Fabio Amorim da Silva

Falhas em Ambiente de Nuvem

São José do Rio Preto

2020

Fabio Amorim da Silva

Falhas em Ambiente de Nuvem

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista ”Júlio de Mesquita Filho”, como parte dos requisitos necessários para obtenção do título de "Bacharel" em Ciência da Computação da UNESP

Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)

Instituto de Biociências, Letras e Ciências Exatas (IBILCE)

Bacharelado em Ciência da Computação

Profa. Dra. Renata Spolon Lobato

São José do Rio Preto

2020

Fabio Amorim da Silva

Falhas em Ambiente de Nuvem

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista ”Júlio de Mesquita Filho”, como parte dos requisitos necessários para obtenção do título de "Bacharel" em Ciência da Computação da UNESP

Profa. Dra. Renata Spolon Lobato Fabio Amorim da Silva

Banca Avaliadora:

Prof. Dr. Aleardo Manacero Junior

Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto
2020

S586f

Silva, Fabio Amorim da

Falhas em Ambiente de Nuvem / Fabio Amorim da Silva. -- São José do Rio Preto, 2020

46 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientadora: Dra. Renata Spolon Lobato

1. Computação em Nuvem. 2. Simulação. 3. Falhas. 4. Métodos de Tratamento de Falhas. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Dedico esse trabalho à minha mãe Maria.

Agradecimentos

Agradeço a Deus por me dar forças e motivação durante todos estes anos de graduação mesmo nos momentos mais difíceis. Agradeço aos membros da minha família pelo suporte emocional, e principalmente à minha mãe Maria Ivonete que sempre se preocupou com minha saúde física e emocional as vezes até em detrimento da própria durante todo este tempo de graduação e ao longo da sua vida.

Agradeço a Prof. Dra. Renata Spolon Lobato pela orientação no projeto principalmente neste ano tão atípico para todo o mundo e o auxílio na busca por novos conhecimentos e preocupação para com a minha pessoa; agradeço também ao Prof. Dr. Aleardo Manacero Junior pela co-orientação durante todos esses anos dentro do grupo de pesquisa e a sua amizade também.

Agradeço as todos que pude conhecer dentro do GSPD, por sua amizade, conhecimento cedidos e apoio durantes os anos que convivemos dentro do laboratório - Lucas Afonso Cazarote, Mario Camara Neto, Bruno Navarro, Renan Miguel Retondo, Luís Vinícius, Camila Tamashiro Okado, Diogo Tavares, Fernanda Peronaglio, Gabriel Covello, Victor Hugo e Renan Alboy.

Por fim, agradeço à UNESP pelo apoio e infraestrutura cedidas para que este trabalho fosse realizado.

.

"Esquecer é uma necessidade. A vida é uma lousa, em que o destino, para escrever um novo caso, precisa de apagar o caso escrito.

."

- Machado de Assis

Resumo

A computação em nuvem é uma área em grande expansão nos anos recentes e tornou-se parte indispensável para uma grande parcela da população mundial, mesmo para aqueles que não desenvolvem suas atividades na área de tecnologia. Um grande desafio nessa área envolve o planejamento de sistemas de forma que possamos reduzir o custo de implementação e maximizar a eficiência ao mesmo tempo, para isso as simulações são de suma importância no momento em que se deseja utilizar a computação em nuvem. Dentro das simulações algo muito buscado é a capacidade de simular falhas que podem ocorrer em ambientes reais acarretando em perdas significativas tanto de tempo quanto financeiramente um tipo de situação que não é desejada por nenhum usuário, empresa, entre outros que fazem uso de sistemas de computação em nuvem. Portanto, esse trabalho propõe a implementação de novas falhas dentro do sistema injetor de falhas do simulador de nuvem iSPD para que sua capacidade de simulação seja aprimorada para que novos usuários possam fazer uso de maneira mais proveitosa da ferramenta.

Palavras-chaves: Computação em nuvem, Simulação, Falhas, Sistema injetor, iSPD.

