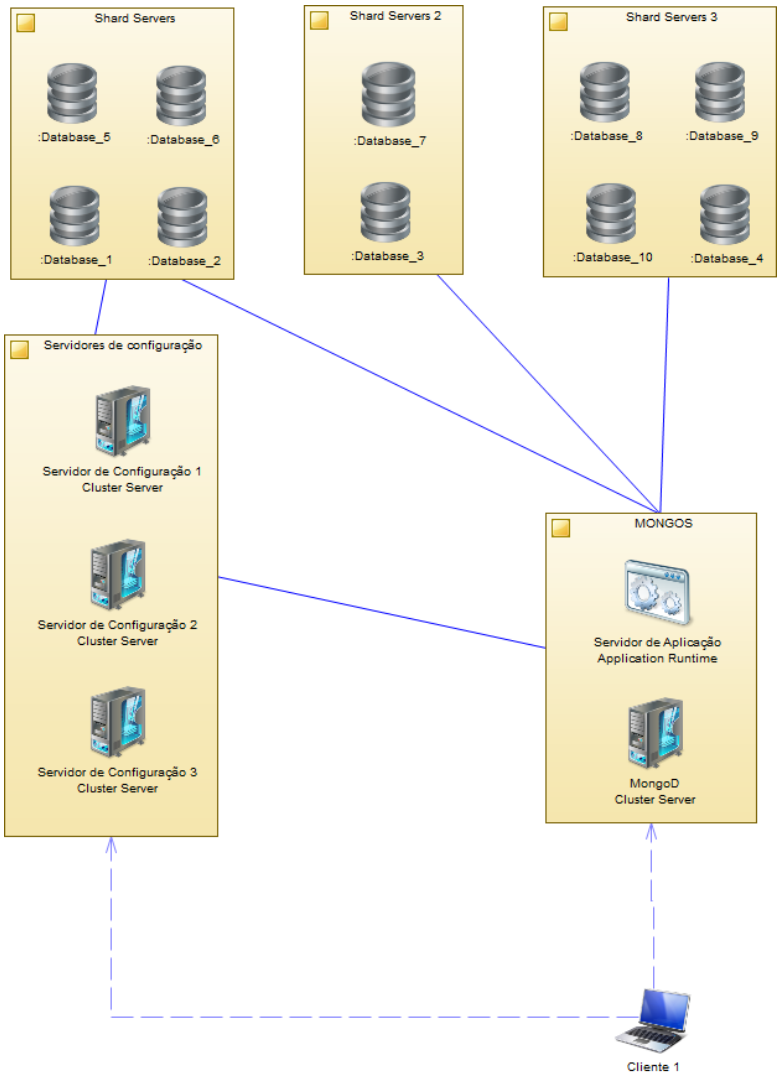


Figura 17 - Estrutura criada. Fonte: Autor.

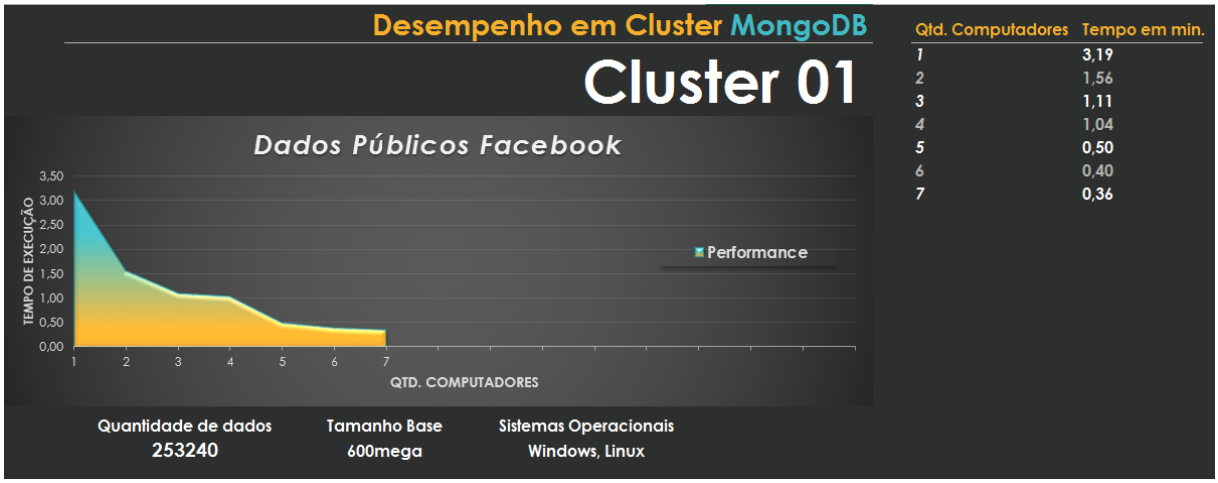


Figura 18 – Resultado com base dados do facebook. Fonte Autor.

