Universidade Federal Rural de Pernambuco

Programa de Pós-Graduação em Informática Aplicada

Modelos para Avaliação de Desempenho e Disponibilidade de Servidores de Arquivos em Nuvem Privada

Alison Vinicius Gomes da Silva

Recife

Agosto de 2023

Alison Vinicius Gomes da Silva

Modelos para Avaliação de Desempenho e Disponibilidade de Servidores de Arquivos em Nuvem Privada

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Recife

Agosto de 2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586m Silva, Alison Vinicius Gomes da
Modelos para Avaliação de Desempenho e Disponibilidade de Servidores de Arquivos em Nuvem Privada / Alison Vinicius Gomes da Silva. - 2023.
78 f. : il.

Orientador: Gustavo Rau de Almeida Callou.
Inclui referências e apêndice(s).

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2023.

1. Desempenho. 2. Disponibilidade. 3. Computação em Nuvem. 4. Servidor de Arquivos. 5. Rede de Petri Estocástica. I. Callou, Gustavo Rau de Almeida, orient. II. Título

CDD 004

Alison Vinicius Gomes da Silva

Modelos para Avaliação de Desempenho e Disponibilidade de Servidores de Arquivos em Nuvem Privada

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

BANCA EXAMINADORA

Prof. Dr. Gustavo Rau de Almeida Callou
Universidade Federal Rural de Pernambuco

Prof. Dr. Fernando Antonio Aires Lins
Universidade Federal Rural de Pernambuco

Prof. Dr. Jamilson Ramalho Dantas
Universidade Federal de Pernambuco

Agradecimentos

Primeiramente queria agradecer a toda minha família. Em especial a minha filha Clara, minha esposa Stephanie, minha mãe Edilene, meu pai Reginaldo e minha irmã Letícia por sempre estar ao meu lado sempre.

Agradeço ao professor Gustavo Callou pela oportunidade e a apoio na condução deste trabalho.

Agradeço também aos colegas do grupo Pedal, especialmente a Thiago Valentim e Ivson Borges por auxiliar na condução dessa pesquisa.

Resumo

A necessidade de serviços em nuvem vem aumentando nos últimos anos, onde usuários da Internet passaram a consumir cada vez mais serviços hospedados na nuvem, devido a serviços online, redes sociais e *e-commerce*. Exemplos de serviços de nuvens que podem ser destacados são: Microsoft Azure, Amazon Web Services (AWS), Google Cloud e Apache Cloudstack. Estes serviços demandam disponibilidade e desempenho para garantir o seu funcionamento. Adotar a computação em nuvem com um sistema de arquivos para fornecer dinamicamente recursos é uma das possíveis soluções para esse problema. No entanto, o grande aumento na utilização desse tipo de serviço pode gerar experiências negativas para os usuários caso os recursos necessários não tenham um desempenho satisfatório. Portanto, esse trabalho realiza uma análise de desempenho e disponibilidade de um sistema de arquivos, Nextcloud, hospedado em uma nuvem privada. Foram propostos modelos em rede estocástica de Petri (SPN) e diagrama de bloco de Confiabilidade (RBD) para avaliar o desempenho e a disponibilidade. O primeiro estudo de caso valida o modelo de desempenho em SPN proposto. O segundo estudo de caso realiza projeções com uma quantidade maior de clientes. O terceiro estudo de caso é a validação do modelo, onde se utiliza o resultado da injeção de falhas para se calcular o intervalo de confiança e compara com o resultado da análise do modelo. O quarto estudo de caso propôs duas novas arquiteturas com redundância com o objetivo de melhorar a disponibilidade do serviço. O quinto estudo realiza uma comparação entre três tipos de acordo de nível de serviço (SLA). O sexto e último estudo realiza uma comparação entre três tipos de equipamentos. Os resultados obtidos demonstram que foi possível obter uma diminuição de quase 85% no downtime em relação à arquitetura base.

Palavras-chave: Desempenho, Disponibilidade, Computação em Nuvem, Servidor de Arquivos, Rede de Petri Estocástica, Diagrama de Bloco de Confiabilidade

Abstract

The need for cloud services has been increasing in recent years, where Internet users have increasingly consumed services hosted in the cloud, due to online services, social networks and e-commerce. Examples of cloud services that can be highlighted are Microsoft Azure, Amazon Web Services (AWS), Google Cloud and Apache Cloudstack. Therefore, there is a growing need for computational resources that need to be offered. Adopting cloud computing with a file system to dynamically provision resources is one possible solution to this problem. However, the large increase in the use of this type of service can generate negative experiences for users if the necessary resources do not perform satisfactorily. Therefore, this work performs a performance and availability analysis of a file system, Nextcloud, hosted in a private cloud. Stochastic Petri Net (SPN) and Reliability Block Diagram (RBD) models were proposed to evaluate performance and availability. The first case study validates the proposed SPN performance model. The second case study performs projections with a larger number of clients. The third case study is model validation, where the fault injection result is used to calculate the confidence interval and compares it with the model analysis result. The fourth case study proposed two new architectures with redundancy in order to improve service availability. The fifth study compares three types of service level agreement (SLA). The sixth and final study compares three types of equipment. The obtained results help designers to estimate the necessary configurations to support the system in different situations. The results obtained show a reduction of almost 85% in downtime concerning the base architecture.

Keywords: Performance, Availability, Cloud Computing, File Server, Stochastic Petri Net, Reliability Block Diagram.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de acrônimos

API	Application Programming Interface
AWS	Amazon Web Services
COA	Capacity-Oriented Availability
CPVM	Console Proxy Virtual Machine
CTMC	Continuous Time Markov Chain
CPU	Central Processing Unit
FPGA	Field Programmable Gate Arrays
IAAS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
MRT	Mean Response Time
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
NIST	National Institute of Standards and Technology
PAAS	Platform as a Service
PN	Petri Nets
RBD	Reliability Block Diagram
SAAS	Software as a Service
SLA	Service Level Agreement

SPN	Stochastic Petri Net
SRN	Stochastic Reward Nets
SSVM	Secondary Storage Virtual Machine
VHDL	VHSIC Hardware Description Language
VR	Virtual Router

Capítulo 1

Introdução

Cada vez mais as pessoas estão usando aplicações por meio da Internet, como os serviços de armazenamento, exemplo o Drive e Dropbox ou usando ferramentas de edição de texto como o Microsoft 365 e Google Documentos, entre outros. Com isso as empresas estão cada vez mais investindo em poder de processamento e recursos computacionais. E como solução, o paradigma de computação em nuvem está sendo usado para compartilhar recursos computacionais, bem como software e aplicativos em geral (DANTAS et al., 2020).

A computação em nuvem utiliza técnicas de virtualização onde usuários podem fazer uso de máquinas virtuais com configurações quaisquer, dependendo dos recursos computacionais disponíveis nas máquinas físicas (ARMBRUST et al., 2010). As suas principais características são rápidas elasticidade, alta disponibilidade, confiabilidade e baixo custo (BAUER; ADAMS, 2012). Atualmente, existem várias empresas que fornecem esse tipo de serviço como a Amazon AWS, Microsoft Azure e Google Cloud (AMAZON, 2022; MICROSOFT, 2023; GOOGLE, 2023a).

Estas empresas têm que garantir por meio de acordos de níveis de serviços (SLA) em fornecer alta disponibilidade, desempenho, segurança e confiabilidade (TORRES et al., 2016). Mesmo assim, grandes empresas não estão isentas de falhas, exemplo a Amazon que enfrentou uma grande queda no serviço de nuvem e deixou vários sites e aplicativos fora do ar (VINHA, 2021; VENTURA, 2020). Então, avaliações de desempenho e disponibilidade são atividades essenciais para promover a melhoria da qualidade do serviço prestado e para o

planejamento de infraestruturas de sistemas (AVIZIENIS; LAPRIE; RANDELL, 2001).

Atualmente, o conceito de *open source* ganhou uma maior visibilidade. Ao se adotar tais soluções, se reduz o custo com licença de *softwares* e, ainda assim, é possível realizar as personalizações conforme as necessidades de cada negócio. Exemplos dessas soluções de *software open source* para a computação em nuvem são o Apache Cloudstack e o Openstack (APACHE, 2022; OPENSTACK, 2022). Com o aumento da demanda da utilização de recursos computacionais para dar suporte tanto aos professores e aos alunos, passou a ser preciso adotar um serviço como o Nextcloud para o armazenamento dos dados. O Nextcloud é uma plataforma de colaboração de conteúdo acessível por meio de interfaces móveis, *desktop* e *Web*. Nesse ambiente, tanto alunos, como professores e pesquisadores podem colaborar em tempo real em documentos, conversar e participar de videochamadas, acessar seus e-mails e agendar reuniões em calendários.

1.1 Motivação

A pandemia do COVID-19 afetou toda a população mundial, e os seus impactos podem ser observados tanto nas universidades como em empresas que passaram a ter que incorporar, rapidamente, o conceito do trabalho *home office*. Devido a esse cenário, houve um aumento da demanda por recursos computacionais (ex., armazenamento e processamento) para dar suporte a tal modalidade de trabalho. E, nesse sentido, a computação em nuvem se apresenta como uma grande aliada para a mitigação desses problemas por oferecer infraestrutura, plataforma e software como serviço. Além de suas principais características como elasticidade e escalabilidade.

No cenário em que empresas e órgãos governamentais tiveram que se adaptar a essa nova realidade, acelerada em virtude da COVID-19, investimentos foram conduzidos na área da tecnologia da informação a fim de suprir essa demanda não esperada. Com isso, empresas que fornecem esses recursos estão ficando sobrecarregadas e estão limitando quantidade para os seus clientes, como o Google que está fornecendo 100 terabytes para ser dividido entre todos os usuários de uma única instituição (PALMEIRA, 2021; LOPES, 2023). Além disso, é importante destacar que tanto governos como empresas estão se afastando dos provedores

de nuvens públicas em meio a crescentes preocupações sobre a soberania e segurança digital dos dados, já que o Brasil sofreu em 2022 mais de 103 bilhões de tentativas de ataques cibernéticos (AMORIM, 2023).

Esse trabalho tem como foco a avaliação de desempenho e disponibilidade do Nextcloud hospedado em um ambiente de nuvem privada com o Apache Cloudstack. Os cenários avaliados têm como objetivo o de atestar a qualidade e a quantidade de acessos simultâneos do serviço oferecido. Além disso, esse estudo sobre o desempenho e disponibilidade desse tipo de sistema auxiliará os projetistas a estimar as configurações demandas para dar suporte ao sistema em diferentes situações.

1.2 Objetivos

O objetivo principal deste trabalho é propor uma estratégia baseada em modelos estocásticos para avaliação de desempenho e da disponibilidade de servidor de arquivos em nuvem privada. Para atingir o objetivo principal foram realizados os seguintes objetivos específicos:

- Propor uma estratégia para medir o desempenho do servidor de arquivos através de um conjunto métricas como vazão e tempo de resposta.
- Propor modelos de desempenho para representar o servidor de arquivos em nuvem privada.
- Propor modelos de disponibilidade para representar o cenário em análise.
- Desenvolver um *script* de injeção de falha e reparo, para auxiliar na validação do modelo de disponibilidade.
- Realizar projeções tanto com os modelos propostos como a partir de regressões lineares para fornecer estimativas em cenários complexos.

1.3 Estrutura da dissertação

O Capítulo 2 apresenta os trabalhos relacionados a essa dissertação. O Capítulo 3 apresenta a fundamentação teórica abordando os seguintes assuntos: computação em nuvem, avaliação de desempenho, redes de Petri, regressão linear, disponibilidade, diagrama de blocos de confiabilidade, injeção de falhas e o método de Keesee. O Capítulo 4 apresenta a metodologia usada no trabalho e o ambiente de teste. O Capítulo 5 apresenta os modelos de desempenho e disponibilidade propostos. O Capítulo 6 descreve os estudos de caso utilizados nesta monografia. E por fim, o último capítulo apresenta a conclusão, contribuições e trabalhos futuros dessa pesquisa.

Capítulo 2

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados a análise de desempenho e disponibilidade de servidores de arquivos em computação em nuvem. Os trabalhos apresentados neste capítulo possuem relação com este trabalho. A maioria destes trabalhos focam em modelagem em rede de Petri estocástica (SPN) e em blocos de diagrama de confiabilidade (RBD). Apenas dois trabalhos focam simultaneamente em desempenho e disponibilidade. A maioria também é relacionada a computação em nuvem, onde alguns usam o Eucalyptus, outros usam o Cloudstack e Openstack e apenas um utiliza a nuvem pública da *Amazon Web Services* (AWS). Ao final desse capítulo é feito um comparativo entre os trabalhos relacionados e o trabalho proposto.

Em (MOREIRA et al., 2021), os autores avaliaram o Nextcloud em uma nuvem privada com o Openstack, e usaram o GlusterFS como ambiente de armazenamento. Foram realizados testes de desempenho na leitura e escrita de arquivos, utilizando o *benchmarks* IOzone. Os testes foram divididos nos cenários: com dois tamanhos de arquivos de 100 MB e 500 MB e teste com e sem o VPN Wireguard. Também foi feito análises de segurança nas transferências e armazenamentos dos dados. Foi concluído que uma solução de armazenamento distribuído em uma IaaS com um serviço do Nextcloud se mostrou eficaz e seguro.

Santos et al. (SANTOS; CAMBOIM; ALENCAR, 2018) realizaram um estudo comparativo do desempenho da CPU, memória e disco do Nextcloud, Owncloud e Pydio. Os testes foram divididos em dois cenários: (i) foram realizados dois *uploads* de arquivos com

tamanhos distintos e, em (ii) foi realizada uma transmissão de um vídeo de duração de três minutos e meio. Para coletar as métricas de interesse foi utilizado a ferramenta Nmon. Como conclusão, foi reportado que o Nextcloud teve melhor desempenho para *uploads* e o Owncloud para transmissão de vídeos.

Shahzadi et al. (SHAHZADI et al., 2017) realizaram testes comparativos de desempenho e escalabilidade entre as nuvens Openstack e Devstack utilizando o *benchmark* Rally. Foram realizados testes em três cenários: (i) 100 iterações nos serviços de Keystone, (ii) 100 iterações nos serviços de Glance e (iii) 100 iterações nos serviços do Nova. Em todos os cenários foram coletados a métrica de tempo de resposta. Os autores reportaram como conclusão do estudo realizado que o Openstack, no geral, acaba possuindo um melhor desempenho em relação à nuvem privada com o Devstack.

Os autores em (VOGEL et al., 2017) avaliaram o desempenho da infraestrutura do Cloudstack utilizando as métricas de tempo de resposta e latência da rede na criação de instâncias. Para a realização desse estudo e quantificação das métricas de interesse, os autores utilizaram os *benchmarks* NetPIPE e Uperf. Foi concluído que o KVM teve uma taxa de transferência próxima ao ambiente nativo e o LXC apresentou uma baixa latência, mas apresenta limitações na compatibilidade com a nuvem.

Lima et al. (LIMA et al., 2021) avaliaram o desempenho e o consumo de energia de um ambiente virtual de aprendizagem hospedado no Apache Cloudstack e no Openstack. O ambiente foi estressado para quantificação de métricas de tempo de resposta, vazão, utilização de Cpu e memória e consumo de energia. Também foi construído um modelo em SPN para representar o cenário. Como conclusão, foi reportado que o Cloudstack obteve um desempenho melhor com um menor consumo de energia.

Torres et al. (TORRES et al., 2016) propuseram modelos em SPN e RBD para uma avaliação integrada de métricas de disponibilidade e desempenho em um serviço de armazenamento em uma nuvem privada gerenciada pelo ambiente Eucalyptus. Os autores utilizaram para as medições as ferramentas JMeter e Apache benchmarking para validação do modelo. Em seus testes foram utilizadas métricas de vazão, utilização do sistema, disponibilidade, *downtime* e *uptime*. Os resultados obtidos no teste de disponibilidade em um cenário sem replicação foi de 98,46% e com replicação foi de 99,99%, já no teste de desempenho se

observou uma taxa de utilização de 95%.

Os autores em (PINHEIRO et al., 2019) propuseram uma avaliação de desempenho e custos relacionados a nuvem pública. Foi utilizado como modelagem a rede de Petri estocástica para representar sistemas elásticos. Foi utilizado a métrica de *Mean Responste Time* (MRT) para relacionar com os custos. Foi concluído que o modelo em SPN consegue avaliar o desempenho e o custo referentes ao uso da máquina virtual (VM) em nuvens públicas.

Matos et al. (MATOS et al., 2020) propuseram modelos hierárquicos para detecção de gargalos no ambiente de nuvem privada. As métricas utilizadas nessa pesquisa foram disponibilidade, utilização da VM, tempo de resposta e vazão. Foi concluído que a análise indicou os gargalos em dois serviços web e no balanceador de carga da nuvem como os mais influentes na perca de desempenho e no tempo de resposta.

Os autores (TORQUATO; UMESH; MACIEL, 2018) propuseram dois modelos de disponibilidade em SPN para avaliar os efeitos de rejuvenescimento de *software* por meio migração ao vivo de VMs e foram utilizados dois esquemas de redundância que são o *Warm-Standby* e o *Cold-Standby*. Eles também realizaram uma análise do consumo de energia dos processos. Os resultados mostraram uma melhora significativa da disponibilidade, tendo uma redução anual de 164 horas na indisponibilidade. Foi concluído que abordagem *Cold-Standby* tem um resultado melhor devido ao número total de migrações ao vivo e um consumo de energia mais eficiente. Em todos os cenários o *Cold-Standby* teve uma economia de 40% nos custos.

Os autores (MELO et al., 2017) propuseram modelos em SPN e RBD para avaliar a capacidade de um nó em um ambiente de computação em nuvem com base quantidade de recursos de *hardware* disponíveis. Foram utilizadas métricas de disponibilidade, *downtime*, *uptime* e *capacity-oriented availability* (COA). Também foi realizado uma análise de sensibilidade para determinar qual componente teve maior impacto na métrica de interesse. Como conclusão, os resultados mostraram uma disponibilidade de 99,27% e o *downtime* do subsistema ficou em 16 horas anuais. Também foi constatado uma degradação na criação das VMs devido a falhas.

Costa et al. (COSTA et al., 2016) avaliaram a disponibilidade da plataforma OpenMobs-

ter e propôs modelos em CTMC e RBD. Os modelos foram projetados a partir de medições e realizando o teste de injeção de falhas. Também foi feito uma análise de sensibilidade para verificar qual componente causa maior impacto na disponibilidade. Os resultados mostraram que teve um aumento na disponibilidade de 89,69% para 90,70% e o tempo de inatividade anual foi reduzido em 10%

Dantas et al. (DANTAS et al., 2020) apresentaram uma estratégia para avaliar a disponibilidade do *capacity-oriented availability* (COA) em uma nuvem privada. A abordagem adotada foi comparada como modelos de cadeias de Markov de tempo contínuo (CTMC) e SPN. Os resultados apresentaram similaridades nas saídas do modelo CTMC e da equação de forma fechada. Outro resultado significativo foi o tempo gasto para realizar o cálculo do COA, que o modelo CTMC durou 76,41 segundos, enquanto na equação de forma fechada foi realizado em 0,05 segundos, sendo realizado no mesmo cenário.

Santos et al. (SANTOS et al., 2017) propuseram modelos para avaliar a disponibilidade de serviços executados em uma infraestrutura de *data center* em uma nuvem. Foi utilizado as normas do TIA-942 sendo propostos modelos de SLA nível I e IV em SPN e RBD para representar o comportamento de falha dos componentes. Também foi feita uma análise de sensibilidade para descobrir qual componente tem maior impacto na disponibilidade do *data center*. Os resultados apresentaram um aumento da disponibilidade do nível I para o nível IV.

Os autores (ROCHA et al., 2017) utilizaram um *hyperscale data center* para verificar a relação entre infraestrutura de energia e a disponibilidade de uma aplicação. Foram definidos quatro arquiteturas de infraestrutura de energia e foi realizado uma análise de sensibilidade para verificar o impacto. Também foram feitos modelos em SPN para representar a infraestrutura de energia e o modelos em RBD para representar a disponibilidade das aplicações. Os resultados mostram que se aumentar a redundância da infraestrutura de energia, se obtém um aumento na disponibilidade. Porém, haverá também um aumento nos custos.

