UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E DA TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Tiézer Costa de Melo

ESTUDO DE CASO DE BANCOS *NOSQL* NO CONTEXTO DE *BIG DATA*

Caxias do Sul
2016

Tiézer Costa de Melo

ESTUDO DE CASO DE BANCOS *NOSQL* NO CONTEXTO DE *BIG DATA*

Monografia apresentada como requisito para a obtenção do grau de Bacharel em Ciência da Computação da Universidade de Caxias do Sul.

Orientadora: Prof. Dra. Helena G. Ribeiro

Caxias do Sul
2016

Dedico este trabalho, especialmente, às três pessoas mais importantes da minha vida, meu pai Gilberto e minha mãe Cláudia, meus heróis e mestres, e à minha namorada Kelen, minha motivadora e porto seguro.

AGRADECIMENTOS

Agradeço, primeiramente, ao meu grandioso Deus, por ter permitido que tudo em minha vida fosse realizado.

Aos meus pais, Cláudia Roseila Costa de Melo e Gilberto Toloredo Fernandes de Melo, por serem os melhores pais que eu poderia ter, que me propiciaram tudo o que tenho ou realizei, colocando as minhas necessidades acima das deles, fazendo, muitas vezes, o impossível para me manter num bom caminho e por terem me dado toda a educação necessária.

À minha namorada, Kelen Salete Correa Soares, por ter me dado todo o apoio possível, por, muitas vezes, com uma palavra, me dar a motivação necessária ou a calma para enfrentar todos os desafios, pela paciência e e atenção.

Aos meus amigos e familiares, pelos momentos de descontração e palas palavras de incentivo.

Aos meus colegas, pelos momentos de risos e ensinamentos durante as aulas ou percurso da graduação.

À todos, citados ou não, mas que, de alguma forma, fizeram com que este trabalho fosse concluído, meus sinceros agradecimentos.. . .

"O insucesso é apenas uma oportunidade para recomeçar de novo com mais inteligência."

(Henry Ford)

RESUMO

Atualmente, a produção de informações atingiu uma escala nunca vista anteriormente. Esse fato traz com si novos desafios para as organizações, pois estas necessitam armazenar e processar essa quantidade de dados. Há muitos anos, os sistemas de gerenciamento de dados relacionais vêm dominando o mercado de sistemas de armazenamento de dados, porém, tendo em vista esse crescimento do volume de dados, estes sistemas não têm apresentado um desempenho satisfatório em consequência a algumas limitações. Por isso, foi necessário buscar diferentes alternativas a estes sistemas. Uma alternativa foi encontrada em sistemas que não se baseiam no modelo relacional, conhecidos como sistemas *NoSQL*, que apresentam novas características e propriedades e que foram desenvolvidos para o tratamento de grandes volumes de dados. O presente trabalho apresenta um estudo sobre sistemas *NoSQL*, dando-se uma maior ênfase aos sistemas orientados a documentos. Com base nesse estudo, será elaborada uma avaliação de três sistemas *NoSQL* orientados a documentos, *MongoDB*, *Couchbase* e *OrientDB*, sendo os seus desempenhos mensurados e avaliados com o uso da ferramenta de *benchmark YCSB* e através de métricas de avaliação específicas.

Palavras-chaves: Bancos de Dados. *NoSQL*. *MongoDB*. *Couchbase*. OrientDB. *Benchmark*. *YSCB*

ABSTRACT

Nowadays, the production of information reached a scale never seen before. This fact bring itself news challenges for the organizations, because they need to store and process that amount of data. For many years, the relational database systems management have dominated the data storage systems market, however, due this growth of data volume, these systems have no shown satisfactory performance due to some limitations. Because of this, was necessary to seek alternatives to those systems. Because of this, the need to seek alternatives to these systems. An alternative has been found in systems that are not based on the relational model, known as NoSQL systems, which introduces new properties and which have been developed for the treatment of large volumes of data. This paper presents a study of NoSQL systems, giving greater emphasis to the document-oriented systems. Based on this study will be compiled an evaluation three document-oriented NoSQL systems, MongoDB, Couchbase and OrientDB, and their performances measured and evaluated using the YCSB benchmark tool and through specific performance metrics evaluation..

Key-words: Databases. NoSQL. MongoDB. Couchbase. OrientDB. Benchmark. YCSB

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	*Application Programming Interface*
BASE	*Basically Available, Soft-State e Eventually Consistent*
BSON	*Binary Structured Object Notation*
CAP	*Consistency, Availability, Partition Tolerance*
CRUD	*Create, Read, Update e Delete*
IBM	*International Business Machines*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
MVCC	*Multi-version Concurrency Control*
NOSQL	*Not only SQL*
PPA	*Personal Package Archives*
RAM	*Random Access Memory*
SAP	*Systeme, Anwendungen und Produkte in der Datenverarbeitung*
SGBD	Sistema de Gerenciamento de Bancos de Dados
SGBDR	Sistema de Gerenciamento de Bancos de Dados Relacional
SQL	*Structured Query Language*
SSB	*Star Schema Benchmark*
TPC	*Transaction Processing Performance Council*
XML	*Extensible Markup Language*
YCSB	*Yahoo! Cloud Serving Benchmark*

SUMÁRIO

1 INTRODUÇÃO

A humanidade nunca produziu tanta informação quanto nas últimas décadas. Numa escala, anteriormente inimaginável e praticamente impossível de se mensurar, os dados foram gerados através dos mais diversos dispositivos e mídias. Especialistas apontam que, atualmente, 80% dos dados são compostos por dados semi ou não estruturados, trazendo uma maior dificuldade para análise dos mesmos (IBM, 2015). Estes dizem que os dados terão, num futuro próximo, o mesmo valor representativo do que, atualmente, tem o petróleo e o ouro (PETRY, 2013). Mas estes dados não eram, até então, corretamente aproveitados, sendo descartados antes mesmo de sua análise. O conceito *Big Data*, que será abordado nos capítulos seguintes, trouxe consigo uma nova abordagem para o tratamento e processamento das informações, juntamente a vantagens competitivas para as organizações. Porém, tais vantagens envolvem uma maior complexidade para este armazenamento e processamento. Com intuito de manter os dados armazenados, as organizações usam de uma base de dados, que pode ser considerada como um conjunto ou coleção de dados ou registros, que são persistidos em algum meio de armazenamento, tal como memória ou disco rígido, visando atender os interesses desta organização. Persistidos entende-se como sendo a característica de que uma vez salvos na base de dados, os dados deverão manter-se lá mesmo após a inicialização do servidor, seja esta inicialização por motivo intencional ou não, podendo, apenas, serem excluídos por meio de transação ou requisição. Transação ou requisição entende-se como sendo algum tipo de operação sobre a base de dados, como uma transação de leitura, escrita ou alteração de dados. Comumente, os dados ou registros são recuperados dos discos para a memória apenas quando estes necessitam ser acessados ou manipulados. Armazenar, gerenciar e extrair os dados, seja essa base pequena ou muito grande, como é o caso do *Big Data*, é função do sistema de gerenciamento de banco de dados, sigla SGBD (RAMAKRISHNAN; GEHRKE, 2008). Os atuais e mais utilizados SGBD no mercado ainda são os relacionais. Mas estes já não conseguem, em um tempo considerável, suprir a demanda de controle da gigantesca quantidade e variedade de dados armazenados e extrair a informação desejada, sem envolver uma maior complexidade, tempo e investimento financeiro (REEVE, 2012). Deparando-se com isso, gigantes da área da tecnologia (Facebook[1], Google[2], Amazon[3], Twitter[4], Yahoo[5], entre outros), buscaram alternativas para este problema, encontrando-as em sistemas com características de armazenamento de dados não relacionais, conhecidos como *NoSQL* (acrônimo, em língua inglesa, de *Not Only SQL*). Utilizando de abordagens diferentes ao modelo

[1] http://www.facebook.com
[2] http://www.google.com
[3] http://www.amazon.com
[4] http://www.twitter.com
[5] http://www.yahoo.com/

relacional, estes sistemas foram desenvolvidos para terem alto desempenho na utilização de grandes volumes de dados, também demonstram-se muito eficientes no controle de dados estruturados, semiestruturados ou não estruturados, além de permitir escalabilidade horizontal (aumentar o número de equipamentos e servidores para processamento), suporte nativo a replicação e ser livre de esquema (estrutura variável) (TIWARI, 2011). Devido a forma como armazenam os dados, os sistemas *NoSQL* podem ser divididos em 4 grupos: documentos, chave/valor, colunas e grafos. Devido a necessidade de identificar os pontos positivos e negativos de sistemas *NoSQL*, faz-se necessário utilizar de ferramentas de avaliação específicas, conhecidas como *benchmark*.

1.1 Justificativa

Soluções de armazenamento de dados relacionais, denominados SGBD relacionais ou SGBDR, apresentam-se ineficientes com o aumento da necessidade de análise e processamento de grandes volumes e variedade de dados (dados semiestruturados ou não estruturados) a serem armazenados e analisados, como é o caso do *Big Data*. As soluções de armazenamento de dados não relacionais, conhecidos como *NoSQL*, apresentam um desempenho significativamente bom nos casos anteriormente citados como ineficiência dos sistemas relacionais, visto que são destinados ao alto desempenho no processamento de dados e são destinados para grandes volumes de dados. Neste trabalho, pretende-se verificar o desempenho de três sistemas *NoSQL* orientados a documentos através do uso de uma ferramenta de *benchmark* e seguindo algumas métricas específicas de avaliação.

1.2 Objetivos

O principal objetivo deste trabalho é avaliar o desempenho de três sistemas de bancos de dados *NoSQL* orientados a documentos (*MongoDB*, *Couchbase* e *OrientDB*) por meio do *benchmark YCSB*. Para tal avaliação, inicialmente, serão identificadas algumas métricas de avaliação adequadas ao trabalho e, posteriormente, através da ferramenta de *benchmark*, serão executados alguns testes de comparação para levantamento de informações sobre o desempenho destes sistemas, adequadas ao contexto do trabalho, visando efetuar a avaliação dos mesmos segundo as métricas definidas.

1.3 Organização do texto

O trabalho está organizado, conforme a estrutura apresentada a seguir: o Capítulo 2 apresenta a fundamentação teórica e os conceitos relativos aos assuntos abordados e necessários para a realização do trabalho, juntamente a alguns trabalhos relacionados ao atual trabalho. O Capítulo 3 apresenta as tecnologias que serão utilizadas no processo de desenvolvimento e realização dos testes nos sistemas selecionados. O Capítulo 4 trata

da proposta de solução a ser desenvolvido neste trabalho. O Capítulo 5 apresenta a realização dos testes sobre os sistemas e a avalização dos resultados destes testes. Por fim, o Capítulo 6 são apresentadas algumas considerações sobre o trabalho, contribuições do mesmo e possíveis temas para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados as definições e conceitos que são considerados relevantes para o trabalho, embasados através de literaturas diversas. Primeiramente, Na seção 2.1, será apresentado o assunto *Big Data*. Na seção 2.2 será apresentado o Teorema CAP. Na seção 2.3, serão apresentados os sistemas de gerenciamento de bancos de dados. Na seção 2.4, será apresentado o conceito de *Benchmark*. Na seção 2.5, serão abordados os trabalhos relacionados ao tema deste trabalho.

2.1 *Big Data*

Setores acadêmico, comercial e social estão gerando informações numa escala inimaginável, nunca vista anteriormente. Estes dados são provenientes das mais diversas fontes (redes sociais, transações financeiras, e-mails, arquivos, dados multimídia, sensores, *e-commerces*, etc.), não mais seguindo um padrão de estruturação e apresentando uma maior complexidade para análise. A dificuldade de análise destes dados se deve ao seu elevado volume e a sua complexidade de modelo de dados (dados semiestruturados ou não estruturados em sua maioria), como é o caso do *Big Data*. *Big Data* pode ser definido como ativos de informação com grande volume, alta velocidade e variedade que procura formas inovadoras e rentáveis de processamento destas informações, permitindo maior compreensão, tomada de decisão e automação de processos (GARTNER, 2015). Ainda, segundo GARTNER (2015), o *Big Data* pode ser resumidamente caracterizado por 3 Vs: volume, velocidade e variedade, conforme descrição abaixo:

- Volume: devido à quantidade de dados gerados, que cresce exponencialmente.
- Velocidade: devido à velocidade com que os dados são gerados e a necessidade de se obter uma resposta imediata, em tempo real, em buscas sobre os mesmos.
- Variedade: devido as diferentes estruturas, formatos e características dos dados, que são gerados das mais diversas fontes.

Alguns outros autores, como MARR (2015) e Demchenko et al. (2013), defendem que o *Big Data* possui mais duas características, 2 Vs, que são:

- Valor: consiste da necessidade ou capacidade de tornar os dados em valor para a organização.
- Veracidade: consiste da necessidade de que os dados sejam verdadeiros ou confiáveis.

De acordo com GARTNER (2012), em tradução livre, "O *Big Data* cria uma nova camada na economia que trata da informação, transformando informações, ou dados, em receita". O termo *Big Data* não está ligado, apenas, ao armazenamento e volume de informações, mas, também, ao processamento e análise dos mesmos. Quando os dados são armazenados, estes não sofrem nenhum tipo de alteração ou tratamento até o momento de sua análise. Isso ocorre com intuito de evitar, que haja a possibilidade de algum dado, considerado dispensável no momento do armazenamento, seja descartado e, posteriormente, ser considerado relevante para a organização. Há algum tempo, os dados não eram corretamente aproveitados, sendo descartados antes mesmo de sua análise (GOMES, 2013). Porém, percebeu-se que é justamente a correta análise dos dados que os torna um diferencial para uma organização. Torna-se possível a tomada de decisões em tempo real, trazendo benefícios favoráveis para as organizações e auxiliando para a entrega de melhores valores para o cliente. Alguns exemplos de empresas que utilizam do *Big Data* em seus ambientes são: os *e-commerces* (*Ebay*[1], *Amazon* e *Walmart*[2]), o *Google*, o *Facebook*, o *Twitter*, o *SAP*[3] e o *CERN*[4].

2.2 O Teorema Cap

Quanto maior o volume de dados, maior é a tendência e necessidade de divisão e distribuição dos mesmos em dois ou mais servidores, visando aumentar o desempenho do sistema. No ano 2000, o professor da Universidade da Califórnia, Eric Brewer, propôs uma conjectura para sistemas distribuídos. Esta conjectura foi posteriormente comprovada em 2002, por Seth Gilberty e Nancy A. Lynch (GILBERT; LYNCH, 2012; CELKO, 2013), se tornando um teorema, conhecido como Teorema de CAP (acrônimo, em língua inglesa, das iniciais de *Consistency*, *Availability* e *Partition Tolerance*), ou Teorema de Brewer (SULLIVAN, 2015). Segundo este teorema, em sistemas distribuídos, é requerido que sejam atendidas as propriedades de consistência, disponibilidade e tolerância a particionamento, mas que só é possível garantir duas destas três propriedades simultaneamente (TIWARI, 2011). Tais propriedades são descritas abaixo:

- Consistência: todos os nós do sistema terão a mesma cópia dos dados. Esta propriedade apresenta definições diferentes em sistemas ACID e no Teorema CAP. Em sistemas ACID, consistência significa que a base de dados sempre apresentará um estado consistente. Já no Teorema CAP, significa que, no mínimo, um dos nós do sistema conterá uma base de dados atualizada.

[1] http://www.ebay.com
[2] http://www.walmart.com
[3] http://go.sap.com
[4] http://home.cern/

- Disponibilidade: esta é a propriedade que diz que uma requisição sempre será atendida, ao menos, por um nó, mesmo que sendo com os dados não mais recentes ou com mensagem de não funcionamento do sistema. Esta propriedade é garantida se ignorando falhas parciais no sistema e desde que não haja falha em todos os nós. Por exemplo, se um banco de dados está dividido em três nós e um deles falha, apenas as requisições feitas a este determinado nó estará prejudicada, e o funcionamento do sistema não será interrompido.

- Tolerância a particionamento: esta propriedade está relacionada a capacidade do sistema manter-se em funcionamento e respondendo a requisições, mesmo que ocorram problemas de comunicação (rede) entre os servidores ou problema de *hardware* nos mesmos. Pode ocasionar a divisão do sistema em dois ou mais grupos que não se comunicam entre si.

De acordo com Monson-Haefel (2009), as três propriedades até poderiam ser atendidas, mesmo que não houvesse a garantia de realização destas três propriedades simultaneamente, mas acarretaria no aumento da complexidade e do custo computacional. Devido à escolha necessária de apenas duas das três propriedades anteriormente citadas, os sistemas dividem-se em três grupos, como visto na Figura 1, que são citados abaixo:

- AP: *availability* e *partition tolerance*, que em língua portuguesa significa disponibilidade e tolerância a particionamento, garantem a disponibilidade e a tolerância a particionamento, e a consistência é eventual.

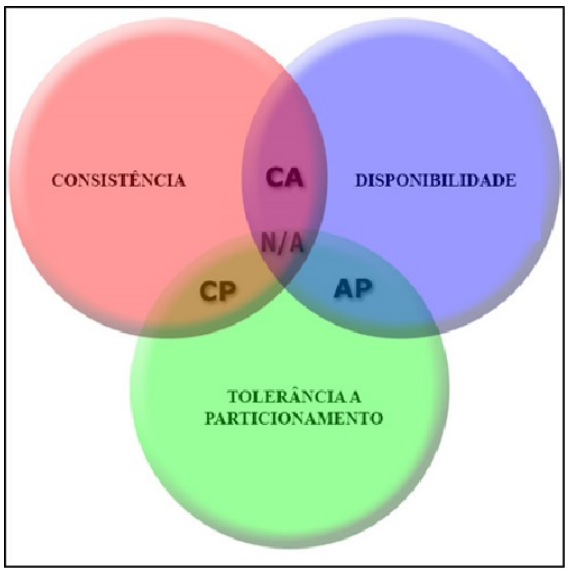
- CA: *consistency* e *availability*, que em língua portuguesa significa consistência e disponibilidade, garantem a consistência e a disponibilidade, mas apresentam problemas com a tolerância a particionamento. Caso ocorra uma falha de comunicação, o sistema todo pode falhar. Não sabem lidar com falhas de particionamento, podendo o sistema como um todo ficar indisponível.

- CP: *consitency* e *partition tolerance*, que em língua portuguesa significa consistência e tolerância a particionamento, garantem consistência e tolerância a particionamento, sacrificando a disponibilidade.

2.3 Sistemas de Gerenciamento de Bancos de Dados

O atual desafio das organizações encontra-se em armazenar e processar o seu volume de dados, sejam estes volumes pequenos ou extremamente grandes, como é o caso do Big Data. Segundo Brito (2010), as funções dos sistemas de gerenciamento de bancos de dados, de sigla SGBD, são: controle de acessos, validação e verificação dos dados, controle de concorrência, recuperação de falhas, segurança do modelo e dos dados, controle de

Figura 1 – Representação do Teorema CAP e suas propriedades.



Fonte: (BROWN, 2011)(Adaptado)

transações, otimização das consultas, recuperação de falhas, entre outras. Basicamente, os SGBD utilizam as operações CRUD, que é o acrônimo, em língua inglesa, das iniciais de Create, Read, Update e Delete, para o gerenciamento de uma base de dados (REDMOND; WILSON, 2012, p. 14). Em português, estas operações significam, respectivamente, Criar, Ler, Atualizar e Excluir. A seguir, são descritas estas operações:

- Criar: operação que cria registros ou dados no banco de dados.
- Ler: operação de busca de dados que já estão armazenados no banco de dados.
- Atualizar: operação que modifica ou altera dados já existentes no banco de dados.
- Excluir: operação que remove dados do banco de dados.

Neste trabalho serão apresentados os sistemas de gerenciamento de bancos de dados relacionais, descritos na subseção 2.3.1, e os sistemas de gerenciamento de bancos de dados não relacionais, também conhecidos como SGBD NoSQL, descritos na subseção 2.3.2. Apesar de existirem alguns outros modelos de SGBD, tais como os sistemas orientados a objetos e os sistemas *NewSQL*, os mesmo não fazem parte deste trabalho, portanto, não serão apresentados.

2.3.1 Sistemas de Gerenciamento de Bancos de Dados Relacionais

Os SGBD ditos tradicionais, seguem a metodologia relacional e são conhecidos como SGBD Relacionais, de sigla SGBDR, sendo, há anos, os sistemas de armazenamento mais utilizados pelas organizações (BRITO, 2010). Alguns exemplos dos SGBDR são

o SQL Server da Microsoft[5], o Oracle Database[6] e o MySQL[7] da Oracle Corporation[8], o IBM DB2[9] da IBM[10] e o PostgreSQL[11] da PostgreSQL Global Development Group. Foram inicialmente propostos por Edgard Frank Todd, pesquisador da IBM, em torno de 1970, como resultado de um estudo teórico e tendo como base a álgebra relacional (VANDERLINDE, 2013). Apesar de ter sido apresentado na década de 1970, este modelo foi apenas implementado na década de 1980, quando versões comercias começaram a ser disponibilizadas (MONIRUZZAMAN; HOSSAIN, 2013). O modelo relacional organiza e representa o banco de dados como relações, geralmente chamadas de tabelas, que pode ser melhor compreendido como uma matriz bidimensional (REDMOND; WILSON, 2012, p. 3). De acordo com Jatana et al. (2012), estas tabelas são compostas por linhas e colunas. Neste modelo, cada coluna representa um atributo ou conjunto de valores, com o respectivo tipo e formato de dado. Cada linha representa um único registro ou instância do dado. Conforme Brito (2010), o relacionamento entre tabelas é representado através de chaves, caso da chave primária, representada na cor vermelha, e da chave estrangeira, representada na cor azul (Figura 2). As chaves primárias servem para a identificação única e exclusiva dos registros ou linhas, servindo para diferenciar um registro de outro, já que uma chave primária nunca será repetida na tabela, evitando, assim, a redundância de dados. A chave estrangeira serve para representar o relacionamento existente entre os dados de duas tabelas distintas, fazendo referência à chave primária,e necessariamente à chave primária, da tabela com quem existe o relacionamento.

Estes sistemas baseiam-se na propriedade ACID acrônimo que deriva das iniciais das propriedades Atomicidade, Consistência, Isolamento e Durabilidade (SOUZA; SANTOS, 2015). Abaixo, são descritas estas propriedades:

- Atomicidade: Em resumo, ou a operação é executada por completo, ou nada é executado e a operação é cancelada;
- Consistência: Após a realização de uma transação, os dados deverão permanecer em estado consistente;
- Isolamento: caso duas transações estejam sendo executadas ao mesmo tempo, concorrentemente, seus efeitos devem ser isolados um do outro, isto é, uma transação é independente e isolada de outra;
- Durabilidade: após uma transação realizada com sucesso, o efeito desta na base de dados não poderá ser desfeito, mesmo após o sistema ser reiniciado.

[5] http://www.microsoft.com/pt-br/server-cloud/products/sql-server/
[6] http://www.oracle.com/
[7] https://www.mysql.com/
[8] http://www.oracle.com/
[9] http://www.ibm.com/software/data/db2/
[10] http://www.ibm.com/
[11] http://www.postgresql.org/

Figura 2 – Representação de uma tabela de um SGBD relacional, das chaves primárias e chaves estrangeiras.

TABELA ALUNO		
ID	NOME	SOBRENOME
100	JOAO	COSTA
101	MARIA	SANTOS
102	JOSE	DINIZ
103	PEDRO	FONSECA
104	PAULO	MELO
105	CARLA	SILVA
106	FRANCISCO	SOARES
107	DAIANE	SOUZA
108	MARCIA	PEREIRA
109	MAICON	NASCIMENTO

TABELA AVALIACAO			
ID	ID_ALUNO	NOTA_1	NOTA_2
1	100	5	8
2	101	10	10
3	109	7	7
4	103	8	6
5	100	9	7
6	108	6,5	8
7	105	9	10
8	102	8	7,5
9	102	6	9
10	107	6	7

Fonte: Elaborado pelo autor.

Estas propriedades visam garantir a integridade de uma transação e, consequentemente, a integridade e consistência da própria base de dados (JATANA et al., 2012). Porém, ao garantir as propriedades ACID, o sistema deve realizar um grande número de bloqueios entre transações, que causa problemas de desempenho, pois as operações necessitam aguardar que outras operações sejam finalizadas quando ambas necessitam acessar os mesmos dados. Outra forte característica dos SGBDR é a utilização de uma linguagem padronizada de gerenciamento, o SQL, acrônimo, em língua inglesa, das iniciais de *Structured Query Language*, que permite manipular os dados de forma eficiente e simples (JATANA et al., 2012). De acordo com Brito (2010), apesar de os SGBDR serem sistemas consolidados no mercado, tais soluções estão apresentando problemas de desempenho e falta de eficiência no tratamento de tipo de dados mais complexos, como é o caso do *Big Data*, seja em virtude do seu volume, ou em virtude da sua complexidade e estrutura (sabe-se que a grande maioria dos dados atualmente, estima-se que 80% destes, são semiestruturados ou não estruturados) (IBM, 2015). Souza e Santos (2015) e Jatana et al. (2012) dizem que estes sistemas não foram desenvolvidos para trabalhar em ambientes distribuídos e a escalabilidade, em geral, se dá pelo chamado escalonamento vertical, que é a melhora da capacidade de processamento do servidor, como , por exemplo, a substituição do processador, unidade de armazenamento, memória, etc., por uma com melhor desempenho.

2.3.2 Sistemas de Gerenciamento de Bancos de Dados NoSQL

Tentando sanar tais problemas encontrados nos sistemas relacionais, suprir as necessidades onde estes bancos são ineficazes (escalabilidade, disponibilidade e suporte a grande volume de dados), gigantes da área da tecnologia (Facebook, Google, Amazon,

Twitter, Yahoo) buscaram encontrar soluções alternativas aos SGBDR. Tais soluções foram encontradas nos chamados bancos de dados não relacionais, que não adotam o modelo relacional como padrão (não se baseiam no modelo relacional), também conhecidos como bancos de dados *NOSQL*, acrônimo, em língua inglesa, de *Not Only SQL* (BRITO, 2010). Estes sistemas foram desenvolvidos para lidar com grandes volumes de dados, atendendo a necessidade de alto desempenho (rápido processamento e baixo tempo de resposta) e disponibilidade (estar disponível o maior tempo possível) (LÓSCIO; OLIVEIRA; PONTES, 2011). Segundo Duda (2012) e Abramova, Bernardino e Furtado (2014b), o termo NoSQL foi inicialmente utilizado em 1998, por Carlo Strozzi, para denominar e identificar bancos de dados que não seguiam o modelo relacional. Mas foi retomado apenas no ano de 2009, no evento *NoSQL Meetup*, organizado por Johan Oskardson, o criador do *Last.fm*[12], onde o nome foi sugerido por Eric Evans, desenvolvedor do *Rackspace*[13], que teve com como objetivo discutir o crescente surgimento de soluções de armazenamento de dados distribuídos não relacionais de código livre (ABRAMOVA; BERNARDINO; FURTADO, 2014b). Atualmente existem, aproximadamente, 225 bancos de dados NOSQL (NOSQL, 2015). De acordo com (BRITO, 2010), os sistemas NoSQL não surgiram com o intuito de substituir os bancos de dados relacionas, mas servir como alternativa para estes, trazendo novas opções e características. Existem casos específicos onde um destes trabalha melhor que o outro, e há casos onde os dois são utilizados juntos. Abaixo, algumas das principais características dos sistemas NoSQL são descritas:

- Suporte a diferentes tipos e estruturas de dados: este modelo apresenta suporte a diferentes estruturas e tipos de dados. Foi designado para manipular dados estruturados, semi ou não estruturados(permite que existam dados com estruturas diferentes na mesma base de dados ou coleção), além de manipular base de dados de diferentes tipos (orientada a documentos, chave/valor, orientada a colunas, orientada a grafos, etc.).

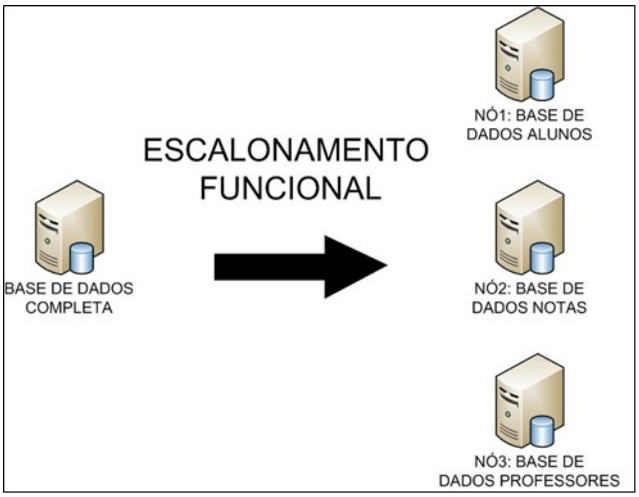
- Escalonamento horizontal: consiste em distribuir os dados em dois ou mais computadores para o seu armazenamento e processamento. Possibilita a divisão do processamento em tarefas menores, deixando de utilizar apenas um único e potente computador, passando a utilizar computadores de menor porte no processamento dos dados, processo conhecido como escalonamento vertical, permitindo, também, a adição de mais computadores ao sistema sem envolver maiores complexidades. Um dos grandes utilizadores desta metodologia é a Google, que utiliza computadores de pequeno e médio porte para armazenagem e distribuição de dados, com utilização mais eficiente dos recursos e mais econômica. Através do escalonamento horizontal, é

[12] http://www.last.fm/
[13] https://www.rackspace.com/

possível distribuir os dados de duas formas: a primeira é através do particionamento funcional,

- Particionamento funcional: consiste na distribuição de diferentes tabelas ou documentos entre diferentes nós do sistema, isto é, distribuir a tabela alunos em um servidor, a tabela notas em outro, e a tabela professores e um terceiro servidor, como visto na Figura 3.

Figura 3 – Representação do escalonamento funcional.



Fonte: Elaborado pelo autor.

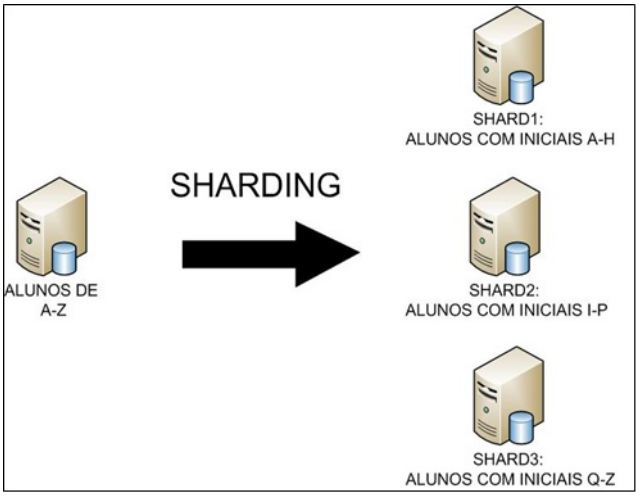
- Sharding: é a distribuição dos dados de uma mesma tabela ou documento em diferentes nós do sistema, seguindo algum critério específico, como, por exemplo, dividir a tabela aluno em três servidores distintos, seguindo o critério de que alunos com iniciais de A a H terão os seus dados armazenados em um servidor, alunos com inicias de I a P terão os seus dados armazenados em um segundo servidor e alunos com iniciais de Q a Z terão seus dados armazenados em um terceiro servidor, como visto na Figura 4.

O escalonamento horizontal possibilita que sejam realizadas requisições ou transações de forma paralela. Caso algum servidor utilizado no escalonamento funcional ou no *sharding* fique indisponível, os dados armazenados neste estarão indisponíveis. Para resolver tais problemas, uma solução é misturar o *sharding* com replicação.

- Sem esquema definido ou esquema flexível: não exigem declaração de tipos de dados, pois não possui estrutura de dados rígida. Assim, não é necessária alguma mudança do esquema do banco, como é exigido no SGBDR (neste, a estrutura dos dados precisa ser definida antes do seu armazenamento).

- Suporte nativo a replicação: esta característica está relacionada a armazenar cópias dos dados em diferentes nós (redundância de dados), implicando diretamente no

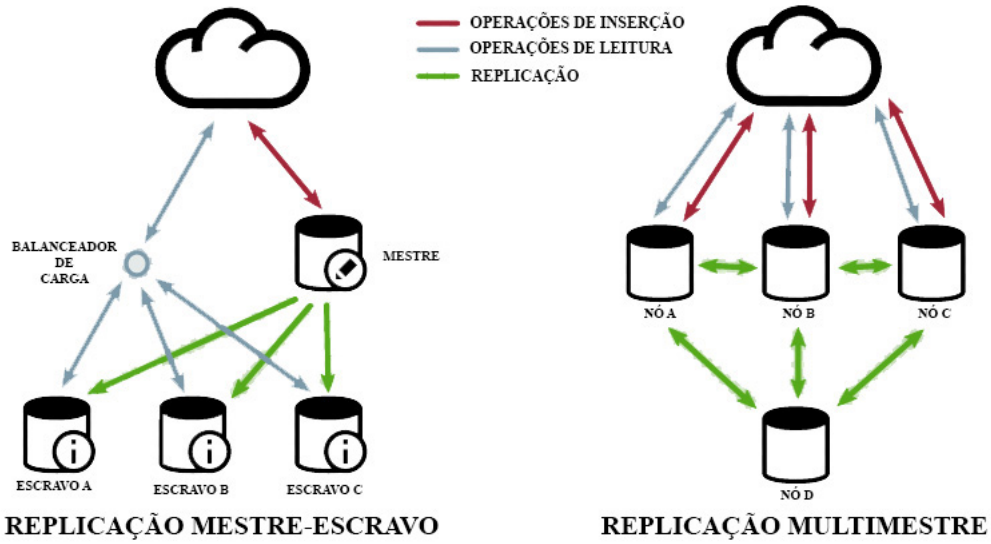
Figura 4 – Representação do *sharding*.



Fonte: Elaborado pelo autor.

grau de disponibilidade e desempenho do sistema. Consiste da junção das técnicas de replicação mestre-escravo (as leituras são realizadas em todos os nós do sistema, mas as inserções são apenas realizadas no nó mestre, que também efetua a replicação dos dados para os nós escravos) e multimestre (todas as operações de leitura, inserção e replicação pode ser realizadas por qualquer nó do sistema), como apresentado resumidamente na Figura 5 e das estratégias de propagação de dados síncrona e assíncrona.

Figura 5 – Exemplo da estrutura dos modelo mestre-escravo e multimestre.



Fonte: (ERB, 2012, p. 111)(Adaptado).

Diferentemente dos sistemas relacionais, a grande maioria dos sistemas NoSQL não fazem uso do modelo ACID, sendo, então, adotado o modelo *BASE*, acrônimo que deriva,

em língua inglesa, das iniciais das propriedades *Basically Available*, *Soft-State* e *Eventually Consistent*. Em português, esta propriedades significam, respectivamente, Basicamente Disponível, Estado Leve e Eventualmente Consistente (ABRAMOVA; BERNARDINO; FURTADO, 2014b). Estas propriedades são caracterizadas pela alta disponibilidade dos dados, enquanto que a consistência dos dados é dispensada, isto é, da-se maior importância para que os dados estejam disponíveis a maior parte do tempo (disponibilidade) do que a necessidade de que todos os nós estejam com a mesma versão dos dados (consistência). O modelo BASE é exemplificado, abaixo:

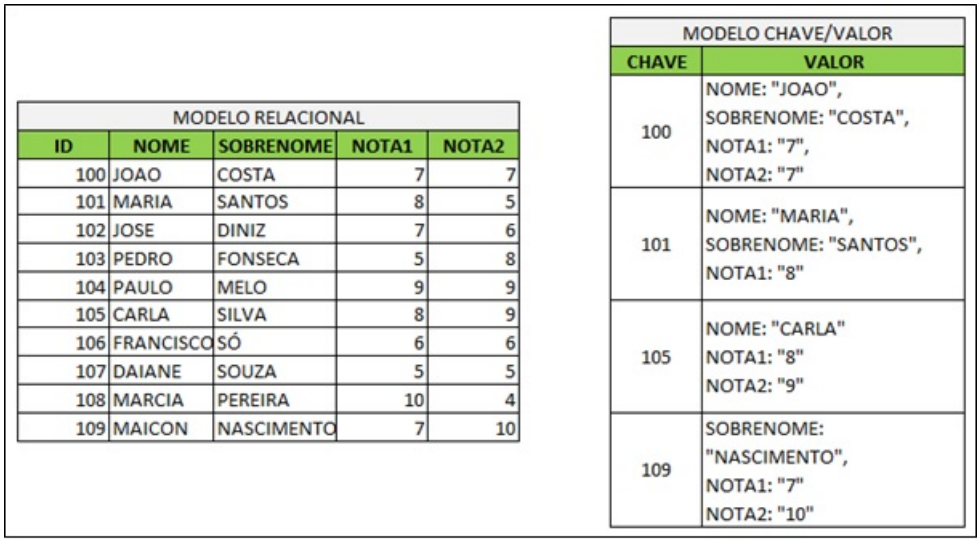
- Basicamente disponível: o sistema deve estar em funcionamento na maior parte do tempo, mesmo que ocorra a indisponibilidade temporária de um ou mais nós do sistema, mas nunca de todos os nós.
- Estado leve: abandona a necessidade de garantia de consistência de dados em tempo integral apresentado no modelo relacional, isto é, o sistema não precisa apresentar-se consistente em tempo integral, mas podendo ser consistente em momentos indeterminados. As réplicas dos dados não precisam ser consistentes entre si o tempo todo, mas necessita funcionar normalmente mesmo com a ocorrência de falhas.
- Eventualmente Consistente: a consistência nem sempre é mantida para todos os nós da rede, isto é, os nós podem não ter a mesma versão dos dados. O período de inconsistência existente entre uma atualização e o momento em que se é garantido que cada nó conterá os dados provenientes desta atualização é chamado de janela de inconsistência. Isto ocorre devido às atualizações serem propagadas eventualmente para todos os nós.

