Pós-Graduação em Ciência da Computação

MATHEUS DORNELAS RODRIGUES

AVALIAÇÃO DE SGDBs NoSQL EM AMBIENTES DE NUVEM PRIVADA: Uma abordagem baseada em modelagem estocástica

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

MATHEUS DORNELAS RODRIGUES

AVALIAÇÃO DE SGDBs NoSQL EM AMBIENTES DE NUVEM PRIVADA: Uma abordagem baseada em modelagem estocástica

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.

Área de Concentração: Avaliação de Desempenho e Disponibilidade de SGBDs NoSQL.

Orientador: Prof. Dr. Eduardo Antônio Guimarães Tavares

Recife

2019

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

R696a Rodrigues, Matheus Dornelas
Avaliação de SGBDs NoSQL em ambientes de nuvem privada: uma abordagem baseada em modelagem estocástica / Matheus Dornelas Rodrigues. – 2019.
89 f.: il., fig., tab.

Orientador: Eduardo Antônio Guimarães Tavares.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências.

1. Avaliação de desempenho. 2. Rede de Petri estocástica. I. Tavares, Eduardo Antônio Guimarães (orientador). II. Título.

004.029 CDD (23. ed.) UFPE- MEI 2019-038

MATHEUS DORNELAS RODRIGUES

Avaliação de SGBDs NoSQL em ambientes de nuvem privada: Uma abordagem baseada em modelagem estocástica

Dissertação de Mestrado 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 em Ciência da Computação.

Aprovado em: 06/02/2019.

BANCA EXAMINADORA

_____

Profa. Dra. Bernadette Farias Lóscio (Examinador Interno)
Centro de Informática/UFPE

_____

Prof. Dr. Gustavo Rau de Almeida Callou (Examinador Externo)
Departamento de Computação/UFRPE

_____

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

Dedico este trabalho a minha família, que foram porto seguro perante as dificuldades durante este percurso.

AGRADECIMENTOS

Primeiramente, eu gostaria de agradecer aos meus pais e irmãos que me deram todo o apoio e suporte desde de minha graduação até o presente momento, onde mais uma etapa de minha vida está perto de ser concluída. Também agradeço aos meus grandes amigos que me acompanham desde de a infância.

Sou grato ao professor Eduardo Tavares pela orientação e apoio que foram essenciais para o desenvolvimento do trabalho e conclusão da pós-graduação. Também quero lembrar dos colegas que fiz durante o curso de mestrado, onde vivenciamos momentos de aprendizado e descontração.

Fui felizardo em ter escolhido o Centro de Informática para estudar, local onde a excelência está diretamente relacionada ao fato dos professores inspirarem os alunos a sempre explorarem novos caminhos e superarem seus limites.

RESUMO

Com o surgimento do Big Data, a necessidade de armazenamento e gerenciamento de grandes quantidades de dados se consolidou como algo fundamental para o crescimento e desenvolvimento de empresas. Nesse contexto, surgem os SGBDs NoSQL que são mais indicados para lidar com grandes volumes de informações e gerenciar dados não-estruturados, já que esses sistemas não possuem modelos de dados rígidos e nem utilizam a linguagem SQL. Muitos serviços de computação em nuvem adotaram os SGBDs NoSQL em seus produtos, mas a avaliação da qualidade de serviço (QoS) apresenta desafios adicionais para esses sistemas. Neste trabalho foi utilizada uma abordagem de modelagem hierárquica e heterogênea que parte da representação de componentes de alto nível até culminar na criação de modelos baseados em redes de Petri estocástica generalizada (GSPN) para avaliação conjunta de desempenho e disponibilidade de ambientes de computação em nuvem privada que adotam um SGBD NoSQL como sistema de armazenamento. Os modelos apresentados são capazes de estimar a taxa de transferência e a disponibilidade, que são indicadores proeminentes de QoS. Além disso, através de experimentos foi possível detectar o efeito da variação de disponibilidade na vazão do sistema e identificar quais são os componentes mais impactantes nos resultados obtidos. Os experimentos realizados demonstram a viabilidade prática da técnica e modelos propostos.

Palavras-chave: Rede de Petri estocástica. NoSQL. Desempenho. Disponibilidade.

ABSTRACT

The advent of Big Data created the need for storage and management of large amounts of data, and these practices has been consolidated as a fundamental requirement for the growth and development of companies. In this context, the NoSQL DBMSs suit better to handle large volumes of information and management of unstructured data since these systems do not have rigid data models and do not use the SQL language. Many cloud computing services have adopted NoSQL DBMSs in their products, but the assessment of quality of service (QoS) has additional challenges for these systems. In this work was utilized a hierarchical and heterogenous modeling approach that begins on the high level representation of components until gets on the creation of models based on generalized stochastic Petri nets (GSPN) for joint evaluation of performance and availability of private cloud computing environment using NoSQL DBMS as storage system. Besides that, through the execution of experiments was possible detect the effect of availability variation on system throughput and identify which component cause more impact on results. The carried out experiments demonstrate the practical feasibility of the proposed technique and models.

Keywords: Stochastic Petri net. NoSQL. Performance. Availability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
CTMC	*Continuos-Time Markov chains*
DBaaS	*Database as a Service*
DoE	*Design of Experiments*
FT	*Fault Tree*
GSPN	*Generalized Stochastic Petri Nets*
HTTP	*Hypertext Transfer Protocol*
IaaS	*Infrastructure as a Service*
IoT	*Internet of things*
MTTF	*Mean Time to Failure*
MTTR	*Mean Time to Repair*
NaaS	*Network as a Service*
NoSQL	*Not only SQL*
PN	*Petri Nets*
QoS	*Quality of Service*
QPN	*Queuing Petri Nets*
RBD	*Reliability Block Diagram*
SDN	*Software Defined Networking*
SGBDs	Sistemas de Gerenciamento de Banco de Dados
SLA	*Service Level Agreement*
SRN	*Stochastic Reward Nets*
YCSB	*Yahoo! Cloud Service Benchmark*

SUMÁRIO

1 INTRODUÇÃO

A evolução da Internet, a popularização de dispositivos móveis, o surgimento da *Internet of things* (IoT) em conjunto com o crescimento das redes sociais e o *e-commerce* fizeram com que o volume de dados trafegados na rede mundial de computadores crescesse drasticamente. Esses fatos contribuíram para a criação de um novo conceito chamado de Big Data, que caracteriza-se pelo volume, velocidade, variedade e valor no qual os dados são gerados (CHEN et al., 2014). Em outras palavras, Big Data é uma coleção de grandes bases de dados com muita diversidade de tipos, o que torna difícil o processamento utilizando métodos tradicionais (CHEN; ZHANG, 2014).

Atualmente, grandes empresas enfrentam problemas relacionados ao Big Data. Foi estimado que em 2015 foram gerados 8ZB (zettabytes), que consistia, em sua maioria, de dados não estruturados como emails e postagens em redes sociais (RAJARAMAN, 2016). Como outro exemplo, podemos citar o caso da rede de lojas de departamento Walmart, onde existem cerca de 267 milhões de transações por dia em suas 6000 lojas ao redor do mundo. Recentemente, o Walmart colaborou com a multinacional de tecnologia Hewlett Packard para estabelecer um *data warehouse* que tem a capacidade de armazenar 4 petabytes, rastreando cada registro de compra nos seus terminais de ponto de venda (CHEN; ZHANG, 2014). Além das áreas comerciais e redes sociais, outras atividades possuem um grande fluxo de informação, como, pesquisas científicas e setores públicos. Para exemplificar, em pesquisas da área astronômica são utilizados telescópios que possuem câmeras que podem gerar um grande número de imagens do universo, o *Large Synoptic Survey Telescope* é capaz de gravar 30 trilhões de bytes de imagens em um único dia (CHEN; ZHANG, 2014).

Os desafios impostos pelo Big Data se expandem para diversas etapas no tratamento de informação, como a captura, o armazenamento, a busca, o compartilhamento, a análise e a visualização. Isso exige que o poder computacional e a capacidade de armazenamento das máquinas sejam cada vez maiores. O surgimento de aplicações de análise em tempo real, serviços baseados em localização, redes sociais, e outros; exigem das bases de dados um grande volume de operações de leitura e inserção. Isso fez com que os Sistemas de Gerenciamento de Banco de Dados (SGBDs) relacionais fossem considerados inadequados para essas aplicações (GUDIVADA; RAO; RAGHAVAN, 2014).

Neste contexto, os SGBDs *Not only SQL* (NoSQL) têm se tornado opções alternativas aos sistemas de armazenamentos relacionais tradicionais para lidar com situações de Big Data. Esses sistemas não-relacionais possuem melhor desempenho para lidar com grande volume de dados, são fáceis de serem implantados como um sistema distribuído e manipulam dados não-estruturados de maneira mais eficiente. Além disso, os sistemas NoSQL permitem a adoção de diferentes níveis de consistência entre servidores. Os fatos citados,

anteriormente, como características dos SGBDs NoSQL são tidos como obstáculos para bancos de dados relacionais (CATTELL, 2011). Os sistemas NoSQL possuem mecanismos nativos de particionamento e replicação de dados. Eles são, tipicamente, desenvolvidos para funcionarem de maneira distribuída, em *cluster* de computadores com configurações modestas, escalando horizontalmente a carga de trabalho a qual são submetidos (GUDIVADA; RAO; RAGHAVAN, 2014).

Para suprir a necessidade crescente de poder computacional, a computação na nuvem aparece como uma das alternativas de solução para os desafios do Big Data pelo fato de oferecer "recursos infinitos", facilidade de aquisição, bom custo benefício e baixa complexidade para o usuário (GONZáLEZ-MARTíNEZ et al., 2015). O paradigma de armazenamento NoSQL vêm sendo adotado como uma alternativa de armazenamento em nuvem, serviço conhecido como *Database as a Service* (DBaaS) (DEKA, 2014). No entanto, desafios estão presentes quando se tenta avaliar a qualidade do serviço desses sistemas. Essa dificuldade é devido a alguns fatores, como, os diferentes tipos de carga de trabalho as quais os SGBDs são submetidos e o fato dos sistemas serem implantados em diferentes arquiteturas, possuindo, assim, valores de disponibilidade distintos (CHALKIADAKI; MAGOUTIS, 2012) (VENTURA; ANTUNES, 2016).

Métricas como vazão, tempo de resposta e disponibilidade são indicadores importantes para a avaliação da qualidade de serviço de sistemas em nuvem (BARDSIRI; HASHEMI, 2014). Na literatura atual, existe uma grande quantidade de estudos avaliando o desempenho de SGBDs NoSQL, mas há falta de trabalhos que abordem aspectos relacionados a disponibilidade (ABRAMOVA; BERNARDINO, 2013) (VENTURA; ANTUNES, 2016). Segundo Chalkiadaki e Magoutis (2012), *Quality of Service* (QoS) para sistemas de armazenamento ainda é considerada uma área de estudo pouco madura quando comparada a outras áreas, como, a área de redes.

Essa dissertação pretende avaliar o impacto que a variação da disponibilidade pode causar no desempenho de um ambiente de nuvem privada que utiliza um SGBD NoSQL como subsistema de armazenamento, considerando arquiteturas distintas, em relação ao número de servidores que formam a nuvem privada e seus níveis de redundância. Uma abordagem de modelagem com redes de Petri estocástica é utilizada para representar sistemas e comparar suas vazões e disponibilidades. A metodologia utilizada durante o desenvolvimento do trabalho ainda envolve a criação de um ambiente de nuvem privada para realização de experimentos de medição, a estimativa de valores médios de falha e reparo dos componentes da nuvem com modelos baseados em *Reliability Block Diagram* (RBD), além do desenvolvimento de modelos estocásticos com *Generalized Stochastic Petri Nets* (GSPN).

1.1 MOTIVAÇÃO

A demanda por escalabilidade de baixo custo levou à adoção de sistemas NoSQL (em contraste com bancos de dados relacionais tradicionais) para lidar com grandes volumes de dados e aplicações que lidam com um alto fluxo de informações (CHALKIADAKI; MAGOUTIS, 2012). Tais sistemas estão bem integrados com plataformas de processamento que são implementadas em infraestruturas de nuvem e oferecidas como serviços. As empresas provedoras têm que garantir a entrega total do serviço por meio do *Service Level Agreement* (SLA). Assim, avaliação de métricas que indicam a qualidade do serviço de acordo com a confiança que é depositada nele é uma atividade essencial para promover a melhoria da qualidade do serviço prestado e para o planejamento de infraestruturas de sistemas.

Enquanto existem muitas pesquisas avaliando o desempenho, a literatura acadêmica apresenta uma escassez de trabalhos que avaliam outros aspectos de qualidade, como aqueles ligadas ao conceito de disponibilidade. Determinadas arquiteturas de computação em nuvem possuem componentes que requerem maiores cuidados, pois além de impactar no provimento do serviço, podem ser importantes no desempenho. Modelagem analítica vem sendo bastante utilizada para avaliar os mais variados tipos de sistemas (LIRA; TAVARES; MACIEL, 2015). A avaliação de desempenho e disponibilidade por meio de modelos analíticos e de simulação é um método viável para propor melhorias aos sistemas de computação em nuvem.

Os resultados dessa dissertação permitem que administradores e analistas de tecnologia da informação possam detectar componentes críticos nas arquiteturas em nuvem que causam grande impacto no funcionamento do sistema.

1.2 OBJETIVOS

O objetivo do trabalho é apresentar um método de avaliação do impacto da disponibilidade no desempenho para ambientes de nuvem privada que utilizam SGBDs NoSQL como subsistema de armazenamento. A principal contribuição da dissertação é o desenvolvimento de modelos de rede de Petri estocástica generalizadas que sejam capazes de representar o sistema real, servindo como abstrações, e estimar os valores de disponibilidade e vazão. A principal vantagem na utilização de modelos é que simplifica e agiliza a realização de experimentos, já que não é necessário montar um ambiente real que envolve custos, alocação de recursos e tempo gasto com configuração do sistema.

A dissertação utiliza uma metodologia composta por etapas que envolvem experimentos no sistema real, modelos para estimar tempos médios de falha e reparo dos componentes da nuvem, concepção de modelos de rede Petri e experimentos que comprovam a utilidade do modelo. A avaliação de desempenho e disponibilidade baseada em modelos de rede de Petri estocástica torna possível a estimativa de métricas de QoS para avaliar

sistemas de armazenamento distribuídos. Mais especificamente, os objetivos do trabalho são:

- Construir uma nuvem privada com um SGBD NoSQL para realizar uma medição do sistema real;
- Estimar os tempos médios de falha e reparo dos servidores que compõe a nuvem privada;
- A construção de modelos de rede de Petri estocásticas que sejam capazes de avaliar o desempenho e a disponibilidade do sistema;
- Realizar experimentos que mostrem o impacto da disponibilidade na vazão do sistema;
- Fornecer conclusões que deem a administradores e analistas uma noção de quais componentes causam mais impacto na vazão e na disponibilidade.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

A dissertação está dividida em 6 capítulos, os quais serão brevemente destacados a seguir.

No Capítulo 2 serão descritos os trabalhos relacionados. O foco é em estudos que abordem a avaliação de desempenho, disponibilidade, técnicas de modelagem, ambientes de experimentação.

O Capítulo 3 apresenta a fundamentação teórica do trabalho proposto, uma visão geral sobre SGBDs NoSQL e suas categorias, conceitos básicos sobre avaliação de desempenho e disponibilidade, além dos principais métodos de avaliação. Também foram abordadas duas técnicas de modelagem de sistemas, RBD e *Petri Nets* (PN) com sua extensão estocástica, GSPN.

O Capítulo 4 mostra a metodologia utilizada para o desenvolvimento da solução proposta. O capítulo descreve as etapas partindo da coleta de dados no sistema real até culminar no modelo analítico capaz de estimar a vazão e a disponibilidade do sistema. Os modelos são apresentados e explicados.

No Capítulo 5 são mostrados os resultados dos três experimentos baseados em *Design of Experiments* (DoE) realizados para avaliar o desempenho e a disponibilidade de sistemas, com e sem redundância.

As conclusões e os trabalhos futuros são apresentados no Capítulo 6.

2 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relevantes ao estudo de SGBDs NoSQL e ambientes de computação em nuvem levando em conta as metodologias e ferramentas utilizadas. Além disso, os artigos citados realizam a avaliação dos sistemas utilizando métricas de QoS compatíveis com a motivação da dissertação. Os trabalhos relacionados foram agrupados em duas seções: avaliação de desempenho de SGBDs NoSQL e estudos de desempenho ou disponibilidade através de modelos analíticos para ambientes em nuvem.

Na última seção do capítulo é feita uma comparação entre artigos escolhidos e esta dissertação, para destrinchar as diferenças e semelhanças entre os estudos, além de deixar claro as contribuições desta pesquisa.

2.1 AVALIAÇÃO DE DESEMPENHO DE SGBDS NOSQL

A avaliação de desempenho pode ser realizada com métodos analíticos, simulação e medição no sistema real (JAIN, 1991). A escolha da técnica de avaliação depende do status de implementação, ou construção, do sistema a ser avaliado. Quando o sistema não pode ser utilizado pelo fato de estar em fase de desenvolvimento ou planejamento, é possível usar o método analítico ou simulação. Já a técnica de medição necessita que o sistema a ser avaliado esteja pronto para uso, ou que pelo menos exista um protótipo (JAIN, 1991).

Avaliar o desempenho de bancos de dados é uma área de pesquisa importante, visto que, o armazenamento de informações é uma função fundamental na maioria dos sistemas computacionais (OSMAN; KNOTTENBELT, 2012). Nos últimos anos, os pesquisadores tem dedicado atenção para a avaliação de desempenho de SGBDs NoSQL. Alguns trabalhos adotam uma abordagem que compara o desempenho de diversos sistemas NoSQL através de experimentos, submetendo os sistemas com diferentes modelos de dados a diversas configurações de cargas e arquiteturas de implantação. Alguns trabalhos que avaliam sistemas com arquiteturas simples e replicadas, geralmente, utilizam a vazão (taxa de transferência), tempo de execução e latência (tempo de resposta) como principais métricas de interesse. Outros, avaliam os sistemas utilizando técnicas de modelagem analítica, como, redes de filas e redes de Petri. Através dessas técnicas é possível avaliar aspectos como nível de consistência e replicação dos dados de maneira mais simples, verificando o impacto desses aspectos no desempenho do sistema sem a necessidade de realizar experimentos em cenários complexos.

Abramova e Bernardino (2013) compara o desempenho dos bancos de dados MongoDB e Cassandra com intuito de verificar a diferença no funcionamento dos SGBDs em uma máquina com poucos recursos, mais especificamente, pouca memória RAM. O critério de comparação foi o tempo que os sistemas levavam para executar operações dos seguintes

tipos: apenas inserção, apenas leitura, leitura-atualização, leitura-modifica-atualização e apenas atualização. Os SGBDs foram escolhidos pelo fato de pertencerem a duas categorias de NoSQL diferentes, o MongoDB possui o modelo de dados baseado em documentos, enquanto o Cassandra, baseado em colunas.

Os sistemas foram testados com cargas de trabalho com tamanhos diferentes: 100.000, 280.000 e 700.000 operações. Para operações de atualização o Cassandra obteve um resultado melhor, ou seja, menor tempo de execução para todos os tamanhos de carga. Para os demais tipos de operações o Cassandra obtém melhores resultados quando há mais dados armazenados no sistema. Conclui-se que o Cassandra é um sistema que tem muito a oferecer, principalmente quando se pretende trabalhar com grandes volumes de dados.

Em Klein et al. (2015) acontece a avaliação de desempenho de três SGBDs NoSQL (Riak, MongoDB e Cassandra) com o objetivo de analisar os sistemas em um cenário de produção, onde, para lidar com uma suposta grande demanda de acessos, os sistemas tiveram que ser implantados de maneira distribuída em nove servidores. Os experimentos foram feitos em duas configurações: um servidor e 9 servidores (representando um sistema distribuído entre *data centers*). Como cada SGBD tem suas particularidades, o modo de realizar as replicações e particionamento dos dados são diferentes e a avaliação dos resultados tenta ser justa, interpretando essas características dos sistemas. As métricas de interesse foram a vazão e a latência de operações de leitura e escrita. Os experimentos foram realizados simulando diferentes quantidades de usuários acessando o sistema, e de maneira geral o Cassandra obteve os melhores resultados quando o aspecto analisado foi a vazão. Quanto a latência das operações foi observada, o sistema que obteve melhores resultados foi o Riak.

Alguns dos fatos observados durante o processo foi que em todos os sistemas testados ao aumentar o número de usuários ocorria um aumento na vazão. Da mesma forma, quando é exigido um nível de consistência maior, a vazão diminui, por causa dos processos de certificação da replicação da informação.

Em Tang e Fan (2016) foram selecionados cinco SGBDs NoSQL para a realização de dois experimentos que comparavam o tempo de execução e a vazão de cada sistema numa configuração com 4 servidores. Os SGBDs escolhidos representam três modelos de dados distintos, MongoDB e Couchbase são orientados a documentos, Cassandra e HBase são orientados a colunas e o Redis possui o modelo chave-valor. O primeiro experimento foi para avaliar qual dos SGBDs consegue o menor tempo de execução, e o segundo experimento, para avaliar qual obtém a maior vazão com operações de leitura e atualização.

No resultado do primeiro experimento, o SGBD Redis obteve o melhor tempo de execução para inserir 100.000 registros, isto foi pelo fato do sistema armazenar os dados na memória RAM para depois persisti-los assincronamente no disco. Para os outros tipos de operação (leitura e atualização) o Redis mais uma vez se destacou com o melhor desempenho, enquanto os sistemas com o modelo de dados baseados em colunas (Cassandra e

HBase) obtiveram os piores resultados. O segundo experimento, verificou a vazão para execuções de uma carga que consistia em 50% de leituras e 50% de atualizações. O Redis mostrou-se performático até certo ponto, pois quando são realizadas muitas operações o seu desempenho cai devido ao fato de trabalhar com armazenamento na memória RAM, dessa forma, ela pode encher rapidamente. Quando a quantidade de dados armazenados no sistemas é grande, os SGBDs baseados em colunas apresentam o melhor desempenho, como foi observado no trabalho de Abramova e Bernardino (2013).

No trabalho de Gomes, Tavares e Junior (2016) além da avaliação de desempenho através de experimentos, como visto nos trabalhos descritos até agora, foi feito uma avaliação do consumo de energia dos SGBDs. O consumo de energia é um fator não-funcional que pode reduzir gastos e diminuir o impacto ambiental, além de impactar na confiabilidade do sistema pois pode estar relacionado com a temperatura em que equipamentos estão funcionando. Os sistemas analisados nesse trabalho já foram citados anteriormente: MongoDB, Cassandra e Redis. A ferramenta utilizada na medição do consumo da energia é o Emeter, que consiste de um hardware e um software capazes de estimar o consumo de energia e o tempo de execução dos sistemas.

Foi utilizado um *design* de experimentos fatorial, com 2 fatores (os SGDBs e os tipos de operação) e 3 níveis (MongoDB, Cassandra e HBase para o fator SGBD; e inserção, leitura e atualização para fator dos tipos de operação). A quantidade de operações foi variada, enquanto o número de usuários e o tamanho dos registros inseridos foi mantido o mesmo. No trabalho foi observado que há uma correlação linear e direta entre o tempo de execução e o consumo de energia elétrica. Assim como no trabalho anterior, o Redis obteve o melhor resultado em desempenho e consumo de energia para as operações de leitura e atualização para a carga de 1000 operações, e para as operações de inserção os resultados foram semelhantes em comparação aos outros sistemas, podendo ser considerado que todos são equivalentes. Para as cargas com 10.000 e 100.000 operações o Cassandra obteve os melhores resultados de desempenho para inserção e atualização, em compensação obteve os piores resultados para leituras em todos tamanhos de carga. O MongoDB obteve os resultados mais estáveis sem grandes alterações no tempo de execução ou no consumo de energia.

Em Barros, Callou e Gonçalves (2017) também é feita uma análise conjunta de desempenho e consumo de energia através da medição de um sistema um sistema de armazenamento de dados distribuídos, o Cassandra foi utilizado para representação. Fui utilizada a ferramenta *Yahoo! Cloud Service Benchmark* (YCSB) para gerar carga em diversos cenários variando a quantidade de operações de inserção e a quantidade de nós no *cluster* do Cassandra. Pelos resultados foi visto que o aumento no número de nós causa um aumento na vazão mas consequentemente também aumenta o consumo de energia elétrica.

2.2 AVALIAÇÃO DE DESEMPENHO OU DISPONIBILIDADE COM MODELOS ANALÍTICOS PARA AMBIENTE DE NUVEM

Modelos analíticos são estratégias comuns para a avaliação de desempenho e disponibilidade de sistemas na nuvem, a abordagem híbrida/heterogênea combinando modelos combinatoriais e analíticos é bastante adequada. Nessa dissertação é usada esta abordagem, como será visto no Capítulo 4.

Nesta seção serão falados sobre trabalhos que utilizam modelagem analítica ou modelagem heterogênea para avaliação de desempenho, disponibilidade ou uma avaliação conjunta desses aspectos sobre sistemas baseados em computação na nuvem, seja pública ou privada. A modelagem integrada com aspectos de desempenho e disponibilidade é chamada de modelagem de performabilidade, e permite a avaliação de desempenho considerando a falha dos componentes do sistema durante um determinado período de tempo (SAHNER; TRIVEDI; PULIAFITO, 1997).

Para realizar avaliações que mostram o efeito de eventos de falhas e manutenções no desempenho de sistemas é comum a utilização de técnicas de modelagem heterogênea que combinam um modelo de disponibilidade com alto nível de abstração e modelos de desempenho com menor nível abstração. Uma abordagem comum é a criação de modelos independentes de desempenho e disponibilidade, a interação desses modelos através de um novo modelo ou pela utilização de parâmetros derivados é o que torna possível rastrear o efeito de falhas no desempenho do sistema.

Em Torres, Callou e Andrade (2018) são avaliadas diversas métricas relacionadas com a disponibilidade e desempenho em serviços de armazenamento em nuvem privada. O estudo foi feito devido a popularidade de serviços como Dropbox, Google Drive e OneDrive. Foi proposta uma abordagem hierárquica baseada na composição de modelos de cadeias de Markov, redes de Petri estocásticas e diagrama de blocos de confiabilidade. A representação do sistema de armazenamento é composto de um controlador da nuvem, um nó de computação, o sistemas de armazenamento e o componente de rede. São avaliados diferentes configurações de redundância e os resultados mostram que quando existe a redundância em todos os componentes o sistema apresenta os melhores valores para disponibilidade.

As métricas calculadas e que são relacionadas ao desempenho, foram: a probabilidade do buffer da rede estar cheio (PBF), probabilidade de ocorrer um *timeout* (POT), probabilidade um usuário não ser atendido devido à falhas (NUF), utilização do sistema (SU) e a vazão do sistema (ST). Os resultados mostram que aumentar a disponibilidade do sistema reduz levemente a probabilidade do *buffer* da rede encher, assim como também reduz a probabilidade de que um *timeout* ocorra. Ao longo do trabalho foi comprovado que o aumento da disponibilidade influência beneficamente as demais métricas de desempenho propostas.

