Pós-Graduação em Ciência da Computação

Carlos Gomes Araújo

AVALIAÇÃO DO CONSUMO DE ENERGIA EM SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS NOSQL

Dissertação de Mestrado Profissional

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
<www.cin.ufpe.br/~posgraduacao>

RECIFE
2016

Carlos Gomes Araújo

AVALIAÇÃO DO CONSUMO DE ENERGIA EM SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS NOSQL

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Eduardo Antonio Guimarães Tavares*
Coorientador: *Meuse Nogueira de Oliveira Junior*

RECIFE
2016

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A662a Araújo, Carlos Gomes
Avaliação do consumo de energia em sistemas de gerenciamento de banco de dados NoSQL / Carlos Gomes Araújo. – 2016.
101 f.: il., fig., tab.

Orientador: Eduardo Antonio Guimarães Tavares.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2016.
Inclui referências e apêndice.

1. Banco de dados. 2. Avaliação de desempenho. I. Tavares, Eduardo Antonio Guimarães (orientador). II. Título.

025.04 CDD (23. ed.) UFPE- MEI 2016-130

Carlos Gomes Araújo

Avaliação do Consumo de Energia em Sistemas de Gerenciamento de Banco de Dados NoSQL

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre Profissional em 08 de agosto de 2016.

Aprovado em: 08/08/2016.

BANCA EXAMINADORA

_____

Prof. Dr Paulo Romero Martins Maciel
Centro de Informática / UFPE

_____

Prof. Dr. Aida Araújo Ferreira
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco

_____

Prof. Dr. Eduardo Antônio Guimarães Tavares
Centro de Informática / UFPE
(Orientador)

Dedico a Deus e minha família, alicerces da minha vida.

Agradecimentos

Agradeço a Deus por mais esta grande oportunidade em minha vida, e pela força para seguir em frente.

Agradeço à minha família, minha querida esposa e filha as quais privei um bom tempo de lazer nesta fase das nossas vidas. Obrigado pela paciência, agora vamos colher alguns frutos. Ao meu pai, onde quer que esteja, e à minha mãe, quanto foi difícil chegar aqui.

Agradeço ao meu orientador, professor Dr. Eduardo Tavares, pela confiança depositada em mim, o rigor de sua orientação foi fundamental para os resultados obtidos, ao meu coorientador e amigo professor Dr. Meuse Nogueira, pelo apoio no *framework* de medição do consumo de energia e apoio de forma geral.

Ao meu amigo Jackson Nunes, por todo o apoio e recepção nesta cidade, meus amigos na Reitoria do IFPE, amigos e colegas da turma do mestrado profissional.

O papel dos infinitamente pequenos na natureza é infinitamente grande.

—LOUIS PASTEUR

Resumo

NoSQL é uma tecnologia de sistemas de gerenciamento de banco de dados (SGBD) emergente, tendo modelos flexíveis focados em desempenho e escalabilidade, proposta para a manipulação de grandes quantidades de dados. NoSQL não substitui as abordagens de sistemas de gerenciamento de banco de dados relacionais, mas sim atende às restrições relacionadas à manipulação de dados em massa. Tal tecnologia já é aplicada em sistemas bem conhecidos em todo o mundo, tais como serviços de *e-commerce* e *middleware*. A importância de tal tecnologia tem motivado muitos trabalhos, principalmente em relação ao desempenho. Poucos trabalhos caracterizam e comparam o consumo de energia no contexto de SGBDs NoSQL, apesar de sua importância. De fato, o consumo de energia não deve ser negligenciado devido ao aumento dos custos financeiros e ambientais. A fim de avaliar essa questão, este trabalho analisa o desempenho e consumo de energia em sistemas de gerenciamento de banco de dados NoSQL, selecionamos o Cassandra (coluna), MongoDB (orientado a documento) e Redis (chave-valor) por serem representativos exemplos desta tecnologia. A metodologia baseia-se em *Design of Experiments*, de tal forma que as cargas de trabalho são geradas por Yahoo! Cloud Serving Benchmark (YCSB) produzindo leitura, escrita e atualização, por ciclos de 1.000, 10.000 e 100.000 operações. Como resultado são avaliados 27 tratamentos. Para a medição do consumo de energia é aplicado um *framework* específico chamado Emeter. As métricas são tempo de execução e consumo de energia, assim como a evolução no incremento da carga de trabalho. Os resultados demonstram que o consumo de energia pode variar significativamente entre os SGBDs para comandos distintos e cargas de trabalho. Conclui-se ainda que mesmo havendo uma correlação positiva entre o consumo de energia e o tempo de execução, o SGBD mais rápido não é, necessariamente o que utiliza menos energia.

Palavras-chave: NoSQL. SGBD. Orientado a Coluna. Orientado a Documento. Chave-valor. Consumo de Energia. Avaliação de Desempenho.

Abstract

NoSQL is an emergent database management systems technology (DBMS), having flexible models focused on performance and scalability, proposed for manipulating massive amounts of data. NoSQL is not intending for replacing the relational database management systems approaches, but to overcome constraints related to massive data manipulation. Such a technology already is applied in well-known systems around the world, such as e-commerce and middleware services. The importance of such technology has motivated lots of works, mainly relating to performance. Few works can be enumerated regarding characterization of energy consumption on NoSQL DataBase Management Systems, despite its importance. In fact the energy consumption is a feature that cannot be neglected due its impact on financial cost and environmental questions. In order to deal with such an issue, this work evaluates not only performance but the energy consumption involved on NoSQL DataBase Management Systems, specifically for Cassandra (Column), MongoDB (Document Oriented) and Redis (Key-Value). The methodology is based on Design of Experiments, in such a way the workloads are generated by Yahoo! Cloud Serving Benchmark (YCSB) producing readings, writings and updatings by cycles of 1.000, 10.000 and 100.000. As result, it is evaluated twenty seven treatments. For measuring energy consumption is applied a specific framework named Emeter. The Emeter captures metrics such as execution time and energy consumption related to treatments under analyze. In addition to the individual evaluation, the performance and energy consumption are analyzed among relevant scenarios, as well as the trends due to increases in the workload. The results demonstrate that energy consumption can differs for each DBMS according to command and workload. Additionally, the results make it possible to infer that despite the well-known positive correlation between performance and energy consumption, the fastest DBMS is not necessarily the best on saving energy.

Keywords: NoSQL. DBMS. Colunm Oriented. Document Oriented. Key-Value. Energy Consumption. Performance Evalution.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ACID Atomicidade, Consistência, Isolamento e Durabilidade

ANOVA *Analysis of variance*

BASE *Basically available, soft state, eventually consistent*

BD Banco de Dados

CAP *Consistency, Availability, Partition tolerance*

DOE *Design of Experiment*

SGBD Sistema de Gerenciamento de Banco de Dados

RAM *Random Access Memory*

SoC *System on Chip*

TCP *Transmission Control Protocol*

TI Tecnologia da Informação

VM *Virtual Machine*

WEB *World Wide Web*

YCSB Yahoo! Cloud Serving Benchmark

Sumário

1

Introdução

O consumo de energia em *data center* está aumentando significativamente. De acordo com relatórios NRDC (2015), em 2013, *data centers* consumiram 91 bilhões de quilowatts nos Estados Unidos, a um custo de US $ 9 bilhões de dólares. Além disso, estima-se que o consumo anual será na ordem de 140 bilhões de quilowatts até 2020 (a um custo de US $ 13,7 bilhões de dólares). Se confirmado, este gasto será responsável pela emissão de 100 milhões de toneladas de carbono, acarretando graves prejuízos ao meio ambiente. Outro dado alarmante aponta que a quota dos grandes *data centers* no consumo de energia é responsável pela parcela entre 1,1 e 1,5% do consumo total de energia elétrica em todo o mundo e está projetado para subir ainda mais (MASTELIC et al., 2014).

A eficiência energética dos equipamentos de computação em *data center* é uma preocupação importante, uma vez que a alimentação e refrigeração deste ambiente tem significativa parcela dos custos de energia. O aumento do uso de energia em *data centers* fez - inclusive - governos em todo o mundo se preocuparem em tentar regulamentar o consumo de energia em tecnologia da informação (TI). Novas formulações de plataformas energeticamente eficientes estão sendo propostas (TSIROGIANNIS; HARIZOPOULOS; SHAH, 2010).

No tocante ao armazenamento de dados, com o avanço do poder de processamento de dados e comunicação houve uma explosão na quantidade de dados disponíveis. Essa grande massa de dados, conhecida como *Big Data*, concentra informações significativas sobre as pessoas, suas relações de comunicação e consumo. Y. DEMCHENKO C. DE LAAT (2014) afirmam que *Big Data* está se relacionando com quase todos os aspectos da atividade humana a partir de eventos de gravação para investigação, concepção, produção de serviços digitais ou produtos de entrega ao consumidor final.

JI et al. (2012) citam três desafios para *Big Data*. O primeiro seria o grande volume de dados: a velocidade de aumento da capacidade de armazenamento é muito menor que o crescimento dos dados. Todos os dias, por exemplo, são criados 2,5 quintilhões de *bytes* de dados, 90% dos dados no mundo de hoje foram criados nos últimos dois anos. Esses dados vêm de toda parte: sensores utilizados para reunir informações sobre o clima, mensagens para sites de mídia social, fotos digitais, vídeos, registros de compras (IBM, 2015). Um exemplo

notável é o Flickr, um site de compartilhamento de imagens públicas, que recebeu em média dois milhões de fotos por dia, em 2015 (FLICKR, 2015). O segundo, a velocidade de acesso a esses dados é um fator muito importante quando se realiza, por exemplo, uma compra em um cartão de crédito, pois é necessário que o sistema de detecção de fraudes responda antes da conclusão da transação. Os sistemas de banco de dados tradicionais - mesmo com seus mecanismos de otimização de desempenho - não conseguem responder a esta demanda no tempo necessário, ou seja, não é possível percorrer toda essa massa de dados no tempo previsto e os algoritmos tradicionais são ineficientes para os grandes dados. Assim, uma solução viável seria distribuir o processo paralelamente em milhares de computadores espalhados na nuvem que funcionam sob demanda. O terceiro ponto seria a segurança: afeta todo o armazenamento e processamento dos grandes dados, uma vez que há um uso massivo de serviços de terceiros e infraestruturas que são utilizadas para hospedar dados importantes ou para realizar operações críticas. A escala de dados e aplicações cresce exponencialmente, trazendo enormes desafios à monitoração de dados dinâmicos e à segurança.

Nesse contexto, surge o NoSQL, conhecido na comunidade de usuários por "*Not Only SQL*" ou não relacional, estabelecendo uma tecnologia desenvolvida para suportar bancos de dados com grande volume, distribuídos e com alta capacidade de escalabilidade. O nascimento do NoSQL originou uma série de gerenciadores de bancos de dados com características distintas de otimização e desempenho para atenderem a determinados segmentos de aplicações ou modelos de dados. Cerca de 150 SGBDs (Sistema de Gerenciamento de Banco de Dados) foram introduzidos nos últimos anos (KUZNETSOV; POSKONIN, 2014).

Esse emergente número de sistemas em nuvem junto com a falta de um padrão de comparação de desempenho, dificulta o entendimento das relações entre os sistemas e a carga de trabalho. Com essa diversidade encontramos trabalhos acadêmicos e *benchmarks* comparando o desempenho nos aspectos velocidade e latência, contudo, ainda não existe maturidade na escolha do sistema de gerenciamento de banco de dados NoSQL, visando à redução do consumo de energia.

Não se conhece a relação entre o desempenho das operações de NoSQL e seu consumo de energia elétrica. Este trabalho é realizado com base na hipótese que os SGBDs podem apresentar diferenças significativas quanto ao desempenho e consumo de energia.

1.1 Motivação

A modelagem relacional em banco de dados CODD (1970), apresenta restrições para *Big Data*, restrições essas que inviabilizam a utilização em serviços como redes sociais, *web* analítica, *e-comerce*, detecção de fraudes e internet das coisas. NoSQL não foi criado para substituir o modelo relacional, mas se adequa a *Big Data*, normalmente substitui características de Atomicidade, Consistência, Isolamento e Durabilidade (ACID) do modelo relacional por um modelo de dados mais flexível, que possibilita escalar este sistema. A característica não

relacional destes SGBDs ao mesmo tempo que flexibiliza as soluções para os mais variados cenários de utilização possibilita a criação de NoSQL em categorias distintas, dificultando a comparação de suas características nos mais variados ambientes.

NoSQL diariamente tem adquirido maior importância no armazenamento de dados, é utilizado não somente por empresas líderes de segmento a exemplo do Google ou Facebook. Também vem se popularizando em milhares de empresas que necessitam de serviços escaláveis, estes hospedam em *web services* fornecidos por empresas como a Amazon com o serviço (AWS) (AMAZON WEB SERVICES, 2016a), fornecendo sua estrutura de TI a mais de um milhão de clientes. A Microsoft, com *web service* Microsoft Azure onde mesmo possuindo o SGBD relacional SQLServer reconhecido em todo o mundo desenvolveu recentemente o seu SGBD NoSQL (MICROSOFT, 2016) para atender a esta nova demanda de gerência de dados para os aplicativos atuais. O Google oferece a Plataforma de Nuvem Google (GOOGLE, 2016).

Deve-se levar em consideração que esta tecnologia já é responsável por uma considerável parcela do consumo de energia elétrica em *data center*. Com o aumento desta demanda há duas alternativas mais evidentes: pode-se aumentar a produção de energia elétrica, que tem sido uma alternativa de alto impacto ambiental e financeiro devido à necessidade da construção de novas fontes geradoras, ou melhorar a eficiência do consumo de energia elétrica que, por consequência reduz o consumo de energia elétrica possibilitando a redução do impacto ambiental e financeiro. REIS (2015) afirma que é necessário aumentar a eficiência do setor energético desde a produção até o consumo, pois grande parte da crescente demanda energética pode ser suprida através de medidas sustentáveis, principalmente em países desenvolvidos, nos quais a demanda deve crescer de forma mais moderada. Neste cenário, as pesquisas voltadas para a análise do consumo de energia elétrica em SGBDs NoSQL ainda não obtiveram enfoque na avaliação da diferença de consumo elétrico desses SGBDs. Os resultados dessas pesquisas podem fomentar e atrair a atenção da comunidade científica a esta promissora área de estudo. Esse fato motiva este trabalho.

1.2 Objetivos

O objetivo geral deste trabalho propõe caracterizar três sistemas gerenciadores de banco de dados (SGBDs), de modelos NoSQL distintos, em função de seu consumo de energia elétrica em situações de leitura, escrita e atualização bem como em cargas de trabalho diferentes em um mesmo ambiente de operação. Sendo assim, este trabalho tem como foco caracterizar:

- o consumo de energia elétrica de três populares SGBDs NoSQL, Cassandra, MongoDB e Redis representantes respectivamente dos modelos Coluna, Orientado a documento e Chave-valor;
- o impacto dos comandos de leitura, escrita e atualização no consumo de energia elétrica dos SBGDs; e

- a correlação do tempo de execução com o consumo de energia elétrica dos SGBDs.

Como objetivos específicos deste trabalho propõe-se:

- desenvolver uma metodologia de análise do consumo de energia utilizando um *benchmark* para gerar a carga de trabalho;
- montar um ambiente de experimentação para execução do *benchmark*;
- analisar estatisticamente o consumo de energia elétrica dos SGBDs NoSQL nos cenários de experimentação deste estudo; e
- relacionar as características dos SGBDs que influenciaram no consumo de energia elétrica.

1.3 Trabalhos Relacionados

De acordo com o relatório Intel Labs (MINAS; ELLISON, 2009), em um servidor, a CPU é o principal consumidor de energia (seguido pela memória RAM), indicando que os processos que requerem o uso intenso de CPU e de acesso à memória, como os utilizados por SGBD, levam à elevação do consumo de energia.

Poucos trabalhos têm avaliado o desempenho e o consumo de energia de SGBDs NoSQL distintos em variados cenários de utilização. Dentre essas avaliações encontradas os estudos mais comuns são comparações relativas a parâmetros como tempo de execução e latência, em ambientes geralmente distribuídos, comumente utilizando maquinas com alta capacidade de memória e processamento e alta carga de trabalho (FLORATOU et al., 2012; SERIATOS et al., 2016; NEVES; BERNARDINO, 2015). Ainda mais difícil é encontrar na literatura trabalhos relacionados à comparação do consumo de energia elétrica em sistemas de gerenciamento de banco de dados NoSQL.

SUBRAMANIAM; FENG (2014) estudam o consumo de energia para os diferentes níveis de cargas abordados em servidores com armazenamento de dados NoSQL. Com esta finalidade foi aferido o consumo de energia e desempenho em um ambiente de *cluster* com quatro servidores do SGBD Cassandra, também foi utilizado o gerador de carga de trabalho YCSB com a distribuição de dados *zipfian*. Utilizando uma interface para medição e controle, que possibilita impor limites do consumo de energia dos subsistemas do *hardware*, adicionalmente a utilização de diferentes técnicas de gerenciamento de energia, se avalia a relação entre a latência das operações, economia de energia e proporção deste consumo nos servidores do *cluster*. Para atender a um determinado objetivo de nível de serviço do SGBD.

O trabalho de NEVES; BERNARDINO (2015) avaliou o desempenho e a escalabilidade do SBGD Voldemort NoSQL utilizando como gerador de carga de trabalho o YCSB, o objetivo foi avaliar o desempenho em ambiente de configuração com 1, 3 e 6 servidores, com cargas

de trabalho distintas. Nos resultados demonstrados o Voldemort não apresentou uma boa escalabilidade, ou seja, o número adicional de servidores não aumentou de forma significativa o seu desempenho, é importante salientar que para estes experimentos os servidores do SGBD foram máquinas virtuais compartilhando o mesmo *hardware* também não foi esclarecido a configuração deste SGBD.

CAI et al. (2013) analisam o ciclo de acesso ao Hbase NoSQL, avaliando as médias de acesso por hora durante 24 horas por 7 dias da semana de modo a elaborar um modelo simulando um ambiente real. Baseando-se neste modelo, parametrizaram o YCSB como gerador de carga de trabalho alterando os parâmetros com referência ao ciclo proposto. Os resultados mostraram que a alteração do tempo de resposta foi correspondente ao número de *threads*[1] utilizadas nos experimentos.

ABRAMOVA; BERNARDINO (2013) avaliaram a evolução do tempo de execução conforme o aumento da carga de trabalho comparando dois populares SGBDs NoSQL, o Cassandra e o MongoDB, os experimentos foram executados em basicamente seis tipos de cargas de trabalhos, tendo-se utilizado o YCSB como ferramenta de *benchmark*. De maneira geral, os resultados mostraram que, com o aumento dos dados, o MongoDB começou a ter um desempenho inferior, enquanto o Cassandra melhorou seu desempenho no aumento desta carga de trabalho. Apesar de demonstrado a diferença de desempenho, este trabalho não abordou as características dos SGBDs que poderiam influenciar para nesses resultados, porém, quando semelhante a este trabalho, foi realizada a comparação do comando de leitura em uma carga de 100.000 operações, e apesar de o ambiente de hardware ser diferente, o resultado da relação de desempenho (Cassandra x MongoDB) foi similar ao realizado neste trabalho.

ABUBAKAR; ADEYI; AUTA (2014), avaliaram quatro SGBDs NoSQL (MongoDB, ElasticSearch, OrientDB e Redis) em ambiente de uma mesma máquina com cargas e operações distintas (leitura, escrita e atualização) foi utilizado como gerador de carga de trabalho o YCSB. Os resultados dos experimentos também demonstraram aptidões específicas de cada SGBD, porém não foi explicitado o ambiente do experimento.

Em (SERIATOS et al., 2016) o desempenho (Latência x *Throughput*[2]) de três SGBDs NoSQL (HBase, MongoDB e Cassandra) foi avaliado em um ambiente de nuvem, utilizando como gerador de carga o YCSB. Os resultados dos experimentos foram avaliados em conjunto com as configurações dos SGBDs.

O trabalho de LI et al. (2014) estudou o comportamento do consumo de energia de banco de dados NoSQL no estado de espera, comparando o consumo de SGBDs configurados em *cluster* no tempo ocioso de cada nó. Como resultado, foi observado, além da diferença de consumo entre os SGBDs, que durante a execução dos casos de testes o tempo de espera da CPU foi responsável pelo disperdício de elevada quantidade de energia. Tomando o HadoopDB

[1] Segmentos de clientes. Por padrão, o YCSB cliente usa um único segmento de trabalho, mas tópicos adicionais podem ser especificados.

[2] Quantidade de dados processados em um determinado espaço de tempo, neste caso (operações por segundo).

como exemplo, o consumo de energia em espera (WEC) é causado principalmente pela carga de trabalho insuficiente e inadequado algoritmo de *scheduling*[3], servidores ficam esperando por demanda de trabalho. Quando uma abordagem eficiente de escalonamento é introduzida, a WEC é reduzida. Tomando como exemplo o HBase, a WEC é causada principalmente pela CPU esperando I/O remoto ou local, espera mútua de nós causada por execução não síncrona.

TSIROGIANNIS; HARIZOPOULOS; SHAH (2010) realizaram a avaliação de diversos cenários simulando execuções em banco de dados utilizando o TPC-H *benchmark* TPC (2016) em diversas operações, os banco de dados PostgreSQL (POSTGRESQL, 2016) e outro SGBD comercial foram utilizados, semelhante ao realizado na aferição do consumo de energia, utilizando um equipamento para medir o consumo de energia total do sistema. Os resultados demonstram que o consumo de energia da CPU utilizada por diferentes operadores pode variar amplamente, em até 60%, e que o consumo da CPU não é linear à sua utilização. Este trabalho não analisou SGBDs NoSQL a relação com este trabalho é a característica de sua execução.

Esta pesquisa propõe casos de testes com diferentes cargas de trabalho, analisando o consumo de energia elétrica apenas no tempo de execução em diferentes SGBDs e as suas relações com o desempenho destes casos de testes. Realizou-se a medição do consumo de energia elétrica com um *hardware* específico de medição, um *framework* e *scripts* foram desenvolvidos especificamente para a execução e análises destes casos de testes. Minimizou-se a possibilidade de que outros processos interferissem nos resultados, para que o servidor hospedeiro do banco de dados executasse apenas o sistema operacional com seus processos e o SGBD avaliado. Comparando aos trabalhos relacionados, este documento difere no estudo do consumo de energia elétrica e desempenho. A execução dos experimentos possibilitou melhor análise estatística dos cenários estudados. Destacou-se este ponto, pois os trabalhos relacionados realizaram a análise baseando-se apenas nos resultados médios em poucas execuções dos experimentos propostos, ignorando os métodos estatísticos de análise. O mais importante é que este trabalho inicia uma nova perspectiva para entendimento das características que mais influenciam no consumo de energia em SGBDs NoSQL.

1.4 Estrutura da Dissertação

Nesta seção, apresenta-se a contextualização dos capítulos que seguem neste trabalho.

Capítulo 2 - Referencial Teórico: é exposto o resultado de uma pesquisa bibliográfica que buscou fundamentar as ideias apresentadas e defendidas neste trabalho. Para alcançar os objetivos do trabalho, os conceitos abordados foram: SGBDs NoSQL, Avaliação de Desempenho e *Benchmark*.

Capítulo 3 - Metodologia: Explicam-se os métodos, ferramentas e configurações utili-

[3]Escalonamento de processos ou agendador de tarefas, é uma atividade organizacional feita pelo escalonador (scheduler) da CPU ou de um sistema distribuído, possibilitando executar os processos mais viáveis e concorrentes, priorizando determinados tipos de processos.

zadas neste trabalho. Inicia demonstrando a utilização da metodologia de *Design of Experiments* neste trabalho, mostra o fluxograma e explica o fluxo do método utilizado, segue explicando a arquitetura básica da execução dos experimentos, a configuração do Yahoo! Cloud Serving Benchmark utilizada, detalha o *framework* Emeter, e, por fim, apresenta os passos para a automação do *benchmark*.

Capítulo 4 - Resultados: Apresentam-se os resultados obtidos neste estudo. O resultado de cada tratamento é apresentado, realiza-se a comparação dos resultados pertinentes e a evolução no acréscimo da carga de trabalho com fundamentos estatísticos; em seguida, apresenta-se a correlação para as métricas tempo de execução e consumo de energia; por fim, executa-se uma discussão dos resultados.

Capítulo 5 - Conclusão: Conclui este trabalho apresentando as principais contribuições e sugerindo possíveis trabalhos futuros.

2

Referencial Teórico

Este capítulo apresenta os conceitos básicos dos temas abordados nesta dissertação, e está dividido em duas partes. A primeira parte expõe os fundamentos do NoSQL, suas principais características, categorias e SGBDs abordados neste estudo. A segunda parte apresenta os conceitos básicos da avaliação de desempenho de sistemas computacionais, *Design of Experiments* (DoE) e uma breve descrição da análise de variância (ANOVA).

2.1 SGBD NoSQL

Há alguns anos, os bancos de dados apareceram como um repositório com dados organizados e estruturados, estes dados são combinados em um conjunto de registros, organizados em uma estrutura regular. Para acessar os dados é comum o uso de um sistema, geralmente conhecido como Sistema de Gerenciamento de Banco de Dados (SGBD). O SGBD pode ser definido como um conjunto de mecanismos que permite o armazenamento, edição e extração dos dados, o conceito de SGBD tornou-se sinônimo de banco de dados (ABRAMOVA; BERNARDINO, 2013).

A necessidade do gerenciamento de grande volume de informações encontra barreiras nas tecnologias tradicionais com características de atomicidade, consistência, isolamento e durabilidade (ACID), esta nova necessidade impulsiona o movimento NoSQL, que minimiza essas dificuldades e oferece com seus benefícios novas possibilidades de estudos para otimização do uso dessa nova tecnologia. BREWER (2012) afirma que NoSQL prioriza a disponibilidade, pois a consistência fica em segundo plano. Os SGBDs que aderem a características ACID fazem o oposto: priorizam a consistência e o desempenho fica em segundo plano.