Como pode ser visto na figura 18, os resultados foram satisfatórios diminuindo a cada instancia adicionada ao cluster. Esses resultados foram obtidos com a execução do algoritmo MapReduce criado nesse projeto, que pode ser visualizado na figura 14.

Ainda cabe ressaltar que consultas excessivas também pode deixar qualquer banco de dados lento. Com isso foi realizado um experimento com várias consultas simultâneas para averiguar possíveis erros ou falhas. Com esse experimento foi possível derrubar um cluster. Esse Cluster contava com dois computadores com sistema operacional Windows XP de 32 bits. Ao deixar por alguns minutos rodando a instancia MongoDB parrou e não iniciou mais, consequentemente foi preciso refazer todo cluster. Com isso foi contado que a versão 32 bits não aconselhada quando se trata em muitas consultas simultâneas.

Um grande problema do MongoDB é essa tolerância a falhas a nível de hardware. Quando uma instancia do MongoDB para todo cluster é afetado, pois o sistema de *sharding* que ele usa para particionar os dados funciona assim, cada nó recebe partes dos dados gerando assim uma dependência entre eles, consequentemente quando um parar todo cluster para de funcionar. Como podemos ver na figura 3. Esse problema pode ser resolvido com o sistema de replicação, onde automaticamente um novo cluster assume quando um se encontra com problemas.

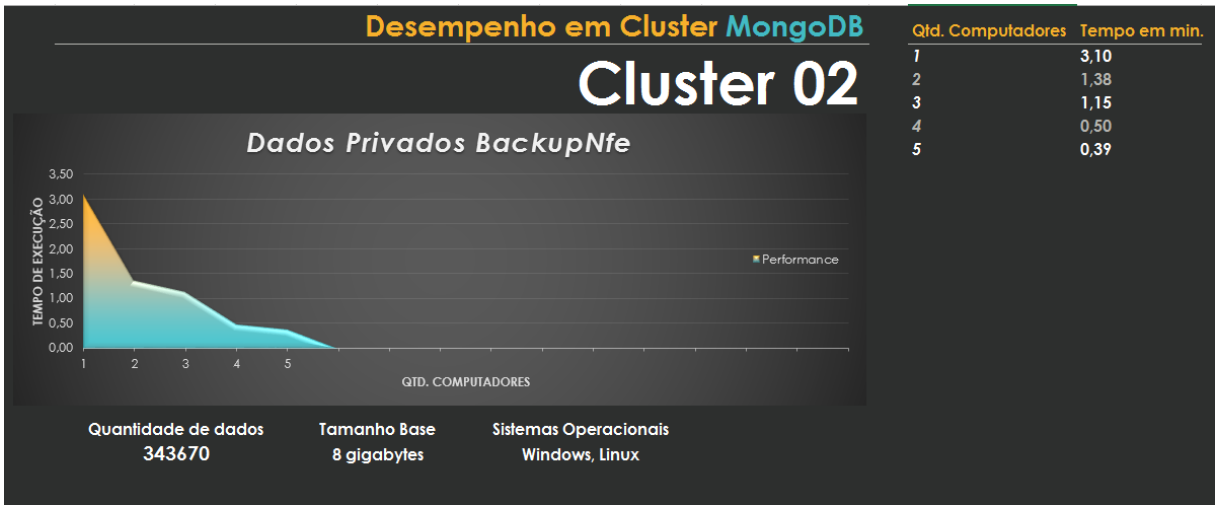


Figura 19 - Resultados com dados do bkpnfe. Fonte: Autor.

Conforme a figura 19 podemos ver também um excelente resultado mesmo após os problemas com a versão 32 bits do MongoDB. Na figura 10 pode ser visto uma das consultas realizadas nesse experimento.

Figura 20 - Resultados com dados da Wikipedia. Fonte: Autor.