Ataie et. al. (ATAIE et al., 2019) propuseram modelos analíticos em redes de recompensas estocásticas (SRN) para avaliar a disponibilidade, desempenho e consumo de energia de uma IaaS. O modelo captura aspectos da nuvem como diferentes *pools* das máquinas físicas, atrasos no provisionamento e comportamento de falhas e reparos das máquinas físicas. O

modelo também realiza uma estratégia de mover máquinas físicas entre *hot* e *colds pools* com objetivo de reduzir o consumo de energia. Os resultados obtidos com a aplicação dos modelos propostos apresentam que os erros introduzidos pelos modelos aproximados são insignificantes enquanto se obtém uma melhoria de várias ordens de grandeza na redução do espaço de estados do modelo monolítico.

Diferente dos trabalhos anteriores, este trabalho faz uma análise de desempenho e disponibilidade do servidor de arquivos (Nextcloud) hospedado na nuvem privada Cloudstack. Para isso, foram propostos modelos em SPN e RBD para avaliar tais métricas de interesse. A Tabela 2.1 mostra as características dos quinze trabalhos relacionados e do trabalho proposto. A tabela mostra que seis trabalhos focaram em desempenho, seis trabalhos em disponibilidade e três focaram em desempenho e disponibilidade. Apenas quatro trabalhos não utilizaram nenhuma forma de modelagem. E apenas um trabalho não utilizou ferramentas de computação em nuvem.

Tabela 2.1: Características dos Trabalhos Relacionados.

Artigos	**Desempenho**	**Disponibilidade**	**SPN**	**RBD**
(MOREIRA et al., 2021)	✓			
(SANTOS; CAMBOIM; ALENCAR, 2018)	✓			
(SHAHZADI et al., 2017)	✓			
(VOGEL et al., 2017)	✓			
(LIMA et al., 2021)	✓		✓	
(TORRES et al., 2016)	✓	✓	✓	✓
(PINHEIRO et al., 2019)	✓		✓	
(MATOS et al., 2020)	✓	✓	✓	
(TORQUATO; UMESH; MACIEL, 2018)		✓	✓	
(MELO et al., 2017)		✓	✓	✓
(COSTA et al., 2016)		✓		✓
(DANTAS et al., 2020)		✓	✓	
(SANTOS et al., 2017)		✓	✓	✓
(ROCHA et al., 2017)		✓	✓	✓
(ATAIE et al., 2019)	✓	✓		
Este Trabalho	✓	✓	✓	✓

2.1 Considerações Finais

Este capítulo apresentou os principais trabalhos relacionados ao estudo proposto. No estado da arte, vários trabalhos abordam avaliação de desempenho, disponibilidade, modelagem e computação em nuvem.

Capítulo 3

Fundamentação Teórica

Este capítulo apresenta os conceitos necessários para um melhor entendimento deste trabalho. Primeiramente, serão apresentados os conceitos sobre computação em nuvem e sobre a IaaS Apache CloudStack. Em seguida, serão abordados os principais conceitos sobre avaliação de desempenho. Em seguida, serão abordados os principais conceitos sobre redes de Petri e sua extensão, a redes de Petri estocástica. Esse capítulo também apresenta a regressão linear que é uma técnica estatística, os conceitos de dependabilidade e a modelagem por Diagramas de Bloco de Confiabilidade (RBD). E, por fim, serão apresentados as formas adotadas na injeção de falhas e o método Keesee utilizados para a validação do modelo de disponibilidade.

3.1 Computação em Nuvem

O modelo de computação em nuvem foi desenvolvido com o objetivo de fornecer serviços de fácil acesso e de baixo custo e garantir características tais como disponibilidade e escalabilidade (RUSCHEL; ZANOTTO; MOTA, 2010). A definição de computação em nuvem é um modelo para acesso ubíquo, conveniente e sob demanda a uma rede compartilhada de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços (MELL, 2014). A computação em

nuvem deve possuir algumas características essenciais como atendimento sob demanda, ter acesso amplo pela rede, agrupamento de recursos, elasticidade rápida, um sistema em nuvem que controla e otimizam os recursos automaticamente. Existem três tipos de serviços de computação em nuvem definidos (MELL, 2014):

Infraestrutura como Serviço (Infrastructure as a Service - IaaS): É um modelo de serviço que fornece recursos de computação como hardware, armazenamento e rede. O consumidor não gerencia ou controla a infraestrutura de nuvem, mas tem controle sobre sistemas operacionais, armazenamento e aplicativos. Podemos citar como exemplos de IaaS são o Apache CloudStack (APACHE, 2022), Amazon EC2 (AMAZON, 2023) e Google Cloud (GOOGLE, 2023a).

Plataforma como Serviço (Platform as a Service - PaaS): É um modelo de serviço onde o usuário pode utilizar para desenvolvimento e execução de aplicações. O consumidor não gerencia ou controla a infraestrutura de nuvem, incluindo rede, servidores, sistemas operacionais ou armazenamento, mas tem controle sobre os aplicativos implantados. Os exemplos de PaaS são Google Cloud Plataform (GOOGLE, 2023b), Microsoft Azure (MICROSOFT, 2023) e Red Hat OpenShift (REDHAT, 2022).

Software como Serviço (Software as a Service - SaaS): É um modelo de serviço onde é fornecido aplicativos executado em uma infraestrutura de nuvem. O consumidor não gerencia ou controla a infraestrutura de nuvem, incluindo rede, servidores, sistemas operacionais, armazenamento ou até mesmo recursos de aplicativos individuais, com a possível exceção de configurações de aplicativos. Os exemplos de SaaS são Google Apps (GOOGLE, 2022), Dropbox (DROPBOX, 2022) e Microsoft Office 365 (MICROSOFT, 2022).

Além disso, também existe quatro tipos de modelos de computação em nuvem (MELL, 2014):

Nuvem pública: a infraestrutura em nuvem é disponibilizada ao público geral.

Nuvem privada: a infraestrutura em nuvem é operada exclusivamente para uma organização.

Nuvem comunitária: a infraestrutura da nuvem é compartilhada por várias organizações.

Nuvem híbrida: a infraestrutura da nuvem por uma composição de duas ou mais nuvens (privada, pública e comunitária).

Este trabalho abordará o uso da IaaS com o software Apache CloudStack que será introduzido logo a seguir. Que será usado para o gerenciamento dos recursos da nuvem privada.

3.1.1 Apache CloudStack

O Apache CloudStack é um software de código aberto projetado para implantar e gerenciar grandes redes de máquinas virtuais com uma IaaS altamente escalável e disponível (APACHE, 2022). Usada por grandes empresas como Dell, Huawei e Walt Disney. Foi criada em 2008 por uma start-up que deu o seu primeiro nome como VMOps. Em 2010, adquiriu o nome Cloud.com e foi lançado o CloudStack. Em 2011, foi vendida para Citrix Systems que no ano seguinte doou para Apache Software Foundation.

O Apache CloudStack gerencia a computação, rede e recurso de armazenamento, utiliza vários hypervisors como KVM, vSphere e XenServer para virtualização e também suporta a API do AWS, além de suas próprias APIs (KUMAR et al., 2014). A Figura 3.1 ilustra os hypervisors, armazenamento e rede que podem ser utilizados pelo ambiente.

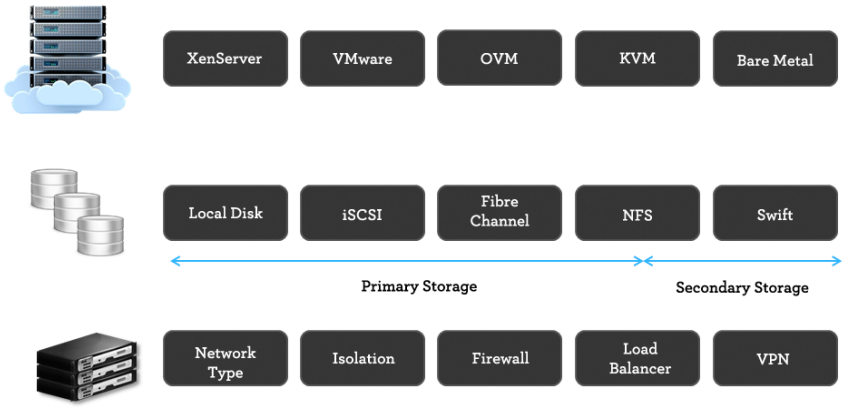


Figura 3.1: Plataforma aberta flexível do Apache CloudStack. (BLUE, 2018)

Segundo o Apache (APACHE, 2022), os recursos dentro da infraestrutura em nuvem do Apache CloudStack são regiões, zonas, pods, clusters, hosts, armazenamento primário e

secundário (APACHE, 2022). A Figura 3.2 ilustra a infraestrutura do Apache CloudStack com várias zonas sendo utilizadas.

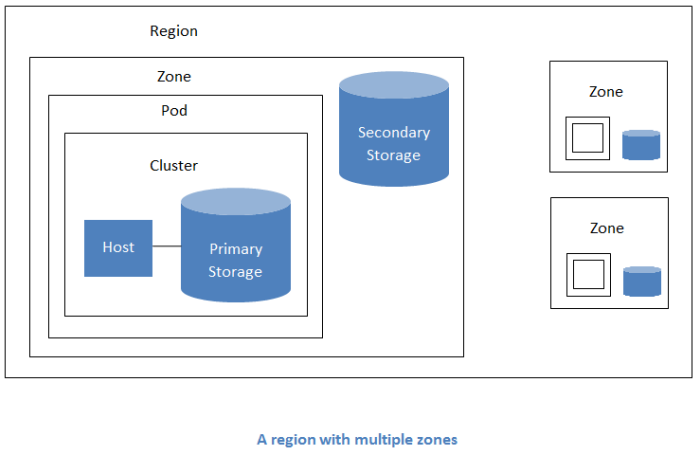


Figura 3.2: Infraestrutura em nuvem do Apache CloudStack com múltiplas zonas. (APACHE, 2022)

Regiões: uma coleção de uma ou mais zonas geograficamente próximas gerenciadas por um, ou mais servidores de gerenciamento (APACHE, 2022).

Zonas: é equivalente a um único *datacenter*. Uma zona consiste em uma ou mais pods e armazenamento secundário (APACHE, 2022).

Pods: um pod geralmente é um rack ou fileira de racks que inclui um switch de camada 2 e um ou mais clusters (APACHE, 2022).

Clusters: um *cluster* consiste em um ou mais hosts homogêneos e armazenamento primário (APACHE, 2022).

Hosts: um único nó de computação dentro de um cluster; muitas vezes um hipervisor (APACHE, 2022).

Armazenamento Primário: um recurso de armazenamento fornecido a um único cluster para a execução real de imagens de disco da instância (APACHE, 2022).

Armazenamento Secundário: um recurso de toda a zona que armazena modelos de disco, imagens ISO e instantâneos (APACHE, 2022).

A Figura 3.3 ilustra um exemplo da arquitetura do Cloudstack. Essa arquitetura é

composta por uma máquina que será o gerenciador Cloudstack com um banco de dados, outra máquina é responsável pelo armazenamento primário e armazenamento secundário e a última máquina será responsável pela virtualização, onde no exemplo é utilizado um *hypervisor* KVM. O KVM é utilizado para virtualizar as três máquinas virtuais de sistema que são o *Virtual Router* (VR), *Console Proxy Virtual Machine* (CPVM) e *Secondary Storage Virtual Machine* (SSVM). O VR é responsável pela comunicação de rede entre as instâncias e a internet, o CPVM é utilizado para acessar o console das instâncias via web e o SSVM é responsável por armazenar *templates* e ISO de sistemas operacionais.

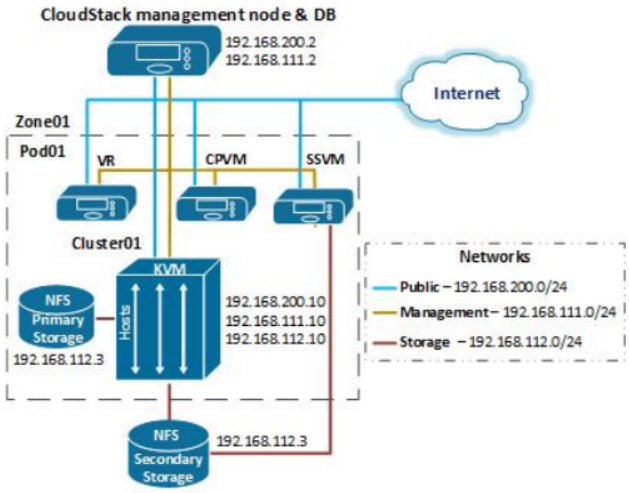


Figura 3.3: Exemplo de Arquitetura do Cloudstack.
(BRASCHER, 2019)

3.2 Avaliação de Desempenho

O principal objetivo para quem desenvolve sistema é buscar alto desempenho com baixo custo. Dessa forma, é necessário conhecer as principais métricas de desempenho, com a finalidade de se encontrar os limites do sistema e ter insumos suficientes para tomar decisões referentes ao comportamento em situações de sobrecarga (BARROS, 2018). Podem ser citadas como principais métricas de desempenho: tempo de resposta, vazão, utilização de CPU, memória ou armazenamento.

A avaliação de desempenho pode ser classificada com medição e modelagem (CALLOU

et al., 2011) conforme é ilustrado na Figura 3.4. A medição de um sistema é o método mais direto para a avaliação de desempenho. Com ele é possível obter resultados mais precisos, pois analisa o ambiente real do sistema. As técnicas de medição, apesar de fornecerem respostas exatas sobre o desempenho do sistema, não podem ser utilizadas se um novo sistema estiver sendo feito (TORRES et al., 2016). Também é importante ressaltar que uma alternativa de solução para tal questão poderia ser a adoção de abordagens estatísticas, que possam assegurar os resultados das medições. No entanto, o esforço computacional e humano pode tornar esta solução inadequada (JOHN; EECKHOUT, 2018). Um exemplo de ferramenta de medição é o Apache Jmeter (APACHE, 2023), que realiza testes de carga e estresse para o sistema e coleta métricas de desempenho e será utilizado nesse trabalho.

A modelagem é dividida em analítica e simulação. A modelagem analítica é utilizado para abstração do sistema. O modelo deve conter detalhes do sistema que são essenciais para o seu comportamento. Podemos citar como exemplos de modelagem analítica a rede de Petri e cadeias de Markov. A simulação tem como objetivo executar o modelo que procura reproduzir o comportamento do sistema que ele representa. A simulação do modelo dá resultados não tão precisos quanto os fornecidos pela medição, mas é possível calcular as estimativas. Para a verificar se o modelo representa o sistema, é necessária uma validação utilizando os resultados obtidos do modelo e da medição comparando com os seus intervalos de confiança. Este trabalho utilizará a rede de Petri estocástica para criação do modelo de desempenho.

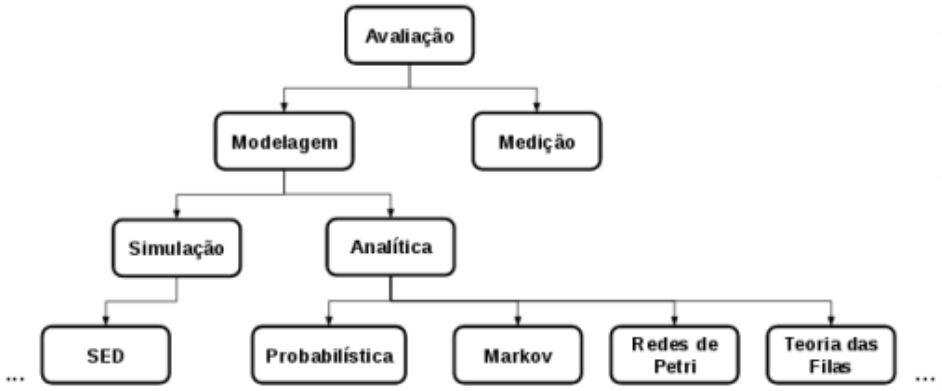


Figura 3.4: Técnicas de avaliação de desempenho.
(CALLOU et al., 2011)

3.3 Redes de Petri

A origem das redes de Petri ocorreu, em 1962, na tese de doutorado de Carl Adam Petri na faculdade de Matemática e Física da Universidade Darmstadt na Alemanha (MURATA, 1989). A rede de Petri é uma técnica de especificação de sistemas que permite uma representação matemática e é usada para modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos (MACIEL; LINS; CUNHA, 1996).

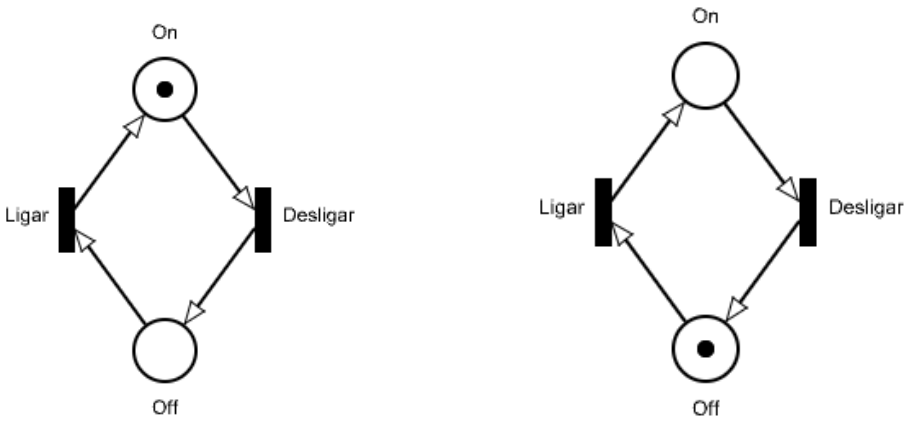
A rede de Petri é composta por 4 elementos: lugares, transições, arcos e tokens. Os lugares representam os estados que o sistema pode ter. As transições são ações realizadas para alterar o estado do sistema. Para uma transição estar habilitada, é preciso que as pré-condições sejam satisfeitas. Os arcos representam o fluxo de token pela rede e os tokens representam o estado que o sistema se encontra. Graficamente, os lugares são representados por círculos, as transições, por barras, os arcos, por setas e os tokens, por pontos. A Figura 3.5 ilustra os elementos da rede de Petri.

Figura 3.5: Elementos da rede de Petri.

A representação formal de uma rede de Petri é uma 5-tupla $PN = \{P, T, F, W, M_0\}$ (MURATA, 1989), onde:

- $P = \{p_1, p_2, ..., p_n\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_1, ..., t_n\}$ é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos;
- $W : F \to \{1, 2, 3, ...\}$ é uma função de peso do arco;
- $M_0 : P \to \{0, 1, 2, 3, ...\}$ é a marcação inicial, $P \cap T = \varnothing$ e $P \cup T \neq \varnothing$

A seguir, a Figura 3.6 mostra um exemplo de PN. O modelo apresenta o funcionamento de um sistema, onde os lugares representam o estado do sistema (on ou off) e as transições representam as ações que altera o estado do sistema (ligar e desligar). O modelo na Figura 3.6a mostra que o sistema está ligado, pois possui token no lugar *On*. A única transição que pode ser disparada é Desligar. Depois do disparo dessa transição, o modelo passará para o estado desligado e irá possuir um token no lugar *Off* (Figura 3.6b). Após isso, a transição habilitada para disparar é a Ligar, e o disparo desta fará o sistema voltar ao estado de ligado como na Figura 3.6a.



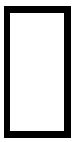
(a) Sistema Ligado. (b) Sistema Desligado.

Figura 3.6: Exemplo de uma rede de Petri.

3.3.1 Redes de Petri Estocásticas

As redes de Petri estocástica são uma extensão das redes de Petri e são utilizadas para a modelagem de desempenho e dependabilidade (MURATA, 1989). Nas SPN existem dois elementos, como pode ser visto na Figura 3.7, que são as transições estocástica Figura 3.7a e o arco inibidor Figura 3.7b. A transição estocástica utiliza tempos associados, de modo que o período de habilitação da transição corresponde ao período de execução da atividade. Já o arco inibidor desativa a transição se houver tokens no lugar de origem do arco inibidor, e vai ser ativada quando não houver tokens (SILVA, 2016).