Big Data é uma expressão utilizada para definir aplicações com campos, ou seja, menor unidade destinada a armazenamento de valores - muito grandes, geralmente sem estrutura (BARBIERATO; GRIBAUDO; IACONO, 2014). O dimensionamento de sistemas que utilizam as transações ACID se mostra um problema, os conflitos decorrem de diferentes aspectos da alta disponibilidade de sistemas distribuídos que não são totalmente solucionáveis, estes aspectos (do inglês, Consistency, Availability, e Partition tolerance) são conhecidos como teorema CAP

(MONIRUZZAMAN; HOSSAIN, 2013). O teorema CAP é demonstrado no seguinte exemplo: Em uma rede sujeita a falhas de comunicação, é impossível que qualquer serviço da *web* implemente uma leitura atômica, gravação compartilhada na memória e que garanta uma resposta a todos os pedidos (GILBERT; LYNCH, 2012).

O termo NoSQL pode ser definido como não somente SQL. O acrônimo NoSQL foi cunhado em 1998 por STROZZI (1998). Esta tecnologia não substitui o modelo relacional, a ideia é que ambas as tecnologias possam coexistir e que cada uma tenha o seu lugar. Em 2007 aconteceu um dos inícios do movimento rumo à tecnologia NoSQL, a Amazon apresentou o NoSQL Dynamo. Esta foi das primeiras empresas a guardar os seus dados em uma base dados de característica não relacional (LEAVITT, 2010). A mobilização à NoSQL começou em junho de 2009 em uma conferência realizada em San Francisco - California para disseminação desta tecnologia, e está crescendo rapidamente. Neste evento, o exemplo dos SGBDs Bigtable PLATAFORM (2016) e Dynamo AMAZON WEB SERVICES (2016b) inspirou muitos projetos com um armazenamento de dados alternativo (NOSQL-MEETUP, 2009; GROLINGER et al., 2013).

Muitas características desta tecnologia são explícitas, tais como, livre de esquema, suporte a fácil replicação, API simples, consistência eventual (não ACID) (NOSQL, 2015). A tecnologia NoSQL tem sido notícia nos últimos anos, muitos líderes da *web* 2.0 adotaram NoSQL. Empresas como Facebook, Twitter, Digg, Amazon, LinkedIn e Google utilizam esta tecnologia, de uma forma ou de outra (MONIRUZZAMAN; HOSSAIN, 2013).

De forma diferente do modelo relacional, que é fundamentado na lógica matemática e na teoria dos conjuntos, possuindo a organização de seus dados em esquemas com estruturas tabulares rígidas (NAVATHE; ELMASRI, 2010), os sistemas NoSQL não seguem essas regras. Eles costumam apresentar flexibilidade em sua estrutura de dados, comumente são sistemas que se apresentam em código aberto, escaláveis e distribuídos (NOSQL, 2015). Estas características tornam os armazenamentos de dados NoSQL especialmente adequados para utilização como sistemas de gestão de dados de nuvem. No entanto, a falta de suporte a transações ACID completa pode ser um grande impedimento para a sua utilização em muitos sistemas de missão crítica (GROLINGER et al., 2013).

Estudos indicam que a latência é um fator crítico para interações *on-line*, o aumento de apenas 100 ms pode reduzir drasticamente a probabilidade de que um cliente continue a interagir ou voltar no futuro para aquele portal (ABADI, 2012). Em outros casos, pode ser preferível lidar com a redução de desempenho em função da grande massa de dados em detrimento a falta de suporte a transações ACID. Além disso, o uso de linguagens de baixo nível de consulta, a falta de interfaces padronizadas, e os enormes investimentos já realizados em SQL por parte das empresas são outras barreiras para a adoção de armazenamento de dados NoSQL.

O DB-Engine avalia mensalmente a popularidade de sistemas de gerenciamento de banco de dados de forma geral, para esta avaliação aspectos como, número de menções do sistema em sites; interesse geral no sistema (Google Trends); frequência de discussões técnicas sobre

o sistema; número de ofertas de emprego; dentre outros são utilizados. Três SGBDs NoSQL (MongoDB, Cassandra e Redis) estão entre os dez SGBDs mais populares em maio de 2016 (IT, 2016).

De forma geral, para os SGBDs NoSQL a ausência de um *schema* definido (*Schema-less*) permite maior flexibilidade. Outras características apoiam o gerenciamento de grandes dados, tais como: não partilham a mesma arquitetura, ou seja, em vez de utilizar um *pool* de armazenamento comum Storage Area Network (SAN) utiliza apenas o armazenamento local, permitindo que a velocidade de acesso seja apenas a de acesso ao disco rígido; sua elasticidade é uma importante característica, ou seja, quando um novo servidor é adicionado um subconjunto de dados deve ser replicado nesses novos servidores; *Sharding*, os registros podem ser divididos em fragmentos, que são pequenos o suficiente para serem gerenciados por um único servidor; replicação assíncrona, permite gravações mais rápidas e suaves uma vez que são independentes do tráfego de rede. No entanto, os dados não são instantaneamente replicados e podem ser perdidos em certas janelas; BASE em vez de ACID, o acrônimo BASE foi propositadamente escolhido por conta do contraste com o paradigma ACID, uma vez que NoSQL enfatiza a disponibilidade e o desempenho (CHANDRA, 2015).

Sistemas de gerenciamento de banco de dados NoSQL são muitas vezes referidos como sistemas BASE (*Basically Available, Soft state, Eventually consistent*), característica que auxilia no ganho de desempenho, (KLEIN et al., 2015) quando aumentaram o nível de consistência dos dados em seu estudo de caso, observou uma queda de rendimento entre 10 e 25%.

2.1.1 Propriedades de persistência BASE

Os SGBDs NoSQL seguem a propriedade de consistência conhecida como BASE (*basically available, soft state, eventually consistent*). Ao contrário da propriedade ACID, que garante um controle de integridade, BASE prioriza o desempenho, a consistência fica em segundo plano, aceita que a consistência esteja em um estado de fluxo, ou seja, tolera inconsistências temporárias para obter maior disponibilidade (PRITCHETT, 2008).

CHANDRA (2015) apresenta as propriedades BASE da seguinte forma:

- ***Basically available:*** Os sistemas de banco de dados sempre parecem funcionar, indica que o sistema garante a disponibilidade. SGBDs NoSQL espalham dados através de muitos subsistemas (nós do *cluster* do banco de dados) de armazenamento com um alto grau de replicação para obter um alto grau de disponibilidade;

- ***Soft state:*** Estado leve e consistente, as cópias de itens de dados podem ser inconsistentes e o estado do sistema pode mudar ao longo do tempo. SOUZA; SANTOS (2015) dizem que esta característica define que os desenvolvedores devem criar mecanismos para a consistência de dados desejada;

- ***Eventually consistent:*** Se certo registro não sofre atualizações ao longo de um determinado período de tempo eventualmente ele está consistente. O período entre atualizações e o momento em que ela é garantida é chamado de janela de inconsistência (PRITCHETT, 2008).

2.1.2 Modelos NoSQL

Os sistemas de gerenciamento de bancos de dados NoSQL tipicamente são classificados em quatro modelos: *(i)* chave-valor exemplo Redis, SimpleDB; *(ii)* orientado a colunas exemplo Cassandra, Hbase, Accumulo; *(iii)* orientado a documento exemplo MongoDB, CouchDB; *(iv)* orientados a grafos exemplo Neo4J, Infinite Graph, InfoGrid (ABRAMOVA; BERNARDINO, 2013; BARBIERATO; GRIBAUDO; IACONO, 2014; SOUZA; SANTOS, 2015).

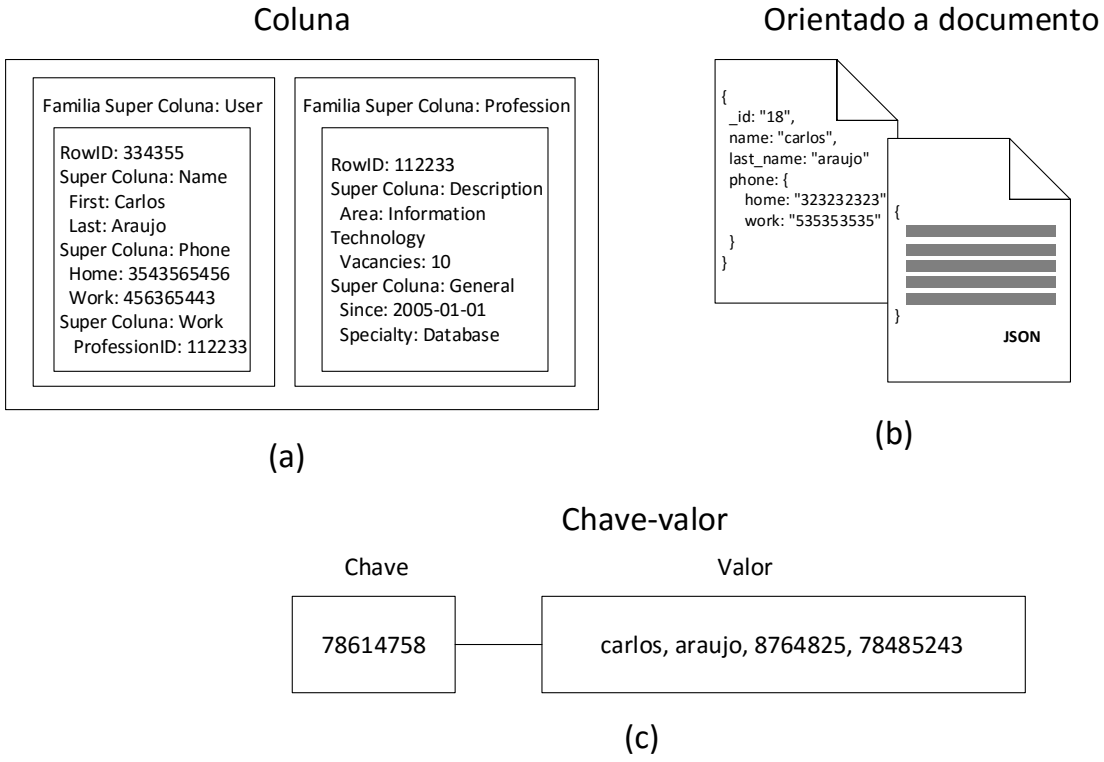

Figura 2.1: Modelos de Dados NoSQL

Neste trabalho os SGBDs avaliados são representantes dos modelos chave-valor, coluna e orientado a documento. Por serem os mais utilizados, na Figura 2.1 são ilustrados estes modelos de dados.

2.1.2.1 Coluna

O modelo orientado a coluna representado na Figura 2.1 (a) é o tipo mais semelhante ao modelo de base de dados relacional. No entanto, em vez de armazenar os dados em linhas,

os dados são estruturados em colunas que podem ser inúmeras. Os dados são armazenados nas tabelas com as etiquetas, cada linha da tabela consiste em uma chave e muitas colunas (CAI et al., 2013).

ABRAMOVA; BERNARDINO (2013) descrevem a seguinte estrutura e organização dos dados para esta categoria:

- Coluna – representa uma unidade de dados composta por chave e valor;
- Super Coluna – agrupado por colunas de informação;
- Família de Coluna – conjunto de dados estruturados semelhante a tabela de banco de dados relacional, constituída por variedades de super colunas.

Cassandra (PLANET, 2015), que é um popular SGBD orientado a coluna desenvolvido em Java, foi inicialmente desenvolvido pelo Facebook usando os SGBDs Dynamo (DECANDIA et al., 2007) e BigTable (CHANG et al., 2008) como base. Além disso, Cassandra fornece um conjunto de funcionalidades para facilitar a implementação em arquiteturas distribuídas, e adota uma linguagem de consulta com uma sintaxe semelhante ao SQL. Altamente escalável, o Cassandra foi projetado para armazenar uma grande quantidade de dados em vários servidores, proporcionando alta disponibilidade e dados consistentes (LAKSHMAN; MALIK, 2010).

Os principais elementos do Cassandra são *keyspaces*, famílias de colunas, colunas e linhas (PARTHASARATHY, 2013). Um *keyspace* contém os passos de processamento da replicação de dados e é semelhante a um esquema de uma base de dados relacional, normalmente, um *cluster* tem uma *keyspace* por aplicação. Uma família de coluna é um conjunto de pares chave-valor contendo uma coluna com sua chave de linha. Uma coluna é o menor incremento de dados, que contém um nome, um valor, e um carimbo de hora. As linhas são colunas com a mesma chave primária. Cassandra utiliza uma arquitetura *masterless*, ou seja, todos os nós desempenham o mesmo papel no sistema de replicação.

Quando uma operação de gravação ocorre, Cassandra armazena imediatamente a instrução no *Commit log* (MARTIN J. FOWLER, 2014), que vai para o disco rígido. Os dados desta operação de gravação são armazenados no *Memtable*, que permanece na memória RAM. Somente quando um limite de memória prédefinido é atingido é que esses dados são escritos em *SSTables* que ficam no disco rígido. Em seguida, o *Commit log* e o *Memtable* são limpos (PARTHASARATHY, 2013). As inserções/atualizações de dados são inseridas em um novo *SSTable*, o Cassandra gerencia o acúmulo de *SSTable* usando a compactação, a exclusão na *SSTable* não é realizada, isso porque a mesma é imutável, em vez disso Cassandra marca os dados a serem excluídos. Durante a compactação, há um pico temporário no uso do espaço em disco e I/O, porque são trabalhadas as *SSTables* antigas e novas ao mesmo tempo (ACADEMY, 2016).

2.1.2.2 Orientado a Documento

Apresenta-se na Figura 2.1 (b) o modelo orientado a documento que representa seus dados usando um formato de documento, geralmente, com base em padrões bem conhecidos, tais como XML e JSON. Cada documento pode ter vários campos, os quais podem incluir matrizes e ainda outros documentos. Um documento tem uma chave única, que é o mecanismo usual para acessar os documentos. Além disso, os documentos podem ser agrupados (ABRAMOVA; BERNARDINO, 2013). Esta abordagem documento torna possível representar uma relação hierárquica complexa em um único registro, também não há esquemas predefinidos, chaves e valores de um documento não são de tipos ou tamanhos fixos CHODOROW (2013).

MongoDB (MONGODB, 2015) é um SGBD representante da categoria orientado a documentos desenvolvido em C++, adota um modelo semelhante ao JSON para armazenamento de dados. Para evitar problemas de desempenho, MongoDB adota amplamente o uso da memória. Os documentos são manipulados usando operações convencionais, tais como a leitura, escrita, exclusão e atualização. Para procurar documentos, alguns mecanismos estão disponíveis com base em projeção (adotado pelo *benchmark* YCSB) e iteração. Além disso, durante a atualização, um documento pode ser modificado total ou parcialmente.

MongoDB usa o mecanismo de replicação mestre-escravo que permite definir um mestre e um ou mais escravos, o mestre pode escrever ou ler arquivos enquanto escravo tem o papel de *backup*, por isso apenas operações de leitura são permitidos, na ausência do mestre, o escravo assume com os dados mais recentes é promovido a mestre. Todas as réplicas são assíncronas, membros da réplica podem ser configurados pelo administrador do sistema (ABRAMOVA; BERNARDINO, 2013).

2.1.2.3 Chave-valor

O modelo chave-valor representado na Figura 2.1 (c) baseia-se em um modelo bastante simples, permite a visualização da base de dados como uma grande tabela *hash*, e é basicamente uma estrutura que consegue mapear cada chave e associar a um valor (NEVES; BERNARDINO, 2015). Devido à forma como está organizada, este tipo de base de dados é de fácil implementação, permitindo assim que o acesso aos dados seja muito rápido utilizando a sua chave, principalmente em sistemas escaláveis, aumentando também a sua disponibilidade (NEVES; BERNARDINO, 2015). SCHREINER; DUARTE; SANTOS MELLO (2015) afirmam que este modelo de dados é o mais simples, segurando um conjunto de pares de chave-valor, sendo o valor acessado através da chave. Um valor pode manter um conteúdo simples ou complexo, mas este conteúdo não pode ser consultado. FREITAS apresenta uma alternativa neste modelo para agrupar um conjunto de chave-valor utilizando a estrutura *hash* que permite o uso de um conjunto de pares chave-valor como se fossem valores rotulados. O primeiro componente do *hash*, a chave, seria usado como nome do campo e o segundo componente como o seu valor agregado.

Neste trabalho, adotou-se o Redis (REDISLABS, 2016a; KUZNETSOV; POSKONIN, 2014), que é um SGBD *open-source* que dá suporte ao modelo chave-valor. Além disso, Redis suporta atomicidade e isolamento, e os dados são mantidos na memória principal. Os valores podem conter não só *strings*, mas também listas e outras estruturas de dados, além das operações normais de leitura, escrita e exclusão de dados por chave. Para permitir o armazenamento persistente, Redis fornece mecanismos diferentes, o mecanismo-padrão armazena periodicamente instantâneos obtidos a partir da memória principal utilizando o *snapshot*.

Todos os comandos em uma transação são serializados e executados sequencialmente. Não pode acontecer de um pedido emitido por outro cliente interromper a execução de uma outra operação. Isso garante que os comandos sejam executados como uma única operação isolada (REDISLABS, 2016b). Redis oferece diferentes opções de persistência.

2.2 Avaliação de Desempenho

O desempenho é um critério fundamental na concepção, aquisição e uso de sistemas computacionais. O objetivo dos provedores de sistemas de informação é obter o melhor desempenho para determinado custo. Para atingir esse objetivo, estes profissionais necessitam, ao menos, conhecer a terminologia de avaliação de desempenho e suas técnicas devem ser capazes de indicar as exigências de seus sistemas de desempenho e comparar diferentes alternativas para encontrar aquela que melhor atenda às suas necessidades (BUKH; JAIN, 1992).

Para a avaliação de desempenho, um conjunto de métricas deve ser escolhido. Uma maneira de preparar este conjunto é listar os serviços oferecidos pelo sistema. Para cada solicitação de serviço feita ao sistema, há vários resultados possíveis. Por exemplo, um banco de dados oferece o serviço de responder a consultas. Quando executada uma consulta, ele pode responder corretamente, ele pode responder de forma incorreta ou não responder. Se o sistema executa o serviço corretamente, o seu desempenho é medido pelo tempo necessário para realizar o serviço, a taxa na qual o serviço é executado, e os recursos consumidos durante a execução do serviço. BUKH; JAIN (1992) atribui a estas três métricas o rótulo de capacidade de resposta.

A maioria dos sistemas oferece mais de um serviço, e, portanto, o número de métricas cresce proporcionalmente. Para muitas métricas, o valor médio é o mais importante. No entanto, o efeito da variabilidade deve ser considerado, pois pode degradar significativamente a produtividade.

LILJA (2004) apresenta vários objetivos tipicos para avaliação de desempenho:

- **Comparar alternativas:** Na escolha de determinado sistema podem existir várias alternativas. Além disso, em cada sistema podem existir diversas possibilidades de configuração que de maneira geral podem afetar tanto o custo quanto o desempenho da solução desejada. Comparar alternativas pode fornecer informações quantitativas a respeito de quais configurações são melhores em condições específicas;

- **Determinar o impacto de uma característica:** Na concepção ou alteração sistemas, muitas vezes é necessário conhecer o impacto dessa nova caracteristica adicionada;
- **Ajuste do sistema:** Neste caso, utilizamos para encontrar a combinação de parametros ideais do sistema para a obtenção de sua melhor performance na métrica desejada;
- **Identificar o desempenho relativo:** O desempenho de um sistema de computador, geralmente, apenas tem significado no contexto da comparação a algum outro sistema ou configuração de outro sistema. Outro objetivo pode ser quantificar o desempenho em relação a uma expectativa; e
- **Definir expectativas:** Através da análise de desempenho, podem ser inferidas as capacidades de uma próxima linha de sistemas de computadores;

O esforço para a execução da análise de desempenho deve levar em consideração o custo da tomada de decisão errada.

De acordo com ARAUJO (2009); BUKH; JAIN (1992), a medição de desempenho consiste essencialmente na monitoração do sistema enquanto está sob ação de uma carga de trabalho. Para adquirir resultados representativos, a carga de trabalho deve ser cuidadosamente selecionada. Essa carga é utilizada nos estudos de desempenho, pode ser real ou sintética. Embora a carga de trabalho real seja uma boa escolha por representar de forma fiel o sistema, ocasionalmente esta opção não é a desejável. Isso acontece quando o tamanho da carga não é considerável, e também quando esses dados recebem muitas perturbações ou, até mesmo, por questões de facilidade de acesso destes. Devido a esses motivos, as cargas de trabalho sintéticas têm sido utilizadas como *benchmark* e programas.

2.2.1 *Design of Experiments* (DoE)

Experiências são realizadas para investigar as características de um sistema. A partir da avaliação dos experimentos, respostas ou características são encontradas. A resposta deve ser analisada de forma adequada de acordo com os objetivos dos experimentos. As fontes que afetam os experimentos são os fatores. Não é trivial identificar todos os fatores e dar valores corretos porque podem existir muitos fatores e distribuições afetadas pelas variações do ambiente (ANALYTIC METHODS FOR DESIGN PRACTICE, 2007). No entanto, o conhecimento técnico do especialista é fundamental, esta ferramenta não pode ser considerada uma solução autônoma. O conhecimento do especialista a respeito do problema junto a esta técnica possibilita bons projetos de experimentos, ou seja, permite menos pontos com menor custo e respostas baseadas em inferência estatística.

De maneira geral, os experimentos são realizados para:

- mensurar ou avaliar valores característicos, sem analisá-los estatisticamente;

- identificação através da experimentação de fatores que são significativos para as respostas e determinação de quão grande é o seu impacto. Os resultados das experiências são analisados estatisticamente;
- identificação estatística dos fatores com pouca influência; e
- determinação dos valores dos fatores significativos. Em seguida, a condição ideal pode ser encontrada.

Utilizar a metodologia de *Design of Experiments* (DoE) é uma forma de reduzir o número de experiências a serem realizadas sem influenciar a qualidade dos resultados (FRIGON; MATHEWS, 1997). A inovação desta metodologia é que ela propõe uma experimentação fatorial, em que os fatores investigados variam simultaneamente, o processamento matemático simples dos dados permite uma avaliação bastante precisa dos efeitos de fatores e interações (DASCALESCU et al., 2008).

Design of Experiments determina a alocação e método de experiências para satisfazerem aos objetivos. Vários fatores são determinados por análise dos resultados experimentais.

COLEMAN; MONTGOMERY (1993) propõe alguns passos para a realização do *Design of Experiments*:

- caracterização do problema;
- escolha dos fatores de influência e níveis;
- seleção das variáveis de resposta;
- determinação de um modelo de planejamento de experimento;
- condução do experimento;
- análise dos dados; e
- conclusões e recomendações.

O planejamento fatorial é indicado para a fase inicial do procedimento experimental, quando há necessidade de se definir os fatores mais importantes e estudar os efeitos sobre a variável de resposta escolhida, a Figura 2.2 a seguir representa o processo de transformação (experimentação). O tipo mais simples de fatorial envolve apenas dois fatores, A e B, em que existem *a* níveis para o fator A e *b* níveis para o fator B (DOUGLAS C. MONTGOMERY, 2013).

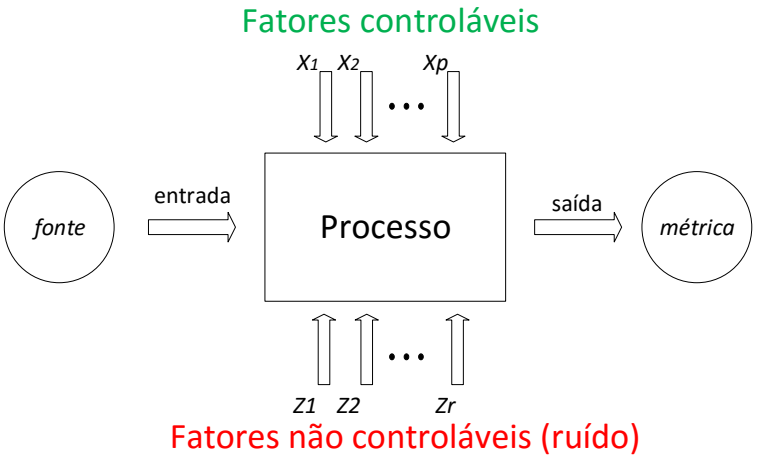

Figura 2.2: Fatores Processo

O experimento possui n réplicas, e cada repetição contém todas as combinações de tratamento de ab. O fatorial de dois fatores é um projeto inteiramente ao acaso. Apresenta-se na Tabela 2.1 o arranjo de dados fatorial de dois fatores, onde, supondo que A e B são fatores fixos, ou seja, os a níveis do fator A e os b níveis dos fatores B são selecionados pelo experimentador, as inferências são restritas apenas a esses níveis.

Tabela 2.1: Fatorial dois Fatores.

		Fator *B*					
		1	**2**	...	***b***	**Total**	**Média**
Fator *A*	1	$y_{111}, y_{112}, \dots, y_{11n}$	$y_{121}, y_{122}, \dots, y_{12n}$		$y_{1b1}, y_{1b2}, \dots, y_{1bn}$	$\overline{y}_{1\cdot\cdot}$	$\overline{y}_{1\cdot\cdot}$
	2	$y_{211}, y_{212}, \dots, y_{21n}$	$y_{221}, y_{222}, \dots, y_{22n}$		$y_{2b1}, y_{2b2}, \dots, y_{2bn}$	$\overline{y}_{2\cdot\cdot}$	$\overline{y}_{2\cdot\cdot}$
	...						
	a	$ya11, ya12, \dots, y_{a1n}$	$y_{a21}, y_{a22}, \dots, y_{a2n}$		$y_{ab1}, y_{ab2}, \dots, y_{abn}$	$\overline{y}_{a\cdot\cdot}$	$\overline{y}_{a\cdot\cdot}$
Totais		$y_{\cdot 1\cdot}$	$y_{\cdot 2\cdot}$		$y_{\cdot b\cdot}$	$y_{\dots}$	
Médias		$\overline{y}_{\cdot 1\cdot}$	$\overline{y}_{\cdot 2\cdot}$		$\overline{y}_{\cdot b\cdot}$		$\overline{y}_{\dots}$

Fonte: DOUGLAS C. MONTGOMERY (2013)

FERREIRA (2013) descreve a *analysis of variance* (ANOVA) como uma metodologia estatística desenvolvida inicialmente com o objetivo de comparar amostras ou tratamentos. O exemplo representado na Tabela 2.1 pode ser utilizado para testar hipóteses sobre os principais efeitos do fator de A e B, e a interação AB. É utilizado para verificar se existem diferenças significativas entre as médias dos tratamentos, que sejam resultado dos efeitos dos tratamentos.