O trabalho de Gandini et al. (2014) tem como objetivo avaliar o desempenho de

três sistemas de armazenamento já vistos anteriormente, o MongoDB, o Cassandra e o HBase. O intuito é entender o efeito de diferentes configurações de máquinas virtuais no desempenho dos sistemas. A partir de experimentos realizados na Amazon EC2 [1] foram coletados dados para parametrizar os modelos de rede de filas capazes de representar o sistema em diferentes configurações. Durante os experimentos para a parametrização do modelo foi considerado o sistema com uma e várias máquinas virtuais. Para o experimento com apenas uma VM, o número de núcleos do processador foi variado; e para caso do sistema distribuído com várias VMs, foram variados a quantidade de nós e o fator de replicação dos dados. O objetivo dos modelos propostos foi o de estimar valores de vazão e tempo de resposta para as operações de leitura e escrita nos SGBDs.

A técnica de modelagem utilizada foi a de redes de filas estocásticas e no modelo são representados os seguintes elementos: o envio de carga, a distinção do tipo de operação, o atraso da rede, o processamento na CPU e o armazenamento no disco. Para as configurações com mais de uma VM, os elementos do modelo que representam o processamento e o armazenamento são replicados de acordo com o número de nós que se queira simular. Os resultados do modelo são validados com os resultados obtidos nos experimentos com um nível de confiança de 95%. Para o MongoDB e Cassandra, o modelo conseguiu reproduzir fielmente a vazão e o tempo de resposta, mas para o sistema HBase, considerado mais complexo, os resultados possuem uma grande diferença significativa.

Em Dipietro, Casale e Serazzi (2017) os autores focam no sistema Cassandra e abordam o desafio de encontrar a configuração ótima em termos de número de nós, número de réplicas e consistência dos dados para evitar o desperdício de recursos e gastos excessivos. Achar a melhor configuração entre consistência de dados, custos e desempenho requer uma abordagem quantitativa que deve avaliar todas essas dimensões. O modelo proposto é capaz de estimar o tempo de resposta e vazão do sistema. Para parametrizar o modelo foram observados a utilização de disco, redes e CPU; e para obter esses valores foi montado um *cluster* com 4 nós do Cassandra em um ambiente de nuvem privada, com uma máquina extra dedicada a ser o gerador de carga. O sistema com 4 nós possuía um fator de replicação 3, ou seja, existem três cópias de cada informação.

Antes de comparar os resultados, foi percebido uma relação entre o nível de consistência adotado, a vazão e o tempo de resposta das operações. Quanto maior o nível de consistência, menor a vazão e maior o tempo de resposta. Comparando os resultados do modelo com os valores do sistema real, foi notado que o modelo captura as principais mudanças nas configurações do sistema. De acordo com os resultados, o erro médio da vazão está abaixo dos 7% e do tempo de resposta abaixo dos 10%. Ainda para verificar aplicabilidade do modelo, ele foi utilizado para avaliar outro sistema NoSQL baseado em colunas. Após ajustes para representar mais fielmente o funcionamento do novo sistema, o

[1] O Amazon Elastic Compute Cloud (Amazon EC2) é o serviço de computação escalável na nuvem da Amazon Web Services (AWS).

modelo obteve resultados considerados bons em comparação ao sistema real, apresentando resultados com erros em torno de 8% para vazão e 12% para tempo de resposta.

Osman e Piazzolla (2014) fizeram um estudo muito parecido com o de Dipietro, Casale e Serazzi (2017), pois avalia a relação entre o tamanho do *cluster*, o fator de replicação e o nível de consistência. Contudo, a técnica de modelagem analítica utilizada é diferente, neste trabalho é usado *Queueing Petri Nets* (QPN). Para parametrização e validação foram feitos experimentos com diferentes números de nós (de 2 até 4), fatores de replicação (1 até o número de nós) e níveis de consistência (de 1 até o fator de replicação). O modelo QPN obteve bons resultados prevendo o aumento linear nos tempos de resposta para todas as configurações do sistema, pois reflete o bloqueio, enfileiramento e processamento das solicitações.

No entanto, os valores do modelo são levemente menores, já que não está representado o efeito do atraso da rede e o aumento do processamento em decorrência da sincronização entre os nós. O modelo QPN fornece excelentes previsões para as configurações com 2 nós. Para *clusters* de 3 e 4 nós, as predições têm um erro médio que variam entre 2% e 20% para a maioria das configurações, tanto nos tempos médios de resposta quanto para a vazão. Na condução dos experimentos, foi notado que quando o fator de replicação se aproxima do tamanho do *cluster*, a estabilidade dos servidores e seu desempenho são afetados, isso pode estar acontecendo devido a alta sincronização que provoca um aumento no tráfego.

A partir dos resultados conclui-se que mantendo o nível de consistência e tamanho do *cluster* aumentar o fator de replicação proporciona melhor desempenho. Em contraste, manter o tamanho de *cluster* e o fator de replicação aumentando o nível de consistência, piorará o desempenho devido ao aumento do tempo de espera para que as solicitações extras sejam roteadas de volta para o nó original. Dado que o fator de replicação e o nível de consistência são fixos, o desempenho melhora com o aumento no número de servidores, apresentando um crescimento linear na vazão.

Bruneo (2014) apresenta um modelo construído com *Stochastic Reward Nets* (SRN) que representa um sistema de nuvem que consegue capturar dois dos conceitos fundamentais de *Infrastructure as a Service* (IaaS): escalabilidade e flexibilidade. O modelo é capaz de representar sistemas compostos por milhares de recursos, físicos ou virtuais, explorando conceito de elasticidade do sistema. Através do modelo concebido é possível calcular diversas métricas que interessam ao provedor do serviço e ao usuário final. Algumas das métricas possíveis de serem calculadas são: utilização do *data center*, disponibilidade, tempo de espera dos usuários na fila, o tempo para uma máquina virtual ser criada e responsividade. O autor ainda realiza uma avaliação de desempenho variando a quantidade de recursos físicos e virtuais, o tamanho da fila, ajustando o nível de elasticidade.

Em Lima et al. (2014) é apresentado uma modelagem hierárquica e heterogênea para avaliar o desempenho de uma central de atendimento de emergências considerando a ocor-

rência de falhas. O termo heterogêneo é pelo fato do processo unir técnicas de modelagem baseadas em estados e combinatoriais, e a modelagem hierárquica é essencial quando o objetivo é representar sistemas grandes com mecanismo de redundância e manutenção. Segundo o processo de avaliação descrito pelo autor, após o entendimento do problema, é necessário a criação de modelos utilizando RBD para estimar os tempos médios de falha e reparo dos componentes do sistemas de nuvem. Com esses modelos ainda é possível calcular a disponibilidade, *downtime*, tempo de funcionamento e confiabilidade. Para representar de maneira mais detalhada os mecanismos de redundância foram criados dois modelos em rede de Petri estocástica.

Para a avaliação de desempenho foi criado um novo modelo em rede de Petri estocástica (SPN) que permite estimar a duração média de chamadas, o número de chamadas perdidas, o número de desligamentos, trotes, ligações válidas e inválidas, entre outras métricas. Essas medidas de desempenho são importantes para validar o modelo e saber a relação entre o número de chamadas e a quantidade de atendentes necessária.

Outro estudo deste tipo para sistemas na nuvem foi feito por Kirsal et al. (2015), que utiliza modelos Markovianos para construir uma solução que avalie o funcionamento do sistema. A técnica utilizada também é considerada hierárquica e oferece meios de avaliar conjuntamente o desempenho e a disponibilidade. Nessa abordagem taxas de recompensa obtidas no modelo de desempenho servem como parâmetros para o modelo de disponibilidade. O desempenho é expresso pela quantidade de requisições simultâneas no sistema. No modelo foi considerado que a chegada e processamento de requisições acontecem de acordo com a distribuição exponencial, a capacidade da fila é infinita e o sistema possui N servidores. Dessa forma um modelo de fila M/M/C se ajusta a solução proposta. De acordo com os resultados apresentados, o modelo que considera a falha dos componentes obtém um desempenho menor do que o modelo de desempenho puro, como era esperado.

Em Ghosh et al. (2010) os autores estão interessados em analisar o desempenho ponta-a-ponta de um sistema de IaaS com a presença de falhas e reparos. Logo no início do artigo é salientado a dificuldade de avaliar métricas de QoS baseados apenas em experimentos e medições, devido ao fato de não ser possível a captação de eventos de falhas suficientes para quantificar seus efeitos nos resultados. A contribuição do autor pode ser resumida em dois pontos: a construção de um modelo aplicável a diversos sistemas e o fato do modelo ser capaz de quantificar o efeito de mudanças na carga de trabalho, falha de componentes e capacidade na qualidade de serviço.

O oferecimento de serviço de computação na nuvem foi dividido em três passos: decisão de oferecimento de recursos, criação da máquina virtual e execução do serviço. Para a representação dessas etapas através de modelagem foi criado um modelo puro de desempenho, outro de disponibilidade e um terceiro modelo, que representa a interação entre os dois primeiros. Este último modelo citado é capaz de calcular métricas de QoS como a probabilidade de rejeição de requisições e o tempo médio de oferecimento de serviço. Além

disso, consegue mostrar como variações na carga de trabalho, falha de equipamentos e capacidade do sistema afetam as métricas de interesse.

A limitação de utilizar apenas experimentos na avaliação de métricas de QoS relacionadas a disponibilidade fica clara no trabalho de Ventura e Antunes (2016), onde um estudo de aspectos relacionados a confiabilidade e desempenho é conduzido dessa forma. É feita uma avaliação em SGBDs NoSQL utilizando injeção de falhas através da reinicialização das máquinas. Os sistemas avaliados foram o MongoDB, Cassandra e Redis e os aspectos avaliados são disponibilidade e integridade dos dados. Durante a execução de cargas, o sistemas é exposto a falhas que encerram o sistema de maneira correta e de maneira abrupta, de modo que a efetividade das técnicas de tolerância falhas é posta à prova. Após a restauração do funcionamento e o término da execução da carga, a integridade dos dados, o tempo de recuperação e a vazão são avaliadas. Os resultados apontam que falhas influenciam na integridade dos dados e possuem efeito negativo na vazão, mas os autores admitem que houve alguns problemas durante o desenvolvimento do estudo mesmo utilizando um modelo de falha simples.

Em Silva, Maciel e Zimmermann (2013) é proposta uma abordagem com modelos SPN para avaliar métricas de desempenho e disponibilidade em IaaS geograficamente espalhado levando em consideração a ocorrência de desastres. Novamente é utilizado uma modelagem heterogênea e hierárquica onde foram criados modelos RBD para calcular valores médios de tempo de falha e reparo, e esses valores são associados aos submodelos SPN, que combinados em um modelo final representam todo o sistema. Para representar a infraestrutura são criados três submodelos: um componente genérico que simula equipamentos que não possuem redundância e possuem dois estados de funcionamento (ligado e desligado), um componente relacionado com desempenho que combina o estado da máquina virtual com a chegada de requisições ao *data center* e a geração de requisições. O último submodelo é o componente de transmissão, responsável por mostrar a migração de máquinas virtuais entre *data centers* quando acontece falhas ou sobrecarga.

Foram definidas duas métricas, a utilização da máquina baseada no número de VMs (que interessa provedor de serviço) e a probabilidade que uma tarefa seja completado sem erros (que interessa ao usuário final). Os resultados mostram que há uma relação entre a distância entre o cliente e o servidor, e outra relação entre a utilização da máquina e a ocorrência de desastres.

Wang, Chang e Liu (2016) conduz um estudo bastante semelhante ao anterior, com o objetivo de modelar a infraestrutura provedora de serviço em nuvem simulando com precisão situações como o intervalo para criação de VMs e acionamento dos mecanismos de recuperação. O modelo é construído usando *Continuos-Time Markov chains* (CTMC) e tem como principais contribuições a descrição detalhadas das regras de transição dos estados do modelo. O artigo ainda mostra os resultados que validam o modelo em comparação a um sistema real para as seguintes métricas: probabilidade rejeição de requisição

e o tempo de resposta para criação de VMs.

2.3 COMPARAÇÃO DOS TRABALHOS RELACIONADOS

Nesta seção será feito uma comparação entre o estudo desta dissertação e os trabalhos relacionados citados anteriormente. O objetivo é mostrar as semelhanças e diferenças para destacar as contribuições que a pesquisa desenvolvida para esta dissertação trouxe. As características abordadas na comparação são: modelos baseados em cadeia de Markov (CTMC), modelos baseados em PN (PN), modelagem com RBD (RBD), modelos baseados em redes de filas (QN), avaliação de disponibilidade (Disp.), avaliação de desempenho (Desemp.), avaliação conjunta de desempenho e disponibilidade (Desemp. + Disp.), medição no sistema real (Medição), utilização de ambiente de nuvem (Ambiente de nuvem) e SGBD NoSQL como objeto de estudo (NoSQL).

A Tabela 2 resume as características de comparação e os trabalhos relacionados, sendo que a primeira linha corresponde a esta dissertação. Conforme pode ser visto na tabela, a maioria dos trabalhos citados que possuem SGBDs NoSQL como objeto de estudo (coluna A10) realizam apenas a avaliação de desempenho (coluna A6) pelo fato desse aspecto ser avaliado de maneira confiável pela técnica de medição (coluna A8). Isso contribui na escassez de pesquisas que envolvem sistemas de armazenamento e avaliação de disponibilidade (coluna A5). Uma das diferenças que a pesquisa desenvolvida nesta dissertação propõe perante os trabalhos relacionados é a concepção de modelos capazes de avaliar o impacto de falhas no desempenho de um ambiente de nuvem privada com um subsistema de armazenamento NoSQL. O processo de desenvolvimento do modelo GSPN possui uma abordagem heterogênea, com a utilização de modelos RBD e GSPN para chegar no resultado final.

Nos artigos citados, a avaliação de desempenho geralmente é feita apenas com a coleta (medição) de valores do sistema real. Apenas 4 dos artigos citados que analisam desempenho utilizam a medição para parametrizar modelos analíticos, esses artigos são os de Gandini et al. (2014), Dipietro, Casale e Serazzi (2017), Osman e Piazzolla (2014) e Wang, Chang e Liu (2016). E, desses trabalhos, apenas o de Osman e Piazzolla (2014) utiliza um modelo baseados em rede de Petri, a técnica aplicada é chamada de *Queuing Petri Nets* (QPN).

Nesta dissertação, a parametrização dos recursos responsáveis pelo cálculo da métrica do desempenho no modelo GSPN são derivados de medições no sistema real. Similarmente a alguns dos trabalhos vistos, um ambiente de nuvem privada, foi criado um sistema para a realização de experimentos e medição do sistema em funcionamento. A utilização da computação em nuvem é uma característica comum nos estudos destacados por esse capítulo, a possibilidade de virtualização de componentes permite criar cenários com diversas configurações.

O trabalho de Ventura e Antunes (2016) é o único, dentro no nosso conhecimento, que realiza uma avaliação conjunta de desempenho e disponibilidade de um SGBD NoSQL através da técnica de injeção de falhas para avaliar o impacto da parada do sistema na integridade dos dados e na vazão. Como foi dito na seção anterior, a avaliação de aspectos de disponibilidade a partir de falhas em sistemas reais não é ideal, pois a quantidade de falhas obtidas pode não ser compatíveis com a realidade. Por isso, é importante a utilização de modelos que além de acelerar o processo de experimentação também facilita na representação de sistemas com configurações diferentes.

A utilização de técnicas de modelagem se torna mais comum quando são analisados aspectos de disponibilidade, confiabilidade, manutenabilidade, entre outros. Os trabalhos de Lima et al. (2014) e Silva, Maciel e Zimmermann (2013) utilizam modelos RBD e GSPN da maneira semelhante ao que será utilizado neste estudo, com modelos RBD representando os componentes da infraestrutura da nuvem e calculando os tempos médios de falha e reparo dos servidores; e modelos analíticos simulando o sistema com maior nível de detalhes, mecanismos de redundância mais complexos, balanceamento de carga, processamento de requisições, etc. As métricas de tempos médio de falha e reparo que são extraídas dos modelos RBD são utilizadas como parâmetros de entrada para os modelos analíticos e possibilitam o cálculo da métricas de disponibilidade.

O presente trabalho pretende apresentar uma forma de avaliação de conjunta para SGBDs NoSQL diferente das que são utilizadas atualmente. Unindo os métodos comuns de avaliação de desempenho e disponibilidade para formar uma solução conjunta de análise para um tipo de sistema que carece de uma análise mais detalhada. E, por fim, mostrar que a metodologia utilizada é bastante sólida e aplicável para diversas arquiteturas de implantação de SGBDs, além de permitir que analistas de sistemas tirem conclusões sobre o funcionamento do sistema.

Tabela 2 – Comparação de trabalhos relacionados.

Artigo	**CTMC**	**PN**	**RBD**	**QN**	**Disp.**	**Desemp.**	**Desemp. + Disp.**	**Medição**	**Ambiente de nuvem**	**NoSQL**
Este trabalho	-	X	X	-	X	X	X	X	X	X
Abramova e Bernardino (2013)	-	-	-	-	-	X	-	X	-	X
Klein et al. (2015)	-	-	-	-	-	X	-	X	X	X
Tang e Fan (2016)	-	-	-	-	-	X	-	X	X	X
Gomes, Tavares e Junior (2016)	-	-	-	-	-	X	-	X	-	X
Barros, Callou e Gonçalves (2017)	-	-	-	-	-	X	-	X	-	X
Torres, Callou e Andrade (2018)	X	X	X	-	X	X	X	X	X	-
Gandini et al. (2014)	-	-	-	X	-	X	-	X	X	X
Dipietro, Casale e Serazzi (2017)	-	-	-	X	-	X	-	X	X	X
Osman e Piazzolla (2014)	-	X	-	-	-	X	-	X	X	X
Bruneo (2014)	-	X	-	-	X	-	-	-	X	-
Lima et al. (2014)	-	X	X	-	X	X	X	-	X	-
Kirsal et al. (2015)	X	-	-	-	X	X	X	-	X	-
Ghosh et al. (2010)	X	-	-	-	X	X	X	-	X	-
Ventura e Antunes (2016)	-	-	-	-	X	X	-	X	-	X
Silva, Maciel e Zimmermann (2013)	-	X	X	-	X	X	X	-	X	-
Wang, Chang e Liu (2016)	X	-	-	-	X	X	X	X	X	-

3 REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos básicos que foram utilizados ao longo do trabalho. A primeira seção apresenta os SGBDs NoSQL e suas principais características. A segunda e terceira seções mostram, respectivamente, as técnicas de avaliação de desempenho e disponibilidade. A quarta seção apresenta a técnica de modelagem de rede de Petri (PN) e rede de Petri estocástica generalizada (GSPN) capazes de construir modelos que descrevem o sistema possibilitando o cálculo de métricas de desempenho e disponibilidade. A quinta seção fala sobre a técnica de modelagem combinatorial RBD, muito comum na avaliação de confiabilidade, manutenabilidade, disponbilidade, entre outras métricas.

3.1 SGBD NoSQL

Com o surgimento do fenômeno Big Data a necessidade de armazenamento e gerenciamento de grandes quantidades de informação foi se consolidando como algo fundamental para o crescimento e desenvolvimento de empresas. A grande quantidade e a alta variedade no formato dos dados é uma das características do Big Data, devido a esse motivo surge como desafio o ato de capturar, armazenar, processar, analisar e interpretar a informação. É neste contexto que surgem os SGBDs NoSQL, que possuem modelos de dados especificamente projetados e otimizados para situações como esta (GUDIVADA; RAO; RAGHAVAN, 2014).

O conceito de NoSQL que é mais difundido atualmente começou a ganhar força por volta dos anos 2000 com o desenvolvimento de SGBDs não relacionais que manipulavam os dados de forma diferente dos sistemas relacionais. A premissa desses sistemas era suprir as dificuldades enfrentadas pelos sistemas de armazenamento mais comuns no mercado, como: ser facilmente escalável para trabalhar como um sistema distribuído, ter alto desempenho na manipulação de grandes volumes de dados e oferecer alta disponibilidade (ABRAMOVA; BERNARDINO, 2013). Os SGBDs DynamoDB (DECANDIA et al., 2007) e Big Table (CHANG et al., 2008) foram desenvolvidos, respectivamente, pelas empresas Amazon e Google, que foram grandes contribuintes no desenvolvimento e afirmação dos bancos de dados NoSQL (ABRAMOVA; BERNARDINO, 2013) (CATTELL, 2011). Em Cattell (2011), as principais características dos sistemas NoSQL atuais são resumidas em seis pontos:

1. Facilidade em escalar horizontalmente, dividindo a carga de trabalho em vários servidores;
2. Mecanismo de replicação e particionamento da base de dados;
3. Interface simples para manipulação da base de dados;

4. Controle de concorrência menos rígido quando comparado ao modelo relacional com suas propriedades ACID, os SGBDs NoSQL utilizam o modelo BASE (MCCREARY; KELLY, 2013):

    a) Essencialmente disponível (*Basically Available*) - isso indica que o sistema permite um certo grau de inconsistência nas informações em favorecimento da disponibilidade;

    b) Estado impreciso (*Soft State*) - essa limitação informa que o sistema não oferece consistência da informação em todas as ocasiões;

    c) Eventualmente consistente (*Eventual Consistent*) - o sistema sempre está apto a receber novas operações sem se importar com o estado de operações anteriores, portanto, o sistema por si só não sabe o estado dos seus dados.

5. Uso eficiente de índices e memória RAM para armazenar os dados;

6. Capacidade de armazenar dados com estruturas mutáveis, atributos podem ser removidos ou inseridos. Pode-se tomar como exemplo o armazenamento de objetos no formato JSON.

3.1.1 Classificação de SGBDs NoSQL

3.1.1.1 Teorema CAP

Um tipo de categorização aplicada aos sistemas NoSQL é em relação a um conceito criado por Eric Brewer chamado de Teorema CAP, o qual afirma que um sistema só pode oferecer, simultaneamente, duas das três características: consistência, disponibilidade e particionamento (MCCREARY; KELLY, 2013). Na Figura 1 é mostrado a classificação de alguns SGBDs NoSQL pelo teorema CAP. O teorema simplifica a decisão na escolha do bancos de dados a ser adotado da seguinte forma:

- Consistência e Disponibilidade - o sistema não divide a base de dados. Todos os dados são replicados por completo procurando manter a mesma consistência;

- Consistência e Particionamento - a base de dados está dividida em várias partes, mas não exitem réplicas, portanto não há necessidade de sincronização de informações, garantindo consistência;

- Disponibilidade e Particionamento - o sistema mantêm-se disponível através das réplicas, que irão sincronizar a cada atualização das informações, não dá para garantir que os dados em todos os servidores estarão no mesmo estado.

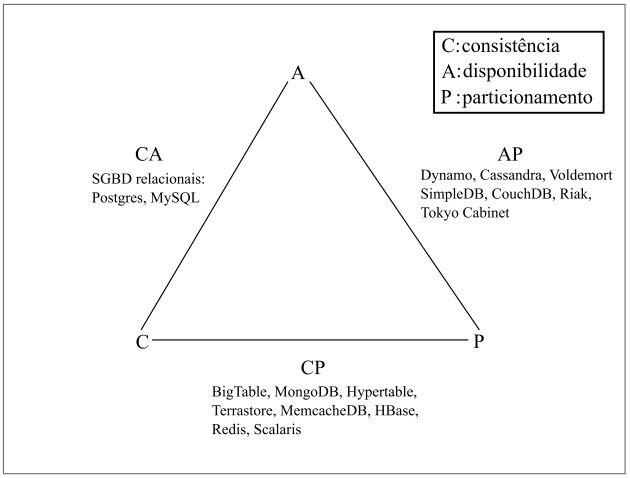


Figura 1 – Classificação de SGBDs NoSQL pelo Teorema CAP.
Fonte: Adaptado de (CAI et al., 2013)

3.1.1.2 Modelos de dados

Outro tipo comum de classificação dos SGBDs NoSQL é quanto ao modelo de dados, os sistemas podem ser classificados em quatro categorias principais: chave-valor, orientado a colunas, orientado a documentos e orientado a grafos (CATTELL, 2011). De maneira geral, os sistemas de cada uma dessas categorias procuram oferecer uma solução para o armazenamento de grande volume de dados não-estruturados de maneira distribuída. Sendo assim, cada sistema de armazenamento tem autonomia sobre outros aspectos importantes como o controle de concorrência, o algoritmo de particionamento dos dados, replicação dos dados e mecanismos de recuperação de falha.

3.1.1.2.1 Modelo chave-valor

É o modelo mais simples de armazenamento NoSQL, pois sua estrutura é composta de duas partes: a chave e o valor. A informação no campo chave deve ser uma cadeia de caracteres única, mas seu conteúdo pode variar podendo ser o diretório para o arquivo, um valor randômico, um rota de requisição para uma interface REST ou até uma consulta SQL (MCCREARY; KELLY, 2013).

A manipulação dos dados nos sistemas chave-valor é simples: não existe linguagem de consulta e todo acesso e modificação na base de dados ocorre em torno de três funções: inserção, remoção e leitura. De maneira geral o acesso à informação ocorre exclusivamente buscando pelo campo chave. Qualquer outro tipo de consulta mais sofisticada deve ser implementado na camada de aplicação (MCCREARY; KELLY, 2013) (HECHT; JABLONSKI, 2011). A Figura 2 mostra como é a estrutura de 3 registros chave-valor. O primeiro registro possui o campo chave "User:2:friends", e é pelo campo chave que o registro deve ser consultado; e o campo valor é um conjunto de números inteiros. Nos outros dois registros, exemplificados na imagem, possuem no campo valor um conjunto de objetos

chave-valor ({Theme:dark, cookies:false}) e um *array* de inteiros ([234,3466,86,55]). Isso mostra a variedade de tipos que os SGBDs NoSQL podem lidar.

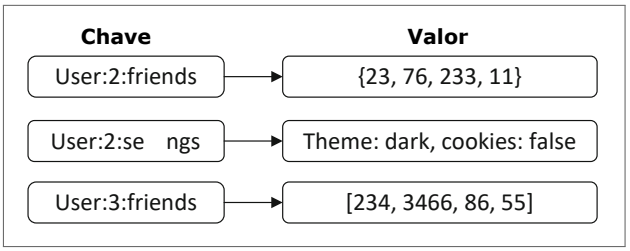


Figura 2 – Representação de um registro no modelo Chave-valor.
Fonte: Adaptado de (GESSERT et al., 2017)

3.1.1.2.2 Modelo orientado a colunas

Esse modo de armazenamento utiliza os conceitos de linhas e colunas semelhantes ao modelo relacional, mas não existem tabelas. As colunas podem ser criadas à medida que os dados são inseridos e não há tipos pré-estabelecidos. A representação visual dos bancos de dados orientados a colunas, como mostrado na Figura 3, é semelhante a uma tabela de banco de dados relacional com o conceito de linhas e colunas. As linhas possuem o campo chave (Marcelo, Vanessa) para acesso e modificação, as colunas podem ser agrupadas em um conceito chamado de "Família de Colunas"(Escola, Informações). A grande diferença desses sistemas para os SGBDs relacionais é que os registros podem ter um número de colunas arbitrário, aumentando conforme os dados são inseridos. Uma funcionalidade comum nesse tipo de banco de dados é o versionamento da informação, na Figura 3 é possível notar que algumas informações possuem várias versões. A maioria das implementações dos sistemas que seguem esse padrão estrutural foram baseadas no sistema BigTable (CHANG et al., 2008).

De forma semelhante ao modelo de dados anterior, o modelo orientado a colunas não se preocupa em representar o relacionamento entre as informações, mas é possível agrupar as colunas em grupos pré-definidos pelo usuário, tentando fazer com que valores que possam ser semanticamente relacionados fiquem próximos quando armazenados. O banco de dados Cassandra introduziu um nível de agrupamento de colunas, chamado colunas compostas, que tornou possível a manipulação de dados com estruturas mais complexas e expressivas (CATTELL, 2011).