Devido à forma de armazenamento dos dados, os SGBD NoSQL podem ser divididos em quatro grupos, conforme descrição abaixo:

- Chave/valor: neste modelo, cada registro é representado por um par de chaves e valores, reconhecido, também, como tabela *Hash*. Uma tabela *Hash* é uma forma de representar uma estrutura de dados, onde através de uma chave, seja possível endereçar diretamente um registro. Esse endereçamento é realizado através de uma função sobre parte da chave do registro para o cálculo da localização do valor do mesmo, de forma semelhante a um mapa. Cada uma destas chaves é única e permite o acesso o direto ao registro, e, devido a haver uma única chave para indexação, apresenta simplicidade nas operações sobre os dados, pois todas as operações efetuadas na base de dados são realizadas através destas chaves, sendo este o motivo do alto desempenho deste tipo de sistema. As operações destes sistemas se resumem em *PUT*, *GET* e *DELETE*. Como exemplo destes bancos de dados, podemos

citar: *Amazon DynamoDB*[14], *Riak*[15], *Redis*[16], Project *Voldemort*[17] e *HamsterDB*[18]. Na Figura 6, um exemplo de uma base de dados em sistemas chave/valor.

Figura 6 – Exemplo de uma base de dados em sistemas chave/valor



MODELO RELACIONAL

ID	NOME	SOBRENOME	NOTA1	NOTA2
100	JOAO	COSTA	7	7
101	MARIA	SANTOS	8	5
102	JOSE	DINIZ	7	6
103	PEDRO	FONSECA	5	8
104	PAULO	MELO	9	9
105	CARLA	SILVA	8	9
106	FRANCISCO	SÓ	6	6
107	DAIANE	SOUZA	5	5
108	MARCIA	PEREIRA	10	4
109	MAICON	NASCIMENTO	7	10

MODELO CHAVE/VALOR

CHAVE	VALOR
100	NOME: "JOAO", SOBRENOME: "COSTA", NOTA1: "7", NOTA2: "7"
101	NOME: "MARIA", SOBRENOME: "SANTOS", NOTA1: "8"
105	NOME: "CARLA" NOTA1: "8" NOTA2: "9"
109	SOBRENOME: "NASCIMENTO", NOTA1: "7" NOTA2: "10"

Fonte: Elaborado pelo autor.

- Orientados a documentos: é considerado por muitos a evolução dos sistemas chave/-valor. Os dados são armazenados em forma de coleções de documentos (analogia de tabela e registro), onde uma coleção pode conter um ou mais documentos. Um documento representa a unidade de armazenamento de dados, que nos SGBDR seriam chamados de registros ou linhas. Em geral, não possuem esquema de dados fixo, ou seja, permite que os documentos armazenados possuam estruturas e tipos diferentes. Permite, que a estrutura de um documento seja atualizada, apenas adicionando ou excluindo campos no documento(LÓSCIO; OLIVEIRA; PONTES, 2011). Cada documento possui um campo que serve como um identificador único e é composto por um conjunto de campos, onde não há restrições quanto ao número de campos no documento. Os documentos são normalmente armazenados em formatos de representação de dados específicos, tais como *JSON*[19], *BSON*[20], *XML*[21], entre outros. O *BSON* (acrônimo, em língua inglesa, de *Binary JSON*) é uma representação binária de documentos *JSON* utilizada pelo sistema *MongoDB*. Os documentos *BSON* possuem todas as características do modelo *JSON*, mas adiciona dados dos tipo data e binário, que não são disponibilizados no *JSON*. Permite, também, que sejam

14 aws.amazon.com/dynamodb/
15 http://basho.com/products/riak-kv/
16 http://redis.io/
17 http://www.project-voldemort.com/voldemort/
18 http://hamsterdb.com/
19 http://www.json.org/
20 http://bsonspec.org/
21 https://www.w3.org/XML/

realizadas consultas através das chaves ou através do conteúdo de um documento (campos do documento). A seguir, a representação de um documento no formato *XML*, representando os dados de um aluno e suas respectivas notas:

```
<pessoa>
    <ID> 100 </ID>
    <nome> Joao </nome>
    <sobrenome> Costa </sobrenome>
    <notas>
        <nota1> 7 </nota1>
        <nota2> 7 </nota2>
    </notas>
</pessoa>
```

Já, abaixo, a representação dos mesmos dados apresentados anteriormente, mas, ao invés de utilizar o *XML*, utilizando o formato *JSON*.

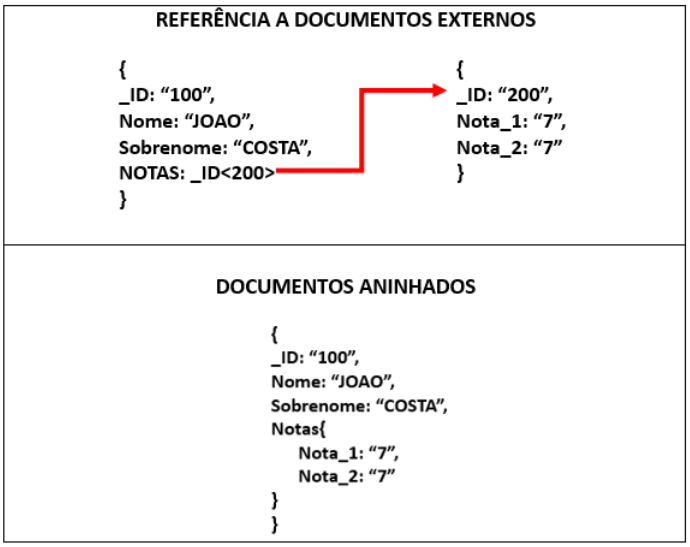
```
 {
    "_ID": "100",
    "pessoa"{
    "nome": "Joao",
    "sobrenome": "Costa",
    "notas"{
        "nota1" = "7",
        "nota2"= "7"
    }
}
```

As relações entre documentos podem ser representadas de duas maneiras: a primeira forma consiste em referenciar um documento externo, isto é, referenciar um outro documento, de forma equivalente a uma chave estrangeira em sistemas relacionais. A segunda, é aninhar documentos dentro de documentos, acarretando em documentos de maiores tamanhos e, também, a possibilidade de redundância de dados. As duas formas são exemplificadas na figura Figura 7.

Estes sistemas possuem suporte a função de *MapReduce*, que terá o seu conceito brevemente apresentado e exemplificado na subseção 2.3.2.1. Na Figura 8, um exemplo de uma base de dados em sistemas orientados a documentos. Como exemplo destes bancos de dados, podemos citar: *MongoDB*[22], *CouchDB*[23], *Couchbase*[24] e *RavenDB*[25].

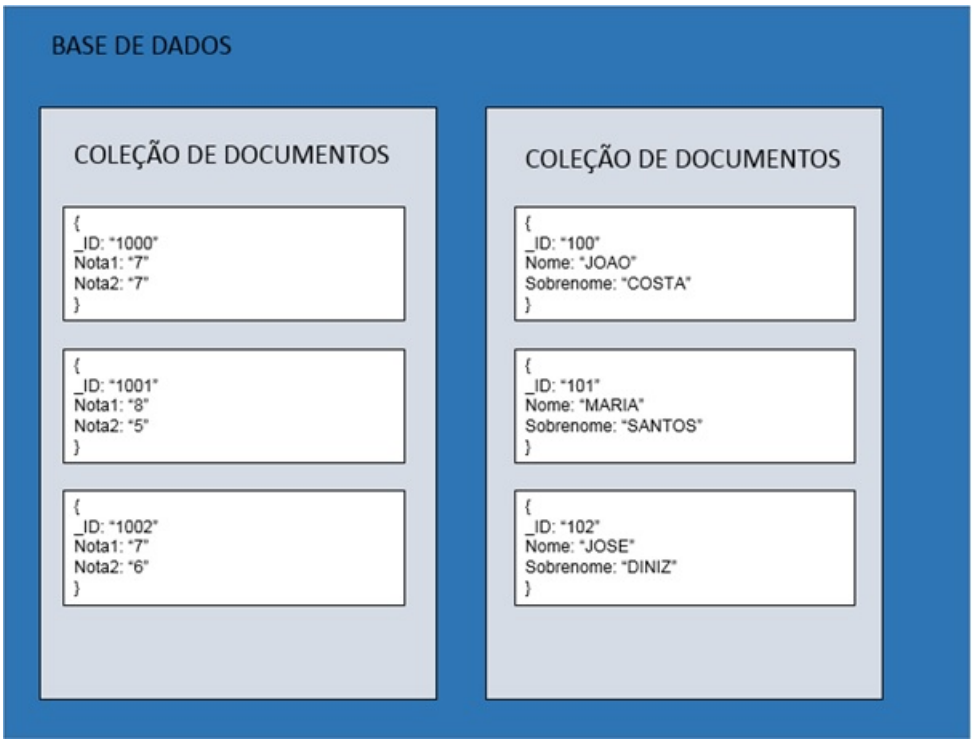
[22] https://www.mongodb.org/
[23] http://couchdb.apache.org/
[24] http://www.couchbase.com/
[25] https://ravendb.net/

Figura 7 – Representação de relação entre documentos através da referência externa e documentos aninhados.



Fonte: Elaborado pelo autor.

Figura 8 – Exemplo de uma base de dados em sistemas orientados a documentos.



Fonte: Elaborado pelo autor.

- Orientados a colunas: foi baseado no sistema *Google Bigtable*[26]. Diferentemente do modelo relacional, o modelo orientado a colunas não possui relacionamentos, portanto, cada coluna é exclusivamente independente em cada tabela. Neste modelo,

[26] https://cloud.google.com/bigtable/

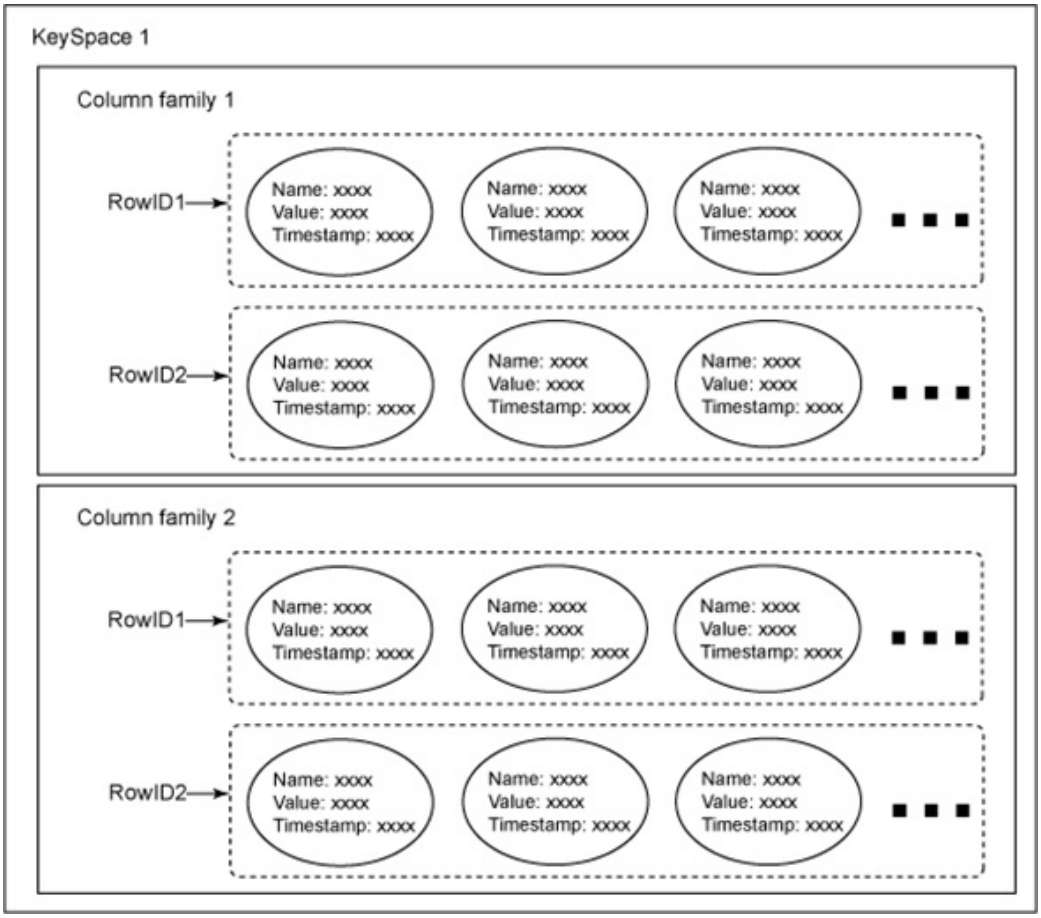
os dados do mesmo tipo são agrupados na mesma coluna, sendo representados em forma semelhante aos dos SGBDR, onde os dados são armazenados em linhas. Desta forma, o valor de cada coluna é armazenado em sequência, aumentando o desempenho de leituras de dados de uma única coluna, pois não carrega para a memória dados que não serão utilizados. Os dados deste modelo são indexados por linha, coluna e *timestamp* (informação do tempo ou data de um registro), onde linhas e colunas são identificadas por chaves e *timestamp*, que permite que um mesmo registro possua diversas versões. O modelo é composto por colunas, super colunas, famílias de colunas, superfamílias de Colunas e *KeySpaces*. Há chaves estáticas que apontam para múltiplas colunas. Podem ser criadas colunas em tempo de execução, sem ser preciso declarar nada, bastando atribuir um valor para a nova coluna com dada chave. As super colunas, ao invés de terem objetos como valores, possuem outras colunas. Famílias de colunas são como tabelas do modelo relacional, mas que, diferentemente das colunas, precisa ser declarada anteriormente em arquivo de configuração. Superfamília de colunas que possui apenas super colunas, que são agrupadas em *keyspaces*, que podem ser comparados aos *schemas* no modelo relacional. Na Figura 9, um exemplo de uma base de dados em sistemas orientados a colunas.

Como exemplo destes bancos de dados, podemos citar: *Google Bigtable*, *Apache Cassandra*[27], *Hbase*[28] e *Hypertable*[29].

- Orientados a grafos: neste modelo, os dados são representados como um grafo. É mais utilizado quando se é necessário representar uma relação entre dados, como, por exemplo, em redes sociais, em detecções de fraudes e em controle de gestão de dados (http://neo4j.com/use-cases/). O modelo pode ser facilmente representado da seguinte forma: os nós ou vértices representam os objetos dos dados e as arestas ou arcos representando os relacionamentos entre estes nós. Os nós e as arestas possuem chave de identificação e conteúdo. A chave de um vértice descreve o conteúdo do próprio vértice e o conteúdo de um vértice possui dados sobre o vértice. A chave de uma aresta descreve o conteúdo de um relacionamento entre vértices e o conteúdo de uma aresta possui dados referentes a um relacionamento entre dois vértices. Pode ser facilmente distribuído em diversos servidores. Como exemplo destes bancos de dados, podemos citar: *Neo4j*[30], *AllegroGraph*[31], *InfiniteGraph*[32] e *FlockDB*[33]. Na Figura 10, um exemplo de uma base de dados em sistemas orientados a grafos.

[27] http://cassandra.apache.org/
[28] https://hbase.apache.org/
[29] http://www.hypertable.com/
[30] http://neo4j.com/
[31] http://franz.com/agraph/allegrograph/
[32] http://www.objectivity.com/products/infinitegraph/
[33] https://github.com/twitter/flockdb

Figura 9 – Exemplo de uma base de dados em sistemas orientados a colunas



Fonte: https://www.ibm.com/developerworks/br/library/os-apache-cassandra/(Adaptado)

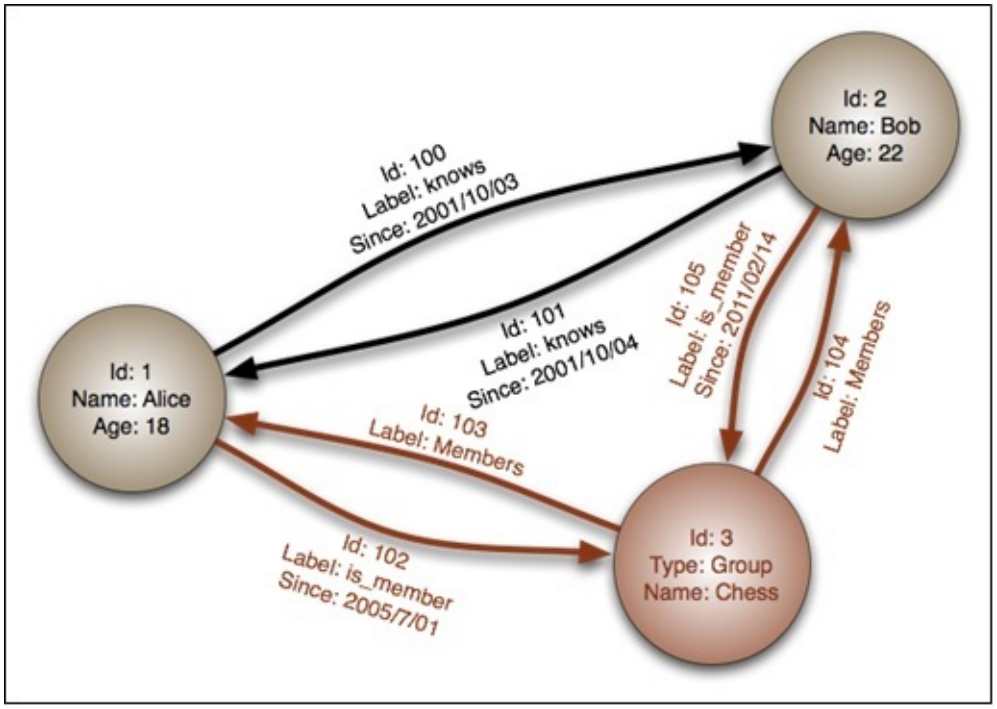
2.3.2.1 MapReduce

O *MapReduce* é um algoritmo criado pelo Google destinado ao processamento de grandes volumes de dados em forma distribuída (DEAN; GHEMAWAT, 2008). Foi desenvolvido segundo o conceito de dividir e conquistar, isto é, as tarefas são divididas em tarefas menores e independentes, sendo processadas de forma paralela (SCHNEIDER, 2012), através do mapeamento dos dados para os diversos nós que compõem o sistema para o seu processamento (VIEIRA et al., 2012). Devido à paralelização dos processos e operações, cada nó do *cluster* fica responsável por processar um pequeno volume de dados ao invés de processar um grande volume de dados em determinada etapa do processamento, conforme a Figura 11.

O algoritmo *Mapreduce* resume-se em duas operações, que são apresentados abaixo:

- *Map*: a função da operação *Map* é receber os dados, dividi-los em pequenos grupos para que possam ser distribuídos entre os nós do sistema. Cada nó recebe um conjunto intermediário de pares de chave e valor, que servem de entrada para a função *Reduce*.

Figura 10 – Exemplo de uma base de dados em sistemas orientados a grafos.



Fonte: https://en.wikipedia.org/wiki/Graph_database(Adaptado)

- *Reduce*:A função da operação *Reduce* é resgatar os conjuntos de dados separados pelo respectivo conjunto de chave/valor gerados na operação *Map*, dividi-los entre os nós do sistema e efetuar o processamento de cada grupo em seu respectivo nó e, após o processamento destes dados, realizar a junção destes grupos, gerando um novo conjunto de chave/valor, devolvendo-o para a aplicação solicitante.

Um exemplo do processo do algoritmo em uma base de dados é representado na Figura 12, onde são apresentados os dados antes da entrada do algoritmo *MapReduce*, após a saída da função *Map* e saída da função *Reduce* com os resultados solicitados.

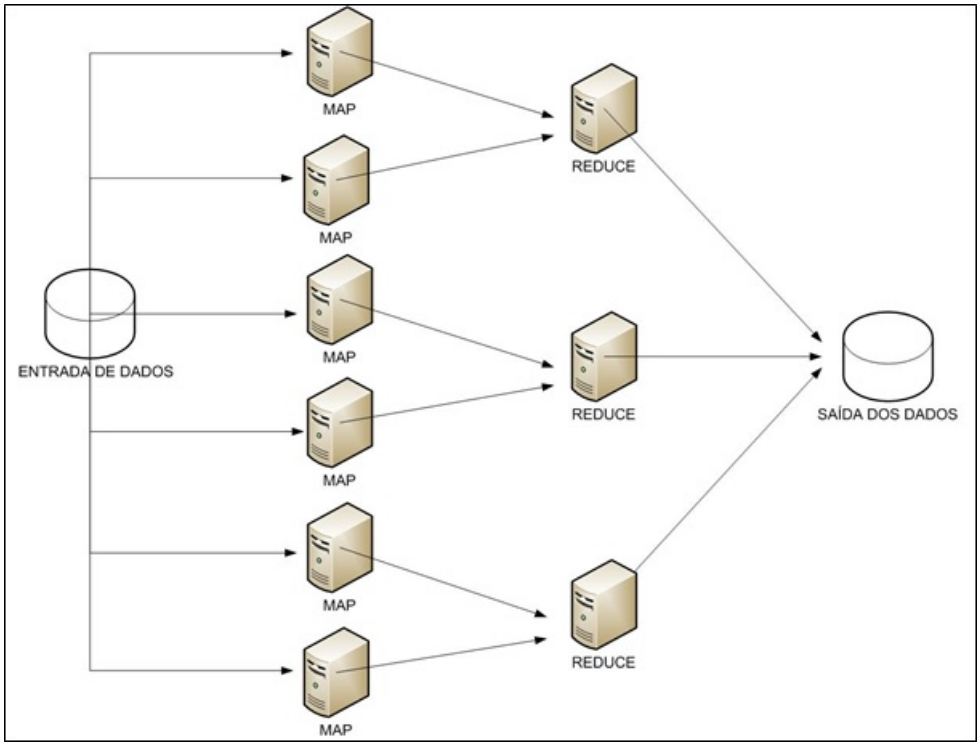
O *framework Apache Hadoop*[34], desenvolvido pela *Apache Software Foundation*[35], é uma implementação de código livre do *MapReduce* e tem obtido significante notoriedade nos últimos anos (WHITE, 2012).

2.4 *Benchmarks*

Devido ao aumento do fluxo, volume e estrutura dos dados, os SGBD vêm sofrendo com uma maior complexidade ao gerenciar estes dados. Dada a importância para uma organização de se ter uma base de dados coerente e um sistema de gerenciamento confiável e com desempenho relativamente bom, se faz necessária a comparação de sistemas de

[34] http://hadoop.apache.org/
[35] http://www.apache.org/

Figura 11 – Representação do esquema de *MapReduce*.



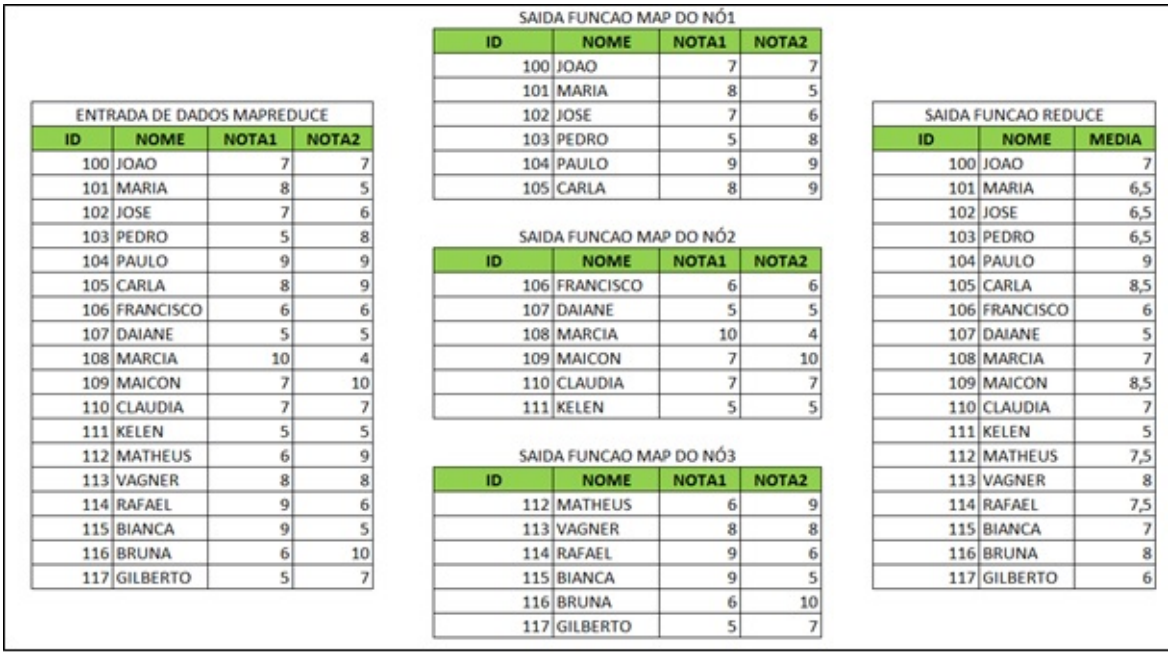
Fonte: Elaborado pelo autor.

gerenciamento, em vista de buscar-se vantagens e desvantagens e melhor caso de uso de cada sistema. Uma das formas mais práticas de se obter uma comparação de desempenho entre sistemas é através do uso de ferramentas específicas de teste de desempenho, conhecidas como *benchmark*. Basicamente, um *benchmark* é uma ferramenta própria para realização de testes, definidos através de métricas, critérios e padrões dedicados às operações ou características de interesse dos testes, visando, assim, mensurar o desempenho e avaliar um sistema ou dispositivo de *hardware*. Devido a sua importância para as organizações, auxiliar na identificação do desempenho e características de sistemas, alguns *benchmarks* destinados aos SGBD foram desenvolvidos e são brevemente apresentados a seguir.

A mais famosa ferramenta de testes para bancos de dados reconhecida nas literaturas é a família de testes *TPC*[36], acrônimo, em língua inglesa, de *Transaction Processing Performance Council*. Esta ferramenta surgiu em 1988, com o objetivo de definir *benchmarks* para avaliar o desempenho de transações em banco de dados (NAMBIAR; POESS, 2015). O teste *TPC-C* possui foco em emular operações em banco de dados para comparar diferentes características e foram construídos para o contexto de transações do mundo real (TPC-C, 2015). O *YCSB* (acrônimo, em língua inglesa, de *Yahoo! Cloud Serving Benchmark*, foi outra ferramenta de *benchmark* que ganhou relativa notoriedade nos últimos anos (PITCHUMANI; FRANK; MILLER, 2015). O *YCSB* foi desenvolvido para

[36] http://www.tpc.org/default.asp

Figura 12 – Representação de dados antes, durante e após a função *MapReduce*.



ENTRADA DE DADOS MAPREDUCE

ID	NOME	NOTA1	NOTA2
100	JOAO	7	7
101	MARIA	8	5
102	JOSE	7	6
103	PEDRO	5	8
104	PAULO	9	9
105	CARLA	8	9
106	FRANCISCO	6	6
107	DAIANE	5	5
108	MARCIA	10	4
109	MAICON	7	10
110	CLAUDIA	7	7
111	KELEN	5	5
112	MATHEUS	6	9
113	VAGNER	8	8
114	RAFAEL	9	6
115	BIANCA	9	5
116	BRUNA	6	10
117	GILBERTO	5	7

SAIDA FUNCAO MAP DO NÓ1

ID	NOME	NOTA1	NOTA2
100	JOAO	7	7
101	MARIA	8	5
102	JOSE	7	6
103	PEDRO	5	8
104	PAULO	9	9
105	CARLA	8	9

SAIDA FUNCAO MAP DO NÓ2

ID	NOME	NOTA1	NOTA2
106	FRANCISCO	6	6
107	DAIANE	5	5
108	MARCIA	10	4
109	MAICON	7	10
110	CLAUDIA	7	7
111	KELEN	5	5

SAIDA FUNCAO MAP DO NÓ3

ID	NOME	NOTA1	NOTA2
112	MATHEUS	6	9
113	VAGNER	8	8
114	RAFAEL	9	6
115	BIANCA	9	5
116	BRUNA	6	10
117	GILBERTO	5	7

SAIDA FUNCAO REDUCE

ID	NOME	MEDIA
100	JOAO	7
101	MARIA	6,5
102	JOSE	6,5
103	PEDRO	6,5
104	PAULO	9
105	CARLA	8,5
106	FRANCISCO	6
107	DAIANE	5
108	MARCIA	7
109	MAICON	8,5
110	CLAUDIA	7
111	KELEN	5
112	MATHEUS	7,5
113	VAGNER	8
114	RAFAEL	7,5
115	BIANCA	7
116	BRUNA	8
117	GILBERTO	6

Fonte: Elaborado pelo autor.

avaliar o desempenho das ferramentas *NoSQL* em ambientes distribuídos (COOPER et al., 2010). Diferentemente do *TPC-C*, que visa simular aplicações do mundo real, o *YCSB* visa avaliar o desempenho de sistemas, em diferentes aspectos, através da realização de testes de estresse de simples operações *CRUD*. Existem alguns outros *benchmarks*, tais como o *TPC-H*(http://www.tpc.org/tpch/default.asp), também da família *TPC*, o *SSB* (acrônimo, em língua inglesa, de *Star Schema Benchmark*) (O'NEIL; O'NEIL; CHEN, 2007) e as ferramentas *YCSB++*(PATIL et al., 2011) e *YCSB+T* (DEY et al., 2014), que foram derivadas do *YCSB* e também visam avaliar o desempenho de sistemas distribuídos.

Para avaliar o desempenho de SGBD, as métricas abaixo são as mais comumente utilizadas nos testes:

- Recursos de *hardware* do sistema: consume de memória e processador e espaço em disco.
- Latência: o tempo médio que corresponde a requisição e a resposta da operação.
- Escalabilidade: número máximo de acessos simultâneos a base de dados.
- Vazão: número médio de operações realizadas em determinado período de tempo.

De acordo com Lourenço et al. (2015), sabe-se que a grande parte dos trabalhos de pesquisa estão focados em avaliar o desempenho de sistemas, mas é importante salientar que a arquitetura dos sistemas do mundo real não é apenas dirigida por requerimentos

de desempenho, tendo-se que se incluir de forma abrangente muitos outros requisitos de atributos de qualidade.

2.5 Trabalhos relacionados

Aos pesquisar-se sobre trabalhos relacionados, foram identificados alguns artigos e pesquisas que abordam alguns assuntos de forma semelhante a este trabalho.

Em comparações de bancos relacionais e bancos *NoSQL*, Cooper et al. (2010) apresentou a ferramenta de *benchmark YCSB*, acrônimo das iniciais de *Yahoo! Cloud Serving Benchmark*, destinada a realizar a avaliação do desempenho de sistemas de gerenciamento de dados. Neste mesmo trabalho, o mesmo realizou a comparação dos bancos *NoSQL Cassandra*, *Hbase* e *Yahoo!'s PNUTS* e do banco relacional *MySQL* através desta ferramenta. Essa comparação possibilitou a avaliação de desempenho e da capacidade de escalonamento destes sistemas.

(ABRAMOVA; BERNARDINO, 2013) teve como objetivo testar o desempenho dos sistemas *NoSQL MongoDB*, em sua versão 2.4.3, e *Cassandra*, em sua versão 1.2.4. Foram descritas as principais características e vantagens dos sistemas *NoSQL* em relação aos sistemas relacionais. Também, foi realizado um *benchmark* envolvendo os sistemas *MongoDB* e *Cassandra*, que testou o gerenciamento e a escalabilidade de grandes volumes de dados e verificar como estes sistemas se comportam em operações de leitura e e atualizações de dados em ambientes com poucos recursos de *hardware*, tais como memória e processador, de forma similar a um computador pessoal ou servidores de menor capacidade. Para a avaliação destes sistemas, foi utilizada a ferramenta de *benchmark YCSB*. O foco de avaliação de desempenho considerado nos testes foi o tempo de execução dos sistemas na carga dos dados e na execução dos *workloads* A,B, C, F, G e H (estes dois últimos criados pelos autores do trabalho). Nos testes, foi possível observar que o sistema *MongoDB* reduzia o seu desempenho enquanto o tamanho dos dados era aumentado e o sistema *Cassandra* aumentava o seu desempenho enquanto o tamanho dos dados aumentava. Já nas operações de atualizações de dados, o desempenho do sistema *Cassandra* era ainda mais superior, o que se resumiu em quase todos os testes do trabalho.

(ABRAMOVA; BERNARDINO; FURTADO, 2014b) realizou a comparação de cinco sistemas *NoSQL* (*Cassandra*, *HBase*, *MongoDB*, *OrientDB* e *Redis*) através da realização de operações de leituras e atualizações de dados com a ferramenta de *benchmark YCSB*. O objetivo do trabalho foi contribuir para que fosse possível para o usuário escolher o sistema mais apropriado de acordo com mecanismos específicos e necessidades. Nos testes, foram utilizados os *workloads* A, C e H (este último criado pelos autores), utilizando de bases de dados de 600.000 documentos compostas por 10 campos. Os resultados obtidos apresentaram que o sistema *Redis* teve o melhor desempenho. Os sistema *Cassandra* e *HBase* apresentaram bons desempenhos em operações de atualização de dados. Os sistemas

MongoDB e *OrientDB* apresentaram os piores desempenhos dos testes.

Com relação à comparação exclusivas entre bancos *NoSQL*, Abramova, Bernardino e Furtado (2014a) comparou dez sistemas (*Cassandra*, *Hbase*, *Oracle NoSQL*, *Redis*, *Scalaris*, *Tarantool*, *Voldemort*, *Elasticsearch*, e *OrientDB*), através da ferramenta *YCSB* e utilizando de um conjunto de operações, para compreender como cada tipo de sistema atende diferentes operações e como o seu desempenho é afetado com estas operações. Nos testes foram utilizados os *workloads* A, B, C, F, G e H (os dois últimos criados pelos autores do trabalho). Os resultados obtidos apresentaram que os sistemas com armazenamento de dados em memória (*Tarantool* e *Redis*) eram os com melhor desempenho. Já, apresentando o pior desempenho, o sistema *Oracle NoSQL*, na carga dos dados, e o sistema *OrientDB* na execução dos *workloads*. Os sistemas *HBase* e *Cassandra* apresentaram bons resultados nas operações de atualização de dados. O sistema *MongoDB* apresentou o maior aumento no tempo de execução, onde este aumentava de forma diretamente proporcional ao aumento das operações de atualizações de dados, em virtude dos bloqueios realizados neste tipo de operações por este sistema como forma de controle de concorrência. Em resumo, o trabalho dividiu os sistema em dois grupos: o primeiro, formado pelos sistemas *MongoDB*, *Redis*, *Scalaris*, *Tarantool* e *OrientDB*, são sistemas otimizados para operações de leituras, enquanto o segundo grupo, formado pelos sistemas *Cassandra*, *Voldemort*, *Oracle NosQL* e *HBase* são sistemas que apresentam melhor desempenho em operações de atualização de dados.

Truica et al. (2015) teve como objetivo testar e comparar a replicação assíncrona de três sistemas *NoSQL* orientados a documentos (*MongoDB*, *CouchDB* e *Couchbase*). Os testes foram designados para comparar o tempo de execução dos sistemas através de operações *CRUD*, quando trabalhando com grandes volumes de dados. Para a comparação do desempenho, os teste foram, inicialmente, executados em uma única instância de cada sistema, posteriormente, sendo executados em ambiente distribuído. O ambiente distribuído utiliza a arquitetura mestre-escravo composta por nove máquinas virtuais, três para cada sistema, compostas pelo mesmo *hardware* e também o mesmo modelo e representação de dados. O tempo de execução também foi comparado com os resultados obtidos com três sistemas relacionais (TRUICA; BOICEA; RADULESCU, 2013). Os testes foram realizados usando, nesta ordem, inserções, atualizações, leituras e exclusão de dados, sendo executados 50 vezes em bases de dados compostas por 1.000, 10.000 e 100.000 registros.

3 TECNOLOGIAS ENVOLVIDAS

Este capítulo tem como objetivo apresentar as tecnologias que serão utilizadas neste trabalho. Inicialmente, na seção 3.1, será apresentado o sistema operacional que servirá como base para os testes. Em seguida, na seção 3.2 serão apresentadas as ferramentas *NoSQL* selecionadas para o trabalho. E, por último, na seção 3.3, será apresentada a ferramenta de *benchmark* que será utilizada para a geração da base de dados e execução dos testes de desempenho. Na seção 3.4, será apresentada a ferramenta de monitoramento dos recursos de *hardware* do sistema operacional. Na seção 3.5, são apresentados os sistemas adicionais necessários para o correto funcionamento da ferramenta de *benchmark*.

3.1 Sistema operacional

Durante a realização dos testes, será utilizado o sistema operacional *Linux Ubuntu Desktop* 12.04.5[1], na versão de 64 *bits*. Destaca-se que o mesmo apresenta-se com versão estável e é baseado no *kernel* 3.13. Deu-se preferência para esta distribuição, primeiramente, devido a mesma já ter sido utilizados em testes anteriores (ABRAMOVA; BERNARDINO, 2013; ABRAMOVA; BERNARDINO; FURTADO, 2014a; ABRAMOVA; BERNARDINO; FURTADO, 2014b) e, apesar destes testes terem sido realizados na versão *Ubuntu Server*, conforme a própria página de ajuda do Ubuntu (UBUNTU, 2012), a diferença entre as duas versões do sistema se resume a falta de pacotes disponíveis na instalação, como por exemplo, pacotes considerados como para computadores domésticos como os pacotes de interfaces gráficas (*X*, *KDE*, *Gnome* e *Unity*) que não são disponibilizados na versão *Server*, e, do lado contrário, pacotes considerados como sendo para servidores (Apache2[2], Bind9[3], entre outros), não estando presentes, por padrão, no instalador da versão *Desktop*. Em resumo, ainda de acordo com a página, executando a instalação mínima da versão *desktop* deste sistema, pode-se obter o mesmo resultado de se fazer a instalação de servidor do sistema. Outro motivo por ter optado por uma distribuição *Linux*, foi devido a estes sistemas serem gratuitos, de licença livre e possuírem um desempenho superior ao *Windows* no uso de sistemas *NoSQL*. O sistema operacional será instalado com o padrão de fábrica, não sendo realizada nenhuma alteração de configuração durante a instalação. Apenas os sistemas de bancos de dados *NoSQL* e a ferramenta de avaliação que participará dos testes serão adicionadas ao sistema operacional, visando minimizar o consumo de memória e recursos de *hardware* do servidor e reduzir a possibilidade de haver alguma interferência por parte de outro programa nos resultados dos testes. Assim, somente os processos nativos

[1] http://releases.ubuntu.com/12.04/
[2] https://httpd.apache.org/
[3] https://www.isc.org/downloads/bind/

do sistema operacional, juntamente aos processos referentes às ferramentas anteriormente citadas, estarão em execução durante a realização dos testes.