Em um experimento cada observação y_{ij} pode ser decomposta conforme o modelo a seguir:

$$Y_{ij} = \mu + \tau_i + \xi_{ij}$$

Onde $i = 1, ...I$ e $j = 1, ...J$, y_{ij} é a observação do i-ésimo tratamento na j-ésima unidade experimental, μ é o efeito constante (média geral), τ_i é o efeito do i-ésimo tratamento, ξ_{ij} é o erro associado ao i-ésimo tratamento na j-ésima unidade experimental.

Para realizar a análise de variância a variável de interesse deve ter distribuição normal e os grupos devem ser independentes, onde as hipóteses H_0 - verdadeira quando não existe diferença significativa entre as médias populacionais, e a hipótese alternativa H_1 - verdadeira quando ao menos uma das médias populacionais é diferente das demais, são testadas.

Para aplicar a análise de variância é necessário calcular a estatística de teste F, composta pela razão de duas estimativas, variância entre amostras e variância dentro de amostras (2.1) (TRIOLA, 2005).

$$F = \frac{\text{variância entre amostras}}{\text{variância dentro das amostras}} \tag{2.1}$$

Se o valor obtido pela equação da estatística de teste F for maior que o valor crítico tabelado pela distribuição F, referente à estatística teórica, rejeita-se a hipótese nula. Isto indica que existe pelo menos uma diferença significativa entre as médias que estão sendo comparadas. A rejeição ou não da hipótese nula também pode ser verificada através do cálculo de P. Se o valor P for menor ou igual a 0,05, para um grau de confiança de 95%, rejeita-se a hipótese nula. Se o valor P for maior que 0,05, não se rejeita a hipótese nula, não existindo assim evidências suficientes para afirmar que existem diferenças significativas entre as médias analisadas.

2.3 *Benchmark*

O *benchmarking* é amplamente utilizado para avaliação de sistemas computacionais e existe para uma grande variedade de níveis de abstração. O emergente número de sistemas junto a falta de padrões de comparação de desempenho, dificulta o entendimento das relações entre os sistemas e sua a carga de trabalho (COOPER et al., 2010).

BORAL; DEWITT (1984) explica que o *benchmark* pode ser aplicado para comparar com precisão a eficiência de diferentes produtos, uma vez que é padronizado e os seus testes são invariáveis e bem definidos. De acordo com CIFERRI, um *benchmark* deve conter as seguintes características: ser relevante para a aplicação-alvo a qual representa; ser portátil entre diferentes arquiteturas; ser escalável, podendo ser executado em diferentes sistemas computacionais e; ser, na medida do possível, simples de entender para manter a sua credibilidade.

2.3.1 Yahoo! Cloud Serving Benchmark (YCSB)

A proposta do Yahoo! Cloud Serving Benchmark (YCSB) é facilitar as comparações de desempenho da nova geração de sistemas (COOPER et al., 2010). Entender as implicações

das decisões de desempenho para um dado tipo de aplicação é um desafio, pois existem muitas variáveis que influenciam de forma individuais ou combinadas o desempenho em finalidades distintas da tecnologia. O YCSB tem como objetivo criar um padrão de referência comparativa para auxiliar na avaliação dos diferentes sistemas em nuvem.

YCSB tem sido o padrão aberto para a avaliação de desempenho comparativo em armazenamentos de dados NoSQL. Existem muitos fatores para decisão de qual o armazenamento de dados a ser usado para aplicações de produção, incluindo recursos básicos, modelo de dados, e as características de desempenho em um determinado tipo de carga de trabalho, é fundamental ter a capacidade de comparar vários armazenamentos de dados de forma inteligente e objetiva para que decisões de arquitetura possam ser realizadas (CLOUDERA, 2016).

Conforme ilustrado na Figura 2.3, a estrutura consiste em um cliente gerador de carga de trabalho que cobre aspectos como trabalhos de leituras/escritas pesadas. Para desenvolver o núcleo do pacote do YCSB, foi examinada uma variedade de sistemas e aplicações com o fim de identificar os tipos fundamentais de carga de trabalho. Alguns sistemas podem ser altamente otimizados para leitura, mas não para escrita, outros podem ser para escrita, mas não para atualizações. Essas cargas são uma variação do mesmo tipo básico de aplicativo, nesta aplicação existe uma tabela básica de registros, com determinados campos, cada registro é identificado por uma chave primaria. Os valores de cada campo são uma sequência aleatória de caracteres ASCII de comprimento determinado (COOPER et al., 2010; ABRAMOVA; BERNARDINO, 2013).

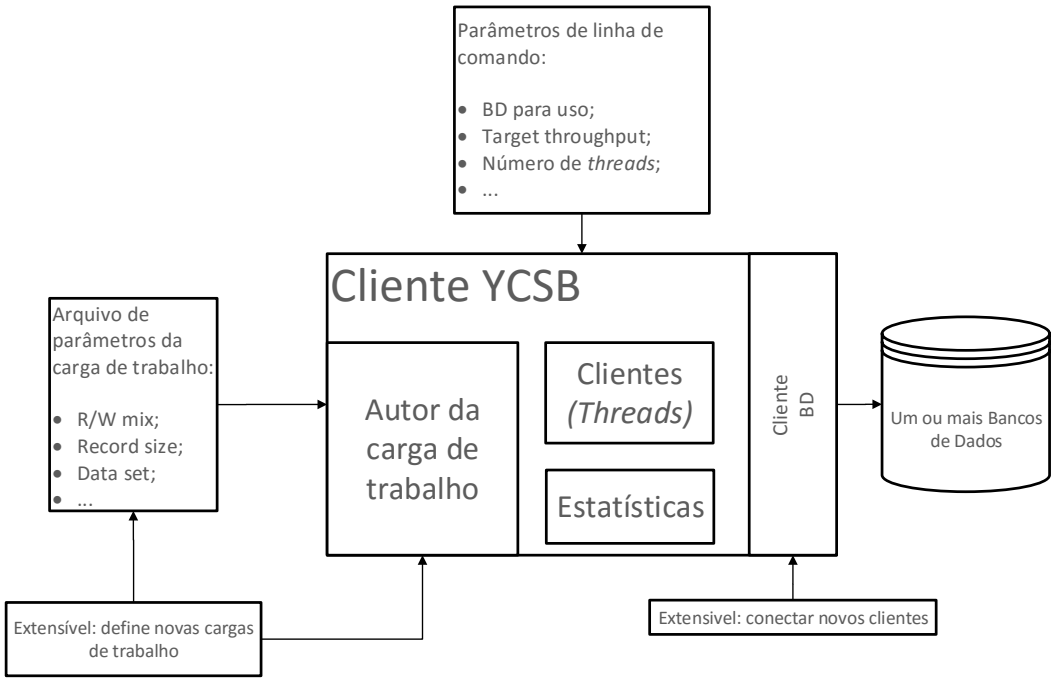

Figura 2.3: Arquitetura Yahoo! Cloud Serving Benchmark

No artigo de COOPER et al. (2010) no qual foi realizada a apresentação do Yahoo! Cloud Serving Benchmark, são propostos dois níveis de referência de avaliação, o desempenho e a escalabilidade de sistemas. Utilizou-se o desempenho como base de comparação para a avaliação do consumo de energia dos SGBDs. O nível de desempenho do *benchmark* incide sobre a latência das solicitações, quando o banco de dados está sob carga. A latência é muito importante em sistemas, uma vez que existe espectativa para o seu bom desempenho, como, por exemplo, o carregamento de uma página *web*. No entanto, existe uma relação entre a latência e a taxa de transferência, em uma dada configuração de *hardware*, com a elevação da quantidade de carga, a latência de pedidos individuais também aumenta devido às requisições em disco, CPU, rede, e assim por diante. Normalmente *designers* de aplicativos devem especificar a respeito do melhor custo-beneficio entre uma latência aceitável e provisão de servidores suficientes para alcançar a taxa de transferência desejada.

Um sistema com melhor desempenho vai atingir a latência desejada e rendimento com menos servidores. O nível de desempenho do *benchmark* pretende caracterizar essa compensação para cada sistema de banco de dados através da medição da latência à medida que aumenta a produção, até o ponto em que o sistema de banco de dados está saturado e o rendimento para de aumentar. No entanto, este trabalho avalia o consumo de energia elétrica em função do tempo de execução de determinadas cargas de trabalho uma vez que o objeto de estudo é o consumo de energia e não a latência de forma isolada.

2.3.2 *Core Package* YCSB

O YCSB disponibiliza um conjunto de núcleos de cargas de trabalho para avaliar diferentes aspectos do desempenho de um sistema, chamado *Core Package* YCSB. Cada carga de trabalho representa uma mistura particular de operações de leitura e escrita, o tamanho dos dados, as distribuições de solicitação, e assim por diante, e pode ser usado para avaliar os sistemas em um ponto específico. Um pacote inclui vários tipos de cargas de trabalho, examina uma fatia mais ampla do espaço de atuação, enquanto o pacote de núcleo examina vários eixos interessantes de desempenho. É possivel desenvolver pacotes próprios, quer através da definição de um novo conjunto de parâmetros de carga de trabalho, ou, se necessário, escrevendo em código Java (WIKI-YCSB, 2016).

Para desenvolver o *Core Package*, uma variedade de sistemas e aplicações foi observada para identificar os tipos fundamentais de aplicações de cargas de trabalho em sistemas de dados em nuvem. Nesta aplicação, há uma tabela de registros, cada um com um ou mais campos. Cada registro é identificado por uma chave primária, que é uma *string* como "user234123". Cada campo é chamado *field0*, *field1*, e assim por diante. Os valores de cada campo são uma sequência aleatória de caracteres ASCII de comprimento L. As operações são realizadas da seguinte forma:

- **Escrita**: Insere um novo registro.
- **Atualização**: Atualiza um registro, substituindo o valor de um campo.

- **Leitura**: Lê um registro, um ou outro campo escolhido aleatoriamente ou todos os campos. O padrão realiza a leitura em todos os campos.

2.3.3 Distribuições

O gerador de carga de trabalho deve realizar muitas escolhas aleatórias, ao gerar os dados a serem carregados no *benchmark* (escrita, atualização, leitura). Essas decisões são regidas por distribuições que utilizam diferentes maneiras de criar esta aleatoriedade dada uma probalidade do dado.

O YCSB tem várias distribuições, a distribuição *zipfian* utilizada neste trabalho, seleciona um item de acordo com a distribuição *zipfian*, customizada pelo Yahoo! para simular um comportamento real das características de acesso a banco de dados. Por exemplo, ao escolher um registro, alguns serão mais populares (o *head* da distribuição) enquanto a maioria dos registros serão impopulares (a *tail*).

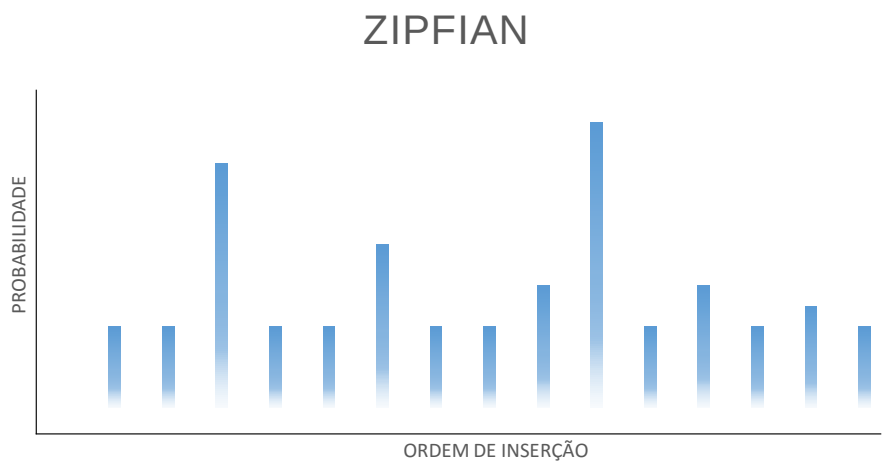

Figura 2.4: Distribuição zipfian YCSB

Tendo em vista que em se tratando de probabilidade os resultados podem variar, a Figura 2.4 apenas exemplifica visualmente a distribução zipfian. Os eixos horizontais na figura representam os itens que podem ser escolhidos na ordem de inserção, enquanto as barras verticais representam a probabilidade de o conjunto de caracter ser escolhido.

3

Metodologia e Ambiente de Experimentação

Este capítulo apresenta a metodologia proposta para a análise do consumo de energia em sistemas de gerenciamento de banco de dados NoSQL. Primeiramente, a explicação da metodologia é apresentada, a qual é baseada na técnica *Design of Experiments*. Em seguida, a configuração dos experimentos é detalhada, incluindo os parâmetros adotados para o gerador de carga de trabalho. Apresenta o embasamento para o cálculo de consumo de energia elétrica e apresenta o *framework* desenvolvido para a medição do consumo e sua forma de trabalho. O fluxo dos *scrips* de automação é apresentado. Em cada Seção é explicada a motivação para escolha dos principais parâmetros nos componentes envolvidos dos experimentos.

3.1 Metodologia

A metodologia é baseada no *Design of Experiments* (DOUGLAS C. MONTGOMERY, 2013), no qual foi adotado um experimento l^k fatorial com r replicações. Consideraram-se dois fatores ($k = 2$) com três níveis ($l = 3$): (i) SGBD - Cassandra, MongoDB, Redis; (ii) comando - escrita, leitura, atualização. Além disso, são consideradas três cargas de trabalho diferentes (1.000 operações, 10.000 operações, 100.000 operações), em um total temos vinte e sete (27) tratamentos para avaliação, com a carga de trabalho gerada por Yahoo! Cloud Serving Benchmark (YCSB) (COOPER et al., 2010), as métricas de interesse são o consumo de energia em joule (J) e tempo de execução em (ms) para cada carga de trabalho (por exemplo, 1.000 operações). A carga de trabalho pode ser um fator explícito para a proposta desta avaliação, mas seria a principal fonte de variação na análise dos resultados, e junto ao ruído de medição global (ou seja, erros aleatórios) impediriam uma comparação mais refinada das fontes de variação.

O fluxograma apresentado na Figura 3.1 denota a sequência das atividades realizadas.

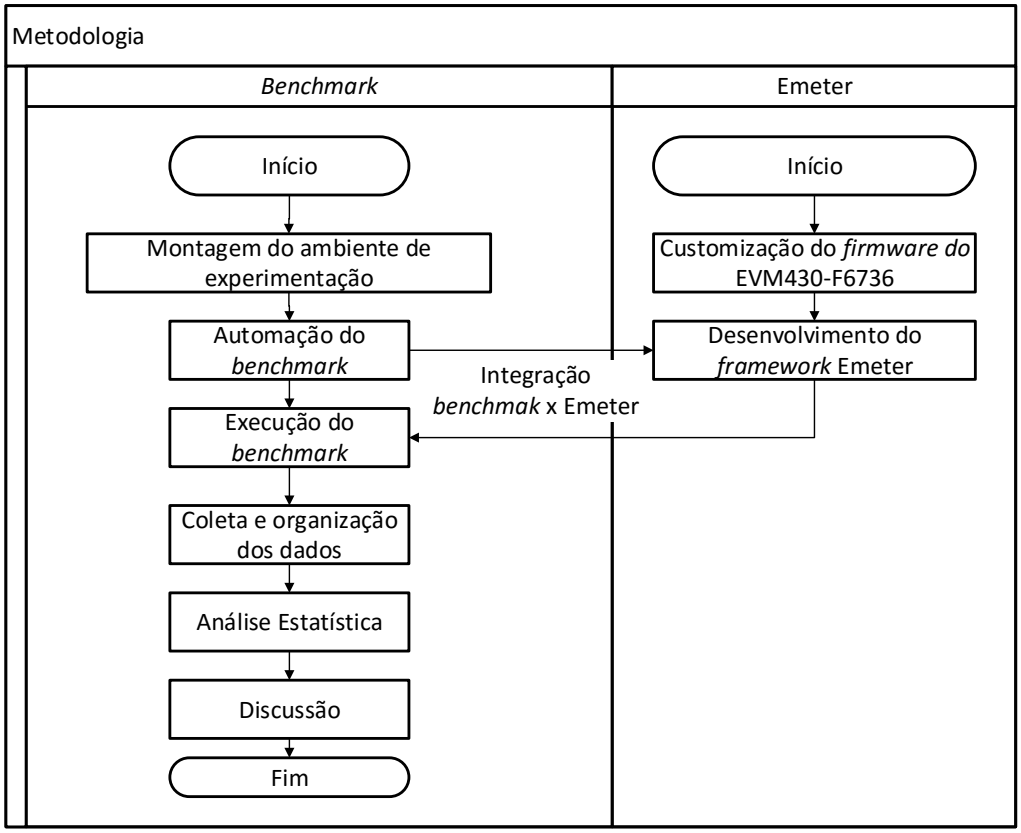

Figura 3.1: Metodologia da dissertação

Diante do número de soluções NoSQL disponíveis se fez necessário entender as categorias destes SGBDs mais populares e selecionar um SGBD de representatividade de cada uma destas categorias. Os SGBDs selecionados foram o Cassandra 2.0.15 (orientado a colunas), MongoDB 2.0.6 (orientado a documentos) e Redis 2.4.14 (chave-valor). O acerto destas escolhas é corroborado pelo IT (2016), o qual elenca os três SGBDs selecionados (Cassandra, MongoDB e Redis) entre os dez SGBDs de maior interesse no mundo. Em seguida, foram selecionados os comandos (leitura, escrita e atualização) por serem os mais utilizados nas comparações encontradas na literatura COOPER et al. (2010); ABRAMOVA; BERNARDINO (2013); FLORATOU et al. (2012); NEVES; BERNARDINO (2015). As cargas de trabalho a serem experimentadas obedeceram a um incremento de 10 vezes a partir da menor carga (1.000, 10.000 e 100.000) levando em consideração o *hardware* da máquina hospedeira dos SGBDs.

De posse do objeto do estudo se fez necessário buscar um gerador de carga de trabalho compatível com SGBDs, comandos e cargas de trabalho selecionados, e que obtivesse boa reputação no meio acadêmico. Deste modo, foi selecionado o *Yahoo Cloud Serving Benchmark* (YCSB) que atendeu a todos os requisitos do estudo e forneceu um bom suporte técnico em sua utilização (COOPER, 2016). O sistema operacional selecionado foi o Debian 7.8 (DEBIAN, 2015), onde foi utilizado para instalação uma versão *netinst*[1], esta escolha foi motivada pelo

[1]Contém apenas a quantidade mínima de software para começar a instalação e obter os outros pacotes através da internet.

fornecimento de suporte de instalação nativo a todos os SGBDs objeto deste trabalho.

Após a seleção do ambiente, realizou-se a montagem de um protótipo em ambiente virtual para validar a arquitetura dos experimentos, configuraram-se os acessos e em seguida executados os primeiros testes de funcionamento.

Com o ambiente virtual validado foi realizada a montagem do ambiente real de experimentação. Após a execução dos primeiros testes foram avaliados os resultados. Após a obtenção de resultados válidos, ou seja, resultados que não obtiveram falhas em sua execução nos distintos tratamentos, foi constatado, devido ao repetidos passos em distintos espaços de tempo, a necessidade da automação do processo do *benchmark*, foi desenvolvido um *script* para esta automação. Em paralelo a estas etapas foi criado um *framework* denominado Emeter, para a aferição do consumo de energia elétrica que será explicado a seguir. Com todo o ambiente validado montou-se o ambiente de produção para a execução do *benchmark*. A automação da execução do processo possibilitou uma massiva quantidade de repetições de cada cenário, com o intuito da obtenção de resultados que representassem uma distribuição normal.

Após a execução desses procedimentos foi necessário o desenvolvimento de outro conjunto de *scripts* para a coleta e organização dos dados gerados pelo YCSB e Emeter, que foram concentrados em um banco de dados relacional. Após a coleta e organização dos dados foi realizada a análise estatística e com base nos resultados desta análise o confronto da análise com as características dos SGBDs estudados.

3.2 Arquitetura Básica

A arquitetura básica do experimento, conforme observado na Figura 3.2, é composta pelo *hardware System on Chip* (SoC) de metrologia EVM430-F6736 (TEXAS-INSTRUMENTS, 2015a), uma máquina física hospedeira dos SGBDs sob configuração Core 2 Duo CPU T5450 1.66 GHz com 2 GB de RAM executando Debian 7.8 (Linux) com sistema de arquivos EXT4, e outro computador físico Core 2 CPU 1.86 GHz 2 GB de RAM que executa o software Emeter e hospeda a máquina virtual do gerador de carga de trabalho YCSB 0.2.0.

O EVM430-F6736, alimentado por uma rede elétrica externa, provê esta energia a máquina fisica hospedeira dos SGBDs, armazenando informações a respeito do consumo de energia total deste servidor em seu *buffer*. A coleta destas informações é realizada pelo *software* Emeter que armazena estas informações na própria máquina fisica, a máquina virtual com o YCSB gera a demanda de trabalho para o servidor do SGBD, coleta e armazena as informações de cada execução no próprio servidor virtual, todo o fluxo de trabalho é controlado por *script*, que sincroniza a execução das ações sinalizando em *timestamp*[2]. Antes de cada execução do *script* todos os horários foram checados. De igual forma foram checados os serviços em execução onde apenas os serviços essenciais estavam iniciados durante a execução dos experimentos.

[2]Cadeia de caracteres denotando a data e hora que certo evento ocorreu. A cadeia é geralmente apresentada num formato consistente, permitindo fácil comparação entre duas marcas temporais distintas.

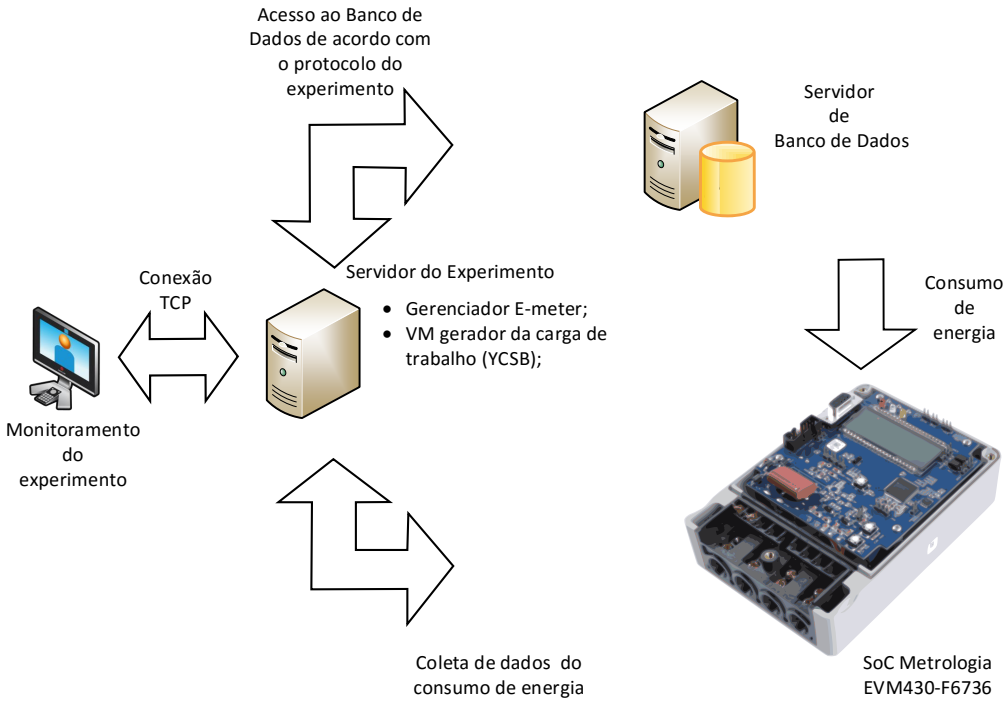

Figura 3.2: Arquitetura Básica

3.3 Configuração YCSB

Para realizar este *benchmark*, necessita-se de um gerador de carga de trabalho que responda a dois requisitos: definir o conjunto de dados carregando no banco de dados e executar operações contra o conjunto de dados aferindo o desempenho (tempo de execução). O consumo de energia é aferido por outro *framework* que será explicado nas próximas seções.

Nos experimentos foram utilizados dez campos por registro, definido por padrão pelo YCSB, sendo o tamanho de cada campo 100 *bytes*. Na leitura todos os campos eram acessados, a proporção das operações foi definida de acordo com os comandos do *benchmark*, por exemplo para leitura foram executados 100% de leitura; a escrita e a atualização seguiram com a mesma proporção, o número de operações seguiu de acordo com o cenário (1.000 operações, 10.000 operações ou 100.000 operações). Foram utilizadas dez *threads* para simular um ambiente de acessos simultâneos. Para cada execução do *benchmark* foi realizado um *load* que realiza a preparação do ambiente, o parâmetro *-target* não foi definido, desta forma o YCSB não limitou a quantidade de operações por segundo utilizando a máxima capacidade de vazão do ambiente.