Na figura 20 temos o experimento com os dados da wikipedia que foi o mais demorada e complexo criado. Nesse experimento foi obtido mais de 14 milhões de artigos que por sua vez gerou 124584900 milhões de tuplas chave/valor. Como podemos ver a figura 23 no final do procedimento MapReduce.

Lembrando que todos os computadores usados para os experimentos eram de porte normal, nenhuma delas era de porte de servidor. Com isso podemos registrar que por serem computadores simples os resultados foram excelentes.

5 CONSIDERAÇÕES FINAIS

Este capítulo apresenta a conclusão de todo o trabalho realizado, citando os resultados que eram esperados e os resultados que foram obtidos, sobre cada experimento realizado. Também cita quais são as futuras implementações do projeto desenvolvido neste trabalho.

5.1 CONCLUSÃO

A prática de um banco de dados NoSQL, em ambientes heterógenos, tem grandes vantagens devido à velocidade e fácil aplicação e interligação com variados frameworks e até linguagens de programação existentes. Devido a maior parte dos banco de dados NoSQL não serem ACID deixam a desejar sobre sua verdadeira capacidade de segurança dos dados, mas se tratando de escalabilidade, desempenho, tolerância a falhas isso cair fora de ponderação.

O MongoDB não se difere neste ponto, pois também tem mais propriedades de BASE do que propriedades ACID. Mas em compensação, a escalabilidade vertical e horizontal proporcionada pelo MongoDB, consegue totalmente se sobrepor à dificuldade quanto à segurança. Vale ressaltar que os sistemas NoSQL não são inseguros o tempo inteiro, e sim, tem picos de momentos “desprotegidos”.

Em ambientes heterogêneos com diversas configurações de computadores e sistemas operacionais o MongoDB se mostrou muito diligente na sua versão 64 bits. Onde foram obtido ótimos resultados. Nesse contexto os frameworks de acesso a dados como Spring Framework mostrou-se muito benéfico, pois além de facilitar o desenvolvimento para o MongoDB, ele dispõe de engenhos que minimizam a falta das características ACID, como controle de versionamento.

Ainda cabe ressaltar que o MongoDB possui um shell poderoso, onde podem ser realizadas a mais complexas consultas e exportações de dados. E ele pode ser totalmente gerenciável pelo seu shell.

5.2 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

O projeto Comparativo de performance do sistema gerenciador de banco de dados MongoDB sobre clusteres heterogêneos, como foi nomeado esse experimento neste trabalho, teve como principal objetivo verificar a o desempenho do MongoDB em diversos ambientes, e também mostrando que não é tão complexo o uso do mesmo quando se fala em computação distribuída. Vários aspectos importantes serão futuramente desenvolvidos, como a comparação com outro banco de dados NoSQL, e uma comparação da implementação MapReduce do Apache Hadoop com a nativa do MongoDB. Mas como principal continuação seria criação de uma interface para que o trabalho possa ser feito visualmente não por comando no *shell*.

REFERÊNCIAS

ABOUNT MapReduce. Disponível em <http ://bigdataprojects.org/?page_id=76/>. Acesso em 21 de agosto de 2014

BANKER, Kyle. MongoDB in Action. United States Of America: Manning, 2012. 311 p.

COULOURIS, George et al. Sistemas Distribuídos: Conceitos e Projeto. 5. ed. São Paulo: Bookman Editora, 2013. 1055 p.

CHODOROW, Kristina. Scaling MongoDB**.** Eua: O'reilly, 2011. 59 p.

C. Monash, "NoSQL Basics, Benefits and Best-Fit Scenarios", 2010.

Disponivel em:

<http://www.informationweek.com/news/software/info_management/showArticle.jhtml?articleID=227701021>. Acesso em 10 de julho de 2014

Dean, J. and Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. Commun. ACM, 51(1):107–113.

DEAN, Jeffrey; S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In Sixth Symposium on Operating System Design and Implementation (OSDI), 2004.

GEPHI.
<https://gephi.github.io/> . Acesso em 07 de Setembro de 2014.

GETTING Started is Simple, Disponível em <https://mms.mongodb.com/learn-more/>. Acesso em 14 de julho 2014

GLOSSARY. Disponível em <http://docs.mongodb.org/manual/reference/glossary/#term-collection/>. Acesso em 21 de Agosto 2014

GLASSFISH.
< https://glassfish.java.net/getstarted.html> . Acesso em 31 de agosto de 2014.