3.2 SGBD NoSQL

Durante a realização dos testes, serão utilizados os SGBDs *NoSQL MongoDB* 2.4.6[4], *Apache CouchDB* 1.6.1[5], o *Couchbase Server* 3.0.1 *Community Edition*[6] e o *OrientDB* 2.1.3[7]. Destaca-se que os mesmos apresentam-se em uma versão estável e que se deu preferência para sistemas de código aberto e que fossem gratuitos. Os *downloads* dos sistemas serão realizados de seus repositórios oficiais e as suas instalações serão realizadas de forma padrão, isto é, sem a alteração de quaisquer características ou propriedade durante este processo. Nas subseção 3.2.1, subseção 3.2.2, subseção 3.2.3 e subseção 3.2.4, serão, respectivamente, apresentados os sistemas que, a princípio, farão parte dos testes. Destaca-se, também, que apenas se fará necessária a alteração de algum destes sistemas caso os mesmos não atendam a algum requisito ou não sejam compatíveis com o sistema operacional ou ferramenta de avaliação. Na subseção 3.2.5, é apresentada uma tabela com a comparação das características dos sistemas *NoSQL* selecionados para compor o trabalho.

3.2.1 MongoDB

O SGBD *NoSQL MongoDB* é um sistema de gerenciamento de bancos de dados orientados a documentos, de esquema flexível e de código livre, desenvolvido na linguagem C++, que tem como características a velocidade, a eficiência e a escalabilidade. Atualmente, é um dos bancos *NoSQL* mais utilizados. Usa o padrão *JSON* para a manipulação dos dados, mas armazenando-os no banco com o formato *BSON*. O *MongoDB* é multiplataforma (o código fonte pode ser baixado para as plataformas *Linux*, *Windows*, *OS X* e *Solaris*). O escalonamento horizontal é garantido através de *sharding*, possibilitando a paralelização das operações sobre os dados do banco. A replicação de dados é garantida através de bloqueios e pelo modelo mestre-escravo assíncrono. Ao criar-se um novo documento, se o "_ID" do documento, que é um identificador único dentro do banco para cada documento, não for informado pelo usuário, o *MongoDB* gerará o identificador automaticamente. A indexação é realizada através do uso de *B-trees*. Possui uma implementação própria de *MapReduce* que auxilia a processar grandes volumes de dados. Para o acesso e monitoramento das informações do banco, utiliza de um protocolo *JSON* proprietário chamado de *MongoDB shell*.

[4] https://www.mongodb.org/downloads
[5] http://couchdb.apache.org/
[6] http://www.couchbase.com/nosql-databases/downloads
[7] http://orientdb.com/download-previous/

3.2.2 CouchDB

O *CouchDB* é um sistemas de gerenciamento de dados *NoSQL* orientado a documentos, de esquema flexível e de código livre, desenvolvida na linguagem de programação Erlang. Os documentos são escritos no formato *JSON*. No *CouchDB*, as consultas são chamadas de *views*, e as consultas de documentos são realizadas através da implementação de uma *MapReduce view* escrita em *JavaScript*. Usa HTTP RESTful (GET, PUT, POST, DELETE). O *MapReduce* é utilizado para pré processar dados para a *view*. Cada documento no *CouchDB* possui um identificador único chamado "_design", que se não for informado pelo usuário, será gerado um "_design"automaticamente pelo sistema para cada versão do dado. Usa o *MVCC* (acrônimo, em língua inglesa, de *Multi-version Concurrency Control*, que, em língua portuguesa, significa controle de concorrência multiversão, também conhecido como bloqueio otimista. Este controle faz com que operações de leituras ou escritas nunca acarretem em bloqueios no banco. Utiliza a indexação *B+Tree* para armazenamento de dados (http://guide.couchdb.org/draft/btree.html), que são atualizados durante as modificações de dados e, por isso, precisa haver periódica compressão dos dados, onde tal operação pode prejudicar o desempenho do sistema. Este sistema suporta tanto a replicação mestre-escravo, quanto a multimestre. O escalonamento é garantido através da replicação assíncrona ou incremental dos dados, isto é, apenas os documentos inseridos ou atualizados que não fizeram parte da última replicação é que serão replicados, mas não suporta o *sharding* de forma nativa.

3.2.3 Couchbase

O SGBD *NoSQL* Couchbase é um sistema de gerenciamento de bancos de dados que derivou da combinação do *CouchDB* e do *Membase* (um SGBD chave/valor com compatibilidade com *Memcached*[8]. Foi desenvolvido na linguagem C, C++, Go e Erlang. Pode ser utilizado como um sistema chave/valor, mas é considerado como sendo um sistema orientado a documentos, que armazena os dados no formato *JSON*. Os dados podem ser armazenados, tanto em memória, quanto em disco, em forma de *data buckets*, que são equivalentes a uma base de dados. Há dois tipos de *buckets* no *CouchBase*: o *Couchbase bucket* e o *Memcached bucket*. Da mesma forma que no *CouchDB*, as consultas são baseadas e construídas usando *MapReduce* em *JavaScript*. Possui bloqueio otimista e a indexação dos dados é realizada através de *B-tree*. A consistência padrão é a eventual. A maior diferença em relação ao *CouchDB* é em relação ao *sharding*, pois no *Couchbase* esta opção está disponível de forma nativa. Outro ponto de diferença é em relação a replicação de dados, onde o *Couchbase* suporta replicação mestre-escravo e multimestre.

[8] https://memcached.org/

3.2.4 OrientDB

O SGBD *NoSQL OrientDB* é um sistema de gerenciamento de bancos de dados considerado multimodelo, devido ao mesmo ser naturalmente orientado a grafos, mas realizar o armazenamento dos dados em forma de documentos *JSON*. Este sistema foi desenvolvido na linguagem *Java* e, por esse motivo, o mesmo é multiplataforma (*Windows*, *Linux*, *OS X* ou qualquer outro sistema com suporte a *JVM*, que será apresentado na seção 3.5). Este sistema suporta múltiplos tipos de dados, tais como chave/valor, orientado a grafos, orientados a documentos e orientados a objetos, mas o relacionamentos entre os dados são representados como em sistemas orientados a grafos com conexões diretas entre os dados. Para a indexação dos dados, utiliza de um novo algoritmo chamado *MVRB-tree*, derivado dos algoritmos *red-black tree* e do *B+tree*. Suporta a replicação do tipo multi-mestre. Apresenta suporte a transações ACID, garantindo que todas as operações no banco sejam processadas com segurança e, em caso de algum evento de falha, todos os documentos pendentes serão restaurados e salvos no banco.

3.2.5 Comparação das ferramentas *NoSQL* selecionadas

Os sistemas *NoSQL* selecionados para realização dos testes, apesar de pertencerem ao grupo de sistemas orientados a documentos, apresentam diferentes características entre si. Abaixo, na Tabela 1, estas características são resumidamente apresentadas:

Pode-se observar que, apesar de pertencerem ao mesmo grupo de sistemas, mais especificamente ao modelo orientados a documentos, estes sistemas possuem características bem distintas, tais como a aprestada em relação a *API* e métodos de acesso e o tipo de replicação.

3.3 Ferramentas de avaliação

O *YCSB*[9] (acrônimo, em língua inglesa, de *Yahoo! Cloud Service Benchmark*) é uma ferramenta de avaliação de sistemas distribuídos. Foi desenvolvido na linguagem Java, por um grupo de engenheiros do *Yahoo* (COOPER et al., 2010), e, em 2009, foi disponibilizada em forma de código livre (KASHYAP et al., 2013). Esta ferramenta avalia diferentes aspectos do sistema de banco de dados através da realização de testes de estresse de operações *CRUD* (FRIEDRICH et al., 2014). Segundo Abramova, Bernardino e Furtado (2014b), o *YCSB* possui alguns pacotes ou *workloads*, que são coleções de cargas de trabalho, compostos por alguns testes pré-definidos, conforme apresentado na Tabela 2. Cooper et al. (2010) diz que esta ferramenta também possibilita a criação de novos pacotes com seus respectivos grupos de cargas de trabalho ou, se necessário, possibilita que um código na linguagem Java seja escrito. Ela também gera a base de dados e as operações de

[9] https://github.com/brianfrankcooper/YCSB

Tabela 1 – Comparação entre as características dos sistemas *NoSQL* selecionados para compor o trabalho.

	MongoDB	*Couchbase*	*OrientDB*
Modelo de Dados	orientado a documentos	orientado a documentos	multimodelo (orientado a documentos, orientado a grafos, chave/valor e orientado a objetos)
Data de Lançamento	2009	2011	2010
Linguagem de Implementação do Sistema	C++	C, C++, Go e Erlang	Java
Escalonamento	Horizontal	Horizontal	Horizontal
Particionamento de Dados	*Sharding*	*Sharding*	*Sharding*
Controle de Concorrência	Bloqueio	MVCC	MVCC
Sistemas Operacionais Suportados	Linux, OS X, Windows e Solaris	Linux, OS X e Windows	Qualquer Sistema Operacional com suporte a JVM
Modelo de Dados	Livre de esquema	Livre de esquema	Livre de esquema
Indexação	B-tree	B-tree	MVRB-tree
API e Métodos de Acesso	Protocolo *Json* proprietário (*MongoDB Shell*)	RESTful HTTP API	Java API e RESTful HTTP/JSON API
Replicação	Mestre-escravo	Multimestre e Mestre-escravo	Multimestre
Suporte a *MapReduce*	Sim	Sim	Não

Fonte: Elaborado pelo autor.

carga de trabalho, também conhecido como sendo o conjunto de operações a ser realizado. A carga de trabalho define os dados que serão carregados para o banco de dados durante a fase de carga, e as operações que serão executadas durante a fase de transações.

Esta ferramenta possui suporte para diferentes sistemas, de diferentes tipos (chave/-valor, documentos, colunares e relacionais), onde a lista completa dos sistemas suportados pode ser encontrada em https://github.com/brianfrankcooper/YCSB. Além disso, possibilita que as operações sejam executadas concorrentemente através de *threads*, simulando vários clientes trabalhando de forma concorrente, onde cada uma destas pode executar as operações das cargas de trabalhos definidas. Através das *threads* é possível mensurar a latência e a quantidade de operações por segundo alcançada(KASHYAP et al., 2013). A escalabilidade e a elasticidade podem ser quantificadas mensurando o aumento de desempenho quando novos nós são adicionados ao sistema. Após a realização dos testes, o módulo estatístico agrega os resultados, possibilitando a criação dos gráficos (KASHYAP et al., 2013). Friedrich et al. (2014) diz que ainda não foram criadas as avaliações de disponibilidade, replicação e tolerância a falhas. Atualmente, é considerado a ferramenta de *benchmark* mais influente para sistemas *NoSQL*, ganhando muita notoriedade e sendo

utilizada em grande número de pesquisas (PITCHUMANI; FRANK; MILLER, 2015).

Rabl et al. (2012) diz que o sucesso do *YCSB* se deve ao motivo do mesmo já incluir um gerador de dados, um gerador de cargas de trabalho e possuir *drivers* para diversos sistemas. Para a realização dos testes de comparação e avaliação dos sistemas *NoSQL* e para mensurar o desempenho destes, será utilizada a ferramenta *YCSB*, acrônimo de *Yahoo! Cloud Serving Benchmark*, mais especificamente com a versão 0.8.0 desta ferramenta. Deu-se preferência por esta, devido ao seu ganho de notoriedade e quantidade de casos de estudo em diversas pesquisas (ABRAMOVA; BERNARDINO; FURTADO, 2014a; ABRAMOVA; BERNARDINO; FURTADO, 2014b; COOPER et al., 2010), e, também, em virtude das características anteriormente apresentadas por esta ferramenta.

Tabela 2 – Os seis *workloads* padrão da ferramenta *YCSB*, com a composição das transações, em porcentagem.

	Leituras	Atualizações	Escritas	Scans
Workload A	50	50	0	0
Workload B	95	5	0	0
Workload C	100	0	0	0
Workload D	95	0	5	0
Workload E	0	0	5	95
Workload F	50	Leitura-Modificação-Atualização		

Fonte: Elaborado pelo autor.

3.4 Ferramenta de monitoramento de consumo de recursos de hardware

Para o monitoramento do consumo dos recursos de *hardware* do sistema, foi selecionada a ferramenta *Glances*[10]. Tal ferramenta foi escrita na linguagem *Python* e é multiplataforma (permite ser executada em vários sistemas operacionais, tais como o *Linux*, *FreeBSD*, *OS X* e *Windows*), e permite que sejam monitorados os recursos de *hardware* do sistema, tais como processador, memória, interface de rede, disco rígido, além de mostrar a lista de processos sendo utilizados no sistema. Esta ferramenta pode ser visualizada, tanto através do terminal de comandos, quanto por um navegador de internet, conhecido como modo *web server mode*. Esta ferramenta foi escolhida aleatoriamente, podendo ter sido escolhido, também, as ferramentas *top*, que já vem pré instalada em muitas distribuições *Linux*, *htop*[11] e *atop*[12].

Para acessar esta ferramenta através do navegador de internet, primeiramente, deve-se, através do terminal de comandos, digitar o comando abaixo, posteriormente, no navegador de internet, acessar o endereço *IP* do servidor onde a ferramenta *Glances* encontra-se instalada, juntamente a porta 61208, onde, no exemplo do atual trabalho, o acesso seria realizado através do endereço http://192.168.0.19:61208.

[10] https://nicolargo.github.io/glances/
[11] http://hisham.hm/htop/
[12] http://www.atoptool.nl/

```
$ glances -w
```

3.5 Programas Adicionais

Nesta seção serão apresentados dois programas que deverão ser instalados para que a ferramenta de testes *YCSB* funcione corretamente. Estes programas são abaixo citados:

- *Java*: O *Java* é uma linguagem de programação e uma plataforma computacional lançada pela primeira vez pela *Sun Microsystems* em 1995, que, atualmente, pertence a *Oracle*[13]. A principal característica do *Java*, é que um código ou programa desenvolvido nessa linguagem de programação pode ser executado em qualquer sistema que possua suporte a *JVM* (acrônimo, em língua inglesa, de *Java Virtual Machine*), pois esta converte e compila o *bytecode* (código reconhecido e interpretado pela *JVM*) para código nativo de máquina.
- *Maven*: é uma ferramenta destinada ao gerenciamento de projetos de software sob o controle da *Apache Software Foundation*[14]. Com esta ferramenta é possível configurar um ambiente de desenvolvimento padronizado, gerenciando as dependências e o ciclo de vida do projeto.

[13] http://www.oracle.com/index.html
[14] http://www.apache.org/

4 PROPOSTA DE SOLUÇÃO

Como visto anteriormente, se faz necessária uma avaliação do desempenho de sistemas, seja por meio de métricas ou por ferramentas de *benchmark*, visando identificar pontos fortes de cada um, em comparação a outros sistemas. Algumas propriedades necessitam ser definidas para a execução dos testes. O capítulo está organizado da seguinte forma: a seção 4.1 apresenta a base de dados definida. A seção 4.2 apresenta as métricas de comparação e avaliação que serão utilizadas. A seção 4.3 apresenta o ambiente de *hardware* onde serão realizados os testes de desempenho.

4.1 Base de dados

Nos testes a serem realizados, serão geradas as bases de dados conforme a quantidade de documentos abaixo citadas:

- Quantidade de documentos com 10 campos: serão geradas bases de dados com 10.000, 50.000, 100.000 500.000 e 1.000.000 documentos, onde cada um destes será composto por 10 campos.
- Quantidade de documentos com 100 campos: serão geradas bases de dados com 10.000, 50.000 e 100.000 documentos, onde cada um destes será composto por 100 campos.

Os valores dos campos serão gerados de forma automática e aleatoriamente pela ferramenta de testes.

4.2 Métricas de comparação e avaliação

Para a avaliação das ferramentas, serão utilizadas as métricas de comparação baseadas em Abramova e Bernardino (2013). Deu-se preferência para estas métricas por considerar-se que estes autores contém avançado conhecimento em sistemas *NoSQL*, por já terem realizado diversas pesquisas sobre o assunto. Da mesma forma como foi realizado no trabalho anteriormente citado, foi criado também um novo *workload*, denominado *workload* G, seguindo-se os passos conforme apresentado na página de internet (UBUNTU, 2010), que será composto por transações com 95% de atualizações e 5% de leituras sobre a base de dados. Tal procedimento foi realizado efetuando a cópia do *workload*B, porém, alterando os parâmetros do novo documento para *readproportion*=0.05 e *updateproportion*=0.95, através da ferramenta de edição de textos vim[1], ou de qualquer outra ferramenta para edição de

[1] http://www.vim.org/

texto (o arquivo pode ser encontrado noApêndice E). Na Tabela 3, são apresentados os *workloads* selecionados para compor os testes do atual trabalho.

Tabela 3 – Os *workloads* da ferramenta *YCSB* que serão utilizados na realização dos testes com os sistemas selecionados.

	Leituras	Atualizações
Workload A	50	50
Workload B	95	5
Workload C	100	0
Workload F	50	Leitura-Modificação-Atualização
Workload G	5	95

Fonte: Elaborado pelo autor.

Abaixo, são apresentados as métricas de comparação levadas em consideração para a realização do trabalho:

- Workloads: serão utilizados os seguintes *workloads* nos testes:
  - *Workload*A load: carga dos dados ou 100% de inserções de dados;
  - *Workload*A run: 50%/50% de leitura/atualização;
  - *Workload*B run: 95%/5% de leitura/atualização;
  - *Workload*C run: 100% leitura;
  - *Workload*F run: 50%/50% de leitura/leitura-modificação-atualização;
  - *Workload*G run: 95%/5% de atualização/leitura;
- Tempo de execução: será considerado o tempo médio de execução dos *workloads*:
- Número de operações por segundo: será considerado o número médio de operações por segundo durante a execução dos *workloads*.
- Consumo de memória e processador;
- Espaço em disco utilizado;

Estas métricas foram definidas para serem utilizadas nos testes devido a algumas já fazerem parte dos pacotes de testes e serem disponibilizadas, por padrão, pela ferramenta *YCSB* e, também, devido a algumas serem citadas em diversos testes em trabalhos de pesquisa (ABRAMOVA; BERNARDINO, 2013; ABRAMOVA; BERNARDINO; FURTADO, 2014a; ABRAMOVA; BERNARDINO; FURTADO, 2014b). Dados estes pontos, definiu-se que tais métricas seriam suficientes para a realização do *benchmark* das ferramentas selecionadas. A avaliação das ferramentas consistirá no desempenho apresentado durante os testes. O método de avaliação entre os sistemas que serão considerados são: o tempo médio e total de execução dos testes , o tempo de resposta médio e o número médio de operações

por segundo durante os testes. Salienta-se que, primeiramente, os testes serão realizados com um único cliente, a fim de estabelecer um parecer inicial a respeito do desempenho dos sistemas. Posteriormente, serão realizados testes com clientes executando transações de forma concorrente. Estes testes consistirão de 100 e 250 *threads*. Para estabelecer a quantidade de operações para a execução das transações, será considerado o tamanho da base de dados como parâmetro, sendo escolhidos, aleatoriamente, os valores de 20% e 50% da quantidade de documentos da base de dados. Como exemplo, se a base de dados for constituída de 10.000 documentos, o número de operações a serem realizadas será 2.000 e 5.000, e, para uma base de dados constituída de 50.000 documentos, o número de operações a serem realizadas será 10.000 e 25.000.

4.3 Arquitetura de *hardware* do ambiente de teste

Os testes serão realizados em um ambiente multiprocessados, já preparadas para sistemas operacionais de 64 *bits*. Na Tabela 4, é descrita a configuração de *hardware* do ambiente de testes:

Tabela 4 – Configuração do ambiente de *hardware* dos testes.

	CONFIGURAÇÃO
PROCESSADOR	PHENOM II X4 SOCKET AM3
MEMÓRIA	14GB DDR3
ARMAZENAMENTO	HD 500GB 7200RPM
PLACA MÃE	Asus M4A785TD-M Evo

Fonte: Elaborado pelo autor.

5 EXECUÇÃO DOS TESTES E AVALIAÇÃO DAS FERRAMENTAS

O propósito da comparação das ferramentas neste trabalho é definir critérios de comparação e identificar o sistema com o melhor desempenho entre os demais, de acordo com uma necessidade específica. Durante a realização dos testes, houve a necessidade de modificar algumas definições estabelecidas na proposta de solução, e essas modificações serão apresentadas na seção 5.1. Na seção 5.2, serão realizados os testes dos sistemas *NoSQL* selecionados através do uso da ferramenta *YCSB*. Na seção 5.3, os resultados obtidos nos testes serão comparados para a posterior análise. Na seção 5.4, os sistemas serão devidamente avaliados, em busca do que melhor se adapta para o problema proposto. Salienta-se que todos os comandos apresentados neste capítulo são comandos destinados e específicos para a distribuição *Ubuntu*, podendo, também, funcionar em sistemas derivados da distribuição *Debian*[1].

5.1 Alterações da Proposta de Solução

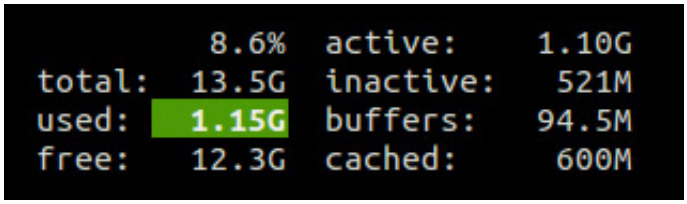
Foram realizados alguns ajustes na proposta de solução definida para o melhor aproveitamento do trabalho. Primeiramente, houve a necessidade de trocar-se o sistema *NoSQL CouchDB* pelo sistema *NoSQL OrientDB*, mesmo este não sendo puramente orientado a documentos, conforme visto no Capítulo 4, devido ao sistema *CouchDB* não ter suporte, por padrão, com a versão 0.8.0 da ferramenta *YCSB*, apesar de já ter tido seu desempenho avaliado em (ARNAUDSJS, 2015), especificamente com a versão 0.1.4 da ferramenta *YCSB*.

Outra modificação é que deixarão de ser considerados no trabalho os consumos de memória e processador. A primeira destas modificações se deve ao fato do sistema *Couchbase* manter os dados em memória após os mesmos terem sido utilizados em transações (leituras, atualizações e inserções de dados). A cada transação, o consumo de memória aumenta gradativamente, conforme observado na Figura 13, onde é apresentado o consumo de memória do sistema operacional antes da realização dos testes no sistema *Couchbase*.

Na Figura 14, é possível observar que o consumo de memória aumentou gradativamente, devido ao sistema *Couchbase* manter os dados mais recentemente utilizados na memória e salvar estes de forma assíncrona no disco (COUCHBASE, 2015). Apesar deste mecanismo ser um diferencial deste sistema em busca de um melhor de desempenho, mantendo estes dados em memória para trabalhar de forma semelhante a uma memória *cache* e reduzir o tempo de acesso aos dados, tal característica inviabiliza o critério de utilizar o consumo de memória dos sistemas por já atribuir esta vantagem de desempenho

[1] https://www.debian.org/

Figura 13 – Consumo de memória do sistema operacional antes da realização das transações no sistema *Couchbase*.

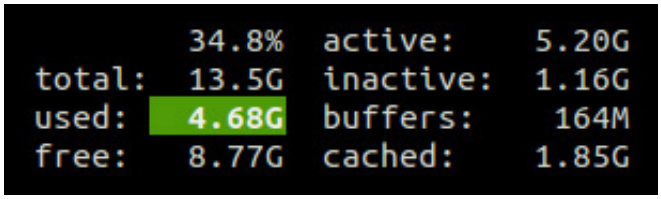


Fonte: *Print Screen* da ferramenta de monitoramento *Glances*.

apenas ao sistema *Couchbase*, que já apresenta este mecanismo de forma padrão.

Já o consumo de processador, deixará de ser utilizado devido a todos os sistemas apresentarem o mesmo consumo durante as execuções dos testes. O consumo normal de processador do sistema operacional, quando não há transações sendo executadas, varia de 6 a 10%, mas quando os testes são executados, indiferentemente do sistema utilizado, o consumo de processador para todos os sistemas varia de 35% a 99%, isto é, não é apresentada diferença no consumo de processador entre os sistemas *NoSQL* durante a execução dos testes.

Figura 14 – Consumo de memória do sistema operacional após a realização das transações no sistema *Couchbase*.



Fonte: *Print Screen* da ferramenta de monitoramento *Glances*.

A última modificação da proposta de solução será a não consideração da quantidade de operações por segundo como métrica de avalização, devido a entender-se que, quanto menor o tempo de execução, maior a quantidade de operações realizadas no tempo de um segundo e, no caso contrário, quanto maior o tempo de execução, menor a quantidade de operações também realizadas no tempo de um segundo.

5.2 Preparação do Ambiente de Testes

Nesta seção será realizada a preparação do ambiente de testes dos sistemas *NoSQL* selecionados através do uso da ferramenta de *benchmark YCSB*. Salienta-se que o processo de instalaçõ e configuração dos sistemas e ferramentas é apresentado no Apêndice A.

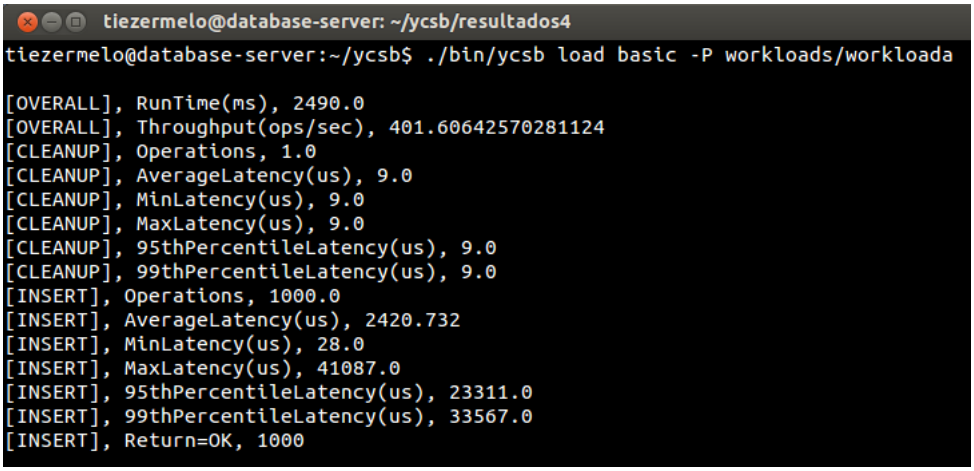
Serão apresentadas algumas operações que podem ser realizadas com a ferramenta de testes. O processo de realização dos testes divide-se em duas partes: a fase de carga ou carregamento dos dados, identificado através do comando "*load*", e que define os dados

a serem inseridos na base, e a segunda, a fase de execução dos *workloads* ou cargas de trabalho, identificado através do comando "*run*". Para verificar o funcionamento da ferramenta, pode-se executar os comandos abaixo:

```
$ cd ~
$ cd ycsb
$ ./bin/ycsb load basic -P workloads/workloada
$ ./bin/ycsb run basic -P workloads/workloada
```

No exemplo acima, com o primeiro comando acessa-se o diretório "*home*"do usuário. Com o segundo comando, acessa-se o diretório "ycsb". Com o terceiro comando será realizada a inserção dos dados na base, onde tal processo pode ser observado através do parâmetro "*load*". Já, com o quarto comando, será executado o *workload*A, observado através do parâmetro "*run*". Na Figura 15, é apresentado o resultado do comando de inserção ou carga dos dados na base.

Figura 15 – Saída da execução do comando *load* na ferramenta *YCSB* com o banco modelo *basic*.



Fonte: *Print Screen* da saída da execução do comando *load* no *YCSB* com o modelo *basic*.

Já, na Figura 16, é apresentado o resultado do comando do execução do *workload* A.

A ferramenta *YCSB* também permite que sejam adicionados novos parâmetros aos comandos de execução dos testes. Alguns desses parâmetros são apresentado abaixo:

- *Fieldcount*: o número de campos de um documento, sendo o padrão 10 campos;
- *Fieldlength*: o tamanho, em *bytes*, de cada documento, sendo, o padrão, 100 *bytes*;
- *Readproportion*: a proporção de operações de leitura a ser realizada sobre os dados;
- *Updateproportion*: a proporção de operações de atualização a ser realizada sobre os dados;

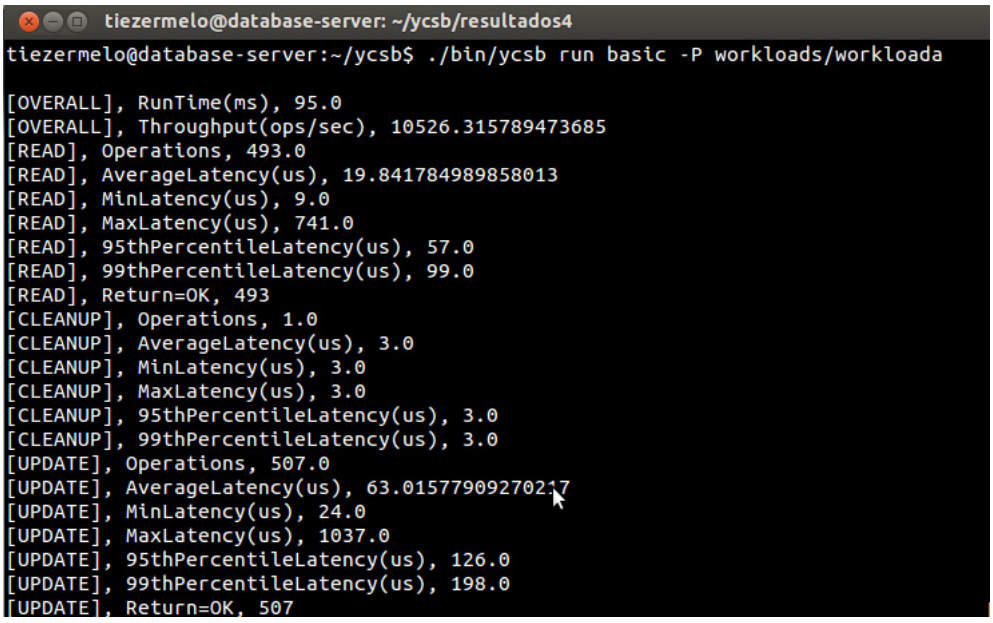
- *Insertproportion*: a proporção de operações de inserção a ser realizada sobre os dados;
- *Readmodifywriteproportion*: a proporção de operações de leitura, modificação, e escrita a ser realizada sobre os dados;
- *Operationcount*: o número de operações a serem executadas em cada workload;
- *Maxexecutiontime*: o tempo máximo de execução, em segundos, de cada *workload*. Por padrão, o teste será executado até que todas as operações tenham sido realizadas ou até que o tempo limite especificado tenha sido esgotado;
- *Recordcount*: o número de registros ou documentos a serem inicialmente carregados para a base de dados, sendo, o padrão, 0 (zero) registros.

Alguns exemplos de utilização dos parâmetros são descritos abaixo:

```
$ cd ycsb
$ ./bin/ycsb load basic -P workloads/workloada -p
   recordcount=10000 -p fieldcount=100
$ ./bin/ycsb load basic -P workloads/workloada -p
   recordcount=100000 -threads 10 > arquivoSaida.txt
$ ./bin/ycsb run basic -P workloads/workloada -p
   operationcount=5000 > arquivoSaida.txt
```

No exemplo acima, com o primeiro comando, o diretório do "*ycsb*"é acessado. Com o segundo comando, é realizada a carga dos dados na base de dados com 10.000

Figura 16 – Saída da execução do comando *run* na ferramenta *YCSB* com o banco modelo *basic*.



Fonte: *Print Screen* da saída da execução do comando *run* no *YCSB* com o modelo *basic*.

documentos, onde cada documento será composto por 100 campos. O terceiro comando realizará a carga da base de dados com 100.000 documentos, onde o processo será realizado por 10 *threads* trabalhando concorrentemente, além de salvar o resultado da execução do comando no arquivo de texto "arquivoSaida.txt". O quarto e último comando realizará a execução do *workload*A, através de 5.000 operações, salvando o resultado do teste no arquivo "arquivoSaida.txt".

Alguns outros parâmetros que podem ser inseridos são específicos para cada sistema *NoSQL*.

```
$ ./bin/ycsb load mongodb -s -P workloads/workloada >
   outputLoad.txt
$ bin/ycsb load couchbase -s -P workloads/workloada
$ bin/ycsb load orientdb -s -P workloads/workloada
$ sudo ./bin/ycsb load orientdb -s -P workloads/workload -p
    recordcount=100000 -p orientd.url=plocal:/opt/orientdb/
   databases/ycsb -p orientdb.newdb=true
```

Então, pode-se executar o comando de execução do *workload*, conforme abaixo:

```
$ ./bin/ycsb run mongodb -s -P workloads/workloada >
outputRun.txt
$ ./bin/ycsb run couchbase -s -P workloads/workloada
$ ./bin/ycsb run orientdb -s -P workloads/workload -p
orientd.url=plocal:/opt/orientdb/databases/ycsb -p
orientdb.newdb=true
```

A ferramenta também permite que o resultado da execução de cada teste seja salvo em um arquivo, bastando adicionar ao final do comando um sinal de maior (>), acompanhado do caminho do arquivo, conforme os exemplos abaixo:

```
$ cd ycsb
$ ./bin/ycsb load basic -P workloads/workloada >
   arquivoSaida.txt
$ ./bin/ycsb load basic -P workloads/workloada > resultados
   /arquivoSaida.txt
$ ./bin/ycsb run basic -P workloads/workloada > /home/
   tiezermelo/ycsb/resultados/arquivoSaida.txt
```

O primeiro comando acima, realizará a execução do teste e salvará o resultado do teste no arquivo "arquivoSaida.txt"do diretório "*ycsb*". O segundo comando, executará o comando e salvará o resultado do teste no arquivo "arquivoSaida.txt"do diretório "resultados". O terceiro e último comando executará o comando e salvará o resultado do teste no arquivo "arquivoSaida.txt"do diretório "/home/tiezermelo/ycsb/resultados/arquivoSaida.txt".

Um exemplo de um registro gerado automaticamente pela ferramenta *YCSB* no sistema *Couchbase* é apresentado no Apêndice F. A seguir, serão executados os testes com os sistemas selecionados. Primeiramente, na subseção 5.2.1, serão executados os testes com o sistema *MongoDB*. Na sequência, na subseção 5.2.2, serão executados os testes com o sistema *Couchbase* e, na subseção 5.2.3, serão executados os testes com o sistema *OrientDB*.

5.2.1 Preparação do ambiente de testes com o sistema *MongoDB*

Como visto anteriormente, antes da execução dos testes referentes a fase de execução do *workload*, é necessário que os dados sejam carregados, onde tal operação será executada através do comando "*load*". Ubuntu (2016) apresenta os passos para a realização da fase de carregamento dos dados para o sistema *MongoDB*, realizado pela ferramenta *YCSB*.

```
$ ./bin/ycsb load mongodb -s -P workloads/workloada
```

Após a realização do carregamento dos dados para a base, pode-se executar o comando de execução do *workload*, conforme abaixo realizado com o sistema *MongoDB*:

```
$ ./bin/ycsb run mongodb -s -P workloads/workloada
```

A cada execução do teste relativo ao *MongoDB*, é automaticamente criada uma base de dados de nome "*ycsb*". A fim de evitar erros na próxima execução de testes nesse mesmo sistema, conforme apresentado na Figura 17, é necessário que esta mesma base de dados (*ycsb*) seja devidamente excluída. Este procedimento pode ser realizado através dos comandos abaixo:

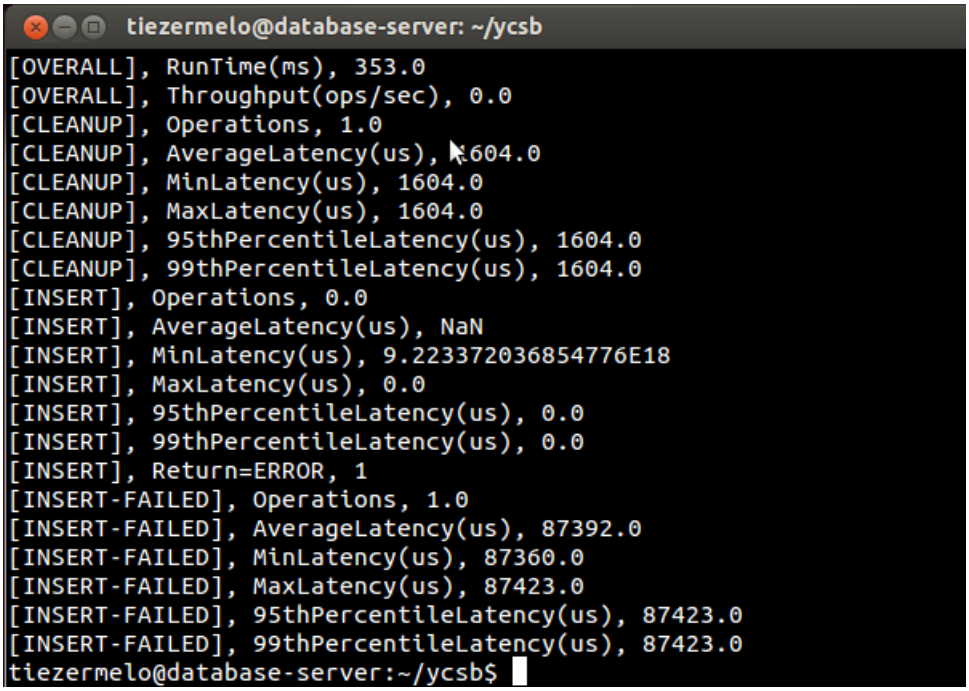
```
$ mongo
$ show dbs
$ use ycsb
$ db.dropDatabase()
$ exit
```

O primeiro comando acima, abre o *MongoDB shell*. O segundo comando apresenta todos as bases existentes no servidor. O terceiro comando define a utilização da base de dados, que, no exemplo, é a base de dados "*ycsb*". O quarto comando exclui a base de dados corrente ou definida, que, no exemplo, exclui a base de dados "*ycsb*". O quinto e último comando faz com que o *MongoDB shell* seja finalizado.