Este trabalho utilizou a distribuição *zipfian*. Esta distribuição é baseada no algoritimo *zipfian* e foi customizada pelo Yahoo! para que os itens populares fossem espalhados por toda a *keyspace* do banco de dados conforme visto em COOPER et al.. Esta distribuição é a definida em quatro dos seis *workloads* padrão do YCSB (*workloada*, *workloadb*, *workloadc*, *workloadd*, *workloade*, *workloadf*). Na prática há uma simulação de alguns picos de informações populares,

durante o *benchmark* ele pega itens fora de um jogo com alguns itens sendo mais prováveis de serem escolhidos, em seguida, outros. Dessa forma, é quase certo que são atualizados muitos registros diferentes, mas há uma boa chance de que certos itens sejam atualizados várias vezes. Para cada execução *run* foi realizada uma preparação do ambiente *load*, isto garante por exemplo que ao executar a leitura o dado esteja no banco de dados.

3.4 Consumo de Energia

Para realizar o cálculo do consumo aproximado de cada amostra, foi utilizada a soma de Riemann para realizar uma aproximação ao consumo de energia real (3.1), representa a energia total consumida em uma execução do experimento ao longo do tempo.

$$E = \sum_{i}^{T} P_i * \Delta t \tag{3.1}$$

O consumo de energia P no intervalo $_i$ foi fornecido pelo *hardware* de metrologia EVM430-F6736 e coletado pelo *framework* Emeter no intervalo de um segundo.

3.5 *Framework* Emeter

O *framework* Emeter combina o *hardware* de metrologia EVM430-F6736 e o *software* Emeter desenvolvido para a medição do consumo de energia elétrica. Neste trabalho, foram apresentadas as suas características de funcionamento e o processo de coleta do consumo de energia.

3.5.1 EVM430-F6736

O *hardware* EVM430-F6736 é um módulo de avaliação de consumo de energia elétrica monofásico com base no dispositivo MSP430F6736 TEXAS-INSTRUMENTS (2015b). A precisão dos resultados é na faixa de 0,1%, com biblioteca de *software* de metrologia programável.

Durante a operação normal, o processo em *background* notifica o processo em primeiro plano, através de um sinalizador de status toda vez que um conjunto de dados está disponível para processamento. Esta estrutura de dados consiste na acumulação do consumo de energia durante um segundo. Isto é equivalente a uma acumulação de 50 ou 60 ciclos de amostras de dados sincronizados com o sinal de tensão de entrada. Além disso, um contador de amostra mantém o controle de quantas amostras foram acumuladas ao longo do período. Esta contagem pode variar de acordo como o software é sincronizado com a frequência da rede de entrada. As amostras do conjunto de dados consistem de corrente processada, potência ativa e reativa.

3.5.2 Emeter JAVA

Para coletar, armazenar e exibir em tempo real o consumo informado pelo EVM430-F6736 foi desenvolvido o *framework* Emeter. Através de uma *interface* de comunicação serial (RS-232), a cada intervalo de um segundo este *framework* realiza uma solicitação de leitura do consumo do EVM, nove leituras são realizadas sendo elas, tensão, corrente, potência ativa, potência reativa, frequência, fator de potência, energia acumulada, e, por último, tempo.

Nos experimentos apenas foi considerada como execução válida aquela onde a quantidade de coletas do *framework* na execução de um cenário foi igual ao intervalo de tempo em segundos da execução do *benchmark* informada pelo YCSB, respeitando a restrição de coleta de consumo de um segundo exemplo; em uma execução de dez segundos e meio o Emeter deve ter coletado onze segundos.

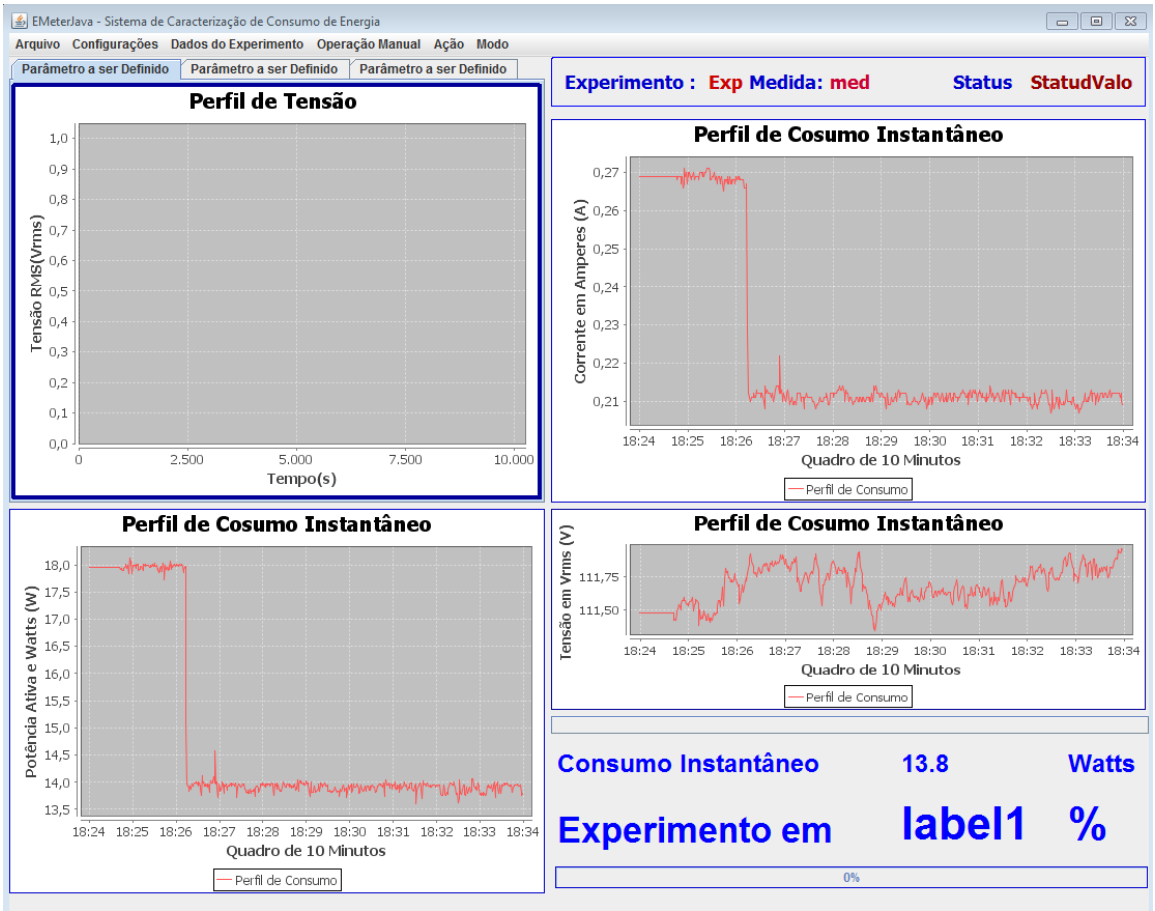

Figura 3.3: *Graphical User Interface* Emeter JAVA

A persistência dos dados coletados pode ser realizada em banco de dados postgreSQL (POSTGRESQL, 2016) ou em arquivo de texto. Neste trabalho optou-se pela opção de arquivo de texto para minimizar a influência de outros processos também na coleta do consumo. A inicialização, nomenclatura, acompanhamento do consumo instantâneo e finalização do processo de leitura de consumo podem ser realizadas através de sua *Graphical User Interface* (GUI)

conforme exemplificado na Figura 3.3, ou através de um *socket*[3]. A opção por conduzir as chamadas do Emeter através do *socket* foi devido à automação do *benchmark* explicado nos parágrafos seguintes.

3.6 Automação *Benchmark*

Cada execução de um dado cenário implica um conjunto de procedimentos que devem ser realizados de forma padronizada minimizando falhas experimentais, considerando que foram avaliados 27 tratamentos distintos e que cada tratamento foi executado cem vezes de forma válida, a condução manual do trabalho seria inviável. Foram desenvolvidos os *scripts* necessários para a execução do *benchmark*, coleta e processamento dos dados. Todos os *scripts* foram desenvolvidos utilizando a linguagem Python 2.7 (FOUNDATION, 2016).

Conforme visto anteriormente, o *framework* Emeter possui uma *interface socket* para as chamadas de suas funções, o *script* emeter.py possui as funções de inicialização e finalização da leitura do consumo, a inicialização faz a chamada da função de nomenclatura que auxilia no acompanhamento em tempo real dos experimentos através da *interface* gráfica e inicia a coleta do consumo.

O fluxograma apresentado na Figura 3.4 apresenta os passos realizados. O *script* comando-nomeSGBD.py realizou a execução do *benchmark* por SGBD, e antes da chamada deste *script* sempre foi checado o sincronismo de tempo nos servidores e a existência de processos desnecessários em execução como ,por exemplo, um serviço de outro SGBD. Em cada cenário a execução consiste nos seguintes passos: carregar os vetores com os números de operações (1.000, 10.000 e 100.000 operações), os comandos (escrita, leitura e atualização) e as chamadas do YCSB (carga e execução); inicia a iteração dos parâmetros nos vetores; realiza a chamadas do emeter.py passando como parâmetro a nomenclatura da execução em função do cenário executado e o comando *startApp* para inicio da leitura do consumo; armazena na memória a variável início do tipo *datetime*; ajusta a chamada do YCSB atribuindo os parâmetros de execução; executa o *benchmark*; armazena na memória a variável fim do tipo *datetime*; insere no final do arquivo resultado da execução do *benchmark* os valores das variáveis início e fim; espera vinte segundos para a próxima execução; finaliza o Emeter através do emeter.py e comando *stopApp*.

O resultado de cada execução do *benchmark* gerou um arquivo contendo dados de *performance* como o tempo de execução. Todos os dados foram enviados a um banco de dados para posterior análise, por meio do *script* extrator.py, a conversão dos dados gerados através do Emeter foi realizada através do file_to_postgres_emeter.py.

[3]Ponto final de uma ligação de comunicação de duas vias entre dois programas em execução na rede. Está ligado a um número de porta de modo que a camada TCP pode identificar os dados destinados á aplicação.

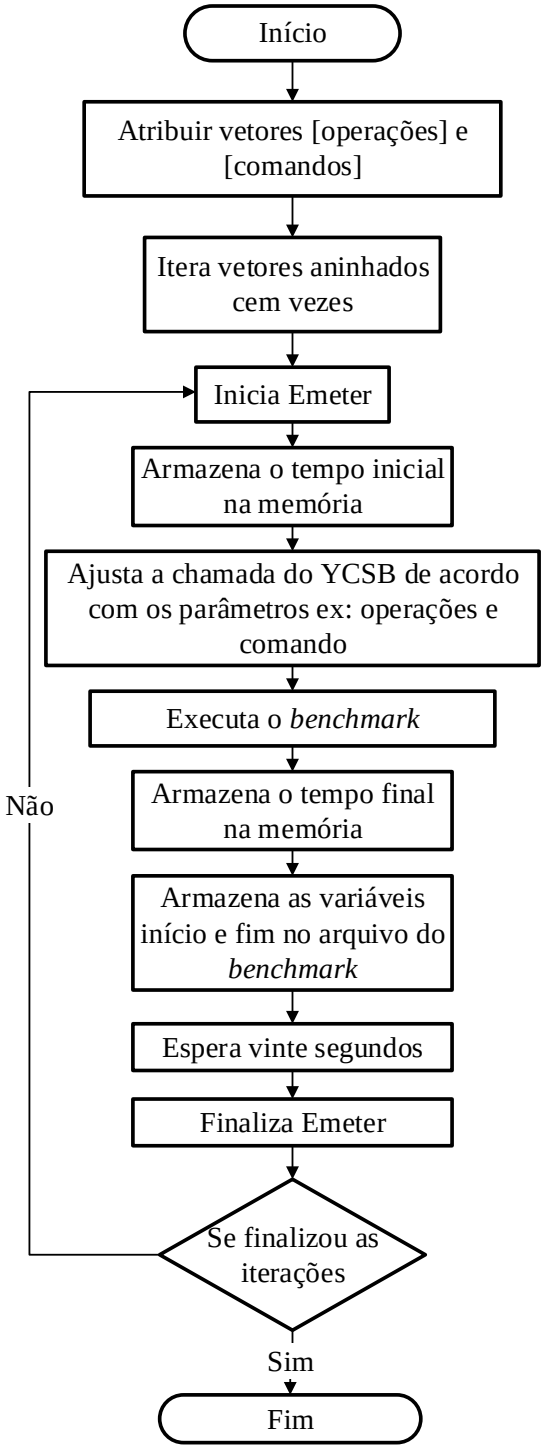

Figura 3.4: Processo *benchmark*

4

Resultados Experimentais

Este capítulo apresenta os resultados obtidos dos experimentos realizados conforme o DoE apresentado no Capítulo 3. Foram consideradas as cargas de trabalho 1.000, 10.000 e 100.000 operações. O número de operações é corresponde a quantidade de um mesmo procedimento que é lançado contra o SGBD por execução. A seleção de três cargas de trabalho obedece a um incremento de dez vezes, é importante, uma vez que possibilita a observação da escalabilidade do SGBD. Foi adotada para cada SGBD a configuração-padrão, e as métricas de interesse são o tempo de execução e o consumo de energia. Além disso, cada registro inserido no banco de dados possui 1 KB (valor-padrão definido pelo YCSB), e 10 *threads* foram adotadas para imitar um ambiente real com acessos simultâneos. O *hardware* EVM430-F6736 e o *framework* Emeter realizaram a leitura do consumo. Os resultados dos experimentos consideraram apenas o consumo de energia no intervalo de tempo de execução do YCSB. Verificou-se a existência de *outliers*[1], a análise dos resultados destes afastamentos apontou um desempenho atípico para operações em NoSQL. Comportamento que não pode ser justificado, por exemplo, com operações em *background* do SGBD, uma vez o sistema praticamente parou de executar operações em considerável período de tempo, indicando para estas execuções problemas na geração da demanda de trabalho. Foram removidos os *outliers* conforme Tabela 4.1, Tabela 4.2 e Tabela 4.3, utilizando a amplitude interquartil, onde são apresentados os resultados gerais para as amostras consideradas em cada tratamento (N) e o intervalo de confiança (IC). Os resultados são apresentados para cada carga de trabalho usando a análise ANOVA(DOUGLAS C. MONTGOMERY, 2013). Em seguida, a correlação é discutida para ambas as métricas seguida pelas observações gerais.

[1] Observação que apresenta um grande afastamento das demais da série.

Tabela 4.1: Tempo de Execução e Consumo de energia para Cassandra

			tempo de execução (ms)		consumo de energia (J)	
comando	carga	N	média	IC 95%	média	IC 95%
escrita	1000	83	4044,8	3951; 4138,6	70,891	68,998; 72,785
escrita	10000	77	14043	13842; 14243	280,79	275,48; 286,1
escrita	100000	75	106341	104748; 107933	2627,8	2591,7; 2663,8
leitura	1000	83	8851	7763; 9938	192,3	167,6; 217,1
leitura	10000	77	31230	27556; 34903	869,8	781,9; 957,8
leitura	100000	75	183991	156628; 211353	5526	4842; 6210
atualização	1000	83	3480,5	3404,2; 3556,7	57,822	56,585; 59,06
atualização	10000	77	9414,6	9257,2; 9572	166,36	164; 168,71
atualização	100000	75	61307	60359; 62254	1253,3	1235,46; 1271,13

Tabela 4.2: Tempo de Execução e Consumo de energia para MongoDB

			tempo de execução (ms)		consumo de energia (J)	
comando	carga	N	média	IC 95%	média	IC 95%
escrita	1000	83	4662,4	4575,4; 4749,5	76,153	74,733; 77,572
escrita	10000	77	20664	20244; 21084	347,72	341,19; 354,25
escrita	100000	75	174525	170292; 178758	2949,7	2881,2; 3018,1
leitura	1000	83	3764,3	3714,7; 3813,9	59,489	58,619; 60,359
leitura	10000	77	13834	13601; 14067	224,66	221,14; 228,17
leitura	100000	75	105858	103430; 108286	1697,2	1658,7; 1735,8
atualização	1000	83	3950,8	3895,9; 4005,6	64,881	63,995; 65,767
atualização	10000	77	14914	14692; 15136	250,78	247,37; 254,19
atualização	100000	75	112142	109961; 114324	1984,3	1946,2; 2022,4

Tabela 4.3: Tempo de Execução e Consumo de energia para Redis

			tempo de execução (ms)		consumo de energia (J)	
comando	carga	N	média	IC 95%	média	IC 95%
escrita	1000	83	3934,8	3879,9; 3989,6	61,721	60,646; 62,795
escrita	10000	77	34549	34232; 34865	654,26	649,09; 659,43
escrita	100000	75	338793	335865; 341721	5938,9	5885,6; 5992,2
leitura	1000	83	2027,6	1991; 2064,2	31,179	30,619; 31,74
leitura	10000	77	17130	16954,4; 17305,7	269,6	266,97; 272,22
leitura	100000	75	167199	165800; 168598	2634,38	2614,75; 2654,01
atualização	1000	83	1999,9	1968,9; 2030,9	30,986	30,524; 31,448
atualização	10000	77	17001	16856,4; 17145,6	268,08	265,93; 270,23
atualização	100000	75	165032	163830; 166233	2933,6	2902,7; 2964,5

4.1 Resultado dos Tratamentos

Tabela 4.4 descreve todos os fatores e suas interações ($fonte$) que impactam significativamente no tempo de execução (estatística F -$Fstat.$- e p-$value$). Dependendo da carga de trabalho (*carga.*), o fator pode ter um impacto diferente sobre a métrica (*var.%*), alguns SGBDs variam seu comportamento para a quantidade de operações executadas. Com o aumento da carga

de trabalho o comando (*Com*) eleva sua influência em relação ao tempo de execução. Em geral, a interação entre SGBD e comando (*SGBD * Com*) é a principal fonte de variação, mas, na carga de trabalho 100.000, a influência da interação muda devido ao comportamento do Cassandra. Resultados semelhantes são obtidos para o consumo de energia Tabela 4.5. No entanto, para a carga de trabalho 100.000 operações, uma pequena diferença ocorre devido ao súbito aumento do consumo de energia do Cassandra em função do tempo de execução. Outras fontes de variação são devido a erros experimentais (isto é, ruído nas medições), e eles são reduzidos com o aumento da carga de trabalho. Tal situação está relacionada com o intervalo de tempo necessário para a execução de uma carga de trabalho, o que diminui, por exemplo, à influência de outros serviços do sistema operacional. As explicações seguintes são baseadas nos resultados obtidos com o procedimento de Tukey (DOUGLAS C. MONTGOMERY, 2013) (ANOVA *post-hoc*).

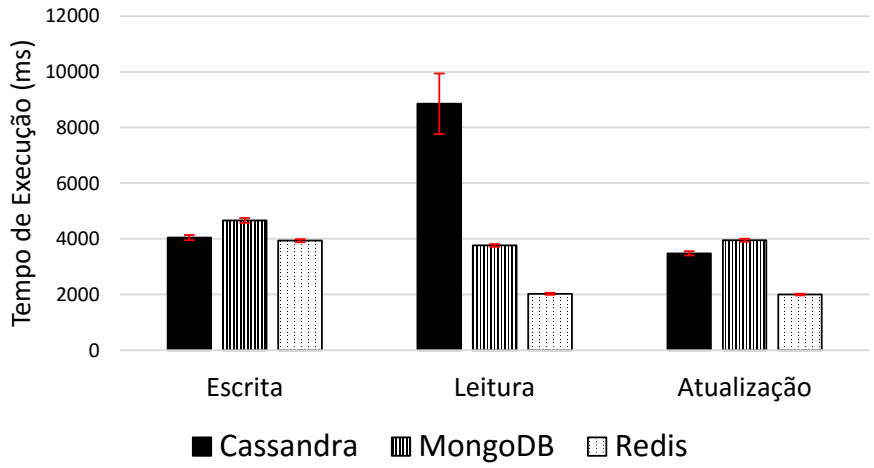

Figura 4.1: Média Tempo de Execução 1.000 Operações

Tabela 4.4 e Tabela 4.5 respectivamente fornecem os resultados para a ANOVA (grau de significância $\alpha = 0,05$) para o tempo de execução e consumo de energia. As Figuras 4.1 a 4.6 apresentam os resultados considerando os valores médios.

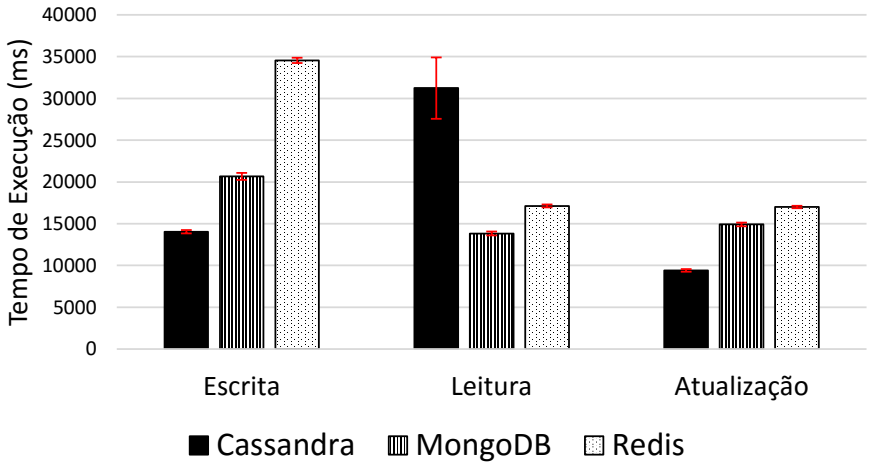

Figura 4.2: Média Tempo de Execução 10.000 Operações

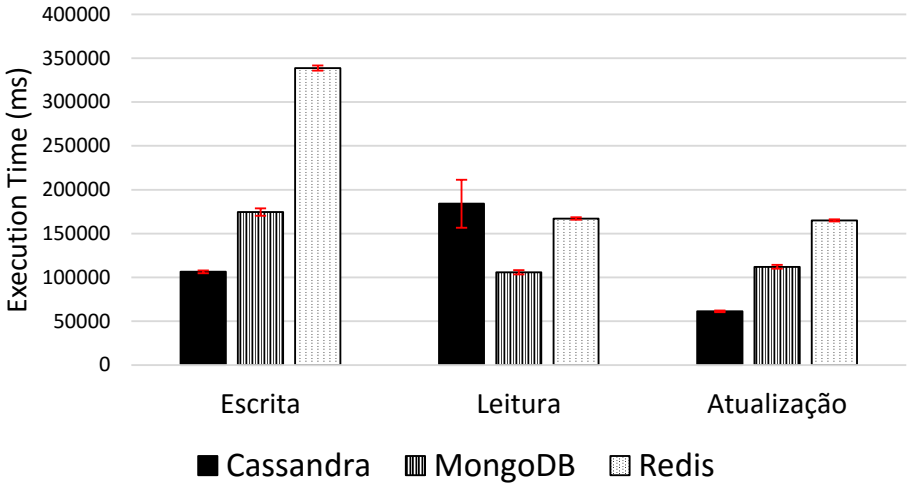

Figura 4.3: Média Tempo de Execução 100.000 Operações

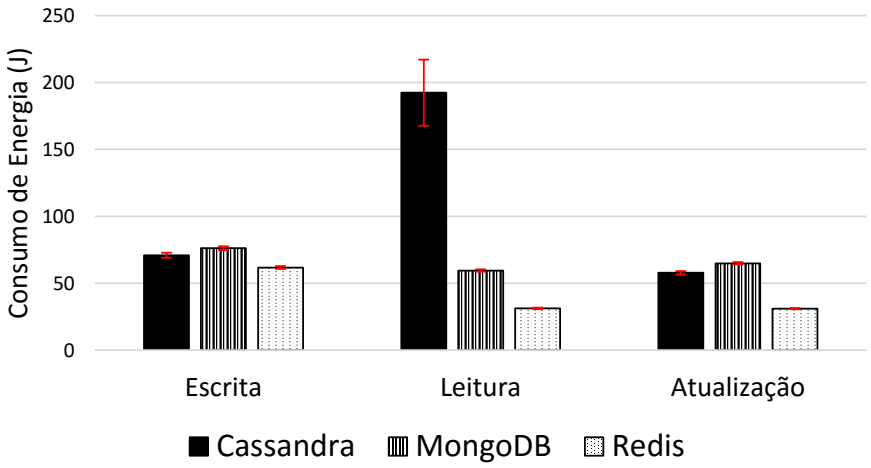

Figura 4.4: Média Consumo de Energia 1.000 Operações

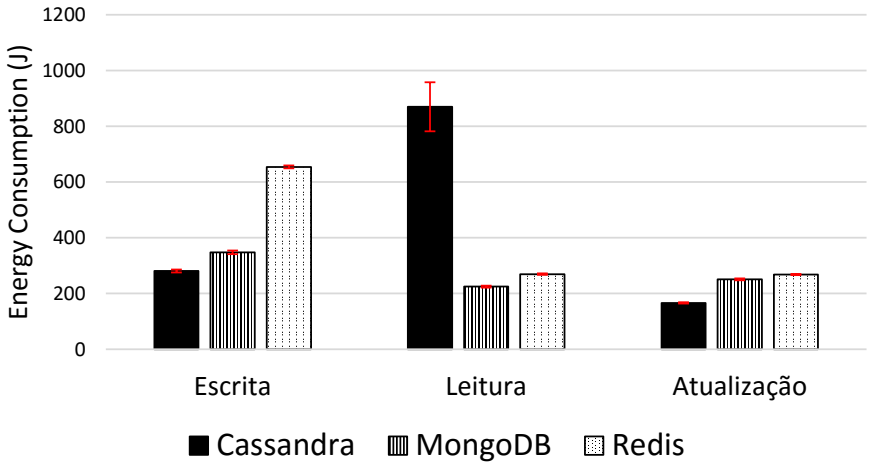

Figura 4.5: Média Consumo de Energia 10.000 Operações

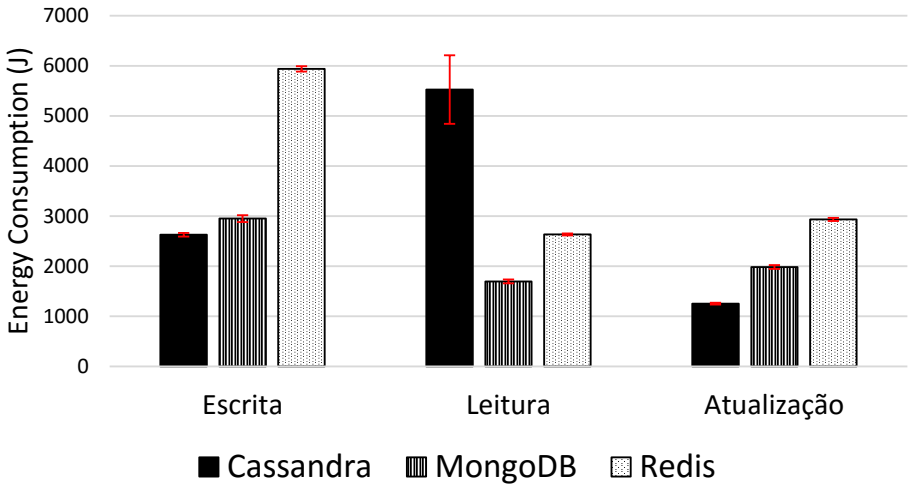

Figura 4.6: Média Consumo de Energia 100.000 Operações

Tabela 4.4: ANOVA Tempo de Execução

carga. 1.000				carga. 10.000				carga. 100.000			
fonte	var.%	F stat.	p-value	fonte	var.%	F stat.	p-value	fonte	var.%	F stat.	p-value
SGBD	23,15	230,41	<0,001	SGBD	7,97	84,31	<0,001	SGBD	30,80	453,88	<0,001
Com	9	89,55	<0.001	Com	16,95	179,28	<0,001	Com	20,32	299,42	<0,001
SGBD*Com	30,78	153,16	<0,001	SGBD*Com	42,74	226,04	<0,001	SGBD*Com	26,28	193,62	<0,001
Erro	37,07			Erro	32,33			Erro	22,60		