3.1.1.2.3 Modelo orientado a documentos

Considerado o padrão de armazenamento que lida melhor com dados de estrutura complexa, pois consegue representar os relacionamentos entre informações de maneira eficiente. São considerados extensões dos sistemas de armazenamento chave-valor nos

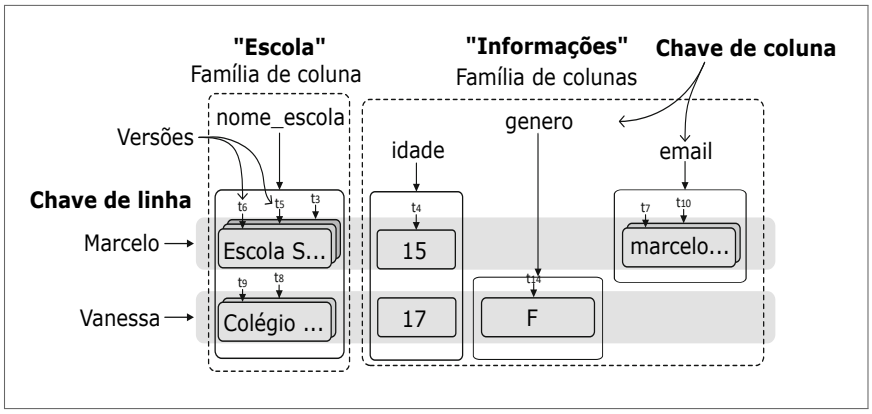


Figura 3 – Representação de um registro no modelo orientado a coluna.

Fonte: Adaptado de (GESSERT et al., 2017)

quais o campo valor é representado como um documento organizado em formatos semi-estruturado, por exemplo, XML, JSON ou BSON. Um documento possui um esquema flexível por meio da adição ou remoção de seus atributos em tempo de execução (DAVOUDIAN; CHEN; LIU, 2018).

Também é comum aos SGBDs orientados a objeto o mecanismo que torna possível relacionar um documento a outro, geralmente isso ocorre fazendo referência a algum dos campos chave do documento que se quer relacionar, assim como uma chave estrangeira do modelo relacional. Outra característica bastante importante é o motor de buscas que permite realizar consultas complexas, geralmente a partir de objetos no formato JSON (DAVOUDIAN; CHEN; LIU, 2018). A Figura 4 mostra como é a estrutura de um registro no modelo de documento, todos os registros possuem um campo chave que é único e obrigatório. O elemento *customer* possui como valor um outro documento aninhado, enquanto o elemento *items* tem como valor um *array* de documentos.

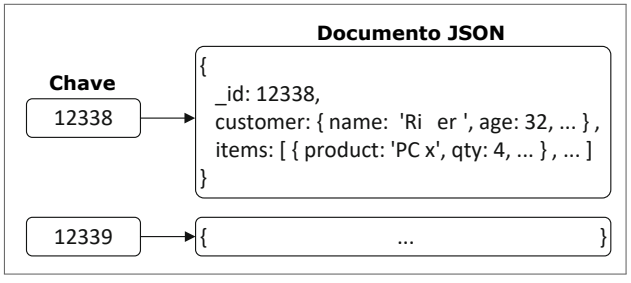


Figura 4 – Representação de um registro no modelo orientado a documento.

Fonte: Adaptado de (GESSERT et al., 2017)

3.1.1.2.4 Modelo orientado a grafos

Como o próprio nome já indica, os sistemas dessa categoria organizam os dados em um modelo de grafo, introduzindo os conceitos de nós e arestas para representar as entidades

e os relacionamentos. As entidades (nós) podem ter um número qualquer de atributos (pares chave-valor), estes são chamados de propriedades. Os relacionamentos entre os nós são representados pelas arestas possuem: direção, nome, nó de partida e nó final.

Assim como os nós, os relacionamentos também podem ter propriedades. Geralmente são utilizados propriedades numéricas como pesos, distâncias ou custos. Bancos de dados baseados em grafos são indicados para casos de uso nos quais os dados estão muito relacionados, pois esses sistema conseguem ótimo desempenho (JAYATHILAKE et al., 2012). A Figura 5 mostra 3 nós que representam as entidades *Pessoa*, *Empresa*, e *Cidade* e contém dados no formato chave-valor. As arestas *trabalhaem* e *localizaem* são os relacionamentos entre as entidades.

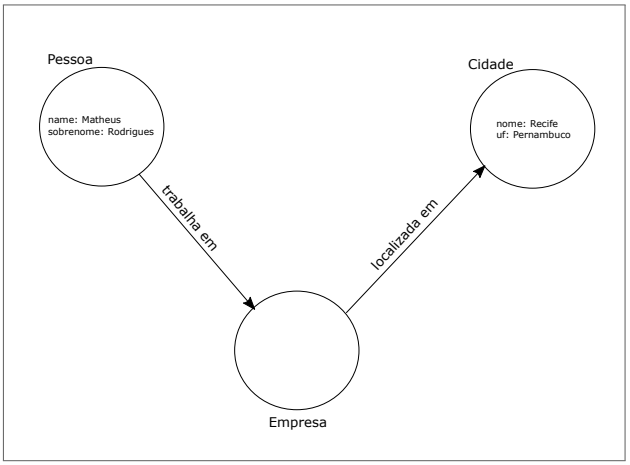


Figura 5 – Representação de um registro no modelo orientado a grafo.

3.1.2 Sistema NoSQL para estudo de caso - MongoDB

Neste trabalho o MongoDB foi escolhido para ser utilizado como subsistema de armazenamento da nuvem privada devido ao fato de ser o de bancos de dados não relacional mais popular segundo o ranking feito pelo website *DB-Engines.com* no ano de 2018 e ser utilizado por grandes empresas como Google, Facebook, Governo norte-americano, Adobe, entre outros. Esse SGBD segue o modelo de dados de documento.

O MongoDB é escalável horizontalmente através de um mecanismo de particionamento (*sharding*) que divide a base de dados em vários servidores com o intuito de dividir a carga de trabalho em sistemas independentes. É dessa forma que o sistema lida com demanda crescente de dados e requisições. A divisão da base de dados ocorre no nível das coleções, e para que uma coleção seja dividida é preciso escolher uma chave de particionamento, que é algum campo chave utilizado como índice e presente em todos os documentos.

Atualmente a disponibilidade dos dados é um fator fundamental para diversos negócios. Para garantir isso, o MongoDB possui um mecanismo de replicação de informação chamado de *replica set*, no qual um grupo de servidores independentes são configurados a

fim de sincronizar seus dados e recuperar o estado do sistema em caso de falhas. MongoDB também suporta o método mestre-escravo, que é considerado ultrapassado.

3.1.2.1 Replica set

Um *replica set* pode ser definido como um grupo de servidores executando o MongoDB e que possuem a mesma base de dados. Em uma configuração padrão de uma implementação do *replica set* um dos servidores é considerado o nó primário e fica responsável por todas as escritas, os outros servidores são chamados de nós secundários ou réplicas, e devem manter-se atualizados conforme as modificações feitas no nó primário. No caso de falha do nó primário, um dos secundários vai automaticamente ser promovido de função (BANKER PETER BAKKUM, 2016). Ainda existe um terceiro tipo de instância possível, e opcional, um tipo de nó chamado de árbitro. Árbitros não guardam dados, eles funcionam para manter o quorum do *replica set* para a realização da votação de eleição do novo nó primário. Essa instância é uma forma barata, computacionalmente falando, de manter um *replica set* com membros suficientes para realizar uma votação de recuperação.

Uma informação não é considerada completamente segura até que esteja escrita na maioria dos membros do *replica set*, ou seja, mais de 50% dos servidores. Por esse motivo a quantidade mínima indicada para criação de um sistema são 3 membros, podendo chegar ao valor máximo de 50 membros (BANKER PETER BAKKUM, 2016). O nós secundários aplicam as mudanças do primário de forma assíncrona através de um arquivo chamado *oplog* que guarda todas as operações que modificaram a base de dados.

Na Figura 6 é mostrado um *replica set* com o nó primário, em destaque, e n nós secundários, as operações de replicação e o "sinal de vida"chamado de *heartbeat* que serve para o reconhecimento entre as máquinas de um mesmo *cluster* de servidores. As setas que representam as replicações dos dados partem do nó primário em direção aos secundários, indicando o sentido do fluxo das operações. Enquanto que o *heartbeat* ocorre em ambas direções, por isso o uso da reta bidirecional.

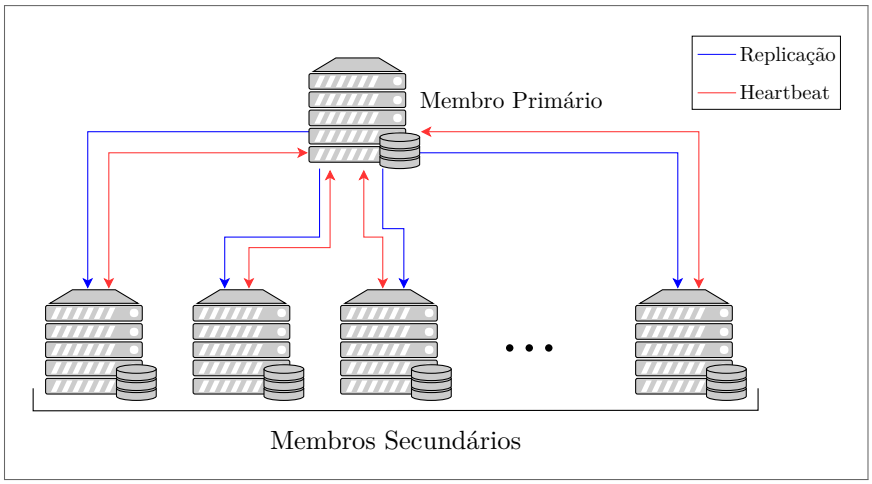


Figura 6 – Representação de um *replica set*.

O mecanismo de recuperação automática utilizado pelo MongoDB entra em ação quando a comunicação com um dos membros do *replica set* não consegue ser estabelecida por um intervalo de tempo predeterminado. É nesse momento que um dos nós secundários convoca uma votação para eleição de um novo nó primário, após o fim do processo, enfim ocorre o reestabelecimento do serviço. A eleição geralmente ocorre em menos de 12 segundos, mas esse valor depende das configurações e latência, caso os servidores estejam localizados em *data centers* geograficamente separados.

Pode-se dizer que o MongoDB utiliza um mecanismo de redundância *warm-standby* no *replica set* pelo fato das réplicas estarem energizadas, mas precisarem de um intervalo de tempo, onde ocorre a votação, para um novo servidor assumir a função de nó primário e o fornecimento de serviço seja retomado por completo (GUIMARãES et al., 2011).

Um dos objetivos deste trabalho é criar um modelo GSPN que represente um *replica set* com três nós. O modelo pretende mostrar o mecanismo de recuperação do sistema de maneira simplificada, pois a implementação do processo de *warm-standby* iria aumentar demais a complexidade do modelo para representar um atraso médio de reestabelecimento menor que 12 segundos, que causa um impacto significante na confiabilidade, mas é pouco efetivo na diminuição da disponibilidade do serviço no ambiente criado.

3.2 AVALIAÇÃO DE DESEMPENHO

A avaliação de desempenho permite tratar problemas comumente encontrados em sistemas computacionais, tais quais identificação de gargalos, caracterização de cargas de trabalho, previsão de desempenho e planejamento de capacidade (JAIN, 1991). As principais técnicas para avaliação de desempenho de sistemas computacionais são: medição de sistemas reais, modelagem analítica e simulação. A escolha da técnica correta depende do tempo e recursos disponíveis para solução do problema, além disso deve ser levado em consideração o nível de detalhes desejado nos resultados.

3.2.1 Medição

A medição de valores só é possível quando o sistema real, ou protótipo, já existe e está operacional. Caso o sistema esteja em fase de concepção, a modelagem analítica ou simulação são as melhores alternativas. De maneira geral, a medição fornece resultados de mais credibilidade pois são baseados nos valores de um sistema real, de maneira semelhante, os resultados de uma modelagem ou simulação transmitem mais confiança quando utilizam valores que foram extraídos de sistemas reais (JAIN, 1991).

Mesmo sendo o método mais confiável, os resultados podem sofrer de grande variabilidade já que alguns parâmetros definidos para coleta de informações podem influenciar nos resultados dos experimentos, pode-se citar como exemplo o tempo de medição (JAIN, 1991). Esta técnica também é a mais custosa, já que necessita de mais tempo quando com-

parada com as duas outras. Ademais, as medições em sistemas reais tornam-se inflexíveis na medida em que fornecem informações específicas de um único sistema medido (LILJA, 2005).

3.2.2 Modelos analíticos

A modelagem analítica descreve matematicamente o comportamento do sistema e pode ser classificada de maneira bastante ampla, em modelos de espaço de estados e modelos sem espaço de estados. Uma das técnicas de modelagem de espaço de estados mais utilizadas são as cadeias de Markov, que consiste em um conjunto de estados e um conjunto de transições rotuladas entre os estados. Um estado da cadeia de Markov pode modelar várias condições de interesse no sistema em estudo, como por exemplo, a quantidade de tarefas esperando por recursos para serem executadas, tarefas que falharam, o número de tarefas simultâneas e assim por diante (LILJA, 2005) (BOLCH et al., 2006).

Quando a técnica de avaliação de desempenho empregada é a modelagem, o primeiro passo a ser dado é criar uma descrição formal do sistema real. Em um segundo momento, deve-se deduzir as métricas de desempenho de acordo com os métodos que serão empregados para resolução do modelo (BOLCH et al., 2006). Quando escolhe-se uma cadeia de Markov para representar o sistema, significa que será criado uma representação formal do espaço de estados. Para conceber um modelo de alto nível podem ser utilizadas técnicas de redes de filas ou redes de Petri. O modelo formal representa o sistema e suas interações com o ambiente no nível conceitual e como resultado o modelo abstrai todos os detalhes que são considerados irrelevantes (BOLCH et al., 2006).

As métricas de desempenho podem ser obtidas através de simulações ou por soluções numéricas/analíticas. Após obter os resultados do modelo, os valores podem ser validados com dados provenientes do sistema real. Essa validação ajuda em possíveis ajustes para que o modelo forneça os resultados desejados. Abaixo são descritas as alternativas para resolução de modelos:

- Soluções analíticas: a principal ideia para soluções analíticas é descrever o comportamento do sistema através de uma equação ou um conjunto de equações das quais é possível calcular medidas do sistema utilizando os algoritmos matemáticos;
  - Soluções numéricas: muitos tipos de equações podem ser derivadas a partir de uma descrição formal do sistema. Quando o sistemas é complexo, geralmente, não é possível representá-lo através de uma única equação;
  - Soluções de fórmulas fechadas: os sistemas podem ser descritos através de um sistema simples de fila ou CTMCs. Uma grande vantagem desse tipo de solução é a complexidade computacional, considerada moderada que permite a resolução rápida para obtenção de medidas de desempenho.

3.2.3 Simulação

A simulação é outra abordagem de avaliação de desempenho feita através de modelagem. Em alguns modelos uma solução analítica não é possível, devido ao sistema representado ser muito grande, e assim, o espaço de estados do modelo vem a ser considerado infinito, ou seja, não é possível analisar todos os estados (ou configurações) que o sistema pode alcançar em um tempo finito (mesmo que muito longo), pois o número de entradas para a análise seria infinitamente grande.

Para o caso do sistema estar indisponível ou em estágio de projeto, um modelo de simulação fornece uma maneira de prever o desempenho ou comparar, mais facilmente, várias configurações. Um modelo de simulação pode ser preferido sobre as medições, pois permite que as alternativas sejam comparadas sob uma ampla variedade de cargas de trabalho e ambientes.

É importante ficar atento que técnicas simulação são muito suscetíveis a erros, principalmente pelo fato de que os profissionais responsáveis por executar a simulação precisam conhecimentos em áreas como estatísticas e desenvolvimento de sistemas (JAIN, 1991). Outros problemas relacionados ao ato de modelar uma simulação podem ser: adotar o nível errado de detalhe, não representar o sistema corretamente, não executar a simulação por tempo suficiente.

Entre a variedade de simulações descritas na literatura, as que seriam de interesse para os cientistas da computação são: Simulação de Monte Carlo, Simulação Orientada por Traços e Simulação de Eventos Discretos (JAIN, 1991).

- Simulação de Monte Carlo é uma simulação que não considera a variação de tempo. Tais simulações são utilizadas para modelar fenômenos probabilísticos que não alteram suas características com o passar do tempo.

- Na simulação de eventos discretos o sistema é modelado identificando seus eventos característicos e, em seguida, escrevendo um conjunto de rotinas que fornece uma descrição detalhada das mudanças de estado ocorridas no momento de cada evento. A simulação evolui executando os eventos em ordem crescente de tempo.

- Simulação Orientada a *Trace* é realizada observando os traços de execução do programa ou acesso ao componente do sistema com o objetivo de predição de desempenho.

3.3 AVALIAÇÃO DE DISPONIBILIDADE

Atualmente, um aspecto relevante para sistemas computacionais é a disponibilidade. Isso fica bastante claro na área de computação na nuvem, onde os provedores de serviço elaboram um contrato, chamado SLA, onde são acordados valores mínimos de fornecimento

do serviço para seus clientes, e onde é comum a definição de valores de disponibilidade para o serviço contratado (ZHAO; SAKR; LIU, 2015).

Disponibilidade pode ser definida como a capacidade de um sistema estar apto para executar sua função em um momento específico ou durante um período de tempo determinado (KUO; ZUO, 2003). Geralmente o valor correspondente à disponibilidade é expresso como a taxa, ou seja, a proporção de tempo que o serviço está realmente disponível para uso pelos clientes dentro das horas de serviço acordadas. A Equação 3.1 mostra como é feito o cálculo da disponibilidade, onde o *Mean Time to Failure* (MTTF) representa o tempo médio para ocorrência de uma falha e *Mean Time to Repair* (MTTR) é o tempo médio para reparo, este valor está relacionado com a política de manutenção e com as características inerentes ao componente que precisa ser reparado. A soma do $MTTF$ e $MTTR$ é o período de tempo total de observação do sistema. A disponibilidade calculada desta forma será um número entre zero e um.

$$D = \frac{MTTF}{MTTF + MTTR} \tag{3.1}$$

O cálculo da confiabilidade, mostrado na Equação 3.2, é a probabilidade de que o sistema irá funcionar corretamente no intervalo de tempo de 0 a t. Ainda nesta equação, a variável aleatória T representa o tempo para falha. A confiabilidade é necessária para o cálculo do MTTF como mostrado na Equação 3.3. O cálculo do MTTR é apresentado na Equação-3.4.

$$R(t) = P(T > t); t \geq 0 \tag{3.2}$$

$$MTTF = \int_0^\infty R(t)dt \tag{3.3}$$

$$MTTR = MTTF \times \frac{ID}{D} \tag{3.4}$$

O elemento ID na Equação 3.4 representa a indisponibilidade do sistema, o cálculo da indisponibilidade é mostrado na Equação 3.5, como uma maneira de facilitar a visualização desse índice, normalmente ele é representado pela quantidade de horas anuais que o sistema fica indisponível. O valor chamado *downtime* anual é calculado pela Equação 3.6, que multiplica o valor de indisponibilidade pelo total de horas de um ano.

$$ID = 1 - D \tag{3.5}$$

$$Downtime = ID \times 8760 \tag{3.6}$$

Um sistema pode ser descrito como um conjunto de componentes realizando uma função específica. Desta forma, um sistema pode ser decomposto em partes que ao falharem,

podem ser repostas ou reparadas. Para a avaliação de confiabilidade é comum se utilizar uma decomposição hierárquica dos componentes, procurando a relação entre a confiabilidade do sistema em termos dos seus componentes (KUO; ZUO, 2003). Diagrama de Blocos de Confiabilidade (RBD) e *Fault Tree* (FT) são técnicas de modelagem combinatoriais comumente adotados na avaliação de confiabilidade, além dessas, CTMC e são modelos de espaço de estados mais utilizados.

Nas seções seguintes serão apresentadas as técnicas de modelagem em PN e RBD, utilizadas neste trabalho para o cálculo de disponibilidade do sistema.

3.4 REDES DE PETRI ESTOCÁSTICAS GENERALIZADAS

Nesta seção, primeiramente, é explicado o conceito de rede de Petri mais básico, e logo após é mostrado a extensão de redes de Petri estocástica generalizada (GSPN).

3.4.1 Redes de Petri

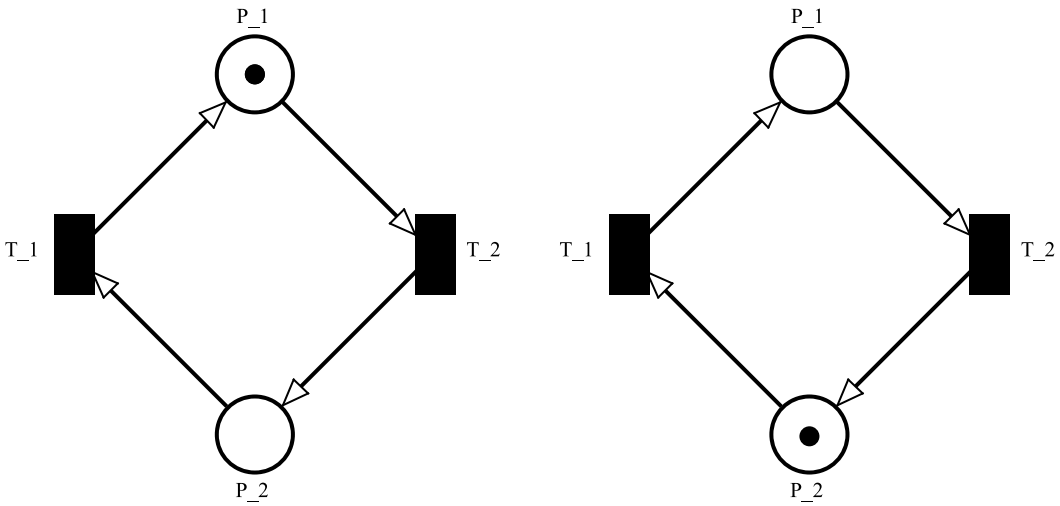
O conceito de redes de Petri foi criado por Carl Adam Petri no ano de 1962, com a sua tese de doutorado "Kommunikation mit Automaten" (Comunicação com Autômatos) na faculdade de Matemática e Física da Universidade Darmstadf na Alemanha (MURATA, 1989). Redes de Petri são ferramentas gráficas e matemáticas muito bem adaptadas para vários tipos de modelagens sistemáticas, descrevendo sistemas com processos concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos ou estocásticos. Em geral, uma rede de Petri é um grafo bipartido dirigido, em que lugares (representados por círculos brancos) denotam os estados locais e transições (descrito como retângulos brancos) representam ações. Arcos (arestas com uma seta na ponta) conectam lugares às transições e vice-versa. Desde o trabalho de Petri, muitas representações e extensões foram propostas, permitindo descrições mais concisas e representando características de sistemas não observadas nos primeiros modelos (MURATA, 1989).

Na Figura 7 estão representados os componentes de uma rede de Petri: lugares (Figura 7(a)), transições (Figura 7(b)), arcos (Figura 7(c)) e marcas (Figura 7(d)). Os lugares são as variáveis de estado e as transições representam ações ou eventos realizados pelo sistema. Existe uma relação entre os lugares e as transições que possibilita ou não a realização de uma determinada ação, que é representada pelo disparo de uma transição. Para que uma transição esteja apta a disparar o lugar que representa a pré-condição de disparo deve possuir uma marca. Após a realização de uma determinada ação, alguns lugares terão suas informações alteradas, ou seja, a ação criará uma pós-condição. Os arcos representam o fluxo das marcas pela rede de Petri, e as marcas representam o estado em que o sistema se encontra em determinado momento (MACIEL; LINS; CUNHA, 1996).

A representação formal de um modelo em rede de Petri é a quíntupla $PN = \{P; T; F; W; M_0\}$ (MURATA, 1989), onde:

Figura 7 – Componentes de uma rede de Petri.

Fonte: (MACIEL; LINS; CUNHA, 1996)



(a) Marcação 1. (b) Marcação 2.

Figura 8 – Exemplo de rede de Petri.

- P é o conjunto finito de lugares;
- T é o conjunto finito de transições, $P \cap T = \emptyset$;
- $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos;
- $W : F \to N - \{0\}$ é a função de atribuição de peso aos arcos;
- $M_0 : P \to N \cup \{0\}$ é a função de marcação inicial, onde $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$.

As propriedades das redes de Petri são divididas em duas categorias: as dependentes de marcação inicial, conhecidas como propriedades comportamentais; e as propriedades não dependentes de marcação, conhecidas como propriedades estruturais (MACIEL; LINS; CUNHA, 1996).

Na Figura 8 é mostrado uma rede de Petri com dois lugares e duas transições. O lugar P_1 com a marca é o estado inicial do modelo (Figura 8a), quando a transição T_2 é disparada simula a ocorrência de uma ação que modifica o estado do modelo e a marca passa para o lugar P_2 representando a mudança de estado (Figura 8b). Quando a transição T_1 é disparada o modelo volta ao estado inicial.

3.4.1.1 Propriedades comportamentais

Em (MACIEL; LINS; CUNHA, 1996) são apontadas um total de 13 propriedades comportamentais, a seguir algumas dessas propriedades serão brevemente explicadas para dar uma noção do funcionamento de uma PN, as características escolhidas foram: alcançabilidade, limitação, segurança, vivacidade e cobertura.

- **Alcançabilidade** (*Reachability*) - indica a possibilidade de uma determinada marcação ser obtida através de um número finito de disparo de transições a partir da marcação inicial. O problema da análise desta propriedade é verificar se uma marcação $M^{'}$, pertencente ao conjunto de possíveis marcações de uma PN é acessível a partir da marcação inicial M_0.

- **Limitação** (*Boundedness*) - um lugar P_1 é dito limitado ou k-limitado, se para toda marcação possível o número de marcas neste lugar é menor ou igual a k. Uma rede é dita k-limitada se todos os seus lugares são limitados e não acumulam mais que k marcas.

- **Segurança** (*Safety*) - é uma propriedade que deriva da propriedade anterior, pois um lugar P_1 é dito seguro, se ele é 1-limitado, ou seja, não acumula mais que uma marca. Da mesma forma, uma PN é segura se for uma rede 1-limitada.

- **Vivacidade** (*Liveness*) - é uma propriedade ligada ao conceito de *deadlock* que é bastante comum na computação. *Deadlock* em uma rede de Petri ocorre quando uma transição é impossibilitada de disparar em uma marcação $M^{'}$. Uma transição T_1 é dita *live* em uma marcação $M^{'}$ se T_1 é potencialmente disparável para todas as marcações do conjunto de marcações de uma rede, ou seja, uma transição é *live* se esta não é passível de *deadlock*. Uma rede é considerada *live* se para toda marcação é sempre possível disparar qualquer transição através de uma sequência de disparos. Um sistema ser livre de *deadlock* não significa que seja live, entretanto um sistema *live* implica em um sistema livre de *deadlocks*.