5.2.2 Preparação do ambiente de testes com o sistema *Couchbase*

De forma similarmente aos comandos executados para o sistema *MongoDB*, os comandos para as respectivas fases de carregamento dos dados e execução do *workload*

Figura 17 – Erro apresentado durante execução do *YCSB* no teste com o sistema *MongoDB*.



Fonte: *Print screen* da tela do erro apresentado no *YCSB*

para a realização dos testes no *Couchbase* são:

```
$ cd ycsb
$ ./bin/ycsb load couchbase -s -P workloads/workloada
```

Então, pode-se executar o devido *workload*:

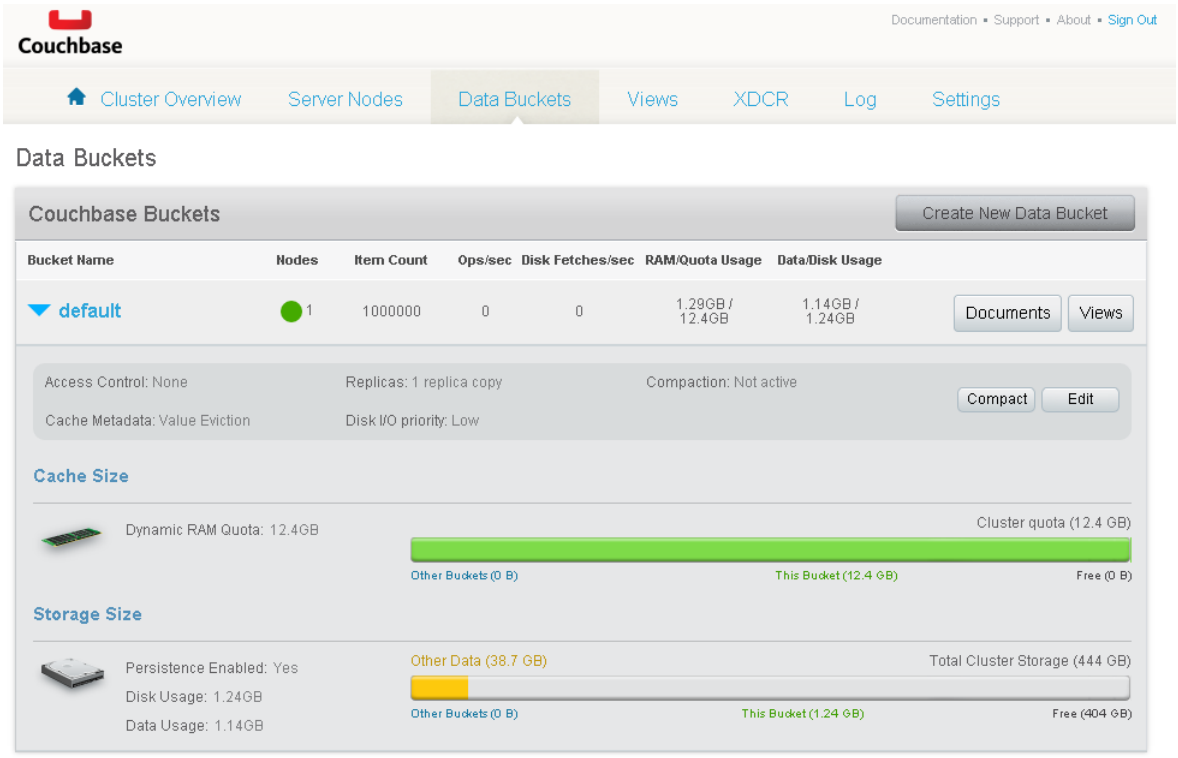
```
$ ./bin/ycsb run couchbase -s -P workloads/workloadb
```

No exemplo acima, foi executado o *workload*B. Após a realização de todos os testes sobre um *data bucket*, e antes de realizar uma nova carga dos dados para a base, da mesma forma que no *MongoDB*, é necessário que a base de dados atual seja excluída. Tal processo pode ser realizado através do procedimento a seguir. Primeiramente, é necessário acessar o servidor através de um navegador de internet, pelo endereço *IP* e porta específicos para conexão no *Couchbase* que, neste trabalho, é http://192.168.0.19:8091, informando o usuário e senha de acesso necessários.

Posteriormente, será necessário acessar o menu *Data Buckets*, na parte superior da tela, selecionando o *data bucket* de interesse, que, no exemplo, foi o *data bucket* "*default*", e clicar no botão "*Edit*"(Figura 18)

Após isso, será necessário clicar no botão "*Delete*", em vermelho, localizado no canto inferior esquerdo da tela apresentada (Figura 19), sendo necessário, também, selecionar a opção "*Delete*"na tela que será apresentada.

Figura 18 – Tela apresentada ao selecionar o menu *Data Buckets* do sistema *Couchbase* onde, no exemplo, é apresentado o *data bucket default* e a quantidade de memória *RAM* e espaço em disco consumidos pelo mesmo.



Fonte: *Print screen* da tela do menu *Data Buckets* do sistema *Couchbase* pelo navegador de internet.

Caso seja de interesse realizar mais testes, após a realização dos procedimentos de exclusão anteriormente descritos, será necessária a criação de um novo *data bucket*. Este procedimento pode ser realizado clicando-se no botão "*Create new Data Bucket*", preenchendo o campo "*Bucket Name*", no exemplos sendo preenchido como "*default* e selecionando "*Bucket Type*"como "*Couchbase*"(Figura 20), e finalizando o processo de criação ao clicar no botão "*Create*", em azul, no canto inferior direito da tela. (Figura 21).

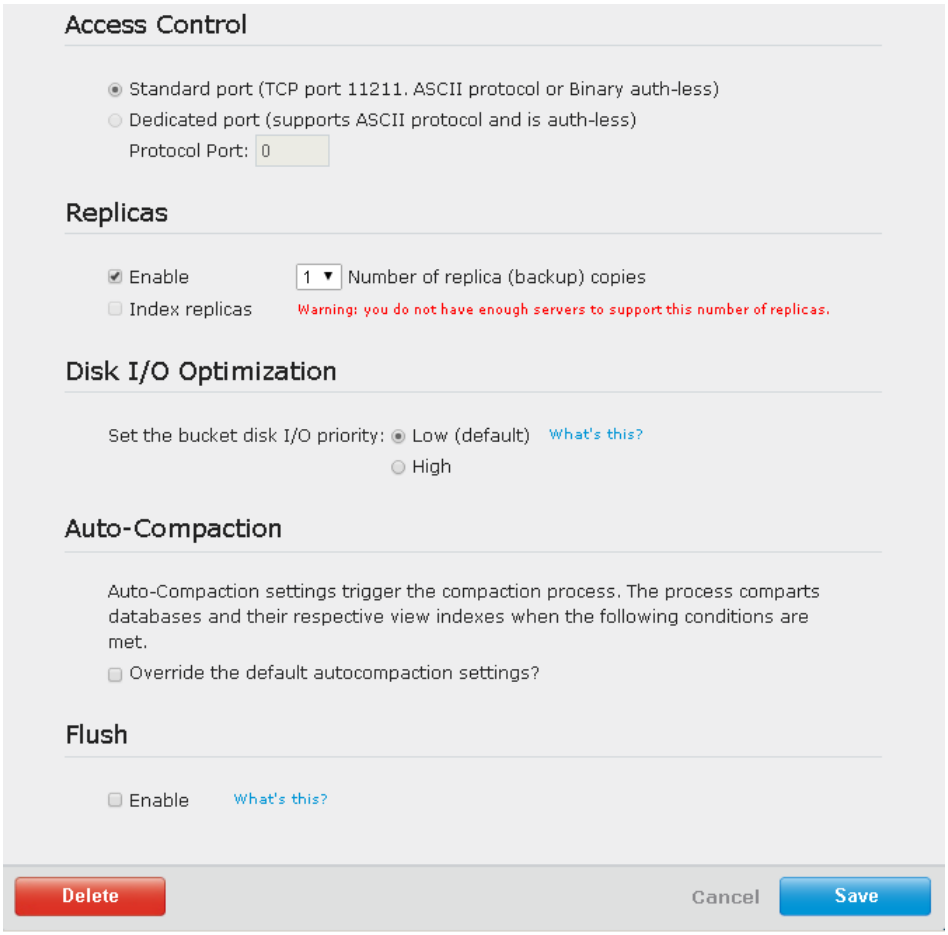
5.2.3 Preparação do ambiente de testes com o sistema *OrientDB*

De forma semelhante à realizada nos sistemas *MongoDB* e *Couchbase*, os comandos para as respectivas fases de carregamento dos dados e execução do *workload* no *OrientDB* são descritos abaixo, porém, sendo necessário que sejam utilizados alguns outros parâmetros para o correto funcionamento dos testes:

```
$ bin/ycsb load orientdb -s -P workloads/workloada -p
   orientdb.url=remote:0.0.0.0:2424 -p orientdb.user=root -
   p orientdb.password=teste -p orientdb.remote.storagetype
   =plocal
```

Acima, foi executado o comando para a carga dos dados para a base "*ycsb*",

Figura 19 – Tela apresentada ao selecionar o botão "*Edit*"para a exclusão de um *data bucket* do sistema *Couchbase*.



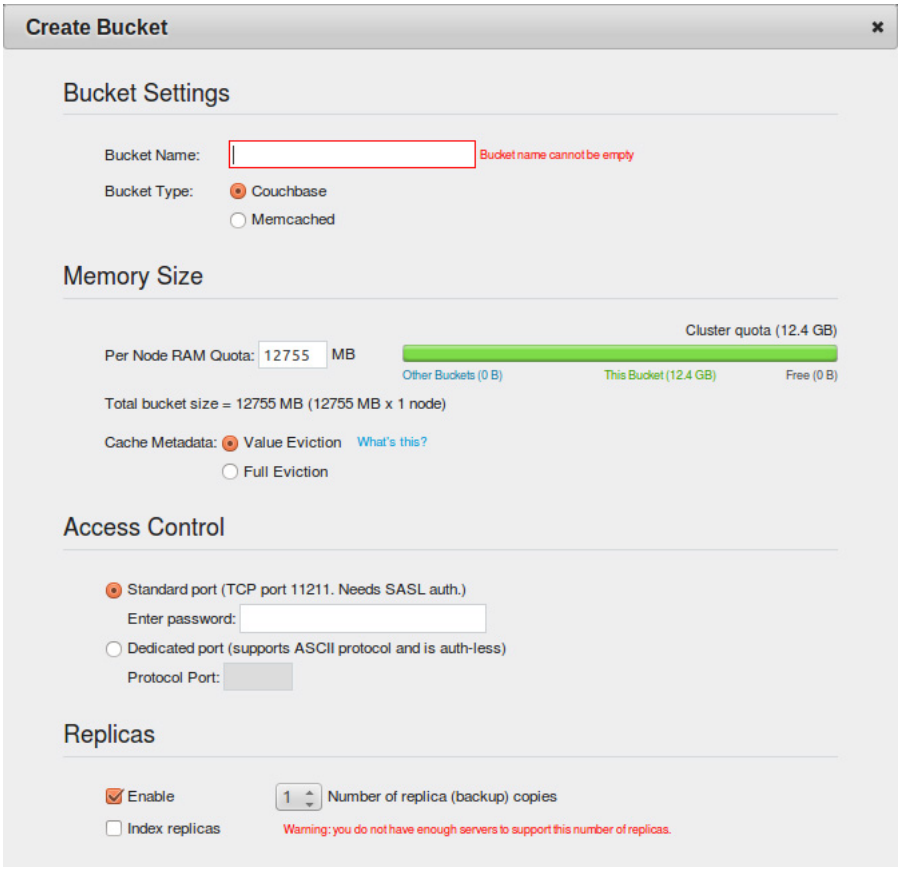
Fonte: *Print screen* da tela de exclusão de um *data bucket* no sistema *Couchbase* pelo navegador de internet.

definido com o parâmetro "*orientdb.url=remote*:0.0.0.0:2424", apresentando também, o usuário, parâmetro "*orientdb.user=root*", e a senha, "*orientdb.password*=teste", utilizados para acesso a base de dados e definidos no momento de criação da mesma, e, finalmente, o parâmetro *orientdb.remote.storagetype=plocal*, necessário para especificar para a ferramenta de testes qual o tipo de base de dados com quem será feita a conexão (este parâmetro pode ser *plocal*, *remote* ou *memory*). Então, pode-se executar o devido *workload* de interesse:

```
$ ./bin/ycsb run orientdb -s -P workloads/workloadf -p
   orientdb.url=remote:0.0.0.0:2424 -p orientdb.user=root -
   p orientdb.password=teste -p orientdb.remote.storagetype
   =plocal
```

No exemplo acima, foi executado o *workload*F sobre o sistema *OrientDB*. Da mesma forma que acontece com os demais sistemas anteriormente citados, também, após a realização de todos os testes sobre a base de dados, e antes de realizar uma nova carga dos dados para a base, é necessário que esta seja excluída. Esse processo pode ser realizado, primeiramente, acessando o endereço do servidor, no exemplo, através do endereço *IP* e

Figura 20 – Primeira parte da tela apresentada ao selecionar o botão "*Create New Data Bucket*"para a criação de um novo *data bucket* no sistema *Couchbase*.

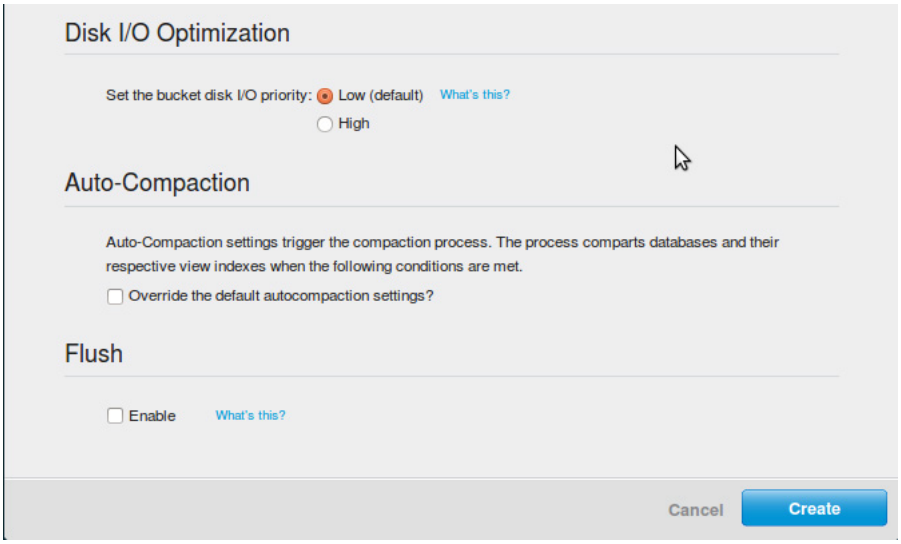


Fonte: *Print screen* da primeira parte da tela de criação *data buckets* no sistema *Couchbase* pelo navegador de internet.

porta http://192.168.0.19:2480 através de um navegador de internet, onde será apresentada a tela inicial do *OrientDB*, sendo necessário selecionar a base de dados a ser excluída, que, no exemplo, foi a base de dados "*ycsb*", e clicar no botão vermelho "*Drop the selected Database*". Posteriormente, informar o usuário e senha específicos da base de dados e clicar no botão "*Drop Database*".

Após o processo anterior, a base de dados terá sido excluída. Caso seja de interesse realizar mais testes com o sistema *OrientDB*, se faz necessário que uma nova base de dados seja criada, podendo, este procedimento, ser realizado clicando-se no botão "*New DB*"da página inicialmente apresentada ao acessar o endereço do servidor, juntamente a porta 2480, onde será apresentada a tela para o preenchimento dos seguintes campos: "*Name*"(nome da base de dados), "*Server User*"(usuário de acesso a base de dados) e "*Server Password*"(senha de acesso a base de dados, sendo, também, necessário marcar a caixa de seleção "*Advanced Options*". Após a seleção desta caixa, será possível selecionar o "*Storage Type* (tipo de armazenamento de dados) como "*plocal*"e o "*Database Type* (tipo da base de dados) como "*document*", finalmente, clicando no botão "*Create databse*", em azul, no canto inferior direito da tela (Figura 22).

Figura 21 – Segunda parte da tela apresentada ao selecionar o botão "*Create New Data Bucket*"para a criação de um novo *data bucket* no sistema *Couchbase*.



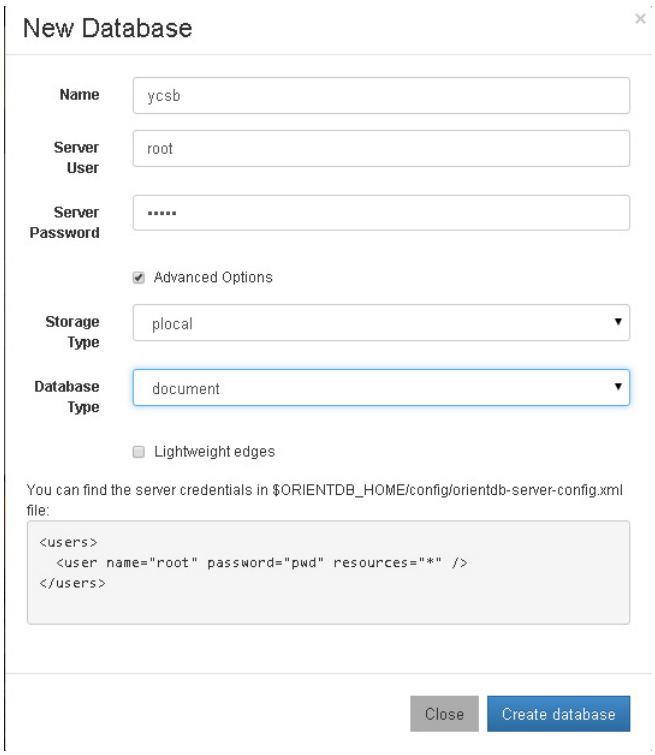
Fonte: *Print screen* da segunda parte da tela de criação de *data buckets* no sistema *Couchbase* pelo navegador de internet.

Ao realizar a criação da base de dados, será apresentada uma nova tela para gerenciamento desta base. Para sair da mesma, basta selecionar a opção "*ycsb(admin)*", no canto superior direito, selecionando, então, a opção "*Log Out*".

5.3 Resultados dos Testes e Comparação do Desempenho dos Sistemas Selecionados

Na seção anterior, foram apresentados os procedimentos para a preparação do ambiente para a execução dos testes com a ferramenta *YCSB*, através de seus respectivos comandos e parâmetros, sobre os sistemas de interesse. Com estes testes foi possível levantar o desempenho dos sistemas, e estabelecer gráficos para a comparação dos resultados obtidos. Na subseção 5.3.1 será apresentado um resumo da composição dos testes executados. Na subseção 5.3.2, será apresentado o resultado obtido nos testes referentes a carga ou carregamento dos dados e os gráficos gerados com estes resultados para a devida comparação dos resultados. Na subseção 5.3.3, será apresentado o resultado obtido nos testes referentes ao *workload* A e os gráficos gerados com estes resultados para a devida comparação dos resultados. Na subseção 5.3.4, será apresentado o resultado obtido nos testes referentes ao *workload* B e os gráficos gerados com estes resultados para a devida comparação dos resultados. Na subseção 5.3.5, será apresentado o resultado obtido nos testes referentes ao *workload* C e os gráficos gerados com estes resultados para a devida comparação dos resultados. Na subseção 5.3.6, será apresentado o resultado obtido nos testes referentes ao *workload* F e os gráficos gerados com estes resultados para a devida comparação dos resultados. Na subseção 5.3.7, será apresentado o resultado obtido nos testes referentes ao *workload* G e os gráficos gerados com estes resultados para a devida comparação dos resultados. Alguns exemplos de comandos utilizados para a realização dos testes com

Figura 22 – Tela apresentada ao selecionar o botão "*New DB*"para a criação de uma nova base de dados no sistema *OrientDB*.



Fonte: *Print screen* da tela de criação de uma nova base de dados no sistema *OrientDB* pelo navegador de internet.

os sistemas *MongoDB*, *Couchbase* e *OrientDD* são, respectivamente, apresentados no Apêndice A, Apêndice B e no Apêndice C.

5.3.1 Resumo da Composição dos Testes

Para a execução dos testes, realizados através da ferramenta *YCSB*, serão seguidos alguns parâmetros, tais como distintos *workloads*, bases de dados de diferentes tamanhos, compostas por documentos de diferentes números de campos e realizados por diferentes números de clientes. Nos testes, serão realizadas operações com 1 cliente (*thread*) e, também, operações com 100 e 250 clientes, visando analisar o desempenho dos sistemas no gerenciamento de operações concorrentes, onde serão utilizados documentos compostos de 10 e 100 campos. Serão apresentados os valores do tempo de execução de cada teste ou *workload*. Para os testes realizados com documentos compostos por 10 campos, serão utilizados 10.000, 50.000, 100.000, 500.000 e 1.000.000 documentos, que serão retratados, respectivamente, como 10K, 50K, 100K, 500K e 1M documentos. Já, para os testes com documentos compostos por 100 campos, serão utilizados apenas 10.000, 50.000 e 100.000 documentos, que serão, respectivamente, representados como 10K, 50K e 100K documentos. Na Figura 23 é apresentado uma tabela com o resumo das operações que serão utilizadas, em azul, as operações realizadas com documentos compostos por 10 campos e, em verde, as operações realizadas com documentos compostos por 100 campos, com a respectiva

quantidade de documentos indicado na coluna "NÚMERO DE DOCUMENTOS". Também, os *workloads* que irão compor os testes foram apresentados anteriormente na Tabela 3.

Figura 23 – Tabela com o resumo das operações a serem realizadas nos testes.

NÚMERO DE DOCUMENTOS	NÚMERO DE CAMPOS DO DOCUMENTO	NÚMERO DE THREADS
10k	10	1
		100
		250
	100	1
		100
		250
50k	10	1
		100
		250
	100	1
		100
		250
100k	10	1
		100
		250
	100	1
		100
		250
500k	10	1
		100
		250
1M	10	1
		100
		250

Fonte: Elaborado pelo autor.

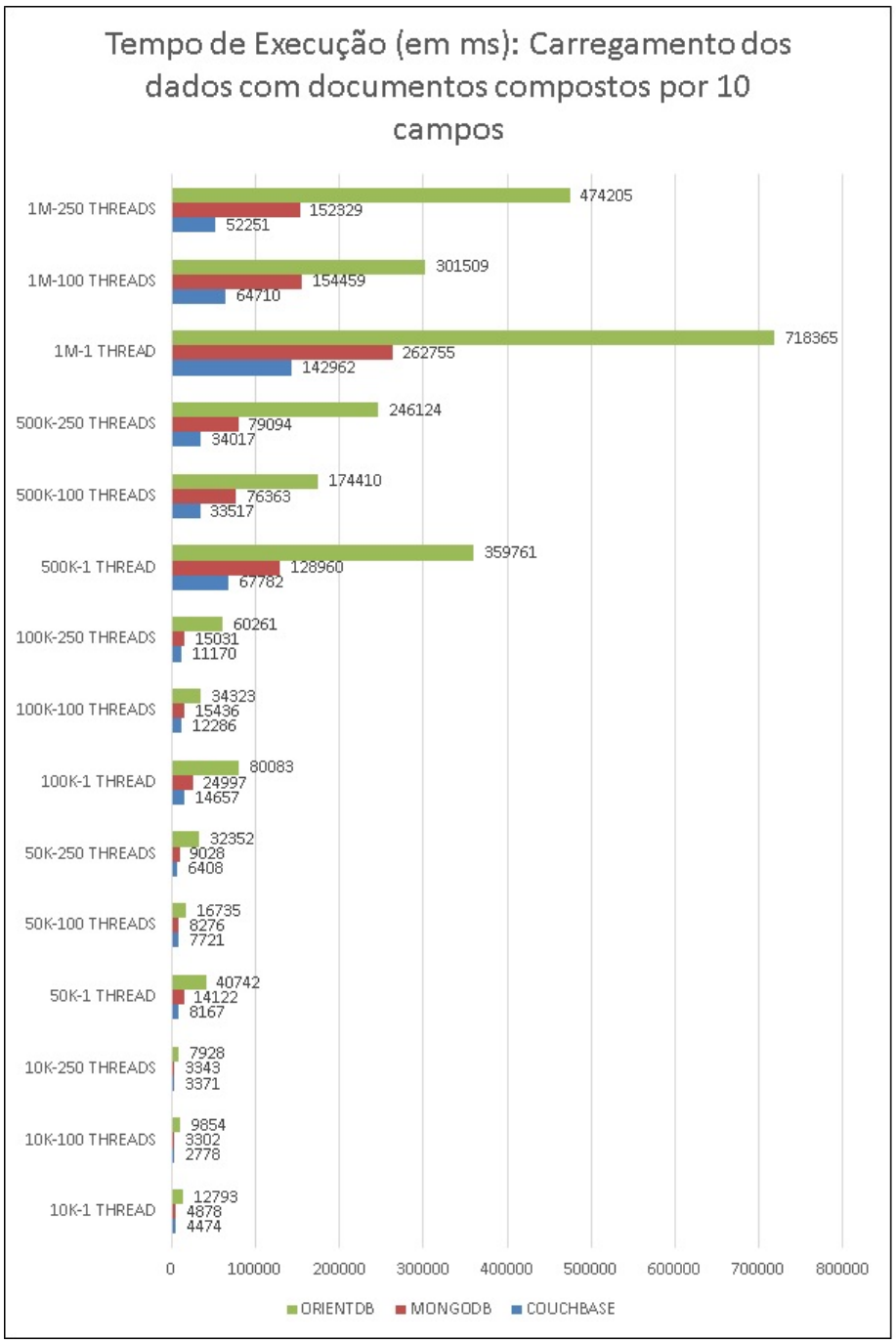
5.3.2 Carga dos dados

Nesta subseção, serão apresentados os valores correspondentes ao carregamento dos dados, sendo levado em consideração o tempo de execução da operação de carga e da quantidade de operações por segundo da mesma.

O resultado apresentado na Figura 24, para a carga de documentos com 10 campos, demonstra que o sistema *OrientDB* apresentou o pior resultado dos testes. Foi possível observar, também, que há diferença no desempenho dos sistemas *MongoDB* e *Couchbase*, onde, o segundo, apresenta melhores resultados. Destaca-se que quando aumenta-se o número de clientes executando operações de forma concorrente, verifica-se que o desempenho do *Couchbase* fica ainda mais visível, sendo, em alguns casos, 50% mais rápido do que o *MongoDB*.

Na Figura 25, para a carga de documentos com 100 campos, foi possível observar, novamente, que o sistema *OrientDB* apresentou o pior desempenho estre todos os demais sistemas, e em todos os tamanhos de bases de dados. Também, novamente, o sistema *Couchbase* foi o que apresentou o melhor resultado do teste, realizando as operações, na grande maioria das execuções, no menor tempo dentre todos. O sistema *MongoDB*, apresentou o melhor dos tempos em apenas um dos testes (10K-250 *Threads*).

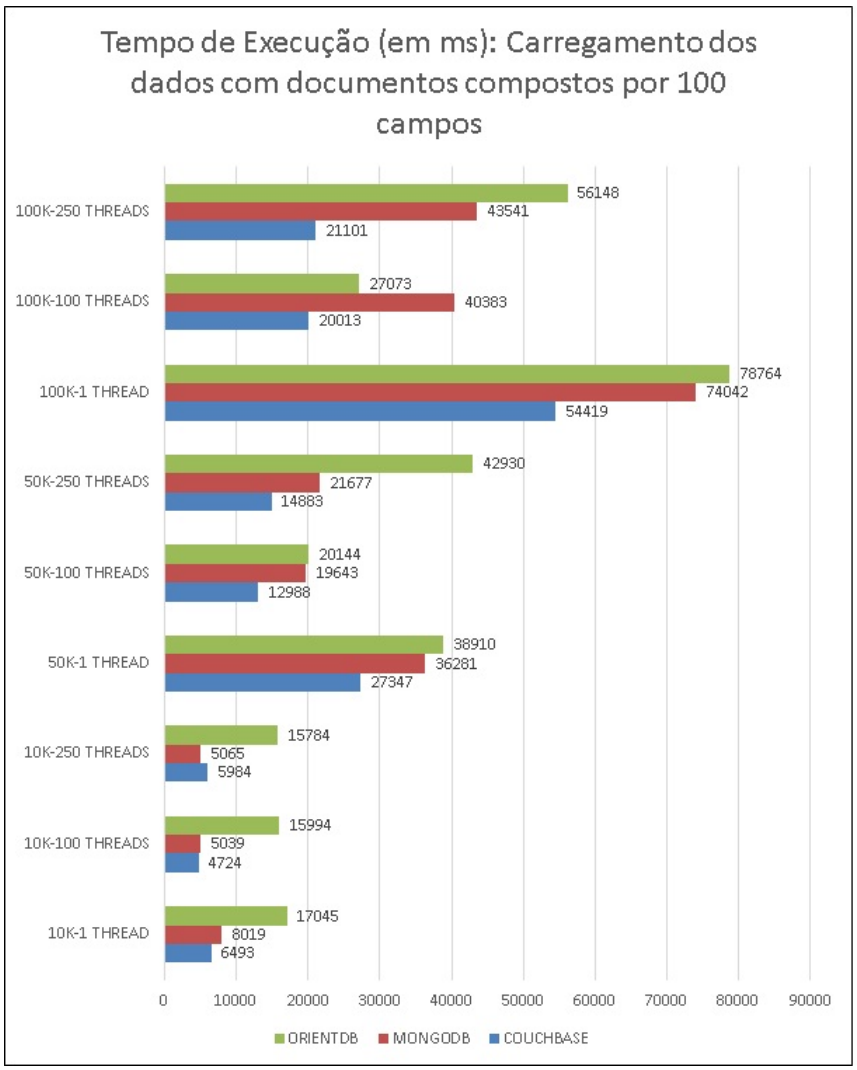
Figura 24 – Gráfico do tempo de execução apresentado no carregamento dos dados para a base e com documentos compostos por 10 campos cada.



Fonte: Elaborado pelo autor.

Em resumo, pode-se observar que, para as operações de carga de dados, dentre os sistemas, o *OrientDB* foi o que apresentou o pior resultado, apresentando o maior tempo de execução em todos os testes, e, no caso contrário, o *Couchbase* foi o sistema que apresentou o melhor desempenho, o menor tempo de execução, na grande maioria dos teste. O sistema *MongoDB* apresentou o melhor desempenho em alguns casos, mas nunca na maior parte destes. Na Tabela 5, o tamanho das bases de dados, em *Megabytes* e

Figura 25 – Gráfico do tempo de execução apresentado no carregamento dos dados para a base e com documentos compostos por 100 campos cada.



Fonte: Elaborado pelo autor.

Gigabytes, após a realização da carga dos dados em cada sistema. Salienta-se que como não são realizados testes de inserção de dados, o tamanho das bases de dados não serão alterados durante a realização dos demais *workloads* dos testes.

5.3.3 *Workload* A

Nesta subseção, serão apresentados os valores correspondentes ao *workload* A, composto por 50% de operações de leitura sobre os dados e 50% de operações de atualização sobre os dados, levando-se em consideração o tempo total de execução do teste e da quantidade média de operações por segundo realizadas. O primeiro teste utilizando o *workload* A foi realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes.

Tabela 5 – Configuração do ambiente de *hardware* dos testes.

	MongoDB	*Couchbase*	*OrientDB*
10K - 10 campos	203,1 MB	18 MB	52 MB
50K - 10 campos	453,1 MB	65,4 MB	125 MB
100K - 10 campos	945,3 MB	124 MB	252 MB
500K - 10 campos	1,93 GB	611 MB	774 MB
1M - 10 campos	3,92 GB	1,15 GB	1,54 GB
10K - 100 campos	453 MB	116 MB	259 MB
50K - 100 campos	1,95 GB	538 MB	696 MB
100K - 100 campos	3,94 GB	1,05 GB	1,59 GB

Fonte: Elaborado pelo autor.

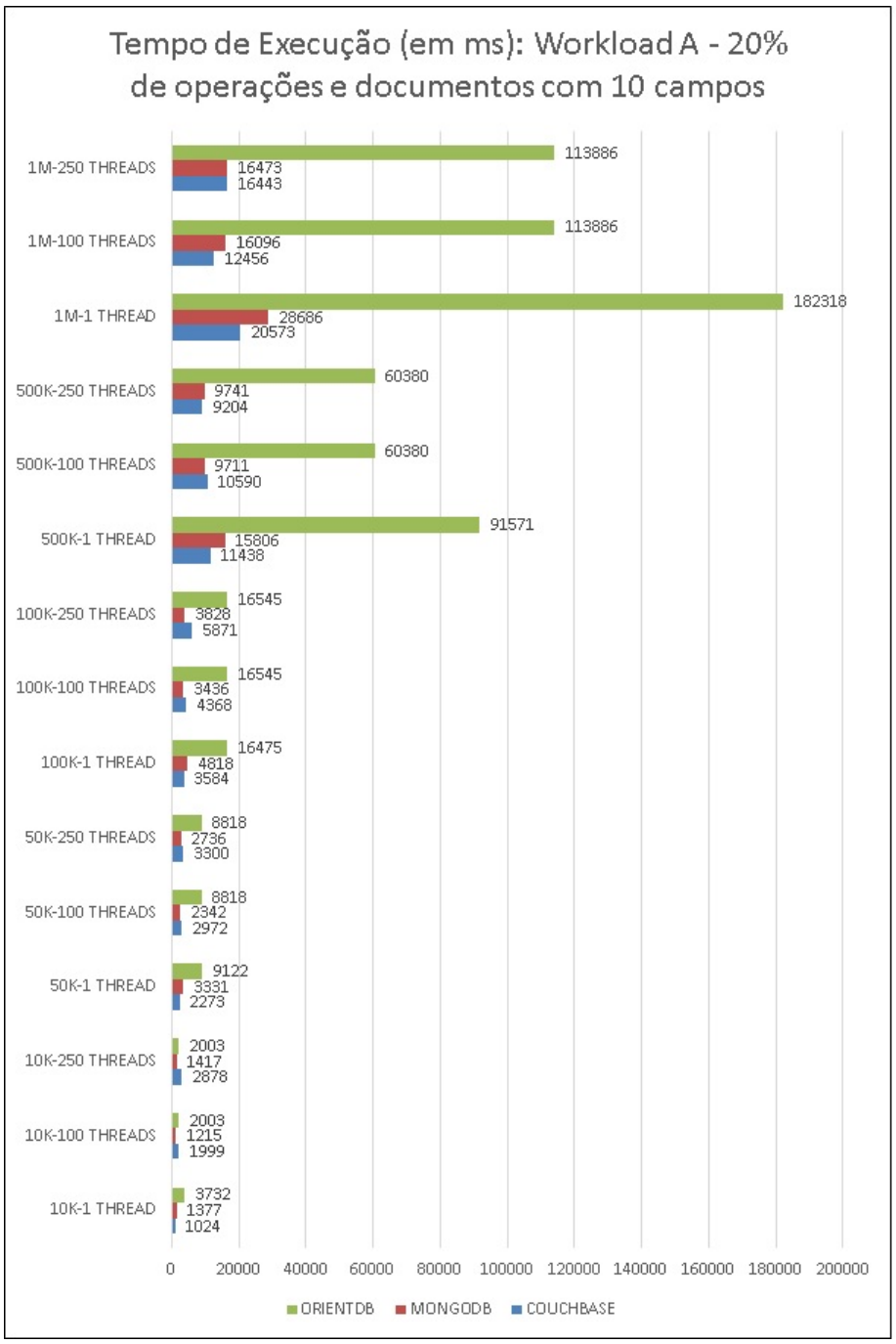
Na Figura 26 pode-se verificar que o sistema *OrientDB*, apresentou o pior dos resultados na grande maioria dos testes, com exceção de um caso apenas. Pode-se verificar, também, que o sistema *Couchbase* foi o que, na maioria das vezes, apresentou o melhor desempenho, mas que em apenas um caso (10K-250 *Threads*), foi o sistema que apresentou o pior desempenho. O sistema *MongoDB* apresentou um desempenho muito similar ao do sistema *Couchbase*, tendo apresentado, em muito casos, o melhor desempenho dos testes.

O segundo teste utilizando o *workload* A foi, também, realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 27 pode-se verificar que o sistema *Couchbase* foi o que apresentou o melhor desempenho entre todos, tendo apresentado o melhor desempenho na grande maioria dos testes. A partir dos testes com 500K documentos, o desempenho superior deste sistema ficou ainda mais visível, devido a este executar as operações em um tempo relativamente mais baixo que os demais sistemas. Diferentemente do caso anterior, o *MongoDB* não apresentou um desempenho similar ao do sistema *Couchbase*, e, em apenas alguns poucos casos, obteve o melhor desempenho. Já o sistema *OrientDB*, manteve-se apresentando o pior desempenho, chegando a ser até 9 vezes mais lento do que o sistema mais rápido (1M-1 *Thread*).

O terceiro teste utilizando o *workload* A foi realizado com 10K, 50K e 100K de documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 28 pode-se verificar que, novamente, o sistema *OrientDB* foi o pior dentre os sistemas, em alguns casos, sendo 6 vezes mais lento que o sistema mais rápido (100K- 1 *Thread*). O sistema *MongoDB* foi o que apresentou o melhor desempenho na maioria dos testes, mas com um desempenho muito próximo ao apresentado pelo sistema *Couchbase*, que, também, apresentou o melhor desempenho em alguns casos.