A definição de uma rede de Petri estocástica é uma 9-tupla $SPN = \{P, T, I, O, H, \Pi, G, M_0, Atts\}$ (GERMAN, 2000), onde:



(a) Transição estocástica.

(b) Arco inibidor.

Figura 3.7: Elementos da SPN

- $P = \{p_1, p_2, ..., p_n\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é um conjunto finito de transições;
- $I \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa a multiplicidade dos arcos de entrada (que podem ser dependentes de marcações);
- $O \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa a multiplicidade dos arcos de saída (que podem ser dependentes de marcações);
- $H \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa os arcos inibidores (que podem ser dependentes de marcações).
- $\Pi \in \mathbb{N}^m$ é um vetor vincula o nível o nível de prioridade para cada transição;
- $G \in (\mathbb{N}^n \to \{true, false\})^m$ é o vetor que associa uma condição de guarda relacionada à marcação do lugar a cada transição;
- $M_0 \in \mathbb{N}^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);
- $Atts = (Dist, W, Markdep, Policy, Concurrency)^m$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial), onde:
  - $Dist \in \mathbb{N}^m \to f$ é uma função de distribuição de probabilidade associada ao tempo de uma transição, onde o domínio de $fé[0, \infty)$ (esta distribuição pode ser dependente de marcação);
  - $W \in \mathbb{N}^m \to \mathbb{R}^+$ é uma função de peso (esta distribuição pode ser dependente de marcação);
  - $Markdep \in \{constant, enabdep\}$ é definido como a distribuição de probabilidade associada ao tempo de uma transição constante (*constant*) ou dependente de marcação (*enabdep*);

- $Policy \in \{prd, prs\}$ é a política de memória adotada pela transição (prd - *preemptive repeat different*, corresponde a *race enabling policy*; prs - *preemptive resume*, corresponde a *age memory policy*).
- $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde ss representa a semântica de *single server* e is representa a semântica de *infinity server*.

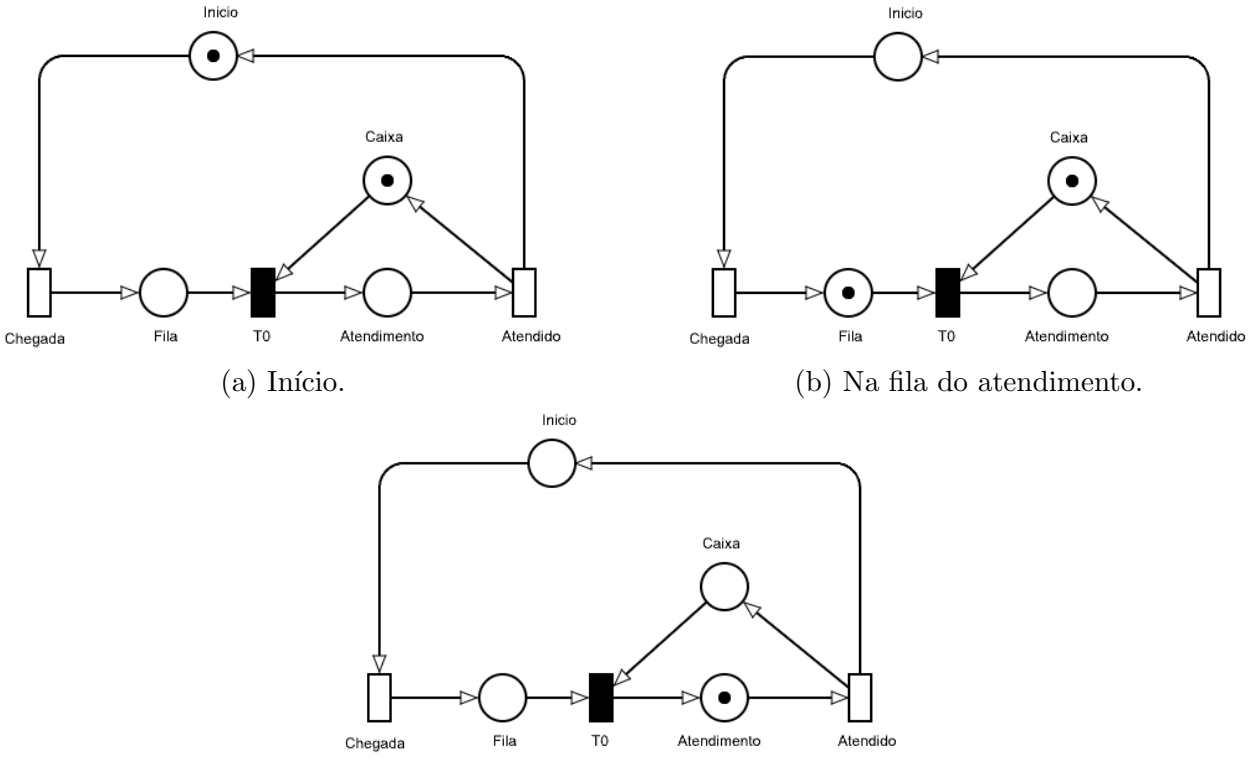
A Figura 3.8 mostra um exemplo de SPN, onde é representado um sistema com uma única fila. Esse modelo contempla o percurso da chegada de clientes à fila até o momento de ser atendido. O lugar *Inicio* representa o começo do sistema, conforme é apresentado na Figura 3.8a. A transição *Chegada* representa a frequência de chegada de clientes ao sistema. O disparo dessa transição faz com que um token do lugar *Inicio* seja consumido e seja gerado um token no lugar *Fila*, conforme pode ser visto na Figura 3.8b. Um token no lugar *Caixa* representa a quantidade de caixas disponíveis para atender aos clientes no sistema. Sendo assim, tendo um token no lugar *Caixa* e um token no lugar *Fila*, pode-se disparar a transição *T0*. O disparo dessa transição tem o tempo zero (0), e consome os tokens dos lugares *Caixa* e *Fila* e gera um token no lugar *Atendimento*, representando que o caixa foi alocado para realizar o seu atendimento (ver Figura 3.8c). O disparo da transição *Atendido* representa o término do atendimento. Esse disparo consome um token do lugar *Atendimento* e gera um token para o lugar *Caixa*, representando que o caixa se encontra livre para atender outro cliente e um token para o lugar *Inicio*, onde o cliente pode novamente entrar no sistema.

Com os modelos em SPN é possível realizar cálculos de métricas de desempenho. Exemplos dessas métricas são: vazão e tempo de resposta. A vazão é responsável por calcular a quantidade de requisições por um determinado tempo e é calculada conforme a Equação 3.1 (MACIEL, 2023).

$$V = \pi(m_p > 0) \times \mu \tag{3.1}$$

onde $\pi(m_p > 0) = P\{\#Atendimento > 0\}$ representa a probabilidade de ter tokens no lugar *Atendimento* e $\mu = \frac{1}{Atendido}$ que é a taxa de atendimento do sistema.

O tempo de resposta, que é o tempo entre o pedido de realização de um serviço pelo usuário e a chegada de resposta do pedido ao usuário (TRIVEDI et al., 1996). Usando a lei



(a) Início.

(b) Na fila do atendimento.

(c) Em atendimento.

Figura 3.8: Exemplo da SPN.

de Little é possível calcular o tempo de resposta, conforme a equação 3.2.

$$T = \frac{N}{V} \tag{3.2}$$

onde T representa o tempo de resposta, N indica o número de requisições no sistema e V representa a vazão do sistema (MACIEL, 2023). Então considerando o modelo representado pela Figura 3.8, $N = E\{\#Fila\} + E\{\#Atendimento\}$ que corresponde ao número de requisições para serem atendidas pelo sistema.

3.4 Regressão Linear

A regressão linear é uma técnica estatística para relacionar uma variável x que é independente com uma variável y que é dependente (MATOS, 1995). A regressão linear simples

é dada pela seguinte Equação 3.3.

$$y = a + bx \tag{3.3}$$

onde y é a resposta de saída prevista, x é a variável de entrada e a e b são os parâmetros de regressão. O cálculo do parâmetro de regressão b é dada pela seguinte Equação 3.4.

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \tag{3.4}$$

onde n é quantidade de pontos (x_i, y_i). Já o cálculo do parâmetro de regressão a é dada pela seguinte Equação 3.5.

$$a = \bar{y} - b\bar{x} \tag{3.5}$$

onde $\bar{y}$ e $\bar{x}$ é a média aritmética da variável de saída e de entrada respectivamente. Existe um grau de erro associado a regressão que é chamado de coeficiente de determinação e é representado por R^2 (LILJA, 2000). Esse grau de erro varia entre zero e um. Quanto mais próximo o valor do coeficiente de determinação estiver de um, maior será a validade da regressão. O cálculo do coeficiente de determinação é dada pela seguinte Equação 3.6.

$$R^2 = \frac{b\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.6}$$

A Tabela 3.1 apresenta um exemplo com vários tamanhos de arquivos (variável independente) e os tempos de leitura (variável dependente) para realizar o cálculo da regressão linear.

Tabela 3.1: Exemplo para cálculo da regressão linear.

Tamanho do Arquivo (bytes)	**Tempo de Leitura (μs)**
10	3,8
50	8,1
100	11,9
500	55,6
1000	99,6
5000	500,2
10000	1006,1

Com esses dados é possível computar a regressão a partir das Equações 3.4, 3.5 e 3.6. Os resultados dos parâmetros de regressão obtidos foram os seguintes: $b = 0,1002$ e $a = 2,24$. O resultado para o coeficiente de regressão é $R^2 = 0,9998$ e a equação da regressão linear obtida para esse exemplo é $y = 2,24 + 0,1002x$. A Figura 3.9 ilustra a regressão linear do exemplo da Tabela 3.1.

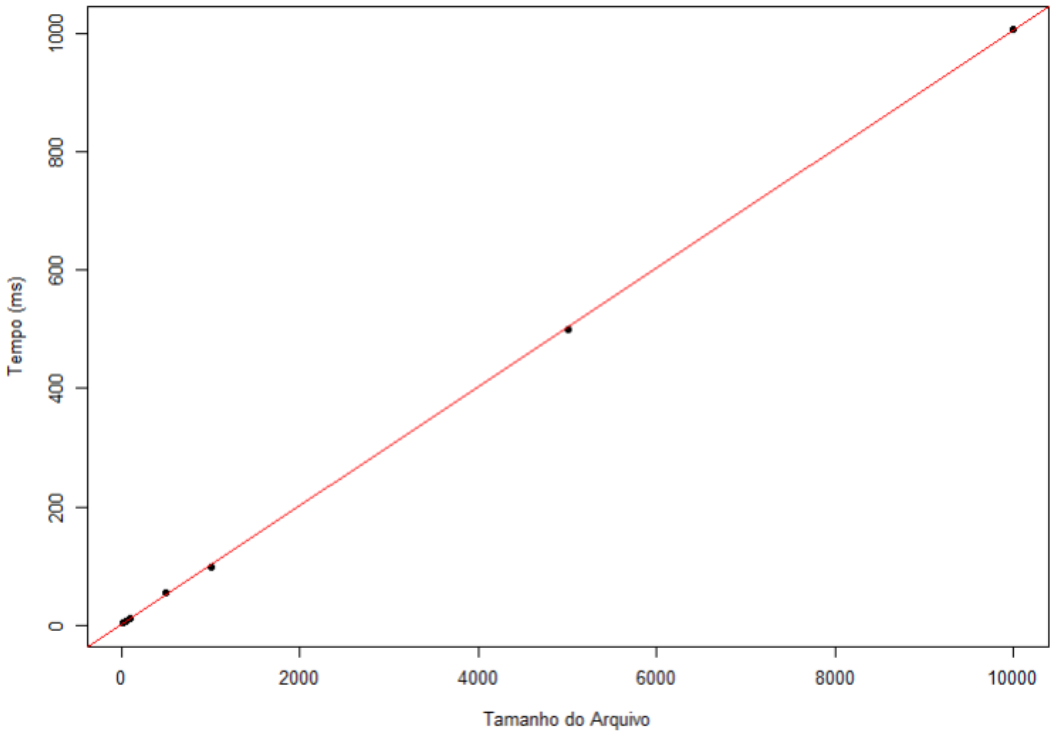


Figura 3.9: Gráfico regressão linear.

3.5 Disponibilidade

A disponibilidade, que é a probabilidade que o sistema esteja operacional durante um determinado período de tempo (MACIEL et al., 2012). A disponibilidade é dada através da razão entre o tempo de funcionamento e a soma do tempo de funcionamento e o tempo de falha, conforme mostrado na Equação 3.7.

$$A = \frac{E[Uptime]}{E[Uptime] + E[Downtime]} \tag{3.7}$$

onde $E[Uptime]$ é o tempo de funcionamento do sistema e $E[Downtime]$ é o tempo de falha do sistema.

Uma outra forma para se calcular a métrica de disponibilidade é através dos tempos médios de falhas (MTTF) e os tempos médios de reparos (MTTR) dos componentes presente no sistema. A equação para o cálculo da disponibilidade é apresentada na Equação 3.8.

$$A = \frac{MTTF}{MTTF + MTTR} \tag{3.8}$$

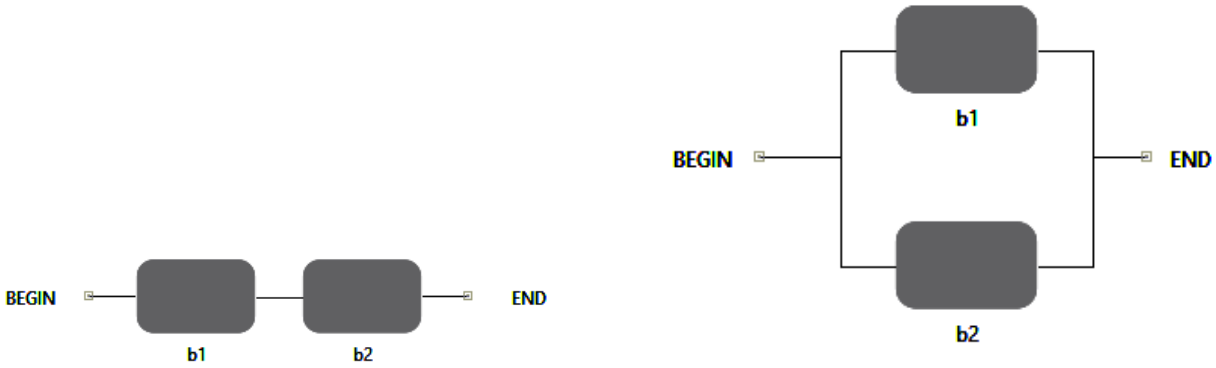
onde $MTTF$ é *Mean Time to Failure* que é o tempo médio para ocorrer falhas no sistema e o *Mean Time to Repair* é o tempo médio para reparar o sistema.

3.6 Diagramas de Bloco de Confiabilidade

Os Diagramas de Bloco de Confiabilidade (RBD) foram inicialmente propostos como uma técnica para calcular a confiabilidade de um sistema, mas atualmente também são utilizados para cálculo de disponibilidade. Em um diagrama de blocos, os componentes são representados por blocos, interligados com outros blocos e podem ter as seguintes composições: série, paralelo ou combinações dessas estruturas (TRIVEDI et al., 1996).

A Figura 3.10 ilustra dois modelos de RBD, onde na Figura 3.10a os blocos são organizados em série e na Figura 3.10b os blocos são organizados em paralelo. No sistema em

série, quando um componente falhar, o sistema todo irá parar. Já no sistema em paralelo, o sistema só irá falhar se em ambos os blocos falhem.



(a) Série. (b) Paralelo.

Figura 3.10: Diagrama de blocos de confiabilidade.

A confiabilidade ou disponibilidade de dois blocos conectados em série (R_s), considerando n componentes, é obtida pela Equação 3.9 :

$$R_s = \prod_{i=1}^{n} R_i \tag{3.9}$$

onde R_i é a confiabilidade $R_i(t)$, disponibilidade instantânea $A_i(t)$ ou disponibilidade estacionária A_i do bloco i.

A confiabilidade ou disponibilidade de dois blocos conectados em paralelo (R_p), considerando n componentes, é obtida pela Equação 3.10:

$$R_p = 1 - \prod_{i=1}^{n} (1 - R_i) \tag{3.10}$$

onde R_i é a confiabilidade $R_i(t)$, disponibilidade instantânea $A_i(t)$ ou disponibilidade estacionária A_i do bloco i.

3.7 Injeção de Falha

Uma falha é um evento que ocorre quando um serviço é entregue incorretamente e é percebido por usuários ou sistemas externos (AVIZIENIS; LAPRIE; RANDELL, 2001). A injeção de falhas é definida como uma técnica de validação de dependabilidade que consiste na realização de experimentos controlados, onde são introduzidos falhas predeterminadas em um sistema (ZIADE et al., 2004).

Segundo (ZIADE et al., 2004), existem cinco tipos de categorias de injeções de falhas que são: injeção de falha baseada em *hardware*, injeção de falha baseada em *software*, injeção de falha baseada em simulação, injeção de falhas baseada em emulação e injeção de falha híbrida.

Injeção de falha baseada em hardware é realizada por um *hardware* adicional para introduzir falhas no *hardware* do destino. A injeção de falha baseada em *hardware* se enquadra em duas categorias, a com contato, onde o injetor produz alterações na tensão ou corrente do chip, ou a sem contato, que o injetor não tem contato com sistema destino e produz algum fenômeno físico, como, por exemplo, radiação pesada ou interferência eletromagnética (HSUEH; TSAI; IYER, 1997).

Injeção de falha baseada em software consiste em reproduzir ao nível de software os erros que seriam produzidas ao ocorrerem falhas no *hardware* (ZIADE et al., 2004). Além disso, eles podem ser usados para direcionar aplicativos e sistemas operacionais, o que é difícil de ser feito com a injeção de falha por *hardware*, e por não precisar de *hardware* caro se torna tão atrativo (HSUEH; TSAI; IYER, 1997).

Injeção de falha baseada em simulação consiste em injetar falhas em modelos de alto nível como VHDL, assim permitindo avaliar antecipadamente a dependabilidade do sistema (ZIADE et al., 2004).

Injeção de falha baseada em emulação se baseia no uso de arranjo de porta programável em campo (FPGA) para acelerar a simulação de falhas e explora FPGA para emulação de circuito efetivo (ZIADE et al., 2004). Esta técnica vem sendo uma solução alternativa a injeção de falha por simulação.

Injeção de falha híbrida tem como objetivo combinar a injeção de falha por *software* e monitoramento de *hardware* (ZIADE et al., 2004).

Este trabalho utiliza a injeção de falha baseada em software, onde serão injetadas falhas nas placas de redes dos *hosts* após um determinado tempo.

3.8 Método Keesee

Para a realizar a validação do modelo de disponibilidade deste trabalho é necessário determinar o intervalo de confiança do teste de injeção de falha, então foi utilizado o método de Keesee (KEESEE, 1965). O método de Keesee tem como objetivo a determinação de um intervalo de confiança para disponibilidade, quando os tempos de MTTF e MTTR são estimados (VASCONCELOS, 2019). O primeiro passo para se calcular o intervalo de confiança é calcular o θ e o ϕ conforme as Equações 3.11 e 3.12.

$$\theta = \frac{\varepsilon}{n} \tag{3.11}$$

$$\phi = \frac{\eta}{m} \tag{3.12}$$

onde ε é o tempo total que o sistema esteve disponível, n é a quantidade total de falhas, η é o tempo total que o sistema esteve indisponível e m é quantidade total de reparos.