HADOOP AND MONGODB Use Cases. <http://docs.mongodb.org/ecosystem/use-cases/hadoop/> . Acesso em 31 de Agosto de 2014.

INTRODUCTION to MongoDB. Disponível em <http://www.mongodb.org/about/introduction/>. Acesso em 22 de maio 2014

JOHNSON, Rod. Expert One-To-One J2EEDevelopmentWithoutEJB. W r o xPress, 2004

Lemay, Laura. Perkins, L. Charles. Teach Yourself JAVA in 21 Days. Editora Sams.net. 1999.

MAPREDUCE. <http://dme.rwth-aachen.de/de/research/projects/mapreduce> . Acesso em 26 de agosto de 2014.

MAP-REDUCE. Disponível em <http://docs.mongodb.org/manual/core/map-reduce/>. Acesso em 21 de Agosto 2014

MAYER-CHONBERGER, Viktor; CUKIER, Kenneth. Big Data: como extrair volume, variedade, velocidade e valor da avalanche de informação cotidiana / Viktor Mayer-Schonberger, Kenneth Cukier ; tradução Paulo Polzonoff Junior. - 1. Ed. – Rio de Janeiro : Elsevier, 2013.

MONGODB CRUD Introduction. <http://docs.mongodb.org/manual/core/crud-introduction />. Acesso em 23 de agosto de 2014.

O que é o NETBEANS ? <https://netbeans.org/index_pt_PT.html>. Acesso em 02 de Setembro de 204.

OVERVIEW. <http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html > . Acesso em 26 de agosto de 2014.

PLUGGE, Eelco; MEMBREY, Peter; HAWKINS, Tim. The Definitive Guide to MongoDB**:** The NoSQL Database for Cloud and Desktop Computing. New York, Eua: Apress, 2010. 329 p.

Porcelli, Alexandre, Revista Java Magazine. Edição 86. Editora DevMedia; 2010.

RESTFB. <http://www.restfb.com/>.Acesso em 01 de março de 2014.

ROVER, D. T.; SUN X.; Scalability of Parallel Algorithm-Machine Combinations. IEEE Transactions on Parallel and Distributed Systems, vol 5. Junho, 1994.

SADALAGE, Pramod J.; FOWLER, Martin. NoSQL Essencial: um Guia conciso para o Munda Emergente da Persistência Poliglota / Pramod J. Sadalage, Martin Fowler ; tradução Acauan Fernandes. - 1. Ed. - Rio de Janeiro : Novatec, 2013.

SILVEIRA, Paulo; TURINI, Rodrigo. Java 8 Prático: Lambdas, Streams e os novos recursos da linguagem. São Paulo: Cada do Código, 2014.

SHARDING in MongoDB. <http://docs.mongodb.org/manual/core/sharding-introduction/>. Acesso em 23 de agosto de 2014.

TANENBAUM, A. S. Distributed Operating Systems. Prentice Hall, 2007.

TRANSLATE sql to MongoDB MapReduce.

<http://nosql.mypopescu.com/post/392418792/translate-sql-to-mongodb-mapreduce>.Acesso em 13 de Março de 2014.

We do Hadoop. <http://br.hortonworks.com/why-hortonworks-for-hadoop/>. Acesso em 01 de Setembro de 2014.

WEISSMANN, Henrique Lobo. Spring Framework: Vire o jogo com. São Paulo: Cada do Código, 2013.

WILSON, Mike. Construindo Aplicações Node com MongoDB e Backbone; - 1. Ed. – São Paulo : Novatec, jan 2013.

WHAT IS APACHE MAHOUT. <https://mahout.apache.org/>. Acesso em 01 de Setembro de 2014.