O quarto teste utilizando o *workload* A foi realizado com 10K, 50K e 100K de

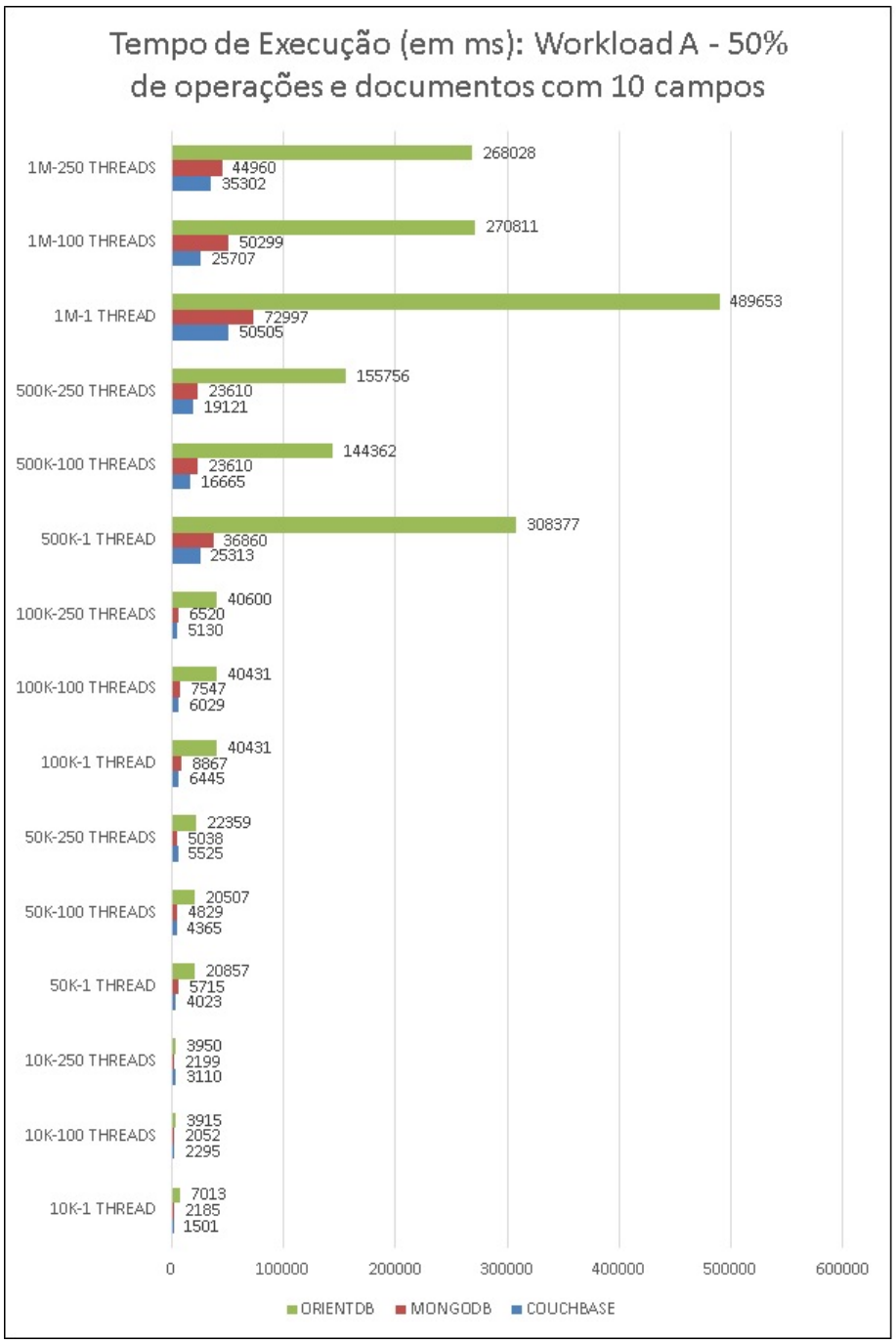
Figura 26 – Gráfico do tempo de execução apresentado com o *workload* A e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.



Fonte: Elaborado pelo autor.

documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 29 pode-se verificar que os sistemas *Couchbase* e *MongoDB* apresentaram um desempenho muito semelhante na maioria dos casos, mas que o primeiro se apresentou como o sistema mais rápido na maioria dos testes. Novamente, o sistema *OrientDB* apresentou o pior desempenho em todos os testes realizados.

Figura 27 – Gráfico do tempo de execução apresentado com o *workload* A e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.

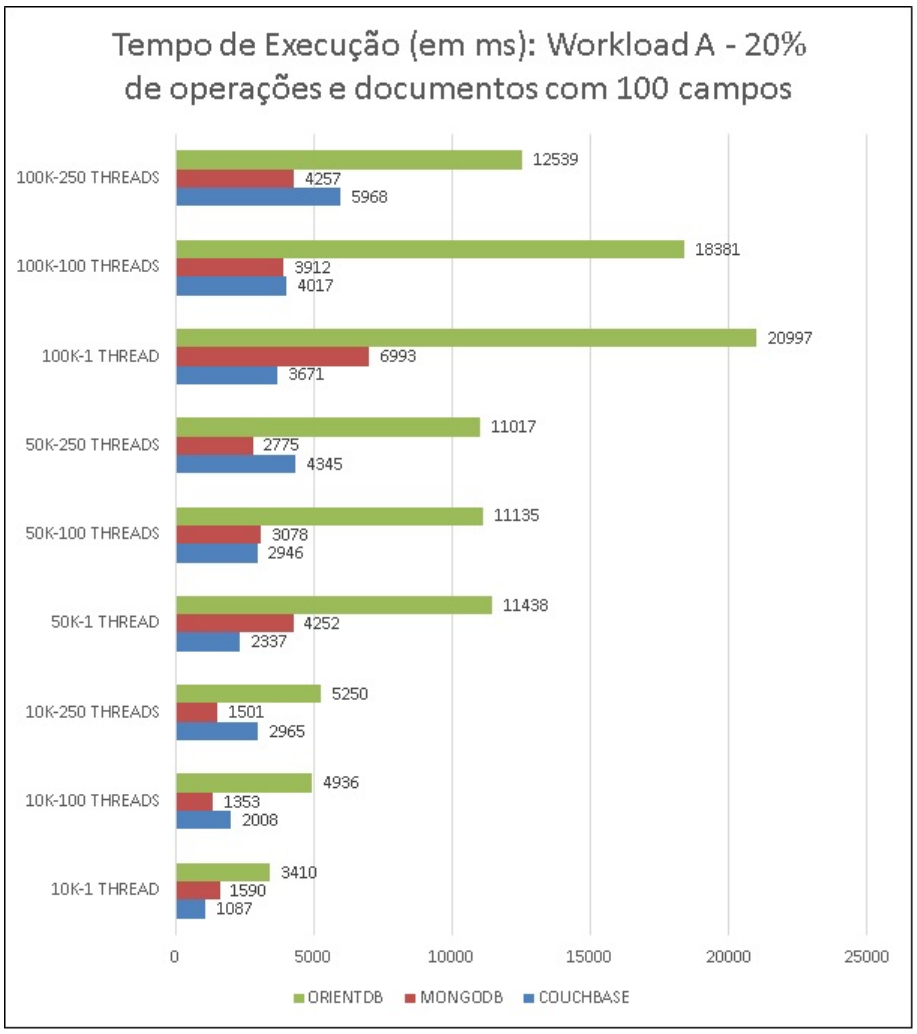


Fonte: Elaborado pelo autor.

5.3.4 *Workload* B

Nesta subseção, serão apresentados os valores correspondentes ao *workload* B, composto por 95% de operações de leitura sobre os dados e 5% de operações de atualização sobre os dados, levando-se em consideração o tempo total de execução do teste da quantidade média de operações por segundo realizadas. O primeiro teste utilizando o *workload* B foi realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por

Figura 28 – Gráfico do tempo de execução apresentado com o *workload* A e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.

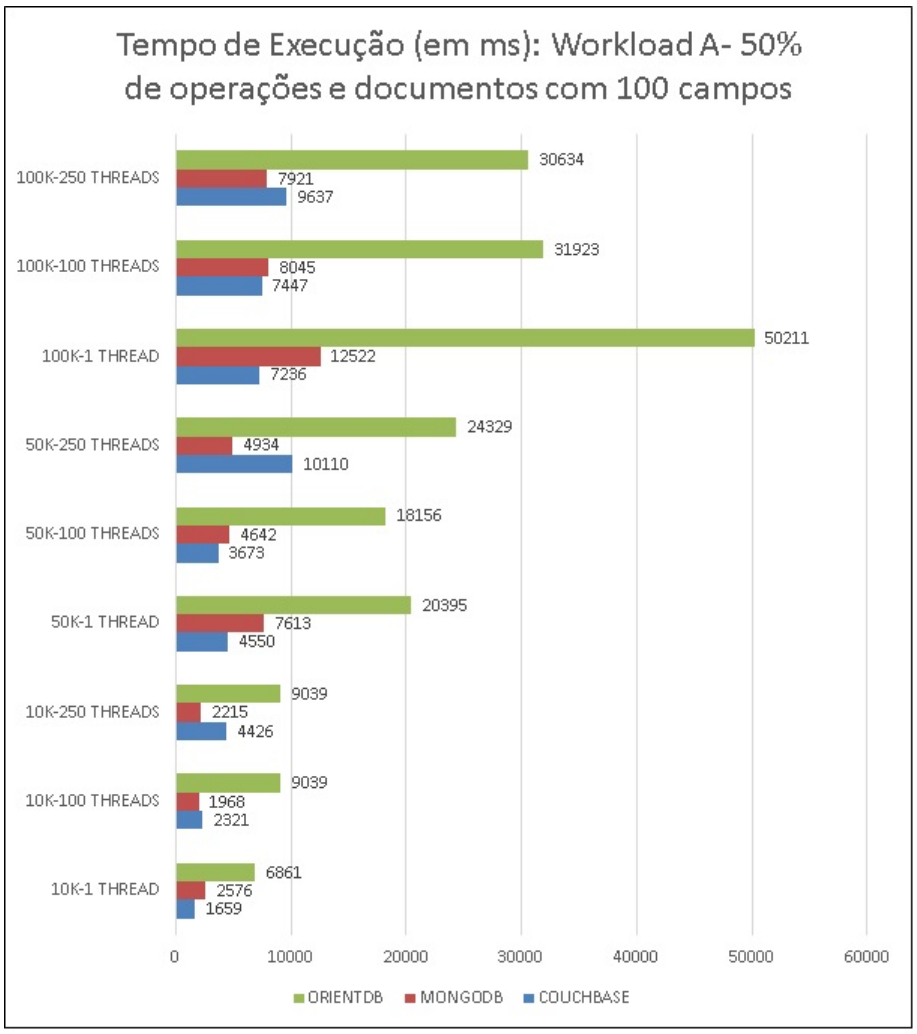


Fonte: Elaborado pelo autor.

10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 30 pode-se verificar que o sistema *OrientDB* apresenta o pior desempenho entre todos os sistemas, onde, em todos os testes, apresentou o tempo de execução mais alto. Em contrapartida, o sistema *Couchbase*, na maioria dos testes, se apresentou com o melhor desempenho entre todos os sistemas, mas com os tempos de execução muito próximos aos apresentados pelo sistema *MongoDB*, que, em alguns casos, chegou a apresentar o melhor desempenho nos testes.

O segundo teste utilizando o *workload* B foi, também, realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 31 pode-se verificar que um padrão de desempenho continua sendo apresentado, onde o sistema *OrientDB* apresenta

Figura 29 – Gráfico do tempo de execução apresentado com o *workload* A e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



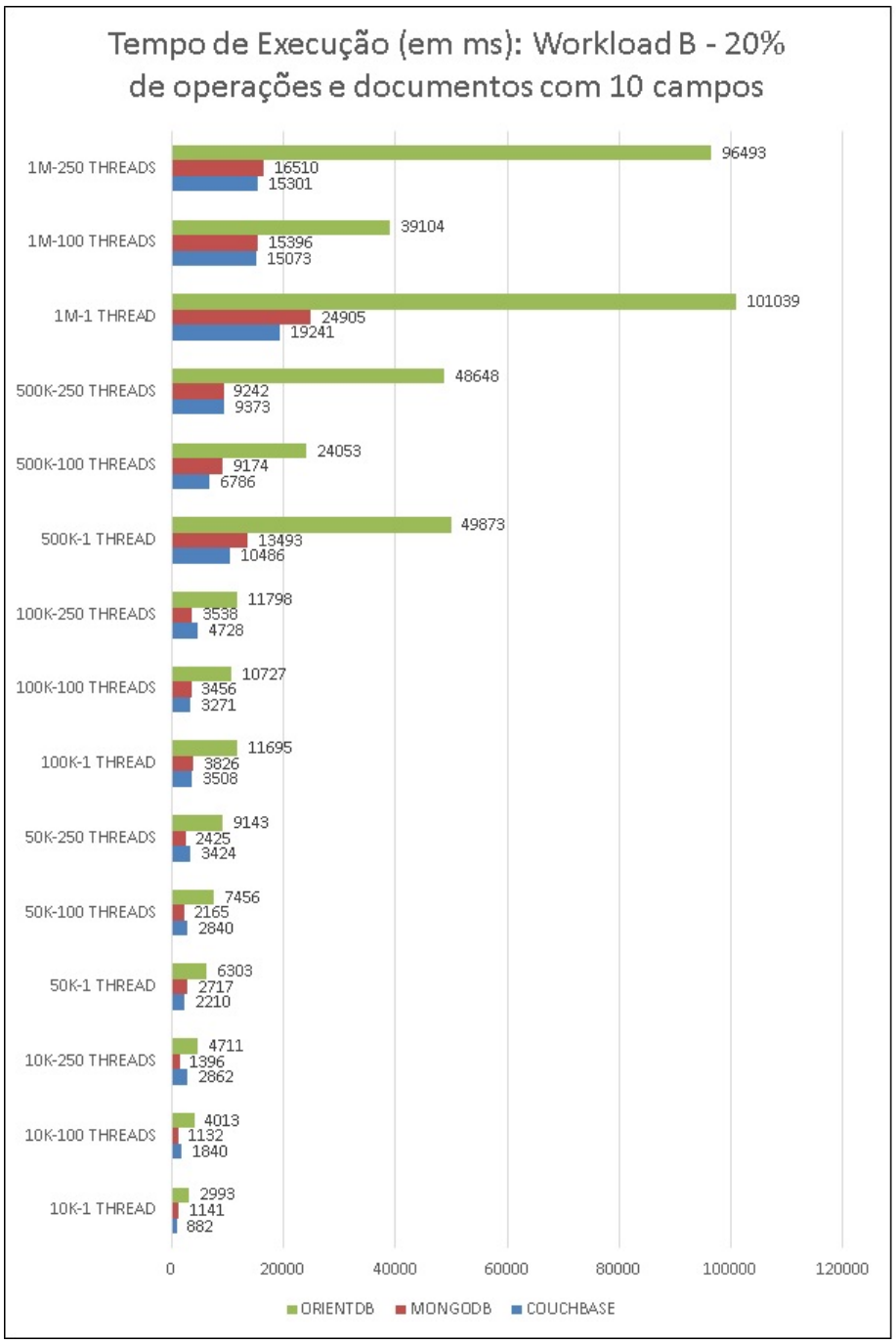
Fonte: Elaborado pelo autor.

o pior dos desempenhos, o sistema *MongoDB*, até os testes com 100K de documentos, apresenta um desempenho muito similar ao do sistema *Couchbase*, porém, a partir dos testes com 500K documentos, fica notável a superioridade do segundo sistema, tendo isso sido apresentado na maioria dos testes.

O terceiro teste utilizando o *workload* B foi realizado com 10K, 50K e 100K de documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 32 pode-se verificar que, mesmo com a utilização de documentos com uma maior número de campos, o sistema *OrientDB* continua sendo o com o pior desempenho. O sistema *MongoDB*, novamente, apresentou o melhor desempenho em alguns casos e o sistema *Couchbase*, o melhor resultado na maioria dos testes.

O quarto teste utilizando o *workload* B foi realizado com 10K, 50K e 100K de

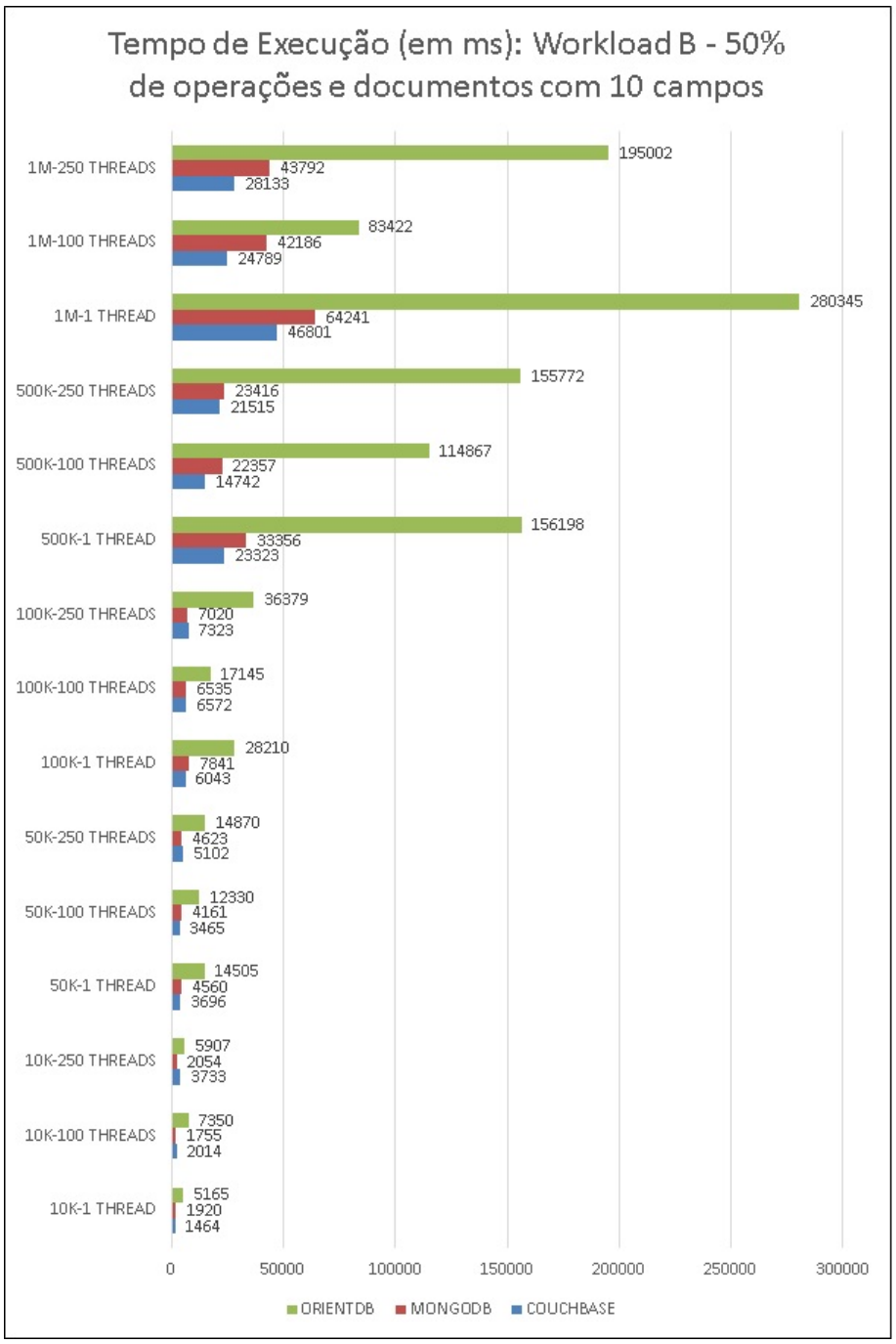
Figura 30 – Gráfico do tempo de execução apresentado com o *workload* B e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.



Fonte: Elaborado pelo autor.

documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 33 pode-se verificar que o sistema *Couchbase* obteve o melhor desempenho na grande maioria dos testes, tendo obtido esse desempenho em 8 casos dos 9 possíveis. O sistema *MongoDB* obteve o melhor resultados em apenas um teste realizado.

Figura 31 – Gráfico do tempo de execução apresentado com o *workload* B e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.



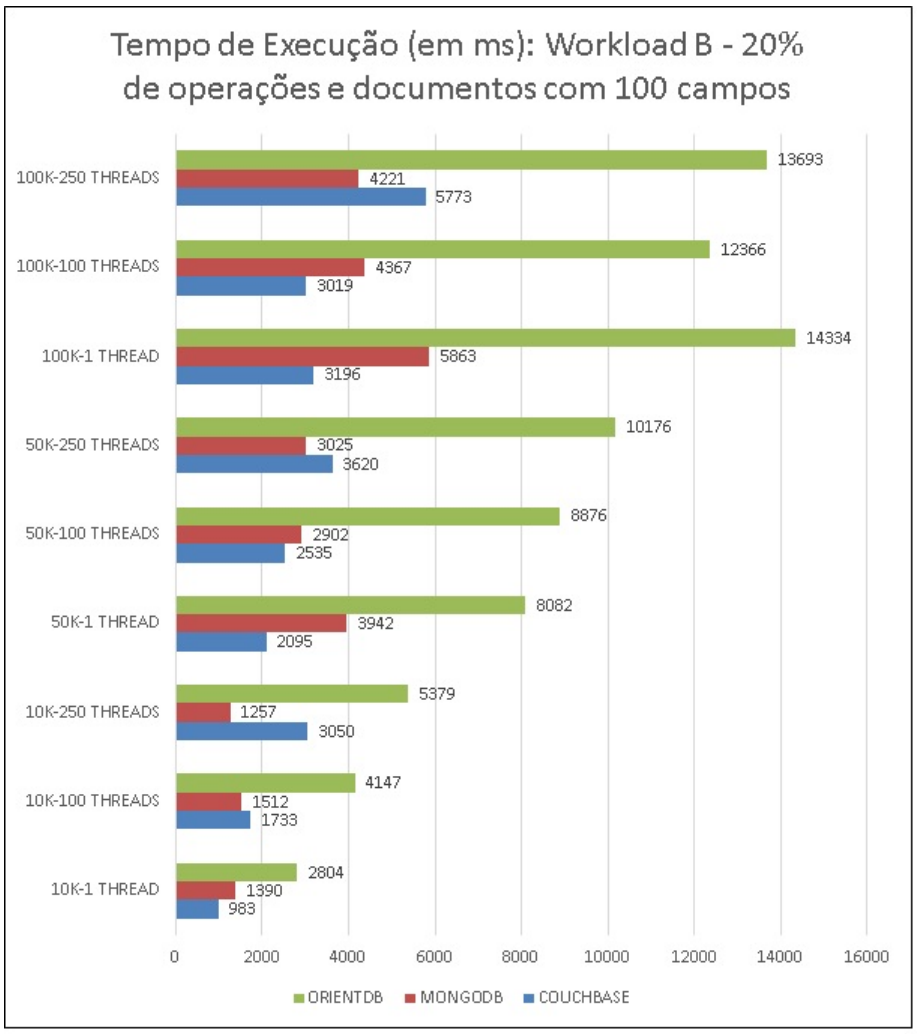
Fonte: Elaborado pelo autor.

5.3.5 *Workload* C

Nesta subseção, serão apresentados os valores correspondentes ao *workload* C, composto por 100% de operações de leitura sobre os dados, levando-se em consideração o tempo total de execução do teste da quantidade média de operações por segundo realizadas.

O primeiro teste utilizando o *workload* C foi realizado com 10K, 50K, 100K,

Figura 32 – Gráfico do tempo de execução apresentado com o *workload* B e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.

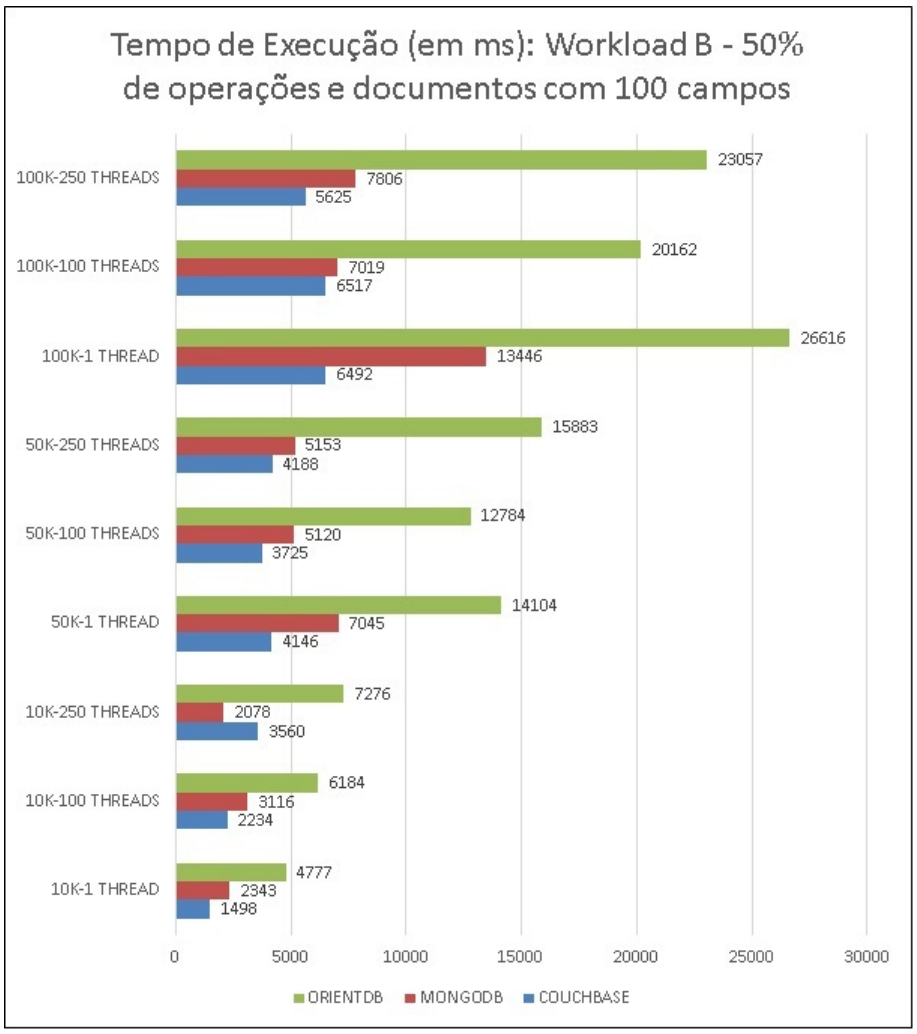


Fonte: Elaborado pelo autor.

500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 34 pode-se verificar que o sistema *OrientDB* foi o que demandou de mais tempo para realizar todos os testes, apresentando-se, assim, como o mais lento dentre todos. Os sistemas *MongoDB* e *Couchbase* apresentaram um tempo de execução muito similar, mas que, na pequena maioria das vezes, foi apresentado pelo primeiro sistema.

O segundo teste utilizando o *workload* C foi, também, realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 35 pode-se verificar que o sistema *Couchbase* foi o que apresentou o menor tempo de execução na grande maioria dos testes, sendo mais visível a partir dos testes realizados com 500K e 1M

Figura 33 – Gráfico do tempo de execução apresentado com o *workload* B e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



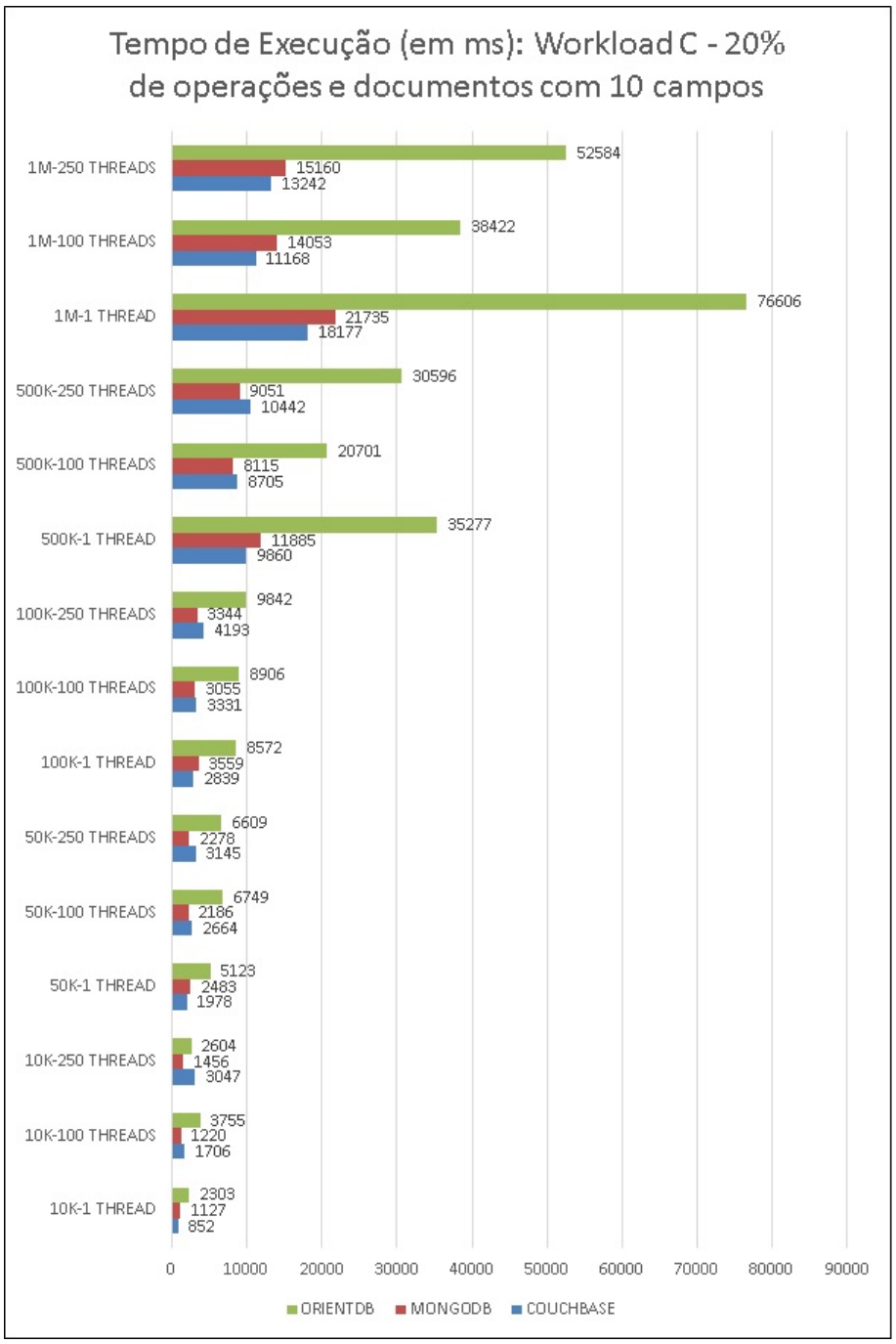
Fonte: Elaborado pelo autor.

documentos. Até os testes compostos por 100K, o sistema *MongoDB* apresentava um desempenho semelhante ao sistema *Couchbase*, porém, na minoria das vezes. Da mesma forma que nos testes anteriores, o sistema *OrientDB* apresentou o pior desempenho dentre todos, chegando a ser sete vezes mais lento do que o sistema mais rápido.

O terceiro teste utilizando o *workload* C foi realizado com 10K, 50K e 100K de documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 36 pode-se verificar que os sistemas *Couchbase* e *MongoDB* apresentaram um desempenho muito semelhante, mas que em sua maioria foi melhor apresentado pelo primeiro sistema, e que o *OrientDB* foi, em média, de duas a três vezes mais lento que os demais.

O quarto teste utilizando o *workload* C foi realizado com 10K, 50K e 100K

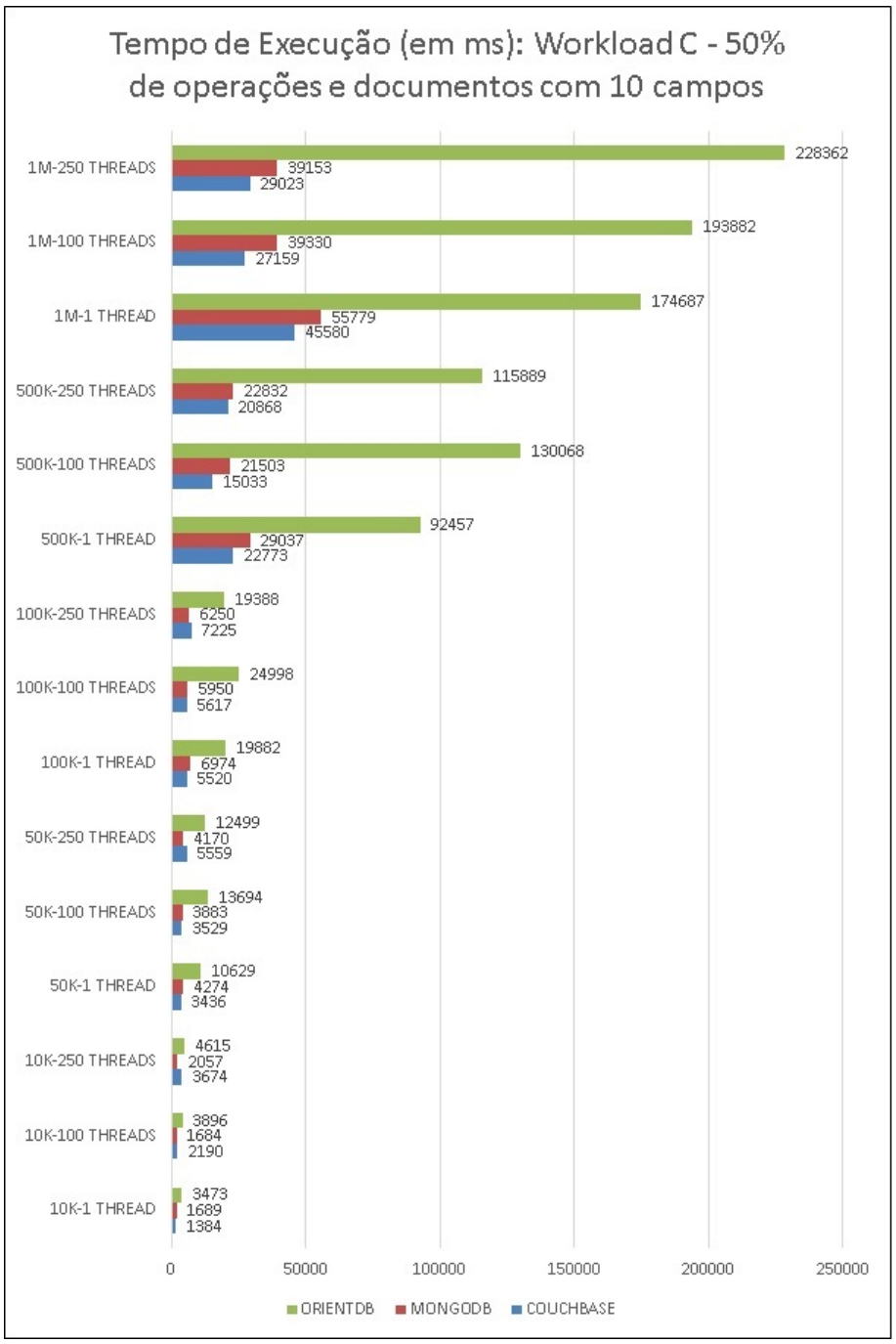
Figura 34 – Gráfico do tempo de execução apresentado com o *workload* C e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.



Fonte: Elaborado pelo autor.

de documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 37 pode-se verificar que, diferentemente aos testes anteriores realizados com documentos também compostos por 100 campos, o sistema *MongoDB* apresentou o melhor desempenho na maioria dos testes, mas o sistema *Couchbase* apresentou um desempenho semelhante ao sistema anterior. Já, o sistema

Figura 35 – Gráfico do tempo de execução apresentado com o *workload* C e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.



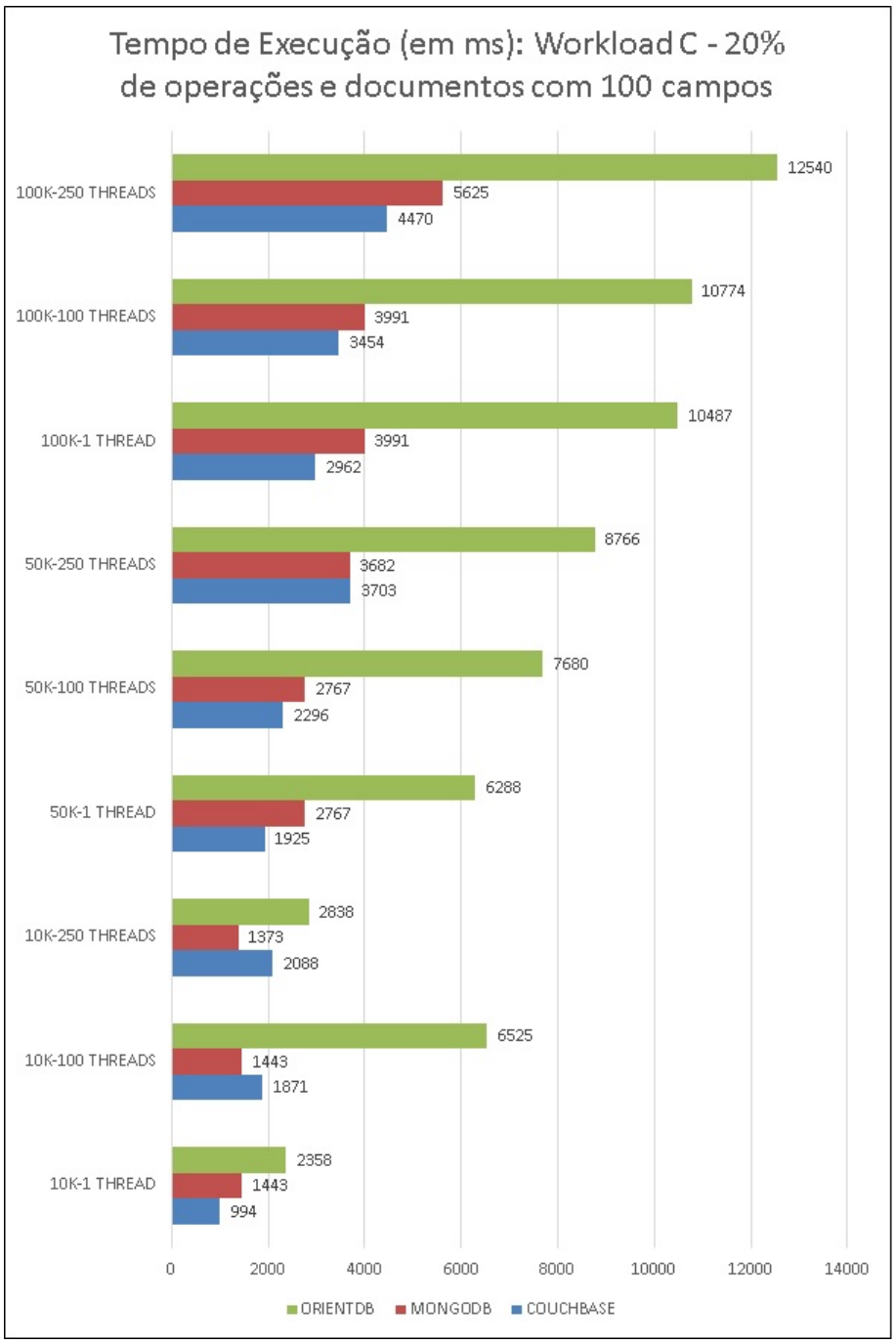
Fonte: Elaborado pelo autor.