- **Cobertura** (*Coverability*) - esta propriedade está ligada ao conceito de alcançabilidade e *liveness*. Uma marcação $M^{'}$ é dita coberta quando existe uma marcação $M^{''} \geq M^{'}$, ou seja, o número de marcas de cada lugar da rede em $M^{''}$ é maior ou igual que em $M^{'}$.

3.4.1.2 Propriedades estruturais

As propriedades estruturais são independentes da marcação, aspectos relacionados com a estrutura dos modelos são visualizados através dessas propriedades. Nesta seção, são apresentadas as propriedades: limitação estrutural, conservativa, repetitividade e consistência (MACIEL; LINS; CUNHA, 1996).

- **Limitação estrutural** - afirma que uma rede Petri é limitada se for limitada para qualquer marcação inicial M_0;
- **Conservação** - é uma propriedade que garante que a quantidade de marcas na rede não irá diminuir ou aumentar, ou seja, o número de marcas não varia;
- **Repetitividade** - uma rede é repetitiva se existe uma marcação e uma sequência de transições disparáveis, para esta marcação, todas as transições da rede são disparadas ilimitadamente;
- **Consistência** - a rede possui esta propriedade se ao disparar uma sequência de transições, partindo de uma marcação inicial M_0, retorna-se para a marcação inicial, mas todas as transições da rede devem ser disparadas pelo menos uma vez.

3.4.1.3 Análise das redes de Petri

Os métodos de análise são utilizados para a verificação das propriedades citadas anteriormente. Aqui serão abordados o método baseado na árvore de cobertura e dois métodos baseados na equação fundamental das redes. Posteriormente, essas técnicas serão utilizadas neste trabalho para verificar as propriedades dos modelos criados.

A equação fundamental das redes de Petri torna possível a checagem de diversas propriedades de modelos. Esta equação serve para analisar a possibilidade de acesso as marcações e o número de vezes que cada transição tem que ser disparada para atingir determinada marcação (MACIEL; LINS; CUNHA, 1996). A Equação 3.7 apresenta a equação fundamental das redes de Petri.

$$M'(p) = M_0(p) + C.\bar{s}, \forall p \in P \tag{3.7}$$

O elemento $\bar{s}$ é o vetor característico onde os seus componentes representam o número de vezes que cada transição foi disparada para chegar na marcação $M'(p)$ a partir de $M_0(p)$. A matriz de incidência C da rede.

- **Árvore de Cobertura** - baseia-se na construção de uma estrutura que mostra todas as possíveis marcações de uma rede. Considerando uma estrutura de árvore (conceito computacional), os nós representarão as marcações e as arestas são as transições disparadas. Com a construção da árvore é possível verificar algumas propriedades, como Limitação, Segurança, Vivacidade, entre outros. Nos caso da rede ser limitada a árvore de cobertura é denominada árvore de alcançabilidade (MACIEL; LINS; CUNHA, 1996).
- **Invariantes de Lugar** - fornecem os componentes conservativos da rede sem a necessidade de observação exaustiva da árvore de cobertura. Esses componentes estão relacionados ao conjunto de lugares em que a soma ponderada de marcas

permanece constante mesmo com o disparo de uma sequência de transições (MACIEL; LINS; CUNHA, 1996).

- **Invariantes de Transição** - esta técnica verifica a existência de componentes repetitivos estacionários e a consistência parcial, ou completa, da rede. A rede possui componentes repetitivos são identificados se partir de uma marcação M' dispararmos cada transição n_i vezes e retorna-se a marcação inicial M' (MACIEL; LINS; CUNHA, 1996).

3.4.2 Redes de Petri Estocásticas Generalizadas

Este trabalho utiliza uma extensão específica das redes de Petri, nomeadas, redes de Petri Estocásticas Generalizadas (GSPN) (MARSAN et al., 1991), que permite associar atrasos probabilísticos às transições usando distribuições exponenciais. Seu respectivo espaço de estados é isomorfo às cadeias de Markov de tempo contínuo (BOLCH et al., 2006) (MARSAN et al., 1991). Além disso, a GSPN permite a utilização de análise estacionária (através da conversão da GSPN para cadeias de Markov) e técnicas de simulação para a obtenção de métricas de desempenho e confiabilidade.

As GSPN apresentam algumas características como a representação dinâmica do sistema modelado com certo grau de detalhamento, a descrição gráfica e formal que permite a obtenção de informações sobre o comportamento do sistema modelado através de suas propriedades comportamentais e estruturais, a representação de sincronismo, assincronismo, concorrência, compartilhamento de recursos, entre outros comportamentos.

Como falado anteriormente, as atividades de um sistema são representadas por transições, nas GSPN as transições que possuem tempos associados são chamadas de transições temporizadas e são identificadas por retângulos brancos como mostrado na Figura 9(a). Nas GSPNs o tempo de habilitação da transição equivale ao tempo para realizar a atividade, e o disparo representa o fim da atividade. As transições imediatas, Figura 9(c), não possuem tempo associado e disparam imediatamente quando habilitadas, essas transições têm prioridade de disparo maior que as transições temporizadas. Podendo também possuir prioridades e probabilidades de disparo entre transições imediatas diferentes. Os arcos inibidores, Figura 9(b), permitem impedir o disparo de uma transição se estiver partindo de um lugar que possua uma ou mais marcas (MARSAN et al., 1991).

Em Marsan et al. (1991) é elaborada uma definição formal, que é apresentada a seguir: Uma GSPN é definida pela 8-tupla $GSPN = (P, T, I, O, H, \Pi, M_0, W)$, onde:

- o conjunto de elementos (P, T, I, O, M_0) representam a rede de Petri,
- $H \subset P \times T$ é o conjunto que contêm os arcos inibidores,
- Π é o vetor que contêm as atribuições de prioridades para as transições temporizadas e imediatas,

- W é o vetor que associa uma condição de guarda relacionada a marcação do lugar à cada transição.

Além disso, transições temporizadas podem ter diferentes graus de concorrência através da semântica *single-server* (ss) ou semântica *infinite-server* (is). No geral, is é adotado para representar disparos paralelos, considerando que a taxa de disparos é aumentada linearmente de acordo com o grau de habilitação da transição. No ss, a taxa de disparo é mantida constante.

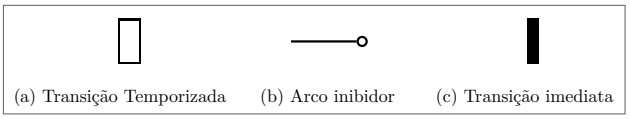


Figura 9 – Componentes de uma rede de Petri estocástica generalizada.

A Figura 10 mostra um exemplo de GSPN onde a transição *Gerador* cria as marcas que correspondem às requisições de um serviço. Cada marca criada é colocada no lugar *Gerado* e, a partir daí, uma escolha é feita. As marcas podem ser enviadas para o $Buffer$, caso haja recursos livres, informada no lugar *Livres*, para o processamento delas. Caso não possam ser processadas a marca pode ser descartada pela transição imediata *Perdidos*, quando todos as n marcas do lugar *Livre* estiverem ocupados. A transição *Perdidos* só estará ativada quando não houver mais marcas em *Livres*, devido a presença do arco inibidor. E a transição *Entra* só estará habilitada se houver uma ou mais marcas em *Livres*, devido ao arco de entrada. A transição *Processa* representa o processamento das requisições pelo servidor.

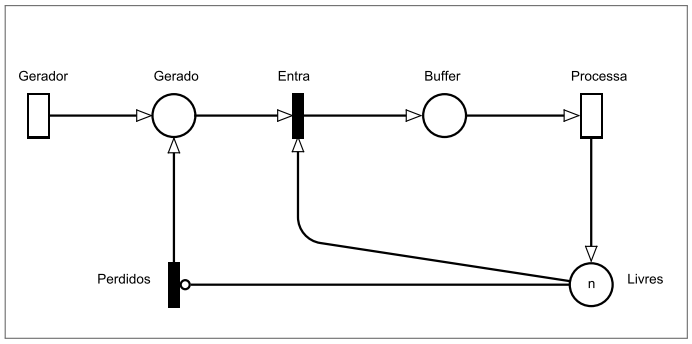


Figura 10 – Exemplo de rede de Petri estocástica generalizada.

Fonte: (GERMAN, 1996)

3.5 DIAGRAMAS DE BLOCO DE CONFIABILIDADE

Diagrama de bloco de confiabilidade (RBD) foi uma técnica pioneira para modelagem de aspectos referentes a confiabilidade de sistemas. Em um modelo RBD cada componente do sistema é representado como um bloco e para representação de um sistema completo

os blocos são combinados com outros. Os blocos podem ser combinados de duas maneiras, em série ou paralelo (TRIVEDI et al., 1996). Dessa forma, modelos RBDs constroem uma visualização gráfica de alto nível representando as interações lógicas entre elementos do sistema. Este tipo de modelagem combinatorial é utilizada para avaliação e o cálculo de métricas como disponibilidade confiabilidade, manutenabilidade, etc.

As medidas são calculadas de acordo com o tipo de interação entre os blocos, essas interações definem a funcionalidade do sistema. Um diagrama que tem blocos conectados em série exige que cada elemento esteja funcionando para que o sistema seja operacional. Os blocos em um arranjo paralelo representam componentes que possuem redundância, ou seja, basta que apenas um esteja operacional para que o sistema funcione normalmente (TRIVEDI et al., 1996). Ainda é possível a configuração de blocos *k-out-of-n*, que exigem o funcionamento de k de n componentes para que o sistema esteja funcional.

A Figura 11 (a) e (b) ilustram, respectivamente, a organização dos blocos em série e em paralelo. A disponibilidade estacionária de um conjunto de n componentes independentes em série é determinada pela Equação 3.8, onde Di é a disponibilidade estacionária do i-ésimo bloco.

$$D_{Série} = \prod_{i=1}^{n} Di \tag{3.8}$$

O cálculo da disponibilidade de uma estrutura em paralelo de n componentes independentes, é mostrada na Equação 3.9.

$$D_{Paralelo} = 1 - \prod_{i=1}^{n} (1 - Di) \tag{3.9}$$

Fazendo aplicações sucessivas destas fórmulas, podemos encontrar a disponibilidade estacionária para um RBD mais complexo formado por diferentes combinações das duas formas de composição de blocos.

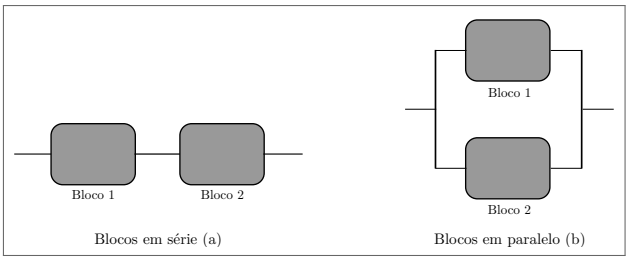


Figura 11 – Organização de blocos em RBD.

4 MÉTODO PROPOSTO E MODELAGEM

Neste capítulo é apresentada a metodologia empregada no desenvolvimento do estudo. Para alcançarmos o objetivo final do trabalho é necessário passar por outras etapas essenciais, como a execução de experimentos para a coleta de informação do sistema, desenvolver modelos, validar modelos e realizar experimentos utilizando os modelos. Também serão descritos o ambiente de nuvem privada, o gerador de carga, o *framework* utilizado para a criação da nuvem, a construção de modelos em RBD e por fim a concepção dos modelos GSPN para a avaliação de disponibilidade e vazão.

4.1 METODOLOGIA

Nesta seção são apresentadas as etapas nas quais o trabalho foi realizado. Na Figura 12 está o fluxo do processo em 6 etapas. A metodologia se inicia na escolha e estudo do sistema NoSQL que irá ser utilizado como subsistema de armazenamento, só então, o ambiente de experimentos é construído e acontece a medição de valores de interesse no sistema real. Por fim, vêm a etapa de modelagem, primeiramente com a criação de modelos RBD para estimar tempos médios de falha e reparo dos componentes da nuvem e depois na concepção de modelos GSPN para avaliação conjunta de disponibilidade e desempenho.

1. **Estudo e implantação do sistema** - Para a escolha do subsistema de armazenamento foi considerado o fato de facilidade de implantação, pois uma característica dos SGBDs NoSQL é fornecer interface de gerenciamento simplificadas, muitas vezes utilizando a arquitetura RESTful aceitando requisições em *Hypertext Transfer Protocol* (HTTP). Além disso, também foi observado como o sistema de armazenamento realiza a replicação dos dados para outros servidores a fim de garantir maior disponibilidade. Com a nuvem privada criada, uma VM é instanciada e o SGBD é instalado ficando apto para receber requisições externas. Foi utilizada uma máquina física externa que age como gerador de carga externa com a ferramenta YCSB.

2. **Estimação de parâmetros de disponibilidade - Modelagem RBD** - Para que fosse possível saber o tempo médio de falha e reparo dos nós que compõe a nuvem foram utilizados modelos RBD. Nos modelos foram considerados os componentes de hardware e software que compõem cada nó.

3. **Criação de modelos para avaliação de sistemas - Modelagem GSPN** - Durante esta fase foram criados dois modelos de GSPN para representar dois sistemas: com um servidor e com três servidores. Para avaliar o sistema foram escolhidas duas métricas relevantes na qualidade de serviço de ambientes de nuvem e banco de dados: vazão (taxa de transferência) e disponibilidade.

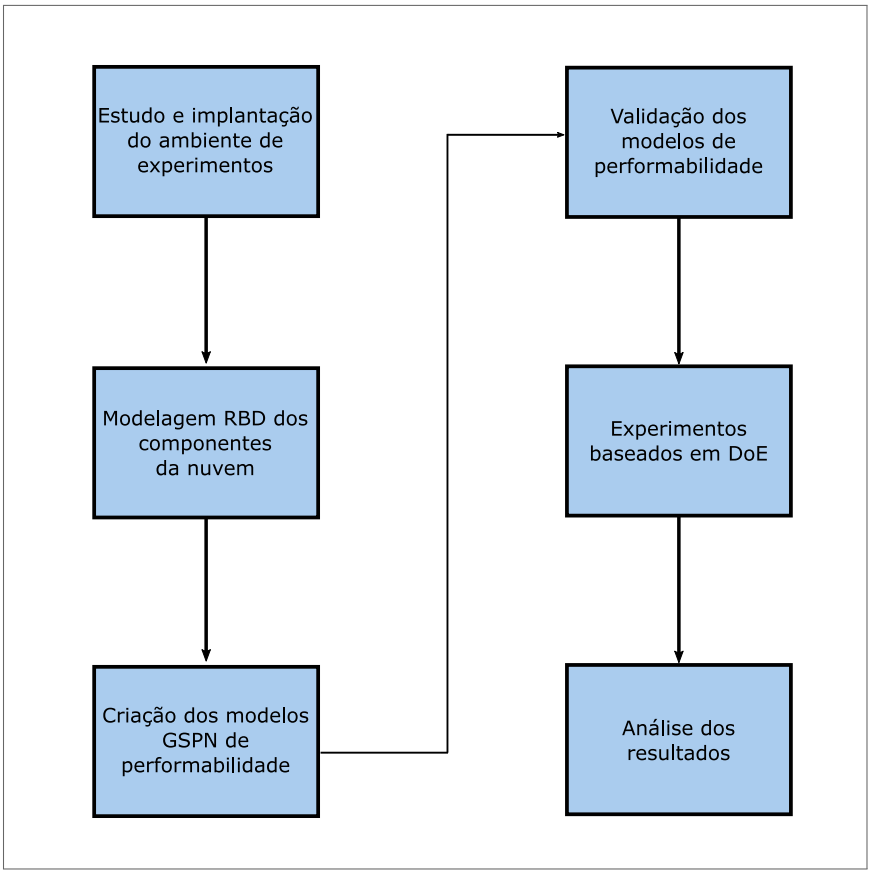


Figura 12 – Metodologia do trabalho.

Vazão - é o número de operações efetuadas pelo banco de dados sobre uma unidade de tempo. As unidades no trabalho são operações por segundo.

Disponibilidade - é o valor que indica o quanto o sistema está operacional, geralmente expresso em porcentagem ou considerando a indisponibilidade em horas, chamado de *downtime*.

4. **Validação do modelo** - Para validar o modelo base, que representa o sistema com um servidor, adotamos o ambiente de nuvem privada, citado anteriormente, e uma máquina externa é utilizada para simular solicitações de clientes, que são geradas usando o YCSB (COOPER, 2018). A carga de trabalho é composta por 50% de operações de leitura e 50% de operações de atualização. Para estimar o atraso de processamento e o intervalo de tempo entre os envios das requisições foram obtidas 30 amostras do funcionamento do MongoDB realizando 100.000 operações e usando diferentes números de *threads* simultâneas (que imitam usuários). Considerando esses valores, diferentes configurações do sistema foram simuladas pela GSPN, a vazão (operações por segundo - ops/s) foi estimada e, em seguida, comparada ao sistema real.

5. **Experimentos baseados em DoE** - Três experimentos são realizados para avaliar a vazão e a disponibilidade do sistema, e ambos são baseados no DoE com um

planejamento fatorial l^k, onde k é o número de fatores e l é o número de níveis. Para cada combinação de níveis de fatores, a saber, tratamento, um modelo GSPN foi criado e avaliado usando análise estacionária.

6. **Análise de resultados** - Os resultados dos experimentos foram, organizados separadamente por experimento em rankings que facilitam a visualização dos componentes mais importantes para a métrica analisada. O ranqueamento ocorre em função dos efeitos que cada componente causa no resultado do experimento.

4.2 ARQUITETURA BÁSICA

Os SGBDs NoSQL são adaptados para o ambiente de computação na nuvem, esses sistemas geralmente possuem mecanismos nativos de replicação e particionamento de dados, mostrando uma preocupação em garantir disponibilidade. Como esses SGBDs estão se consolidando entre muitas empresas como a preferência para sistemas de armazenamento, é comum que provedoras de serviço de computação na nuvem ofereçam opções de SGBDs não-relacionais no seu catálogo de DBaaS (DEKA, 2014). Para aproximar o estudo presente ao cenário em que bancos de dados são oferecidos em serviços de nuvem foi criado um ambiente de nuvem privada para realizarmos os experimentos de carga no sistema NoSQL.

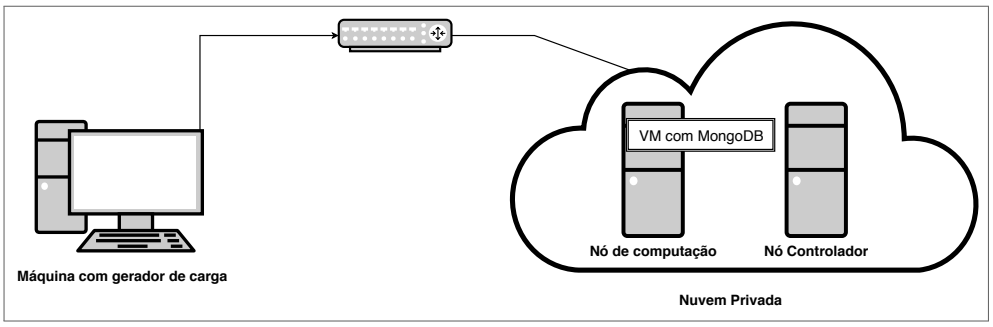


Figura 13 – Arquitetura básica.

A nuvem privada foi criada utilizando o *framework* Openstack (OPENSTACK, 2018), o ambiente é composto por um nó controlador e um nó de computação. O controlador é responsável por coordenar a nuvem hospedando serviços de gerenciamento de rede, imagens, credenciais, entre outros. Por sua vez, o nó de computação executa o *hypervisor* que gerencia instâncias de máquinas virtuais (VMs). O banco de dados MongoDB foi instalado em uma VM inicializada no nó de computação. Para simular o papel de usuários foi utilizado a ferramenta YCSB. Desta forma o ambiente de medição ficou como mostra a Figura 13, com uma uma máquina física dedicada para enviar requisições ao banco de dados e duas outras máquinas físicas pertencentes a nuvem privada, um controlador e uma nó de computação.

Tabela 3 – Especificações das máquinas físicas e VM utilizadas.

Componente	**Nó controlador**	**Nó de computação**	**VM**	**Gerador de carga**
Processador	Intel i3-2100	Intel i7-3770	1 core	Intel i5-4200U
Memória RAM	8GB DDR3	8GB DDR3	512MB	8GB DDR3
Disco Rígido	350GB	250GB	10GB	SSD 250GB
SO	Centos 7.0	Centos 7.0	Ubuntu 16.04	Centos 7.0

A Tabela 3 apresenta as configurações das máquinas físicas e da máquina virtual utilizada neste trabalho. Uma das vantagens dos sistemas NoSQL é funcionar nas chamadas máquinas *commodities*, que são máquinas com configurações simplórias. Isso justifica o fato da VM onde o SGBD está instalado possuir configurações modestas. No nó de computação foi utilizado HD ao invés de SSD devido a limitações do hardware disponível, mas vale atentar ao fato de discos rígidos tradicionais serem opções mais baratas e com bom custo benefícios se tornando a principal opção em serviços de nuvem. A geração de carga foi feito a partir de um computador pessoal, por isso a presença de SSD.

4.3 YAHOO! CLOUD SERVICE BENCHMARK (YCSB)

O YCSB é uma ferramenta que foi desenvolvida para facilitar a vida de desenvolvedores que queriam avaliar e comparar diferentes sistemas de armazenamento baseados na nuvem (COOPER et al., 2010). A grande variedade nos modelos de dados de cada SGBD foi uma das principais motivações para a criação desse software de *benchmark*. Além do modelo de dados, outros fatores como particionamento dos dados, replicação e nível de consistência afetam no desempenho e levavam desenvolvedores a utilizarem cargas ideais para avaliar os sistemas de interesses. O objetivo do YCSB é ser uma ferramenta padrão capaz de auxiliar na avaliação de diferentes sistemas gerando tráfego de diversos tipos que cobrem elementos chave para o desempenho (COOPER et al., 2010).

A ferramenta é muito comum em trabalhos que avaliam aspectos de desempenho em SGBDs NoSQL (CATTELL, 2011) (TANG; FAN, 2016) (ABRAMOVA; BERNARDINO, 2013), pois traz opções interessantes para experimentos de carga em banco de dados. Por exemplo, é possível configurar a quantidade de usuários simultâneos conectados ao sistema, a quantidade de operações feitas por segundos, os intervalos de tempo entre o envio de operações, quantidade de operações a serem enviadas e o tipo de operações à serem realizadas.

Nos experimentos para coletar dados que serão utilizados na validação do modelo GSPN foi utilizada uma carga com operações mistas, sendo, 50% de leitura e 50% de atualização. O número de operações foi fixado em 100.000 e a quantidade de *threads*, que simulam usuários simultâneos, foi variada entre 1, 5, 10 e 20. O intervalo entre o envio das requisições foi definido com a utilização da distribuição $zipfian$, uma versão personalizada pelo Yahoo! do algoritmo $zipfian$ original. A modificação do algoritmo visa

fazer com que o comportamento de envio das operações seja mais próximo de cenários reais, fazendo com que alguns dados sejam mais acessados que outros, e depois de um tempo, essa frequência de acesso passe para outros dados (COOPER et al., 2010).

4.4 OPENSTACK

O OpenStack é um *software* de computação em nuvem de código aberto utilizado para a criação de diferentes tipos de nuvem que proveem três modelos de fornecimento de serviços: Infraestrutura como Serviço (IaaS), Software como Serviço (SaaS) e Plataforma como Serviço (PaaS). Existem vários serviços funcionando em conjunto no Openstack, cada um deles fornece funcionalidades diferentes como armazenamento, rede e processamento que os usuários gerenciam por meio de um painel que pode ser acessado como um *website*. A utilização mais comum do Openstack é para criação de nuvens privadas (GAIKWAD et al., 2017). Uma breve descrição dos principais componentes do Openstack é feita a seguir:

- Keystone - provê uma *Application Program Interface* (API) de autenticação e gerenciamento de autorização de usuários através da *Identity API*. Esta interface de autorização é compatível vários serviços de autenticação, como LDAP, OAuth, OpenID Connect, SAML and SQL.
- Nova - torna possível a implementação de serviços escaláveis sob demanda para computação em máquinas físicas, máquinas virtuais e contêineres.
- Neutron - é um projeto de rede *Software Defined Networking* (SDN) com o propósito de fornecer *Network as a Service* (NaaS) em ambientes virtualizados.
- Glance - possui a função de descobrir, registrar e obter imagens de máquinas virtuais. Possui uma API RESTful que permite recuperar informações de uma imagem instalada em uma VM, assim como a própria imagem. Através do Glance, as imagens das VMs podem ser armazenadas de diversas maneiras.
- Swift - é um SGBD NoSQL de documentos ideal para armazenar dados não estruturados. Altamente disponível, distribuído e eventualmente consistente.
- Cinder - é o serviço de armazenamento de blocos e também é implementado de maneira que possibilita oferecer um serviço de DBaaS altamente disponível, tolerante a falhas de maneira transparente ao usuário.
- NTP - é um protocolo responsável pela sincronização temporal (hora e data) entre os nós de um *cluster*. No Openstack é recomendado utilizar o controlador como referência para os outros nós.
- RabbitMQ - é o serviço padrão de troca de mensagem do Openstack, e que utiliza o protocolo AMQP. O RabbitMQ tem a função de realizar a comunicação entre os

serviços e além dele podem ser utilizados outros software para a troca de mensagens como Qpid ou ZeroMQ.

- *Hypervisor* - o KVM é um *hypervisor* de software livre capaz de gerenciar máquinas virtuais. É também a solução mais adaptada e usada para computação no OpenStack. Assim, em poucas palavras, os serviços e APIs do Nova se comunicam com o *hypervisor* KVM para iniciar, parar e criar instâncias de máquinas virtuais em sua nuvem.

- Horizon - é a implementação de uma *dashboard* do OpenStack, que fornece uma interface no formato de um website para usuários gerenciar serviços como: Nova, Swift, Keystone, etc.

A criação da nuvem foi feita através de uma instalação chamada de *all-in-one*, que configura um ambiente do Openstack *stand-alone*. Este tipo de instalação utiliza apenas uma interface de rede e os serviços essenciais para o funcionamento da nuvem podem ser instalados em uma única máquina física.

4.5 PROJETO DE EXPERIMENTOS - (DoE)

Projetar experimentos (*design of experiments*) possui o objetivo de obter o máximo de informações com o mínimo de experimentos, economizando recursos, tempo e esforço gasto neste processo (JAIN, 1991). Alguns dos principais termos utilizados no DoE são descritos a seguir:

- Fator - são as variáveis que afetam a variável resposta e possuem diversos valores;

- Nível - são os valores que os fatores podem assumir;

- Variável resposta - é a saída do experimento, pode ser uma medida de interesse.

- Interação - é um fator que ocorre quando o efeito de um fator A depende do nível de um fator B.

O DoE fatorial é a melhor alternativa para estudos que pretendem avaliar o efeito de diferentes fatores independentes no resultado. O termo fatorial indica que todas as combinações de fatores e níveis serão investigadas. Para uma análise mais precisa, é ideal que os efeitos principais de cada fator sejam considerados isoladamente, assim como suas interações, para que afirmações mais significantes possam ser feitas para cada uma das configurações do sistema (JAIN, 1991).

O efeito principal de um fator é medido pela mudança dos valores da variável resposta quando os níveis são modificados. E as interações ocorrem quando o impacto de um fator na resposta depende das configurações de outro fator. A Figura 14 mostra o gráfico de

interação para um experimento com dois fatores (A e B) e cada um com dois níveis (- e +), onde a Figura 14(a) mostra os efeitos quando não há interações entre os fatores, já que as retas seguem paralelas. Na Figura 14(b) ocorre interação, pois as retas dos efeitos se cruzam.