Tabela 4.5: ANOVA Consumo de Energia

carga. 1.000				carga. 10.000				carga. 100.000			
fonte	var.%	F stat.	p-value	fonte	var.%	F stat.	p-value	fonte	var.%	F stat.	p-value
SGBD	23,24	243,66	<0,001	SGBD	7,54	99,73	<0,001	SGBD	13,28	147,94	<0,001
Com	9,78	102,55	<0,001	Com	15,71	207,92	<0,001	Com	16,62	185,14	<0,001
SGBD*Com	31,36	164,45	<0,001	SGBD*Com	51,59	341,34	<0,001	SGBD*Com	40,19	223,83	<0,001
Erro	35,62			Erro	25,16			Erro	29,90		

A Figura 4.1 representa o tempo de execução médio para a carga de trabalho de 1.000 operações. No que diz respeito ao comando de escrita, mesmo Cassandra obtendo menor desempenho, 2,79% em relação ao Redis, e o MongoDB obtendo um desempenho 15,27% inferior ao Cassandra, as diferenças entre os valores não são estatisticamente significativas e, portanto, não se encontram evidências estatísticas suficientes para rejeitar a igualdade entre os tempos de execução. Para os comandos de leitura, a diferença é significativa, e Cassandra leva muito mais tempo do que outros SGBDs para concluir esta carga de trabalho. Por exemplo, é, respectivamente, 57,47% e 77,09% mais lento do que o MongoDB e Redis (que forneceu o menor tempo de execução). Em relação à atualização, Redis também proporciona o melhor valor para essa carga de trabalho com diferença estatística significativa, completando a execução deste comando em 57,46% do tempo do Cassandra e 50,62% do tempo do MongoDB. Analisando a aptidão por comando do SGBD, para o Cassandra não demonstram-se argumentos estatísticos para afirmar que escrita e atualização são diferentes; a leitura obteve diferença significativa sendo 54,30% mais lenta que escrita e 60,68% que atualização. MongoDB, semelhante ao Cassandra, não possui diferença estatística entre escrita e atualização; a leitura foi diferente, sendo 23,86% mais rápido que escrita e 4,95% mais rápido que atualização. Redis apresenta um comportamento diferente dos outros SGBDs, leitura e atualização não são diferentes estatisticamente, escrita se destaca com um valor médio diferente sendo mais lento em 48,47% com relação à leitura e 49,17% à atualização.

As Tabelas 4.6 (a) e (b), 4.8(a) e (b) e 4.10(a) e (b) apresentam as comparações segundo o teste de Tukey, a coluna (Grupo) dessas tabelas, associa todos os resultados que não podem ser considerados diferentes segundo este procedimento, os resultados diferentes são sinalizados com letras distintas.

Avaliando os resultados contidos na Tabela 4.6 (a) correspondente ao teste de Tukey, há evidências de que os valores médios por SGBD são estatisticamente diferentes, Redis obteve melhor desempenho, seguido por MongoDB e Cassandra. Os comandos de leitura e atualização influenciaram o melhor desempenho do Redis; por outro lado, o menor desempenho do Cassandra foi fortemente influenciado pelo comando de leitura. Os comandos também possuem valores médios diferentes, leitura foi quem mais demandou tempo para sua conclusão enquanto a atualização foi o comando mais ágil.

Tabela 4.6: Tempo de Execução Tukey 1.000 Operações

(a)

SGBD	N	Média	Grupo
Cassandra	249	5458,75	A
MongoDB	249	4125,84	B
Redis	249	2654,10	C

(b)

SGBD X Comando	N	Média	Grupo
Cassandra Leitura	83	8850,95	A
MongoDB Escrita	83	4662,45	B
Cassandra Escrita	83	4044,82	B C
MongoDB Atualização	83	3950,76	B C
Redis Escrita	83	3934,80	B C
MongoDB Leitura	83	3764,31	C
Cassandra Atualização	83	3480,47	C
Redis Leitura	83	2027,59	D
Redis Atualização	83	1999,93	D

No contexto do consumo de energia para a carga de trabalho de 1.000 operações, os valores (Figura 4.4) apresentam diferença estatística e indicam uma correlação com o tempo de execução para a comparação entre SGBDs e entre comandos, uma vez que podem ser fornecidos argumentos semelhantes. Na Tabela 4.7 (b) observa-se que os valores para o comando de inserção também estão próximos, o teste de Tukey indica que essas diferenças entre SGBDs não são expressivas.

No que diz respeito ao comando de leitura, Cassandra fornece o maior consumo de energia, que é 69,06% maior que o MongoDB e 83,79% que Redis para esse comando. Redis tem o menor consumo de energia para atualizações sendo a diferença de 46,41% para Cassandra e 52,24% para o MongoDB. Essa diferença é significativa, MongoDB e Cassandra não têm valores médios que diferem significativamente entre eles neste cenário. Analisando a aptidão por comando do SGBD, a alteração do comportamento em função do tempo de execução foi apenas entre a leitura e escrita do MongoDB onde para o consumo não possui diferença significativa. De modo geral, Cassandra para 1.000 operações obteve menor eficiência energética, consumindo

61,41% mais que Redis e 37,53% mais que MongoDB Tabela 4.7 (a).

Tabela 4.7: Consumo de Energia Tukey 1.000 Operações

(a)

SGBD	N	Média	Grupo
Cassandra	252	107,033	A
MongoDB	252	66,841	B
Redis	252	41,295	C

(b)

SGBD X Comando	N	Média	Grupo
Cassandra Leitura	84	192,324	A
MongoDB Escrita	84	76,153	B
Cassandra Escrita	84	70,954	B
MongoDB Atualização	84	64,881	B
Redis Escrita	84	61,721	B
MongoDB Leitura	84	59,489	B
Cassandra Atualização	84	57,822	B
Redis Leitura	84	31,179	C
Redis Atualização	84	30,986	C

Considerando 10.000 operações (Figura 4.2), Redis parece ser menos escalável que outros SGBDs para o comando de inserção. Redis foi 59,35% mais lento que o MongoDB e 40,19% que o Cassandra, agora diferente da comparação da carga de trabalho de 1.000 operações todas as operações de escrita foram significativamente diferentes. Esta situação influencia o aumento do comando (*Com*) como uma fonte de variação na Tabela 4.4. Comparando com a carga de trabalho de 1.000 operações, tempo de execução da escrita Redis aumentou 88,61%, enquanto para Cassandra e MongoDB aumentou 71,20% e 77,44%, respectivamente.

O comportamento do comando de leitura entre os SGBDs permanece com diferença entre as médias (Tabela 4.8 (b)), Cassandra se mantém com o tempo de execução notavelmente maior, sendo 45,15% mais lento que Redis e 55,70% mais lento que MongoDB, percebe-se que o Redis, semelhante ao ocorrido com a escrita nesta carga de trabalho, também aparenta menor escalabilidade, comparando com a carga de trabalho de 1.000 operações, o tempo de execução da leitura Redis aumentou 88,16%, enquanto para Cassandra e MongoDB aumentou 71,66% e 72,79%, respectivamente.

Neste contexto, MongoDB fornece o melhor desempenho e melhorou sua eficiência nesse aumento de carga de trabalho. Para atualização, Cassandra oferece melhor tempo de execução, MongoDB e Redis não obtiveram médias diferentes entre eles, são respectivamente 36,87% 44,62% mais lentos que Cassandra. Neste comando Redis também foi menos escalável, comparando com a carga de trabalho de 1.000 operações, tempo de execução Redis aumentou 88,24%, enquanto para Cassandra e MongoDB aumentou 63,03% e 73,51%, respectivamente.

Analisando a aptidão por comando do SGBD, para o Cassandra, diferente da carga de trabalho de 1.000 operações, todos os comandos são diferentes, outra vez leitura é o comando de maior utilização de tempo sendo respectivamente 55,03% e 69,84% mais lento que escrita e atualização. MongoDB agora apresenta leitura e atualização sem diferença significativa, escrita

demonstrou diferença sendo 27,83% mais lento que atualização e 33,05% mais lento que leitura. Redis não apresentou diferença entre leitura e atualização, escrita dentre os comandos ficou evidenciada com o maior tempo de execução, sendo 50,42% mais lento que leitura e 50,79% mais que atualização.

Avaliando os resultados contidos na Tabela 4.8 (a), os valores médios por SGBD são estatisticamente diferentes, houve uma mudança na classificação do desempenho geral em relação à carga de trabalho de 1.000 operações, agora em 10.000 operações MongoDB obteve melhor desempenho seguido por Cassandra e Redis, a característica que influenciou esse resultado foi o fato de que Cassandra e MongoDB melhoraram seu desempenho sensivelmente enquanto o Redis manteve o mesmo desempenho proporcionalmente a 1.000 operações. Os comandos permanencem com valores médios diferentes, agora a escrita foi quem mais demandou tempo para sua conclusão enquanto a atualização continua sendo o comando mais ágil. O menor desempenho da escrita neste contexto sofre forte influência do Redis.

Tabela 4.8: Tempo de Execução Tukey 10.000 Operações

(a)

SGBD	N	Média	Grupo
Redis	231	22893,2	A
Cassandra	231	18228,9	B
MongoDB	231	16470,6	C

(b)

SGBD X Comando	N	Média	Grupo
Redis Escrita	77	34548,5	A
Cassandra Leitura	77	31229,5	B
MongoDB Escrita	77	20663,9	C
Redis Leitura	77	17130,0	D
Redis Atualização	77	17001,0	D
MongoDB Atualização	77	14914,1	D E
Cassandra Escrita	77	14042,6	E
MongoDB Leitura	77	13833,9	E
Cassandra Atualização	77	9414,6	F

Figura 4.5 mostra os valores de consumo de energia para essa carga de trabalho, que também apontam para a correlação entre o tempo de execução e o consumo de energia. Os resultados para o teste de Tukey na Tabela 4.9 (a) apresentam diferença no consumo geral entre SGBDs, agora leitura e escrita não podem ser considerados comandos diferentes, o que é verdadeiro para tempo de execução desta mesma carga de trabalho. Cassandra foi o SGBD que mais consumiu energia entre os SGBDs, o comando de leitura utilizou mais energia e não pode ser considerado diferente da escrita entre os comandos na carga de trabalho de 10.000 operações. Este impacto é influenciado pela interação (leitura Cassandra) que utilizou mais energia igualando estatisticamente a escrita geral.

Outras alterações de valores médios das interações, não possuem diferença significativa. Comparando com a carga de trabalho de 1.000 operações, o consumo de energia da escrita

Redis aumentou 90,57%, enquanto para Cassandra e MongoDB aumentou 74,75% e 78,10%, respectivamente. Para leitura, o incremento no consumo foi de 88,43% no Redis, enquanto para Cassandra e MongoDB aumentou 77,89% e 73,52%, respectivamente.

De modo geral, Cassandra para 10.000 operações obteve menor eficiência energética, mas nesta carga o MongoDB melhorou, Cassandra e Redis são respectivamente 37,49% e 30,94% mais lentos que MongoDB conforme Tabela 4.9 (a).

Tabela 4.9: Consumo de Energia Tukey 10.000 Operações

(a)

SGBD	N	Média	Grupo
Cassandra	225	438,993	A
Redis	225	397,311	B
MongoDB	225	274,386	C

(b)

SGBD X Comando	N	Média	Grupo
Cassandra Leitura	75	869,834	A
Redis Escrita	75	654,259	B
MongoDB Escrita	75	347,721	C
Cassandra Escrita	75	280,786	D
Redis Leitura	75	269,596	D
Redis Atualização	75	268,078	D
MongoDB Atualização	75	250,780	D
MongoDB Leitura	75	224,655	D E
Cassandra Atualização	75	166,359	E

Figura 4.3 mostra os valores médios para os tempos de execução em matéria da carga de trabalho para 100.000 operações. O comportamento em geral se assemelha à carga de trabalho 10.000 operações, o desempenho geral de todas as interações (*SGBD * Com*) segue um mesmo *ranking* de desempenho por comando, há alteração em relação à diferença significativa entre estas interações, algumas foram agrupadas como não diferentes significativamente. Observando o tempo médio, executando a escrita, Redis foi 48,49% mais lento que o MongoDB e 68,61% que Cassandra, que obteve o melhor rendimento. Esses resultados foram estatisticamente diferentes.

A execução deste comando por Redis tem o pior desempenho em todos os experimentos (superando o tempo de execução de comando de leitura para Cassandra). Verificando agora o rendimento em comparação com a carga de 10.000 operações, o aumento do tempo de execução para Redis, MongoDB e Cassandra foram de respectivos 89,80%, 88,16% e 86,79%. Neste contexto, Cassandra apresentou melhor escalabilidade, SGBD Redis perdeu sutilmente eficiência em relação aos outros SGBDs na escrita.

Executando o comando de leitura, diferente das cargas de trabalho anteriores, agora com 100.000 operações Cassandra melhorou o tempo de execução em relação ao Redis e MongoDB, nesta circunstância Cassandra não possui desempenho diferente de Redis e é apenas 36,69% mais lento que MongoDB. Confrontando com os resultados médios de 10.000 operações, Cassandra elevou apenas 83,03% ao mesmo tempo que Redis e MongoDB elevaram de modo respectivo 89,75% e 86,93% para a resposta da execução. Para esta situação MongoDB continua com

o menor tempo de execução. Cassandra oferece melhor tempo de execução no comando de atualização, MongoDB e Redis agora obtiveram médias estatisticamente diferentes entre eles, são respectivamente 45,33% e 62,85% mais lentos que Cassandra. Neste comando Redis também foi menos escalável, comparando com a carga de trabalho de 10.000 operações, o tempo de execução Redis aumentou 89,70%, enquanto para Cassandra e MongoDB o elevaram 84,64% e 86,70%, respectivamente.

Avaliando em 100.000 operações os comandos por SGBD, Cassandra, para o comando de leitura, continua utilizando o maior tempo de execução; atualização é o mais rápido, atualização e escrita são nesta ordem 66,68% e 42,20% mais lentos que leitura, estas médias são estatisticamente diferentes. MongoDB continua com leitura e atualização sem diferença significativa, escrita utilizou mais tempo, sendo 39,34% mais lento que leitura e 35,74% mais lento que atualização. Redis possui as médias de leitura e atualização semelhantes, não podendo ser consideradas diferentes; a escrita continua sendo o comando de maior custo sendo 50,75% mais lento que leitura e 51,29% mais lento que atualização.

Tabela 4.10: Tempo de Execução Tukey 100.000 Operações

(a)

SGBD	N	Média	Grupo
Redis	225	223674	A
MongoDB	225	130842	B
Cassandra	225	117213	C

(b)

SGBD X Comando	N	Média	Grupo
Redis Escrita	75	338793	A
Cassandra Leitura	75	183991	B
MongoDB Escrita	75	174525	B
Redis Leitura	75	167199	B
Redis Atualização	75	165032	B
MongoDB Atualização	75	112142	C
Cassandra Escrita	75	106341	C
MongoDB Leitura	75	105858	C
Cassandra Atualização	75	61307	D

A Figura 4.6 fornece um comportamento interessante. Ela também fornece evidências de correlação entre o tempo de execução e o consumo de energia, o consumo de energia em Redis é maior do que outros SGBDs. Da mesma forma, o tempo de execução Redis para os comandos de inserção são maiores que os comandos de leitura de Cassandra com diferença estatística significativa. Cassandra consumiu mais por tempo de execução e agora não possui diferença significativa em relação à escrita Redis. Tal comportamento indica que há mudança de variação (*var* : %) de cada um dos fatores para ambas as métricas.

Em particular, a contribuição do fator do comando aumenta, bem como o ruído na medição do consumo de energia Tabela 4.5. O teste de Tukey na Tabela 4.11 (a) apresenta diferença no consumo geral entre SGBDs, MongoDB se apresenta mais uma vez com o menor consumo de energia, Redis assume como o maior consumidor com forte influência da escrita

corroborado nas Tabelas 4.9 (b) e 4.9 (c), essa escrita é 50,33% maior que MongoDB e 55,75% maior que Cassandra. Cassandra tem seu maior consumo notoriamente na leitura sendo 52,45% maior que escrita e 77,32% maior que atualização. Entendendo a mudança do consumo de energia comparando a carga de trabalho de 10.000 operações nesta carga de 100.000 operações, o consumo de energia da escrita Redis aumentou 88,98%, enquanto para Cassandra e MongoDB o aumentou 89,31% e 88,21%, respectivamente.

O aumento do consumo de energia do Cassandra para a escrita foi maior do que o aumento do tempo de execução. Realizando a leitura o incremento no consumo foi de 89,77% no Redis, enquanto para Cassandra e MongoDB aumentou 84,26% e 86,76%, respectivamente. Este menor aumento de consumo de Cassandra influenciou para a equivalência de consumo com a escrita Redis. Para que atualização obtivesse os resultados médios diferentes entre os SGBDs, o que não havia acontecido em 10.000 operações, o Redis degradou sensivelmente sua eficiência elevando 90,86%, MongoDB e Cassandra elevaram apenas 87,36% e 86,72% respectivamente.

De modo geral, Redis para 100.000 operações obteve menor eficiência energética, Cassandra foi quem mais melhorou sua eficiência em detrimento das menores cargas, mas ainda não foi suficiente para assumir a primeira posição do MongoDB, Cassandra e Redis consomem respectivamente 29,50% e 42,37,45% mais energia que MongoDB, conforme Tabela 4.11 (a).

Tabela 4.11: Consumo de Energia Tukey 100.000 Operações

(a)

SGBD	N	Média	Grupo
Redis	225	3835,62	A
Cassandra	225	3135,65	B
MongoDB	225	2210,41	C

(b)

SGBD X Comando	N	Média	Grupo
Redis Escrita	75	1874,44	A
Cassandra Leitura	75	1601,32	A
MongoDB Escrita	75	972,36	B
Redis Atualização	75	883,82	B
Redis Leitura	75	810,03	B
Cassandra Escrita	75	780,15	B
MongoDB Atualização	75	642,60	C
MongoDB Leitura	75	577,26	C D
Cassandra Atualização	75	389,78	D

4.2 Correlação

Nesta seção é apresentada de maneira geral a correlação da variação dos escores nos eixos tempo de execução e consumo de energia.

As Figuras 4.7, 4.8 e 4.9 mostram a correlação entre o consumo de energia e o tempo de execução para cada comando (escrita, leitura e atualização) contemplando o SGBD Cassandra.

Para todos os comandos, observa-se forte correlação linear positiva, corroborada pelo coeficiente de determinação (R^2). Os valores respectivos são $0,9916$, $0,9861$ e $0,9811$ para escrita, leitura e atualização. A Figura 4.7 representa os valores para escrita, em que a equação $y = 0,0253x$ - $49,915$ representa esta correlação. A equação $y = 0,028x$ - $104,57$ representa a correlação para leitura (Figura 4.8), e $y = 0,0211x$ - $22,088$ para atualização (Figura 4.9). O coeficiente angular da reta (isto é, a primeira derivada) fornece uma informação interessante sobre o impacto da execução dos procedimentos sobre o consumo médio de energia. Para os comandos adotados, a atualização tem a energia mais eficiente (consumo / tempo de execução 0,0211 energia) e escrita é o mais próximo (0,0253). Leitura é a responsável pelo maior consumo que é representado pela inclinação 0,0280.

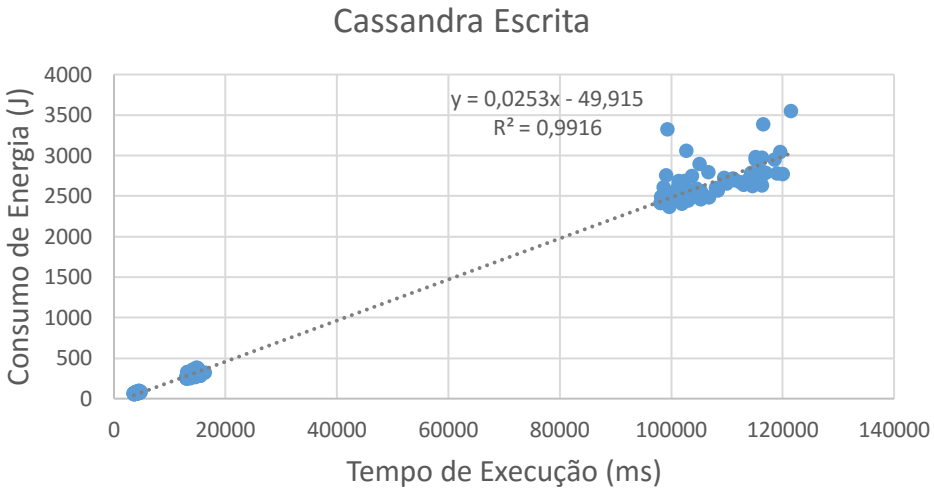

Figura 4.7: Correlação Tempo de Execução x Consumo de Energia Escrita Cassandra

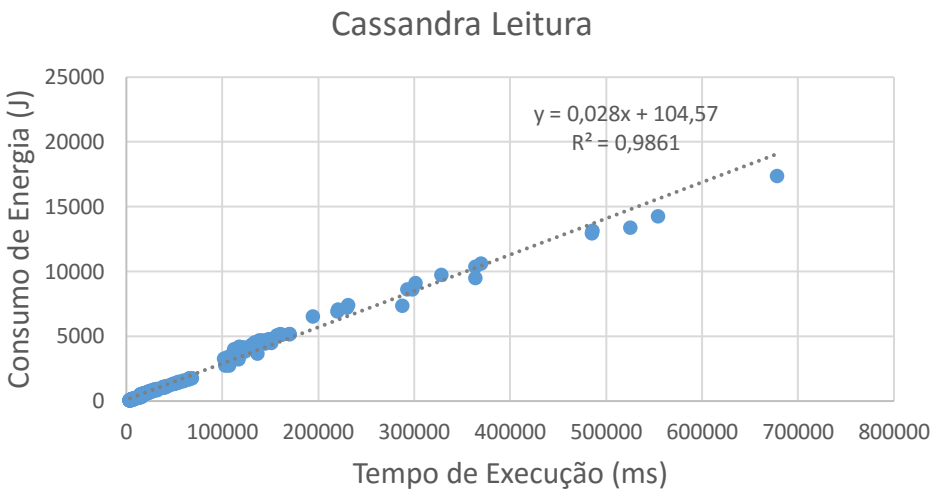

Figura 4.8: Correlação Tempo de Execução x Consumo de Energia Leitura Cassandra

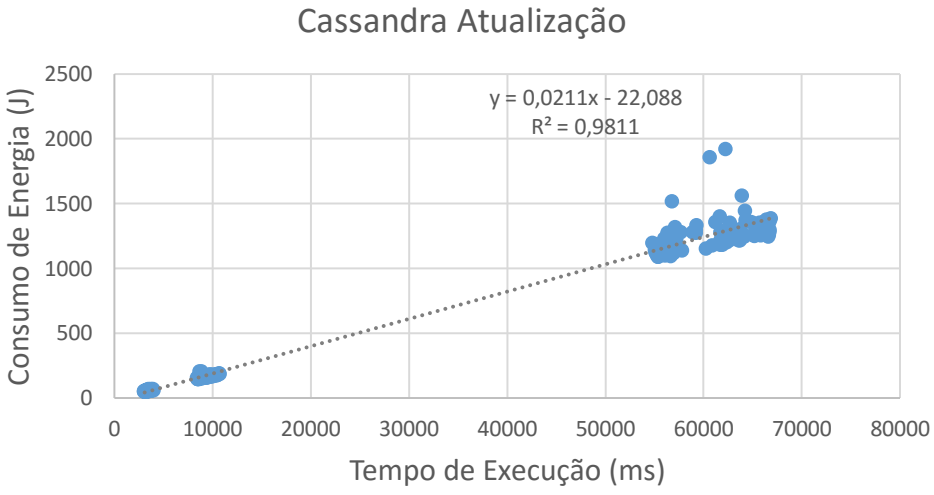

Figura 4.9: Correlação Tempo de Execução x Consumo de Energia Atualização Cassandra

As Figuras 4.10, 4.11 e 4.12 mostram a correlação entre o consumo de energia e o tempo de execução para cada comando (escrita, leitura e atualização) contemplando o SGBD MongoDB. Para todos os comandos, uma forte correlação (linear) foi obtida, informação corroborada pelo coeficiente de determinação (R^2). Os valores respectivos são $0,9999$, $0,9999$ e $0,9998$ para escrita, leitura e atualização. A Figura 4.10 representa os valores para escrita, em que a equação $y = 0,0169x$ - $1,5301$ representa esta correlação. A equação $y = 0,016x$ - $1,08$ representa a correlação para leitura (Figura 4.11), e $y = 0,0178x$ - $9,4464$ para atualização (Figura 4.12). De acordo com o coeficiente angular da reta, verifica-se que o MongoDB foi o SGBD mais eficiente considerando as somas destes coeficientes para os SGBDs avaliados. Para os comandos adotados, diferente do Cassandra, leitura tem o consumo mais eficiente (consumo / tempo de execução 0,016 energia) e escrita fica bem próximo com (0,0169). Atualização vem em seguida, representada pelo maior consumo que é demonstrado pela inclinação 0,0178.