OrientDB apresentou-se, novamente, como o sistema mais lento entre todos.

5.3.6 *Workload* F

Nesta subseção, serão apresentados os valores correspondentes ao *workload* F, composto por 50% de operações de leitura sobre os dados e 50% de operações de leitura-modificação-atualização dos dados, onde o dado será lido, modificado e salvando as

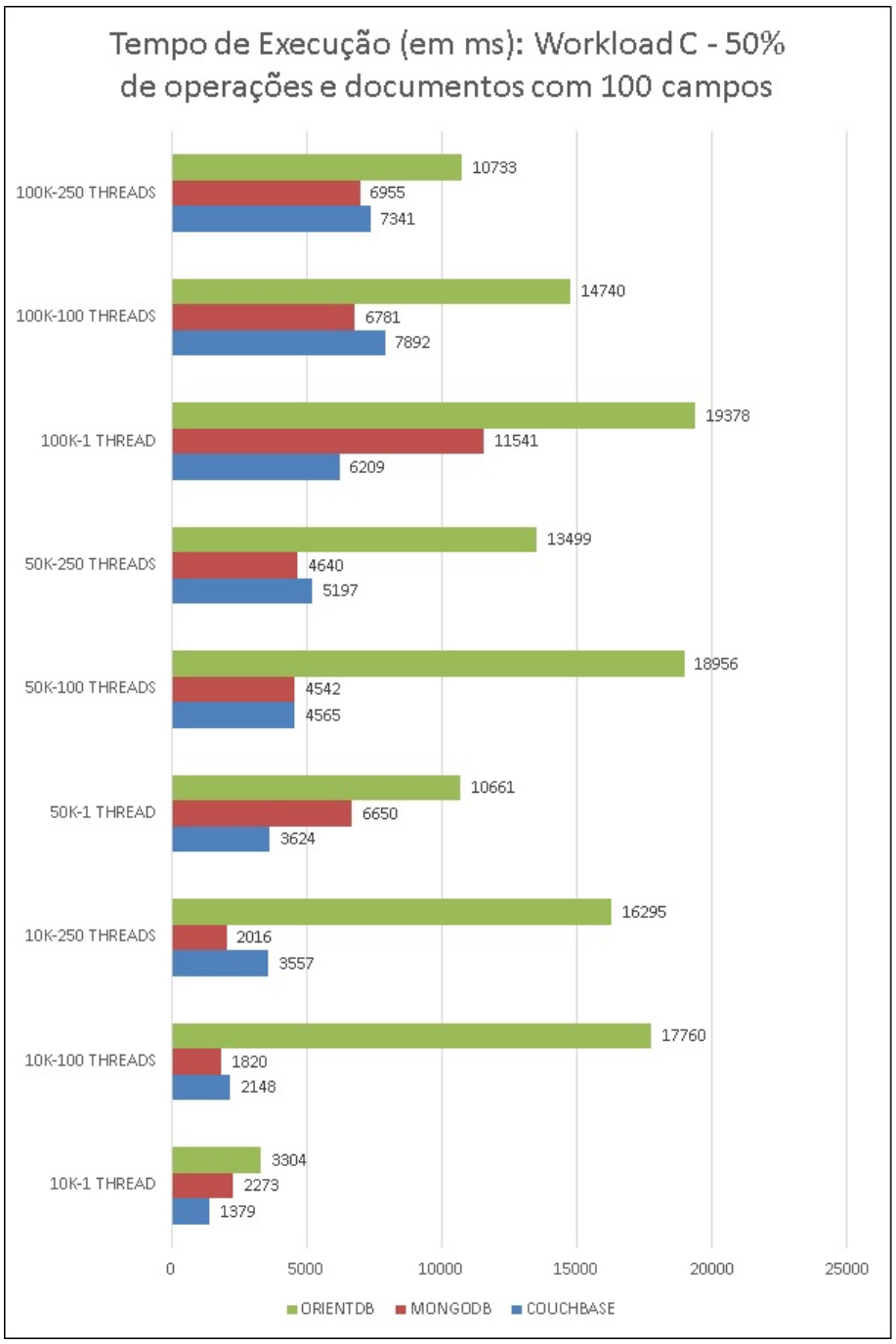
Figura 36 – Gráfico do tempo de execução apresentado com o *workload* C e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



Fonte: Elaborado pelo autor.

modificações na base, levando-se em consideração o tempo total de execução do teste da quantidade média de operações por segundo realizadas. O primeiro teste utilizando o *workload* F foi realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 38 pode-se verificar que, novamente, os sistemas *Couchbase* e *MongoDB*

Figura 37 – Gráfico do tempo de execução apresentado com o *workload* C e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.

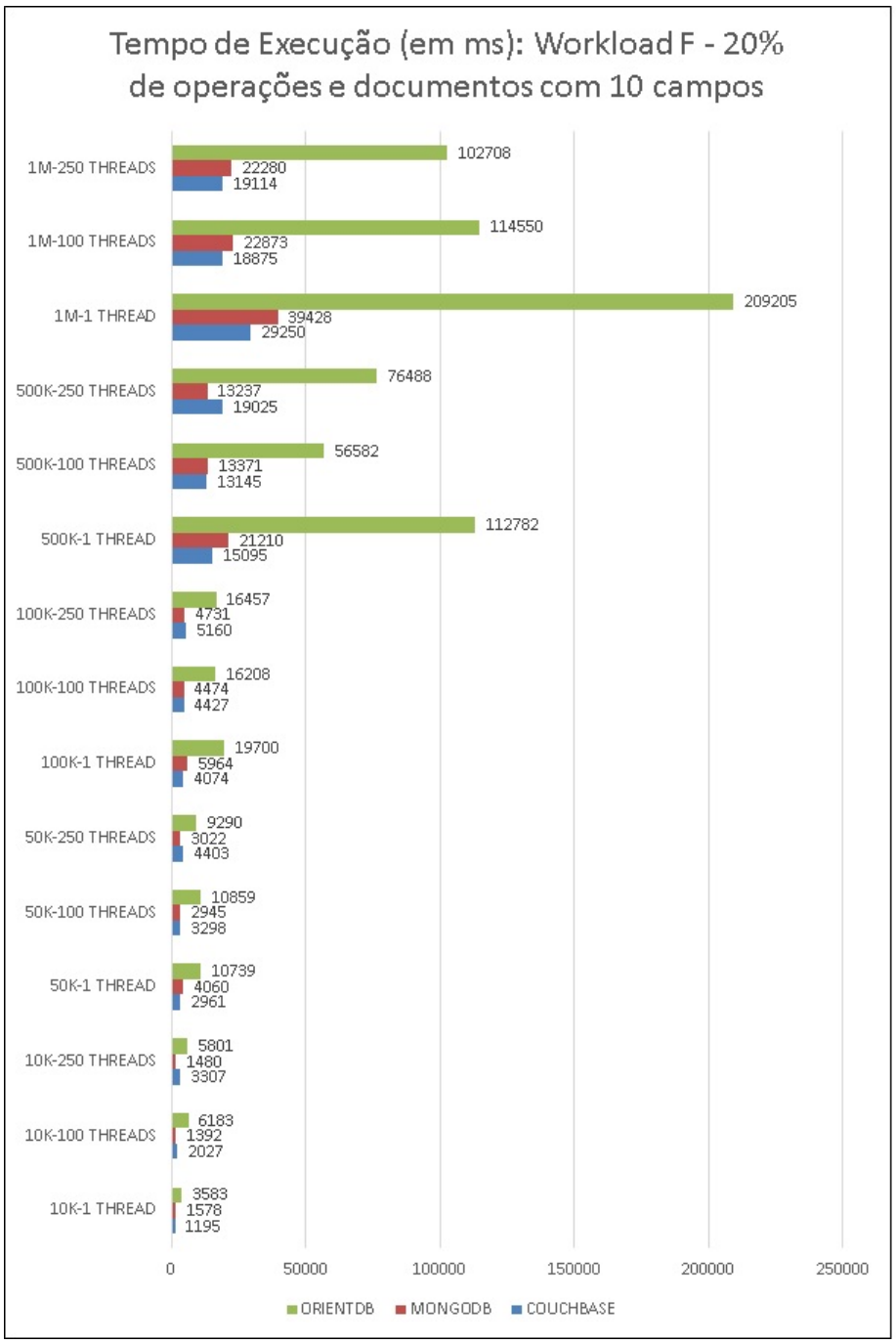


Fonte: Elaborado pelo autor.

apresentaram um desempenho muito semelhante. Porém, o sistema *Couchbase* apresentou o melhor desempenho na maioria dos casos. O sistema *OrientDB*, mais uma vez, foi o sistema mais lento dentre todos.

O segundo teste utilizando o *workload* F foi, também, realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados

Figura 38 – Gráfico do tempo de execução apresentado com o *workload* F e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.

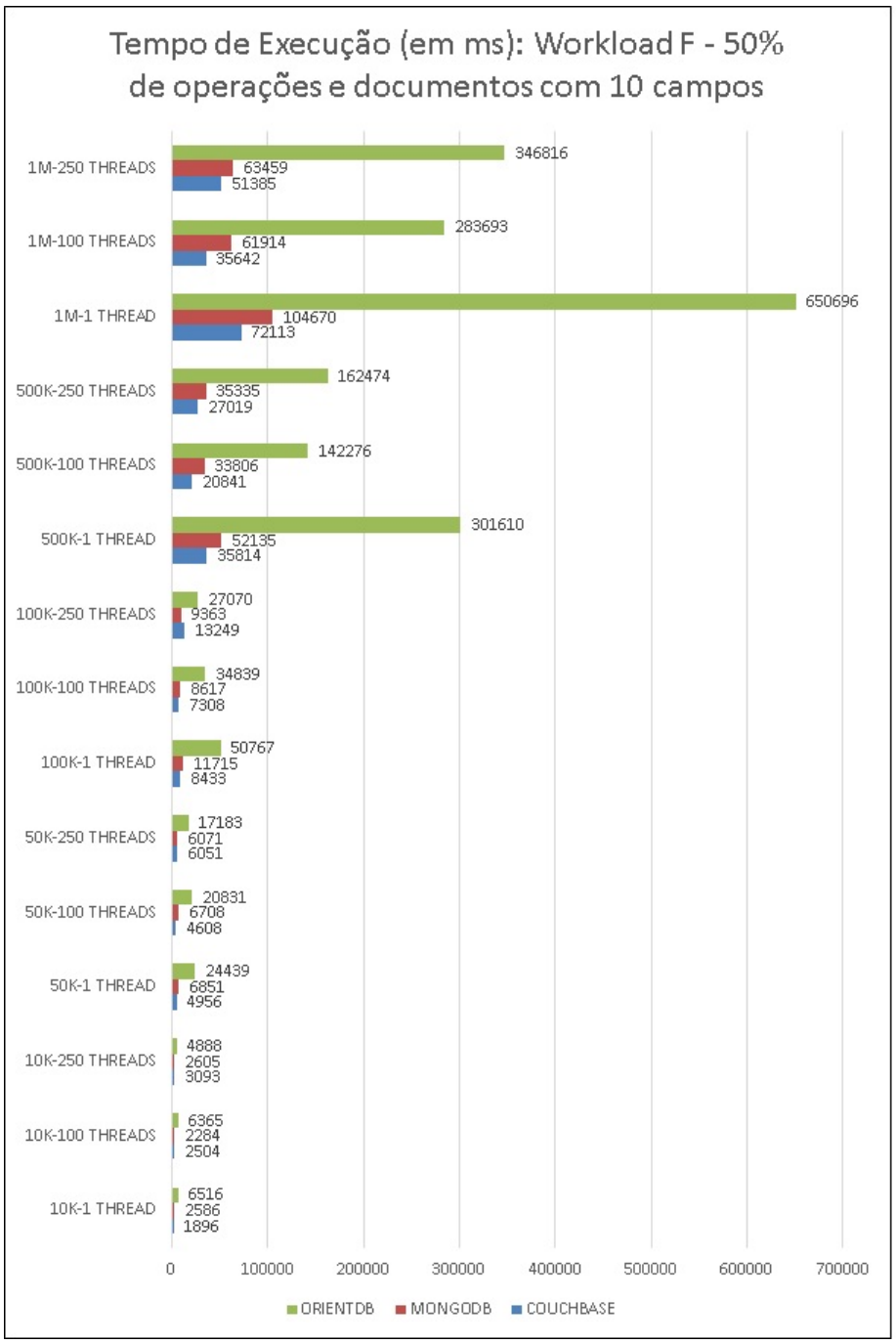


Fonte: Elaborado pelo autor.

como o número de operações a serem realizadas nos testes. Na Figura 39 pode-se verificar que, até os testes com 100K documentos, os sistemas *Couchbase* e *MongoDB* apresentaram um desempenho equivalente, porém, a partir dos testes com 500K, o sistema *Couchbase* apresentou tempos mais baixos em todos os casos. O sistema *OrientDB* foi, mais uma vez, o pior dentre todos os sistemas.

O terceiro teste utilizando o *workload* F foi realizado com 10K, 50K e 100K de

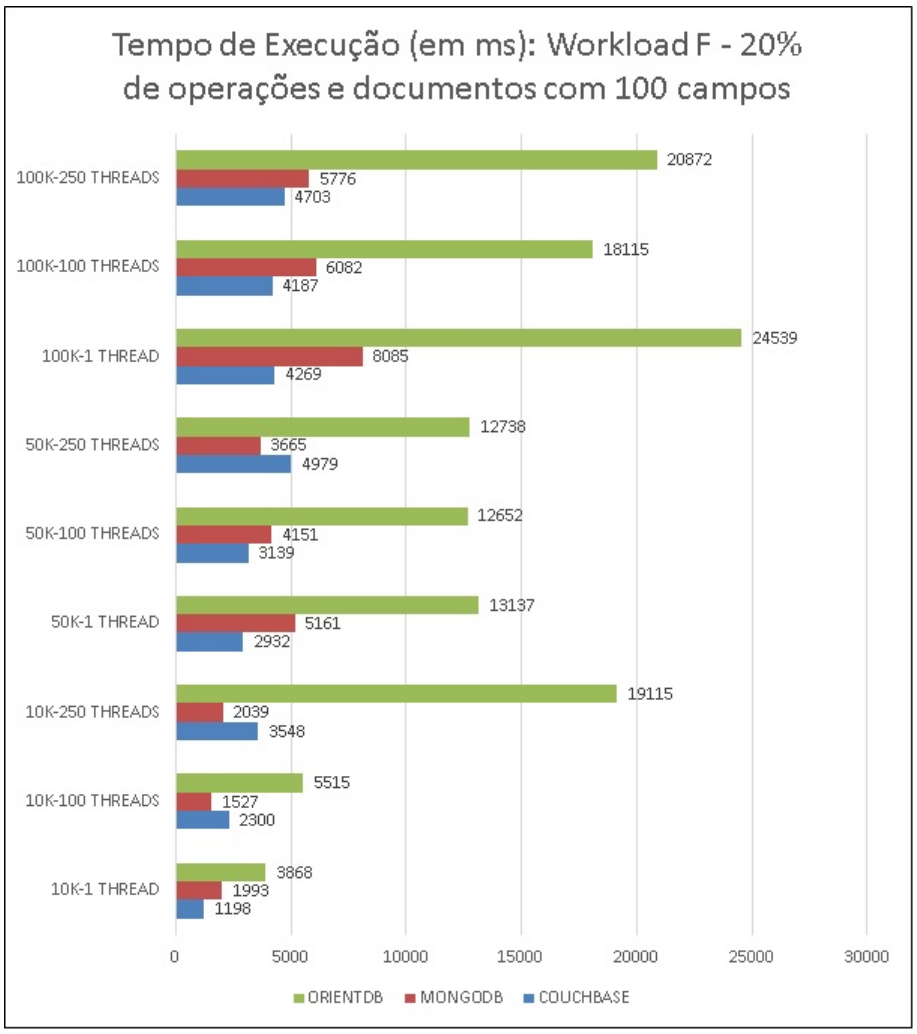
Figura 39 – Gráfico do tempo de execução apresentado com o *workload* F e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.



Fonte: Elaborado pelo autor.

documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 40 pode-se verificar que, novamente, o sistema *OrientDB* apresentou o pior dos desempenhos e que os sistemas *Couchbase* e *MongoDB* apresentaram um desempenho relativamente semelhante, mas o primeiro dos sistemas apresentando um melhor desempenho na maioria dos casos.

Figura 40 – Gráfico do tempo de execução apresentado com o *workload* F e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



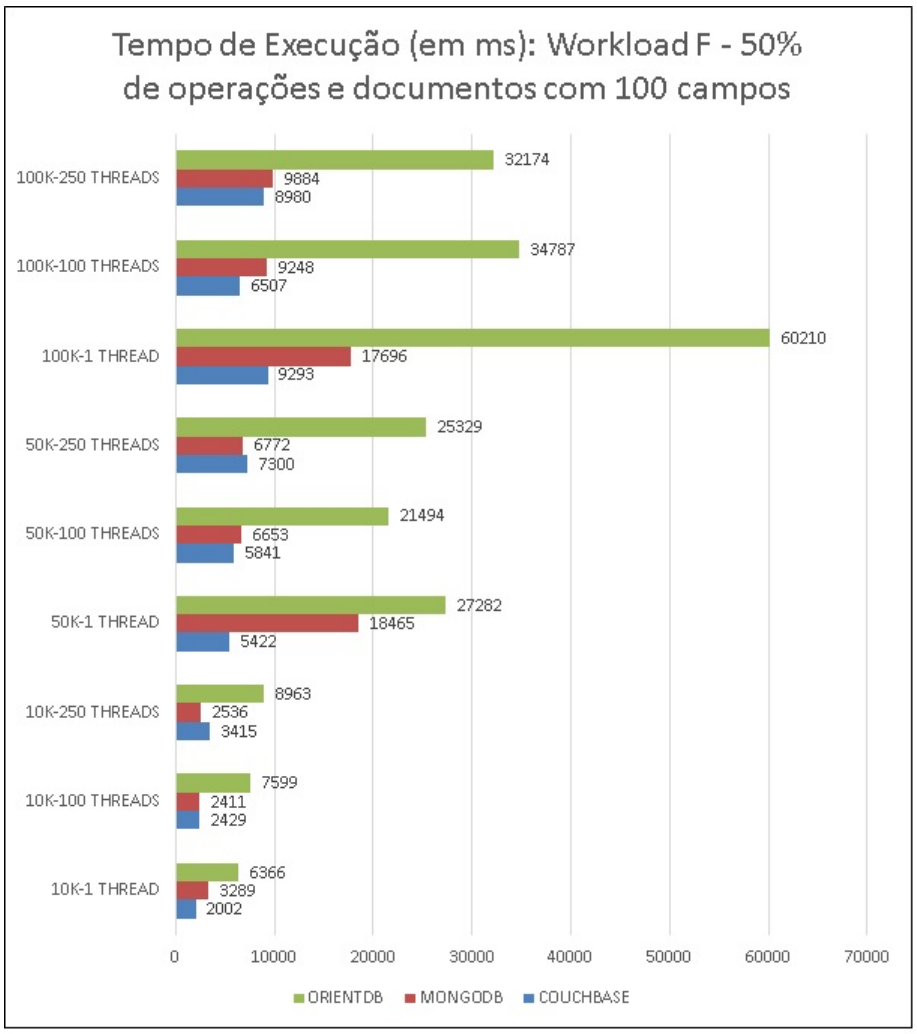
Fonte: Elaborado pelo autor.

O quarto teste utilizando o *workload* F foi realizado com 10K, 50K e 100K de documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 41 pode-se verificar que o sistema *Couchbase* apresentou o melhor desempenho na maioria dos casos e o sistema *MongoDB* o melhor desempenho nos casos restantes. O pior desempenho foi, mais uma vez, apresentado pelo sistema *OrientDB*, que chegou a ser seis vezes mais lento do que o sistema mais rápido no teste 100k-1 *Thread*.

5.3.7 *Workload* G

Nesta subseção, serão apresentados os valores correspondentes ao *workload* G, composto por 95% de operações de atualização sobre os dados e 5% de operações de leitura sobre de dados, levando-se em consideração o tempo total de execução do teste da

Figura 41 – Gráfico do tempo de execução apresentado com o *workload* F e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



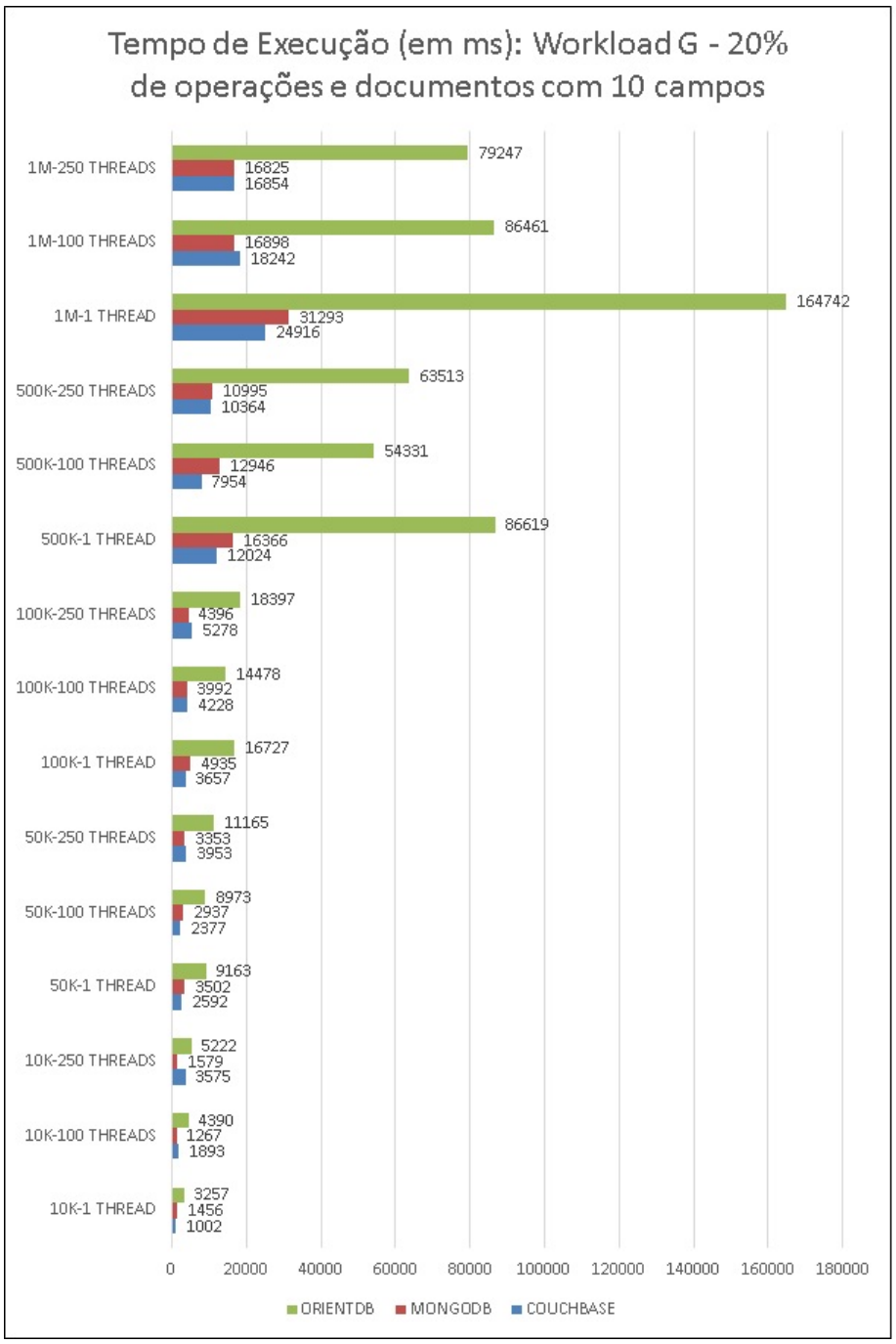
Fonte: Elaborado pelo autor.

quantidade média de operações por segundo realizadas.

O primeiro teste utilizando o *workload* G foi realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 42 pode-se verificar que os testes demonstram que o sistema *OrientDB* continua apresentando o pior desempenho dentre os sistemas e o *Couchbase* o melhor desempenho na grande maioria dos testes. Pode-se observar também que o sistema *MongoDB* apresenta, em alguns momentos, um melhor desempenho se comparado ao *Couchbase*. Porém, na maioria das vezes, o sistema *Couchbase* é quem apresenta os menores tempos.

O segundo teste utilizando o *workload* G foi, também, realizado com 10K, 50K, 100K, 500K e 1M de documentos compostos por 10 campos, e 1, 100 e 250 *threads*, utilizando

Figura 42 – Gráfico do tempo de execução apresentado com o *workload* G e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.

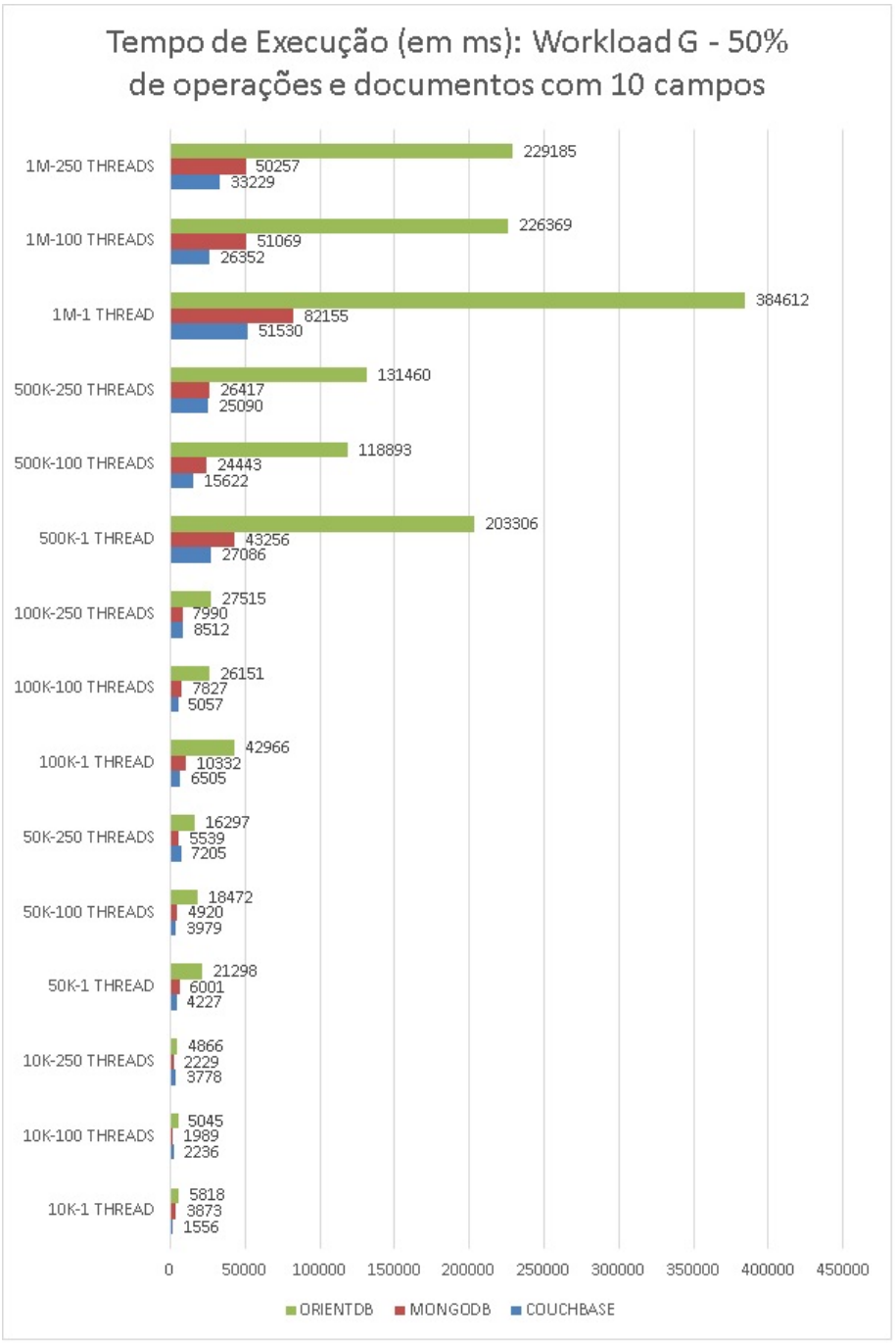


Fonte: Elaborado pelo autor.

como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 43 pode-se verificar que, não diferentemente do teste anterior, o sistema *Couchbase* apresenta o melhor desempenho na grande maioria dos testes executados, tendo o sistema *OrientDB* apresentado os tempos mais altos e, consequentemente, os piores resultados.

O terceiro teste utilizando o *workload* G foi realizado com 10K, 50K e 100K de

Figura 43 – Gráfico do tempo de execução apresentado com o *workload* G e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 10 campos cada.

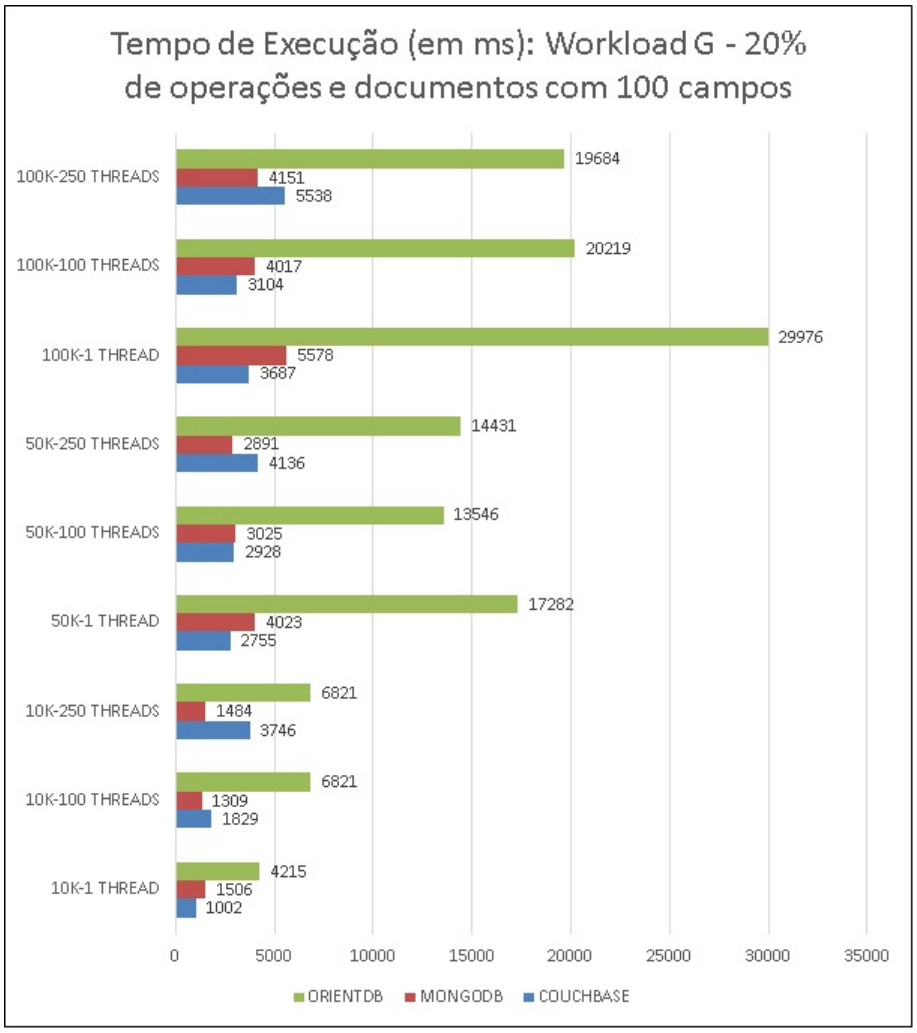


Fonte: Elaborado pelo autor.

documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 20% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 44 pode-se verificar que, mesmo os testes sendo realizados com documentos compostos por um maior número de campos, os resultados apresentados são muito parecidos com os anteriores. O sistema *Couchbase* apresenta o melhor desempenho na grande maioria dos testes executados, o sistema *MongoDB*, em

algumas vezes, apresentou os tempos mais baixos, mas não numa maior quantidade, e o sistema *OrientDB* os piores resultados.

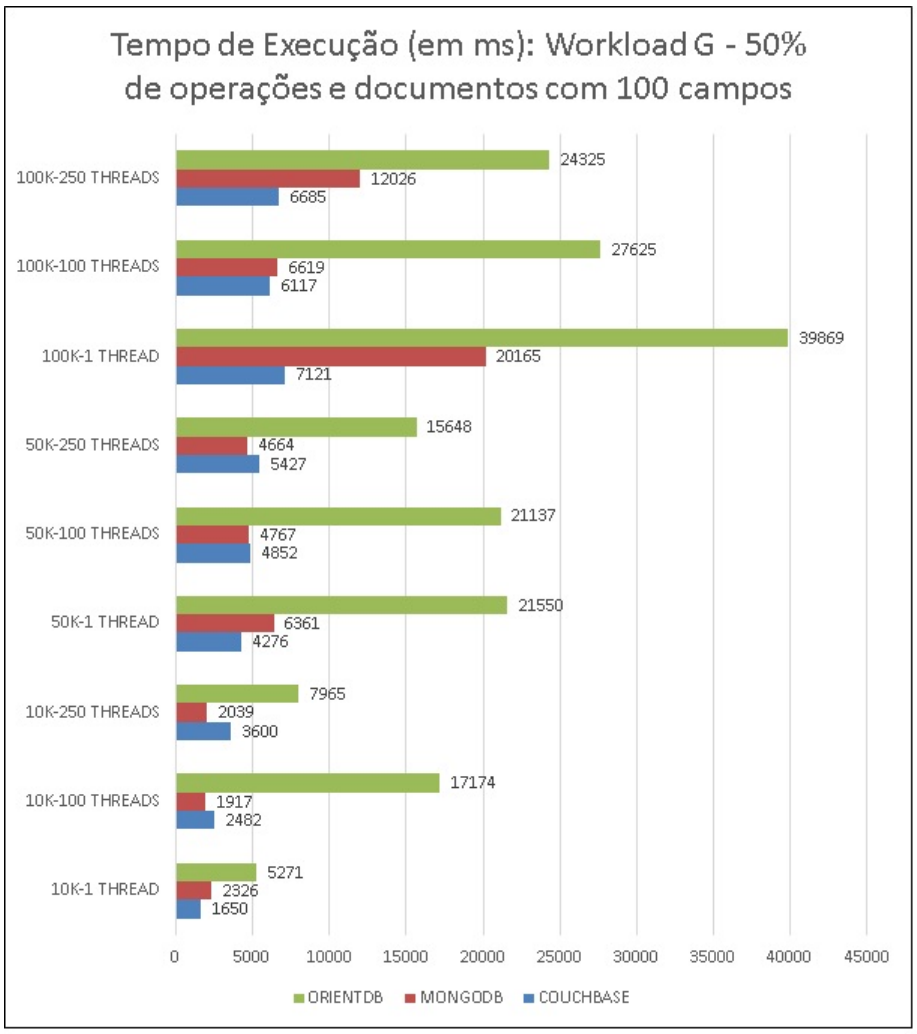
Figura 44 – Gráfico do tempo de execução apresentado com o *workload* G e com um total de operações de 20% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



Fonte: Elaborado pelo autor.

O quarto teste utilizando o *workload* G foi realizado com 10K, 50K e 100K de documentos compostos por 100 campos, e 1, 100 e 250 *threads*, utilizando como parâmetro 50% do total de documentos que compõem a base de dados como o número de operações a serem realizadas nos testes. Na Figura 45 pode-se verificar que, na x maioria dos testes, o sistema *Couchbase* foi o sistema que apresentou os tempos de execução mais baixos, e, no teste 100K- 1 *Thread*, menos da metade do tempo de execução do sistema *MongoDB*, que teve, em alguns casos, os melhores tempos de execução. Como o padrão, o sistema *OrientDB* apresentou os piores resultados, sendo, em algumas vezes (teste 10K-100 *Threds*), nove vezes mais lento do que o sistema que apresentou o tempo de execução mais baixo.

Figura 45 – Gráfico do tempo de execução apresentado com o *workload* G e com um total de operações de 50% em relação ao tamanho da base de dados e com documentos compostos por 100 campos cada.



Fonte: Elaborado pelo autor.