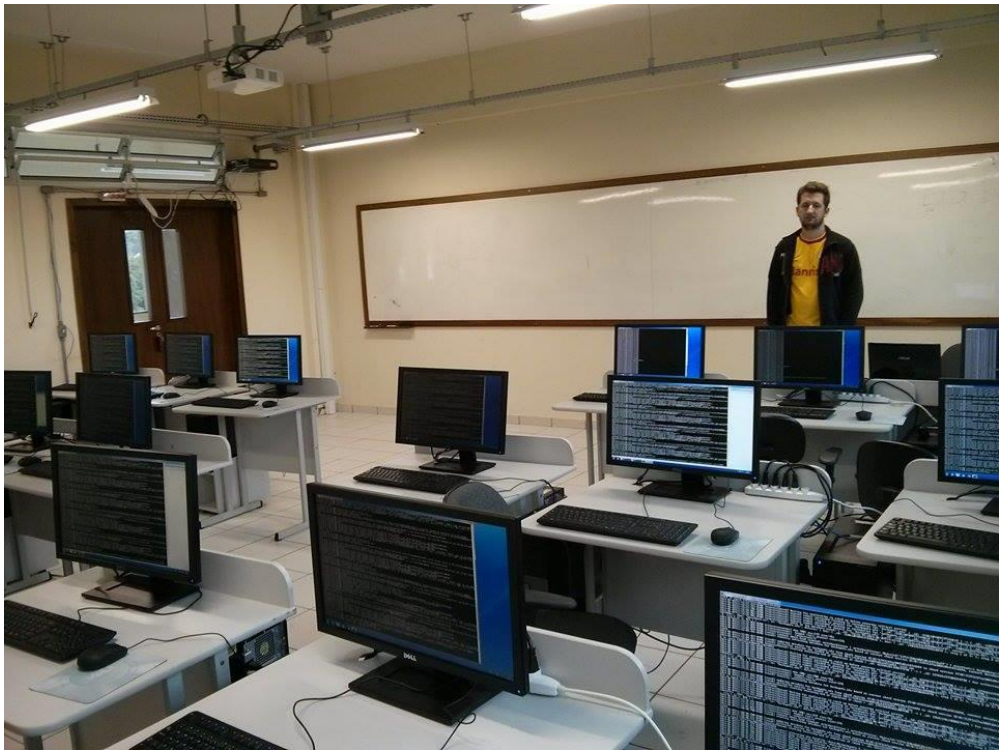
WHO Uses MongoDB?: Use Cases. Eua: Mongodb.com, 2014. 1 p. Disponível em: <http://www.mongodb.com/who-uses-mongodb>. Acesso em: 14 ago. 2014.

Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R., and Stoica, I. (2008). Improving mapreduce performance in heterogeneous environments. In Proceedings of the 8th USENIX conference on Operating systems design and implementation, OSDI'08, pages 29–42, Berkeley, CA, USA. USENIX Association.

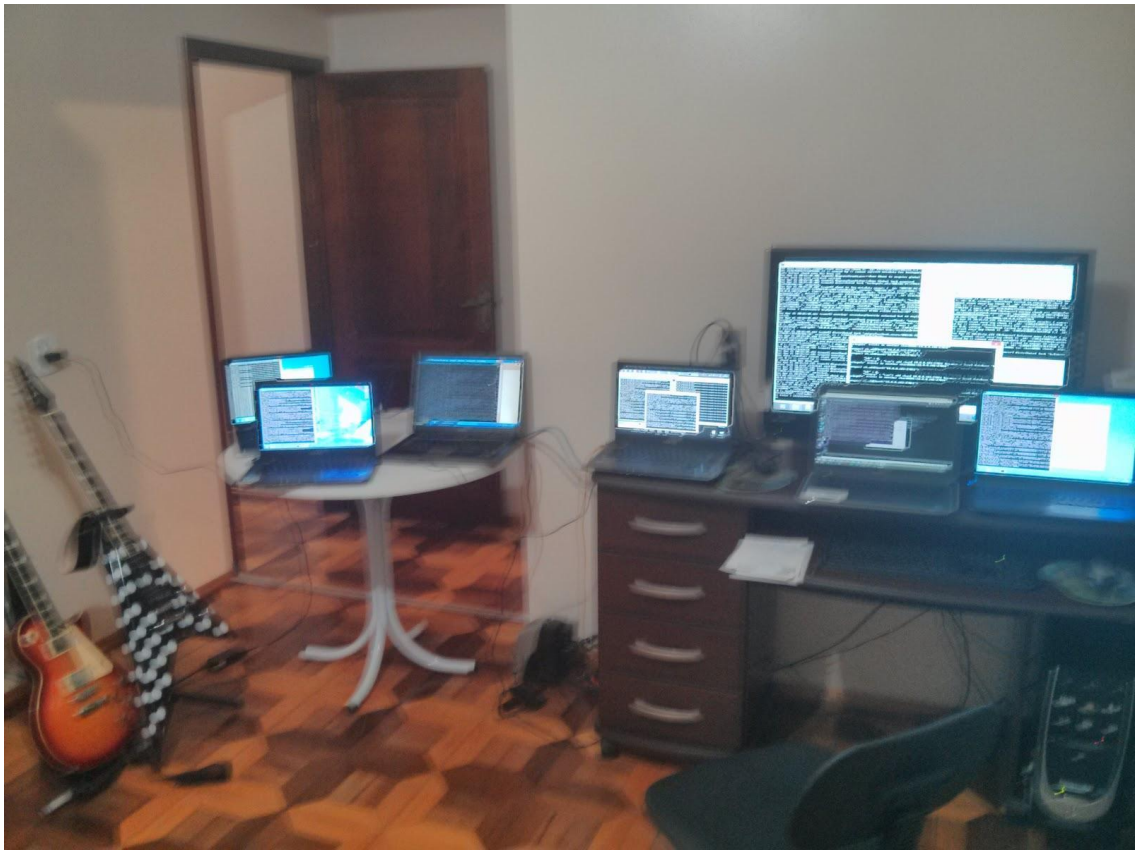
ZASLAVSKY, Arkady; PERERA, Charith; GEORGAKOPOULOS, Dimitrios; "Sensing as a service and big data," … Prepr. arXiv1301.0159, 2013.