Com o resultado dessas equações podemos calcular a disponibilidade conforme a Equação 3.13.

$$A = \frac{\theta}{\theta + \phi} \tag{3.13}$$

Após calcular a disponibilidade do sistema, será feito o cálculo para obter os limites de confiança superiores e inferiores do intervalo de confiança conforme as Equações 3.14 e 3.15.

$$(\frac{\phi}{\theta})_U = \frac{n}{n-1}\frac{\phi}{\theta}F_{\frac{1-a}{2};2n;2m} \tag{3.14}$$

$$(\frac{\phi}{\theta})_L = \frac{n}{n-1}\frac{\phi}{\theta}F_{\frac{a}{2};2n;2m} \tag{3.15}$$

onde o $F_{\frac{1-a}{2};2n;2m}$ é a parte superior da função de densidade F com 2n e 2m graus de liberdade e $F_{\frac{a}{2};2n;2m}$ é a parte inferior da função de densidade F com 2n e 2m graus de liberdade. Já o $1-a$ é o nível de confiança. Desta forma, a disponibilidade pode ser expressa como apresentada na Equação 3.16.

$$A = \frac{1}{1+(\frac{\phi}{\theta})} \tag{3.16}$$

Então, os limites inferiores e superiores do intervalo de confiança da disponibilidade pode ser obtido das relações apresentadas nas Equações 3.17 e 3.18.

$$A_U = \frac{1}{1+(\frac{\phi}{\theta})_L} \tag{3.17}$$

$$A_U = \frac{1}{1+(\frac{\phi}{\theta})_U} \tag{3.18}$$

3.9 Considerações Finais

Este capítulo apresentou os conceitos básicos sobre computação em nuvem e mostrou o ambiente Apache CloudStack que será utilizado neste trabalho. Também foi introduzido a avaliação de desempenho e os principais conceitos sobre redes de Petri e sua extensão SPN. Também foi apresentado a regressão linear e um exemplo de como realizar os cálculos para obtenção da equação. Foram descritas as definições de dependabilidade e do RBD e suas organizações. Por fim, foram apresentados os conceitos de injeção de falha e o método de Keesee para calcular o intervalo de confiança do teste de injeção de falha e, assim, poder

conduzir a validação dos modelos SPN propostos.

Capítulo 4

Metodologia e Ambiente de Teste

Este capítulo apresenta as duas metodologias utilizadas para: (i) avaliação de desempenho do Nextcloud em uma nuvem privada e (ii) avaliação da disponibilidade da infraestrutura do Cloudstack. Também são contextualizadas as atividades realizadas e os objetos adquiridos em cada uma das atividades. Por fim, é apresentado o ambiente de teste implantado em laboratório para a realização dos cenários a serem avaliados no estudo de caso.

4.1 Metodologia Geração de Modelo de Desempenho

A Figura 4.1 ilustra a metodologia adotada para a geração do modelo de desempenho, destacando as suas atividades que são: estudar o sistema, obter métricas de desempenho, realizar medições, criar modelo, calcular regressão linear e validar o modelo, realizar projeções e os objetos adquiridos ao final de cada atividade. A seguir, cada fase da metodologia adotada será detalhada.

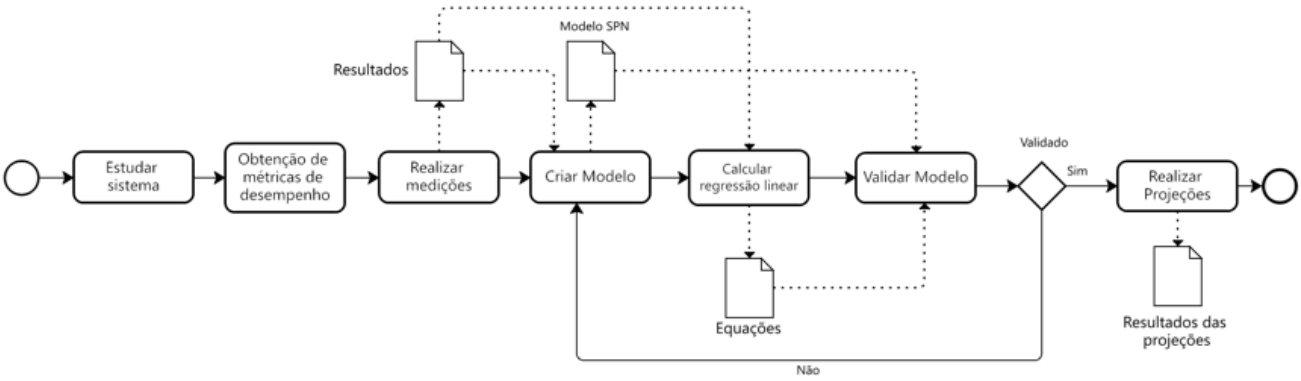


Figura 4.1: Metodologia de geração de modelo de desempenho.

A primeira etapa da metodologia corresponde ao estudo do sistema que será analisado. Sendo assim, o objetivo dessa atividade é ter um entendimento do funcionamento da plataforma Nextcloud. Após conhecer a plataforma, é realizado um estudo para definir quais métricas de desempenho serão utilizadas. Nesse trabalho será utilizado o tempo de resposta e a vazão. Em seguida, medições são realizadas, com o auxílio da ferramenta Jmeter para gerar a carga de trabalho ao Nextcloud e coletar as métricas. Nesta etapa também deve-se definir as atividades de carga de trabalho que serão realizadas. Por exemplo, foram definidas seis atividades para serem realizadas no Nextcloud que são: acessar o ambiente do Nextcloud, realizar *login*, acessar a base de arquivos, realizar o *upload* de um arquivo de 1 MB e sair do ambiente do Nextcloud ao realizar o *logout*.

Na atividade de medição também deve-se buscar identificar a quantidade máxima de usuários simultâneos que o sistema irá ter capacidade de gerenciar simultaneamente. Por exemplo, em nossos testes foram utilizados no máximo 90 usuários simultâneos, pois ao se utilizar mais do que essa quantidade perdas de pacotes eram recorrentes.

A etapa seguinte corresponde a criação do modelo de desempenho. É importante destacar que o modelo SPN criado nessa atividade será refinado na atividade seguinte com a utilização da regressão linear. A expressão obtida através da regressão será parâmetro de entrada para as transições do modelo SPN proposto para representar os tempos que cada atividade demanda nos mais diferentes cenários sob análise. Por exemplo, ao se variar a quantidade de usuários no sistema. Uma vez criado o modelo, deve-se partir para a próxima etapa, a da regressão linear. Por exemplo, foi preciso fazer uma regressão para a definição da relação

entre a quantidade de clientes no sistema e o tempo de atendimento. Ao final foram geradas equações que serão utilizados no modelo SPN de desempenho proposto.

A etapa seguinte da metodologia corresponde a validação do modelo SPN. Esta fase tem como objetivo validar o modelo para garantir que ele é confiável e realiza o que foi proposto. Para a realizar a validação foi utilizado as técnicas do teste t pareado (STUDENT, 1908) e seus pressupostos sendo os teste de normalidade de Shapiro-Wilk (SHAPIRO; WILK, 1965) e o teste de variâncias homogêneas (SNEDECOR; COCHRAN, 1967).

Uma vez de posse do modelo validado, deve-se ir para a etapa de realização das projeções. Esta fase será responsável para realizar projeções do comportamento do sistema em cenários que não tivemos como analisar por limitações, sejam de usuários para testar o ambiente ou até limitações físicas dos *hardwares* disponíveis em laboratório.

4.2 Metodologia Geração de Modelo de Disponibilidade

A Figura 4.1 ilustra a metodologia adotada para a geração do modelo de disponibilidade, destacando as suas atividades que são: estudar o sistema, obter métricas de dependabilidade, criar modelo, realizar o teste de injeção de falha, obter intervalo de confiança, validar o modelo e realizar projeções e os objetos adquiridos ao final de cada atividade. A seguir, cada fase da metodologia adotada será detalhada.

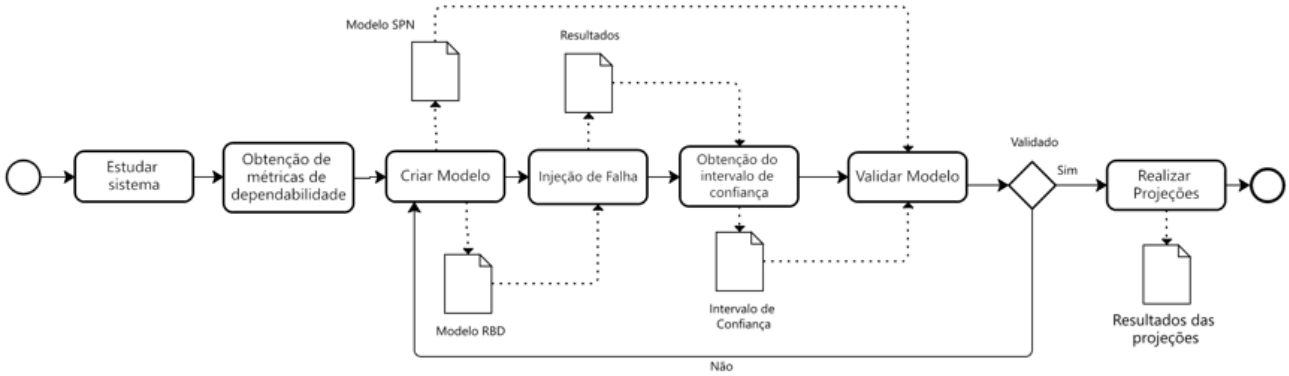


Figura 4.2: Metodologia de geração de modelo de disponibilidade.

A primeira etapa da metodologia corresponde ao estudo do sistema que será analisado. Sendo assim, o objetivo dessa atividade é realizar um estudo sobre a infraestrutura do Apache Cloudstack. Em seguida, será realizado a obtenção das métricas de dependabilidade como tempo médio de falha, tempo médio de reparo e disponibilidade.

A atividade seguinte é a criação dos modelos de disponibilidade, nessa etapa será criado um modelo em RBD que será utilizado na atividade de injeção de falha e um modelo em SPN para o cálculo da disponibilidade da arquitetura da Figura 4.3, que será explicada na próxima seção. Na atividade de injeção de falha, é criado um *script* em Python (ver Apêndice A) para injetar falhas e reparos na infraestrutura da nuvem no objetivo de obter dados de tempos e quantidades de falhas e reparos no sistema. Com os resultados do teste de injeção de falha, será utilizado o método de Keesee (KEESEE, 1965) para se obter o intervalo de confiança da disponibilidade. Logo após, será feito a atividade de validação do modelo que é feito comparando o resultado do modelo SPN com o intervalo de confiança.

Por fim, com o modelo validado é feito a última atividade de projeções. Com o objetivo de tentar melhorar a disponibilidade da arquitetura base de teste.

4.3 Ambiente de Teste

A Figura 4.3 ilustra a arquitetura do ambiente de teste. Esse ambiente é composto por 4 máquinas físicas, sendo três computadores HP e um servidor Dell. As máquinas HP possuem as mesmas configurações (processador AMD A8-550B 3.2 GHz com 4 cores, 8 GB de memória RAM, 500 GB de armazenamento e sistema operacional CentOS 7). Já o servidor Dell tem um processador Intel Xeon e3-1220 3.0 GHz com 4 cores, 16 GB de memória RAM, 1 TB de armazenamento e sistema operacional CentOS 7.

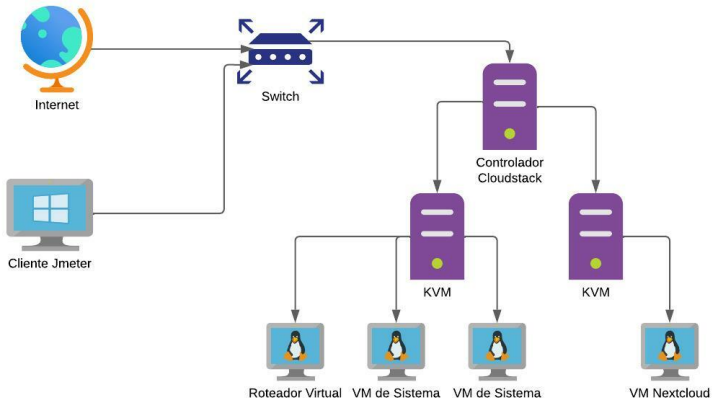


Figura 4.3: Arquitetura do ambiente de teste.

Uma máquina física HP foi utilizada como cliente, onde foi instalada a ferramenta de geração de carga utilizada que foi o JMeter. Um plano de teste foi definido nessa ferramenta para a realização dos experimentos no ambiente do Cloudstack implementado em laboratório com o NextCloud configurado. Por exemplo, foi construída uma base de dados de usuários com a função de executar uma carga de trabalho para os testes no ambiente gerenciado pelo Cloudstack. Foram definidas cinco atividades para realizar no ambiente: (i) acessar o Nextcloud, (ii) realizar o login, (iii) entrar nos arquivos, (iv) realizar um *upload* de arquivo de 1 MB e (v) finalizar realizando o logout.

O ambiente do Cloudstack foi configurado com três hosts (máquinas físicas), sendo o servidor Dell destinado ao ambiente que foi diagnosticado por demandar mais carga, no caso, o host3 com a VM do Nextcloud. O gerenciador da nuvem do Cloudstack foi configurado em uma máquina HP, sendo ela o responsável por fazer o gerenciamento dos recursos, por exemplo, descobrir novos host e os recursos por eles disponibilizados. Com o objetivo de isolar mais o ambiente que estamos analisando, no caso o host3 com o Nextcloud, optamos por colocar todas as VMs que são nativas do ambiente do Cloudstack (máquinas virtuais do sistema e o roteador virtual) em um host em separado (host2). O host2 é responsável pela alocação de todos os recursos de rede (ex., IPs das VMs), alocação de hardware (ex., espaço em disco e mémoria RAM para as VMs), segurança, entre outros. A instância do Nextcloud (VM) foi configurada com 2 cores, 4 GB de memória RAM e 50 GB de armazenamento.

4.4 Considerações Finais

Este capítulo apresentou as atividades que compõem a metodologia para avaliação de desempenho do Nextcloud em uma nuvem privada e avaliação da disponibilidade da infraestrutura do Cloudstack. Também foi apresentado o ambiente teste criado em laboratório. Com as atividades apresentadas é possível criar modelos SPN e RBD para a avaliação de desempenho e de disponibilidade do sistema.

Capítulo 5

Modelos

Esse capítulo apresenta o modelo proposto para avaliar o desempenho do Nextcloud configurado na nuvem privada Apache Cloudstack. Na apresentação do modelo são detalhadas as métricas adotadas, os parâmetros de entrada e, também, as regressões lineares utilizadas nas transições do modelo e que representam cada uma das atividades realizadas no Nextcloud. Posteriormente, esse capítulo apresenta os modelos RBDs que serão utilizados na fase de injeção de falhas. A injeção de falhas foi utilizada como uma técnica a fim de auxiliar no processo de validação do modelo de disponibilidade proposto. Por fim, o capítulo termina com a apresentação do modelo de disponibilidade em SPN, mostrando seus parâmetros e métricas adotadas.

5.1 Modelo de Desempenho

A Figura 5.1 ilustra o modelo proposto em SPN para analisar o desempenho do Nextcloud nos cenários em estudo. O modelo representa o funcionamento do sistema levando em consideração todo o comportamento desde a chegada de clientes até a resposta do servidor (VM na nuvem privada) com o Nextcloud. O cenário de estudo leva em consideração as seguintes atividades que são realizadas por cada um dos clientes que chegam: (i) acesso ao Nextcloud, (ii) realizar o *login*, (iii) acesso aos arquivos, (iv) realizar o *upload* de um arquivo e, por fim, (v) realizar o *logout*.

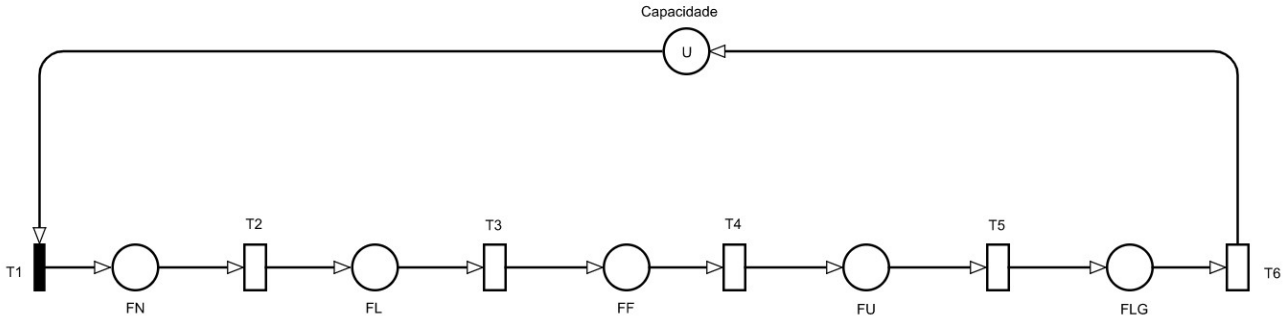


Figura 5.1: Modelo de desempenho proposto.

Nesse modelo, o lugar *Capacidade* representa a quantidade de clientes *(U)* que se encontram no sistema sendo atendidos pelo serviço simultaneamente. A transição imediata *T1* será disparada sempre que existir clientes para serem atendidos. Ao realizar o disparado dessa transição, um token é consumido do lugar *capacidade* e outro token é gerado no lugar *FN*, representando que um cliente chegou ao sistema. Os lugares *FN*, *FL*, *FF*, *FU*, *FLG* representam as filas de requisições para acessar o Nextcloud, realizar o *login*, acessar os arquivos, realizar o *upload* do arquivo e realizar o *logout*, respectivamente. As transições *T2*, *T3*, *T4*, *T5* e *T6* representam o tempo demandado para realizar cada uma dessas atividades. O disparo destas transições faz com que U tokens sejam consumidos do lugar que antecede a transição e gere *tokens* para o lugar subsequente. Isso ocorre devido à semântica adotada em cada transição temporizada do modelo (*T2* a *T6*) que é a de *Infinite Server*, conforme pode ser visto na Tabela 5.1. Além disso, a Tabela 5.2 apresenta uma descrição dos lugares do modelo e mostra o número inicial de tokens para cada lugar. A seguir, mais detalhes sobre os parâmetros de entrada e definições das métricas adotadas serão apresentados.

Tabela 5.1: Descrição das transições do modelo de desempenho.

Transição	Tipo	Tempo	Tipo Serv.
T1	Imediata	-	-
T2	Exponencial	$tNextcloud$	Infinite Server
T3	Exponencial	$tLogin$	Infinite Server
T4	Exponencial	$tFile$	Infinite Server
T5	Exponencial	$tUpload$	Infinite Server
T6	Exponencial	$tLogout$	Infinite Server

Tabela 5.2: Descrição dos lugares do modelo de desempenho.

Lugar	N.º Token Inicial	Descrição
Capacidade	U	Quantidade de clientes que serão atendidos pelo serviço.
FN	0	Fila de clientes para realizar o acesso ao Nextcloud.
FL	0	Fila de clientes para realizar o login.
FF	0	Fila de clientes para realizar o acesso aos arquivos.
FU	0	Fila de clientes para realizar o upload do arquivo.
FLG	0	Fila de clientes para realizar o logout.

5.1.1 Parâmetros de entrada e Métricas

O modelo SPN proposto possui seis parâmetros que devem ser especificados para que se consiga fazer a avaliação do modelo. O primeiro parâmetro é U que representa a quantidade de clientes que serão atendidos pelo serviço. Os outros parâmetros são o *tNextcloud*, que é o tempo demandado para acessar e ter a resposta do Nextcloud; *tLogin*, tempo demandado para a realização do login no sistema do Nextcloud; o *tFile*, tempo decorrido para se acessar o ambiente de arquivos do Nextcloud; *tUpload*, tempo para se subir um arquivo local ao sistema; e o *tLogout*, tempo necessário para se sair do ambiente do sistema. É importante destacar que estamos lidando sempre com quantidades distintas de usuários acessando o sistema. Sendo assim, adotamos a estratégia de medir o comportamento do sistema com diferentes quantidades de usuários realizando operações simultaneamente. Dessa forma, foi possível definir equações matemáticas para cada uma das atividades a partir de regressões lineares conforme será explicado na Seção 5.1.2.