5.4 Avaliação do Desempenho dos Sistemas Testados

Comparados os desempenhos das ferramentas, pode-se, agora, avaliar qual das ferramentas apresentou o melhor desempenho geral, isto é, qual dos sistemas apresentou o menor tempo após a realização de todos os testes. De acordo com o levantamento de informações adquiridas através dos gráficos da seção anterior, pode-se observar que os sistemas *Couchbase* e *MongoDB* apresentaram um desempenho bastante semelhante, no tempo de execução em grande parte dos testes. Porém, o sistema *Couchbase* apresentou-se, no geral, com melhor desempenho do que o sistema *MongoDB*, podendo ser apontado como sendo o melhor sistema no requisito desempenho. Foi possível observar também, de acordo com os mesmos gráficos, que o sistema *OrientDB* apresentou o pior desempenho em quase todos os testes, com exceção de um caso, sendo o sistema que executou os testes no maior tempo dentre todos os sistemas. Um resumo do desempenho dos sistemas nos

testes, em forma de *ranking* é apresentado abaixo. Na Tabela 6 é apresentado o *ranking* dos sistemas nas operações de carga dos dados.

Tabela 6 – O resultado, em forma de *ranking* de colocação, do tempo de execução nas operações de carga de dados.

Campos do Documento	Colocação do Sistema	Sistema
10 campos	1º	*Couchbase*
	2º	*MongoDB*
	3º	*OrientDB*
100 campos	1º	*Couchbase*
	2º	*MongoDB*
	3º	*OrientDB*

Fonte: Elaborado pelo autor.

Na Tabela 7 é apresentado o *ranking* dos sistemas com os testes realizados com o *workload* A.

Tabela 7 – O resultado, em forma de *ranking* de colocação, do tempo de execução no *workload* A.

Quantidade de Operações	Campos do Documento	Colocação do Sistema	Sistema
20%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*MongoDB*
		2º	*Couchbase*
		3º	*OrientDB*
50%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*

Fonte: Elaborado pelo autor.

Na Tabela 8 é apresentado o *ranking* dos sistemas com os testes realizados com o *workload* B.

Na Tabela 9 é apresentado o *ranking* dos sistemas com os testes realizados com o *workload* C.

Na Tabela 10 é apresentado o *ranking* dos sistemas com os testes realizados com o *workload* F.

Na Tabela 11 é apresentado o *ranking* dos sistemas com os testes realizados com o *workload* G.

Na Tabela 12, é apresentado o número total de testes realizados no atual trabalho como sendo 264 testes, juntamente a colocação de cada sistema em relação ao seu

Tabela 8 – O resultado, em forma de *ranking* de colocação, do tempo de execução no *workload* B.

Quantidade de Operações	Campos do Documento	Colocação do Sistema	Sistema
20%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
50%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*

Fonte: Elaborado pelo autor.

Tabela 9 – O resultado, em forma de *ranking* de colocação, do tempo de execução no *workload* C.

Quantidade de Operações	Campos do Documento	Colocação do Sistema	Sistema
20%	10 campos	1º	*MongoDB*
		2º	*Couchbase*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
50%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*MongoDB*
		2º	*Couchbase*
		3º	*OrientDB*

Fonte: Elaborado pelo autor.

desempenho, levando-se em consideração o número de testes onde cada sistema apresentou o melhor tempo de execução, o segundo melhor tempo de execução e o terceiro tempo de execução. Foi possível observar que o sistema *Couchbase* apresentou o melhor tempo de execução em 172 das execuções e o sistema *MongoDB* apresentou o melhor tempo de execução nas 92 execuções restantes, tendo apresentado o melhor desempenho quase na metade das vezes se comparado ao número de casos apresentados pelo sistema *Couchbase*. Já o sistema *OrientDB*, como seu melhor resultado, apenas apresentou em 3 das execuções o segundo melhor desempenho, mas nunca apresentando o melhor desempenho nos testes, e, no restante dos testes, sempre apresentando o pior desempenho dentre todos os sistemas selecionados. Na Figura 46 é apresentado o tempo de execução total, em milissegundos (ms) de cada sistema nos testes executados. Foi possível observar que o tempo de execução total dos testes foi de 21.143.863 milissegundos, e que o tempo de execução do sistema

Tabela 10 – O resultado, em forma de *ranking* de colocação, do tempo de execução no *workload* F.

Quantidade de Operações	Campos do Documento	Colocação do Sistema	Sistema
20%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
50%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*

Fonte: Elaborado pelo autor.

Tabela 11 – RO resultado, em forma de *ranking* de colocação, do tempo de execução no *workload* G.

Quantidade de Operações	Campos do Documento	Colocação do Sistema	Sistema
20%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
50%	10 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*
	100 campos	1º	*Couchbase*
		2º	*MongoDB*
		3º	*OrientDB*

Fonte: Elaborado pelo autor.

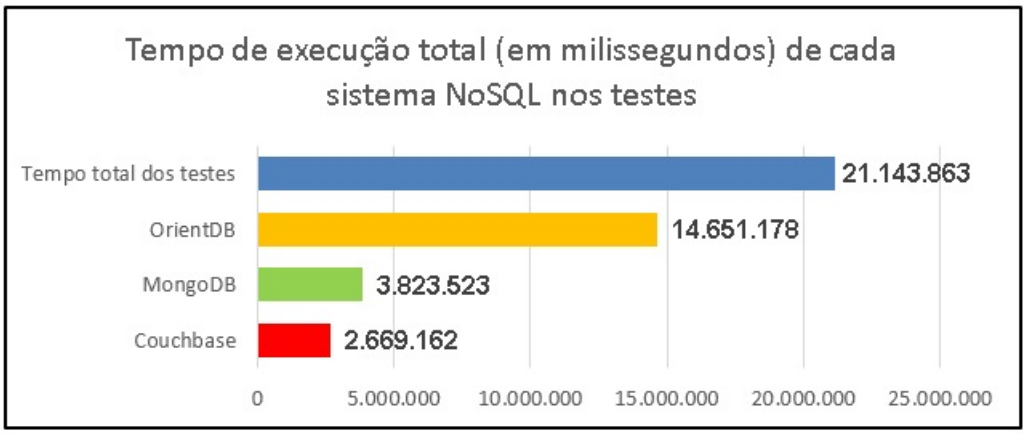
Tabela 12 – Número total de testes realizados e colocação dos sistemas por ordem de desempenho.

	1º Melhor	2º Melhor	3º Melhor
Couchbase	172	90	2
MongoDB	92	171	1
OrientDB	0	3	261
Número total de testes			264

Fonte: Elaborado pelo autor.

Couchbase foi o mais baixo dentre todos, tendo sido realizado em 2.669.162 milissegundos, de que o tempo total de execução do sistema *MongoDB* foi o segundo melhor, tendo sido realizado no tempo de 3.823.523 milissegundos e que o sistema *OrientDB* foi o sistema menos eficiente, tendo realizado os testes no tempo de 14.651.17 milissegundos. Por fim, na Figura 47 é apresentado, em porcentagem, o quanto cada sistema contribuiu para o tempo de execução total dos testes. Em relação ao tempo de execução total dos testes, o

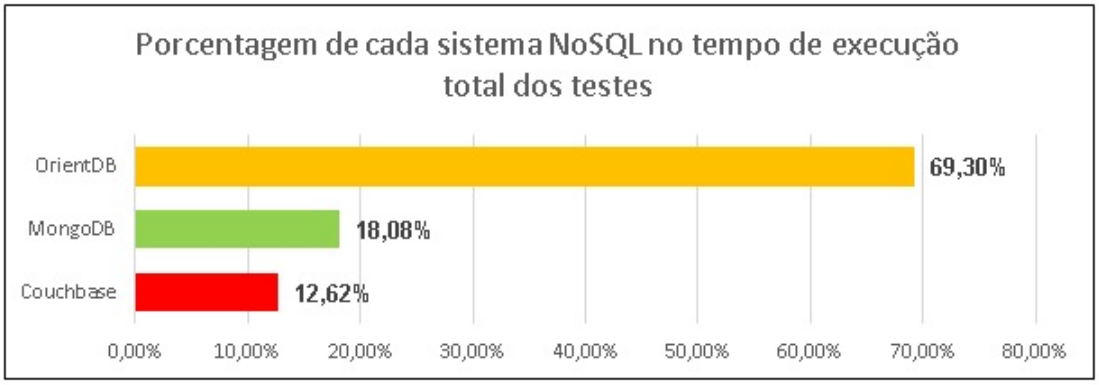
Figura 46 – Gráfico do tempo de execução total apresentado nos testes por cada sistema.



Fonte: Elaborado pelo autor.

tempo de execução do sistema *Couchbase* contribuiu para 12,62% do total, o tempo do sistema *MongoDB* contribuiu para 18,08% do tempo total e o tempo do sistema *OrientDB* contribuiu para 69,30% do tempo de execução total dos testes.

Figura 47 – Gráfico da porcentagem do tempo de execução de cada sistema em relação ao tempo de execução total dos testes.



Fonte: Elaborado pelo autor.

6 CONSIDERAÇÕES FINAIS

Neste capítulo, serão apresentadas as considerações finais sobre o atual trabalho, já realizando a projeção da estrutura para possíveis trabalhos futuros.

Como consequência do aumento gradativo do volume dos dados disponíveis, juntamente à mudança da estrutura dos mesmos, está sendo exigida uma maior complexidade para o eficiente gerenciamento destes dados e os sistemas relacionais, mesmo estabelecidos no mercado e sendo os sistemas mais utilizados pelas as organizações, apresentam algumas limitações para este gerenciamento. Os sistemas *NoSQL* se apresentam, de forma muito eficiente, como alternativa aos sistemas relacionais, visto que suas características são específicas para o alto desempenho e gerenciamento de grandes volumes de dados, também, permitindo que haja a distribuição destes dados, de forma nativa, em diversos servidores.

A realização dos testes do presente trabalho, de forma integral, foi realizada no sistema operacional *Linux Ubuntu Desktop* 12.04.5, que é gratuito e de código livre. O principal objetivo deste trabalho foi avaliar o desempenho de três sistemas de bancos de dados *NoSQL* orientados a documentos (*MongoDB*, *Couchbase* e *OrientDB*) por meio do *benchmark YCSB*.

Para tal avaliação, inicialmente, foram identificadas como métricas de avaliação, consideradas no trabalho, a quantidade de memória utilizada, o tempo de uso do processador, o espaço em disco utilizado por cada sistema em cada teste, o número de operações realizadas por segundo e o tempo de execução, mas acabou sendo utilizada no trabalho o tempo de execução das operações. Posteriormente, através da ferramenta de *benchmark YCSB*, foram executados alguns testes de comparação para levantamento de informações sobre o desempenho destes sistemas, visando efetuar a avaliação dos mesmos segundo a métrica definida. Mesmo que não tenham sido utilizadas bases de dados muito grandes, foi possível apresentar e avaliar, de maneira satisfatória, o desempenho de alguns sistemas *NoSQL* selecionados com a metodologia definida.

Foi possível concluir que dentre os três sistemas selecionados, apenas dois, o *Couchbase* e o *MongoDB*, apresentaram um desempenho satisfatório e, em grande parte dos testes, muito semelhante. O terceiro sistema, *OrientDB*, apresentou, com exceção de um caso, o pior tempo de execução em todos os testes, chegando, em alguns momentos, a apresentar um tempo de execução nove vezes maior do que o sistema com o melhor tempo de execução.

O sistema *Couchbase* apresentou o melhor desempenho na grande maioria dos testes, e, em alguns casos, apresentou uma superioridade quando tratando de operações que são executadas de forma concorrente (100 e 250 *threads*), realizando as operações

na metade do tempo do *MongoDB*. O sistema *MongoDB*, segundo melhor sistema do atual trabalho, em alguns testes, chegou a apresentar o melhor desempenho entre todos os demais, porém, no geral, o número de casos onde isso ocorreu foi menor do que em comparação com o sistema *Couchbase*.

Sendo assim, é possível considerar o sistema *Couchbase* como o melhor sistema dentro do ambiente definido no atual trabalho, o sistema *MongoDB* como o segundo melhor e o sistema *OrientDB* como o pior. Salienta-se que neste trabalho não foi realizado qualquer tipo de otimização dos sistemas, o que pode ter influenciado no desempenho de algum sistema.

6.1 Contribuições

As principais contribuições deste trabalho podem ser citadas como a realização de testes experimentais entre os sistemas *NoSQL* orientados a documentos *MongoDB*, *Couchbase* e *OrientDB*, sendo utilizadas, em cada sistema, bases de tamanhos distintos, compostos por documentos com diferentes quantidades de campos, juntamente a quantidade de acessos diversos, realizados com diferentes quantidades de *threads* e com distintos *workloads* que simulam distintos padrões de transações.

Foram, também, identificadas métricas de avaliação que podem ser utilizadas para o devido levantamento do desempenho dos sistemas selecionados e para a avaliação dos mesmos. Essa métricas podem ser utilizadas, tanto em outros sistemas *NoSQL*, quanto em sistemas relacionais.

Posteriormente, a comparação dos desempenhos de cada um dos sistemas selecionados realizada através da ferramenta de *benchmark YCSB*, que permitiu concluir que o uso da mesma é bastante adequado.

Por fim, foram comparados e avaliados os desempenhos de cada sistema, levando-se em consideração as métricas estabelecidas. Foi possível observar que o alto desempenho destes sistema se comprova, visto que nos testes executados, os tempos de execução são relativamente baixos.

6.2 Trabalhos Futuros

Devido aos sistemas *NoSQL* serem amplamente abordados em trabalhos e pesquisas recentemente e, também, devido as características destes sistemas, os mesmos tem ganhado bastante notoriedade.

Pelo fato de não terem sido utilizados sistemas do modelo relacional, considera-se importante a realização de testes que comparem o desempenho deste tipo frente a sistemas *NoSQL*, visto que a ferramenta *YCSB* dispõe de tal funcionalidade, porém, limitando os

testes a utilizar apenas os tempos de execução de cada sistema nas operações de leitura, inserção e atualização de dados.

Outro ponto que pode ser abordado futuramente, é a utilização do sistema *Elasticsearch*[1], outro sistema orientado a documentos, visto que a ferramenta *YCSB* possui suporte padrão para este sistema.

Também, pode ser adaptada a ferramenta *YCSB* para que esta possua suporte para a realização de testes dedicados ao sistema *CouchDB*, visto que este sistema é, dentre os sistemas *NoSQL* orientados a documentos, um dos mais utilizados pelas organizações.

Pode-se, também, realizar testes com os mesmos sistemas utilizados no atual trabalho mas em um ambiente distribuído, visando avaliar a o desempenho da escalabilidade (tolerância a particionamento) e elasticidade destes sistemas, seja através de máquinas virtuais ou de diferentes servidores físicos, pois estas são consideradas umas das principais características dos sistemas *NoSQL*.

Por fim, utilizar do mesmos sistemas do presente trabalho (*MongoDB*, *Couchbase* e *OrientDB*), porém realizando otimizações nestes sistemas para a verificação de alguma alteração do desempenho dos mesmos em relação ao desempenho apresentado no atual trabalho.

[1] https://www.elastic.co/products/elasticsearch

REFERÊNCIAS

ABRAMOVA, V.; BERNARDINO, J. Nosql databases: Mongodb vs cassandra. In: ACM. **Proceedings of the International C* Conference on Computer Science and Software Engineering**. [S.l.], 2013. p. 14–22.

ABRAMOVA, V.; BERNARDINO, J.; FURTADO, P. Experimental evaluation of nosql databases. **International Journal of Database Management Systems**, Academy & Industry Research Collaboration Center (AIRCC), v. 6, n. 3, p. 1, 2014.

ABRAMOVA, V.; BERNARDINO, J.; FURTADO, P. Which nosql database? a performance overview. **Open Journal of Databases (OJDB)**, RonPub, v. 1, n. 2, p. 17–24, 2014.

ARNAUDSJS. **GitHub - arnaudsjs/YCSB-couchdb-binding: Couchdb database interface for YCSB**. 2015. Disponível em: < https://github.com/arnaudsjs/YCSB-couchdb-binding >. Acesso em: 04 mar. 2016.

BRITO, R. W. Bancos de dados nosql x sgbds relacionais: análise comparativa. **Faculdade Farias Brito e Universidade de Fortaleza**, 2010.

BROWN, C. **NoSql Tips and Tricks: CAP Theorem Diagram for distribution**. 2011. Disponível em: < http://blog.nosqltips.com/2011/04/cap-diagram-for-distribution.html >. Acesso em: 14 set. 2015.

CELKO, J. **Joe Celko's complete guide to NoSQL: What every SQL professional needs to know about non-relational databases**. [S.l.]: Newnes, 2013.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking cloud serving systems with ycsb. In: ACM. **Proceedings of the 1st ACM symposium on Cloud computing**. [S.l.], 2010. p. 143–154.

COUCHBASE. **Caching layer — Couchbase Server 3.0/3.1**. 2015. Disponível em: < http://docs.couchbase.com/admin/admin/Concepts/concept-cacheLayer.html >. Acesso em: 04 mar. 2016.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM, v. 51, n. 1, p. 107–113, 2008.

DEMCHENKO, Y.; GROSSO, P.; LAAT, C. D.; MEMBREY, P. Addressing big data issues in scientific data infrastructure. In: IEEE. **Collaboration Technologies and Systems (CTS), 2013 International Conference on**. [S.l.], 2013. p. 48–55.

DEY, A.; FEKETE, A.; NAMBIAR, R.; ROHM, U. Ycsb+t: Benchmarking web-scale transactional databases. In: IEEE. **Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on**. [S.l.], 2014. p. 223–230.

DUDA, J. Business intelligence and nosql databases. **Information Systems in Management**, Katedra Informatyki Szkoła Główna Gospodarstwa Wiejskiego w Warszawie, v. 1, n. 1, p. 25–37, 2012.

ERB, B. **Concurrent Programming for Scalable Web Architectures**. Dissertação (Diploma Thesis) — Institute of Distributed Systems, Ulm University, April 2012. Disponível em: <http://www.benjamin-erb.de/thesis>.

FRIEDRICH, S.; WINGERATH, W.; GESSERT, F.; RITTER, N. Nosql oltp benchmarking: A survey. In: **GI-Jahrestagung**. [S.l.: s.n.], 2014. p. 693–704.

GARTNER. **Gartner Says Big Data Creates Big Jobs: 4.4 Million IT Jobs Globally to Support Big Data By 2015**. 2012. Disponível em: < http://www.gartner.com/newsroom/id/2207915>. Acesso em: 24 ago. 2015.

GARTNER. **What Is Big Data? - Gartner IT Glossary - Big Data**. 2015. Disponível em: <http://www.gartner.com/it-glossary/big-data>. Acesso em: 21 ago. 2015.

GILBERT, S.; LYNCH, N. A. Perspectives on the cap theorem. In: INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. [S.l.], 2012.

GOMES, C. C. **A sabedoria dos dados**. [S.l.]: Redação Link - O Estado de São Paulo, 2013. Disponível em: http://link.estadao.com.br/noticias/geral,a-sabedoria-dos-dados,10000032270. Acesso em: 13 jan. 2016.

IBM. **IBM Analytics – IT Business Intelligence - United States**. 2015. Disponível em: <http://www.ibm.com/smarterplanet/us/en/business_analytics/article/it_business_intelligence.html>. Acesso em: 03 set. 2015.

JATANA, N.; PURI, S.; AHUJA, M.; KATHURIA, I.; GOSAIN, D. A survey and comparison of relational and non-relational database. **International Journal of Engineering Research & Technology**, Citeseer, v. 1, n. 6, 2012.

KASHYAP, S.; ZAMWAR, S.; BHAVSAR, T.; SINGH, S. Benchmarking and analysis of nosql technologies. **Int J Emerg Technol Adv Eng**, Citeseer, v. 3, p. 422–426, 2013.

LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. Nosql no desenvolvimento de aplicações web colaborativas. **VIII Simpósio Brasileiro de Sistemas Colaborativos, Brasil**, 2011.

LOURENÇO, J. R.; CABRAL, B.; CARREIRO, P.; VIEIRA, M.; BERNARDINO, J. Choosing the right nosql database for the job: a quality attribute evaluation. **Journal of Big Data**, Springer, v. 2, n. 1, p. 1–26, 2015.

MARR, B. **Why only one of the 5 Vs of big data really matters | The Big Data Hub**. 2015. Disponível em: <http://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters/>. Acesso em: 12 out. 2015.

MONIRUZZAMAN, A.; HOSSAIN, S. A. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. **arXiv preprint arXiv:1307.0191**, 2013.

MONSON-HAEFEL, R. **97 things every software architect should know: collective wisdom from the experts**. [S.l.]: O'Reilly, 2009.

NAMBIAR, R.; POESS, M. Reinventing the tpc: From traditional to big data to internet of things. In: **Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things**. [S.l.]: Springer, 2015. p. 1–7.

NOSQL. **NOSQL Databases**. 2015. Disponível em: < http://nosql-database.org/>. Acesso em: 01 nov. 2015.

O'NEIL, P. E.; O'NEIL, E. J.; CHEN, X. The star schema benchmark (ssb). **Pat**, 2007.

PATIL, S.; POLTE, M.; REN, K.; TANTISIRIROJ, W.; XIAO, L.; LÓPEZ, J.; GIBSON, G.; FUCHS, A.; RINALDI, B. Ycsb++: benchmarking and performance debugging advanced features in scalable table stores. In: ACM. **Proceedings of the 2nd ACM Symposium on Cloud Computing**. [S.l.], 2011. p. 9.

PETRY, A. O berço do big data: a monumental abundância de dados, sua variedade e velocidade com que trafegam no universo digital estão revolucionando a civilização. **Revista Veja, São Paulo, ed**, v. 2321, p. 71–81, 2013.

PITCHUMANI, R.; FRANK, S.; MILLER, E. L. Realistic request arrival generation in storage benchmarks. In: IEEE. **Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on**. [S.l.], 2015. p. 1–10.

RABL, T.; GÓMEZ-VILLAMOR, S.; SADOGHI, M.; MUNTÉS-MULERO, V.; JACOBSEN, H.-A.; MANKOVSKII, S. Solving big data challenges for enterprise application performance management. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 5, n. 12, p. 1724–1735, 2012.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados-3**. [S.l.]: AMGH Editora, 2008.

REDMOND, E.; WILSON, J. R. **Seven databases in seven weeks: a guide to modern databases and the NoSQL movement**. [S.l.]: Pragmatic Bookshelf, 2012.

REEVE, A. Big data and nosql: The problem with relational databases. **Sep-2012.[Online]. Available at: http://infocus. emc. com/april_reeve/big-data-and-nosql-the-problem-with-relationaldatabases/.[Accessed: 21-Feb-2013]**, 2012.

SCHNEIDER, R. D. Hadoop for dummies special edition. **John Wiley&Sons Canada**, 2012.

SOUZA, V. C. O. de; SANTOS, M. V. C. dos. Amadurecimento, consolidação e performance de sgbds nosql-estudo comparativo. 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbsi/2015/033.pdf>.

SULLIVAN, D. **NoSQL for Mere Mortals**. [S.l.]: Addison-Wesley Professional, 2015.

TIWARI, S. **Professional NoSQL**. [S.l.]: John Wiley & Sons, 2011.

TPC-C. **TPC-C - Homepage**. 2015. Disponível em: < http://www.tpc.org/tpcc/ >. Acesso em: 08 ago. 2015.

TRUICA, C.; BOICEA, A.; RADULESCU, F. **Asynchronous Replication in Microsoft SQL Server, PostgreSQL and MySQL, The 2013 International Association for Cyber Science and Engineering (CYBERSE2013)**. [S.l.]: DEStech Publications Inc, 2013.

TRUICA, C.-O.; RADULESCU, F.; BOICEA, A.; BUCUR, I. Performance evaluation for crud operations in asynchronously replicated document oriented database. In: IEEE. **Control Systems and Computer Science (CSCS), 2015 20th International Conference on**. [S.l.], 2015. p. 191–196.

UBUNTU. **Implementing New Workloads · brianfrankcooper/YCSB Wiki · GitHub**. 2010. Disponível em: < https://github.com/brianfrankcooper/YCSB/wiki/Implementing-New-Workloads >. Acesso em: 01 fev. 2016.

UBUNTU. **ServerFaq - Community Help Wiki**. 2012. Disponível em: < https://github.com/arnaudsjs/YCSB-couchdb-binding >. Acesso em: 24 mar. 2016.

UBUNTU. **YCSB/mongodb at master · brianfrankcooper/YCSB · GitHub**. 2016. Disponível em: < https://github.com/brianfrankcooper/YCSB/tree/master/mongodb>. Acesso em: 01 fev. 2016. Disponível em: <https://github.com/brianfrankcooper/YCSB/tree/master/mongodb>.

VANDERLINDE, T. **Checklist para auxiliar na definição da arquitetura de banco de dados** . 2013. Disponível em <http://www.ceavi.udesc.br/arquivos/id_submenu/787/tiago_vanderlinde_versao_final_.pdf>. Acesso em 20 nov. 2015.

VIEIRA, M. R.; FIGUEIREDO, J. M. d.; LIBERATTI, G.; VIEBRANTZ, A. F. M. Bancos de dados nosql: conceitos, ferramentas, linguagens e estudos de casos no contexto de big data. **Simpósio Brasileiro de Bancos de Dados**, 2012.

WHITE, T. **Hadoop: The definitive guide**. [S.l.]: "O'Reilly Media, Inc.", 2012.

Apêndices

APÊNDICE A – COMANDOS PARA EXECUÇÃO DOS TESTES COM O SISTEMA *MONGODB*

A.1 Exemplos de Comandos de Carga dos Dados

```
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=10000 -threads 1
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=50000 -threads 100
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=100000 -threads 250
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=500000 -threads 250
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=1000000 -threads 1
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=10000 -p fieldcount=100 -threads 100
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=50000 -p fieldcount=100 -threads 250
$   ./bin/ycsb load mongodb -s -P workloads/workloada -p
   recordcount=100000 -p fieldcount=100 -threads 1
```

A.2 Exemplos de Comandos de Execução dos *Workloads*

```
$   ./bin/ycsb run mongodb -s -P workloads/workloada -p
   operationcount=2000 -threads 1
$   ./bin/ycsb run mongodb -s -P workloads/workloadb -p
   operationcount=2000 -threads 100
$   ./bin/ycsb run mongodb -s -P workloads/workloadc -p
   operationcount=2000 -threads 250
$   ./bin/ycsb run mongodb -s -P workloads/workloadf -p
   operationcount=2000 -threads 250
$   ./bin/ycsb run mongodb -s -P workloads/workloadg -p
   operationcount=2000 -threads 250
```

APÊNDICE B – COMANDOS PARA EXECUÇÃO DOS TESTES COM O SISTEMA *COUCHBASE*

B.1 Exemplos de Comandos de Carga dos Dados

```
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=10000 -threads 1
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=50000 -threads 100
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=100000 -threads 250
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=500000 -threads 250
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=1000000 -threads 1
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=10000 -p fieldcount=100 -threads 100
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=50000 -p fieldcount=100 -threads 250
$   ./bin/ycsb load couchbase -s -P workloads/workloada -p
   recordcount=100000 -p fieldcount=100 -threads 1
```

B.2 Exemplos de Comandos de Execução dos *Workloads*

```
$   ./bin/ycsb run couchbase -s -P workloads/workloada -p
   operationcount=2000 -threads 1
$   ./bin/ycsb run couchbase -s -P workloads/workloadb -p
   operationcount=2000 -threads 100
$   ./bin/ycsb run couchbase -s -P workloads/workloadc -p
   operationcount=2000 -threads 250
$   ./bin/ycsb run couchbase -s -P workloads/workloadf -p
   operationcount=2000 -threads 250
$   ./bin/ycsb run couchbase -s -P workloads/workloadg -p
   operationcount=2000 -threads 250
```

APÊNDICE C – COMANDOS PARA EXECUÇÃO DOS TESTES COM O SISTEMA *ORIENTDB*

C.1 Exemplos de Comandos de Carga dos Dados

```
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=10000 -threads 1 -p orientdb.url=remote
   :0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
   password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=50000 -threads 100 -p orientdb.url=remote
   :0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
   password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=100000 -threads 250 -p orientdb.url=remote
   :0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
   password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=500000 -threads 250 -p orientdb.url=remote
   :0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
   password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=1000000 -threads 1 -p orientdb.url=remote
   :0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
   password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=10000 -p fieldcount=100 -threads 100 -p
   orientdb.url=remote:0.0.0.0:2424/ycsb -p orientdb.user=
   root -p orientdb.password=teste -p orientdb.remote.
   storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=50000 -p fieldcount=100 -threads 250 -p
   orientdb.url=remote:0.0.0.0:2424/ycsb -p orientdb.user=
   root -p orientdb.password=teste -p orientdb.remote.
   storagetype=remote
$   ./bin/ycsb load orientdb -s -P workloads/workloada -p
   recordcount=100000 -p fieldcount=100 -threads 1 -p
   orientdb.url=remote:0.0.0.0:2424/ycsb -p orientdb.user=
```

```
root -p orientdb.password=teste -p orientdb.remote.
storagetype=remote
```

C.2 Exemplos de Comandos de Execução dos *Workloads*

```
$   ./bin/ycsb run orientdb -s -P workloads/workloada -p
operationcount=2000 -threads 1 -p orientdb.url=remote
:0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb run orientdb -s -P workloads/workloadb -p
operationcount=2000 -threads 100 -p orientdb.url=remote
:0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb run orientdb -s -P workloads/workloadc -p
operationcount=2000 -threads 250 -p orientdb.url=remote
:0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb run orientdb -s -P workloads/workloadf -p
operationcount=2000 -threads 250 -p orientdb.url=remote
:0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
password=teste -p orientdb.remote.storagetype=remote
$   ./bin/ycsb run orientdb -s -P workloads/workloadg -p
operationcount=2000 -threads 250 -p orientdb.url=remote
:0.0.0.0:2424/ycsb -p orientdb.user=root -p orientdb.
password=teste -p orientdb.remote.storagetype=remote
```

APÊNDICE D – *SCRIPT - ORIENTDB.SH* G

```
#!/bin/sh
# OrientDB service script
# Copyright (c) Orient Technologies LTD (http://www.
   orientechnologies.com)
# chkconfig: 2345 20 80
# description: OrientDb init script
# processname: orientdb.sh
# You have to SET the OrientDB installation directory here
#ORIENTDB\_DIR="YOUR\_ORIENTDB\_INSTALLATION\_PATH"
#ORIENTDB\_USER="USER\_YOU\_WANT\_ORIENTDB\_RUN\_WITH"
ORIENTDB\_DIR="/opt/orientdb-2.1.3"
ORIENTDB\_USER="root"

usage() {
        echo "Usage: `basename $0`: <start|stop|status>"
        exit 1
}

start() {
        status
        if [ $PID -gt 0 ]
        then
                echo "OrientDB server daemon was already
                   started. PID: $PID"
                return $PID
        fi
        echo "Starting OrientDB server daemon..."
        cd "$ORIENTDB\_DIR/bin"
#       su $ORIENTDB\_USER -c "cd \"$ORIENTDB\_DIR/bin\"; /
   usr/bin/nohup ./server.sh 1>../log/orientdb.log 2>../log/
   orientdb.err &"
sudo -u $ORIENTDB\_USER sh -c "cd \"$ORIENTDB\_DIR/bin\"; /
   usr/bin/nohup ./server.sh 1>../log/orientdb.log 2>../log/
   orientdb.err &"
}
```

```
stop() {
        status
        if [ $PID -eq 0 ]
        then
                echo "OrientDB server daemon is already not
                   running"
                return 0
        fi
        echo "Stopping OrientDB server daemon..."
        cd "$ORIENTDB\_DIR/bin"
#       su $ORIENTDB\_USER -c "cd \"$ORIENTDB\_DIR/bin\"; /
   usr/bin/nohup ./shutdown.sh 1>>../log/orientdb.log 2>>../
   log/orientdb.err &"
sudo -u $ORIENTDB\_USER sh -c "cd \"$ORIENTDB\_DIR/bin\"; /
   usr/bin/nohup ./shutdown.sh 1>>../log/orientdb.log 2>>../
   log/orientdb.err &"
}

status() {
        PID=‘ps -ef | grep ’orientdb.www.path’ | grep java |
           grep -v grep | awk ’{print $2}’‘
        if [ "x$PID" = "x" ]
        then
                PID=0
        fi
        # if PID is greater than 0 then OrientDB is running,
           else it is not
        return $PID
}

if [ "x$1" = "xstart" ] \
then
        start
        exit 0
fi
if [ "x$1" = "xstop" ]
then
        stop
        exit 0
```

```
fi
if [ "x$1" = "xstatus" ]
then
        status
        if [ $PID -gt 0 ]
        then
                echo "OrientDB server daemon is running with
                   PID: $PID"
        else
                echo "OrientDB server daemon is NOT running"
        fi
        exit $PID
fi
usage
```

APÊNDICE E – *SCRIPT* - *WORKLOAD* G

```
#Copyright (c) 2010 Yahoo! Inc. All rights reserved.
#Licensed under the Apache License, Version 2.0 (the "License
   "); #you may not use this file except in compliance with
   the License. #You  may obtain a copy of the License at
   http://www.apache.org/licenses/LICENSE-2.0

#Unless required by applicable law or agreed to in writing, #
   software distributed under the License is distributed on
   an "AS IS" BASIS,
#WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
    or implied.
#See the License for the specific language governing
   permissions and limitations under the License.
#See accompanying LICENSE file.

#Yahoo! Cloud System Benchmark
#Workload G: Update mostly workload
#Application example: user profile cache, where profiles are
   constructed elsewhere (e.g., Hadoop)
#Read/update ratio: 5/95
#Default data size: 1 KB records (10 fields, 100 bytes each,
   plus key)
#Request distribution: zipfian

recordcount=10000
operationcount=5000
workload=com.yahoo.ycsb.workloads.CoreWorkload

readallfields=true

readproportion=0.05
updateproportion=0.95
scanproportion=0
insertproportion=0

requestdistribution=zipfian
```

APÊNDICE F – EXEMPLO DE REGISTRO GERADO PELA FERRAMENTA *YCSB*


```
{
   "id": "usertable:user6473314962453170 32",
   "field1": "?'v.4b+\"0(3\"4Om.Va%.v2,6$R%!#x9T+=*~5Ak3E3(
      .5+b9V/(\\m)Nu\".r:<f?P'6.l+>*42$ &<*/:$@3+'|+.&+Lm;R7
      , 5",
   "field0": "\"%t*!>0U9<N+ Oa/On#<x'!,),&9#n6'n#Fm=Bw6Tm\"*v
      &/j4])%,j2*45Qi+X/$N{$8 (H-/$85@k&Y+4<&,K?, b5B{:Fk
      /?<,",
   "field7": "\"!d\",4/%\"5-f,148\\5&+d*N{.H'0\",?8&1)|0Zi>=
      h9Vg6^e%Vg;6b=68!-$67~7\\a0B18M?$\\-&_a.>h&U-/Mi78j-(8$
      <\"$Cg+",
   "field6": "6F!8[k $8 &6?N/,\\)$9j**l%B-:F=22*8O;7%$!-x
      ,!.<3l1\\c:So,\\g;-h;Ki>/:.& 1\"v&2v3Yq>_) [m: p&@#.Nk
      **|9!d/",
   "field9": " X'-N- ?b7#,$1l>!,#X;5Fa:.v+Y3*@k:Zg 0488b-5t.
      Kw&#,-P9<,0)R!-Pe22j(-8(-<4?z:\\u23~\"P?9W+4N)??
      <\".-#0=",
   "field8": "5L-2Ky5I#$J)/.& %p:=,1G-0&204\"9'l#X36G)%58&[%0
      W-\"<49J#*T}/8~)06$7|!$~2I}45\"-*h?58%Ng,D39
      ^#(.*=<:-3~:",
   "field3": " >,57&<2*-Oo\"@3 4\"%\"n&9v10.-A?/3n+1r,%>)
      Va79x>3<.N/0P- 6d?(b86|;Xm;Es'Uk8A}/$h$,n23$12‘'A}4Xg#5
      r/+2-",
   "field2": "!Mi&0d#Hw2Tm&G%4Qy!4‘6_+ 34 6$94p+@++P
      %5'*#=>=': 36<Ee'S+8@7&Ms:I%=Ua.: -Ns?Ra*<0-%z7K#)Zq+Q
      +0Zq2^m2",
   "field5": ".Zu/4>03$1$n!^w#,f,Z!)$ /No>7p)T?(+>0@11S{?Us
      +\\3\"Go*Zm0>&1% 'Oq\")x-\\c/-$5)r.C'#Fo*3j<P#9V;$C!(N)
      4-<!",
   "field4": "!O3%Xy,\\=%Em8W'(-<.Ig;Z'>C;!]/?Zo&.82G+(Cs8Q
      -+(<6Ve?/h0L};Hi'!v2<r1*8,Ce%+2%A?,-<1G=,P=$00&*n2*b5K'
      $"
}
```

Anexos

ANEXO A – INSTALAÇÃO DAS FERRAMENTAS E SISTEMAS DO TRABALHO

Neste capítulo serão apresentados os procedimentos de instalação ou configuração dos sistemas e ferramentas necessários para a realização dos testes definidos neste trabalho. Na seção A.1 será apresentado o passo a passo para a instalação dos sistemas e ferramentas necessárias.

A.1 Preparação do sistema e ferramentas

Nesta seção serão apresentados os passos para a instalação e configuração dos sistemas *NoSQL* selecionados para compor o trabalho. Na subseção A.1.1, será apresentado o sistema operacional. Na subseção A.1.2, serão apresentados os passos para o *download*, instalação e configuração dos sistemas adicionais necessários para o correto funcionamento da ferramenta *YCSB*. Na subseção A.1.3, serão apresentados os passos para o *download* e instalação da ferramenta de *benchmark YCSB*. Finalmente, na subseção A.1.4, serão apresentados os passos para o *download*, instalação e configuração dos SGBD *NoSQL* selecionados.