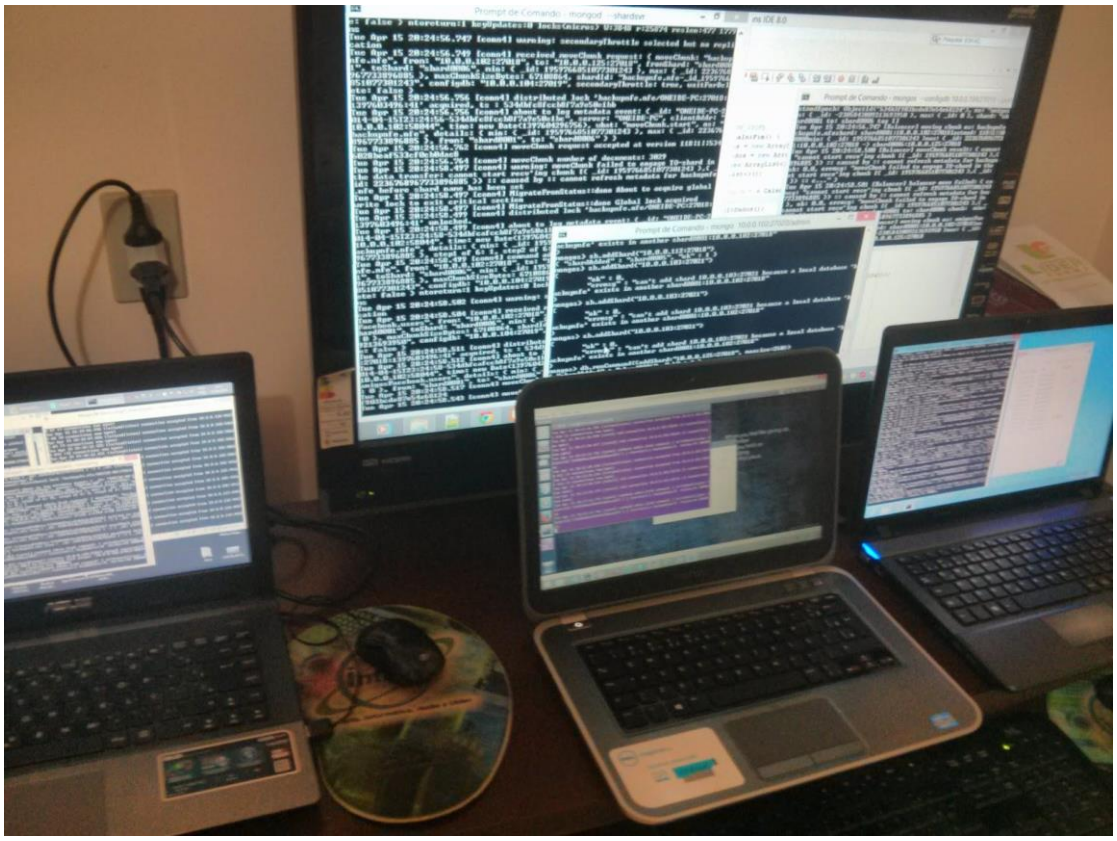
APÊNDICES

APÊNDICE A – Experimentos feitos nos Laboratórios da UTFPR. Fonte: Autor.