No modelo SPN foram definidas duas métricas que são a de tempo de resposta e vazão. A métrica de tempo de resposta é representada no modelo por TR, sendo o período entre o envio da requisição para o sistema até a chegada da resposta ao cliente. A Equação 5.1 é a responsável por esse cálculo no modelo proposto.

$$TR = \frac{E\{\#FN\}+E\{\#FL\}+E\{\#FA\}+E\{\#FU\}+E\{\#FLG\}}{\frac{E\{\#FLG\}}{T6}} \tag{5.1}$$

onde $E\{\#FN\}$, $E\{\#FL\}$, $E\{\#FF\}$, $E\{\#FU\}$ e $E\{\#FLG\}$ representa a esperança de tokens nos lugares de nomes correspondentes, ou seja, indicam o número médio de tokens nos seus respectivos lugares. O $T6$ indica o tempo para ser realizado o logout dos usuários.

A métrica do modelo V é responsável pelo cálculo da vazão do sistema.A vazão representa a quantidade de requisições que o sistema atende durante um período. A fórmula da vazão é definida na Equação 5.2.

$$V = P\{\#FLG > 0\} \times \frac{1}{T6} \tag{5.2}$$

onde $P\{\#FLG > 0\}$ indica a probabilidade de ter tokens no lugar FLG multiplicado pela taxa média de atendimento de um cliente do lugar FLG que dado pelo tempo *T6*.

5.1.2 Regressão Linear

A Figura 5.2 apresenta a regressão linear entre a quantidades de clientes (x, independente) e o tempo decorrido para obter a resposta da requisição (y, dependente) da atividade de acesso ao Nextcloud. É possível notar que os pontos estão bem próximos da linha de tendência, e o coeficiente de determinação R^2 foi de 0,9665. Quanto mais próximo o R^2 for de um, significa um melhor ajustamento da regressão (GUJARATI; PORTER, 2011). A regressão linear entre a quantidade de clientes e o tempo de atendimento para acessar o Nextcloud é dada pela seguinte Equação 5.3.

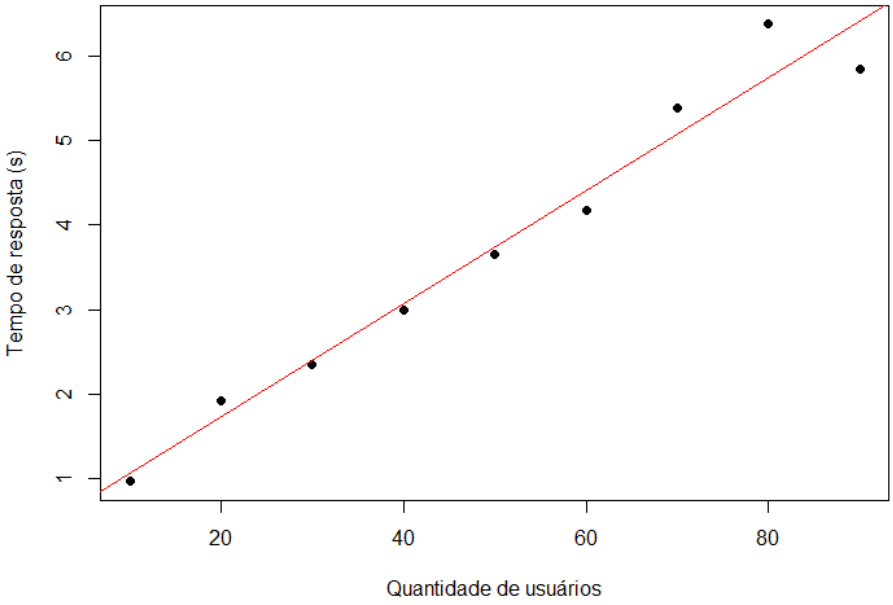


Figura 5.2: Regressão linear para acessar o Nextcloud.

$$y_1 = (66,81 \times U) + 399,55 \tag{5.3}$$

Onde o y_1 é a previsão do tempo de resposta para acessar o Nextcloud e U é a quantidade de clientes no sistema. A Equação 5.3 será utilizada na transição $T2$ do modelo de desempenho.

A Figura 5.3 apresenta a regressão linear da atividade de realizar *Login*. O coeficiente de determinação R^2 foi de 0,9671 e obteve a seguinte Equação 5.4.

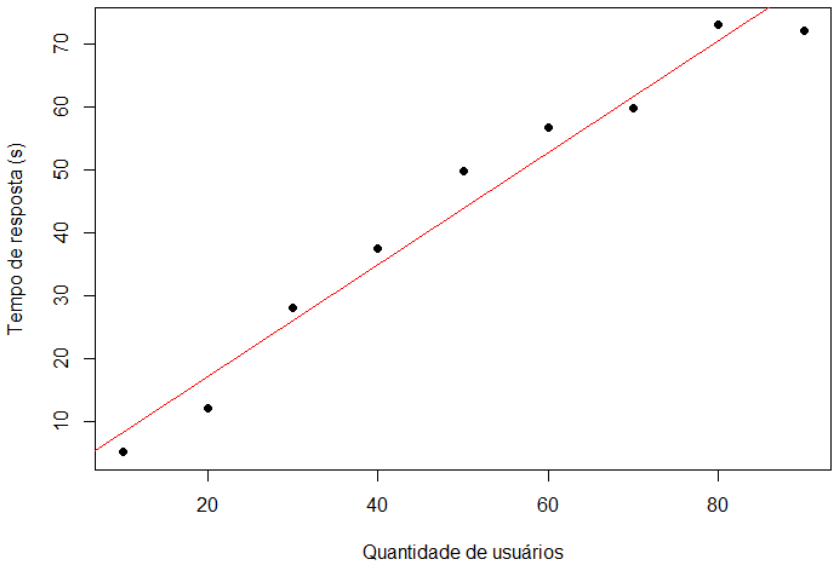


Figura 5.3: Regressão linear para realizar *login*.

$$y_2 = (889,80 \times U) - 729,20 \tag{5.4}$$

Onde o y_2 é a previsão do tempo de resposta para realizar o *login* no Nextcloud e U é a quantidade de clientes no sistema. A Equação 5.4 será utilizada na transição $T3$ do modelo.

A Figura 5.4 apresenta a regressão linear da atividade de acessar os arquivos. O coeficiente de determinação R^2 foi de 0,9542 e obteve a seguinte Equação 5.5.

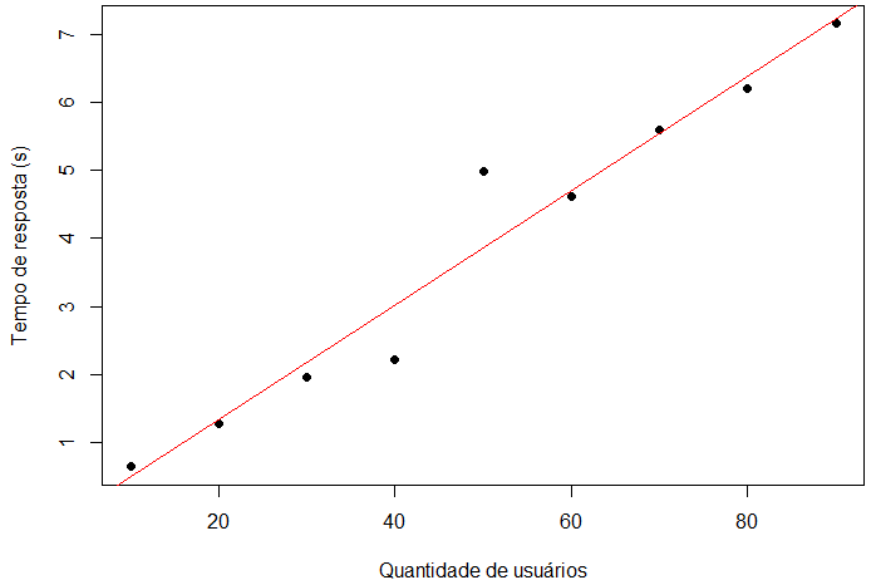


Figura 5.4: Regressão linear para acessar os arquivos.

$$y_3 = (84,17 \times U) - 357,35 \tag{5.5}$$

Onde o y_3 é a previsão do tempo de resposta para acessar os arquivos e U é a quantidade de clientes no sistema. A Equação 5.5 será utilizada na transição $T4$ do modelo.

A Figura 5.5 apresenta a regressão linear da atividade de realizar *upload* do arquivo. O coeficiente de determinação R^2 foi de 0,9626 e obteve a seguinte Equação 5.6.

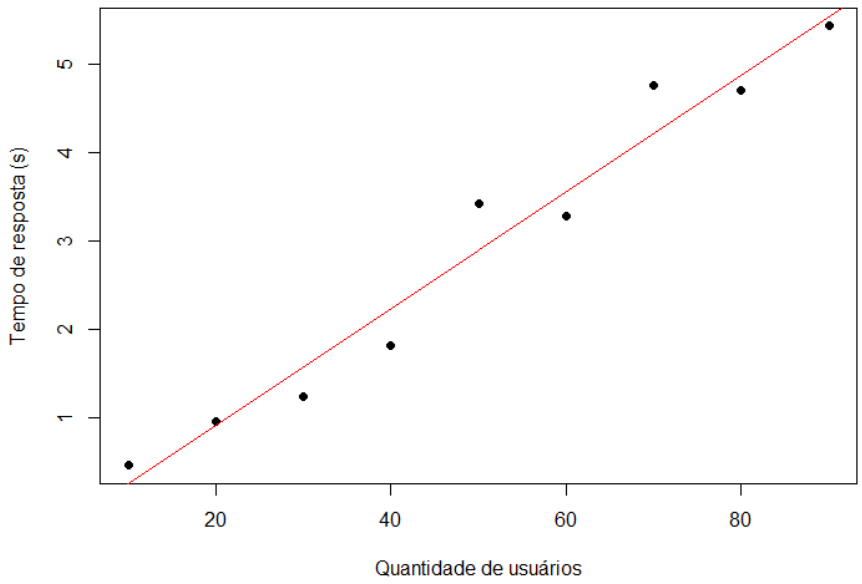


Figura 5.5: Regressão linear para realizar *upload* do arquivo.

$$y_4 = (66,12 \times U) - 408,99 \tag{5.6}$$

Onde o y_4 é a previsão do tempo de resposta para realizar *upload* do arquivo e U é a quantidade de clientes no sistema. A Equação 5.6 será utilizada na transição $T5$ do modelo.

A Figura 5.6 apresenta a regressão linear da atividade de realizar *logout*. O coeficiente de determinação R^2 foi de 0,9750 e obteve a seguinte Equação 5.7.

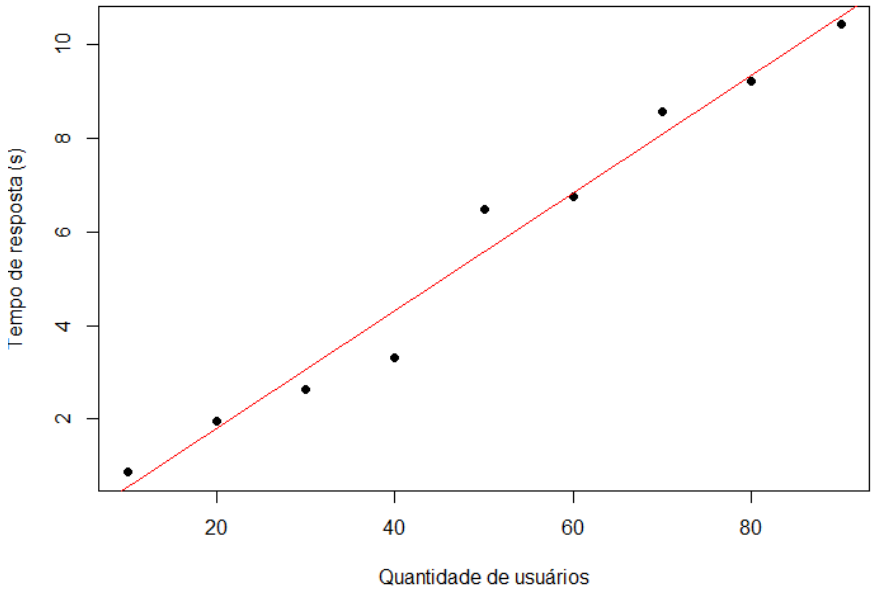


Figura 5.6: Regressão linear para realizar *logout*.

$$y_5 = (66,12 \times U) - 408,99 \tag{5.7}$$

Onde o y_5 é a previsão do tempo de resposta para realizar *logout* do Nextcloud e U é a quantidade de clientes no sistema. A Equação 5.7 será utilizada na transição $T6$ do modelo.

5.2 Modelo da Arquitetura em RBD

Os RBDs apresentam uma vantagem em relação ao fornecimento dos resultados, pois apresentam cálculos mais rápidos por meio de suas fórmulas do que as simulações e as análises numéricas dos SPNs, porém, as SPNs têm um maior poder de representação (BALBO, 2000; KUO; ZUO, 2003). O RBD utilizado nesse trabalho será utilizado no teste de injeção de falhas.

A Figura 5.7 ilustra o modelo RBD para representar a arquitetura do sistema que foi apresentado na Figura 4.3. O modelo composto pelos componentes: *Host1* e *Host2*. O *Host1* vai ser responsável pela virtualização das máquinas virtuais de sistemas, já o *Host2* é responsável pela virtualização da instância do Nextcloud. A confiabilidade da arquitetura

é apresentada na Equação 5.8. A Tabela 5.3 apresenta os parâmetros utilizados no modelo RBD da Figura 5.7 e suas descrições.

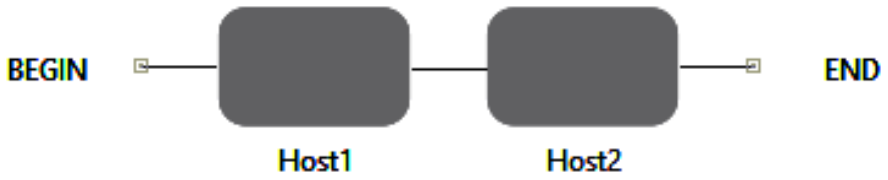


Figura 5.7: Modelo RBD da infraestrutura.

$$R_{Arq} = R_{Host1} \times R_{Host2} \tag{5.8}$$

Tabela 5.3: Parâmetros do modelo RBD da infraestrutura do CloudStack.

Parâmetro	**Descrição**
$MTTF_{Host1}$	Tempo médio para defeito no *Host1*
$MTTR_{Host1}$	Tempo médio para reparo no *Host1*
$MTTF_{Host2}$	Tempo médio para defeito no *Host2*
$MTTR_{Host2}$	Tempo médio para reparo no *Host2*

É importante destacar que o modelo RBD apresentado anteriormente representa uma visão de alto nível do ambiente do sistema em análise. Isso acontece pelo fato de que cada um dos dois blocos que compõe o modelo representam o resultado obtido pela avaliação de outros modelos. Por exemplo, a Figura 5.8 ilustra o modelo proposto para representar a disponibilidade do *Host1*. O *Host1* é composto pelos seguintes componentes: *hardware* (HW), sistema operacional (SO), *hypervisor* (KVM), roteador virtual (VR), máquina virtual de armazenamento secundário (SSVM) e a máquina virtual do *console proxy* (CPVM). A confiabilidade do *Host1* é computada a partir da avaliação desse modelo RBD e que pode ser também obtida pela avalição da Equação 5.9. A Tabela 5.4 apresenta os parâmetros utilizados no modelo RBD da Figura 5.8 e as suas descrições.

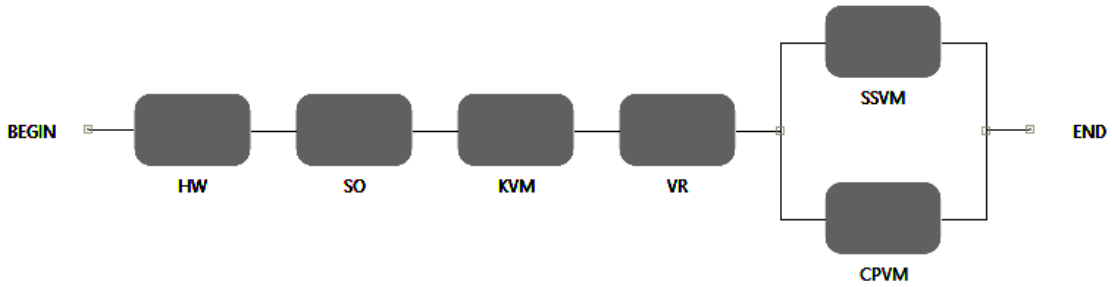


Figura 5.8: Modelo RBD do *Host1*.

$$R_{Host1} = (R_{HW} \times R_{SO} \times R_{KVM} \times R_{VR}) \times (1 - ((1 - R_{SSVM}) \times (1 - R_{CPVM}))) \quad (5.9)$$

Tabela 5.4: Parâmetros do modelo RBD do *Host1*.

Parâmetro	**Descrição**
$MTTF_{HW}$	Tempo médio para defeito do hardware
$MTTR_{HW}$	Tempo médio para reparo do hardware
$MTTF_{SO}$	Tempo médio para defeito do sistema operacional
$MTTR_{SO}$	Tempo médio para reparo do sistema operacional
$MTTF_{KVM}$	Tempo médio para defeito do *hypervisor*
$MTTR_{KVM}$	Tempo médio para reparo do *hypervisor*
$MTTF_{VR}$	Tempo médio para defeito no roteador virtual
$MTTR_{VR}$	Tempo médio para reparo no roteador virtual
$MTTF_{SSVM}$	Tempo médio para defeito na máquina virtual de armazenamento secundário
$MTTR_{SSVM}$	Tempo médio para reparo na máquina virtual de armazenamento secundário
$MTTF_{CPVM}$	Tempo médio para defeito na máquina virtual do *Console Proxy*
$MTTR_{CPVM}$	Tempo médio para reparo no roteador virtual do *Console Proxy*

A Tabela 5.5, a seguir, mostra os parâmetros utilizados nos componentes do *Host1*. Os tempos utilizados nesses parâmetros foram obtidos a partir de (MACIEL, 2016).

Tabela 5.5: Parâmetros de dependabilidade dos componentes do *Host1*.

Componente	MTTF (h)	MTTR (h)
HW	8760	1,66
SO	2893	0,25
KVM	2990	1
VR	619,56	0,84
SSVM	619,56	0,84
CPVM	619,56	0,84

Similarmente ao que foi feito para representar o *Host1*, iremos fazer uma abordagem similar para analisar e obter os temos médios de falha e reparo para o *Host2*. Esses tempos são utilizado no modelo RBD apresentado anteriormente na Figura 5.7. Sendo assim, a Figura 5.9 ilustra o modelo RBD utilizado para representar os componentes do *Host2*. O *Host2* é composto pelos seguintes componentes: *hardware* (HW), sistema operacional (SO), *hypervisor* (KVM) e a máquina virtual do Nextcloud (VR). Sendo assim, cada um desses componentes são representados como um bloco no modelo RBD proposto. É importante destacar que esses blocos devem ficar em série, pois a falha de qualquer um dessses componentes irá afetar a disponibilidade do serviço ofertado nesse host que é o NextCloud. Assim, a disponibilidade do *Host2* pode ser computada pela Equação 5.10. A Tabela 5.6 apresenta os parâmetros utilizados no modelo RBD da Figura 5.9 e as suas respectivas descrições.

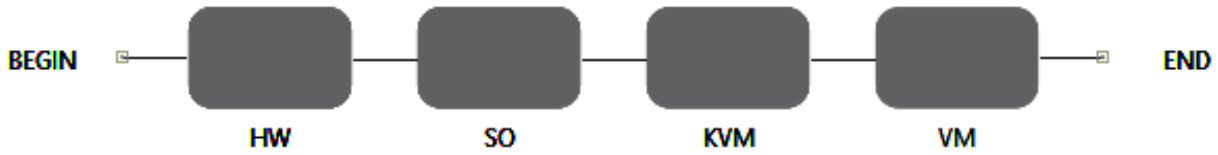


Figura 5.9: Modelo RBD do *Host2*.

$$R_{Host2} = R_{HW} \times R_{SO} \times R_{KVM} \times R_{VM} \tag{5.10}$$

Tabela 5.6: Parâmetros do modelo RBD do *Host1*.