Abstract

Cloud computing is a area in great expansion in the recently years and have begin to be indispensable for great majority of world population, even for who do not work directly with the technological area. One of the great challenge in this field wrap the planning of systems in the manner to decrease costs of implementation and maximize the efficient at the same time, for this the simulations are very important in the moment the cloud computing is choose to make part of the project. In the context of this simulations something highly sought after is the fault injection capacity which can occur in the real environments that can lead to great losses financially something not requested for any user. Therefore, this work proposes the implementation of new faults in the fault injection system for the cloud simulator iSPD, aiming for the capacity to make the simulations in more enhanced manner for the new users can make a better use this tool.

keywords: Simulation, Cloud Computing, Faults, iSPD.

Lista de ilustrações

Figura 2.1 – Exemplo de computação em nuvem 15
Figura 2.2 – Arquiteturas de nuvem 17
Figura 2.3 – Arquitetura do Openstack 22
Figura 2.4 – Arquitetura do Opennebula 23
Figura 2.5 – Diagrama de interação dos módulos do iSPD 25
Figura 2.6 – Interface principal iSPD 26
Figura 2.7 – Propagação de uma Falha 28
Figura 2.8 – Fluxo de Falhas 29
Figura 3.1 – Fluxograma de Simulação do iSPD 31
Figura 3.2 – Fluxograma da identificação e tratamento de falha 32
Figura 3.3 – Interface de inserção de falhas iSPD 32
Figura 3.4 – Diagrama da Máquina de Estados da Modelagem das Falhas 33
Figura 3.5 – Inserção de falha durante da simulação 34
Figura 3.6 – Ciclo de deteccção e tratamento de falha no iSPD 36
Figura 4.1 – Desempenho dos grupos 41

Lista de abreviaturas e siglas

GSPD	Grupo de Sistemas Paralelos e Distribuídos
iSPD	iconic Simulator for Parallel and Distributed Systems
IaaS	Infrastructure as a Service
PaaS	Plataform as a Service
SaaS	Software as a Service
AWS	Amazon Web Services
VPN	Virtual Private Network
CPU	Central Processor Unit
REST	Representational State Transfer
API	Application Programming Interface
XML	Extensible Markup Language

Sumário

1 INTRODUÇÃO 12
1.1 Objetivos 12
1.2 Motivação 12
1.3 Justificativa 13
1.4 Metodologia 13
1.5 Exequibilidade 13
1.6 Organização do texto 14

2 REVISÃO BIBLIOGRÁFICA 15
2.1 Computação em Nuvem 15
2.2 Características da Computação em Nuvem 16
2.2.1 Tipos de Nuvem 16
2.2.2 Serviços de Nuvem 18
2.3 Provedores de serviços de nuvem 19
2.3.1 Microsoft Azure Service Plataform 19
2.3.2 Amazon Web Services 20
2.3.3 IBM Cloud 20
2.4 Gestores de Computação em Nuvem 21
2.4.1 OpenStack 22
2.4.2 OpenNebula 23
2.4.3 VMware vRealize Suite 23
2.5 iSPD 24
2.5.1 Estado atual 26
2.6 Falhas em Nuvem 27
2.6.1 Modelos de falha 27
2.7 Considerações finais 30

3 DESENVOLVIMENTO 31
3.1 Apresentação e organização geral do iSPD 31
3.2 Injetor de Falhas iSPD 32
3.3 Injeção e tratamento de falhas no iSPD 34
3.4 Considerações Finais 36

4 TESTES E RESULTADOS 37
4.1 Grupos e testes 37
4.2 Considerações finais 41

5 CONCLUSÕES E TRABALHOS FUTUROS 42
5.1 Conclusões 42
5.2 Trabalhos Futuros 42

REFERÊNCIAS 44

1 Introdução

Com o avanço da computação novos tipos de tecnologia tem surgido para atender as mais variadas demandas e situações ao redor do mundo. Uma das principais áreas que teve origem dessa evolução foi a computação em nuvem que permitiu a utilização de recursos de acordo com as escolhas e preferências dos clientes. Esses serviços podem ser de multimídia, armazenamento, processamento, banco de dados, entre outros. Isso gerou um ganho enorme para as empresas que puderam diminuir seus custos com equipamentos de *hardware*, além do aumento de produtividade com a terceirização de serviços de computação.