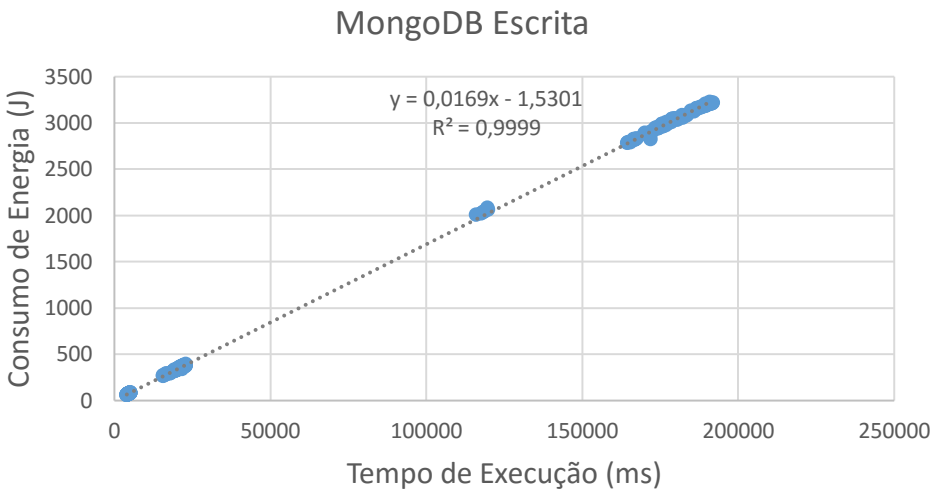

Figura 4.10: Correlação Tempo de Execução x Consumo de Energia Escrita MongoDB

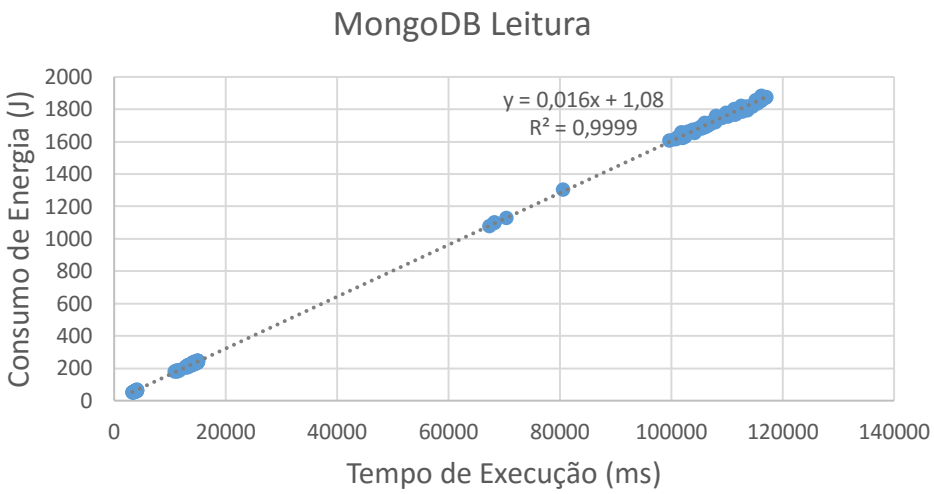

Figura 4.11: Correlação Tempo de Execução x Consumo de Energia Leitura MongoDB

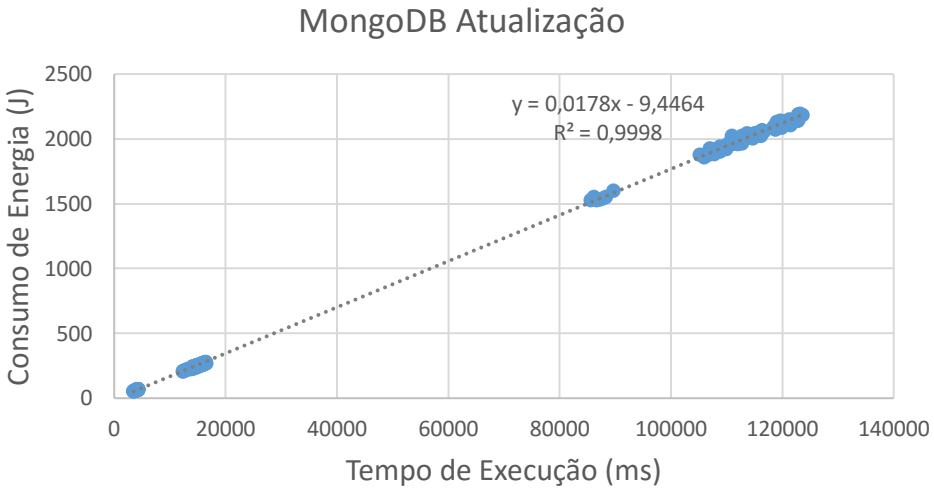

Figura 4.12: Correlação Tempo de Execução x Consumo de Energia Atualização MongoDB

Apresenta-se a correlação entre o consumo de energia e o tempo de execução para cada comando (escrita, leitura e atualização) contemplando o SGBD Redis nas Figuras 4.13, 4.14 e 4.15. Para todos os comandos, uma forte correlação (linear) foi obtida, os seguintes coeficientes de determinação (R^2) respectivamente são $0,9998$, $0,9992$ e $0,9992$ para escrita, leitura e atualização. A Figura 4.13 representa os valores para escrita, em que a equação $y = 0,0175x$ - $19,625$ representa esta correlação. A equação $y = 0,0158x$ - $0,2523$ representa a correlação para leitura (Figura 4.14), e $y = 0,0179x$ - $20,563$ para atualização (Figura 4.15). Para os comandos adotados, escrita tem o consumo menos eficiente (consumo / tempo de execução 0,0175 energia), leitura é a mais eficiente (0,0158). Atualização vem em seguida, representada pelo maior consumo que é demonstrado pelo coeficiente angular da reta 0,0178.

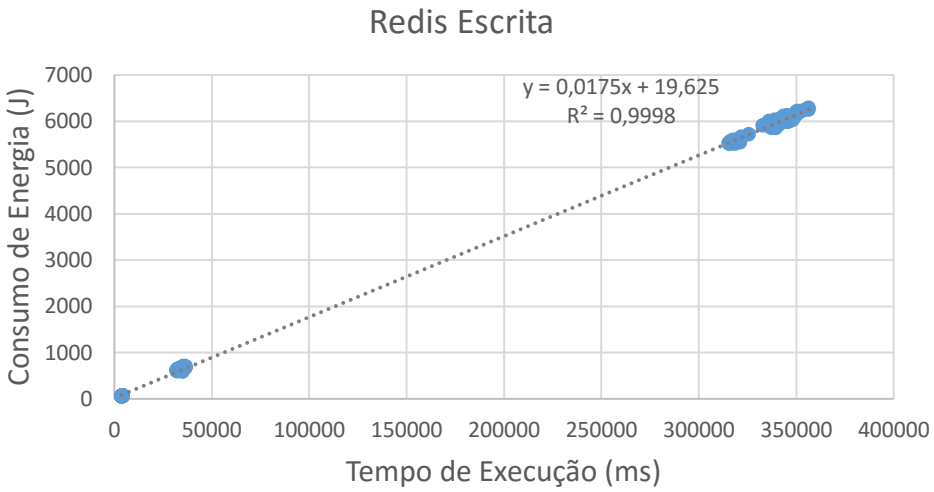

Figura 4.13: Correlação Tempo de Execução x Consumo de Energia Escrita Redis

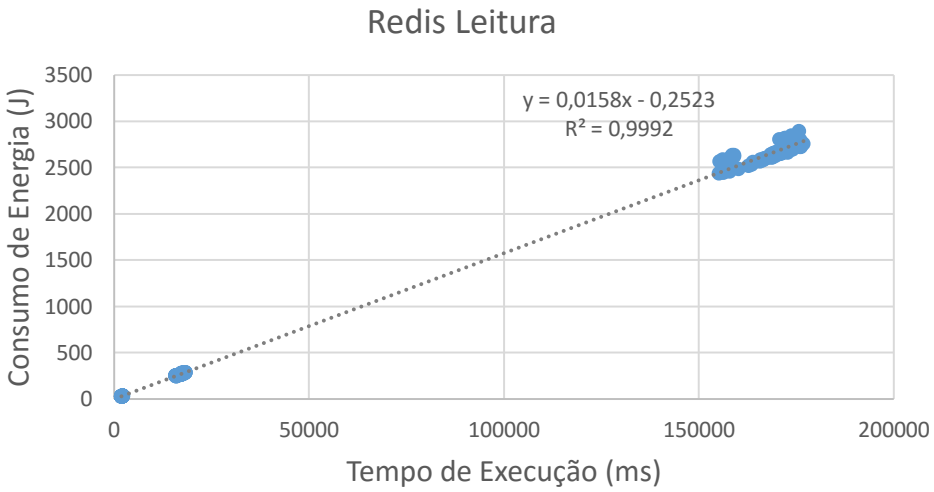

Figura 4.14: Correlação Tempo de Execução x Consumo de Energia Leitura Redis

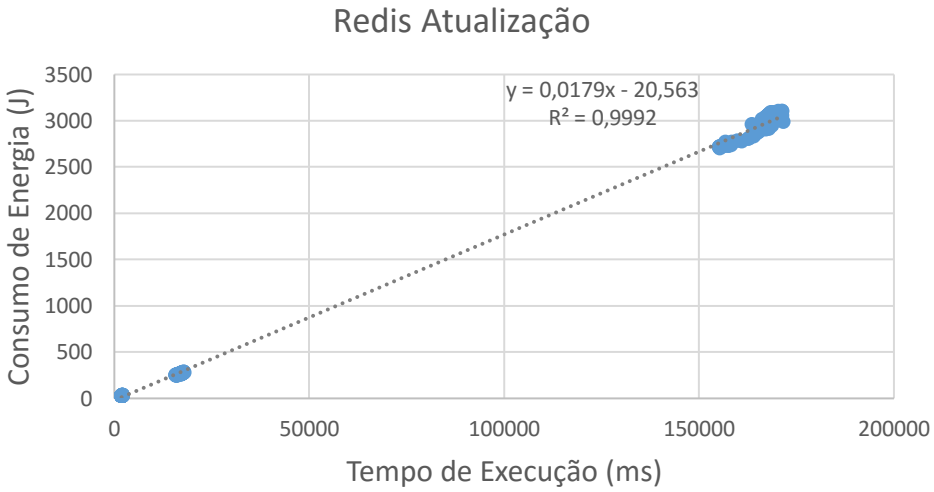

Figura 4.15: Correlação Tempo de Execução x Consumo de Energia Atualização Redis

4.3 Discussão

Em todos os experimentos foram utilizados para os SGBDs sua configuração padrão de instalação, desta forma qualquer otimização de configuração destes SGBDs estão fora do escopo deste trabalho, assim como a mudança do ambiente de experimentação. Esses fatores podem gerar resultados diferentes dos obtidos em nosso estudo. Além disso, adotou-se apenas um único computador para reduzir possíveis interferências de outras questões associadas a um ambiente de *cluster* sobre os resultados. Antes de cada execução do *benchmark* foi realizado o procedimento de *load* para aquele *benchmark*, de forma que cada execução foi realizada com base em um novo conjunto de dados gerados com base no algoritmo *zipfian* do YCSB.

Redis é baseado *in-memory* REDISLABS (2016a), mas com persistência em disco, essa característica influenciou na carga de trabalho de 1.000 operações para o melhor rendimento nas métricas tempo de execução e consumo de energia entre os SGBD comparados. O menor desempenho da escrita dentre os comandos executados por este SGBD pode estar relacionado ao *snapshot*, que é um procedimento de persistência em disco (acesso mais lento que a memória RAM) do Redis, ainda assim a escrita para esta carga de trabalho não obteve desempenho inferior à escrita dos outros SGBDs. Com o aumento da carga de trabalho o *snapshot* foi mais utilizado.

Executando 10.000 operações, há indícios de que este comando influenciou no desempenho do Redis, uma vez que a escrita obteve a maior degradação de desempenho no consumo de energia. Em comparação com os experimentos com carga de trabalho de 1.000 operações, leitura e atualização obtiveram uma relação de desempenho (operações x consumo) mais próxima à carga de trabalho de 1.000 operações. Apesar de o Redis não degradar seu desempenho, de maneira geral, em função da carga de trabalho anterior, a sua estabilidade foi um ponto negativo uma vez que os outros SGBDs melhoraram sua eficiência, com exceção da leitura do Cassandra, que mesmo melhorando sua eficiência não obteve melhor rendimento.

Em 100.000 operações uma relação similar de desempenho a carga de trabalho de 10.000 operações foi demonstrada nos três comandos (leitura, escrita e atualização), o padrão anterior se mantêm, Redis segue mantendo uma estabilidade no desempenho enquanto os outros SGBDs continuam melhorando sua eficiência. Essa característica levou o Redis a ser o SGBD com menor eficiência geral nas cargas de trabalho de 10.000 e 100.000 operações.

O SGBD Cassandra, ao contrário do Redis, melhorou sua eficiência notavelmente com o aumento da carga de trabalho. Nos experimentos Cassandra apresentou para leitura menor rendimento em detrimento da escrita e atualização. A explicação mais plausível para este acontecimento refere-se à descompactação dos dados lidos por Cassandra; para cada execução do experimento foi realizada uma nova carga de dados por YCSB, diminuindo a possibilidade da existência de dados disponíveis na memória RAM, os dados armazenados no disco rígido estão compactados, o acesso a estas informações implicam na descompactação, representando um incremento no custo deste acesso (CASSANDRA, 2015).

Operações em *background* realizadas pelo Cassandra influenciam no desempenho do SGBD, percebeu-se ao comparar o desvio-padrão dos comandos do Cassandra com os outros SGBDs avaliados. Mas observou-se também que o dispêndio deste conjunto de procedimentos começa a ser justificado ainda nestes cenários de único nó, apenas com o incremento da carga de trabalho. Por exemplo, na carga de trabalho de 100.000 operações o tempo de execução da leitura do Cassandra já não pode ser mais considerado diferente da leitura do Redis e possui desempenho próximo ao MongoDB.

Para o consumo de energia, a leitura Cassandra significativamente ainda consome mais energia que os demais SGBDs mesmo obtendo melhor eficiência em analogia às menores cargas de trabalho, percebeu-se ainda que a eficiência energética do Cassandra nesta carga de 100.000 operações não melhorou tanto quanto o tempo de execução comparando com as menores cargas de trabalho.

O MongoDB se mostrou o mais estável com relação a modificação das diferentes cargas de trabalho 1.000, 10.000 e 100.000 operações. Mesmo com este comportamento estável o MongoDB também melhorou de forma sutil sua eficiência com o aumento da carga de trabalho. O menor desempenho do MongoDB para escrita dentre seus próprios comandos pode estar relacionado ao recurso de *journaling* para prover a durabilidade dos dados em caso de falha do sistema. De maneira geral, MongoDB obteve o menor consumo de energia nas maiores cargas de trabalho.

É importante ressaltar que nenhum SGBD proporcionou um comportamento dominante (tempo de execução e consumo de energia) em todas as cargas de trabalho. De maneira geral, o SGBD mais rápido não foi o que obteve o menor consumo de energia e os resultados variaram em função dos SGBDs, comandos e carga de trabalho. A análise de variância demonstra que a iteração (SGBD * Comando) é a maior fonte de variação do consumo de energia, ou seja, a escolha de um SGBD NoSQL tem importante relação na eficiência do consumo em função do procedimento realizado.

A Tabela 4.12 apresenta a ordem de desempenho geral dos SGBDs nos diferentes tratamentos analisados.

Tabela 4.12: Ranking de Desempenho Geral

posição	tempo de execução	consumo de energia
1	Cassandra	MongoDB
2	MongoDB	Cassandra
3	Redis	Redis

5

Conclusão

Nas últimas décadas, as mudanças na sociedade se deram por influência do avanço tecnológico e à forma de comunicação mais rápida e acessível. O crescimento da comunicação presente em dispositivos móveis, *web sites*, redes sociais, sensores, *e-commerce*, internet das coisas (IoT), está modificando o cenário de gerenciamento dos dados em todo o mundo. Segundo CISCO (2016), o tráfego anual global vai ultrapassar um *zettabyte*(ZB) equivalente a 1.000 *exabytes* (EB), até o final de 2016, com projeção de, a partir de 2020 crescer 2,1 *zettabyte* ano. Este cenário fomenta o desenvolvimento de tecnologias NoSQL para o armazenamento de dados desta recente realidade. Entretanto, o aspecto do consumo de energia para o armazenamento destes dados ainda não possui o devido enfoque necessário.

Neste contexto, para este trabalho foi realizado um estudo da avaliação do consumo de energia em Sistemas de Gerenciamento de Banco de Dados NoSQL. Foram investigados 27 tratamentos, com comandos e cargas de trabalhos distintas com o objetivo de caracterizar o desempenho e o consumo de energia nestes cenários. Para a avaliação do desempenho e geração da carga de trabalho utilizou-se o *benchmark* Yahoo! Cloud Serving Benchmark e o consumo de energia foi aferido com o *framework* Emeter, que foi desenvolvido baseado no *hardware* EVM430-F6736. Esta avaliação foi realizada em exaustivas repetições a fim de ser possível realizar uma análise estatística mais profunda.

Considerando a diversidade de SGBDs NoSQL e que cada um foi desenvolvido para especialidades distintas, foram selecionadas as três categorias mais populares com um SGBD representante de cada classe (Colunas - Cassandra, Orientado a Documentos - MongoDB e Chave-valor - Redis). Os resultados apresentaram interessantes informações, há diferença significativa em diversos cenários de execução tanto para desempenho quanto para consumo de energia.

O desempenho médio das execuções entre os cenários relevantes entre SGBDs e a evolução para o incremento da carga de trabalho foi avaliado. Investigou-se também, por meio de análise estatística, mais precisamente a análise de variância (ANOVA), o impacto das fontes

de variação nas métricas, tempo de execução e consumo de energia dos cenários analisados.

Avaliou-se também a correlação entre o tempo de execução e o consumo de energia em cada comando por SGBD, considerando os distintos cenários de utilização verifica-se que há diferença significativa entre eles. A iteração (SGBD * Comando) é a maior fonte de variação no consumo de energia em todas as cargas de trabalho (1.000, 10.000 e 100.000 operações), este fato não se repetiu quando a métrica foi o tempo de execução onde na carga de trabalho de 100.000 operações o SGBD foi a maior fonte de variação. As diferenças de consumo de energia encontradas justificam a continuidade de estudos em ambiente de *cluster* e outros espaços não avaliados neste trabalho.

5.1 Contribuições

Apresentam-se as principais contribuições deste trabalho na listagem a seguir:

- A apresentação do *framework* Emeter baseado no *hardware* EVM430-F6736 para aferição do consumo de energia elétrica. Este *framework* foi validado e aperfeiçoado neste trabalho, podendo inclusive ser utilizado em trabalhos futuros;
- A apresentação de um procedimento para automatização de um *benchmark* de SGBDs NoSQL, esta apresentação foi realizada de maneira que possa ser implementada em diversas linguagens de *script*;
- Uma metodologia de avaliação de consumo de energia e desempenho. Esta metodologia utilizou o *Projeto de Experimentos* para organizar as comparações, utilizou *scripts* para a automação dos processos, utilizou um *benchmark* para gerar a carga de trabalho, validou as informações geradas e analisou estatisticamente os resultados;
- Os resultados individuais do tempo de execução e consumo de energia de vinte e sete cenários de *benchmark* de SGBDs NoSQL dispostos no Apêndice A deste trabalho;
- Uma discussão a respeito de vinte e sete cenários de utilização da tecnologia de Banco de Dados NoSQL, analisando as comparações pertinentes e influência dos principais fatores nas métricas tempo de execução e consumo de energia, e;
- Publicação de artigo no WPerformance 2016, selecionado na sessão técnica *Best Papers*:
  ARAÚJO, C. G.; TAVARES, E. A. G.; JUNIOR, M. Energy Consumption Evaluation of NoSQL DBMSs. In: CSBC WPERFORMANCE 2016 (), Porto Alegre. Anais. . . [S.l.: s.n.], 2016..

5.2 Limitações

Algumas limitações são encontradas neste trabalho. A carga de trabalho equivalente em *petabyte* ou superior, utilizada em *Big Data*, não foi considerada devido a restrições de *hardware* e tempo, necessários para a replicação dos experimentos em número suficiente para a obtenção de resultados que representem uma distribuição normal. Outra restrição refere-se à obtenção do consumo de energia por componente do *hardware* o equipamento de metrologia utilizado em nosso estudo fornece o consumo de energia total do servidor.

5.3 Trabalhos Futuros

Este trabalho apresenta resultados que motivam a realização de estudos que ofereçam continuidade a esta pesquisa para contribuição no cenário de eficiência no consumo de energia. Os seguintes itens são considerados para a avaliação em trabalhos futuros.

- Replicar estes experimentos em ambiente de *cluster*, de modo a comparar os resultados em *cluster* com os resultados em ambiente de um único nó de banco de dados;
- Avaliar a influência dos componentes de *hardware* (processador, memória, disco rígido) no tempo de execução e o consumo de energia de SGBDs NoSQL;
- Replicar estes experimentos em configurações otimizadas para desempenho e segurança dos SGBDs NoSQL;
- Tendo como pré-requisito os trabalhos anteriores, criar modelos de simulação do tempo de execução e consumo de energia com base no ambiente de execução do SGBD NoSQL.

Referências

ABADI, D. Consistency Tradeoffs in Modern Distributed Database System Design: cap is only part of the story. **Computer**, [S.l.], v.45, n.2, p.37–42, Fev 2012.

ABRAMOVA, V.; BERNARDINO, J. NoSQL Databases MongoDB vs Cassandra. In: INTERNATIONAL C* CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING, New York, NY, USA. **Proceedings...** ACM, 2013. p.14 – 22. (C3S2E '13).

ABUBAKAR, Y.; ADEYI, T.; AUTA, I. G. Performance evaluation of nosql systems using ycsb in a resource austere environment. **Performance Evaluation**, [S.l.], v.7, n.8, 2014.

ACADEMY, D. **The write path to compaction**. Acesso em: 30 jun. 2016, http://docs.datastax.com/en/cassandra/2.0/cassandra/dml/dml_write_path_c.html.

AMAZON WEB SERVICES. **Computação em nuvem com a Amazon Web Services**. Acesso em: 1 jan. 2016, https://aws.amazon.com/pt/what-is-aws/.

AMAZON WEB SERVICES. **Amazon DynamoDB**. Acesso em: 4 abr. 2016.

Analytic Methods for Design Practice. London: Springer London, 2007. p.309–391.

ARAUJO, C. J. M. **Avaliação e modelagem de desempenho para planejamento de capacidade de sistema de transferência eletrônica de fundos utilizando tráfego em rajada**. [S.l.]: Universidade Federal de Pernambuco, 2009.

BARBIERATO, E.; GRIBAUDO, M.; IACONO, M. Performance evaluation of NoSQL big-data applications using multi-formalism models. **Future Generation Computer Systems**, [S.l.], v.37, p.345 – 353, 2014.

BORAL, H.; DEWITT, D. J. **A methodology for database system performance evaluation**. [S.l.]: ACM, 1984. v.14, n.2.

BREWER, E. CAP twelve years later: how the "rules"have changed. **Computer**, [S.l.], v.45, n.2, p.23–29, Fev 2012.

BUKH, P. N. D.; JAIN, R. **The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling**. [S.l.]: JSTOR, 1992.

CAI, L. et al. Performance testing of HBase based on the potential cycle. In: COMPUTER AND INFORMATION SCIENCE (ICIS), 2013 IEEE/ACIS 12TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.359 – 363.

CASSANDRA, W. **Architecture Internals**. Acesso em: 29 jun. 2016, https://wiki.apache.org/cassandra/ArchitectureInternals.

CHANDRA, D. G. {BASE} analysis of NoSQL database. **Future Generation Computer Systems**, [S.l.], v.52, p.13 – 21, 2015. Special Section: Cloud Computing: Security, Privacy and Practice.

CHANG, F. et al. Bigtable: a distributed storage system for structured data. **ACM Trans. Comput. Syst.**, New York, NY, USA, v.26, n.2, p.4:1–4:26, June 2008.

CHODOROW, K. **MongoDB**: the definitive guide. 2.ed. [S.l.]: "O'Reilly Media, Inc.", 2013.

CIFERRI, R. R. Um benchmark voltado a análise de desempenho de sistemas de informações geográficas. **Campinas, SP, Brasil: UNICAMP**, [S.l.], 1995.

CISCO. **The Zettabyte Era**: trends and analysis. Acesso em: 9 jun. 2016, http://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/visualnetworkingindex-vni/vni-hyperconnectivitywp.pdf.

CLOUDERA. **Yahoo! Cloud Serving Benchmark (YCSB) Now Available in Cloudera Labs**. Acesso em: 30 jun. 2016, http://www.cloudera.com/about-cloudera/press-center/press-releases/2015-09-04-yahoo-cloud-serving-benchmark-now-available-cloudera-labs.html.