A.1.1 Instalação e Configuração do Sistema Operacional

Como apresentado anteriormente, o sistema operacional que servirá como base para a realização do trabalho será o *Ubuntu Desktop* 12.04.5. Conforme reportado pela página de internet do *Ubuntu*[1], há a necessidade de excluir a lista de atualizações do sistema, localizada no diretório "/var/lib/apt/lists", devido a existência de um *bug*, por se tratar de um sistema de versão antiga. Também, conforme orientado na mesma página, é possível corrigir este problema através da execução dos seguintes comandos:

```
$ sudo rm -fr /var/lib/apt/lists
$ sudo apt-get update
```

Através do primeiro comando, é realizada a exclusão do diretório que possui o registro da lista de atualizações do sistema, fazendo com que seja representado que o mesmo não tenha recebido nenhuma atualização. Através do segundo comando, é realizada a atualização do sistema, sendo feita a comunicação com os repositórios remotos da *Canonical*[2], responsável por manter as versões dos aplicativos desenvolvidos para o Ubuntu. Após a realização destes comandos, o sistema estará atualizado e pronto para ter os demais sistemas e ferramentas instalados e configurados.

[1] https://bugs.launchpad.net/ubuntu/+source/apt-setup/+bug/1434699
[2] http://www.canonical.com/

A.1.2 Instalação e Configuração dos Sistemas Adicionais

Após o sistema operacional ter sido devidamente atualizado, pode-se dar sequência a instalção dos demais programas, iniciando pelos que são necessários para a correta instalação, montagem e funcionamento da ferramenta *YCSB*. Tais programas, já anteriormente apresentados no capítulo 3, são o *Java*[3] e o *Maven*[4].

Primeiramente, será realizado o download do pacote *python-software-properties* que é responsável por prover e permitir a utilização do comando "*add-apt-repository*", que será necessário para instalar um pacote por meio dos repositórios *PPA* (acrônimo, em língua inglesa, das iniciais de *Personal Package Archives*), sendo, então, necessário para a posterior instalação do *Java*. Um repositório *PPA* é um servidor na internet onde se encontram os programas que não estão nos repositórios oficias de uma distribuição, que, no caso, não estão nos servidores da *Canonical*. Abaixo, é apresentado o comando para a instalação do pacote *python-software-properties*:

```
$ sudo apt-get install -y python-software-properties
```

Após a instalação do pacote *python-software-properties*, poderá ser realizado o *download* e a instalação do *Java*, sendo executados através dos comandos abaixo:

```
$ java -version
$ sudo add-apt-repository -y ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install -y oracle-java8-installer
$ sudo apt-get install -y oracle-java8-set-default
```

Com o primeiro comando, verifica-se o *Java* já encontra-se instalado e, caso positivo, a versão do mesmo. Com o segundo comando, adiciona-se o repositório *PPA* que disponibiliza o *Java* à lista de repositórios do *Ubuntu*. Com o terceiro comando, é realizado o *download* e a instalação do *Java*, no caso, é realizada a instalação da versão 8 do programa. Durante o processo de instalação, serão apresentadas algumas telas de confirmação para aceitar os termos de uso do *Java*, conforme as Figura 48 e Figura 49. O quarto e último comando serve para adicionar e configurar as variáveis do *Java* às variáveis de ambiente do sistema operacional.

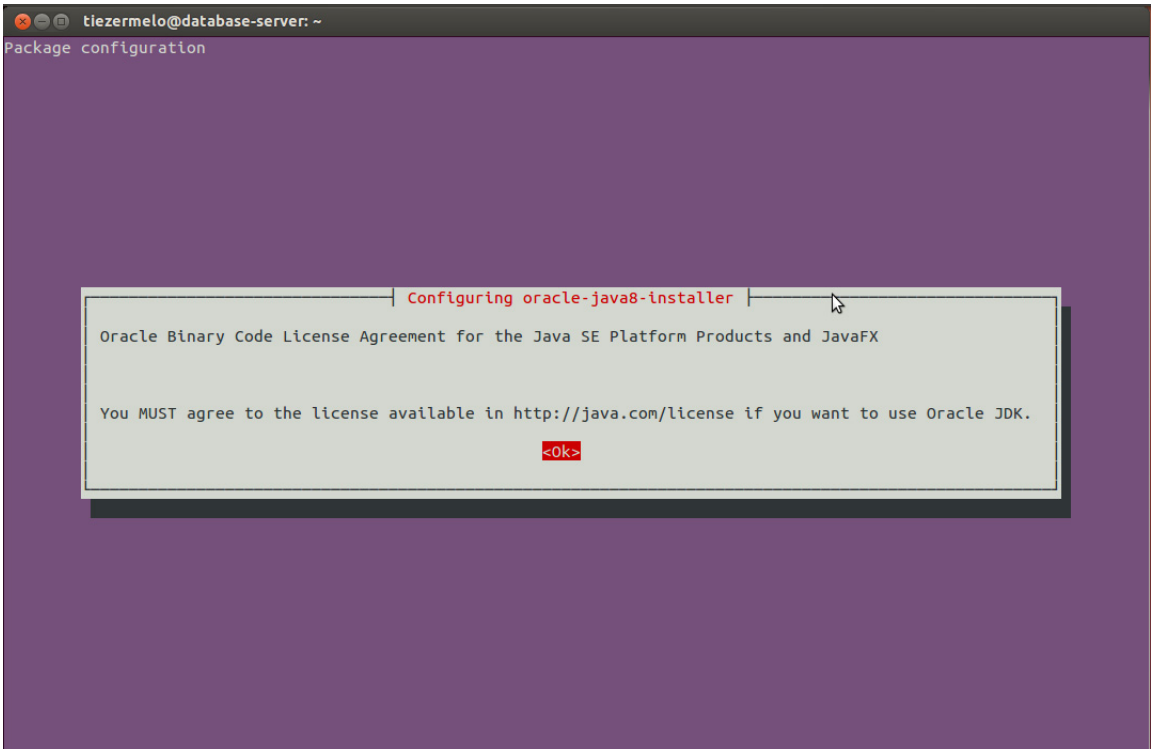
Na tela anterior, basta selecionar a opção "*OK*"pressionar a tecla "*ENTER*", onde será apresentada a tela abaixo:

Na segunda tela, basta selecionar a opção "*YES*"e pressionar a tecla "*ENTER*".

O segundo programa a ser instalado será o *Maven*. Segundo *Github*[5] para o correto funcionamento da ferramenta *YCSB* é requerido que seja utilizado o programa

[3] http://java.com/
[4] https://maven.apache.org/
[5] https://github.com/brianfrankcooper/YCSB/blob/master/README.md

Figura 48 – Primeira tela de aceitação do termo de uso apresentada durante a instalação do Java.



Fonte: Print Screen da tela de aceitação de termo de uso do Java.

Maven3[6] e, caso seja utilizado o *Maven2*, poderão ocorrer alguns erros, conforme mencionado em *Github*[7]. Já, conforme *Apache Maven*(https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html) o *Maven* é uma ferramenta *Java*, por isso, é necessário que o *Java* seja anteriormente instalado para o seu correto funcionamento.

```
$ sudo apt-get install maven
```

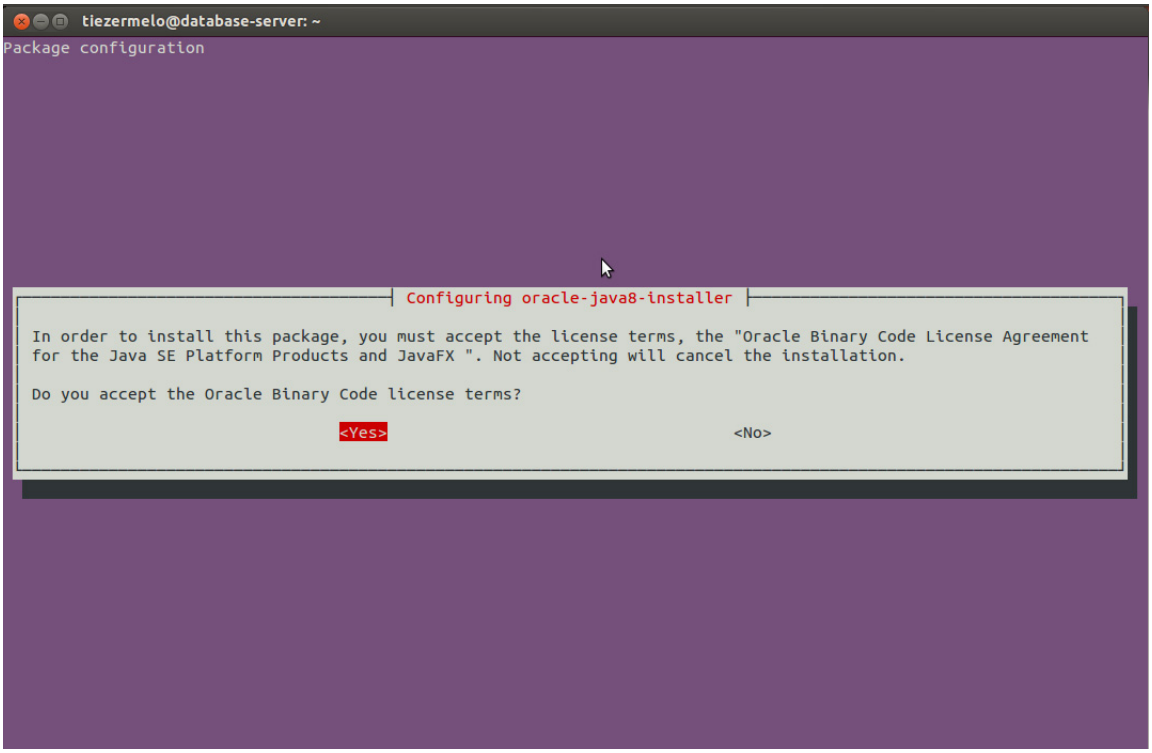
O comando acima efetua o *download* e, automaticamente, instalação deste programa. Agora, é necessário configurar as variáveis de ambiente do programa no sistema operacional, para o correto funcionamento do programa. Tal processo pode ser realizado através dos comandos abaixo:

```
$ export M3_HOME=/usr/share/maven
$ PATH=$M3_HOME/bin:$PATH
```

Após a instalação dos programas *Java* e *Maven*, o próximo passo será a instalação da ferramenta de *benchmark YCSB*.

[6] https://maven.apache.org/download.cgi
[7] https://github.com/brianfrankcooper/YCSB/issues/406

Figura 49 – Segunda tela de aceitação do termo de uso apresentada durante a instalação do *Java*.



Fonte: *Print Screen* da tela de aceitação de termo de uso do *Java*.

A.1.3 Instalação e Configuração da Ferramenta de *Benchmark*

Nesta seção, será realizado o *download* e a instalação da ferramenta de *benchmark YCSB*. Esta ferramenta está disponível no site do *Github*[8], mais especificamente no diretório de *Brian Frank Cooper*[9], um dos autores utilizados como referência neste trabalho. Poderão ser seguidos dois passos para o *download* da ferramenta, que dependerá do intuito do executante ou interessado. O primeiro passo, demonstrado logo abaixo, realizará o download da ferramenta pronta para o uso, com aproximadamente 313 *Megabytes*, já contendo todas as versões dos sistemas disponíveis, e não havendo a necessidade de efetuar a montagem da mesma.

```
$ cd ~
$ sudo wget https://github.com/brianfrankcooper/YCSB/
   releases/download/0.8.0/ycsb-0.8.0.tar.gz
$ tar xfvz ycsb-0.8.0.tar.gz
$ sudo mv ycsb-0.8.0 ycsb
$ cd ycsb
```

Através dos comando acima apresentados, considera-se que a ferramenta estará localizada no diretório "*home*"do usuário, no exemplo deste trabalho, no diretório "/ho-

[8] https://github.com/
[9] https://github.com/brianfrankcooper/YCSB/

me/tiezermelo/ycsb". O segundo caso, apresentado logo abaixo, deverá ser seguido caso haja a necessidade de realizar modificações da ferramenta ou se há o interesse de montar a ferramenta apenas com os sistemas de interesse, sendo necessário que sejam executados os seguintes comandos:

```
$ cd ~
$ sudo wget https://github.com/brianfrankcooper/YCSB/
   archive/0.8.0.tar.gz
$ tar xfvz YCSB-0.8.0.tar.gz
$ sudo mv YCSB-0.8.0 ycsb
$ cd ycsb
```

O primeiro comando aponta para o diretório *home* do usuário. O segundo comando executa a operação de clonar os arquivos do diretório remoto do *Github* para a unidade local selecionada. Após ter sido clonada, a mesma precisa ser montada, para que seja possível executá-la posteriormente. Salienta-se que há duas maneiras de realizar a montagem da ferramenta: a primeira, é realizando a montagem com todos os sistemas *NoSQL* suportados pela ferramenta (a lista completa dos sistemas pode ser encontrada na página de internet https://github.com/brianfrankcooper/YCSB/), ou, a segunda maneira, é realizando a montagem da ferramenta apenas com os sistemas *NoSQL* de interesse. A opção de montagem desejada pode ser realizada através dos comandos a seguir:

```
$ cd ~
$ cd YCSB
$ sudo mvn clean package
```

Através dos comandos acima, a ferramenta *YCSB* é montada com todos os sistemas suportados, conforme mencionado anteriormente. A seguir, os comandos necessários para a montagem do *Maven* apenas para os sistemas de interesse:

```
$ cd ~
$ cd ycsb
$ mvn -pl com.yahoo.ycsb:mongodb-binding -am clean package
$ mvn -pl com.yahoo.ycsb:couchbase-binding -am clean
   package
$ mvn -pl com.yahoo.ycsb:orientdb-binding -am clean package
```

Através dos comandos acima, conforme informado anteriormente, a ferramenta *YCSB* é montada apenas com os sistemas de interesse, que, no caso, são o *MongoDB*, o *Couchbase* e o *OrientDB*. A ferramenta de testes já está pronta para a execução dos testes, restando, apenas, a instalação e configuração dos SGBD *NoSQL*.

A.1.4 Instalação e Configuração dos Sistemas NoSQL

O último passo para a preparação do ambiente antes da realização dos testes é a instalação dos sistemas *NoSQL* de interesse. No atual trabalho, como já mencionado, foram selecionados os sistemas *MongoDB*, o *Couchbase* e o *OrientDB*. O *download* e a instalação destes sistemas, será apresentado a seguir.

Inicialmente, será instalado o *MongoDB*, conforme seguindo os procedimentos apresentados no site do próprio *MongoDB*(https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/):

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com
   :80 --recv 7F0CEB10
$ sudo echo ‘‘deb http://downloads-distro.mongodb.org/repo/
   ubuntu-upstart dist 10gen’’ | sudo tee /etc/apt/sources.
   list.d/mongodb.list
$ sudo apt-get update
$ sudo apt-get install mongodb-10gen=2.4.6
```

O segundo SGBD *NoSQL* a ser instalado é o *Couchbase* e este procedimento será realizado seguindo-se os passos a seguir:

```
$ sudo wget ‘‘http://packages.couchbase.com/releases/3.0.1/
   couchbase-server-community_3.0.1-ubuntu12.04_amd64.deb’’
$ sudo mv couchbase-server-community_3.0.1-ubuntu12.04_amd6.
   deb couchbase-3.0.1.deb
$ sudo dpkg -i couchbase-3.0.1.deb
```

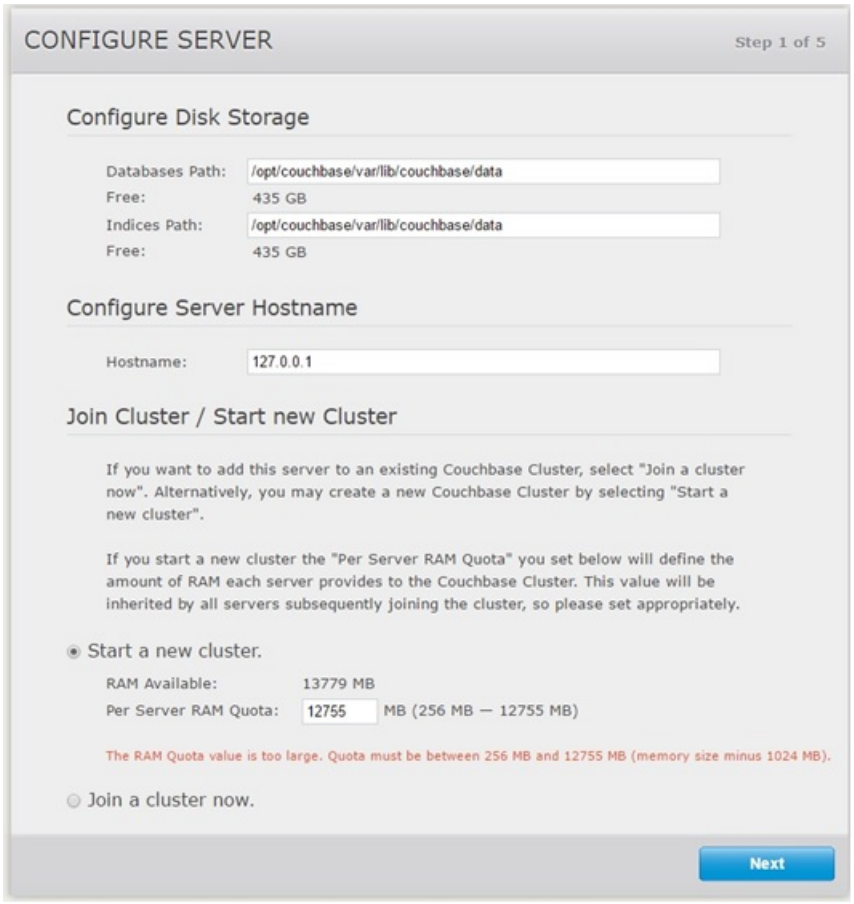
Após a execução do terceiro comando, o sistema será automaticamente instalado no diretório "*/opt/couchbase*". Agora, será necessário criar a base de dados que será destinada ao teste. A ferramenta "*YCSB*", por padrão, utiliza da base de dados chamada "*default*". Como sugestão, a criação da base de dados pode ser realizada acessando o *IP* do servidor, juntamente com a porta de conexão 8091, através de um navegadores de internet. No caso deste trabalho, o *IP* do servidor é http://192.168.0.19:8091. Os procedimentos para a criação da base de dados serão apresentados a seguir.

A Figura 50 apresenta a tela de configuração do servidor do *Couchabse*. Nessa tela deverão ser apontados o diretório do *data bucket*, o nome do servidor e se deseja-se integrar o servidor a um *cluster* já existente ou criar um novo *cluster*.

A Figura 51 apresenta a tela com alguns exemplos para demonstração de *data buckets* (*beer sample* e *gamesin-sample*). No caso do trabalho, nenhum dos dois exemplos foram selecionados, sendo apenas clicado o botão "*Next*".

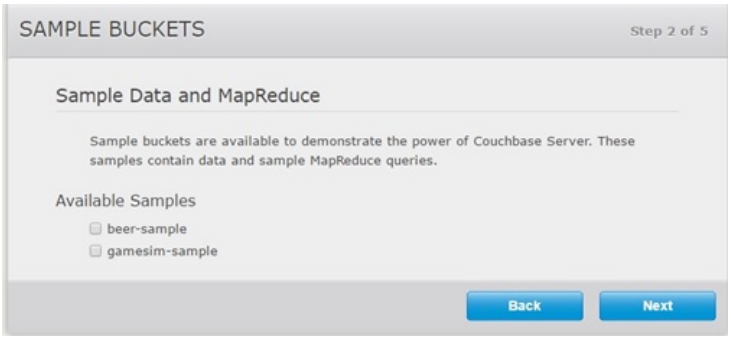
A Figura 52 apresenta a tela para configuração do *data bucket*, sendo necessário

Figura 50 – Tela de configuração do servidor *Couchbase*.



Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

Figura 51 – Tela para seleção de algum dos exemplos de *data buckets* do sistema *Couchbase*.

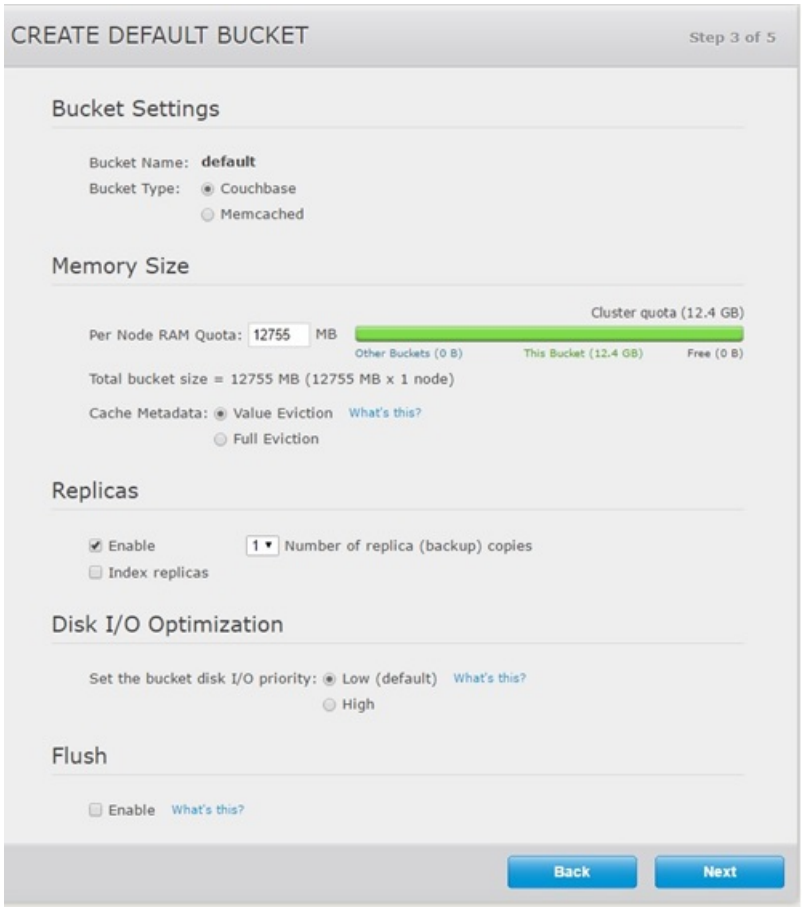


Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

selecionar a configuração do *bucket* entre *Couchbase* e *Memcached*, o tamanho da memória a ser utilizado pelo *data bucket*, o número de réplicas, a otimização do disco rígido e o *Flush*. Para o trabalho, as configurações selecionadas foram as apresentadas como preenchidas e selecionadas na figura.

Na Figura 53, é apresentada a tela para recebimento de notificações referentes

Figura 52 – Tela para especificação das propriedades para criação do *data bucket* no sistema *Couchbase*.



Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

a atualizações e. Não foram preenchidos nenhum dos campos da tela, sendo apenas pressionado o botão "*Next*"no canto inferior direito da tela.

NaFigura 54, será solicitado que sejam preenchidas as informações de usuário e senha utilizados para acesso ao *data bucket*.

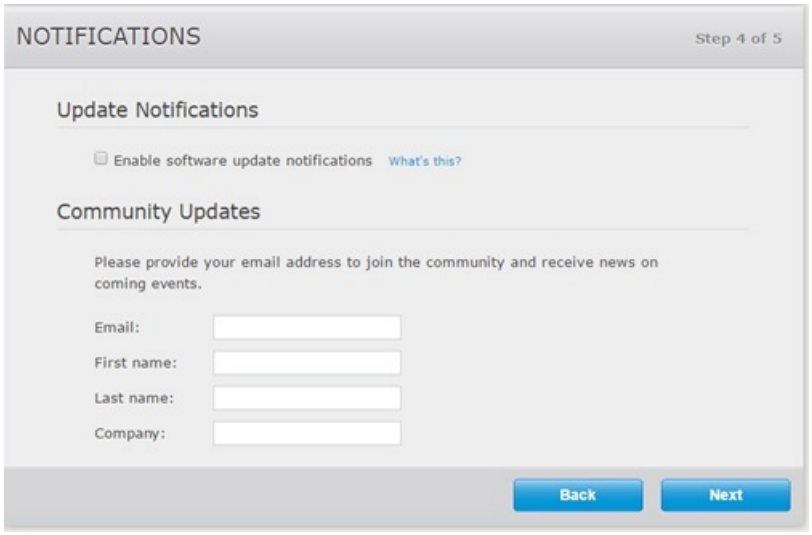
Na Figura 55, é apresentada a tela que representa os servidores que compõem o *cluster* do sistema do trabalho.

Na Figura 56 é apresentada a tela dos *data buckets* contidos no servidor do trabalho. No exemplo da figura, é apresentado o *data bucket* "*default*"que será utilizado nos testes referentes ao *Couchbase*.

O terceiro e último SGBD a ser instalado é o *OrientDB* e os procedimentos de instalação e configuração do mesmo serão realizado através dos passos abaixo:

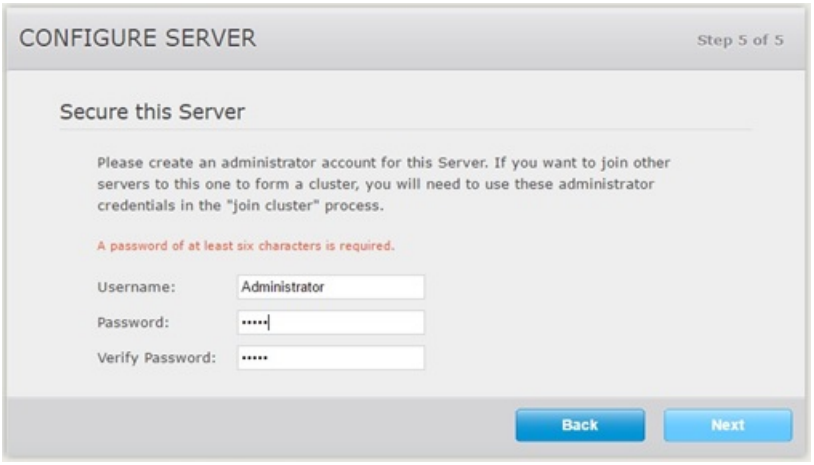
```
$ sudo wget ‘‘http://orientdb.com/download.php?file=orientd
  -graphed1.5.1.zip\&os=multi’’
$ sudo mv download.php?orientdb-graphed-1.5.1.zip\&os=
  multiorientdb
```

Figura 53 – Tela para preenchimento dos dados em caso de interesse de recebimento de notificações e atualizações disponíveis



Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

Figura 54 – Tela para preenchimento do dados de acesso ao *data bucket* recém criado



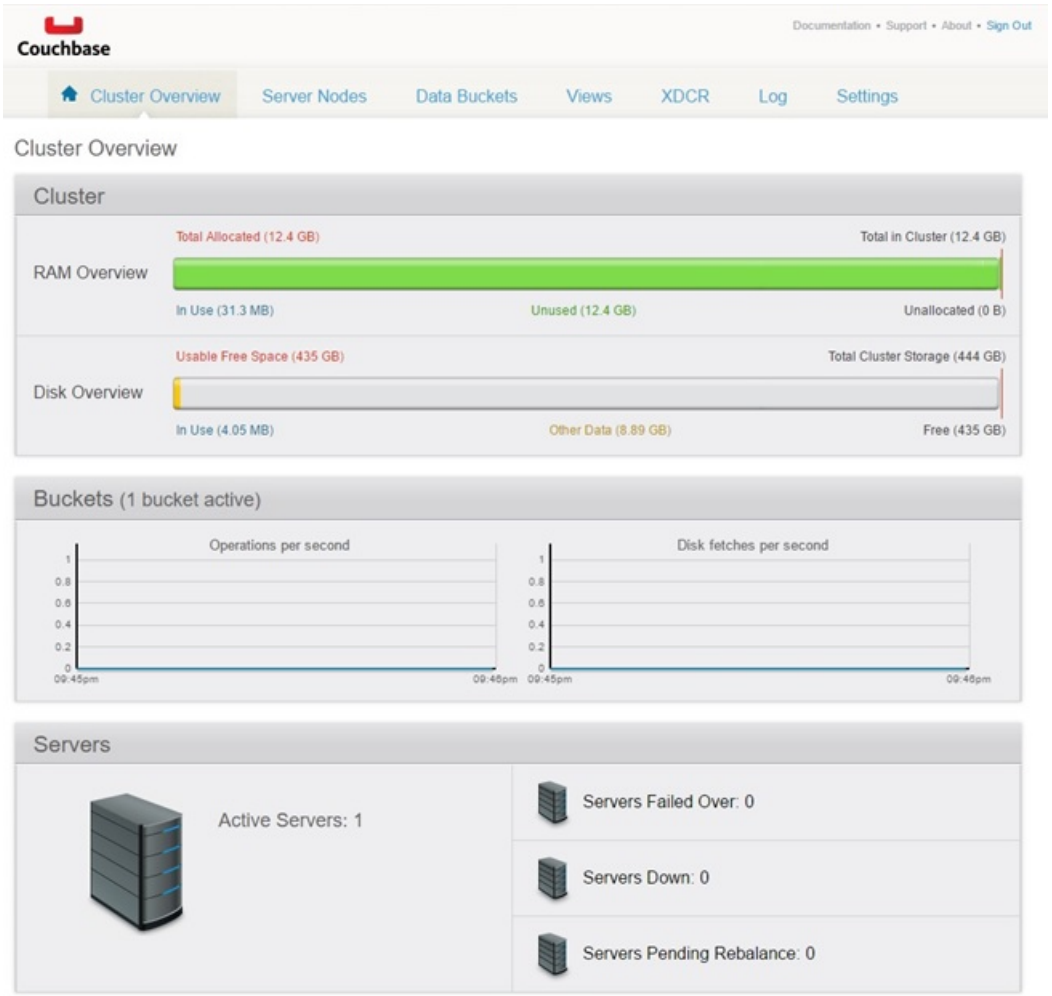
Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

```
$ sudo unzip -a orientdb
$ sudo mv orientdb /opt/
$ sudo cd /opt/orientdb/
```

Após a realização destes comandos, o sistema *OrientDB* estará disponível no diretório "*/opt/orientdb*". Para o seu correto funcionamento, é necessário que sejam alteradas algumas configurações, ou linhas, no arquivo *orientdb.sh*, encontrado no diretório "*/opt/orientdb/bin*"(o arquivo já alterado é apresentado no Apêndice D). Tais alterações podem ser realizadas por qualquer editor de texto, neste trabalho tendo sido escolhido o editor "*vim*"[10].

[10] http://www.vim.org/

Figura 55 – Tela apresentando os *cluster* que compõem o sistema.



Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

Com estas configurações alteradas, pode-se iniciar o servidor, sendo tal processo realizados através dos comandos abaixo:

```
$ cd /opt/orientdb
$ ./bin/server.sh
```

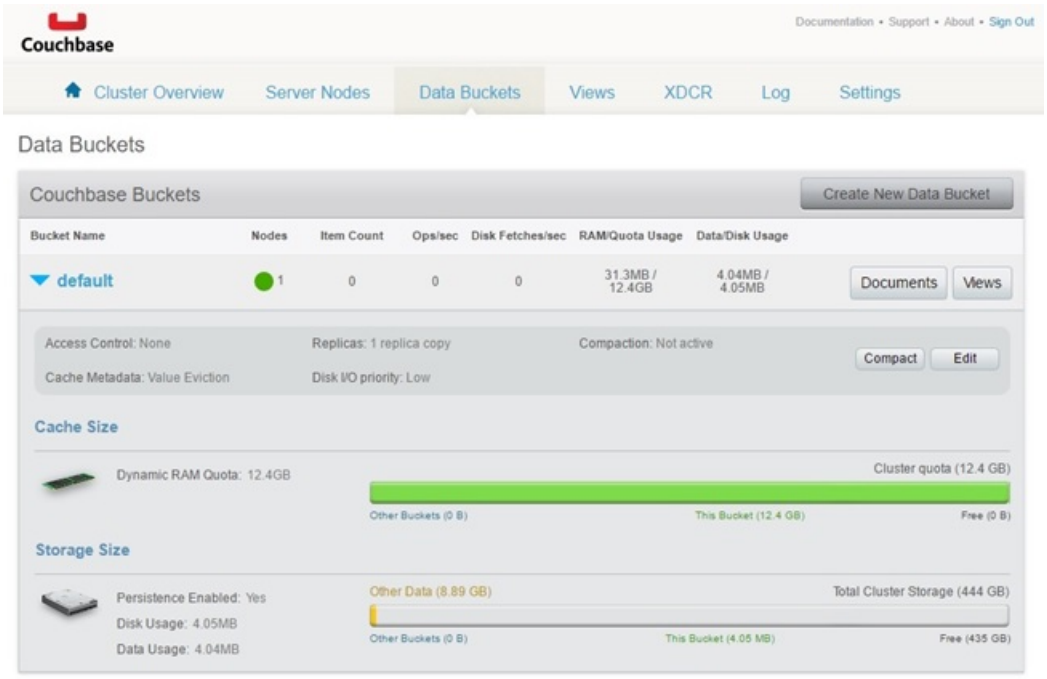
Após a realização dos comando anteriores, será apresentada um tela, conforme apresentado na Figura 57.

Salienta-se que da forma como o sistema foi instalado, o mesmo não será inciado automaticamente quando o sistema operacional inicializar, sendo necessário que seja executado o comando abaixo:

```
$ sudo bin/server.sh start
```

A criação da base de dados a ser utilizado nos testes, poderá ser realizada por meio de um navegador de internet através do *IP* do servidor do sistema, juntamente a porta

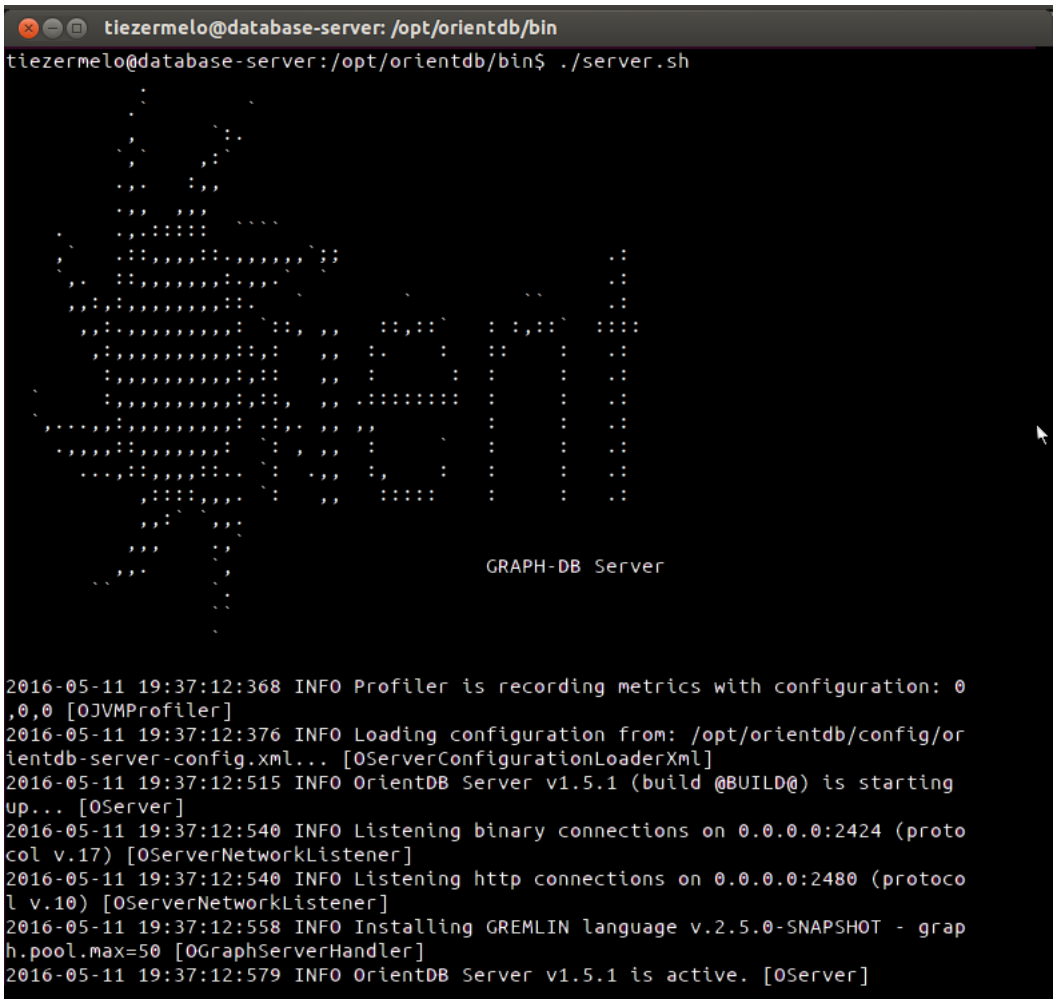
Figura 56 – Tela que apresenta os *data buckets* de determinado servidor no sistema *Couchbase*.



Fonte: *Print Screen* do acesso ao *Couchbase* pelo navegador de internet.

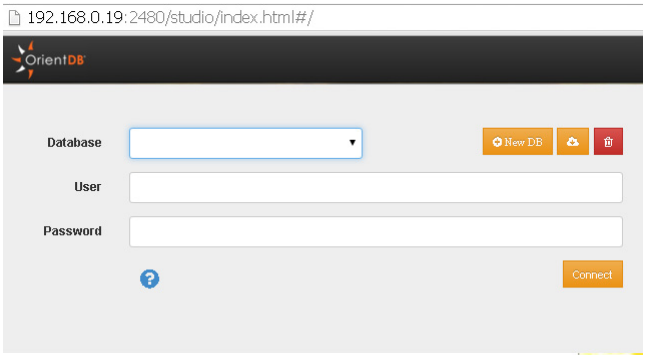
2480. No caso deste trabalho, o *IP* do servidor a ser utilizado é http://192.168.0.19:2480, conforme apresentado na Figura 58. Pode atribuir qualquer nome a base de dados, mas, neste caso, será escolhido o nome "*ycsb*". A base de dados criada é armazenada no diretório "*/opt/orientdb/databases*"com o nome "*ycsb*".

Figura 57 – Tela apresentada ao iniciar o servidor do sistema *OrientDB*, processo realizado através da execução do arquivo *orientdb.sh*



Fonte: *Print Screen* da tela apresentada ao executar o arquivo *server.sh*

Figura 58 – Criação da base de dados do *Orientdb* através do navegador de internet.



Fonte: *Print Screen* da tela do *Orientdb* acessado através do navegador de internet