Um passo importante para decidir a implementação de um serviço de nuvem é analisar sua viabilidade e custo, assim serviços de simulação tornaram-se essenciais para avaliar o risco de um investimento do tipo. Entre os dados que podem ser retirados de uma simulação as falhas estão entre os mais importantes, pois com uma boa identificação e meios de tratamento um determinado projeto pode ter seu custo reduzido e sua segurança certificada algo de suma importância para um cliente.

Com os constantes aumentos na complexidade e nos riscos envolvendo os projetos a utilização de simulação com a possiblidade de verificação e tratamento de falhas tornou-se essencial para qualquer cliente que deseje ter confiança e tranquilidade na hora de contratar um serviço de nuvem.

1.1 Objetivos

O objetivo deste trabalho é realizar a implementação de novas falhas e técnicas de tratamento de falhas no iSPD (MANACERO A.; MENEZES, 2012), isso é necessário para que os usuários possam ter mais dados para efetuar simulações com mais precisão e de acordo com suas necessidades. Como a ferramenta pode ser utilizada tanto por alunos, professores, pesquisadores, entre outros torna-se de grande importância a inserção de novas falhas e tratamentos de falha para expansão de suas possibilidades de ação.

1.2 Motivação

O atual sistema de injeção e tratamento de falhas do iSPD (TAMASHIRO C. B. O.; LOBATO, 2020) possui poucas falhas e tipos de tratamentos disponíveis para escolhas dos usuários por ter sido implementado recentemente. Então é necessário que novas falhas e técnicas de tratamento sejam inseridas para que os usuários possam optar por modelarem a simulação em ambiente de nuvem da maneira mais adequada aos seus gostos e preferências.

1.3 Justificativa

Quando projeta-se um sistema o cuidado com os possíveis erros é essencial para que o plano de implementação tenha o menor custo e seja concluído no menor tempo possível. Usando simulações antes de uma implementação de fato é algo de extrema importância nos ambientes mais modernos. Utilizando injeção de falhas e tratamento falhas podemos averiguar todos os contratempos eventuais possam ocorrer no projeto real, assim evitando quaisquer disperdícios de recursos sendo eles financeiros ou humanos.

1.4 Metodologia

A metodologia adotada para realização do trabalho consiste em diversas etapas, sendo elas: pesquisa sobre o tema escolhido e o simulador de nuvem iSPD, especificação do funcionamento de cada módulo e funcionalidades da ferramenta e análise de falhas e tratamento para inserção na aplicação, implementação e a realização de testes e análise dos resultados obtidos. Tendo como objetivo a efetuação do trabalho proposto foi necessário a execução das etapas citadas anteriormente, primeiro um estudo sobre a computação em nuvem foi realizado, serviços e gestores de nuvem e a ferramenta iSPD, o material utilizado foi retirado principalmente via da internet através artigos científicos, sites, documentação online, entre outros meios gratuítos e de fácil acesso.

Um estudo aprofundado foi realizado na arquitetura, módulos e funcionamento do iSPD para que foi possível compreender todo o funcionamento e particularidades da ferramenta afim de propor o melhor método para inserção de novas falhas e tratamentos de falha no módulo de injeção de falhas. Para a realização dessa parte a linguagem de programação Java e o ambiente de desenvolvimento integrado Netbeans serão utilizados.

1.5 Exequibilidade

Para a realização desse trabalho serão utilizados os recursos de *hardware* do próprio autor já que o *software* pode ser executado localmente e não necessita de uma máquina específica para execução, além disso a linguagem de programação Java será usada para realização do trabalho pois é nela que o programa encontra-se escrito. Os itens citados anteriormente podem ser obtidos de forma gratuita, assim podem ser usados sem preocupações com custos e isso aliado com as informações citadas acima o trabalho proposto pode ser efetuado dentro do prazo estipulado sem maiores preocupações com equipamentos ou outros tipos de recursos.