CODD, E. F. A Relational Model of Data for Large Shared Data Banks. **Commun. ACM**, New York, NY, USA, v.13, n.6, p.377 – 387, Junho 1970.

COLEMAN, D. E.; MONTGOMERY, D. C. A systematic approach to planning for a designed industrial experiment. **Technometrics**, [S.l.], v.35, n.1, p.1–12, 1993.

COOPER, B. F. **Issues brianfrankcooper/YCSB**. Acesso em: 30 jun. 2016, https://github.com/brianfrankcooper/YCSB/issues.

COOPER, B. F. et al. Benchmarking Cloud Serving Systems with YCSB. In: ACM SYMPOSIUM ON CLOUD COMPUTING, 1. **Proceedings...** [S.l.: s.n.], 2010. p.143–154. (SoCC '10).

DASCALESCU, L. et al. Using Design of Experiments and Virtual Instrumentation to Evaluate the Tribocharging of Pulverulent Materials in Compressed-Air Devices. **IEEE Transactions on Industry Applications**, [S.l.], v.44, n.1, p.3–8, Jan 2008.

DEBIAN. **Debian - O Sistema Operacional Universal**. Acesso em: 29 jun. 2016, https://www.debian.org/.

DECANDIA, G. et al. Dynamo: amazon's highly available key-value store. **SIGOPS Oper. Syst. Rev.**, New York, NY, USA, v.41, n.6, p.205–220, Oct. 2007.

DOUGLAS C. MONTGOMERY, G. C. R. **Applied Statistics and Probability for Engineers**. 6th.ed. [S.l.]: Wiley, 2013.

FERREIRA, M. C. C. d. S. **Modelos de Regressão**: uma aplicação em medicina dentária. 2013. Tese (Doutorado em Ciência da Computação) — .

FLICKR. **How many public photos are uploaded to Flickr every day, month, year?** 2015.

FLORATOU, A. et al. Can the elephants handle the NoSQL onslaught? , [S.l.], p.1712 – 1723, 2012.

FOUNDATION, P. S. **Python 2.7.0 Release**. Acesso em: 30 jun. 2016, https://www.python.org/download/releases/2.7/.

FREITAS, M. C. d. **Mapeamentos conceituais entre modelo relacional e estruturas NoSQL um estudo de caso com documentos**. [S.l.]: Universidade Federal de Pernambuco, 2015.

FRIGON, N. L.; MATHEWS, D. **Practical guide to experimental design**. [S.l.]: John Wiley & Sons, 1997.

GILBERT, S.; LYNCH, N. A. Perspectives on the CAP Theorem. **Computer**, [S.l.], v.45, n.2, p.30–36, Feb 2012.

GOOGLE. **Cloud Datastore**. Acesso em: 4 abr. 2016, https://cloud.google.com/datastore/.

GROLINGER, K. et al. Data management in cloud environments: nosql and newsql data stores. **Journal of Cloud Computing: advances, systems and applications**, [S.l.], v.2, n.1, p.22, 2013.

IBM. **What is big data? Bringing big data to the enterprise**. Acesso em: 29 mar. 2016.

IT solid. **DB-Engines Ranking**. Acesso em: 6 jul. 2016, http://db-engines.com/en/ranking.

JI, C. et al. Big data processing Big challenges and opportunities. **Journal of Interconnection Networks**, [S.l.], v.13, n.03n04, 2012.

KLEIN, J. et al. Performance Evaluation of NoSQL Databases: a case study. In: WORKSHOP ON PERFORMANCE ANALYSIS OF BIG DATA SYSTEMS, 1., New York, NY, USA. **Proceedings...** ACM, 2015. p.5–10. (PABS '15).

KUZNETSOV, S. D.; POSKONIN, A. V. NoSQL data management systems. **Programming and Computer Software**, [S.l.], v.40, n.6, p.323 – 332, 2014.

LAKSHMAN, A.; MALIK, P. Cassandra: a decentralized structured storage system. **SIGOPS Oper. Syst. Rev.**, New York, NY, USA, v.44, n.2, p.35–40, Apr. 2010.

LEAVITT, N. Will NoSQL Databases Live Up to Their Promise? **Computer**, [S.l.], v.43, n.2, p.12–14, Feb 2010.

LI, T. et al. Analyzing the Waiting Energy Consumption of NoSQL Databases. In: DEPENDABLE, AUTONOMIC AND SECURE COMPUTING (DASC), 2014 IEEE 12TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.277 – 282.

LILJA, D. J. **Measuring Computer Performance A practitioner's guide**. 1th.ed. [S.l.]: Cambridge Univerty Press, 2004.

MARTIN J. FOWLER, P. J. S. **Nosql Distilled**: a brief guide to the emerging world of polyglot persistence. [S.l.]: Pearson, 2014.

MASTELIC, T. et al. Cloud Computing Survey on Energy Efficiency. **ACM Comput. Surv.**, New York, NY, USA, v.47, n.2, p.33:1 – 33:36, Dezembro 2014.

MICROSOFT. **Banco de Dados de Documentos**. Acesso em: 4 abr. 2016, https://azure.microsoft.com/pt-br/services/documentdb/.

MINAS, L.; ELLISON, B. **Energy efficiency for information technology How to reduce power consumption in servers and data centers**. [S.l.]: Intel Press, 2009.

MONGODB. **Bring your giant ideas to life with MongoDB**. Acesso em: 30 jun. 2016, https://www.mongodb.com/what-is-mongodb.

MONIRUZZAMAN, A. B. M.; HOSSAIN, S. A. NoSQL Database: new era of databases for big data analytics - classification, characteristics and comparison. **CoRR**, [S.l.], v.abs/1307.0191, 2013.

WESLEY, P. A. (Ed.). **Sistemas de Banco de Dados**. 6.ed. [S.l.: s.n.], 2010.

NEVES, R.; BERNARDINO, J. Performance and scalability of voldemort NoSQL. In: INFORMATION SYSTEMS AND TECHNOLOGIES (CISTI), 2015 10TH IBERIAN CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.1 – 6.

NOSQL. **Your Ultimate Guide to the Non-Relational Universe**. Acesso em: 30 jun. 2016, http://nosql-database.org/.

NOSQL-MEETUP. **NoSQL meetup**. Acesso em: 30 jun. 2016, http://nosql.eventbrite.com/.

NRDC. **America's Data Centers Consuming and Wasting Growing Amounts of Energy**. Accessed: 2016-03-01, http://www.nrdc.org/energy/data-center-efficiency-assessment.asp.

PARTHASARATHY, V. **Learning Cassandra for Administrators**. [S.l.]: Packt Publishing Ltd, 2013.

PLANET, C. **What is Apache Cassandra?** Accessed: 2016-06-02, http://www.planetcassandra.org/what-is-apache-cassandra/.

PLATAFORM, G. C. **Cloud Bigtable**: a high performance nosql database service for large analytical and operational workloads. Acesso em: 30 jun. 2016, https://cloud.google.com/bigtable/.

POSTGRESQL. **The world's most advanced open source database**. Acesso em: 29 jun. 2016, http://www.postgresql.org/.

PRITCHETT, D. BASE: an acid alternative. **Queue**, New York, NY, USA, v.6, n.3, p.48–55, May 2008.

REDISLABS. **Redis**. Acesso em: 30 jun. 2016, http://redis.io/.

REDISLABS. **Redis Transactions**. Acesso em: 30 jun. 2016, http://redis.io/topics/transactions.

REIS, L. B. dos. **Geração de energia elétrica**. 2.ed. [S.l.]: Editora Manole, 2015.

SCHREINER, G. A.; DUARTE, D.; SANTOS MELLO, R. dos. SQLtoKeyNoSQL: a layer for relational to key-based nosql database mapping. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 17., New York, NY, USA. **Proceedings...** ACM, 2015. p.74:1–74:9. (iiWAS '15).

SERIATOS, G. et al. **Comparison of database and workload types performance in Cloud environments**. [S.l.: s.n.], 2016.

SOUZA, V. C. O. de; SANTOS, M. V. C. dos. Maturing, Consolidation and Performance of NoSQL Databases-Comparative Study. In: BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS: INFORMATION SYSTEMS: A COMPUTER SOCIO-TECHNICAL PERSPECTIVE-VOLUME 1. **Proceedings...** [S.l.: s.n.], 2015. p.32.

STROZZI, C. **NoSQL**: a non-sql rdbms. Acesso em: 30 jun. 2016, http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page.

SUBRAMANIAM, B.; FENG, W.-c. On the Energy Proportionality of Distributed NoSQL Data Stores. In: INTERNATIONAL WORKSHOP ON PERFORMANCE MODELING, BENCHMARKING AND SIMULATION OF HIGH PERFORMANCE COMPUTER SYSTEMS. **Anais...** [S.l.: s.n.], 2014. p.264–274.

TEXAS-INSTRUMENTS. **EVM430-F6736 - MSP430F6736 EVM for Metering**. Acesso em: 29 jun. 2016, http://www.ti.com/tool/EVM430-F6736.

TEXAS-INSTRUMENTS. **Implementation of a Single-Phase Electronic Watt-Hour Meter Using the MSP430F6736(A)**. Acesso em: 29 jun. 2016, http://www.ti.com/lit/zip/slaa517.

TPC. **TPC-H is an ad-hoc, decision support benchmark**. Acesso em: 29 jun. 2016, http://www.tpc.org/tpch/.

TRIOLA, M. F. **Introdução à estatística**. 9th.ed. [S.l.]: LTC, 2005.

TSIROGIANNIS, D.; HARIZOPOULOS, S.; SHAH, M. A. Analyzing the Energy Efficiency of a Database Server. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2010., New York, NY, USA. **Proceedings...** ACM, 2010. p.231 – 242. (SIGMOD '10).

WIKI-YCSB. **Core Properties**. Acesso em: 29 jun. 2016, https://github.com/brianfrankcooper/YCSB/wiki/Core-Properties.

Y. DEMCHENKO C. DE LAAT, P. M. Defining architecture components of the Big Data Ecosystem. In: COLLABORATION TECHNOLOGIES AND SYSTEMS (CTS), 2014 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.104 – 112.

Apêndice

A

Apresentação dos Dados

Apresentamos 75 amostras resultados da execução de cada tratamento apresentado neste trabalho.

Tabela A.1: Dados: Correlação Cassandra 1.000 Operações

Cassandra					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
3562	65,38986	3795	67,60375	3075	50,16025
3569	63,28054	3819	68,86975	3078	49,56304
3586	62,90606	3862	76,23342	3080	55,6776
3586	58,20376	3881	69,51168	3096	51,98528
3597	64,36603	3895	85,03835	3120	52,3004
3599	58,65688	3895	80,3073	3121	52,486
3602	60,40986	4046	85,22598	3125	50,8925
3607	63,66706	4075	90,57975	3153	52,17598
3608	58,54856	4139	75,37392	3205	55,96155
3622	58,80856	4280	78,7488	3213	59,51948
3648	64,12768	4290	82,2772	3214	52,86848

3684	66,90188	4290	74,2336	3221	54,01351
3765	61,88285	4315	76,29015	3227	53,08482
3768	68,12976	4442	96,02818	3247	50,95946
3785	70,33135	4720	102,9668	3250	56,78
3818	74,11624	4739	83,45843	3256	57,47688
3823	78,62402	5113	115,46873	3257	51,07439
3872	69,14344	5144	121,08448	3266	52,0929
3897	80,74423	5331	114,71099	3286	58,96874
3897	61,20161	5431	119,84808	3290	58,8915
3926	67,18154	5592	112,64848	3298	59,05184
3940	68,0122	5622	108,99526	3316	54,98428
3961	83,28357	5630	118,1762	3324	51,28748
3997	61,71449	5780	114,3464	3324	54,67368
4003	67,05605	5855	123,7315	3327	55,36713
4014	62,70182	6106	140,77798	3332	53,27628
4027	72,49067	6134	131,1039	3363	63,35266
4044	84,1664	6148	140,04992	3387	60,45593
4049	71,20705	6554	146,33052	3395	58,46385
4050	70,2435	6640	152,8816	3413	54,68846
4051	66,56622	6754	148,4301	3427	56,1812
4059	67,07158	6801	129,39933	3438	56,81454
4061	63,00049	6819	133,32889	3439	63,23107
4069	69,22156	6868	154,84232	3450	56,204
4073	67,77179	7060	161,951	3451	58,25639
4077	68,54118	7431	154,23396	3452	56,00944
4078	74,89112	7915	167,5112	3495	56,37755

4082	64,48788	7944	160,696	3495	56,5921
4086	69,34504	7957	176,60436	3510	56,4893
4091	63,33138	8029	183,60254	3510	58,4775
4093	70,7536	8144	165,896	3519	67,32792
4100	69,586	8448	195,31664	3521	55,44609
4106	70,9859	8453	193,21116	3535	61,57735
4111	69,56272	8579	197,69505	3539	54,75514
4116	68,76176	8845	196,65385	3544	56,62496
4126	68,73906	8974	211,99542	3562	56,1686
4131	78,06858	9236	216,81048	3565	57,08185
4138	78,03832	9463	192,50872	3566	60,72186
4138	70,13828	9954	212,00504	3574	59,60198
4159	68,93316	10018	216,67342	3589	58,22526
4168	77,5036	10157	230,20019	3590	57,7306
4179	74,35543	10185	215,662	3601	60,43328
4183	73,72807	10827	251,37504	3608	57,672
4196	69,239	11751	247,15319	3633	64,62339
4209	81,55982	11957	258,17221	3641	59,61294
4215	74,6552	11959	266,03639	3694	58,28596
4231	65,75503	12055	278,74185	3696	58,4284
4309	70,24753	12297	270,63192	3714	65,42128
4315	69,6032	12349	275,71185	3718	58,45976
4340	73,1714	12356	271,8724	3745	62,3934
4345	70,1271	12633	285,93187	3756	63,79876
4381	91,77597	12712	284,95088	3770	63,3712
4382	74,55302	12809	294,05816	3772	67,23528

4390	78,0093	12830	295,0618	3786	59,3458
4410	74,6779	13037	296,43238	3787	60,90962
4412	74,80132	13248	299,58424	3805	59,66235
4421	86,59236	13291	274,16103	3820	61,4328
4438	79,13602	13867	306,31978	3856	65,29304
4445	70,643	14890	337,8162	3869	66,33684
4507	71,95147	14939	332,94389	3881	61,88335
4541	93,13288	15243	342,42884	3886	63,32906
4547	76,92645	15417	329,40253	3899	62,9046
4615	76,1762	16009	349,85139	3928	62,64984
4737	78,63977	16679	378,49697	3949	67,14103
4798	85,1576	16871	384,14985	3999	61,93478

Tabela A.2: Dados: Correlação Cassandra 10.000 Operações

Cassandra					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
12966	252,93824	14611	484,06054	8467	157,01129
12983	253,73817	14695	436,244	8483	147,22711
12984	268,09784	14785	473,8473	8490	148,0274
12985	247,3414	14807	397,42254	8494	148,75038
12990	255,3646	14829	457,85374	8515	155,23905
13012	255,56828	14914	407,45958	8534	149,37486
13051	247,43274	15047	486,63944	8536	149,34648
13052	264,4464	15119	497,20118	8565	152,49135
13056	272,27032	15250	475,1625	8573	158,86734

13090	248,914	15369	499,08592	8579	152,86082
13113	249,53783	15456	470,05688	8583	153,65808
13121	261,43397	15578	493,07496	8591	157,48921
13122	257,08296	15643	447,4694	8602	156,71876
13125	299,135	15735	504,5767	8670	157,063
13135	331,5146	16063	526,62382	8673	205,4953
13146	263,76614	16410	514,1161	8697	148,35016
13180	260,7096	16693	544,94202	8698	151,71806
13208	257,5348	17869	562,43177	8708	156,38772
13213	266,74836	17883	534,87809	8731	155,22255
13214	262,7484	18034	587,71972	8741	159,21874
13278	259,88792	18475	592,539	8754	155,47454
13353	258,9956	18938	581,3945	8786	160,6839
13355	319,3864	20891	553,21794	8829	154,61251
13356	264,88956	21105	617,2664	8854	155,9608
13405	262,7995	22002	670,12072	8854	204,15332
13411	258,97037	22155	704,09375	8860	160,8314
13477	266,86074	22393	642,2636	8883	156,99246
13545	287,00035	22559	701,23475	8940	154,2714
13560	263,8344	22627	682,87692	8985	160,66345
13564	256,58024	23145	691,6711	9095	155,5616
13564	256,601	23180	672,8388	9137	163,68446
13611	253,77553	23188	658,24904	9146	162,9457
13625	261,16125	23519	626,7383	9390	167,318
13628	275,71584	23739	639,38626	9396	164,13196
13725	277,49025	24064	724,74288	9430	159,1046

13745	259,6624	24069	674,1688	9458	161,79152
13756	280,11704	24513	741,77455	9489	165,40351
13786	258,66152	24958	679,34412	9543	165,11102
13789	264,82436	25028	793,26532	9544	169,02552
13835	323,5787	26277	775,84101	9550	167,7885
14041	267,54197	27553	758,41136	9551	164,12969
14070	281,005	27758	774,6058	9571	161,24636
14113	358,51715	28042	798,55756	9571	169,34917
14217	273,74057	28138	794,00242	9614	169,1049
14248	271,66976	28223	859,71657	9616	164,66632
14318	279,18268	29118	877,20652	9679	166,23117
14423	283,19652	29729	833,05538	9705	169,38305
14489	271,37038	30453	879,68851	9717	169,72803
14579	270,46975	30527	904,27952	9724	180,84336
14623	279,69321	32616	850,08784	9759	167,78378
14675	286,291	32830	926,3624	9761	176,20487
14688	289,38832	38116	1068,3012	9774	168,74604
14695	276,9542	38954	1106,54988	9787	183,56377
14714	294,45708	39600	1087,484	9846	172,58138
14748	280,3144	39620	1054,754	9892	174,73192
14779	293,1203	40174	1076,14176	9924	175,69364
14783	293,97234	41126	1080,9889	9949	168,36817
14797	383,30846	41612	1114,69616	9967	171,86345
14801	358,46545	41702	1122,32014	9973	175,8269
14880	291,424	42260	1140,0032	9974	173,85506
14881	294,6217	45817	1238,34206	9995	175,9938

14885	353,88325	48737	1331,33258	10149	174,15821
14889	283,75835	49854	1347,33612	10158	178,87582
14901	284,13451	50846	1332,26304	10163	180,3539
14965	287,24665	53880	1483,4264	10267	175,64374
14975	282,641	54090	1414,6743	10292	178,74132
14982	365,62334	54408	1429,46336	10321	177,24377
15010	374,6843	55792	1494,36088	10332	179,04276
15015	352,8698	58668	1540,11692	10408	181,0764
15265	286,09885	59134	1568,08278	10455	175,536
15315	292,0962	60584	1555,4168	10483	177,3999
15428	283,88128	65326	1688,09802	10528	181,46728
15510	285,6981	66099	1744,05945	10530	180,5408
15684	309,96616	68256	1785,51376	10713	189,46289
16254	320,63366	68528	1778,64376	10733	188,23286

Tabela A.3: Dados: Correlação Cassandra 100.000 Operações

Cassandra					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
98032	2413,58464	102398	3291,11452	54786	1197,26316
98159	2492,62266	103555	2713,7401	55094	1145,4605
98373	2417,64102	105197	3360,30866	55120	1122,756
98691	2606,39629	105820	3162,5586	55143	1189,32942
99103	2753,51358	106439	3381,04353	55230	1106,1124
99191	2516,53318	107030	3270,3224	55385	1091,6676
99365	3325,5208	107610	3351,9286	55407	1092,44211

99422	2540,18604	107665	2725,238	55519	1102,25452
99487	2458,00084	109178	3373,76188	55846	1142,10976
99589	2393,39151	111496	3639,412	55924	1124,633
99626	2390,54796	111871	3652,32309	56036	1229,86772
99629	2368,08741	112484	4010,4884	56068	1100,71088
99651	2433,23674	112549	3398,27793	56162	1125,04474
99754	2521,16892	113034	3729,7802	56191	1200,11825
99804	2517,17184	113138	3773,88484	56364	1273,53388
100044	2483,55344	113211	3502,50509	56539	1176,50386
100046	2493,22772	113891	3665,02325	56644	1096,57876
100435	2498,07735	116078	3505,38566	56773	1519,46414
100781	2484,22426	117007	3213,50927	57014	1121,92204
101058	2587,97176	118057	4191,89086	57123	1319,71725
101064	2498,91152	119479	3841,6008	57136	1195,39928
101322	2685,51032	119930	3664,1455	57230	1228,7909
101377	2643,39021	120792	3878,46552	57575	1150,97775
101388	2449,91804	121443	3885,43917	57675	1281,69775
101474	2552,11348	122436	4164,41804	57793	1139,88491
101496	2436,99584	124094	3845,76486	58907	1281,60253
101529	2614,29704	124620	3978,5744	59230	1294,7683
101893	2519,75047	128631	4145,98471	59257	1272,73747
101930	2405,884	130457	4174,27544	59283	1332,92951
101962	2487,25238	130497	4224,78115	60247	1155,32181
101963	2448,4543	130737	4350,93937	60652	1859,93204
102198	2483,55246	130917	4249,38678	60926	1176,13228
102311	2537,57667	131537	4348,88868	61220	1357,2162

102340	2686,6482	133130	4388,3076	61673	1402,25952
102481	2601,05423	133453	4436,54475	61711	1216,58119
102509	2562,72202	134513	4513,47864	61734	1183,89882
102690	3056,5568	135686	4366,22904	61758	1279,36046
103083	2447,08496	136894	3648,11138	61953	1182,80902
103786	2747,27008	136895	4299,95045	62281	1921,97495
104131	2491,46168	137707	4505,15595	62387	1202,35011
104532	2594,1924	138791	4694,57272	62573	1317,11533
105056	2523,06624	140418	4540,58514	62605	1217,6997
105069	2899,33952	141601	4607,67176	62729	1353,32912
105284	2559,43952	141976	4665,36008	63199	1300,84664
105332	2459,14296	144753	4553,49572	63482	1243,36064
105371	2530,5292	144981	4444,44075	63579	1215,29135
106646	2795,53448	146037	4658,54351	63757	1217,39911
106773	2486,69009	148577	4771,42771	63793	1304,27669
108009	2603,61716	149883	4680,82747	63911	1560,22532
108314	2570,5999	151210	4497,6949	64201	1244,97766
109477	2723,15735	151597	4769,50589	64232	1443,64192
109985	2652,05945	152045	4798,84995	64260	1254,9824
111071	2717,90381	157134	5085,51132	64326	1373,77498
111546	2690,1922	160497	5151,90781	64891	1355,4932
112458	2667,30076	162722	5121,59092	65045	1349,53685
112984	2642,42536	170086	5154,11032	65199	1253,36793
113928	2711,16032	170627	5183,22509	65217	1250,3386
114344	2787,59224	194734	6545,21072	65338	1279,83942
114620	2622,022	220254	6912,64286	65731	1350,57389

114701	2644,10344	221030	7039,5522	65874	1257,79558
114767	2667,60219	230157	7228,77111	65885	1320,0505
114776	2677,11608	231703	7390,17156	65893	1333,55755
115115	2952,85685	288048	7352,25384	65925	1286,62
115123	2977,9896	292893	8641,32788	65982	1285,76488
116137	2741,43624	298215	8632,81165	66057	1344,24469
116280	2799,8624	301707	9097,84948	66123	1359,22356
116305	2970,52685	328417	9735,79015	66164	1284,38232
116327	2633,89348	363846	9502,71034	66203	1319,40996
116513	3384,6676	364109	10367,73215	66440	1376,4412
116974	2787,74466	369782	10625,6524	66580	1246,8988
118497	2953,74392	485516	12923,04896	66693	1262,73405
119008	2775,8932	486169	13147,19446	66716	1363,08652
119608	3039,94944	525327	13380,49603	66737	1282,69564
119961	2768,5372	554520	14254,5096	66758	1298,09772
121496	3551,324	678384	17371,02296	66900	1385,793

Tabela A.4: Dados: Correlação MongoDB 1.000 Operações

MongoDB					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
3966	62,95878	3272	52,17056	3448	56,40592
4006	64,98228	3287	51,57684	3469	56,31791
4030	65,1384	3375	53,75625	3490	55,7825
4103	67,23062	3383	53,52841	3557	59,07086
4211	68,82481	3411	54,36118	3617	60,32667

4253	70,18343	3470	53,8845	3684	61,29548
4260	68,1722	3482	55,98388	3688	61,82256
4297	70,0014	3525	56,151	3692	62,26268
4317	71,15788	3530	56,4297	3700	60,244
4348	70,54964	3555	56,5993	3703	61,70315
4365	70,604	3557	55,94565	3715	59,54805
4378	72,61474	3569	56,8464	3729	61,35858
4397	72,06713	3575	55,7345	3753	61,22848
4402	71,03326	3576	57,93072	3779	63,10002
4420	71,1608	3582	57,03666	3784	61,77512
4443	72,44295	3582	55,7937	3803	61,57543
4446	72,26558	3587	57,48047	3811	62,05373
4447	74,17352	3605	57,0575	3812	62,10636
4450	72,254	3613	57,5889	3812	62,46628
4453	72,14646	3615	56,6671	3833	61,87983
4489	72,48351	3618	56,05832	3834	62,71918
4493	74,77363	3643	56,98861	3863	62,76321
4507	72,96724	3647	57,73615	3868	62,97424
4532	73,62604	3653	57,19702	3869	63,34697
4566	74,31186	3672	58,33992	3873	64,50435
4567	74,33489	3676	58,28448	3874	64,25132
4574	74,21772	3684	60,6298	3898	63,34512
4576	74,574	3688	58,0984	3904	63,22592
4599	74,77648	3702	58,97868	3908	65,7028
4599	75,13237	3708	58,97488	3918	65,43504
4600	74,362	3711	59,45027	3922	63,87796