APÊNDICE B – Experimentos feitos em ambiente domestico. Fonte: Autor.





Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos

LEI Nº 9.610, DE 19 DE FEVEREIRO DE 1998.

Mensagem de veto

Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.

O PRESIDENTE DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Título I
Disposições Preliminares

Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

Art. 2º Os estrangeiros domiciliados no exterior gozarão da proteção assegurada nos acordos, convenções e tratados em vigor no Brasil.

Parágrafo único. Aplica-se o disposto nesta Lei aos nacionais ou pessoas domiciliadas em país que assegure aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes.

Art. 3º Os direitos autorais reputam-se, para os efeitos legais, bens móveis.

Art. 4º Interpretam-se restritivamente os negócios jurídicos sobre os direitos autorais.

Art. 5º Para os efeitos desta Lei, considera-se:

I - publicação - o oferecimento de obra literária, artística ou científica ao conhecimento do público, com o consentimento do autor, ou de qualquer outro titular de direito de autor, por qualquer forma ou processo;

II - transmissão ou emissão - a difusão de sons ou de sons e imagens, por meio de ondas radioelétricas; sinais de satélite; fio, cabo ou outro condutor; meios óticos ou qualquer outro processo eletromagnético;

III - retransmissão - a emissão simultânea da transmissão de uma empresa por outra;

IV - distribuição - a colocação à disposição do público do original ou cópia de obras literárias, artísticas ou científicas, interpretações ou execuções fixadas e fonogramas, mediante a venda, locação ou qualquer outra forma de transferência de propriedade ou posse;

V - comunicação ao público - ato mediante o qual a obra é colocada ao alcance do público, por qualquer meio ou procedimento e que não consista na distribuição de exemplares;

VI - reprodução - a cópia de um ou vários exemplares de uma obra literária, artística ou científica ou de um fonograma, de qualquer forma tangível, incluindo qualquer armazenamento permanente ou temporário por meios eletrônicos ou qualquer outro meio de fixação que venha a ser desenvolvido;

VII - contrafação - a reprodução não autorizada;

VIII - obra:

a) em co-autoria - quando é criada em comum, por dois ou mais autores;

b) anônima - quando não se indica o nome do autor, por sua vontade ou por ser desconhecido;

c) pseudônima - quando o autor se oculta sob nome suposto;

d) inédita - a que não haja sido objeto de publicação;

e) póstuma - a que se publique após a morte do autor;

f) originária - a criação primígena;

g) derivada - a que, constituindo criação intelectual nova, resulta da transformação de obra originária;

h) coletiva - a criada por iniciativa, organização e responsabilidade de uma pessoa física ou jurídica, que a publica sob seu nome ou marca e que é constituída pela participação de diferentes autores, cujas contribuições se fundem numa criação autônoma;

i) audiovisual - a que resulta da fixação de imagens com ou sem som, que tenha a finalidade de criar, por meio de sua reprodução, a impressão de movimento, independentemente dos processos de sua captação, do suporte usado inicial ou posteriormente para fixá-lo, bem como dos meios utilizados para sua veiculação;

IX - fonograma - toda fixação de sons de uma execução ou interpretação ou de outros sons, ou de uma representação de sons que não seja uma fixação incluída em uma obra audiovisual;

X - editor - a pessoa física ou jurídica à qual se atribui o direito exclusivo de reprodução da obra e o dever de divulgá-la, nos limites previstos no contrato de edição;

XI - produtor - a pessoa física ou jurídica que toma a iniciativa e tem a responsabilidade econômica da primeira fixação do fonograma ou da obra audiovisual, qualquer que seja a natureza do suporte utilizado*;*

XII - radiodifusão - a transmissão sem fio, inclusive por satélites, de sons ou imagens e sons ou das representações desses, para recepção ao público e a transmissão de sinais codificados, quando os meios de decodificação sejam oferecidos ao público pelo organismo de radiodifusão ou com seu consentimento;

XIII - artistas intérpretes ou executantes - todos os atores, cantores, músicos, bailarinos ou outras pessoas que representem um papel, cantem, recitem, declamem, interpretem ou executem em qualquer forma obras literárias ou artísticas ou expressões do folclore.

Art. 6º Não serão de domínio da União, dos Estados, do Distrito Federal ou dos Municípios as obras por eles simplesmente subvencionadas.