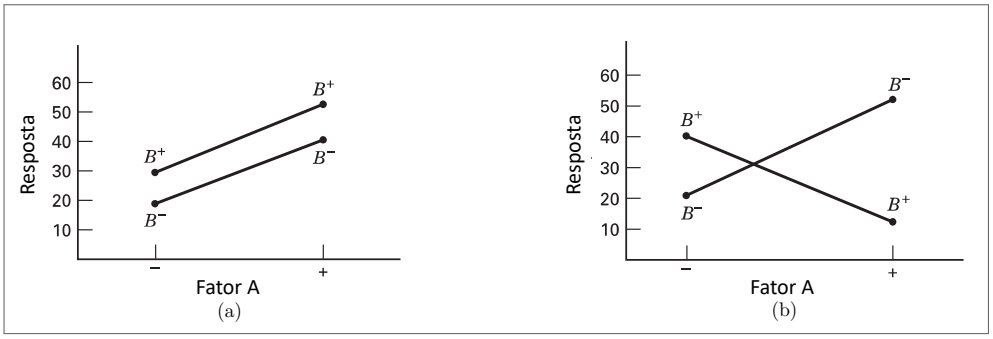


Figura 14 – Exemplo de gráfico de interação.

Fonte: Adaptado de (MONTGOMERY, 2008)

Neste trabalho foi utilizado o *design* 2^k fatorial que é útil para o estágio inicial de projetos, já que esse é um momento onde muitos fatores devem ser investigados e esse *design* permite que k fatores sejam observados a partir de um número pequeno de execuções. O modelo estatístico de um *design* 2^k pode incluir k efeitos principais e k efeitos de interações (MONTGOMERY, 2008). Realizar apenas uma execução para cada configuração do experimentos não nos dá a certeza de que o erro experimental é pequeno. Dessa forma, é uma boa prática na análise estatística dos resultados adotar um grande espaçamento entre os valores dos níveis e não levar em consideração os efeitos de interações de ordem mais alta (MONTGOMERY, 2008).

Para analisar os resultados dos modelos deste trabalho será construído um ranking com os efeitos principais e de interações, até segundo grau, para todas as combinações de fatores e níveis adotados nos experimentos. Esse ranqueamento ordena os resultados pelos valores em módulo, ou valores absolutos, das métricas de interesse (vazão e disponibilidade) calculadas do maior efeito para o menor. Dessa forma, vai ser possível averiguar qual o componente tem mais impacto na variável resposta, assim como verificar a interação entre fatores relevantes para métricas diferentes.

Em resumo, utilizar o design fatorial possui como pontos positivos o fato de ser mais eficiente do que experimentos que analisam um fator por vez e fornece uma análise das interações entre fatores permitindo observar como uma determinada combinação pode afetar no resultado (MONTGOMERY, 2008).

4.6 MODELOS DE DISPONIBILIDADE PARA COMPONENTES DA NUVEM

Esta seção apresenta os modelos RBDs dos nós da nuvem considerando os componentes que formam cada máquina física, o objetivo desta etapa é estimar o MTTF e MTTR

Tabela 4 – Parâmetros dos modelos RBD.

Componente	**MTTF** (h)	**MTTR** (h)
Hardware	8760	1.6666
SO	2893	0.25
Banco de dados	1440	0.3333
Módulos do Openstack	788.4	1
Hypervisor	2990	1
Máquina Virtual	2880	0.0183
MongoDB	788.4	1

para que esses valores sejam utilizados nos modelos de rede de Petri estocástica. Dessa forma, alguns dos componentes representados por blocos para modelar os servidores são: hardware, sistema operacional, *hypervisor*, máquina virtual, módulos do Openstack e o banco de dados. Foram gerados dois modelos RBDs, um para o nó controlador e outro para o nó de computação.

O modelo na Figura 15 é utilizado para representar o nó controlador, responsável por coordenar os demais nós da nuvem privada. O funcionamento desse computador é fundamental para o provimento de serviço. No modelo são avaliados os tempos de falha e reparo dos seguintes componentes: hardware, sistema operacional (SO), NTP, RabbitMQ, Swift, Keystone, Neutron, Nova e Glance.

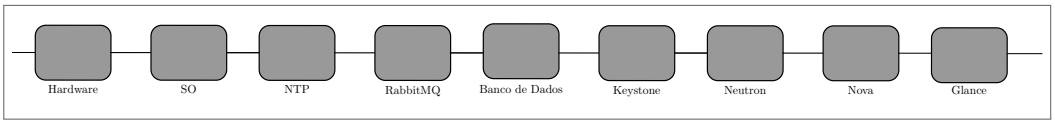


Figura 15 – Modelo RBD do nó controlador.

Na Figura 16 está o outro modelo com os componentes do nó de computação. Para representar esse computador utilizamos: hardware, sistema operacional (SO), Neutron, Nova, *Hypervisor*, Máquina Virtual e MongoDB.

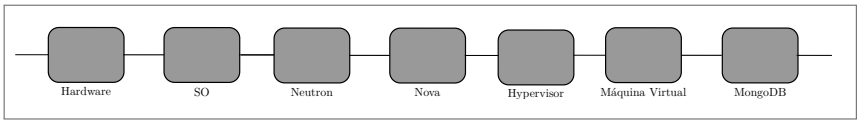


Figura 16 – Modelo RBD do nó de computação da nuvem.

Como parâmetro de entrada para os modelos das Figuras 15 e 16 são utilizados os valores contidos na Tabela 4. Os tempos dos componentes do Openstack foram retirados do trabalho de Melo et al. (2018).

As Equações 4.1 e 4.2 representa o cálculo da disponibilidade pelos os modelos das Figuras 15 e 16, respectivamente. Como os blocos estão todos dispostos em série para

obter a disponibilidade do sistema é necessário multiplicar a disponibilidade de todos os componentes.

$$D_{Cont.} = D_{HW} \times D_{SO} \times D_{NTP} \times D_{RabbitMQ} \times D_{Swift} \times D_{Keystone} \times D_{Neutron} \times D_{Nova} \times D_{Glance} \tag{4.1}$$

$$D_{Comp.} = D_{HW} \times D_{SO} \times D_{Neutron} \times D_{Nova} \times D_{Hypervisor} \times D_{VM} \times D_{MongoDB} \tag{4.2}$$

Esta técnica de modelagem combinatorial representa o sistema com um alto nível de abstração o que facilita para a visualização de dependência entre componentes ou serviços envolvidos na modelagem. Em projetos de modelagem de sistemas complexos os modelos podem ser divididos de maneira hierárquica, partindo da modelagem de subsistemas até culminarem na representação do sistema por completo. Com o aumento do grau de abstração, o sistema criado neste estudo pode ser representado pela Figura 17. A fim de fornecer outro exemplo de modelo com esta técnica de modelagem, a Figura 18 representa um sistema redundante com 2 controladores e 3 nós de computação. Os tempos médios de falha e reparo calculados anteriormente serão utilizados nos modelos de rede de Petri estocástica.

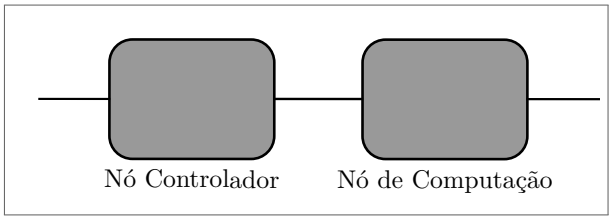


Figura 17 – Modelo RBD da nuvem privada.

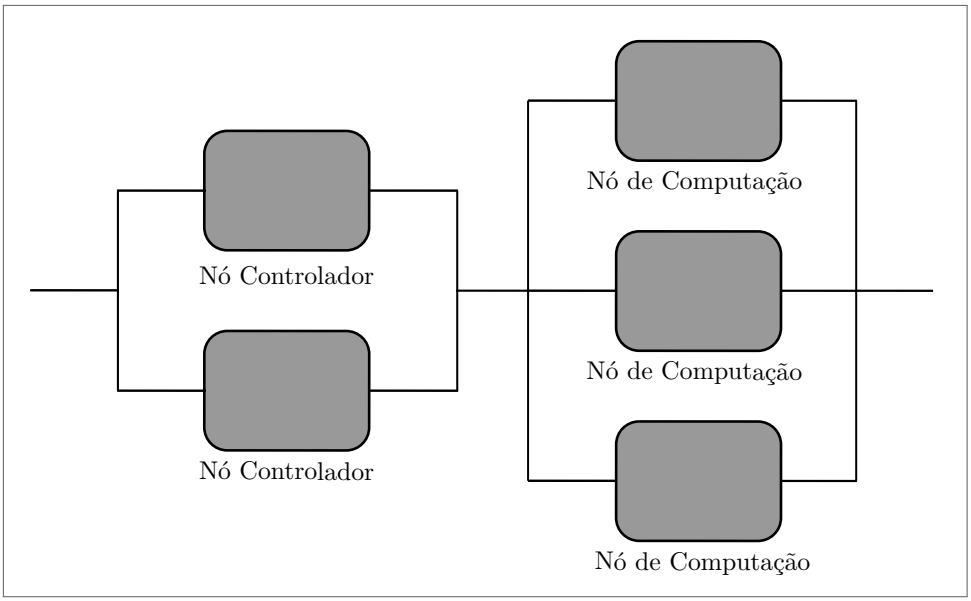


Figura 18 – Modelo RBD da nuvem com redundância.

4.7 MODELOS DE DESEMPENHO E DISPONIBILIDADE PARA O SISTEMA DE NUVEM

Essa seção apresenta os modelos GSPN criados para representar o sistema de nuvem privada com o subsistema de armazenamento NoSQL. O modelo é uma abstração de um sistema real que possui a capacidade de representar algumas de suas características. Realizar experimentos em sistemas pode ser muito trabalhoso e custoso, pois podem envolver custos com recurso, ocupação de espaço físico, gasto de tempo para montagem, configuração e execução de experimentos. A utilização de modelos torna mais simples a realização de experimentos, o teste de sistemas com diferentes possibilidades de configurações, além de reduzir o tempo para a obtenção de resultados.

Neste trabalho, foram construídos modelos capazes de estimar a vazão do sistema sob diferentes cargas de trabalho, disponibilidade e *downtime*. A ferramenta TimeNET (ZIMMERMANN et al., 2000), utilizada para desenvolver os modelos GSPN, é capaz de criar e analisar modelos de rede de Petri estocástica. A avaliação dessas redes de Petri podem ser feitas através de métodos numéricos de análise transiente e estacionária e por simulação, para os casos onde o espaço de estados é muito grande. Esta ferramenta calcula os resultados das avaliações por meio da criação de métricas, que são representadas por expressões matemáticas com uma sintaxe particular do *software*.

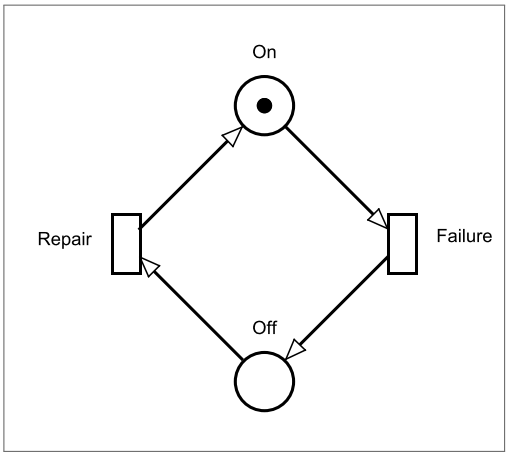
Para facilitar a compreensão do modelo final, a sua construção foi dividida em blocos menores. Primeiramente, serão descritos os blocos que formam o modelo do sistema com um único servidor (*Single Server*). Em seguida, será mostrado o modelo que tem como objetivo a representação de um *Replica Set*, a estrutura de redundância do MongoDB, com três nós (um primário e dois secundários).

Todas as transições adotam distribuição exponencial e semântica de disparo *single-server*, exceto quando indicado. Além disso, serão adotados os seguintes operadores:

$P\{exp\}$ estima a probabilidade da expressão (exp); $E\{exp\}$ representa o valor médio da expressão (exp); e $\#p$ representa o número de marcas no lugar p.

4.7.1 Bloco Componente Simples

A Figura 19 mostra o bloco que será chamado de componente simples (SILVA; AL., 2013), o qual é comumente adotado para representar o estado operacional de um sistema. O lugar On representa o estado onde o sistema está funcionando como o esperado e o lugar Off representa o estado de falha. As transições $failure$ e $repair$ representam, respectivamente, a falha e a manutenção de um componente. A transição $failure$ utiliza como parâmetro o tempo médio para falha (MTTF), enquanto a transição $repair$ adota o tempo médio para reparo (MTTR).



Fonte: Adaptado de (SILVA; AL., 2013)

Figura 19 – Modelo GSPN do componente simples.

O MTTF e MTTR consideram o tempo de falha e de recuperação de todos os componentes do servidor, este valores podem ser obtidos com os modelos RBD. Para este trabalho, os valores médios para os servidores são baseados nos modelos apresentados na seção anterior.

4.7.2 Bloco *Workload*

A Figura 20 mostra o submodelo $Workload$ que tem a intenção de representar o envio de operações ao sistema de armazenamento. A marca atribuída ao lugar $ClientApplication$ representa uma operação pronta para ser enviada, o intervalo entre envios é controlado pelo valor colocado na transição temporizada $WorkloadForwarding$ que possui a semântica de concorrência *single-server*. O lugar $ConnectServer$ representa o estado do sistema quando o lado cliente estabelece comunicação com o servidor. A transição imediata $ForwardRequest$ é a ação do encaminhamento da requisição, o atraso de rede não foi considerado já que a validação do modelo será feita em um ambiente local.

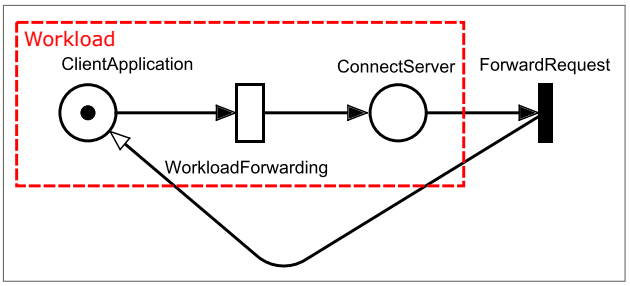


Figura 20 – Bloco GSPN - *Workload*.

4.7.3 Bloco *Node Response*

O bloco *Node Response* simula a chegada da requisição ao servidor e o processamento/armazenamento da operação pelo SGBD. O lugar *ReceivingRequest* é o estado onde a requisição chega ao sistema, a transição temporizada *ProcessingRequest* simula a ação de processamento e possui os atrasos que representam o tempo que uma requisição leva para ser processada. A transição imediata *SendingResponse* são responsável pelo envio da resposta ao usuário. As marcas *connections* no lugar *ConnectionPool* representam a quantidade de conexões simultâneas que podem ser estabelecidas com o SGBD.

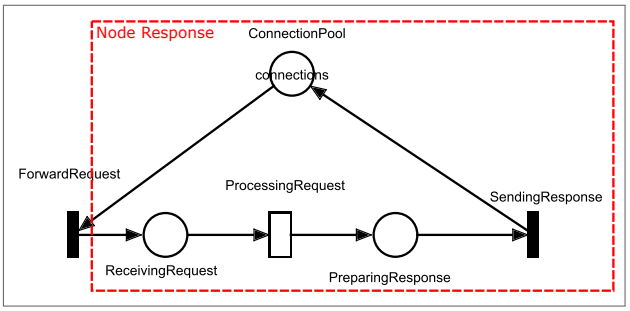


Figura 21 – Bloco GSPN - *Node Response*.

4.7.4 Blocos *Node Availability* e *Controller*

As Figuras 22 e 23 são baseadas no componente simples mostrado anteriormente e são reponsáveis por representar a disponibilidade, respectivamente, do nó de computação e do controlador. As transições temporizadas $NodeFailure$ e $ControllerFailure$ possuem os correspondentes aos MTTF, enquanto as transições $NodeRepair$ e $ControllerRepair$ possuem o valor do MTTR. Os lugares que possuem a terminação On representam o estado onde os sistemas estão funcionando normalmente, e os lugares com terminação Off os estados de falha ou manutenção.

4.7.5 Modelo *Single Server*

A Figura 24 mostra o modelo desenvolvido para um sistema de armazenamento que conta com um servidor. O modelo é dividido em quatro blocos: $Workload$, *Node Response*, *Node*

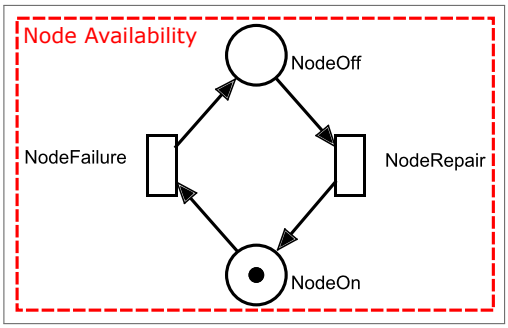


Figura 22 – Bloco GSPN - *Node Availability*.

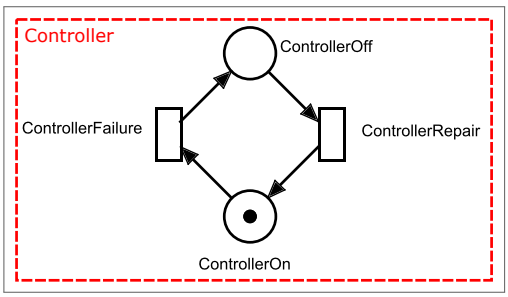


Figura 23 – Bloco GSPN - *Controller*.

Availability e *Controller*.

Como descrito anteriormente o bloco $Workload$ exerce a função de representar o envio de requisições contendo as operações para o SGBD, a transição temporizada *Workload-Forwarding* possui o intervalo de tempo entre o envio de requisições. O bloco *Node Availability* é um componente simples que expressa o estado operacional do nó de computação. Então, o servidor só aceitará requisições se houver uma marcação no lugar $NodeOn$ e $ControllerOn$. Um arco inibidor garante que a transição não ficará habilitada em caso de falha de ($NodeOff$ ter uma marca) e o estado do controlador é verificado através de uma função de habilitação colocada na transição $ForwardRequest$ com a seguinte expressão: $P\{\#ControllerOn > 0\}$.

Assumindo que o nó está operacional, o sistema pode aceitar novas requisições caso possua conexões disponíveis no banco de dados. A quantidade conexões é definida através de um parâmetro de configurações do MongoDB e é representado pela quantidade de marcações *connections* no lugar $ConnectionPool$. O tempo que leva para processar uma requisição e enviar a resposta ao cliente é representado pela transição $ProcessingRequest$ que adota uma semântica de disparo *infinite-server*. Depois de processar uma requisição, a conexão é liberada e volta a ficar disponível nos recursos do sistema.

O bloco *Controlador* adota um modelo de componente simples para representar um controlador de nuvem. Como o controlador é passível de falhas, ele também será considerado no cálculo da disponibilidade do sistema. Os atributos das transições do modelo da Figura 24 são mostrados na Tabela 5.

Como dito anteriormente, o modelo é avaliado pela ferramenta a partir de métricas

definidas como uma expressão que pode conter números, operadores lógicos e algébricos (ZIMMERMANN et al., 2000). Essas métricas são criadas a partir de duas medidas básicas:

- $P\{< expressão >\}$ - calcula a probabilidade da rede encontrar uma marcação que satisfaça a expressão booleana entre chaves.
- $E\{\# < expressão >\}$ - calcula o número médio de marcas no lugar determinado.

4.7.5.1 Métricas de Desempenho

Em um sistema que não sofre com gargalos, pode-se assumir que a taxa de transferência é igual a taxa de chegada de operações, ou seja, o número médio de operações processadas por unidade de tempo (JAIN, 1991). O modelo da Figura 24 tem apenas um nó e, por isso, a vazão do sistema é igual a vazão deste nó, o cálculo é mostrado na Equação 4.3, na qual o valor $processingDelay$, associado a transição $ProcessingRequest$, é o tempo gasto por cada operação no sistema. A sub-expressão $E\{\#ReceivingRequest\}$ representa o número de operações no sistema.

$$TP_{System} = TP_{node} = E\{\#ReceivingRequest\} * (1/processingDelay) \tag{4.3}$$

4.7.5.2 Métricas de Disponibilidade

Para calcular a disponibilidade do sistema requer que o controlador e um dos servidores com o banco de dados estejam funcionando, a Equação 4.4 mostra o cálculo desta métrica. Uma forma de facilitar a visualização de pequenos valores de disponibilidade é expressar a métrica em termos de *downtime*, que é calculado usando a indisponibilidade do sistema. O *downtime* anual em horas é calculado pela Equação 4.5. A disponibilidade é dada pela probabilidade do controlador e do nó de computação estarem funcionando. A indisponibilidade utilizada no cálculo do *downtime* é o complemento do valor da disponibilidade.

$$A_{System} = P\{(\#NodeOn = 1)AND(\#ControllerOn = 1)\} \tag{4.4}$$

$$DT_{System} = (1 - A_{System}) * 8760 \tag{4.5}$$

4.7.6 Modelo *Replica Set*

O modelo da arquitetura replicada na Figura 25 representa a implementação de um *replica set* do MongoDB com dois nós secundários que funcionam como redundâncias para o caso de falhas no primário. Segundo a documentação, a eleição de um novo nó primário não excede 12 segundos, o que configura um cenário de *warm-standby*. Devido o valor desse

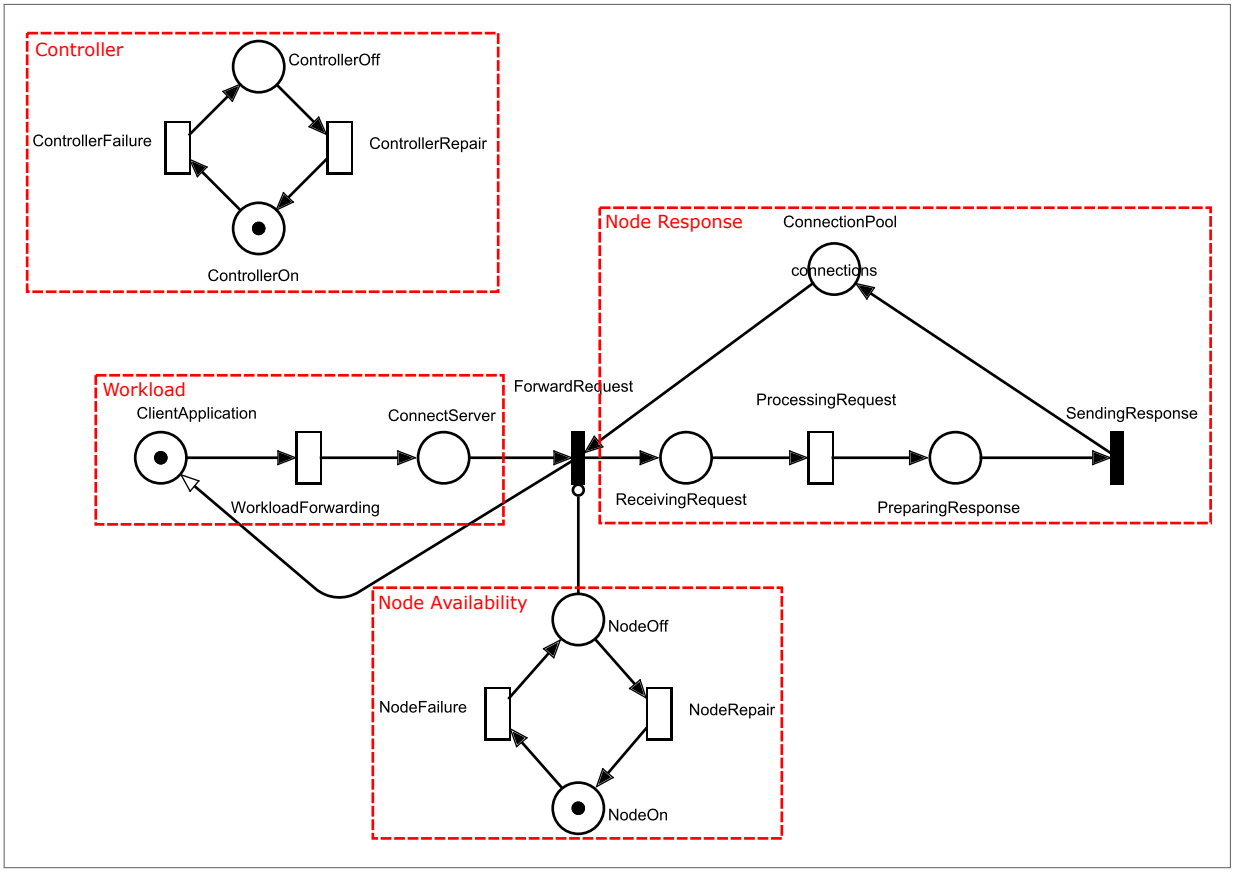


Figura 24 – Modelo GSPN do sistema *Single Server*.

Tabela 5 – Transições do modelo - *Single Server*.

Transição	**Tipo**	**Tempo**	**Concorrência**
$WorkloadForwarding$	T.T.	$arrivalDelay$	s.s.
$ForwardRequest$	T.I.	-	-
$ProcessingRequest$	T.T.	$processingDelay$	i.s.
$SendingResponse$	T.I.	-	-
$ControllerFailure$	T.T	$MTTF_{Controlador}$	s.s.
$ControllerRepair$	T.T.	$MTTR_{Controlador}$	s.s.
$NodeFailure$	T.T.	$MTTF_{NódeComp.}$	s.s.
$NodeRepair$	T.T.	$MTTR_{NódeComp.}$	s.s.

intervalo de ativação ser muito pequeno e a implementação desse de tipo de mecanismo aumentar muito a complexidade do modelo, foi considerado um mecanismo de *hot-standby*. Dessa forma, não há atraso entre a falha do nó primário e a ativação do nó secundário.

Este modelo foi criado com a mesma abordagem do modelo anterior, a estrutura que representa o funcionamento do servidor e seu estado de funcionamento foram replicadas para representar a adicionado de mais máquinas ao sistema. Um arco partindo do lugar $ConnectServer$ para $ForwardRequest\_R1$ e $ForwardRequest\_R2$ são adiciona-

dos e arcos inibidores são incluídos do lugar $NodeOn$ (nó primário) para as transições $ForwardRequest\_R1$ e $ForwardRequest\_R2$. Para o cálculo da disponibilidade, expressões de guarda verificam a condição de funcionamento do controlador em cada transição $ForwardRequest\_Rx$. Arcos inibidores desativam os nós secundários quando o primário está operacional.