Parâmetro	Descrição
$MTTF_{HW}$	Tempo médio para defeito do hardware
$MTTR_{HW}$	Tempo médio para reparo do hardware
$MTTF_{SO}$	Tempo médio para defeito do sistema operacional
$MTTR_{SO}$	Tempo médio para reparo do sistema operacional
$MTTF_{KVM}$	Tempo médio para defeito do *hypervisor*
$MTTR_{KVM}$	Tempo médio para reparo do *hypervisor*
$MTTF_{VM}$	Tempo médio para defeito na máquina virtual do Nextcloud
$MTTR_{VM}$	Tempo médio para reparo na máquina virtual do Nextcloud

A Tabela 5.7, a seguir, mostra os tempos médios de falha e reparo utilizados como parâmetros de entrada para os componentes do *Host2*. Esses tempos utilizados nesses parâmetros foram obtidos a partir de (MACIEL, 2016).

Tabela 5.7: Parâmetros de dependabilidade dos componentes do *Host2*.

Componente	MTTF (h)	MTTR (h)
HW	8760	1,66
SO	2893	0,25
KVM	2990	1
VM	619,56	0,84

Realizando a avaliação dos modelos RBDs chegaremos ao resultado dos MTTF E MTTR dos *Host1* e *Host2*, conforme mostrado na Tabela 5.8. Para o *Host1*, foi obtido o valor de 415,23 horas para o tempo médio de falha e de 0,25 horas para o tempo médio de reparo. Assumindo o *Host2*, obteve-se 1259,03 horas para o tempo médio de falha e 0,7685 horas para o tempo médio de reparo. Esses tempos serão utilizados no teste de injeção de falhas para realizar a validação do modelo de disponibilidade proposto em SPN.

Tabela 5.8: Parâmetros de dependabilidade dos componentes da arquitetura do sistema

Componente	MTTF (h)	MTTR (h)
Host1	415,23	0,25
Host2	1259,03	0,7685

5.3 Modelo de Disponibilidade em SPN

A Figura 5.10 ilustra o modelo proposto em SPN para computar a disponibilidade do ambiente conforme foi apresentado na Figura 4.3. Esse modelo é composto por dois hosts (Host1 e Host2), três VMs de sistema (SSVM, CPVM e VR) e uma VM com o serviço Nextcloud.

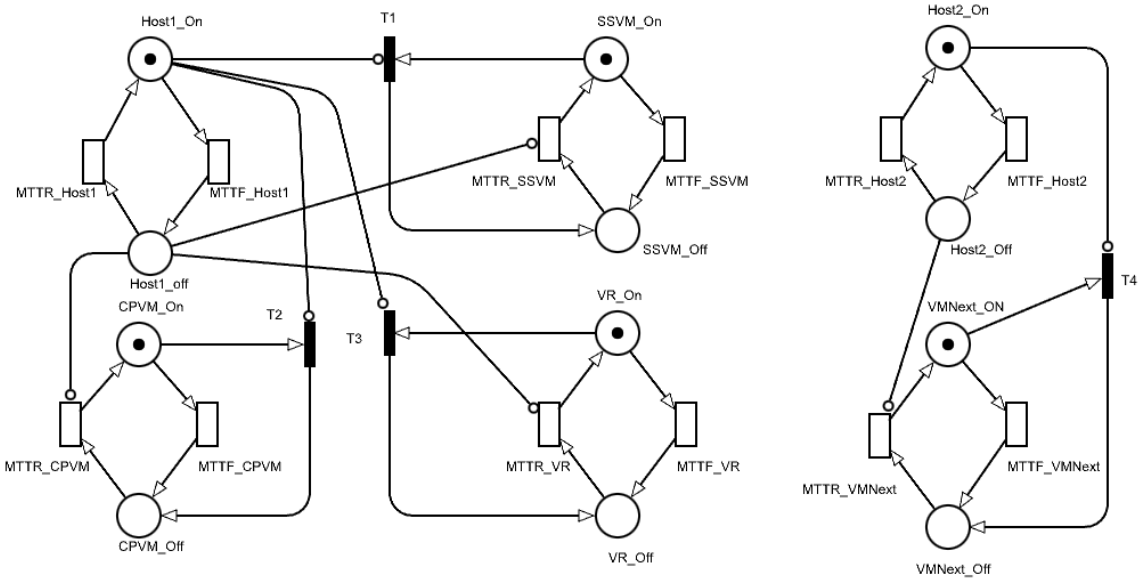


Figura 5.10: Modelo de disponibilidade proposto.

Um token nos lugares *Host1_On* e *Host2_On* representa que os servidores físicos $Host1$ e $Host2$ estão em funcionamento, respectivamente. Por outro lado, um token nos lugares *Host1_Off* e *Host2_Off* representa que os servidores estão indisponíveis. Um token nos lugares *SSVM_On*, *CPVM_On* e *VR_On* representam que a VM do *Secondary Storage*, a VM do *Console Proxy* e o roteador virtual estão todos eles disponíveis. Já um token em *SSVM_Off*, *CPVM_Off* e *VR_Off* mostram que essas VMs de sistemas estão indisponíveis. Um token no lugar *VMNext_On* representa que a VM está executando o serviço Nextcloud e está disponível, já o lugar *VMNext_Off* representa que a VM está indisponível.

O disparo da transição $MTTF\_Host1$ faz com que um token seja consumido do lugar *Host1_On* e um token seja gerado no lugar *Host1_Off*, representando que o $Host1$ saiu do estado de funcionamento para em falha. A ausência de token em *Host1_On* faz com que

as transições *T1*, *T2* e *T3* fiquem habilitadas para disparar. Como essas transições são imediatas e elas têm maior prioridade que as demais transições exponenciais presentes no modelo, isso faz com que seja possível representar que a falha do *Host1* derrube os demais serviços executados nesse dispositivo. Em outras palavras, a falha do *Host1* faz com que os tokens presentes nos lugares *SSVM_On*, *CPVM_On* e *VR_On* sejam consumidos e são gerados em *SSVM_Off*, *CPVM_Off* e *VR_Off*, indicando que essas VMs também entraram em falha instantaneamente com a falha do *Host1*. Similarmente, o $Host2$ ao ter o seu status modificado para em falha irá acionar a transição imediata *T4*. O disparo dessa transição imediata faz com que um token de *VMNext_On* seja consumido e se gera um token em *VMNext_Off*, representando que a VM com o serviço do NextCloud também entrou em falha.

A Tabela 5.9 apresenta o tipo, o tempo e a semântica das transições. Já a Tabela 5.10 apresenta uma descrição dos lugares do modelo e mostra o número inicial de tokens para cada lugar. A seguir, mais detalhes sobre os parâmetros de entrada e definições das métricas adotadas serão apresentados.

Tabela 5.9: Descrição das transições do modelo de disponibilidade.

Transição	Tipo	Tempo	Tipo Serv.
T1	Imediata	-	-
T2	Imediata	-	-
T3	Imediata	-	-
T4	Imediata	-	-
MTTF_Host1	Exponencial	$mttfh$	Single Server
MTTR_Host1	Exponencial	$mttrh$	Single Server
MTTF_Host2	Exponencial	$mttfh$	Single Server
MTTR_Host2	Exponencial	$mttrh$	Single Server
MTTF_SSVM	Exponencial	$mttfvm$	Single Server
MTTR_SSVM	Exponencial	$mttrvm$	Single Server
MTTF_CPVM	Exponencial	$mttfvm$	Single Server
MTTR_CPVM	Exponencial	$mttrvm$	Single Server
MTTF_VR	Exponencial	$mttfvm$	Single Server
MTTR_VR	Exponencial	$mttrvm$	Single Server
MTTF_VMNext	Exponencial	$mttfvm$	Single Server
MTTR_VMNext	Exponencial	$mttrvm$	Single Server

Tabela 5.10: Descrição dos lugares do modelo.

Lugar	N.º Token Inicial	Descrição
Host1_On	1	Host1 está em funcionamento.
Host1_Off	0	Host1 está indisponivel.
Host2_On	1	Host2 está em funcionamento.
Host2_Off	0	Host1 está indisponivel.
SSVM_On	1	SSVM está em funcionamento
SSVM_Off	0	SSVM está indisponível.
CPVM_On	1	CPVM está em funcionamento
CPVM_Off	0	CPVM está indisponível.
VR_On	1	VR está em funcionamento
VR_Off	0	VR está indisponível.
VMNext_On	1	A instância do Nextcloud está em funcionamento
VMNext_Off	0	A instância do Nextcloud está indisponível.

5.3.1 Parâmetros de entrada e métricas

O modelo SPN apresentado na Figura 5.10 possui 4 parâmetros que devem ser especificados para realizar a análise de disponibilidade do modelo. O primeiro parâmetro é $mttfh$ que representa o tempo médio de falha do *host*. O segundo parâmetro é $mttrh$ que é responsável pelo tempo médio de reparo do *host*. O próximo parâmetro é $mttfvm$ que é o tempo médio de falha de uma máquina virtual e o último parâmetro é $mttrvm$ que é o tempo médio de reparo de uma máquina virtual. Os tempos dessas transições foram retirados de (MACIEL, 2016).

O modelo também possui uma métrica que é responsável pelo cálculo de disponibilidade do sistema que é a métrica A. O serviço é considerado em funcionamento quando o roteador virtual e a VM com o Nextcloud também estiverem disponíveis. Então, o cálculo da disponibilidade do sistema é dado pela seguinte Equação 5.11:

$$A = P\{(\#VMNext\_On > 0)AND(\#VR\_On > 0)\} \tag{5.11}$$

onde $P\{(\#VMNext\_On > 0)AND(\#VR\_On > 0)\}$ é a probabilidade que tenha tokens nos lugares *VMNext_On* e *VR_On*.

5.4 Considerações Finais

Este capítulo apresentou o modelo de desempenho, os seus parâmetros e as métricas de tempo de resposta e de vazão que foram utilizadas. Também foram mostrados os modelos em RBD utilizados para representar a arquitetura do Cloudstack e que foi utilizado na injeção de falha. Além disso, foi apresentado também o modelo SPN proposto para quantificar a disponibilidade do serviço do Nextcloud configurado na nuvem privada na plataforma Cloudstack.

Capítulo 6

Estudo de Caso

Este capítulo apresenta seis estudos de caso. O primeiro estudo de caso apresenta a validação do modelo de desempenho proposto. Uma vez tendo sido realizada a validação do modelo SPN a partir do estudo anterior, o segundo estudo de caso realiza projeções feitas no modelo de desempenho. O terceiro estudo tem como foco a validação do modelo de disponibilidade. No quarto estudo de caso realiza uma comparação entre a arquitetura base e as arquiteturas com redundância. E no quinto e último estudo de caso, mostra-se a comparação entre diversos tipos de contratos SLA.

6.1 Estudo de Caso I

Esse estudo tem como objetivo a validação do modelo de desempenho proposto (ver Figura 5.1). A validação é um procedimento utilizado para certificar que os valores gerados pelo modelo apresentam coerência com os gerados pelo sistema real. No nosso caso, a validação foi feita a partir da comparação do tempo de resposta medido no sistema real com o tempo obtido via análise estacionária no modelo.

A Tabela 6.1 apresenta um resumo dos resultados obtidos para o tempo de resposta da medição que feito na atividade apresentada na Figura 4.1 e para a modelagem. Como parâmetros de entrada da análise do modelo, adotamos o nível de confiança de 95% e o

erro relativo em 10%. É possível perceber que os intervalos de confiança apresentados para a medição e modelagem se sobrepõe. Similarmente, a Figura 6.1 apresenta os resultados graficamente.

Tabela 6.1: Resultado para validação do modelo com tempo de resposta.

	Medição		**Modelagem**	
Qtd. Clientes	Tempo de Resp.(s)	Int. Conf.	Tempo de Resp.(s)	Int. Conf.
10	8,53	[7,89; 8,75]	10,53	[8,41; 10,83]
20	18,74	[17,12; 19,43]	22,83	[19,29; 24,12]
30	37,97	[36,73; 39,20]	35,21	[29,15; 41,91]
40	51,24	[50,71; 52,47]	47,70	[39,85; 57,48]
50	71,18	[70,35; 72,89]	59,17	[53,10; 71,01]
60	77,56	[75,99; 78,72]	71,97	[62,24; 78,50]
70	86,73	[85,44; 87,21]	84,72	[75,68; 92,73]
80	101,74	[100,32; 103,04]	96,86	[82,14; 103,24]
90	103,33	[102,46; 104,98]	109,20	[99,35; 116,89]

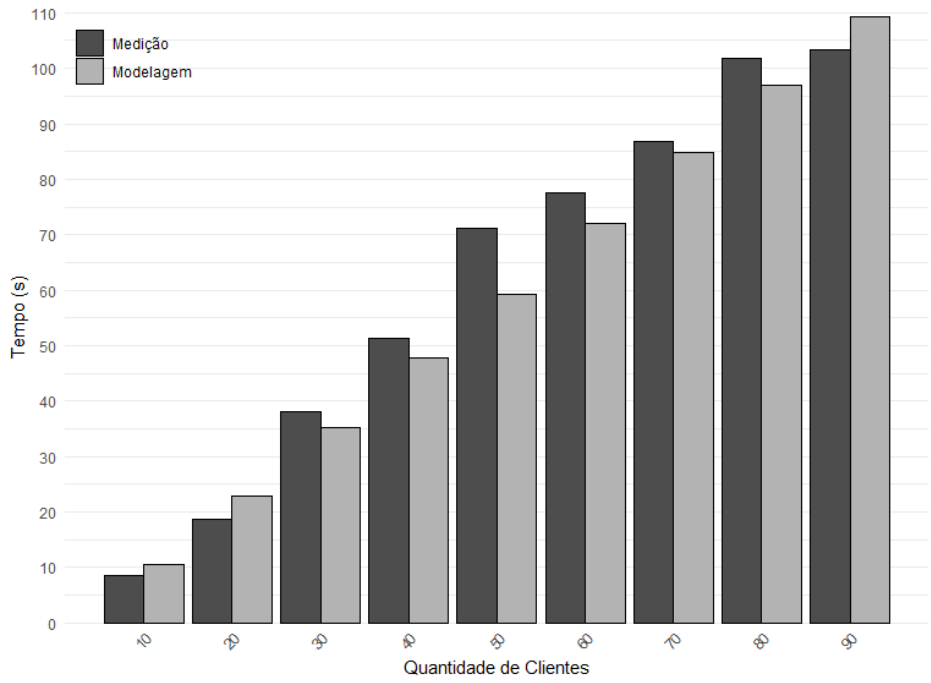


Figura 6.1: Resultado da medição e modelagem.

A fim de realizar a validação do modelo, aplicamos o teste-t pareado com os tempos de medição e os obtidos via modelagem. O teste t é um teste de hipótese e tem como objetivo comparar duas médias extraídas do mesmo grupo. Para a realização do teste t pareado é necessário validar alguns pressupostos.

O primeiro pressuposto é para verificar se as amostras seguem uma distribuição normal, então a hipótese nula seria que as médias são de uma distribuição normal e a hipótese alternativa que não são de uma distribuição normal. Para a realização de todos os testes será adotado um nível de significância de 5%. Utilizando o teste de Shapiro-Wilk (SHAPIRO; WILK, 1965), o p-valor encontrado para medição e modelagem são de 0,5114 e 0,9133, respectivamente. Como os p-valores foram maior que o nível de significância, então aceitamos a hipótese nula.

O segundo pressuposto é para avaliar se as variâncias das amostras são homogêneas (hipótese nula) ou não são homogêneas (hipótese alternativa). Realizando o teste de variâncias homogêneas, o resultado do p-valor foi de 0,9394, como o resultado foi maior que o nível de significância aceita-se a hipótese nula. Após a realização dos pressupostos é realizado o teste-t, onde a hipótese nula é se as médias são iguais e a hipótese alternativa se são diferentes. O p-valor para este teste foi de 0,2843, como o p-valor foi maior que o nível de significância, aceitamos a hipótese nula que não há diferença entre as médias das amostras.

6.2 Estudo Caso II

Esse estudo tem como objetivo utilizar o modelo proposto para a realizar projeções com uma maior quantidades de usuários realizando as atividades propostas. As simulações foram feitas de 100 a 600 clientes. Então o sistema vai atender 600 requisições com 100 clientes e no máximo 3600 requisições com 600 clientes. A Tabela 6.2 e a Figura 6.2 apresenta os resultados obtidos das projeções. O nível de confiança dos resultados é de 95% e o erro relativo da simulação 10%.

Tabela 6.2: Resultado das projeções com intervalo de confiança.

Qtd. Clientes	Tempo de Resp.(s)	Int. Conf.
100	132,38	[120,39; 140,01]
200	253,21	[241,05; 259,37]
300	403,94	[392,86; 415,34]
400	557,06	[460,78; 618,68]
500	751,70	[699,85; 803,49]
600	855,76	[810,61; 1001,90]

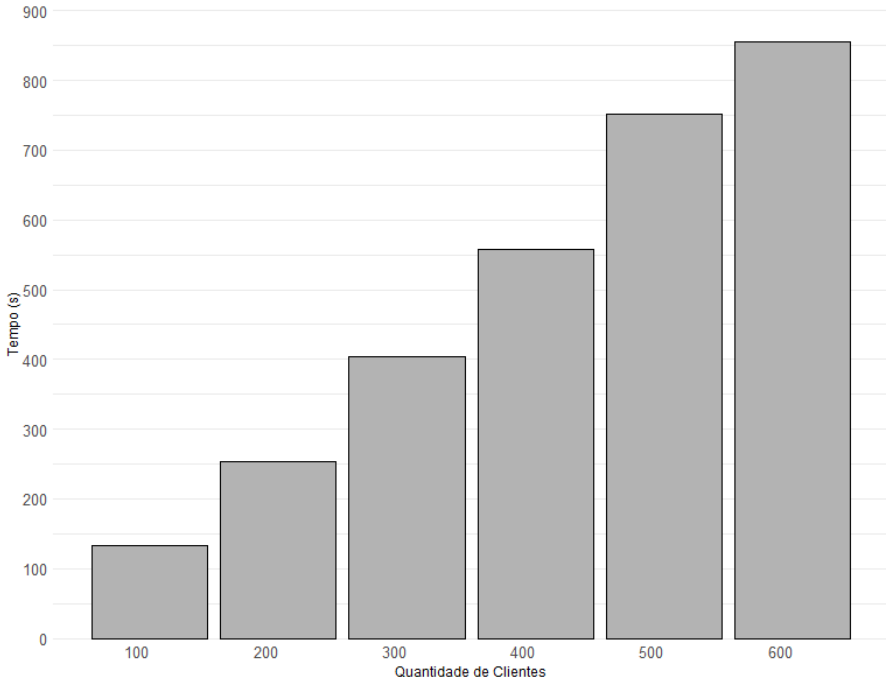


Figura 6.2: Resultado das projeções.

Como pode ser visto na Tabela 6.2 e na Figura 6.2 a medida que aumenta a quantidade de clientes simultâneos no mesmo sistema, o tempo de atendimento aumenta consideravelmente. Por exemplo, o tempo para realizar as atividades sequencialmente de acesso ao Nextcloud, realizar o *login*, acesso aos arquivos, realizar o *upload* de um arquivo e realizar o *logout* partiu de pouco mais de 2 minutos, cenário com 100 usuários, para acima de 14 minutos considerando 600 clientes simultâneos. Logo, nesse caso, pode ser inviável dado o tempo demandado de espera. Então, seria necessários ajustes na VM, como aumento de quantidades de processador ou aumento na quantidade de memória, para se ter um melhor resultado no tempo de atendimento.

6.3 Estudo de Caso III

Esse estudo tem como objetivo realizar a validação do modelo de disponibilidade proposto (ver Figura 5.10). Sendo assim, esse estudo é responsável por certificar que os valores gerados pelo modelo apresentam coerência com os gerados pelo sistema real. O modelo só será considerado válido, se os resultados da disponibilidade obtidos pelo modelo estiverem dentro do intervalo de confiança dos resultados obtidos por medições realizadas no ambiente real.

Como primeiro passo da validação, foi criado um *script* responsável por realizar as injeções de falhas e os reparos na infraestrutura do Cloudstack. Os tempos de falhas e reparos foram reduzidos por um fator de 100 com o objetivo de se acelerar o tempo em que ocorrem as falhas. Como parâmetros de entrada da análise do modelo, foi adotado o nível de confiança de 95% e o erro relativo em 10%. O tempo total do experimento realizado foi de um período de 164,31 horas (6,84 dias). A Tabela 6.3 apresenta os resultados do teste de injeção de falhas.