1.6 Organização do texto

Este texto está dividido em cinco capítulos, além desse capítulo introdutório, o capítulo 2 contendo parte de revisão bibliográfica e estado atual da arte, nele são detalhados os seguintes tópicos: detalhamento sobre a computação em nuvem, descrição dos tipos de nuvem, serviços de nuvem, provedores e gestores de serviços de nuvem e a descrição de organização e funcionamento do iSPD. O objetivo desse capítulo é descrever todos os tópicos necessários ao leitor para entendimento sobre o trabalho e os assuntos correlatos. O capítulo 3 apresenta a descrição das atividades que serão desenvolvidas e como serão feitas, o quarto é a parte de apresentação dos testes e resultados obtidos para validação do trabalho proposto. Finalizando, o capitulo 5 irá mostrar a conclusão do trabalho com base nos dados colhidos e a proposição de trabalhos futuros para melhoramento da ferramenta.

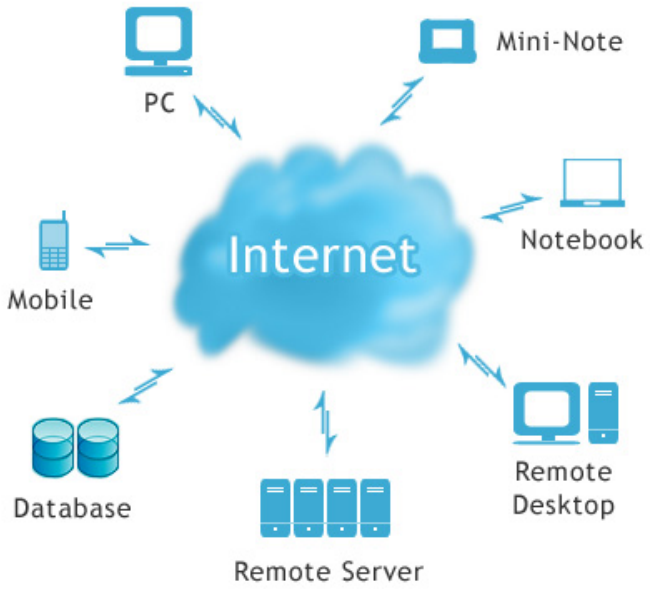
2 Revisão Bibliográfica

Neste capítulo serão apresentados e definidos os conceitos necessários para entendimento e compreensão deste trabalho. Como por exemplo: computação em nuvem, tipos de nuvem, serviços de nuvem, falhas em ambiente de nuvem, principais falhas na nuvem computacional, apresentação do iSPD, sua arquitetura e tecnologias envolvidas.

2.1 Computação em Nuvem

Com a evolução da computação e a necessidade de utilização de serviços cada vez mais especializados foi necessária a criação de uma nova forma de sanar essas novas demandas. A computação em nuvem surgiu como uma forma de prover serviços *on demand* para cada necessidade especifíca, ou seja, o cliente pode contratar o apenas o serviço que for fazer uso e pelo tempo adequado aos seus interesses (ARMBRUST M.; FOX, 2010), assim facilitando uso de tecnologias que não estariam ao seu alcance normalmente e reduzindo os custos de uso. Na Figura 2.1 temos um exemplos do que é a computação em nuvem.

Figura 2.1 – Exemplo de computação em nuvem



Fonte: https://www.techtudo.com.br/artigos/noticia/2012/03/o-que-e-cloud-computing.html

As melhorias que decorreram foram muitas e foram necessárias novas definições e separações para o novo sistema criado e as tecnologias que o compõem. Assim surgiram os tipos de

nuvem (pública, privada, híbrida) (AZUREMICROSOFT, 2020) e serviços *Infrastructure as a Service* (IaaS), *Plataform as a Service* (PaaS), *Software as a Service* (SaaS) (AMAZONSERVICES, 2020) que estão sendo expandidos de forma bem acelerada nos anos mais recentes.