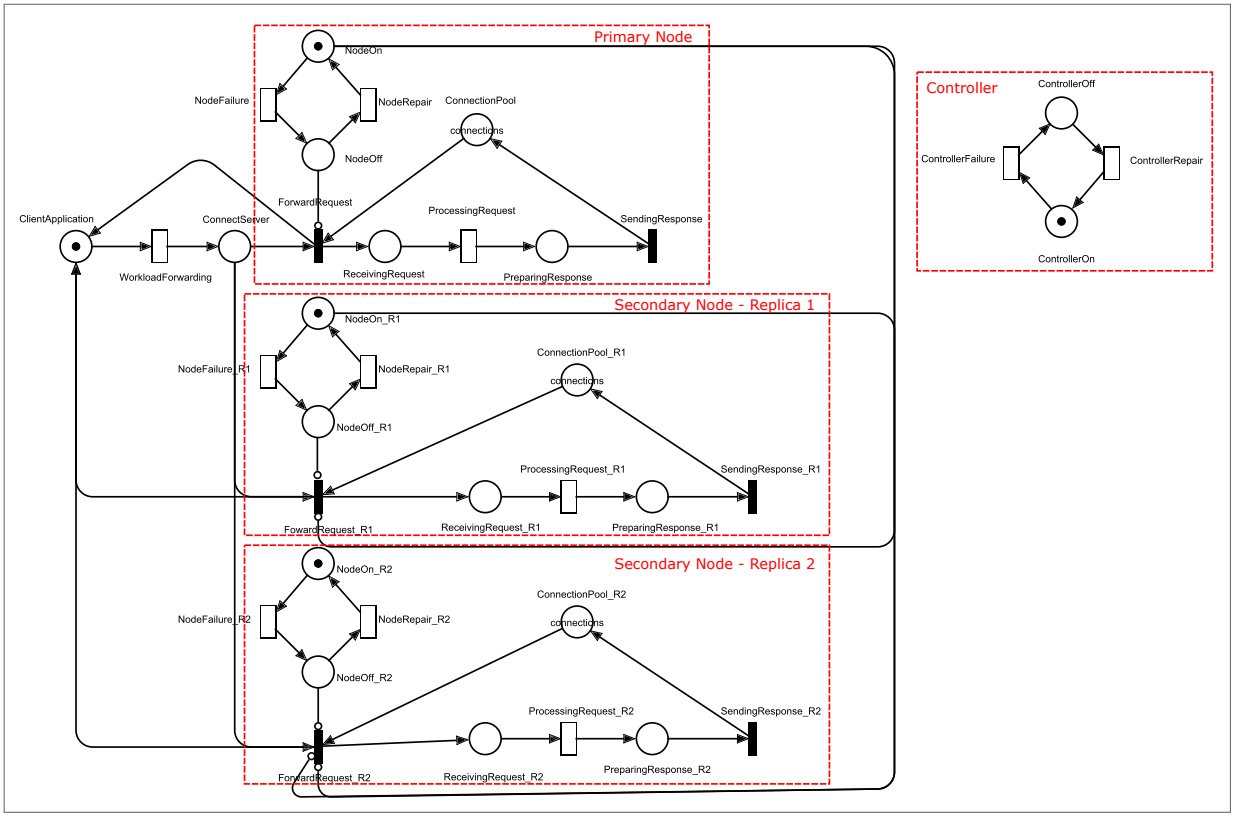


Figura 25 – Modelo GSPN do sistema *Replica Set*.

4.7.6.1 Métricas de Desempenho

A vazão do sistema é a soma das taxas de transferências de todos os nós, baseado na Figura 25, a métrica é mostrada na Equação 4.6.

$$TP_{System} = (E\{\#ReceivingRequest\} + E\{\#ReceivingRequest\_R1\} + E\{\#ReceivingRequest\_R2\}) \times (1/processingDelay) \tag{4.6}$$

4.7.6.2 Métricas de Disponibilidade

A disponibilidade é estimada considerando que o controlador e pelo menos um nó com o SGBD funcionando, a Equação 4.7 mostra o fórmula utilizada para a estimativa.

$$A_{System} = P\{((\#NodeOn = 1)OR(\#NodeOn\_R1 = 1)OR(\#NodeOn\_R2 = 1)) AND(\#ControllerOn = 1)\} \tag{4.7}$$

Tabela 6 – Transições do modelo de *Replica Set*.

Transição	**Tipo**	**Tempo**	**Concorrência**
$WorkloadForwarding$	T.T.	$arrivalDelay$	s.s.
$ForwardRequest$, $ForwardRequest\_R1$, $ForwardRequest\_R2$	T.I.	-	-
$ProcessingRequest$, $ProcessingRequest\_R1$, $ProcessingRequest\_R2$	T.T.	$processingDelay$	i.s.
$SendingResponse$, $SendingResponse\_R1$, $SendingResponse\_R2$	T.I.	-	-
$ControllerFailure$	T.T.	$MTTF_{Controlador}$	s.s.
$ControllerRepair$	T.T.	$MTTR_{Controlador}$	s.s.
$NodeFailure$, $NodeFailure\_R1$, $NodeFailure\_R2$	T.T.	$MTTF_{NódeComp.}$	s.s.
$NodeRepair$, $NodeRepair\_R1$, $NodeRepair\_R2$	T.T.	$MTTR_{NódeComp.}$	s.s.

4.7.7 Análise das propriedades dos modelos

Aqui será feita uma análise para verificar quais são as propriedades comportamentais e estruturais que os modelos das Figuras 24 e 25 possuem. As técnicas de análise e verificação de redes de Petri podem ser categorizadas em três grupos: análise baseada na árvore de cobertura, métodos baseados na equação fundamental e técnicas de redução. Neste trabalho foi utilizado grafo de cobertura e uma técnica derivada da equação fundamental, o cálculo de invariantes de transição e lugar (MACIEL; LINS; CUNHA, 1996).

Segundo Maciel, Lins e Cunha (1996), a partir de uma rede de Petri marcada $N = (L, T, I, O, K, M_0)$, define-se um grafo de cobertura pelo par $GC = (E, A)$, onde E é o conjunto de todos os nós da árvore de cobertura e A os aros rotulados pelas transições pertencentes a rede N representando os possíveis disparos de transição tal que $M_i[t_k > M_j$, onde $M_i, M_j \in S$. Nos casos onde a rede possui a propriedade de limitação, pode ser chamado de grafo de alcançabilidade (MACIEL; LINS; CUNHA, 1996). A partir da análise do grafo de cobertura da rede de Petri dos modelos criados, foi verificado que ambos possuem as mesmas propriedades estruturais e comportamentais.

A escolha das propriedades a serão analisadas foram por suas importâncias no funcionamento do modelo e como isso se traduz no funcionamento de um sistema computacional real, avaliando a possibilidade de *deadlocks*, utilizando recursos não previstos inicialmente e a capacidade do sistema retornar ao estado inicial a partir da execução de uma ta-

refa. Verificar a presença dessas propriedades é importante para identificar problemas nos modelos e saber como eles podem ser resolvidos, se por simulação ou análise estacionária.

A primeira propriedade comportamental a ser verificada foi a de Limitação, isso quer dizer que todos seus lugares podem ter um número finito de marcas para toda marcação possível, representando o uso limitado de recursos e a ausência da criação de recursos infinitos. Podem ser destacados os lugares *ConnectionPool* e *RecievingRequest*, que são k-limitados, onde k é o número de marcas no lugar *ConnectionPool* que representam as conexões de disponíveis. As redes limitadas garantem que o grafo de alcançabilidade será finito, logo, o sistema modelado possui uma quantidade limitada de estados.

O fato da rede ser Viva significa que não existe uma marcação que impossibilite o disparo de uma transição do modelo. Essa propriedade está ligada ao conceito de *deadlock*, pois quando todas as transições ficam bloqueadas, significa que todo o sistema também está bloqueado. Ao verificar esta propriedade pode ser identificado problemas sérios, prevendo cenários de falha de um sistema real. Também é importante citar que ambos modelos não possuem transições mortas.

Outra propriedade observada foi a de Reversibilidade que é a capacidade do modelo de retornar a marcação inicial a partir de qualquer marcação. Isso é facilmente visto no modelo criado, pois em todos os blocos nos quais o modelo foi dividido é possível ver que as marcações são ciclos curtos, que em um determinado momento volta ao estado inicial. O fato da rede ser reversível pode ser vista com bons olhos pelo fato que não há ações irreversíveis, no sentido de que levam o sistema para um estado que não possa mais voltar ao estado inicial naturalmente (VALMARI, 1998).

Algumas propriedades não podem ser analisadas através do grafo de cobertura, nesses casos, outros tipos de técnicas de análise são recomendadas. Duas técnicas baseadas na equação fundamental são os cálculos dos invariantes de transição e lugar. Foi verificado que as redes dos modelos criados nessa dissertação são cobertas por ambos tipos de invariantes. Através do cálculo de invariantes de transição, é possível identificar componentes repetitivos e a averiguar se a rede é consistente. Com o cálculo de invariantes de lugar é verificado se a rede é estruturalmente limitada e conservativa.

4.7.8 Espaço de estados

Métodos que analisam o sistema baseados no espaço de estados partem do princípio de construir uma estrutura que consistem em todos os estados que um sistema pode alcançar, além de todas as transições entre os estados. A construção desse espaço de estados pode ser feita de maneira completamente automática, e isso é uma das vantagens de se trabalhar com métodos desse tipo. Outra vantagem que pode ser apontada, é o fato da representação de estados possibilitar o *debug* da rede fornecendo informações sobre o funcionamento do sistema (VALMARI, 1998).

Tabela 7 – Espaço de estados para os modelos GSPN.

Modelos	**Quantidade de estados**
Modelo *single-server* com 5 conexões	168
Modelo *single-server* com 20 conexões	1.848
Modelo *replica set* com 5 conexões	296.352
Modelo *replica set* com 20 conexões	394.444.512

Apesar das vantagens citadas no parágrafo anterior, esse método apresenta um problema chamado de explosão de estados, que pode inviabilizar a utilização da técnica dependendo do sistema que será modelado. Como o próprio nome diz, o problema está relacionado com o crescimento desenfreado da quantidade de estados no grafo de cobertura. O número de estados está relacionado com o tamanho do modelo e a representação de tarefas ou recursos dentro do sistema.

Durante o processo de análise dos modelos criados neste trabalho foram gerados os grafos de cobertura e a partir deles foi computado a quantidade de estados possíveis em cada um. Foram consideradas duas situações, quando o SGBD possui 5 e 20 conexões disponíveis para os usuários, isso foi feito modificando a quantidade de marcas no lugar *ConnectioPool*. Os valores adotados para as quantidades de conexões se mostraram adequados para representar a explosão de estados e criar uma quantidade de estados que permitia analisar o modelo rapidamente. Os resultados podem ser vistos na Tabela 7.

Observando os valores da Tabela 7 pode ser notado que tanto o aumento da quantidade de servidores representados no modelo, quanto o aumento de recursos, neste caso, as conexões, são fatores chaves para que ocorra um aumento no número de estados. A adição de dois servidores acarreta num aumento de mais de 295 mil estados no grafo de cobertura, e uma arquitetura como essa é considerada pequena caso o objetivo seja criar um ambiente de produção. A quantidade de conexões ou recursos disponíveis é outro fator que causa a explosão de estados nos modelos. Os valores utilizados para a quantidade de conexões no cálculo do espaço de estados reportado na Tabela 7 foram pequenos quando comparado a implementações reais, onde esse valor pode ser na ordem de centenas ou milhares.

O modelo da Figura 25 possui apenas a representação de 3 servidores, para modelar um cenário com mais servidores seria necessário a repetição de submodelos aumentando o espaço de estados e a complexidade do modelo final. A álgebra de composição de modelos GSPN proposta no trabalho de Junior (2007) pode ser uma aliada para a criação de modelos que simulem sistemas maiores, pois a abordagem de composição *bottom-up* baseia-se na fusão de lugares e transições de mesmo nome de duas GSPNs. A composição é realizada através da fusão de transições ou lugares. Os invariantes de transição (T-invariantes) e invariantes de lugar (L-invariantes) do modelo composto serão obtidos pela combinação linear dos L-invariantes e T-invariantes dos modelos anteriores. Esses

invariantes determinam se as redes de Petri possuem certas propriedades que permitam a computação das métricas a partir de análise transiente ou estacionária (JUNIOR, 2007).

5 VALIDAÇÃO DO MODELO E RESULTADOS

Este capítulo apresenta a validação do modelo de arquitetura *single-server* e mostra os resultados de experimentos realizados nos modelos GSPN apresentados no Capítulo 4 utilizando um *design* de experimentos fatorial para avaliar o impacto de diferentes fatores no desempenho e disponibilidade do sistema.

5.1 VALIDAÇÃO DO MODELO

A validação tem como objetivo comprovar que o modelo é capaz de servir como um meio de abstração do sistema real apresentando valores próximos como resultado. Neste trabalho para validar o modelo GSPN foi criado um ambiente de nuvem privada utilizando o $framework$ Openstack, a infraestrutura consistia de um nó controlador e um nó de computação. Neste nó de computação foi criada uma máquina virtual (VM) e instalado o SGBD MongoDB que funciona como sistema de armazenamento da nuvem. Com o ambiente configurado, já está apto para receber requisições HTTP contendo as operações que serão realizadas no banco de dados.

A geração da carga de trabalho fica a cargo de outra máquina física, que é dedicada a essa função através da ferramenta YCSB (COOPER et al., 2010). Nos trabalhos citados no Capítulo 2 essa é a ferramenta mais comum em experimentos que desejam avaliar o desempenho de SGBDs através da medição dos sistemas reais. Com o YCSB é possível configurar o tipo de operação, o tamanho da informação a ser inserida, a quantidade de *threads* simultâneas enviando operações, taxa de transferência fixa, a quantidade de operações, entre outros.

Para realizarmos medições na nuvem privada e coletar dados para parametrizar o modelo foi utilizada uma carga de trabalho composta de operações de leitura (50%) e atualização (50%). Devido aos recursos limitados não foi possível realizar os testes com operações de inserção, e a carga mista com leitura e atualização se mostrou adequada para coleta de informação. A quantidade de operações enviada ao sistema foi fixada em 100.000, independentemente do número de usuários simultâneos simulados. A quantidade de usuários foi variada entre 1, 5, 10 e 20; nos casos com mais de um usuário o número de operação enviadas era dividido igualmente. Exemplificando, quando há 10 usuário conectados cada um envia 10.000 operação, se existem 20 usuários conectados cada um envia 5.000 operações. Para cada valor de usuários foram obtidas 30 amostras de execuções no MongoDB. A Tabela 8 mostra os valores médios obtidos.

Como era de se esperar, a vazão do sistema aumenta e o tempo de execução diminui conforme o número de usuários simultâneos cresce. O aumento mais acentuado acontece quando o número de usuários é aumentado de 1 para 5, nesse cenário a vazão aumenta

Tabela 8 – Dados coletados no experimento de carga.

Usuários	**Vazão do sistema (ops/s)**	**Tempo de Execução (s)**	**Uso de CPU (%)**	**Uso de Memória RAM (%)**
1	804.68	124.74	9.31	18.57
5	3897.86	25.65	38.79	17.55
10	5823.43	17.17	59.66	18.32
20	7039.27	14.2	78.57	19.05

em quase 5 vezes, a partir daí os valores vão aumentando de maneira gradual. Outro dado observado durante a execução das cargas foi a utilização de CPU e memória RAM. Na Tabela 8 é possível notar que a variação dos valores de utilização de CPU acompanha o mesmo padrão da taxa de transferência, aumentando a medida que o número de usuários aumenta. O valor de utilização de memória RAM praticamente não varia, independente da quantidade de usuários a utilização fica entre 17% e 19%, o que é uma variação pequena quando comparado com outros dados mostrados pela tabela.

Para parametrizarmos o modelo GSPN com o intervalo entre o envio das requisições pelo cliente e o tempo gasto para processar uma operação foram utilizadas duas leis operacionais chamadas *Little's Law* e *Utilization Law*. A *Utilization Law* estabelece uma relação entre a vazão X e o tempo de serviço S de um dispositivo i para fornecer a utilização U desse dispositivo, como é mostrado na Equação 5.1 (JAIN, 1991) (OEHLERT, 2010). Na etapa de coleta de dados foi observado que o processador era o componente mais exigido pelo aumento da carga de trabalho no sistema. Aplicando a *Utilization Law*, calculamos o tempo gasto para o processamento de uma operação. O valor dos intervalos entre o envio de requisições e o tempo de processamento de cada configuração são mostrados na Tabela 9, nas colunas *Intervalo entre requisições* e *Tempo de processamento*.

$$U_i = X_i \times S_i \tag{5.1}$$

Para representar o aumento da carga de trabalho que é submetida ao sistema a medida que o número de usuários simultâneos cresce, o tempo entre envios vai diminuindo. Já o tempo de processamento não varia bruscamente o que revela uma relação proporcional entre a vazão e a utilização da CPU das configurações que foram testadas.

No modelo, a transição $WorkloadForwarding$ adota os valores da coluna *Intervalo entre requisições* e a transição $ProcessingRequest$ utiliza os valores da coluna *Tempo de processamento*. Para cada configuração, a vazão foi estimada, então, os resultados do modelo são comparados ao do sistema real. A Tabela 10 apresenta os resultados da vazão do modelo, na coluna *Vazão do modelo*, e a vazão do sistema na nuvem privada, na coluna *Vazão do sistema*. Como os resultados do modelo estão contidos no intervalo de confiança ($I.C.$) com 95% de confiança (nível de significância $\alpha = 0.05$), obtidos com as

Tabela 9 – Parâmetros do modelo de GSPN.

Usuários	**Intervalo entre requisições (ms)**	**Tempo de processamento (ms)**
1	1.247	1.16×10^{-1}
5	2.566×10^{-1}	9.93×10^{-2}
10	1.717×10^{-1}	1.03×10^{-1}
20	1.420×10^{-1}	1.13×10^{-1}

Tabela 10 – Resultados do modelo para validação.

	Sistema		**Modelo**
Chegada de cliente (ms)	**Vazão (ops/s)**	**I.C.(95%) (ops/s)**	**Vazão (ops/s)**
1.247	801.64	[780.51; 823.72]	803.54
2.566×10^{-1}	3897.86	[3868.92, 3939.48]	3909.64
1.717×10^{-1}	5823.43	[5717.55, 5824.11]	5823.29
1.420×10^{-1}	7039.27	[6788.86, 7067.13]	7038.84

vazões medidas do sistema não há evidências estatísticas para indicar que o modelo não representa o sistema real.

5.2 EXPERIMENTOS BASEADOS EM DOE

O objetivo dos experimentos é verificar o impacto que a variação da disponibilidade causa no desempenho. Neste estudo, isso será feito pela variação nos valores de MTTF, MTTR e número de servidores, que são fatores que impactam diretamente na disponibilidade do sistema. Os experimentos utilizarão o modelo da arquitetura simples mostrado na Figura 24 e o modelo que representa o sistema distribuído mostrado na Figura 25. Esta etapa conta com três experimentos que submetem os modelos a diferentes configurações e avaliam o desempenho, disponibilidade e *downtime*.

Como foi descrito no Capítulo 4, o DoE fatorial é baseado nos conceitos de fatores e níveis. Estruturar os experimentos dessa maneira traz algumas vantagens, como o fato de tornar possível a análise das interações entre fatores e evitar a realização de experimentos desnecessários (JAIN, 1991) (OEHLERT, 2010).

Os experimentos realizados são baseados em um *design* de experimentos fatorial l^k, onde l são os níveis e k os fatores. Para cada combinação fator/nível, chamada de tratamento, um modelo GSPN foi avaliado usando análise estacionária. Os resultados são apresentados usando um ranking de efeitos com os valores obtidos em cada tratamento. Os *rankings* são ordenados em ordem decrescente, levando em conta o valor absoluto de todos os efeitos.

Para exemplificar o cálculo dos efeitos principais e efeitos de interações entre os fatores,

Tabela 11 – Projeto de experimentos.

	Fatores			**Interações**			
Tratamento	**A**	**B**	**C**	**AB**	**AC**	**BC**	Resposta
1	-	-	-	+	+	+	Y_1
2	+	-	-	-	-	+	Y_2
3	-	+	-	-	+	-	Y_3
4	+	+	-	+	-	-	Y_4
5	-	-	+	+	-	-	Y_5
6	+	-	+	-	+	-	Y_6
7	-	+	+	-	-	+	Y_7
8	+	+	+	+	+	+	Y_8
Efeito	X_A	X_B	X_C	X_{AB}	X_{AC}	X_{BC}	

a Tabela 11 representa um projeto de experimentos com três fatores (A, B e C), e cada fator possui dois níveis (+ e -). Na tabela estão representadas as interações até dois níveis. A diferença entre a média das *Respostas* dos níveis positivos, e a média das *Respostas* dos símbolos negativos, representa o cálculo do efeito, a Equação 5.2 apresenta a fórmula. Para as interações os padrões de positivos e negativos são obtidos multiplicando os sinais dos fatores relacionados. Para obter o efeito da interação basta aplicar a Equação 5.2.

$$Efeito = \frac{\sum Y+}{n} - \frac{\sum Y-}{n} \tag{5.2}$$

Exemplificando com dados deste estudo, conforme a Tabela 13, para calcular o efeito do fator *ConnectionNumber* no primeiro experimento, é necessário calcular a média da vazão para o nível 20 (conexões simultâneas), cujo o resultado é 23758.73 operações por segundo; e depois, calcular a média para o nível 5, que resulta em 15007.24 operações por segundo. Com a subtração desses valores médios é obtido o efeito, cujo valor é de 8751.49 operações por segundo. Dessa forma, foi o *ranking* foi criado, calculando os efeitos principais e interações de segunda ordem, ordenando os valores em ordem decrescente.

5.2.1 Experimento 1

O primeiro experimento, nomeado de experimento de performabilidade considera 4 fatores ($k = 4$) com 2 níveis ($l = 2$). Os fatores e níveis são descritos a seguir:

- o número de conexões disponíveis no banco de dados ($ConnectionNumber$) - 5, 20;
- o tempo necessário para processar uma operação ($ProcessingDelay$) - $1.29\times10^{-1}ms$, $6.20 \times 10^{-1}ms$;
- o tempo médio de falha de um nó de computação ($MTTF$) - $50h$, $228.69h$;

Tabela 12 – Experimento 1 - *Ranking* de efeitos na vazão.

Fatores e interações	**|Efeito|** (ops/s)
$ConnectionNumber$	8751.5
$ProcessingDelay$	-7865.7
$ConnectionNumber * ProcessingDelay$	6880.6
$MTTF$	550.4
$MTTR$	-504.5
$MTTR * MTTF$	314.8
$ConnectionNumber * MTTF$	124.3
$ConnectionNumber * MTTR$	-113.9
$ProcessingDelay * MTTF$	-111.7
$ProcessingDelay * MTTR$	102.4

- o tempo médio de reparo de um nó de computação ($MTTR$) - $0.775h$, $3h$.

$ConnectionNumber$ é um parâmetro do SGBD definido pelo administrador do sistema para controlar a quantidade de usuários conectados simultaneamente ao sistema. A possibilidade de configurar este tipo de parâmetro é útil em sistemas com poucos recursos, pois é uma forma de controlar o gasto de recursos do sistema. O fator $ProcessingDelay$ é o tempo para processar uma operação e a variação desse valor pode ser alcançada modificando a capacidade dos servidores e qualidade da rede. Este valor é associado com a transição *ProcessingRequest*. $MTTF$ e $MTTR$ são relacionados com o nó de computação e seus valores podem variar de acordo com a qualidade do hardware ou com a qualidade do serviço de manutenção.

Os resultados dos experimentos utilizando o DoE descrito são apresentados na Tabela 12 que contém os efeitos dos fatores e de suas interações de segunda ordem, por exemplo, $ProcessingDelay * ConnectionNumber$. Como dito no Capítulo 4, no momento de realizar a análise dos experimentos pode-se assumir que as execuções de ordem mais alta são negligenciáveis (MONTGOMERY, 2008).

Observando os resultados, $ConnectionNumber$ tem a maior influência no desempenho, isso pode ser pelo fato da quantidade de usuários conectados no SGBD ser diretamente proporcional a quantidade de requisições por segundo que eram submetidas ao sistema. Aumentando o número de usuários conectados ao sistema de 5 para 20, nos garantiu um aumento de aproximadamente 8.800 operações por segundo na vazão. O tempo de processamento ($ProcessingDelay$) também tem um grande impacto na vazão do sistema, quanto menor o seu valor maior é a vazão, isso é evidenciado pelo sinal negativo no valor do efeito. Devido aos dois fatores terem grande impacto individualmente ocasiona que sua interação, $ConnectionNumber * ProcessingDelay$, tem o maior efeito entre todas as interações.

Os outros dois fatores, $MTTF$ e $MTTR$, também podem levar a uma variação no desempenho do sistema apesar dos valores obtidos no experimento não serem tão significativos. Dependendo dos valores adotados, os resultados podem ser mais impactantes, além de que esses fatores são de extrema importância na avaliação da disponibilidade do sistema. Conforme era esperado, o aumento do tempo entre as falhas causam um impacto positivo no desempenho, e o aumento do tempo para reparo, um impacto negativo. E as interações desse fatores com $ConnectionNumber$ e $ProcessingDelay$ apresentam um equilíbrio causando um variação com valores próximos a 100 ops/s.

A Tabela 13 apresenta na coluna $Vazão$ a taxa de transferência para cada tratamento do DoE. O tratamento que apresenta a maior vazão é para os parâmetros onde o sistema suporta mais usuários simultâneos e possui maior tempo entre falhas, em conjunto do menor tempo para processamento e menor tempo para reparo.

Neste caso, um projetista de sistema deve avaliar melhor no momento da aquisição dos componentes para aumentar o número de conexões disponíveis e, também, reduzir o tempo para processar uma requisição. A maioria dos valores ficam entre 24.000 e 21.000 operações por segundo, e nesses resultados os fatores $ConnectionNumber$ e $ProcessingDelay$ mostram sua influência. As últimas 4 linhas possuem os piores resultados apresentam uma queda abrupta na vazão, saindo da faixa das 20.000 operações por segundo e caindo para menos de 8.000.

Uma maneira de visualizarmos o impacto dos fatores e suas interações é através dos chamados gráficos de efeitos principais e gráficos de interação. Um gráfico de efeito principal representa a diferença causada na variável resposta (variável dependente) pela variação de valores nos níveis do fator (variável independente) alvo. A média dos resultados para cada nível são representadas por pontos conectados por uma linha. Quando a linha é horizontal (paralela ao eixo x), então não há efeito principal pois cada nível do fator afeta a variável dependente da mesma forma. Quando a linha não é horizontal, há um efeito principal e os diferentes níveis do fator afetam a resposta de maneira diferente. Quanto mais íngreme a inclinação da linha, maior a magnitude do efeito principal.

No entanto, pode ser um erro avaliar apenas os efeitos principais, os efeitos de interação ocorrem quando variáveis independentes interagem entre si, aumentando a complexidade na relação para variável resposta (dependente). Na Figura 26 são mostrados os efeitos principais para todos os fatores, e na Figura 27 algumas das interações mais relevantes.

Na Figura 26 é possível perceber que os fatores que obtiveram os grandes valores de efeito possuem uma reta com grande inclinação, enquanto $MTTF$ e $MTTR$ são representados por uma reta pouco inclinada. A Figura 27 só apresenta as interações com maior relevância estatística definida pelo *p-value*. No eixo y está a variável dependente, ou resposta, e no eixo x a primeira variável independente enquanto as linhas representam a segunda variável independente. A Figura 27a mostra a interação $ProcessingDelay * ConnectionNumber$, com os valores utilizados não foi possível mos-

Tabela 13 – Experimento 1 - Resultados do experimento para o cálculo de vazão.