Tabela 6.3: Resultado teste de injeção de falha.

Parâmetro	Descrição	Valor
ε	Tempo total disponível (s)	588996
η	Tempo total indisponível (s)	2533
ω	Quantidade total de falhas	840
κ	Quantidade total de reparos	840

Para realizar a validação do modelo, foi preciso calcular o intervalo de confiança da disponibilidade. Para realizar esse cálculo foi utilizado o método de Keesee(KEESEE, 1965), onde se computa o ρ e α conforme a Equação 6.1 e Equação 6.2, respectivamente. Com o resultado de ρ e α é possível se calcular a disponibilidade do sistema conforme mostrado na Equação 6.3.

$$\rho = \frac{\varepsilon}{\omega} \tag{6.1}$$

$$\alpha = \frac{\eta}{\kappa} \tag{6.2}$$

$$\widehat{A} = \frac{\rho}{\rho + \alpha} \tag{6.3}$$

Ao serem realizados esses cálculos foi obtida a disponibilidade do sistema de 99,57%. Em seguida, deve-se calcular o limite máximo e mínimo da distribuição F com 95% de confiança. Para auxiliar na realização desse cálculo, pode-se utilizar a ferramenta Minitab (Minitab,), por exemplo. A distribuição dos dados obtidos via medição no sistema real utiliza um grau de liberdade determinado pelo somatório da quantidade de falhas e reparos do teste. A Figura 6.3 apresenta o gráfico da distribuição F com os valores máximo e mínimo como mostra a Tabela 6.4.

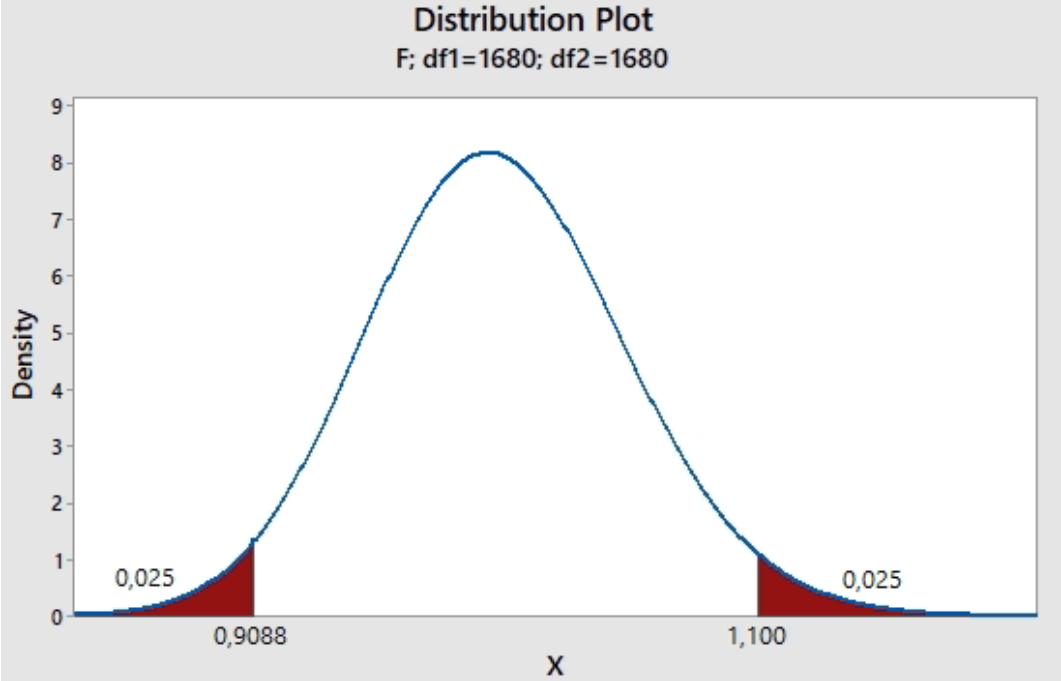


Figura 6.3: Gráfico da distribuição F.

Tabela 6.4: Limites da distribuição F.

Descrição	Valor
Limite mínimo	0,9088
Limite máximo	1,1

Após obter os limites máximo e mínimo, será calculado o γl e γu, conforme a Equação 6.4 e Equação 6.5.

$$\gamma l = (\frac{\eta}{\varepsilon}) \times lmin \tag{6.4}$$

onde $lmin$ é o limite mínimo.

$$\gamma u = (\frac{\eta}{\varepsilon}) \times lmax \tag{6.5}$$

onde $lmax$ é o limite máximo.

Com os valores de γl e γu, pode-se calcular o intervalo de confiança da disponibilidade dada pela Equação 6.6, que resultará na disponibilidade inferior, enquanto Equação 6.7 resultará no limite da disponibilidade superior.

$$A_u = \frac{1}{1 + \gamma u} \tag{6.6}$$

$$A_l = \frac{1}{1 + \gamma l} \tag{6.7}$$

A Tabela 6.5 apresenta os valores obtidos através dessas equações.

Tabela 6.5: Resultados de γ e A.

Descrição	Valor
γu	0,0047
γl	0,0039
A_u	0,9953
A_l	0,9961
Disponibilidade Modelo	99,60%
Disponibilidade Teste	99,57%

Para ocorrer a validação do modelo proposto é necessário que a disponibilidade do modelo esteja dentro do intervalo de confiança obtido pela disponibilidade no sistema real monitorado. Como a disponibilidade do modelo foi de 90,60% e o intervalo de confiança é de [99,53%; 99,61%], pode-se afirmar com 95% de confiança que o resultado obtido pelo modelo

corresponde ao do ambiente real.

6.4 Estudo de Caso IV

Esse estudo tem como objetivo comparar a disponibilidade da arquitetura base com duas arquiteturas com redundâncias. A primeira arquitetura implementa uma redundância da VM do Nextcloud, isso garante se houver uma parada de uma VM, o serviço continuaria disponível. A Figura 6.4 apresenta a arquitetura com uma redundância da VM Nextcloud.

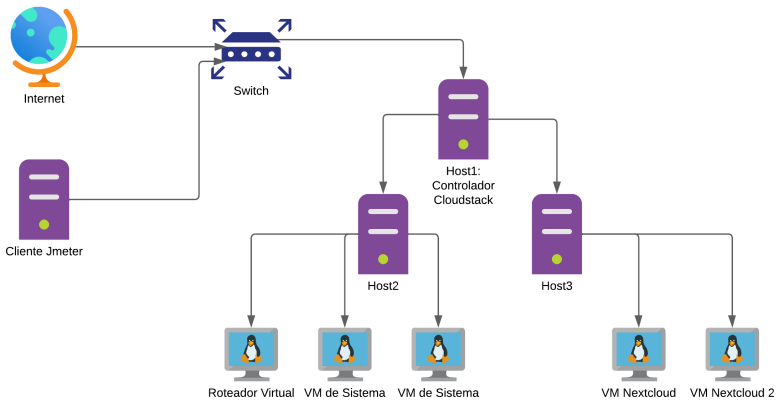


Figura 6.4: Arquitetura com uma redundância.

A Figura 6.5 apresenta o modelo de disponibilidade com uma redundância que representa a arquitetura apresentada na Figura 6.4. Então, será adicionado um *token* no lugar *VM-Next_ON* que representará a nova redundância do Nextcloud e as transições *MTTF_VMNext* e *MTTR_VMNext* mudarão de semântica e ficarão em *Infinite Server*.

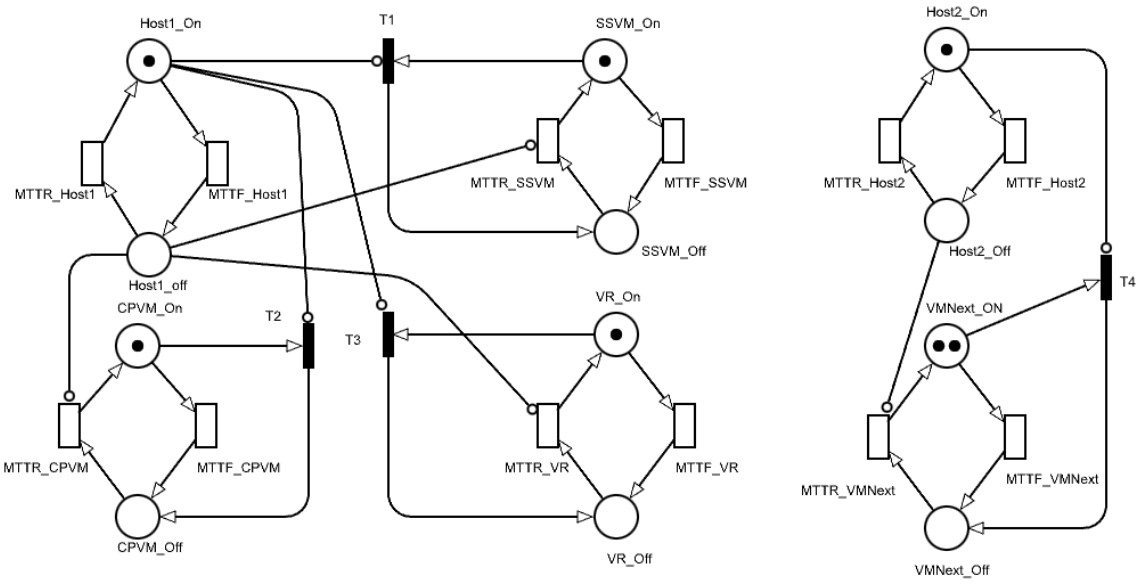


Figura 6.5: Modelo com uma redundância.

A segunda arquitetura implementa mais uma redundância da VM do Nextcloud, essa instância seria colocada no *Host1*. Com essa arquitetura, se houvesse queda do *Host2*, o serviço continuaria funcionando normalmente. A Figura 6.6 ilustra a arquitetura com duas redundâncias da VM Nextcloud.

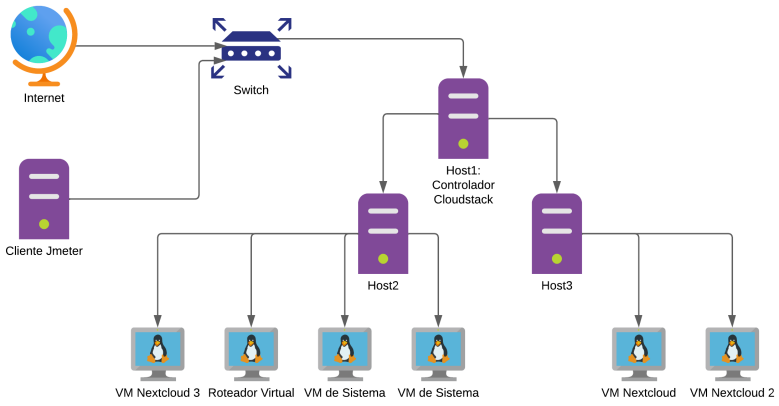


Figura 6.6: Arquitetura com duas redundância.

A Figura 6.7 apresenta o modelo de disponibilidade com duas redundâncias que representa a arquitetura apresentada na Figura 6.6. Então, será adicionado um *token* no lugar *VR_ON* que representará a segunda redundância do Nextcloud no *Host1* e as transições

MTTF_VR e *MTTR_VR* mudarão de semântica e ficarão em *Infinite Server*.

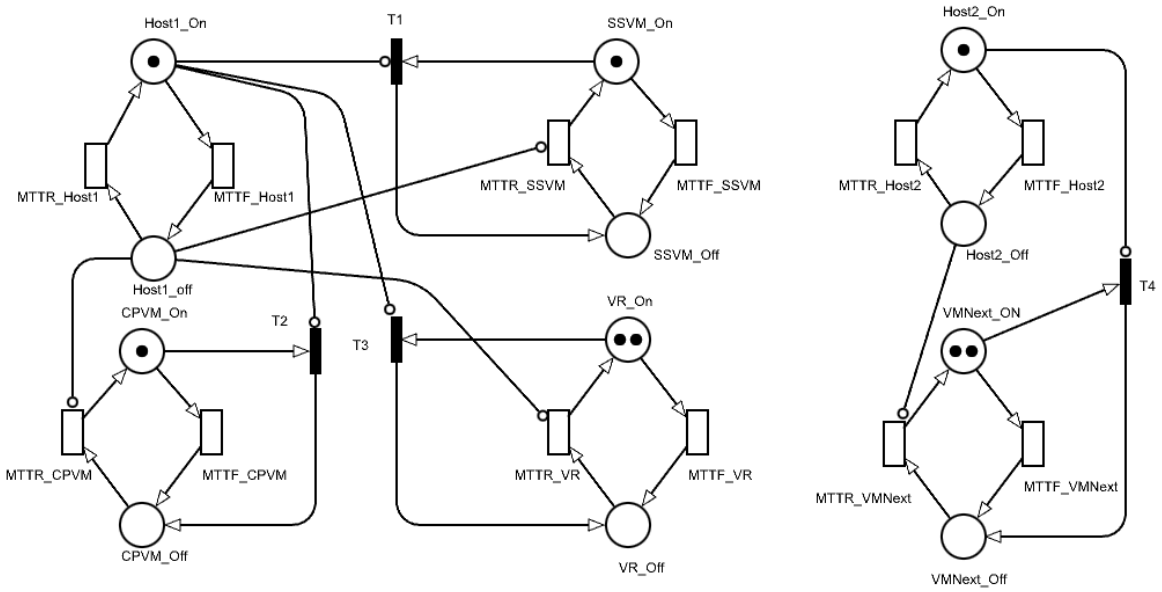


Figura 6.7: Modelo com duas redundâncias.

A Tabela 6.6 apresenta os resultados das três arquiteturas apresentadas. A arquitetura base teve uma disponibilidade de 99,60%, quando é aplicada uma redundância a disponibilidade aumenta para 99,77% e quando adicionarmos outra redundância a disponibilidade fica em 99,94%. Há também uma queda do *downtime* anual da arquitetura base de 34,40 horas para 20,15 horas com uma redundância e 5,25 horas com duas redundâncias. Comparando a arquitetura base com duas redundâncias, houve uma diminuição de 84,73% no downtime. A Figura 6.8 apresenta uma comparação da disponibilidade das três arquiteturas.

Tabela 6.6: Resultados da disponibilidade das arquiteturas.

Modelo	Disponibilidade (%)	9's	Downtime (h)
Arq. Base	99,60%	2,4	34,40
Uma redundância	99,77%	2,6	20,15
Duas redundância	99,94%	3,0	5,25

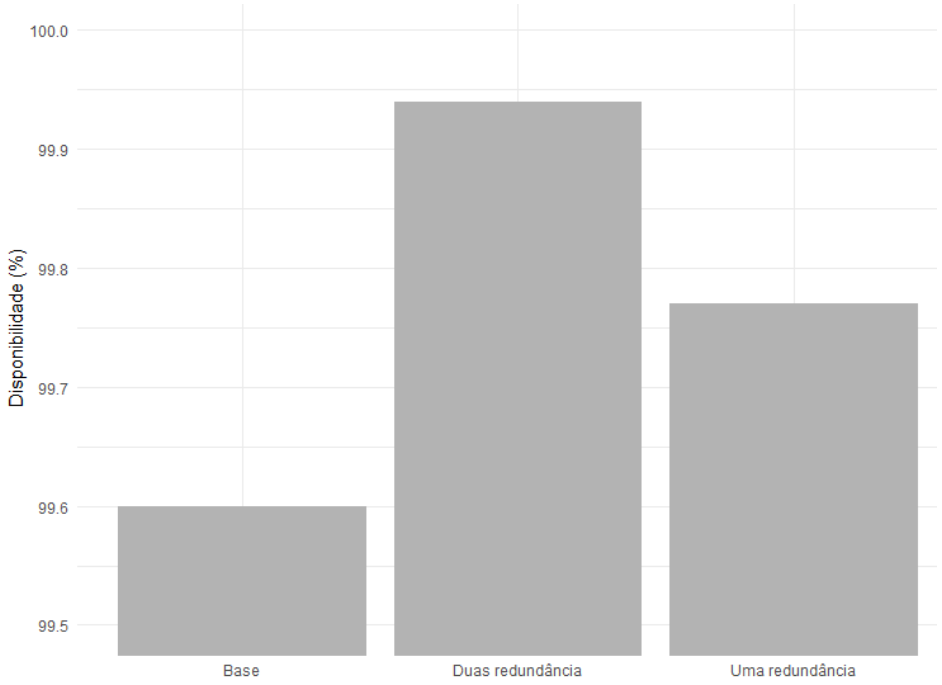


Figura 6.8: Disponibilidade das arquiteturas.

6.5 Estudo de Caso V

Esse estudo tem como objetivo comparar a disponibilidade de diversos tipos de Acordo de Nível de Serviço (SLA). Um SLA é um documento que define um conjunto de compromissos entre uma empresa prestadora de serviços e um cliente (CALLOU et al., 2010). Nesse estudo, vai ser utilizado três tipos de SLAs. O primeiro contrato terá uma melhoria de 20% nos tempos de MTTR dos *hosts*, o segundo vai melhorar em 40% e o último terá uma melhoria de 60% nos tempos de reparo. A Tabela 6.7 apresenta o resultado de disponibilidade obtidos com os diferentes contratos.

Tabela 6.7: Resultados da disponibilidade dos SLAs.

Cenários	Disponibilidade (%)	9's	Downtime (h)
Arq. Base	99,60%	2,4	34,40
SLA I	99,62%	2,4	33,28
SLA II	99,65%	2,4	30,65
SLA III	99,67%	2,4	28,90

Conforme apresentado na Tabela 6.7, podemos perceber um pequeno aumento nas dis-

ponibilidades do serviço e uma redução no *downtime* do serviço. O SLA I teve um *downtime* de 33,28 horas, o SLA II teve um *downtime* de 30,65 horas e o SLA III teve 28,90 horas. Comparando o SLA I com SLA II, houve um aumento da disponibilidade e uma redução no *downtime* de 7,90%. Já comparando SLA I com SLA III, que durante um ano o serviço ficará indisponível por quase 29 horas no SLA III, no SLA I o serviço ficará indisponível por mais de 33 horas durante o ano. Isso corresponde a uma redução de 13,16% no *downtime*. Se for comparar com a arquitetura base que teve um *downtime* de 34,40 horas com SLA III, haverá uma queda no tempo de quase 16%.

6.6 Estudo de Caso VI

Esse estudo tem como objetivo comparar a disponibilidade se houvesse troca dos equipamentos. Vão ser utilizados três equipamentos que possui um desempenho melhor do que o atual. Onde o primeiro equipamento melhora o tempo de MTTF dos hosts e VMs em 20%, o segundo melhora em 40% e o último em 60%. A Tabela 6.8 apresenta o resultado de disponibilidade obtidos com os diferentes equipamentos.

Tabela 6.8: Resultados da disponibilidade dos equipamentos.

Equipamentos	Disponibilidade (%)	9's	Downtime (h)
Arq. Base	99,60%	2,4	34,40
Equipamento I	99,69%	2,4	27,15
Equipamento II	99,75%	2,6	21,09
Equipamento III	99,80%	2,7	17,51

Como foi visto na Tabela anterior, com as trocas de equipamentos ocorre um ganho na disponibilidade. O equipamento I teve um *downtime* durante um ano de mais de 27 horas, enquanto o equipamento II ficou indisponível durante mais de 21 horas, isso corresponde em uma redução de 22,32%. Já o equipamento III teve um downtime de mais de 17 horas. Comparando com a arquitetura base que teve um *downtime* de 34,40 com o equipamento III, há uma redução de 49,09% no tempo que o serviço ficará indisponível.

6.7 Considerações Finais

Este capítulo apresentou os seis estudos de casos realizados para este trabalho. No primeiro estudo de caso foi realizado a validação do modelo de desempenho, foi utilizado teste t pareado com os tempos da medição e da modelagem. Foi realizado alguns pressupostos como a distribuição normal e o teste de variâncias homogêneas.

No segundo estudo de caso foi realizado projeções com o modelo de desempenho validado. Esse estudo visa projetar outras quantidades de usuários que não era possível efetuar na medição. O teste mostrou que a partir de uma determinada quantidade o sistema não consegue atender as requisições em tempo hábil.