2.2 Características da Computação em Nuvem

A computação em nuvem tem como objetivo permitir que os seus usuários possam acessar serviços conforme sua necessidade, assim essa tecnologia apresenta um conjunto de características que são necessárias para o bom funcionamento dos serviços oferecidos. As principais caracteristícas serão apresentadas e exemplificadas abaixo:

- Escalabilidade: é a capacidade que um determinado sistema tem de se expandir para atender as necessidades dos usuários e administradores. Essa capacidade pode ser limitada por recursos técnicos ou financeiros, o que pode ser variável em cada sistema.
- Disponibilidade: o tempo que o serviço fica disponível para uso pelos usuários. Um alto nível de disponibilidade é a meta sempre buscada em sistemas que utilizam a computação em nuvem.
- Confiabilidade: A capacidade que um sistema tem de se recuperar de condições anormais, ou seja, recuperação de erros sendo estes os mais diversos (queda de energia, comunicação, codificação,*hardware*, etc).
- Uso por demanda: A computação em nuvem permite que os usuários possam utilizar apenas o serviço de interesse e pelo tempo que for adequado, assim os clientes podem ter liberdade para fazer uso dos recursos conforme sua necessidade.

2.2.1 Tipos de Nuvem

Com a complexidade da computação em nuvem aumentando foi necessário criar maneiras mais eficientes para implementar a tecnologia, para isso foram criados tipos de nuvem computacional de modo a facilitar sua utilização conforme os objetivos buscados em cada caso. Basicamente, existem quatro tipos principais de nuvens computacionais (REDHAT, 2020):

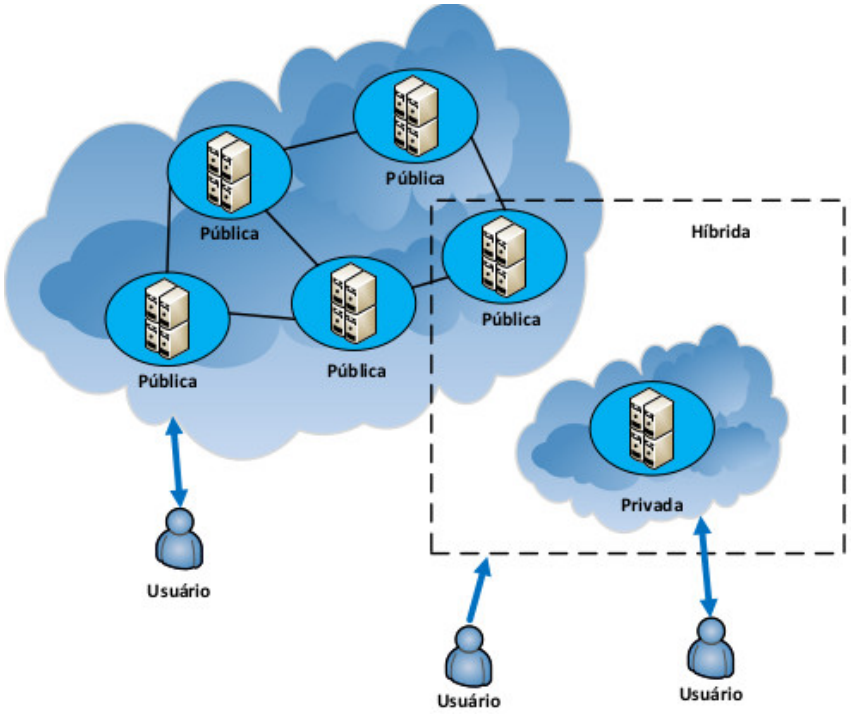
- Pública: São ambientes onde o usuário final não é proprietário e os serviços (hardware e software) são distruibuídos com os diversos "locadores"espalhados pela rede. Os principais fornecedores desse tipo de nuvem são: Google (Google Cloud), Amazon (AWS), Microsoft (Azure) e IBM (IBM Cloud).
- Privada: Esse tipo de nuvem tem como objetivo fornecer um serviço com exclusividade para um determinado usuário ou grupo, os recursos são utilizados apenas por aqueles que contratarem o serviço. Com a maior utilização da nuvem não é mais necessário que as

empresas tenham que possuir uma estrutura própria para ter sua própria nuvem e, isso gerou uma subdivisão nos tipos de nuvem privadas:

- Gerenciada: Os clientes usam uma nuvem criada e configuração por um fornecedor terceirizado.
- Dedicada: Uma nuvem dedicada dentro de outra nuvem, por exemplo: Uma nuvem dedicada para um determinado departamento de uma empresa localizada dentro da nuvem privada da empresa em questão.
- Híbrida: Entre os tipos de nuvem essa é a mais distinta. Uma nuvem híbrida é caracterizada pela junção de uma ou mais nuvens privadas, públicas ou privadas e públicas. Graças à essa configuração é possível mover as aplicações e serviços fornecidos por diversos ambientes de acordo com a vontade do cliente.
- *Multicloud*: Essa abordagem do serviço de nuvem ocorre quando mais de um serviço e de um fornecedor de nuvem, pública ou privada. Apesar de parecer com o conceito de nuvem híbrida, deve-se lembrar que toda núvem híbrida é um multicloud, entretanto nem toda multicloud é uma híbrida. Uma multicloud só será híbrida quando diversas nuvens se conectarem de alguma forma.

Na figura 2.2 é fornecido um exemplo simples de como os tipos principais de nuvem estão organizados.

Figura 2.2 – Arquiteturas de nuvem



Fonte: Adaptado de Furht (2010)

2.2.2 Serviços de Nuvem

Os serviços de nuvem são o meio pelo qual o provedor de serviços fornece os recursos para os clientes de acordo com a necessidade apresentada. Existem três principais serviços de nuvem que são oferecidos amplamente pelos provedores (ERI T.; MAHMOOD, 2013): *Platform-as-a-Service* (PaaS), *Infrastructure-as-a-Service* (IaaS) e *Software-as-a-Service* (SaaS), cada um desses serviços pode ser implantado de forma individual ou em conjunto conforme a vontade do cliente. Existem serviços mais especializados que são disponibilizados por especialmente por nuvems privadas entre eles: *Storage as a service*, *Database-as-a-Service*, *Security-as-a-Service*, *Communication-as-a-Service*, neste trabalho eles não serão abordados entretanto é visível a grande variedade de serviços que a computação nuvem pode prover com a evolução da tecnologia. Abaixo cada serviço será explicado em maiores detalhes, assim como exemplos da sua aplicação serão dados.

- PaaS: O modelo de serviço do PaaS representa um ambiente que está pré-definido para uso, ou seja, já possui os recursos de tecnologia configurados e prontos para uso. A principal vantagem desse tipo de serviço é fornecer um ambiente pronto para o uso, assim os clientes podem se concentrar apenas em utilizar os recursos para suas aplicações e não necessitam se preocupar com configuração de serviços, consequentemente reduz o preço operacional para o cliente. Alguns exemplos de aplicações PaaS são: Elastic Beanstalk, Windows Azure, Heroku, Force.com, como pode se observar todos esses aplicativos citados estão prontos para uso fazendo com que o usuário tenha apenas a preocupação de usá-los da forma que melhor preferir.

- IaaS: Neste serviço de nuvem os recursos podem ser acessados e gerenciados via interfaces e ferramentas de nuvem. O ambiente que pode ser manipulado incluí hardware, rede, conectividade, sistemas operacionais e outros dispositivos "brutos"de tecnologia. O objetivo de dar acesso a esses recursos é fornecer um alto nível de controle para os usuários do serviço, assim pode o uso pode ser feito conforme as preferências do consumidor. Os principais alvos desse tipo de serviço são aqueles que buscam ter o maior controle sobre a utilização dos recursos e sabem lidar com a infraestrutura fornecida. Exemplos desse tipo de serviço são: *Amazon Web Services* (AWS), Microsoft Azure, Digital Ocean, *Google Computing Engine* (GCE), a infraestrutura está localizada nos *datacenters* das empresas forncedoras, mas em alguns casos podem estar em *datacenters* de terceiros.