MTTF (h)	**MTTR** (h)	**ConnectionNumber**	**ProcessingDelay** (ms)	**Vazão** (ops/s)
228.69	0.775	20	1.29×10^{-1}	24714.26
228.69	3.000	20	1.29×10^{-1}	24476.92
50.00	0.775	20	1.29×10^{-1}	24419.53
228.69	0.775	20	6.20×10^{-1}	23710.37
228.69	3.000	20	6.20×10^{-1}	23482.67
50.00	0.775	20	6.20×10^{-1}	23427.63
50.00	3.000	20	1.29×10^{-1}	23394.37
228.69	0.775	5	1.29×10^{-1}	22807.63
228.69	3.000	5	1.29×10^{-1}	22588.61
50.00	0.775	5	1.29×10^{-1}	22535.65
50.00	3.000	20	6.20×10^{-1}	22444.12
50.00	3.000	5	1.29×10^{-1}	21589.58
228.69	0.775	5	6.20×10^{-1}	7779.84
228.69	3.000	5	6.20×10^{-1}	7705.13
50.00	0.775	5	6.20×10^{-1}	7687.12
50.00	3.000	5	6.20×10^{-1}	7364.41

trar uma interação entre os fatores, pois as retas não se cruzam, o fato interessante a ser observado é que quando $ConnectionNumber$ é 20 a variação no valor $ProcessingDelay$ não faz grande diferença, enquanto que para um sistema com 5 conexões simultâneas ter um tempo de processamento baixo é essencial. Na Figura 27b está representada a interação $MTTR * MTTF$ e mostra que quando o tempo de reparo é baixo (menos de 1 hora) a frequência de falhas não importa muito para o desempenho do sistema, mas conforme a manutenção leva mais tempo, a tendência é que haja um distanciamento entre os pontos e dessa forma ocorra um impacto significativo no desempenho.

5.2.2 Experimento 2

O segundo experimento adota disponibilidade e *downtime* como métricas de interesse, e por esse motivo foi nomeado de experimento de disponibilidade. Agora são levados em consideração os seguintes fatores ($k = 4$) e níveis ($l = 2$):

- o número de servidores ou réplicas do *replica set* ($ServerNumber$) - 1, 3;
- o tempo médio de falha de um nó de computação ($MTTF$) - $50h$, $228.69h$;
- o tempo médio de reparo de um nó de computação ($MTTR$) - $0.775h$, $3h$;
- o número de controladores ($ControllerNumber$) - 1, 2.

Neste experimento, o tempo entre o envio de requisições e tempo de processamento foram mantidos fixos, respectivamente, em $4 \times 10^{-1}ms$ e $6.20 \times 10^{-1}ms$. O número de servidores varia entre 1 e 3 representando a arquitetura de servidor único ou um *replica*

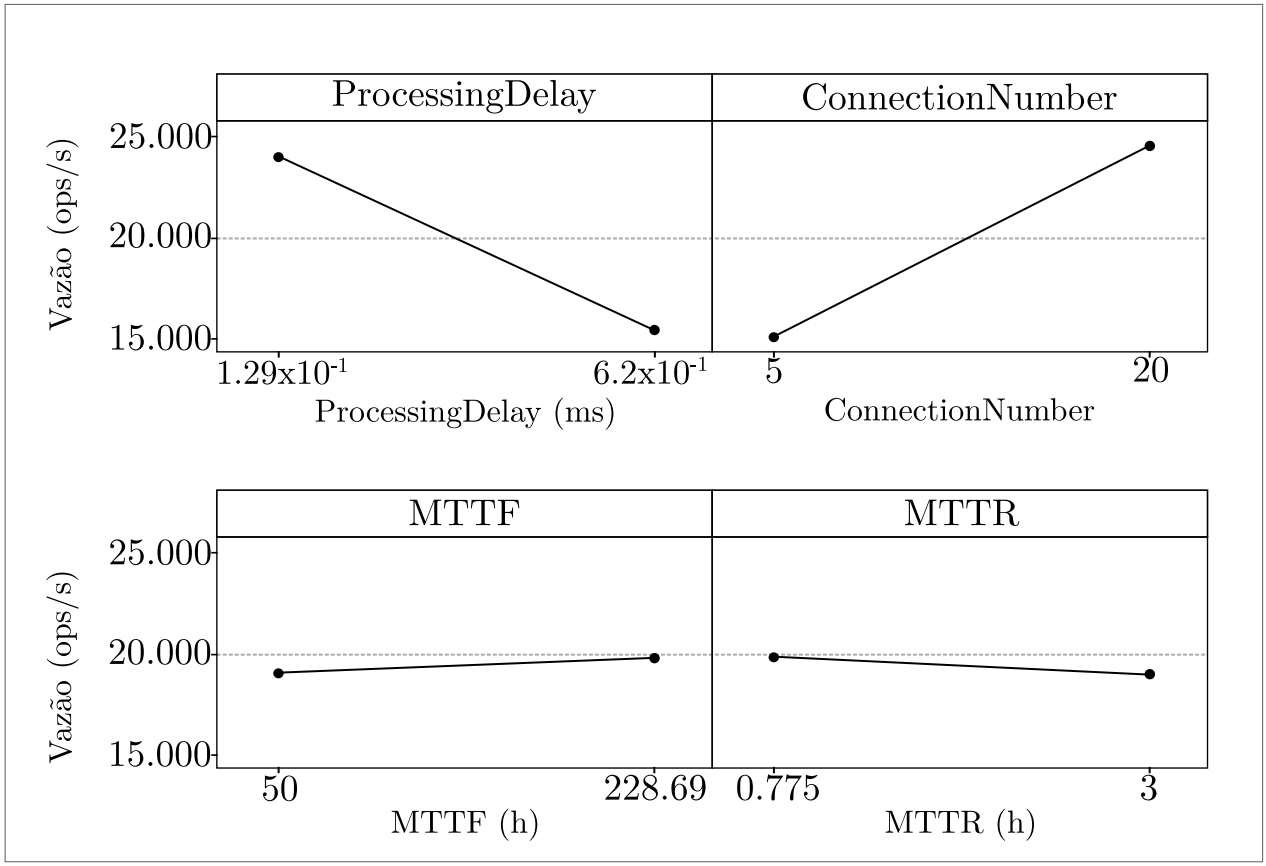
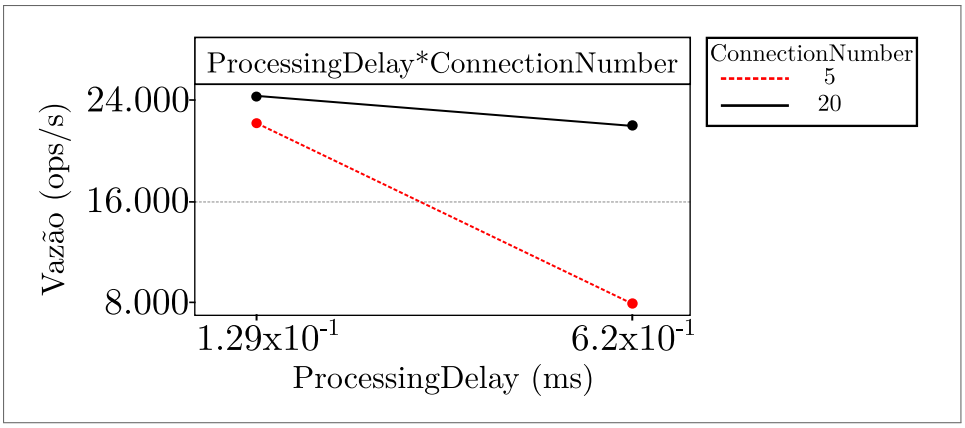


Figura 26 – Experimento 1 - Gráficos de efeito principal.

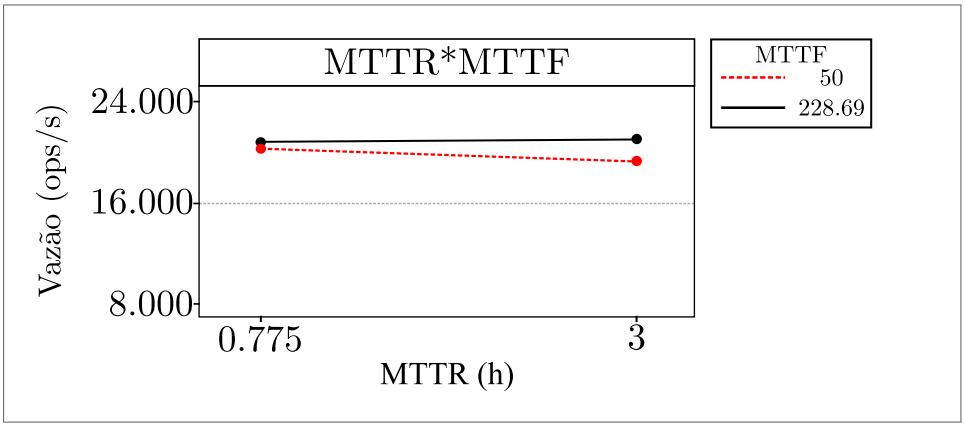
set. Assim como no experimento anterior, MTTF e MTTR são utilizados como fatores e os níveis são mantidos os mesmos. O número de controladores é trazido como um fator pela importância desse componente na nuvem, já que seu funcionamento tem um impacto direto na disponibilidade do serviço.

A Tabela 14 apresenta o *ranking* dos efeitos usando *downtime* (em horas por ano), já que desta forma é mais fácil de visualizar os menores valores de disponibilidade. A quantidade de réplicas ($ServerNumber$), tempo médio para falhas ($MTTF$) e tempo médio para reparo ($MTTR$) são fatores marcantes que afetam a disponibilidade. Sendo que $ServerNumber$ desponta como o mais impactante contribuindo com aproximadamente 183 horas de funcionamento anual quando existe redundância. O $MTTF$ e o $MTTR$ também são fatores bastante relevantes, sendo responsáveis por mais de 100 horas de *downtime* em um ano. As interações de $ServerNumber$ com $MTTF$ e $MTTR$ também tem influência semelhantes no funcionamento do sistema.

A redundância do controlador da nuvem, $ControllerNumber$, possui um impacto menos significante quando comparado aos outros fatores, nas interações com outros fatores a variação dos níveis causa uma variação mínima de menos de 1 hora. Um engenheiro de sistema não pode negligenciar a redundância dos nós dos SGBDs, a indisponibilidade do serviço pode causar multas devido a violação da qualidade dos termos de serviço, perca de usuários e inviabilidade do serviço.



(a)



(b)

Figura 27 – Experimento 1 - Gráfico de interação.

Tabela 14 – Experimento 2 - *Ranking* de efeitos na disponibilidade.

Fatores e interações	**|Efeito|** (h)
$ServerNumber$	-183.34
$MTTF$	-124.04
$MTTR$	119.13
$ServerNumber * MTTF$	118.17
$ServerNumber * MTTR$	-102.57
$ControllerNumber$	-67.27
$MTTR * MTTF$	-72.18
$ControllerNumber * ServerNumber$	-0.78
$ControllerNumber * MTTF$	-0.61
$ControllerNumber * MTTR$	-0.17

Tabela 15 – Experimento 2 - Resultados do experimento para o cálculo da disponibilidade.

MTTF (h)	**MTTR** (h)	**ServerNumber**	**ControllerNumber**	**Disponibilidade**
228.69	0.775	3	2	0.999560
50.00	0.775	3	2	0.999550
228.69	3.00	3	2	0.998476
50.00	3.000	3	2	0.997071
228.69	0.775	1	2	0.996271
228.69	0.775	3	1	0.991902
50.00	0.775	3	1	0.991892
228.69	3.000	3	1	0.990532
50.00	3.000	3	1	0.989264
228.69	0.775	1	1	0.988573
228.69	3.000	1	2	0.986771
50.00	0.775	1	2	0.984395
228.69	3.000	1	1	0.979079
50.00	0.775	1	1	0.976783
50.00	3.000	1	2	0.943137
50.00	3.000	1	1	0.935777

A Tabela 15 apresenta os resultados de cada tratamento. As maiores disponibilidades são obtidas quando há 3 servidores (um primário e dois secundários) e dois controladores alcançando uma disponibilidade com três digitos 9 no seu resultado. Os menores valores estão presentes quando não há redundância nos servidores, o valor de MTTF é baixo e a manutenção em caso de falhas é mais demorada. A Figura 28 mostra o gráfico de efeito para os fatores sem considerar as interações.

Como foi apontado pelos resultados, a replicação dos dados é um fator essencial para a disponibilidade do serviço. A qualidade dos equipamentos e a rápida manutenção do sistema que são expressas, respectivamente, pelos valores de $MTTF$ e $MTTR$ também se mostram fundamentais para garantir um sistema disponível e confiável. Na Figura 29a foi destacada a interação $ServerNumber * MTTF$ que mostra que quando o sistema funciona com apenas 1 servidor é importante ter componente confiáveis, pois há uma grande relação entre o $MTTF$ do nó de computação e a disponibilidade do sistema. Enquanto para a configuração com redundância de infraestrutura o impacto do $MTTF$ é praticamente nulo. Na Figura 29b a interação $ServerNumber * MTTR$ mostra um cenário semelhante, onde a redundância da arquitetura praticamente anula os efeitos negativos que falhas possam causar no sistema.

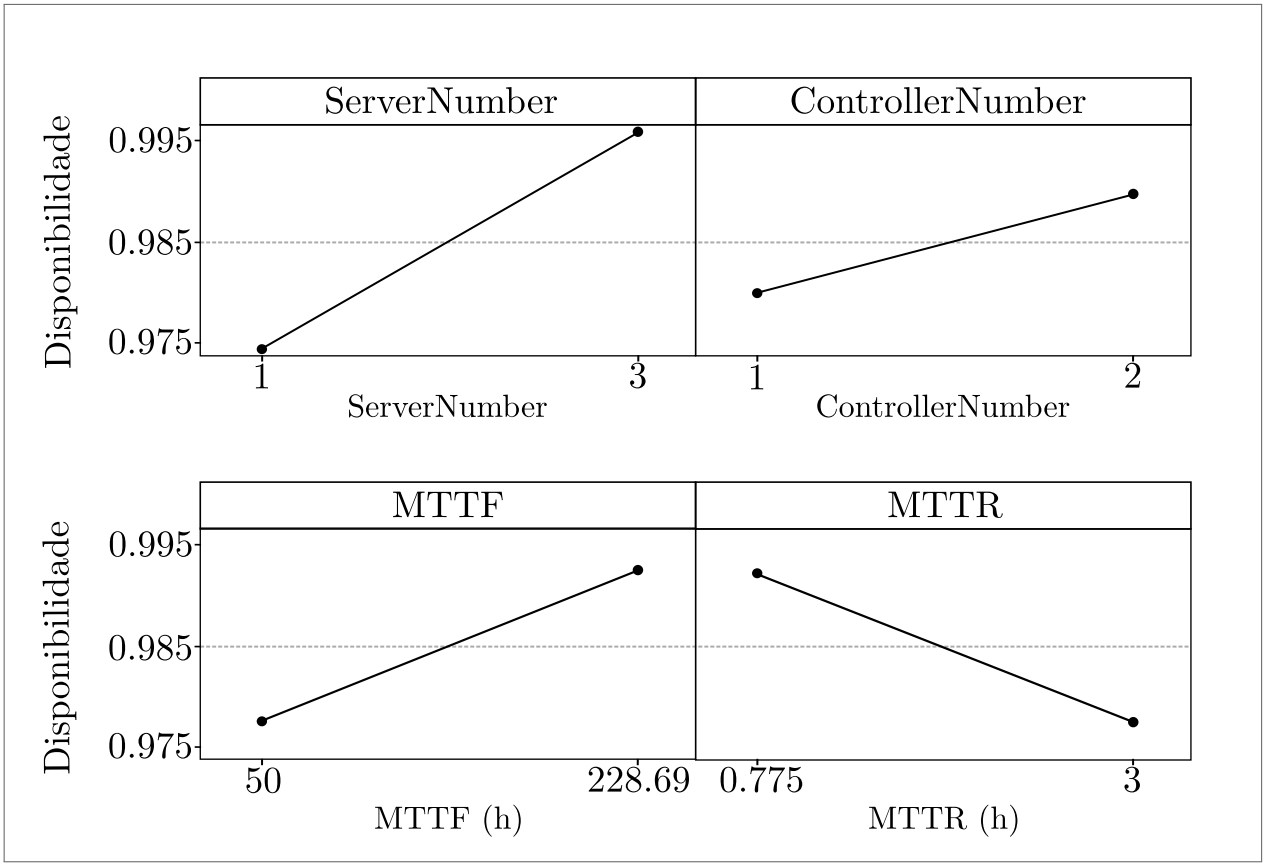
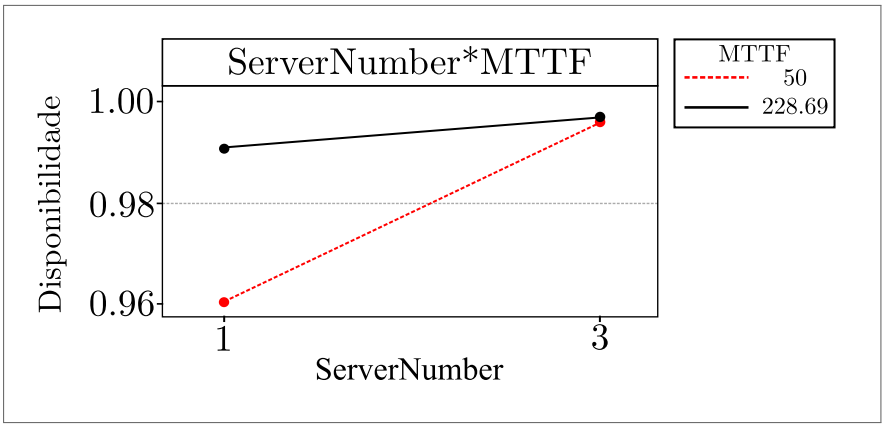


Figura 28 – Experimento 2 - Gráficos de efeito principal.

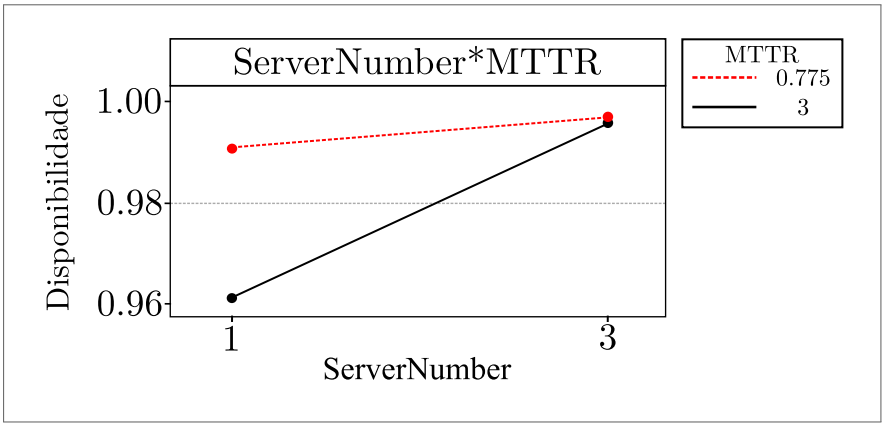
5.2.3 Experimento 3

O modelo criado para representar o sistema com redundância nos servidores funciona de forma que simula o *replica set* do MongoDB em sua configuração padrão, isto é, define o nó primário como o único que está apto a receber e responder operações de escrita e leitura. O MongoDB possibilita outros tipos de configurações, nas quais os nós secundários podem responder requisições de leitura ou que o servidor mais próximo geograficamente seja responsável por atender as requisições. Este tipo de configuração não é recomendado em certos casos de uso pelo fato de não haver garantias que os dados que estão sendo enviados estejam atualizados, pois a sincronização de informações é feita assincronamente, não tendo ordem predefinida. Outro motivo, é o fato que tráfego de escrita é equivalente a todos os nós do *replica set*, ou seja, o nó secundário vai realizar as leituras em um volume próximo ao que o próprio nó primário faria, não obtendo vantagem significativa se o objetivo for a divisão do trabalho.

O terceiro experimento quer mostrar a influência do número de réplicas na vazão do sistema. Como dito anteriormente, é esperado que a arquitetura com 1 servidor ou um *replica set* possua desempenho semelhante, visto que a função da adição de redundâncias não é aprimorar a taxa de transferência do SGBDs distribuídos. Como fatores foram considerados os tempos médio de falha e reparo, o tempo de processamento para cada operação



(a)



(b)

Figura 29 – Experimento 2 - Gráfico de interação.

e o número de servidores. Os valores do $MTTF$ e $MTTR$ são diferentes dos utilizados nos outros experimentos, visando mudar a situação que o experimento pretende simular. A simulação ocorre considerando um sistema menos confiável e com uma manutenção demorada levando até 12 horas para que o serviço seja restaurado.

- o número de conexões disponíveis no banco de dados ($ProcessingDelay$) - $1.29 \times 10^{-1}ms$, $5.83 \times 10^{-1}ms$;
- o número de servidores ($ServerNumber$) - 1, 3;
- o tempo médio de falha de um nó de computação ($MTTF$) - $24h$, $228.69h$;
- o tempo médio de reparo de um nó de computação ($MTTR$) - $0.775h$, $12h$;

A Tabela 16 mostra os resultados do terceiro experimento e como esperado, de acordo com o experimento 1, o único fator que tem grande impacto na vazão do sistema é $ProcessingDelay$ e por isso a tamanha diferença nos efeitos. Como visto anteriormente a estrutura com redundância nos dados possui o maior impacto na disponibilidade, e deve

Tabela 16 – Experimento 3 - *Ranking* de efeitos na vazão.

Fatores e interações	**|Efeito|** (ops/s)
$ProcessingDelay$	-12962
$MTTR$	-2528
$MTTF$	1669
$MTTF * MTTR$	1428
$MTTR * ProcessingDelay$	1414
$MTTF * ProcessingDelay$	-769
$MTTF * ServerNumber$	-757
$ServerNumber$	343
$MTTR * ServerNumber$	187
$ProcessingDelay * ServerNumber$	123

ter pouco impacto na taxa de transferência do sistema. A variação no número de servidores acarreta numa diferença de apenas 343 operações por segundo tendo um impacto muito menos significativo que os outros fatores voltados à disponibilidade do sistema. A Tabela 17 mostra a vazão para todos os tratamentos simulados com o modelo, além da clara distinção nos resultados para o tempo de processamento, a rápida manutenção do sistema aparece como segundo fator mais importante para se obter um bom desempenho. O número de servidores aparece de maneira irregular, sem um padrão bem definido isso exemplifica o baixo efeito que possui na análise dos resultados.

Na Figura 30 a reta do gráfico de efeitos principal do fator $ServerNumber$ é próximo a uma linha reta enquanto os outros gráficos não possuem linhas com inclinações aparentes. A Figura 31a mostra uma interação entre os fatores $MTTF * ServerNumber$ que revela uma dependência entre eles, pois quando há apenas um servidor e o tempo médio de falhas é de 24 horas a vazão é mais baixa que para configuração com três servidores, mas conforme o tempo entre falhas aumenta a arquitetura com um servidor obtém melhores resultados.

Afim de aumentar a capacidade do sistema deve-se optar pelo estratégia do particionamento (*sharding*) dos dados em vários servidores. Sistemas de banco de dados com grandes conjuntos de dados ou aplicativos de alto rendimento podem desafiar a capacidade de um único servidor. Por exemplo, altas taxas de consulta podem esgotar a capacidade da CPU do servidor. Tamanhos de cargas de trabalho maiores que a RAM do sistema exigem mais da capacidade de entrada e saída das unidades de disco. Para implementar um *cluster* particionado é preciso de mais dois tipos de nós que são chamados de *mongo router*, que é responsável por encaminhar as requisições para nós que contenham as repostas. O outro tipo de nó é o *Config Server* que é um componente que possui metadados sobre o SGBD e os dados armazenados, e trabalha em conjunto com o *mongo router* para coordenar o *cluster* de armazenamento.

Tabela 17 – Experimento 3 - Resultados do experimento para o cálculo da vazão.

MTTF (h)	**MTTR** (h)	**ServerNumber**	**ProcessingDelay**	**Vazão**
228.69	0.775	1	1.29×10^{-1}	22807.63
228.69	0.775	3	1.29×10^{-1}	22703.47
24	0.775	3	1.29×10^{-1}	22619.85
24	0.775	1	1.29×10^{-1}	22169.1
228.69	12	1	1.29×10^{-1}	21743.96
228.69	12	3	1.29×10^{-1}	20037.39
24	12	3	1.29×10^{-1}	17495.32
24	12	1	1.29×10^{-1}	15256.66
228.69	0.775	3	5.83×10^{-1}	8272.53
24	0.775	3	5.83×10^{-1}	8262.62
228.69	0.775	1	5.83×10^{-1}	8245.19
24	0.775	1	5.83×10^{-1}	8014.48
228.69	12	3	5.83×10^{-1}	7987.11
228.69	12	1	5.83×10^{-1}	7860.66
24	12	3	5.83×10^{-1}	6976.23
24	12	1	5.83×10^{-1}	5515.52

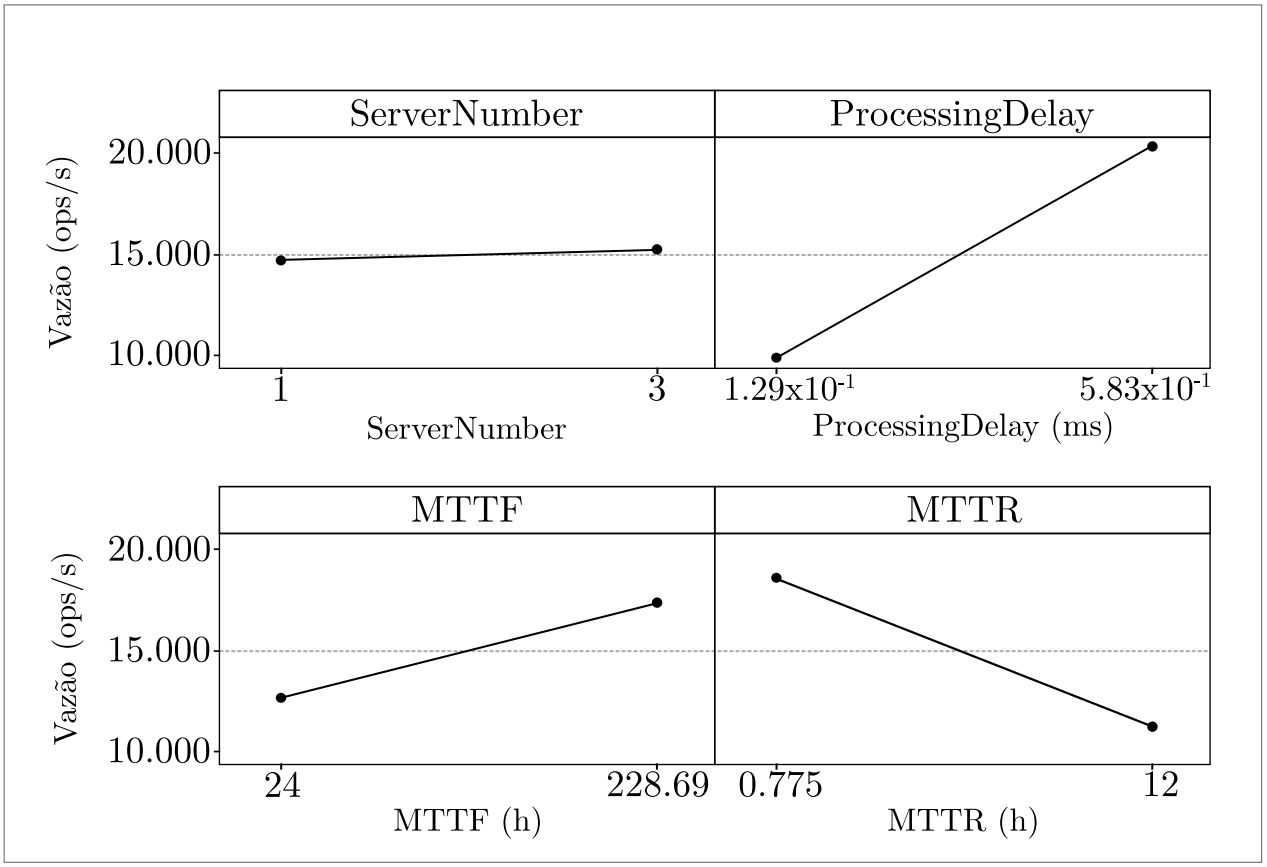
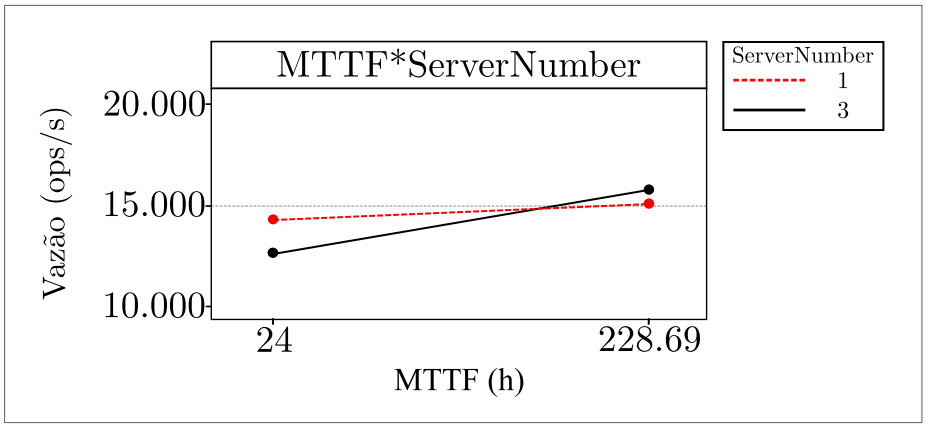
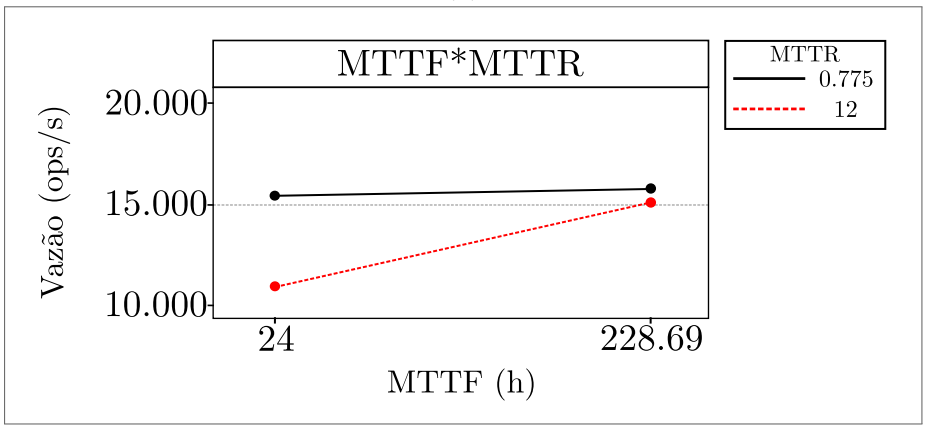


Figura 30 – Experimento 3 - Gráficos de efeito principal.