4616	75,61832	3712	60,99624	3930	63,6958
4624	76,55928	3716	58,78652	3934	66,00694
4630	78,1987	3738	60,3873	3940	66,1482
4637	75,60118	3744	57,79528	3946	65,43114
4637	75,56659	3746	59,09808	3960	66,4984
4651	75,47647	3764	60,62144	3961	67,45416
4653	78,04532	3764	58,83144	3967	63,97761
4666	76,2426	3767	59,29986	3972	63,85928
4680	76,3876	3772	58,093	3975	66,8155
4685	75,67995	3774	59,98994	4002	65,26182
4695	77,83115	3783	60,07858	4004	67,16564
4698	76,78034	3784	61,12848	4005	65,08765
4699	75,65373	3789	60,99299	4008	65,79432
4712	76,67872	3805	59,227	4019	64,76089
4723	76,88866	3806	60,1727	4020	65,3078
4723	77,3557	3813	60,47963	4021	65,08432
4723	77,45509	3816	60,70432	4048	68,1004
4736	76,8628	3825	60,6315	4048	67,60624
4767	77,69734	3829	60,33898	4052	65,48976
4781	77,51798	3829	58,64909	4056	66,05704
4805	80,70485	3832	60,99416	4082	66,38706
4829	77,53567	3834	58,51174	4093	65,51453
4846	77,97296	3836	60,88472	4096	68,3216
4890	80,0699	3840	60,5728	4098	67,74662
4901	82,40561	3890	61,3003	4099	65,92777
4929	79,43015	3895	62,58725	4104	68,46544

4935	79,68185	3903	62,39844	4109	66,29675
4938	80,61312	3906	62,08112	4110	66,2569
4966	79,9896	3910	61,6006	4116	66,83132
4974	81,0157	3936	60,47272	4127	69,10343
4976	81,26032	3938	62,81272	4133	68,43081
4988	80,63872	3954	62,82562	4141	68,41023
4995	80,44235	3963	62,77724	4162	69,60988
5011	83,71435	3963	63,47503	4176	67,04984
5019	81,96051	3972	64,97368	4177	69,51076
5027	81,86903	3977	62,96419	4181	68,12342
5032	81,41992	3982	62,15118	4207	67,95471
5044	82,28728	4016	63,7796	4216	67,46184
5083	82,75326	4016	63,75352	4223	68,11744
5088	82,35424	4029	63,91138	4254	71,1956
5099	84,65539	4038	64,11026	4256	68,51912
5121	83,56041	4065	67,0575	4265	69,7318
5135	86,9168	4065	65,37765	4277	69,67599
5135	82,7487	4066	66,17214	4288	71,01632

Tabela A.5: Dados: Correlação MongoDB 10.000 Operações

MongoDB					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
15721	265,58246	10820	181,0972	12309	207,32697
15750	273,4975	11029	183,53327	12352	211,45032
15803	274,33787	11293	182,6229	12528	212,45424

16394	283,53308	11349	187,41534	12654	214,8205
16717	288,53425	11410	183,7689	13094	227,10084
16733	293,43825	11690	187,9456	13450	224,653
17790	294,2893	12754	206,48914	13520	226,78
18165	302,92615	12789	209,75996	13769	228,18792
18604	312,53756	13074	209,67888	14032	232,8896
18899	314,77076	13091	209,0623	14062	244,9616
19163	318,36575	13095	208,6554	14208	233,72536
19352	320,5064	13132	216,926	14251	242,65811
19387	334,50073	13134	219,61132	14280	244,418
19519	324,81172	13158	212,71846	14293	235,62876
19639	329,38558	13190	218,027	14295	246,3628
19817	331,12474	13249	217,9697	14297	243,6717
19915	329,4114	13270	217,2275	14384	248,38672
19988	331,27124	13366	213,65418	14414	245,2361
20120	334,6132	13390	214,3255	14416	244,78016
20195	336,1154	13422	217,1244	14418	241,91212
20232	334,75248	13440	215,7432	14436	248,7908
20256	339,32352	13590	218,137	14530	240,4372
20396	338,79672	13638	219,87072	14612	246,763
20408	351,00976	13646	219,30088	14663	242,33071
20455	342,29145	13661	225,2316	14705	246,01
20487	342,09207	13688	224,44648	14738	250,1383
20495	341,9706	13783	220,58943	14771	258,24005
20584	344,71928	13839	222,94509	14798	248,05952
20622	339,60306	13895	224,5478	14828	243,31904

20626	356,31592	13908	227,40504	14831	244,56416
20657	342,66611	13931	222,23982	14845	248,2537
20719	358,49772	13934	224,02284	14919	247,31964
20873	346,38087	13953	235,28847	14920	250,308
20926	346,9715	14041	227,74976	14923	253,198
20944	349,31888	14041	224,7191	14981	246,19177
20948	361,50544	14061	224,05049	15005	257,4081
20959	357,41245	14108	226,69484	15043	252,45102
20993	348,65947	14117	232,37105	15090	249,9625
20995	356,66925	14147	234,06734	15119	249,93404
21136	363,82128	14156	229,64588	15123	249,20714
21137	361,92871	14235	228,4908	15130	256,8519
21214	363,61112	14250	232,275	15134	257,5272
21243	359,19276	14274	232,67398	15154	261,22004
21368	352,772	14291	228,9432	15156	250,40068
21396	353,6992	14297	229,11925	15172	252,41816
21463	362,51912	14305	237,2882	15201	249,49585
21538	368,89712	14317	233,92157	15254	256,4151
21596	362,86952	14339	234,42653	15295	262,52625
21625	338,7475	14358	234,90394	15356	255,03628
21652	367,51676	14378	229,15024	15356	255,53272
21788	367,17316	14381	235,13358	15361	253,17164
21791	374,63529	14392	236,70224	15363	255,47945
21804	374,97224	14417	229,59589	15368	266,53024
21835	359,5519	14429	233,57799	15416	262,41208
21863	360,76841	14441	230,6214	15510	256,3833

21921	362,94654	14466	231,30912	15524	259,62292
21932	375,14376	14513	238,78351	15552	255,3728
21956	364,31448	14550	241,486	15619	257,68118
21998	375,50872	14561	231,57916	15620	262,6764
22029	367,09877	14563	234,29701	15640	264,7312
22133	366,5616	14567	232,75108	15702	262,13742
22135	365,7666	14583	238,19658	15753	261,95071
22239	383,4867	14597	239,72141	15815	260,6774
22288	372,12368	14705	242,71145	15843	269,96889
22334	384,07708	14708	236,26408	15844	270,11684
22349	381,32789	14784	236,28952	15876	262,048
22377	375,06548	14787	237,18666	15917	272,51508
22455	370,22345	14860	245,0714	15926	261,87668
22471	370,96565	14906	236,31966	15941	261,75078
22788	379,79184	14908	245,2078	15957	273,12683
22806	385,64614	14919	236,59556	16290	271,4093
22849	376,54554	14955	237,9278	16363	279,84821
22908	381,47432	14970	239,1916	16368	279,38064
22910	381,0038	15006	246,22396	16459	278,28794
22936	393,17936	15020	245,1152	16473	271,42988

Tabela A.6: Dados: Correlação MongoDB 100.000 Operações

MongoDB					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
115945	2005,9367	67366	1078,96372	85617	1528,24244

117673	2023,16074	68173	1102,55766	86148	1554,75104
118217	2033,74793	68247	1097,76663	86557	1530,44721
118584	2044,30104	70391	1129,87531	87144	1533,14216
119665	2083,78465	80504	1303,29584	87876	1544,34128
119920	2065,182	99601	1606,39323	88319	1552,05156
164377	2784,44417	100676	1616,03476	89598	1604,71986
164691	2790,16247	101010	1620,1845	105109	1878,43462
165489	2794,93769	101159	1630,09521	105916	1862,1986
166515	2824,9349	101730	1659,7588	106799	1877,4744
166825	2817,746	101985	1626,12155	106898	1904,63164
166921	2822,8776	102007	1625,48906	107024	1930,15056
167015	2826,82265	102155	1637,69475	107517	1919,45806
167588	2835,8982	102484	1631,22716	107632	1900,65552
169871	2881,18244	102710	1652,4833	107788	1884,42412
170075	2894,98	102944	1662,73712	108341	1921,25366
170342	2877,12446	103308	1661,75008	108586	1911,63428
171709	2895,42717	103653	1670,52519	108814	1906,28448
171856	2822,49976	103762	1674,1629	108825	1932,19025
172148	2910,21368	103983	1660,38514	108836	1945,22048
172825	2914,41475	104118	1654,40658	109155	1921,73715
173274	2941,06384	104253	1672,1356	109750	1942,95
173625	2945,895	104519	1676,15399	109879	1924,32507
174265	2955,5027	104522	1673,20276	109923	1950,28883
174397	2944,19219	104664	1682,90936	110334	1968,86732
174479	2950,44122	105025	1684,76075	110980	2024,3502
174615	2945,3672	105404	1695,35908	110989	1960,34972

175511	2990,5072	105417	1683,70184	111524	1990,83352
175525	2989,28375	105527	1698,94675	111745	2002,10145
175682	2963,90322	105534	1693,0868	111814	1989,80396
175931	2966,05808	105657	1698,70239	111819	1984,28685
176253	2972,89596	105919	1715,68503	111989	1960,56214
176361	2978,63442	105988	1689,5052	112150	1983,604
176399	2987,83233	106063	1709,00327	112162	1987,83698
176708	2975,79576	106233	1697,13927	112209	1967,57516
176900	2988,217	106297	1699,39192	112427	1974,19186
177026	3012,23638	106426	1695,9965	112602	1994,8113
177260	2991,1098	106741	1704,65219	112712	1968,03752
177410	2994,0111	106868	1716,68964	112732	2025,02104
177995	3007,33025	106949	1708,7748	113390	2024,7409
178489	3016,14378	107208	1721,68408	113644	2042,99776
178575	3012,56425	107912	1719,97744	114365	2023,5453
178811	3041,32287	108028	1758,06336	114692	2005,16756
178854	3044,98664	109067	1749,87649	115044	2043,2854
179161	3036,65414	109787	1780,6711	115427	2030,27186
179524	3053,3648	110084	1757,38696	115465	2031,3181
180280	3035,9464	110500	1774,035	115819	2052,79192
180697	3034,37078	111271	1800,5992	116110	2025,6637
182051	3086,42407	111273	1788,96871	116335	2068,82585
182218	3059,51004	111374	1780,95862	116662	2060,65734
183032	3076,19984	111390	1768,6399	118441	2091,39252
183066	3080,69528	111554	1769,67376	118493	2089,35181
183694	3087,35044	111678	1799,06882	118690	2073,2061

184967	3129,04015	111842	1798,03784	118755	2090,39715
185820	3128,646	111948	1786,74544	119017	2133,18605
186700	3156,451	111968	1796,9056	119082	2112,52166
186719	3160,3002	112107	1807,24101	119244	2130,06768
186958	3153,45076	112392	1786,39816	119646	2142,61538
187883	3169,74171	112480	1820,6052	119695	2109,42705
187928	3163,0008	112508	1801,677	119835	2090,95035
187999	3174,14434	112681	1787,09869	120537	2137,09775
188543	3171,97139	113127	1814,94342	120659	2124,26542
189586	3206,14618	113538	1816,7183	120687	2128,73171
189608	3189,81904	113602	1794,6103	120724	2131,56204
189775	3187,01425	113698	1810,51202	120877	2131,16387
190192	3208,94944	113824	1813,99312	121177	2128,44067
190260	3197,5864	114508	1818,1932	121271	2152,42191
190270	3201,4797	114543	1823,01677	121275	2128,52
190957	3225,84361	115101	1857,82328	121505	2106,68065
191128	3205,44664	115426	1836,22484	122257	2144,33831
191175	3218,36475	116124	1883,97084	122828	2141,13196
191443	3213,93588	116129	1854,42402	122871	2190,93891
191536	3216,80096	116790	1871,9576	123135	2193,50865
191691	3213,53702	116928	1870,99768	123167	2185,76876
191902	3217,43826	116985	1876,5399	123621	2184,43961

Tabela A.7: Dados: Correlação Redis 1.000 Operações

Redis		
Escrita	Leitura	Atualização

Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
3639	55,69724	1846	28,3815	1847	28,39451
3645	55,93915	1858	28,69036	1851	28,27818
3675	57,49625	1862	28,25344	1860	29,2862
3677	56,21947	1870	28,8993	1865	29,4051
3682	56,6305	1874	28,90086	1873	29,47245
3684	56,4768	1878	28,95998	1874	28,67606
3686	56,18802	1883	28,8816	1887	28,7324
3703	58,43571	1893	30,10047	1889	29,58726
3706	58,24536	1897	28,68264	1892	28,84624
3741	57,42284	1903	28,76433	1896	29,13776
3743	58,96365	1914	29,2771	1898	29,14042
3751	57,2401	1924	29,46556	1900	29,196
3752	58,33312	1927	29,64164	1903	28,85239
3758	57,85708	1934	29,64878	1909	29,23498
3758	57,72676	1951	29,73863	1915	30,2387
3770	57,8417	1951	29,85716	1919	29,86908
3816	58,57848	1956	30,1546	1932	30,3758
3818	58,5845	1957	30,0064	1938	30,78668
3829	61,82229	1961	30,17447	1941	29,71379
3841	58,78751	1965	30,9491	1943	29,80904
3875	59,11375	1970	29,867	1946	30,76788
3892	60,48572	1976	30,48016	1950	29,898
3901	59,75223	1987	30,40253	1957	30,05339
3901	59,98728	1988	30,18844	1960	31,7764

3907	64,30921	1991	30,60248	1966	30,16082
3909	61,96312	1997	31,4829	1971	30,2263
3911	60,19097	2004	31,59252	1971	31,05282
3919	59,99204	2011	30,88665	1978	30,4323
3923	62,93108	2016	31,1232	1980	30,226
3933	60,39691	2021	31,13773	1982	30,5328
3942	60,39072	2022	30,71286	1984	30,55376
3943	60,32303	2026	30,83312	1986	30,5251
3962	60,87202	2032	31,16448	2006	30,78144
3968	62,91664	2034	31,34646	2008	30,892
3987	61,14253	2042	31,00588	2009	30,9268
3993	67,11534	2048	31,4472	2013	31,02838
4000	66,82	2050	31,614	2013	30,66747
4003	61,26524	2061	31,55598	2013	30,51708
4033	61,49764	2070	31,9254	2015	30,7074
4038	62,00722	2074	31,4411	2026	31,05364
4049	62,29382	2082	31,88394	2026	31,92508
4061	62,37988	2085	31,9552	2030	31,126
4063	67,06666	2089	32,05568	2030	31,1163
4069	61,99983	2092	32,04104	2031	30,89996
4070	64,4413	2100	31,886	2045	32,95875
4072	63,62952	2102	32,37244	2050	31,7655
4086	62,25774	2103	32,8762	2051	31,55826
4103	65,2268	2109	32,47661	2052	31,27572
4119	64,05356	2110	32,5231	2052	31,22572
4123	68,05971	2116	32,4874	2054	32,8443

4134	65,04608	2117	32,46372	2061	31,63086
4149	66,55228	2117	32,18138	2065	31,5754
4157	65,7592	2121	34,46404	2071	32,81257
4161	65,67677	2131	32,73382	2075	31,71
4176	68,88488	2138	32,90002	2078	31,90638
4176	67,07496	2138	32,61794	2082	31,98214
4180	64,0716	2142	32,83704	2082	33,57312
4181	65,17654	2144	33,08024	2088	32,1176
4196	66,22564	2154	32,94156	2088	31,73232
4197	69,18609	2157	32,80698	2100	32,064
4200	64,444	2165	33,29955	2102	32,28938
4201	64,72329	2168	33,10176	2103	32,94371
4203	64,47966	2172	33,31752	2104	32,11392
4214	66,96408	2184	33,50968	2105	32,7333
4216	67,0996	2186	33,5565	2108	32,29944
4217	66,24869	2198	33,61772	2112	32,47688
4223	64,68284	2210	33,551	2114	31,98254
4226	71,21176	2212	33,8318	2118	32,45478
4241	65,29669	2214	34,52346	2133	32,6116
4249	66,82693	2224	33,80016	2134	32,74082
4263	67,33393	2241	34,37597	2147	32,74146
4264	65,25168	2250	34,5875	2150	33,026
4281	67,98889	2252	34,4678	2161	33,16491
4298	74,48012	2259	34,32421	2161	33,13364
4311	67,03251	2273	34,63049	2186	33,51278

Tabela A.8: Dados: Correlação Redis 10.000 Operações

Redis					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
32081	594,31687	15795	252,2171	15807	251,95692
32111	625,50163	15808	253,25936	15834	252,65202
32116	628,59088	15826	252,25952	15906	247,4318
32120	610,862	15857	244,38641	15936	254,96008
32125	631,73375	15859	253,54027	15960	255,5916
32149	628,47821	15863	248,24746	15964	254,02948
32162	608,0943	15919	252,0745	15974	256,6418
32402	611,4104	15924	250,73428	15994	255,41652
32439	629,76524	15936	251,93992	16021	251,19109
32479	621,10685	15956	253,71328	16022	256,34298
32491	643,38446	15963	255,64502	16040	255,6724
32540	609,6582	15984	251,10344	16054	250,24458
32572	604,98156	16064	253,50816	16131	251,18299
32601	599,53333	16064	250,25088	16150	258,0555
32645	659,50305	16085	255,2858	16219	258,83764
32697	637,73107	16094	255,28042	16238	259,84804
32833	606,06822	16160	255,9776	16239	260,17362
32926	618,6189	16264	253,30056	16322	252,66542
32967	633,31444	16283	253,46915	16327	260,82793
33379	617,79491	16312	254,31432	16343	253,3536
33662	654,0286	16760	260,7692	16723	262,64805
34131	668,54513	17026	264,75104	16865	264,6818

34221	631,71572	17055	264,95105	16905	269,87805
34639	644,14587	17083	268,15903	16932	259,90028
34663	676,32379	17136	274,5216	16955	265,5297
34707	659,25952	17181	266,88405	16972	262,17832
34758	646,52772	17196	268,01448	17002	264,12062
34770	682,027	17239	268,0606	17035	266,6655
34795	638,0545	17249	276,6844	17045	264,65895
34836	658,97096	17258	268,64062	17053	272,61051
34846	649,79456	17276	268,68484	17093	271,2922
34926	643,6378	17283	277,91763	17120	273,8544
34935	651,0016	17291	275,70378	17142	273,46378
34980	671,0272	17294	268,83528	17153	265,18937
34984	690,8344	17313	269,77447	17164	263,53952
35012	674,59348	17314	276,67898	17167	267,09009
35046	586,98242	17318	273,6272	17190	271,9641
35096	656,17264	17339	269,83975	17211	276,77802
35123	652,1719	17342	269,47104	17230	267,2552
35124	651,68844	17359	270,66347	17233	265,85024
35136	658,56216	17411	270,2361	17233	275,0748
35138	652,61554	17434	279,35172	17237	272,94506
35151	648,24708	17440	271,6408	17259	275,41817
35163	650,93553	17448	276,09952	17269	268,17225
35180	660,4012	17469	269,82853	17306	276,51584
35238	663,7414	17471	272,15398	17308	268,94624
35247	702,94711	17477	280,6612	17317	265,8784
35259	660,49752	17485	271,7038	17334	268,76352

35267	660,7651	17489	279,77862	17335	277,33955
35295	693,36695	17514	271,5399	17368	276,56712
35332	656,28656	17516	280,87508	17373	277,59507
35338	686,03154	17517	270,89629	17392	272,6976
35381	669,66549	17529	280,08776	17393	268,85932
35424	672,96144	17542	280,32436	17407	278,10327
35431	651,36154	17560	279,2316	17423	275,97882
35439	655,00426	17571	273,24204	17427	277,31693
35457	667,64124	17588	281,6228	17442	274,85702
35465	642,76705	17607	273,18889	17468	270,9202
35486	692,81122	17610	280,8472	17471	276,84159
35503	663,31105	17650	279,609	17481	280,83765
35506	685,04638	17671	274,70656	17484	275,76036
35506	665,97602	17701	279,98241	17485	279,6557
35511	667,26385	17749	283,06699	17525	279,40675
35565	668,0758	17841	281,95663	17526	271,81306
35600	661,262	17858	277,03148	17529	275,76428
35654	661,32198	17894	278,11304	17534	271,3202
35769	666,42184	17946	279,71002	17601	275,5353
35820	697,597	17958	278,78622	17661	279,47652
36228	689,49068	18018	289,2017	17740	282,8366
36261	672,46852	18072	280,99944	17768	274,9768
36321	684,70167	18114	282,78534	17870	278,0097
36341	674,01753	18141	286,76845	17884	280,26288
36479	705,18248	18311	284,99921	17888	287,74056
36523	694,0219	18347	290,01727	17927	285,71171

36788	687,20216	18417	286,20836	17961	281,16369

Tabela A.9: Dados: Correlação Redis 100.000 Operações

Redis					
Escrita		Leitura		Atualização	
Execução (ms)	Energia (J)	Execução (ms)	Energia (J)	Execução (ms)	Energia (J)
315269	5525,1757	155267	2439,64714	155129	2714,66532
315594	5516,58226	155373	2565,31571	155162	2718,44726
315668	5519,89372	156075	2577,4395	155295	2721,04465
316384	5566,13136	156197	2587,54061	155341	2718,22661
316467	5528,22043	156286	2448,69576	155380	2703,7626
316920	5574,4368	157400	2523,708	155454	2716,87626
317195	5521,90155	157459	2465,07254	155490	2717,2274
317473	5594,18392	157548	2488,84276	155566	2716,7398
317848	5551,2744	157687	2517,43171	156253	2721,4503
318128	5594,8784	157997	2474,86413	156315	2738,3232
318618	5513,85944	158007	2462,06563	156762	2749,76382
318765	5588,1745	158104	2477,64768	156854	2769,00766
318779	5586,1087	158196	2614,7798	157087	2742,14196
319683	5589,27624	158237	2573,51743	157420	2733,3476
320143	5586,13647	158632	2633,25424	157672	2729,61424
321116	5547,79132	159011	2620,25588	158204	2735,9106
321511	5595,39319	159037	2634,22796	158475	2769,87325
321650	5659,292	160198	2487,2993	159909	2784,92043
325670	5723,7207	162789	2521,17234	160969	2780,43973
332756	5905,919	163679	2532,79944	162655	2807,77875

334706	5911,99766	163907	2558,76245	162913	2813,16543
335998	5997,59942	165775	2571,09175	163528	2961,19312
337299	5860,19274	165835	2589,70735	164093	2834,07104
337574	5945,41924	166145	2583,62865	164954	2870,21712
338885	6024,8635	166518	2583,88216	165039	2878,82475
339323	5961,89898	166832	2598,65488	165232	2950,54088
339740	5860,7176	166889	2594,81615	165291	2954,02521
340458	6006,13366	166944	2593,21536	165313	2881,22439
340822	6034,92576	167865	2607,4907	166223	3015,53845
341502	6043,3602	168472	2641,06936	166260	2918,4934
341621	5938,2713	168736	2613,96624	166374	2941,46114
343279	5989,15196	168743	2611,75505	166622	2926,56598
343388	6116,9264	168817	2627,40363	166686	2918,95894
343508	5988,93256	169010	2619,881	166936	2912,194
343575	6052,47375	169035	2653,02865	166989	3034,10423
343937	6004,6761	169280	2650,9408	166995	2942,96455
344020	6052,116	169293	2653,85169	167024	3018,75168
344470	6025,0322	169355	2624,94115	167198	2907,50356
344639	6096,52031	169498	2653,83928	167496	2957,754
345082	6104,40738	169772	2663,22248	167629	3054,94112
345108	5999,47132	169916	2634,7374	167650	3057,4585
345124	6133,25472	169964	2672,68776	167758	2973,89224
345129	5989,36241	170008	2646,72296	167868	3068,03548
345342	6076,8984	170009	2665,14797	167919	3064,62447
345365	6113,60515	170140	2662,3868	167987	3007,12344
345403	6022,6159	170151	2646,31785	168019	3060,72747

345501	6074,15023	170202	2665,86878	168039	2915,34436
345635	5991,56105	170245	2642,7746	168108	3080,9608
345648	6022,69496	170329	2666,4179	168114	2980,3642
345995	6008,2436	170442	2672,71746	168161	3051,17559
346000	5996	170478	2670,50906	168369	3053,83928
346072	6002,10016	170671	2664,33933	168400	2952,942
346132	6055,46828	170691	2803,15758	168467	3068,24307
346208	6017,87616	170731	2653,0712	168523	3088,72207
346395	6004,81955	170818	2675,13816	168535	3068,0848
346680	6040,1872	171069	2669,87052	168546	3077,21288
347062	6036,20736	171069	2656,10983	168664	3033,6292
347360	6096,3928	171101	2649,15813	168668	3064,02704
347415	6014,4229	171189	2680,50658	168797	2950,03613
348219	6025,46537	171608	2661,15104	168866	3082,08258
348752	6096,98808	171880	2679,5168	168923	3094,36267
348767	6070,0651	172011	2670,17632	168924	2990,58644
349405	6159,48245	172054	2695,57242	168945	3048,27565
349463	6138,88927	172105	2818,9055	169022	3070,45462
349767	6114,46976	172117	2821,58787	169134	3080,36886
350062	6188,95806	172786	2665,71502	169314	3083,15768
350640	6220,36	173224	2784,74512	169349	3066,45084
351023	6172,94144	173769	2849,97646	170368	3103,1136
351605	6209,52045	174057	2700,51127	170456	3007,47224
351928	6204,11984	174450	2725,9285	170799	3101,85088
352724	6227,06548	175776	2893,30936	170869	3101,27963
355326	6269,08476	175934	2726,38888	171290	3056,1803

355743	6268,26561	175987	2802,3698	171306	3106,42568
356504	6243,5852	176209	2724,06754	171380	3104,1856
356510	6283,5359	176830	2758,283	171643	2990,64504