No terceiro estudo de caso é realizado a validação do modelo de disponibilidade proposto. Para essa validação foi feito o teste de injeção de falha e com o seu resultado se utilizou o método Keesee, que obtém o intervalo de confiança do teste. Se a disponibilidade do modelo estiver dentro do intervalo de confiança do método de Keesee, o modelo está validado.

O quarto estudo de caso realizou a comparação da disponibilidade entre a arquitetura base, arquitetura com uma redundância e arquitetura com duas redundâncias. O objetivo desse estudo foi propor uma melhoria na disponibilidade da arquitetura base.

O último estudo de caso comparou três tipos de contratos de SLA. Onde o primeiro contrato teve uma melhoria no tempo de falha e reparo de 20%, o segundo 40% e o terceiro 60%.

Capítulo 7

Conclusão

A computação em nuvem está cada vez mais presente em empresas e instituições, pois pode fornecer serviços de alta disponibilidade com baixo custo. Isso pode ser visto em serviços de armazenamento, onde se pode guardar arquivos a qualquer momento e em vários tipos de plataformas. Então, a análise e modelagem desses sistemas ajuda a ter melhores decisões para se definir o que será feito para se ter um bom desempenho com alta disponibilidade.

Este trabalho propôs modelos de desempenho e disponibilidade do Nextcloud em nuvem privada. O primeiro modelo em SPN foi para avaliar o desempenho da arquitetura proposta. Foram realizadas medições do tempo de resposta de atividades feitas no Nextcloud e a validação do modelo proposto para representar o cenário de teste. Além disso, o modelo contempla regressões lineares realizadas em cada uma das atividades que representam os cenários de testes realizados. Sendo assim, foi possível também analisar outros cenários que foram inviáveis de medir, mostrando a validade do modelo na análise de cenários com maior quantidade de usuários no sistema.

O segundo modelo em SPN foi para avaliar a disponibilidade do serviço. Foi feito a validação do modelo proposto com a disponibilidade do modelo e com o resultado do teste de injeção, onde foi feito por meio do intervalo de confiança utilizando o método de Keesee. Também foi proposto duas arquiteturas com objetivo de melhorar a disponibilidade do serviço. E também analisou SLA, onde se melhorou os tempos de falhas e reparos para se obter uma melhor disponibilidade.

7.1 Contribuições

Esta seção apresenta as principais contribuições deste estudo:

- Foi proposto uma metodologia para a realização da avaliação de desempenho e avaliação da disponibilidade de servidor de arquivos Nextcloud em nuvem privada.
- Foi proposto um modelo de desempenho em SPN que simula o funcionamento do sistema Nextcloud e que calcula métricas de tempo de resposta e vazão.
- Foi proposto um modelo em RBD para a realizar a injeção de falha e obter o intervalo de confiança da disponibilidade do sistema e também foi feito um modelo em SPN para representar a infraestrutura do serviço.
- Foram propostas duas extensões da arquitetura base com redundância do serviço, tendo com objetivo o aumento da disponibilidade.

Além das contribuições mencionadas acima, tivemos um artigo publicado em uma conferência:

- Alison Silva, Gustavo Callou, Thiago Valentim e Jamilson Dantas, "Análise de Desempenho de Servidores de Arquivos em Nuvem Privada"no Seminário Integrado de Software e Hardware (SEMISH, 2023) do Congresso da Sociedade Brasileira de Computação (CSBC), João Pessoa, Brasil, 2023.

7.2 Trabalhos Futuros

Esta seção apresenta o que pode ser feito como trabalhos futuros.

- Como este trabalho utilizou o Apache Cloudstack como infraestrutura de nuvem privada, pode ser realizado em outras nuvens como Openstack, Eucalyptus e Open Nebula. Assim realizando uma comparação de desempenho e disponibilidade entres essas infraestruturas.

- Também pode ser realizado com outros tipos de nuvens, como nuvem pública ou híbrida.
- Integrar ao estudo o consumo de energia e ver qual entrega melhor desempenho ao baixo consumo.
- Utilizar o auto escalonamento de máquinas virtuais para garantir que trabalhem em níveis de desempenho aceitáveis.
- Analisar outros atributos de dependabilidade do ambiente proposto.

Referências Bibliográficas

AMAZON. *Amazon AWS*. 2022. Disponível em: ⟨https://aws.amazon.com/pt/⟩. páginas 1

AMAZON. *Amazon EC2*. 2023. Disponível em: ⟨https://aws.amazon.com/pt/ec2/⟩. páginas 13

AMORIM, D. *Brasil precisa com urgência de Marco de Cibersegurança e Soberania Digital, diz FGV*. 2023. CNN. Disponível em: ⟨https://www.cnnbrasil.com.br/economia/brasil-precisa-com-urgencia-de-marco-de-ciberseguranca-e-soberania-digital-diz-fgv⟩. páginas 3

APACHE. *Apache CloudStack*. 2022. Disponível em: ⟨https://cloudstack.apache.org/⟩. páginas 2, 13, 14, 15

APACHE. *Jmeter*. 2023. Disponível em: ⟨https://jmeter.apache.org/index.html⟩. páginas 17

ARMBRUST, M. et al. Above the clouds: A view of cloud computing: Technical report. 2010. páginas 1

ATAIE, E. et al. Hierarchical stochastic models for performance, availability, and power consumption analysis of iaas clouds. *IEEE Transactions on Cloud Computing*, v. 7, n. 4, p. 1039–1056, 2019. páginas 8, 10

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B. Fundamental concepts of dependability. *Department of Computing Science Technical Report Series*, Newcastle University, 2001. páginas 2, 27

BALBO, G. Introduction to stochastic petri nets. In: SPRINGER. *School organized by the European Educational Forum*. [S.l.], 2000. p. 84–155. páginas 45

BARROS, J. R. A. de. *Modelagem de Desempenho do Banco de Dados Cassandra*. Dissertação (Mestrado) — Universidade Federal Rural de Pernambuco, 2018. páginas 16

BAUER, E.; ADAMS, R. *Reliability and availability of cloud computing*. [S.l.]: John Wiley & Sons, 2012. páginas 1

BLUE, S. *Apache CloudStack*. 2018. Disponível em: ⟨https://www.shapeblue.com/pt-br/cloudstack-101-2/⟩. páginas 14

BRASCHER, G. B. *CloudStack in a few minutes.* 2019. The Developer's Conference. páginas 16

CALLOU, G. et al. Energy consumption and execution time estimation of embedded system applications. *Microprocessors and Microsystems*, Elsevier, v. 35, n. 4, p. 426–440, 2011. páginas 17

CALLOU, G. et al. Impact analysis of maintenance policies on data center power infrastructure. In: IEEE. *2010 IEEE international conference on systems, man and cybernetics*. [S.l.], 2010. p. 526–533. páginas 64

COSTA, I. et al. Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform. *Quality and Reliability Engineering International*, Wiley Online Library, v. 32, n. 7, p. 2191–2205, 2016. páginas 7, 10

DANTAS, J. et al. Estimating capacity-oriented availability in cloud systems. *International Journal of Computational Science and Engineering*, Inderscience Publishers (IEL), v. 22, n. 4, p. 466–476, 2020. páginas 1, 8, 10

DROPBOX. *Dropbox*. 2022. Disponível em: ⟨https://www.dropbox.com/pt_BR/⟩. páginas 13

GERMAN, R. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. New York, NY, USA: John Wiley & Sons, Inc., 2000. ISBN 0471492582. páginas 19

GOOGLE. *Google Apps*. 2022. Disponível em: ⟨https://gsuite.google.com.br/intl/pt-BR/⟩. páginas 13

GOOGLE. *Google Cloud*. 2023. Disponível em: ⟨https://cloud.google.com/?hl=pt-br⟩. páginas 1, 13

GOOGLE. *Google Cloud Plataform*. 2023. Disponível em: ⟨https://edu.google.com/intl/pt-BR/products/google-cloud-platform/⟩. páginas 13

GUJARATI, D. N.; PORTER, D. C. *Econometria básica /*. 5. ed.. ed. Porto Alegre :: AMGH,, 2011. Tradução de: Basic econometrics. páginas 40

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. *Computer*, IEEE, v. 30, n. 4, p. 75–82, 1997. páginas 27

JOHN, L. K.; EECKHOUT, L. *Performance evaluation and benchmarking*. [S.l.]: CRC Press, 2018. páginas 17

KEESEE, W. A method of determining a confidence interval for availability. In: *Misc. Publication No. NMC-MP-65-8*. [S.l.]: US Naval Missile Center, 1965. páginas 28, 34, 58

KUMAR, R. et al. Apache cloudstack: Open source infrastructure as a service cloud computing platform. *Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science*, p. 111–116, 2014. páginas 14

KUO, W.; ZUO, M. J. *Optimal reliability modeling: principles and applications*. [S.l.]: John Wiley & Sons, 2003. páginas 45

LILJA, D. J. *Measuring Computer Performance: A Practitioner's Guide*. [S.l.]: Cambridge University Press, 2000. páginas 23

LIMA, C. J. d. et al. Performance and power consumption evaluation of the moodle environment. *Research, Society and Development*, v. 10, n. 5, p. e57010515191, May 2021. Disponível em: ⟨https://rsdjournal.org/index.php/rsd/article/view/15191⟩. páginas 6, 10

LOPES, R. *E-mails e arquivos limitados: Google modifica contrato de armazenamento digital com a UFF*. 2023. O Globo. Disponível em: ⟨https://oglobo.globo.com/rio/bairros/niteroi/noticia/2023/01/e-mails-e-arquivos-limitados-google-encerra-contrato-de-armazenamento-digital-com-a-uff.ghtml⟩. páginas 2

MACIEL, P. *Performance, Reliability, and Availability Evaluation of Computational Systems, Volume I: Performance and Background*. [S.l.]: Chapman and Hall/CRC, 2023. páginas 21, 22

MACIEL, P.; LINS, R. D.; CUNHA, P. R. F. Introdução às redes de petri e aplicações. 05 1996. páginas 18

MACIEL, P. R. M. Modeling availability impact in cloud computing. In: *Principles of Performance and Reliability Modeling and Evaluation*. [S.l.]: Springer, 2016. p. 287–320. páginas 47, 49, 52

MACIEL, P. R. M. et al. Dependability modeling. In: CARDELLINI, V.; CASALICCHIO, E.; NETTO, M. A. S. (Ed.). *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*. IGI Global, 2012. p. 53–97. Disponível em: ⟨https://doi.org/10.4018/978-1-60960-794-4.ch003⟩. páginas 25

MATOS, M. A. Manual operacional para a regressão linear. *Faculdade de Engenharia da Universidade do Porto*, p. 63, 1995. páginas 22

MATOS, R. et al. Bottleneck detection in cloud computing performance and dependability: Sensitivity rankings for hierarchical models. *Journal of Network and Systems Management*, v. 28, 10 2020. páginas 7, 10

MELL, T. G. P. *NIST Cloud Computing Program*. 2014. Disponível em: ⟨http://www.nsti.gov/itl/cloud/index.cfm⟩. páginas 12, 13

MELO, C. et al. Capacity-oriented availability model for resources estimation on private cloud infrastructure. In: IEEE. *2017 IEEE 22nd Pacific rim international symposium on dependable computing (PRDC)*. [S.l.], 2017. p. 255–260. páginas 7, 10

MICROSOFT. *Microsoft Office 365*. 2022. Disponível em: ⟨https://www.office.com/⟩. páginas 13

MICROSOFT. *Microsoft Azure*. 2023. Disponível em: ⟨https://azure.microsoft.com/pt-br/⟩. páginas 1, 13

Minitab. *Minitab*. Disponível em: ⟨https://www.minitab.com/pt-br/⟩. páginas 59

MOREIRA, C. et al. Compartilhamento de arquivos em home office: uma solução de armazenamento de arquivos segura e altamente disponível em um ambiente de nuvem privada. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, v. 01, p. 409–424, 02 2021. páginas 5, 10

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, April 1989. ISSN 0018-9219. páginas 18, 19

OPENSTACK. *OpenStack*. 2022. Disponível em: ⟨https://www.openstack.org/⟩. páginas 2

PALMEIRA, C. *Google encerra Drive ilimitado e universidades buscam alternativas*. 2021. Tecmundo. Disponível em: ⟨https://www.tecmundo.com.br/ciencia/216882-google-encerra-nuvem-ilimitada-universidades-buscam-alternativas.htm⟩. páginas 2

PINHEIRO, T. et al. Performance and resource consumption analysis of elastic systems on public clouds. In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. [S.l.: s.n.], 2019. p. 2115–2120. páginas 7, 10

REDHAT. *Red Hat OpenShift*. 2022. Disponível em: ⟨https://www.redhat.com/pt-br/technologies/cloud-computing/openshift⟩. páginas 13

ROCHA, et al. Analyzing the impact of power infrastructure failures on cloud application availability. In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2017. p. 1746–1751. páginas 8, 10

RUSCHEL, H.; ZANOTTO, M. S.; MOTA, W. d. Computação em nuvem. *Curitiba, abr*, p. 1–3, 2010. páginas 12

SANTOS, D. L.; CAMBOIM, K.; ALENCAR, F. Avaliação de desempenho de nuvens privadas: Um comparativo entre owncloud, nextcloud e pydio. In: *Anais do XVI Workshop em Clouds e Aplicações*. Porto Alegre, RS, Brasil: SBC, 2018. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/wcga/article/view/2373⟩. páginas 5, 10

SANTOS, G. L. et al. Analyzing the it subsystem failure impact on availability of cloud services. In: *2017 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.: s.n.], 2017. p. 717–723. páginas 8, 10

SHAHZADI, S. et al. Infrastructure as a service (iaas): A comparative performance analysis of open-source cloud platforms. In: IEEE. *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. [S.l.], 2017. p. 1–6. páginas 6, 10

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, JSTOR, v. 52, n. 3/4, p. 591–611, 1965. páginas 33, 56

SILVA, B. A framework for availability, performance and survivability evaluation of disaster tolerant cloud computing systems. Universidade Federal de Pernambuco, 2016. páginas 19

SNEDECOR, G. W.; COCHRAN, W. G. Book. *Statistical methods / George W. Snedecor, William G. Cochran*. 6th ed.. ed. [S.l.]: Iowa State University Press Ames, 1967. xiv, 593 p. : p. páginas 33

STUDENT. The probable error of a mean. *Biometrika*, JSTOR, p. 1–25, 1908. páginas 33

TORQUATO, M.; UMESH, I. M.; MACIEL, P. Models for availability and power consumption evaluation of a private cloud with vmm rejuvenation enabled by vm live migration. *The Journal of Supercomputing*, v. 74, p. 1–25, 09 2018. páginas 7, 10

TORRES, E. et al. Performance and availability evaluation of storage services in private cloud. In: *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2016. p. 1–6. páginas 1, 6, 10, 17

TRIVEDI, K. S. et al. *Reliability Analysis Techniques Explored Through a Communication Network Example*. 1996. páginas 21, 25

VASCONCELOS, B. J. R. de. *Modelos para avaliação de disponibilidade e cálcuolo de capacidade de um software de compressão de vídeo distribuído em nuvem openstack*. 2019. páginas 28

VENTURA, F. *Amazon AWS fica fora do ar e derruba Alexa, Roku e outros serviços*. 2020. Tecnoblog. Disponível em: ⟨https://tecnoblog.net/noticias/2020/11/25/aws-da-amazon-fica-fora-do-ar-e-derruba-servicos-novembro-2020/⟩. páginas 1

VINHA, F. *Amazon AWS está fora do ar e derruba iFood, Disney+, LoL e outros*. 2021. Tecnoblog. Disponível em: ⟨https://tecnoblog.net/noticias/2021/12/07/amazon-aws-esta-fora-do-ar-e-derruba-ifood-disney-lol-e-outros/⟩. páginas 1

VOGEL, A. et al. An intra-cloud networking performance evaluation on cloudstack environment. In: *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. [S.l.: s.n.], 2017. p. 468–472. páginas 6, 10

ZIADE, H. et al. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, v. 1, n. 2, p. 171–186, 2004. páginas 27, 28

Apêndice A

Script Injeção de Falha

Este apêndice apresenta o *script* utilizado para injeção de falha, onde foi desenvolvido em linguagem Python. Primeiramente, tem que ser definidos os blocos do modelo com seu respectivo MTTF e MTTR que serão as *threads* no *script*. O código tem quatro funções que são: *ttf*, *ttr*, *connectionDown* e *connectionUp*. A função *ttf* irá receber um parâmetro de tempo de falha e gerar um valor exponencial aleatório de tamanho máximo do parâmetro recebido, já a função *ttr* realizará o mesmo código da função *ttf*, só que realizando para o tempo de reparo. A função *connectionDown* vai receber dois parâmetros, o primeiro vai ser o nome do bloco e o segundo será o tempo de falha e assim será realizado o corte na conexão da placa de rede do bloco pelo determinado tempo gerado na função *ttf*. Já a função *connectionUp* tem o mesmo objetivo da função *connectionDown*, onde realizará o reparo a conexão da placa de rede.

```
import random
import math
import time
import os
import numpy
from threading import Thread
from datetime import datetime

# arquivo de armazenamento do log de falhas e reparos realizados para comparar
    com o monitoramento do ambiente
```

```
nome_arq = 'fail_repair_log.csv'
arq = open(nome_arq, 'a+')
arq.write('Reparo;'+datetime.now().strftime('%d/%m/%Y %H:%M:%S')+';''\n')
arq.close()

# inicializacao de variavel. Tempos de down (quando falha) e up (quando e
    reparado)
time_down = 0
time_up = 0

# configuracao de cada bloco do modelo (bloco,mttf,mttr)
blocks = [('host1',4.1523,0.0025),('host2',12.5903,0.007685)]

# Calculo do TTF (Time To Fail), ou seja, quanto tempo o link ira falhar.
def ttf(timetofail):
    ft = numpy.random.exponential(timetofail)
    # converte em segundos
    time_down = (ft * 360)
    return time_down

# Calculo do TTR (Time To Repair), ou seja, quanto tempo o link ira voltar a
    ter conectividade.
def ttr(timetorepair):
    ft = numpy.random.exponential(timetorepair)
    # converte em segundos
    time_up = (ft * 360)
    return time_up

def connectionDown(a, time_down):
    # Corta a conexao de rede.
    os.system('echo ifconfig cloudbr0 down')
    arq = open(nome_arq, 'a+')
    arq.write('Falha;'+datetime.now().strftime('%d/%m/%Y %H:%M:%S')+';'+str(a)
        +';'+str(time_down)+'\n')
    arq.close()

# A interface de rede deve estar com IP fixo.
def connectionUp(a, time_up):
    # Corta repara a conexao de rede.
```

```
    os.system('echo ifconfig cloudbr0 up')
    arq = open(nome_arq, 'a+')
    arq.write('Reparo;'+datetime.now().strftime('%d/%m/%Y %H:%M:%S')+';'+str(a
        )+';'+str(time_up)+'\n')
    arq.close()

# executa a falha ou reparo em funcao do parametro do block com tempos
    exponencialmente distribuidos
def control(a,b,c):
    ttf_total_block = 0
    ttr_total_block = 0
    while True:
        # Fail
        time_down = ttf(b)
        ttf_total_block = ttf_total_block + time_down
        time.sleep(time_down)
        print(str(ttf_total_block)+"  TTF Total "+str(a))
        connectionDown(a, time_down)

        # Repair
        time_up = ttr(c)
        ttr_total_block = ttr_total_block + time_up
        time.sleep(time_up)
        print(str(ttr_total_block)+"  TTR Total "+str(a))
        connectionUp(a, time_up)


def main():
    # Tempos acumulados para o calculo de disponbilidade
    try:
        print('Inicio: ' + datetime.now().strftime('%d/%m/%Y %H:%M:%S'))
        t0 = Thread(target=control, args=(blocks[0]))
        t0.start()
        t1 = Thread(target=control, args=(blocks[1]))
        t1.start()
    except KeyboardInterrupt as e:
        print ('Fim: ' + datetime.now().strftime('%d/%m/%Y %H:%M:%S'))
```

```
if __name__ == '__main__':
    main()
```