(a)



(b)

Figura 31 – Experimento 3 - Gráfico de interação.

Criar um modelo para representar um replica set particionado adiciona grande complexidade ao problema, devido o fato de exigir a representação de muitos tipos nós com diferentes funções, o algoritmo de distribuição dos dados de forma a equilibrar carga, etc. O modelo construído que representa o replica set mostra que a replicação dos dados possui um grande impacto na disponibilidade e baixa relevância na vazão do sistema.

5.3 CONSIDERAÇÕES FINAIS

Nesse capítulo foram apresentados os experimentos e seus resultados, que servem para alcançar alguns dos objetivos definidos no início do trabalho. Um desses objetivos era avaliar a disponibilidade de um SGBD NoSQL, dado a constatação da escassez de trabalhos que avaliam esse aspecto. Outro foco do trabalho é analisar o impacto da disponibilidade no desempenho do sistema, mais especificamente na vazão, por isso o foco do estudo no impacto da variação da disponibilidade na vazão do sistema.

Durante os experimentos, foram considerados como fatores o poder de computação das máquinas, que é expresso através do tempo necessário para processar uma requisição,

e a quantidade de conexões simultâneas que o SGDB pode estabelecer com usuários. A qualidade dos componentes de hardware e software utilizados também são fatores, e são representados pelos valores de tempo médio de falha e reparo dos nós de computação e controlador. A contribuição da redundância dos servidores também é avaliada tanto na disponibilidade, quanto no desempenho.

No experimento 2 foi estudado a disponibilidade do serviço, mostrando o efeito do número de réplicas, número de controladores e valores de MTTF e MTTR dos servidores. A importância da utilização do *replica set* fica clara no aumento do tempo de funcionamento do sistema, pois é o fator que mais impacta, diminuindo o *downtime* em aproximadamente 183 horas. Da mesma forma, os tempos médios de falha e reparo são consideravelmente significantes, contribuindo para uma variação de mais de 100 horas no funcionamento anual do serviço.

Verificado a relevância dos fatores anteriores no estudo de disponibilidade, os experimentos 1 e 3 avaliam os impactos desses fatores, além de outros, na vazão do sistema. Os fatores mais relevantes na avaliação de desempenho estão relacionados com o poder computacional das máquinas, mais especificamente, o processador. A cada nova geração esses *hardwares* evoluem, ficam mais velozes e contam com mais núcleos físicos e virtuais. E devido ao aumento do número de núcleos, outro benefício que essa evolução oferece, é o maior poder de paralelismo que pode fazer com que mais conexões possam ser gerenciadas paralelamente de maneira mais eficiente, relacionando o fator com maior efeito, o *ConnectionNumber*.

Diferente do que se poderia imaginar, os discos rígidos não obtiveram relevância significativa na etapa de medição da vazão no sistema real. Dois motivos podem ser apontados como possíveis responsáveis por isso ter acontecido, o primeiro é o fato de terem sido utilizadas poucas operações de atualização, durante a medição eram feitas 50.000 operações de atualização por execução. O segundo motivo é por causa dos sistemas NoSQL fazerem uso eficiente da memória RAM para aumentar a velocidade de armazenamento, é comum que SGBDs NoSQL armazenem primeiramente na memória RAM para depois, assincronamente, persistir a informação no disco.

Apesar de ser utilizado um ambiente pequeno e com valores experimentais, neste trabalho fica claro o efeito das falhas frequentes e da demora para reparação no desempenho do sistema. Isso fica evidente, principalmente, no experimento 3 quando são consideradas a ocorrência de falhas diárias e uma demora de até 12 horas para o reparo do sistema, causando, respectivamente, uma variação de mais de 2.500 operações por segundo e 1.669 operações por segundo. Como já dito ao longo do trabalho, o modelo criado simula o *replica set* com suas configurações padrões e portanto apenas um servidor é responsável por atender as requisições submetidas. Devido a isso, o número de servidores causa um efeito mínimo no desempenho do sistema como é relatado nos resultados do experimento 3.

Atualmente grandes empresas provedoras de serviços na nuvem informam em seus respectivos SLAs os níveis de disponibilidade garantidos para seus clientes, é comum que esses valores variem dependendo da região onde estão localizados os servidores. A localização dos servidores pode ser um ponto importante no contrato do serviço, pois latência pode ser um fator determinante em alguns tipos de aplicações. Também é comum, por parte das empresas, a definição de perfis de máquinas para facilitar aos usuários que não possuem conhecimento avançado a aquisição de um serviço que preencha suas expectativas. As configurações dos servidores geralmente possuem os perfis otimizados para armazenamento, processamento, memória, uso geral, etc. De acordo com os experimentos, o ideal para o caso apresentado são máquinas otimizadas para computação e localizadas numa região que provenha um bom nível de disponibilidade com a menor latência possível.

6 CONCLUSÃO

Utilizar serviços de armazenamento em ambientes de nuvem pública é algo comum com a popularização dos SGBDs NoSQL e DBaaS. Atualmente existem cerca de 51 sistemas de armazenamento NoSQL no mercado (GOMES; TAVARES; JUNIOR, 2016). As características desses SGBDs fazem com que se adéquem em aplicações com alta taxa de transferência de dados, informação em diferentes formatos e dados distribuídos geográficas.

Uma das contribuições deste trabalho são os modelos hierárquicos para avaliação de forma conjunta disponibilidade e desempenho do SGBD MongoDB. Através dos modelos é feita uma análise do impacto de diferentes fatores no funcionamento do sistema. A validação do modelo só foi possível com implementação de uma nuvem privada utilizando o *framework* Openstack para realizar medições no sistema real afim de parametrizar o modelo GSPN.

Foram realizados ao todo três experimentos que avaliaram a vazão, a disponibilidade e o impacto da disponibilidade na vazão do sistema em diferentes configurações. O primeiro experimento teve o objetivo de avaliar o impacto no desempenho do sistema considerando fatores como o número de conexões simultâneas ao banco de dados, o tempo necessário para o sistema processar uma requisição, e os tempo médio de falha e restauração dos servidores. Nos resultados foi visto que o número de conexões é o fator com maior impacto, causando uma diferença de mais 8.000 operações por segundo com a variação dos níveis. O segundo fator com maior relevância foi o tempo de processamento. No segundo experimento foi avaliada a disponibilidade e *downtime* do sistema considerando diferentes números de servidores e controladores, além dos MTTF e MTTR dos servidores. A redundância nos servidores com as cópias dos dados foi fator mais importante para garantir a disponibilidade do serviço.

No terceiro e último experimento verificou o impacto dos causado no desempenho pelo número de servidores responsáveis por aumentar a disponibilidade no serviço de armazenamento. Foi comprovado que a estrutura de replicação do MongoDB, o replica set, não melhora a vazão do sistema. Além do número de servidores, nesse experimento também foram considerados como fatores o tempo de processamento, os MTTF e MTTR dos servidores. Comparando os resultados com os do primeiro experimento podemos observar a importância das falhas e reparo no desempenho, isto está relacionado com a qualidade dos componentes e a eficiência de sua manutenção.

6.1 CONTRIBUIÇÕES

Como resultado do estudo que foi apresentado nesta dissertação, as seguintes contribuições podem ser destacadas:

- Este trabalho propôs uma avaliação conjunta do desempenho e disponibilidade de sistemas de computação em nuvem privada adotando o SGBD NoSQL como o subsistema de armazenamento. A abordagem é baseada em redes de Petri estocásticas generalizadas (GSPN) e os modelos permitem a estimativa de taxa de transferência (vazão) e disponibilidade como métricas proeminentes de QoS para avaliar sistemas de armazenamento distribuído.

- Foram criados modelos RBD para estimar os tempos médios de falha e reparo do sistema de nuvem privada baseado nos valores de falha e reparo dos principais componentes que formam o sistema.

- Para atingir os objetivos do trabalho foi formulada uma metologia que passa por diversas etapas, começando com a escolha do sistema que será utilizado para servir de base para a construção do modelos GSPN. Os experimentos no sistema real servem para fornecer dados nos quais as taxas do modelo GSPN serão baseadas. Após ter mãos o modelo com as taxas, é necessário comprovar que o modelo representa fielmente o comportamento do sistema em termos de vazão, esta etapa é chamada de validação. Com o modelo validado, foram projetados experimentos para averiguar as métricas de interesse do estudo em diversos cenários de funcionamento.

- Na análise dos resultados obtidos nos experimentos com modelos foram criados rankings que organizavam os resultados destacando os componentes que mais influenciaram na obtenção dos resultados. O valor utilizado para ordenar o ranking é chamado de efeito e expressa o impacto de um fator nos resultados do experimento.

- Conclusões sobre o impacto de diferentes componentes no funcionamento do serviço de armazenamento. Através dos modelos são possíveis a alteração de parâmetros como, o tempo entre as chegadas de requisições, o tempo para o processamento de operação pelo banco de dados, a quantidade de usuário simultâneos conectados, o número de controladores e servidores da nuvem.

Além da contribuição mencionada, um artigo que apresenta os resultados desta dissertação foi produzido e está para ser publicado na *SYSCON 2019 (The 13th Annual IEEE International Systems Conference)*:

- *Matheus Rodrigues, Breno Vasconcelos, Carlos Gomes and Eduardo Tavares, "Evaluation of NoSQL DBMS in private cloud environment: An Approach Based on Stochastic Modeling".*

6.2 LIMITAÇÕES E TRABALHOS FUTUROS

Os modelos apresentados na dissertação possuem algumas limitações, como o fato de representar apenas um SGBD NoSQL e não diferenciar os tipos de operações, que possuem

características diferentes em termos de utilização de hardware. Além disso, durante a validação não foi considerado a utilização de rede e disco. Implementação de melhorias para lidar com essas limitações contribuem para o refinamento dos modelos de maneira que forneça mais detalhes do funcionamento dos sistemas e sejam capazes de oferecer uma ferramenta de avaliação mais poderosa.

A partir do estudo atual, é possível abordar diferentes aspectos de QoS através da evolução dos modelo criados. Incrementar os modelos com a representação de novos componentes ou processos do sistema permitem que sejam calculados novas métricas de desempenho como o tempo de resposta, a taxa de descarte, o tempo de espera para um requisição ser atendida e a utilização dos componentes. Evoluir a modelagem para representação da divisão da carga de trabalho entre vários servidores faz com que seja possível a simulação de um SGBD com a base de dados distribuída. O particionamento da base de dados entre diversos servidores é uma das maneiras de aumentar o desempenho através da divisão do trabalho para evitar a sobrecarga de apenas uma das máquinas.

Como foi visto na subseção 4.7.8, espaço de estados, a explosão do espaço de estados é um problema que os modelos atuais apresentam. Então são necessários estudos e técnicas para contornar as condições atuais, tanto para o aumento do modelo com a adição de novos lugares e transições, como na representação de recursos.

REFERÊNCIAS

ABRAMOVA, V.; BERNARDINO, J. Nosql databases: Mongodb vs cassandra. In: *Proceedings of the International C* Conference on Computer Science and Software Engineering*. New York, NY, USA: ACM, 2013. (C3S2E '13), p. 14–22. ISBN 978-1-4503-1976-8. Disponível em: <http://doi.acm.org/10.1145/2494444.2494447>.

BANKER PETER BAKKUM, S. V. D. G. T. H. K. *MongoDB in Action: Covers MongoDB version 3.0*. [S.l.]: Manning, 2016.

BARDSIRI, A. K.; HASHEMI, S. M. Qos metrics for cloud computing services evaluation. *International Journal of Intelligent Systems and Applications*, Modern Education and Computer Science Press, v. 6, n. 12, p. 27, 2014.

BARROS, J.; CALLOU, G.; GONÇALVES, G. Análise integrada de desempenho e consumo de energia em sistemas de armazenamento de dados distribuídos. In: SBC. *Anais do XV Workshop em Clouds e Aplicações (WCGA-SBRC 2017)*. [S.l.], 2017. v. 15, n. 1/2017.

BOLCH, G.; GREINER, S.; MEER, H. D.; TRIVEDI, K. S. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. [S.l.]: John Wiley & Sons, 2006.

BRUNEO, D. A stochastic model to investigate data center performance and qos in iaas cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, v. 25, n. 3, p. 560–569, 2014. ISSN 1045-9219.

CAI, L.; HUANG, S.; CHEN, L.; ZHENG, Y. Performance analysis and testing of hbase based on its architecture. In: *2013 IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS)*. Niigata, Japan: IEEE, 2013. p. 353–358.

CATTELL, R. Scalable sql and nosql data stores. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 39, n. 4, p. 12–27, maio 2011. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/1978915.1978919>.

CHALKIADAKI, M.; MAGOUTIS, K. Managing service performance in nosql distributed storage systems. In: *7th Workshop on Middleware for Next Generation Internet Computing*. [S.l.]: ACM, 2012. (MW4NG '12). ISBN 978-1-4503-1607-1.

CHANG, F.; DEAN, J.; GHEMAWAT, S.; HSIEH, W. C.; WALLACH, D. A.; BURROWS, M.; CHANDRA, T.; FIKES, A.; GRUBER, R. E. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, ACM, New York, NY, USA, v. 26, n. 2, p. 4:1–4:26, jun. 2008. ISSN 0734-2071. Disponível em: <http://doi.acm.org/10.1145/1365815.1365816>.

CHEN, C. P.; ZHANG, C.-Y. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, v. 275, p. 314 – 347, 2014. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025514000346>.

CHEN, M.; MAO, S.; ZHANG, Y.; LEUNG, V. *Big Data: Related Technologies, Challenges and Future Prospects*. Springer International Publishing, 2014. (SpringerBriefs in Computer Science). ISBN 9783319062457. Disponível em: <https://books.google.com.br/books?id=0wwqBAAAQBAJ>.

COOPER, B. F. *Yahoo! Cloud System Benchmark (YCSB)*. 2018. Disponível em: <https://github.com/brianfrankcooper/YCSB>.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking cloud serving systems with ycsb. In: *Proceedings of the 1st ACM Symposium on Cloud Computing*. New York, NY, USA: ACM, 2010. (SoCC '10), p. 143–154. ISBN 978-1-4503-0036-0. Disponível em: <http://doi.acm.org/10.1145/1807128.1807152>.

DAVOUDIAN, A.; CHEN, L.; LIU, M. A survey on nosql stores. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 51, n. 2, p. 40:1–40:43, abr. 2018. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/3158661>.

DECANDIA, G.; HASTORUN, D.; JAMPANI, M.; KAKULAPATI, G.; LAKSHMAN, A.; PILCHIN, A.; SIVASUBRAMANIAN, S.; VOSSHALL, P.; VOGELS, W. Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 41, n. 6, p. 205–220, out. 2007. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/1323293.1294281>.

DEKA, G. C. A survey of cloud database systems. *IT Professional*, v. 16, n. 2, p. 50–57, 2014. ISSN 1520-9202.

DIPIETRO, S.; CASALE, G.; SERAZZI, G. A queueing network model for performance prediction of apache cassandra. In: *10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*. [S.l.]: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2017. ISBN 978-1-63190-141-6.

GAIKWAD, C.; CHURI, B.; PATIL, K.; TATWADARSHI, P. N. Providing storage as a service on cloud using openstack. In: *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. Coimbatore, India: IEEE, 2017. p. 1–4.

GANDINI, A.; GRIBAUDO, M.; KNOTTENBELT, W. J.; OSMAN, R.; PIAZZOLLA, P. Performance evaluation of nosql databases. In: *Computer Performance Engineering*. [S.l.]: Springer International Publishing, 2014. ISBN 978-3-319-10885-8.

GERMAN, R. A concept for the modular description of stochastic petri nets (extended abstract). In: *Proc. 3rd Int. Workshop on Performability Modeling of Computer and Communication Systems*. Bloomingdale, IL, USA: [s.n.], 1996. p. 20–24.

GESSERT, F.; WINGERATH, W.; FRIEDRICH, S.; RITTER, N. Nosql database systems: a survey and decision guidance. *Computer Science - Research and Development*, v. 32, n. 3, p. 353–365, Jul 2017. ISSN 1865-2042. Disponível em: <https://doi.org/10.1007/s00450-016-0334-3>.

GHOSH, R.; TRIVEDI, K. S.; NAIK, V. K.; KIM, D. S. End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach. In: *2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing*. [S.l.: s.n.], 2010.

GOMES, C.; TAVARES, E. A. G.; JUNIOR, M. O. de N. Energy consumption evaluation of nosql dbmss. In: *15º WPerformance - Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. Porto Alegre, RS, Brasil: [s.n.], 2016.

GONZáLEZ-MARTíNEZ, J. A.; BOTE-LORENZO, M. L.; GóMEZ-SáNCHEZ, E.; CANO-PARRA, R. Cloud computing and education: A state-of-the-art survey. *Computers & Education*, v. 80, p. 132 – 151, 2015. ISSN 0360-1315. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360131514001985>.

GUDIVADA, V. N.; RAO, D.; RAGHAVAN, V. V. Nosql systems for big data management. In: *2014 IEEE World Congress on Services*. Anchorage, AK, USA: IEEE, 2014. p. 190–197. ISSN 2378-3818.

GUIMARãES, A.; MACIEL, P.; JR, R. M.; CAMBOIM, K. Dependability analysis in redundant communication networks using reliability importance. *Information and Network Technology*, v. 4, p. 12–17, 2011.

HECHT, R.; JABLONSKI, S. Nosql evaluation: A use case oriented survey. In: *2011 International Conference on Cloud and Service Computing*. Hong Kong, China: IEEE, 2011. p. 336–341.

JAIN, R. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley, 1991. I-XXVII, 1-685 p. (Wiley professional computing). ISBN 978-0-471-50336-1.

JAYATHILAKE, D.; SOORIAARACHCHI, C.; GUNAWARDENA, T.; KULASURIYA, B.; DAYARATNE, T. A study into the capabilities of nosql databases in handling a highly heterogeneous tree. In: *2012 IEEE 6th International Conference on Information and Automation for Sustainability*. Beijing, China: IEEE, 2012. p. 106–111. ISSN 2151-1802.

JUNIOR, G. A. de A. *Avaliação de desempenho de cadeias de suprimentos utilizando componentes GSPN*. Dissertação (Mestrado) — Universisdade Federal de Pernambuco, Recife, Brasil, 2007.

KIRSAL, Y.; EVER, Y. K.; MOSTARDA, L.; GEMIKONAKLI, O. Analytical modelling and performability analysis for cloud computing using queuing system. In: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. Limassol, Cyprus: IEEE, 2015.

KLEIN, J.; GORTON, I.; ERNST, N.; DONOHOE, P.; PHAM, K.; MATSER, C. Performance evaluation of nosql databases: A case study. In: *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*. New York, NY, USA: ACM, 2015. (PABS '15), p. 5–10. ISBN 978-1-4503-3338-2. Disponível em: <http://doi.acm.org/10.1145/2694730.2694731>.

KUO, W.; ZUO, M. J. *Optimal reliability modeling: principles and applications*. [S.l.]: John Wiley & Sons, 2003.

LILJA, D. J. *Measuring computer performance: a practitioner's guide*. [S.l.]: Cambridge university press, 2005.

LIMA, M. A. de Q.; MACIEL, P. R.; SILVA, B.; GUIMARãES, A. P. Performability evaluation of emergency call center. *Performance Evaluation*, v. 80, p. 27 – 42, 2014. ISSN 0166-5316. 'SI: Service Science of Queues.

LIRA, V.; TAVARES, E.; MACIEL, P. An automated approach to dependability evaluation of virtual networks. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 88, n. C, p. 89–102, set. 2015. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2015.05.016>.

MACIEL, P. R. M.; LINS, R. D.; CUNHA, P. R. F. *Introdução às Redes de Petri e Aplicações*. Campinas, São Paulo, Brasil: X Escola de Computação, 1996.

MARSAN, M.; BALBO, G.; CHIOLA, G.; CONTE, G.; DONATELLI, S.; FRANCESCHINIS, G. An introduction to generalized stochastic petri nets. *Microelectronics Reliability*, v. 31, n. 4, p. 699 – 725, 1991. ISSN 0026-2714. Disponível em: <http://www.sciencedirect.com/science/article/pii/0026271491900105>.

MCCREARY, D.; KELLY, A. *Making Sense of NoSQL: A Guide for Managers and the Rest of Us*. Manning, 2013. ISBN 9781617291074. Disponível em: <https://books.google.com.br/books?id=DegwmgEACAAJ>.

MELO, C.; DANTAS, J.; OLIVEIRA, A.; OLIVEIRA, D.; Fé, I.; ARAUJO, J.; MATOS, R.; MACIEL, P. Availability models for hyper-converged cloud computing infrastructures. In: *2018 Annual IEEE International Systems Conference (SysCon)*. [S.l.: s.n.], 2018. p. 1–7. ISSN 2472-9647.

MONTGOMERY, D. *Design and Analysis of Experiments*. 8. ed. [S.l.]: John Wiley & Sons, 2008. ISBN 9780470128664.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989.

OEHLERT, G. W. *A first course in design and analysis of experiments*. [S.l.]: W. H. Freeman, 2010.

OPENSTACK. *OpenStack Cloud Software*. 2018. Disponível em: <https://www.openstack.org/software/>.

OSMAN, R.; KNOTTENBELT, W. J. Database system performance evaluation models: A survey. *Performance Evaluation*, v. 69, n. 10, p. 471 – 493, 2012. ISSN 0166-5316.

OSMAN, R.; PIAZZOLLA, P. Modelling replication in nosql datastores. In: *Quantitative Evaluation of Systems*. [S.l.]: Springer International Publishing, 2014. ISBN 978-3-319-10696-0.

RAJARAMAN, V. Big data analytics. *Resonance*, Springer, v. 21, n. 8, p. 695–716, 2016.

SAHNER, R.; TRIVEDI, K.; PULIAFITO, A. Performance and reliability analysis of computer systems (an example-based approach using the sharpe software. *IEEE Transactions on Reliability*, IEEE, v. 46, n. 3, p. 441–441, 1997.

SILVA, B.; AL. et. Astro: An integrated environment for dependability and sustainability evaluation. *Sustainable Computing: Informatics and Systems*, v. 3, n. 1, p. 1 – 17, 2013. ISSN 2210-5379.

SILVA, B.; MACIEL, P.; ZIMMERMANN, A. Performability models for designing disaster tolerant infrastructure-as-a-service cloud computing systems. In: *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*. London, UK: [s.n.], 2013. p. 647–652.

TANG, E.; FAN, Y. Performance comparison between five nosql databases. *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, p. 105–109, 2016.

TORRES, E.; CALLOU, G.; ANDRADE, E. A hierarchical approach for availability and performance analysis of private cloud storage services. *Computing*, v. 100, n. 6, p. 621–644, Jun 2018. ISSN 1436-5057. Disponível em: <https://doi.org/10.1007/s00607-018-0588-7>.

TRIVEDI, K. S.; HUNTER, S.; GARG, S.; FRICKS, R. *Reliability Analysis Techniques Explored Through a Communication Network Example*. 1996.

VALMARI, A. The state explosion problem. In: _____. *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 429–528. ISBN 978-3-540-49442-3. Disponível em: <https://doi.org/10.1007/3-540-65306-6_21>.

VENTURA, L.; ANTUNES, N. Experimental assessment of nosql databases dependability. In: *2016 12th European Dependable Computing Conference (EDCC)*. Gothenburg, Sweden: IEEE, 2016.

WANG, T.; CHANG, X.; LIU, B. Performability analysis for iaas cloud data center. In: *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. Guangzhou, China: IEEE, 2016. p. 91–94.

ZHAO, L.; SAKR, S.; LIU, A. A framework for consumer-centric sla management of cloud-hosted databases. *IEEE Transactions on Services Computing*, v. 8, n. 4, p. 534–549, July 2015. ISSN 1939-1374.

ZIMMERMANN, A.; FREIHEIT, J.; GERMAN, R.; HOMMEL, G. Petri net modelling and performability evaluation with timenet 3.0. In: SPRINGER. *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. [S.l.], 2000. p. 188–202.