- SaaS: Esse é modelo de serviço caracteriza-se por fornecer o *software* pronto para uso pelo cliente, ou seja, é o mesmo que aplicações de uso geral como Microsoft Word, Google Chrome e Adobe Reader, mas com a integração do serviço de nuvem para utilização por internet. Entre os três tipos de serviços de nuvem é o que oferece o menor grau de liberdade de configuração para o usuário. O alvo desse serviço no geral são organizações que querem fazer o uso mais rápido e proveitoso da aplicação sem se preocupar o com o

seu funcionamento interno, o que ocasiona na redução de custos e otimização de alocação de recursos por parte da empresa. Os principais exemplos desse tipo de serviço são Google Apps, DropBox, SalesForce, Cisco WebEx, Concur, GoToMeeting.

Na tabela 2.1 um resumo das principais características de cada serviço (PaaS, SaaS, IaaS).

Tabela 2.1: Serviços de Nuvem

Serviço	Nível de Controle da nuvem	Recursos disponíveis
PaaS	Recursos limitados pelos administradores da nuvem	Nível moderado de controle dos recursos de tecnologia de acordo com a plataforma
IaaS	Totalmente controlada pelos administradores	Acesso total a infraestrutura virtualizada e possiblidade de acesso aos recursos fisícos (*hardware*)
SaaS	Permite uso dos recursos e configuração de uso pelo usuário	Acesso para as interfaces de usuário

2.3 Provedores de serviços de nuvem

Nesta seção será feita um descrição dos principais provedores de serviços de nuvem ou gestores de nuvem.

2.3.1 Microsoft Azure Service Plataform

Microsoft Azure Service Plataform(Azure) é uma plataforma que permite o uso de soluções híbridas, que podem envolver tanto a utilização de data centers quanto a nuvem pública, isso facilita aspecto importantes para os clientes como o armazenamento de dados e o backup. Como boa parte dos provedores de serviços de nuvem o Azure fornece seus serviços através da virtualização de hardware assim os clientes só precisam se preocupar com a utilização do serviço contratado deixando a manutenção para os fornecedores. Segundo (ZHANG Q.; CHENG, 2010) o Azure consiste em três componentes e cada um deles provê um conjunto específico de serviços para os usuários da nuvem. Abaixo cada um dos componentes serão descritos brevemente.

- Windows Azure: O primeiro desses componentes Windows Azure fornece ao usuário um ambiente baseado em windows para rodar aplicações e guardar dados nos servidores localizados em data centers de acordo com o serviço adquirido pelo usuário.
- SQL Azure: O segundo componente SQL Azure provê serviços de banco de dados na nuvem baseados no SQL Server.

- .NET Services: O último componente .NET Services oferece serviços de infraestrutura distribuída para nuvens e aplicações locais.

2.3.2 Amazon Web Services

Amazon Web Services(AWS) é a plataforma de nuvem mais utilizada atualmente. Oferece serviços de infraestrutura, armazenamento de dados, entrega de conteúdo e diversas outras funcionalidades de acordo com as escolhas do cliente (AMAZONAWS, 2020). Assim como o Azure também utiliza a virtualização para fornecer os seus serviços pela rede. O AWS é divido em alguns componentes, e cada um oferece um tipo de serviço diferente conforme o requisitado pelo cliente. O primeiro desses componentes *Amazon Elastic Compute Cloud*(Amazon EC2) permite que os usuários da nuvem possam executar e gerenciar servidores localizados nos *data centers* usando APIs ou outras ferramentas e utilidades. Após fazerem suas configurações os usuários podem instalar o software nos servidores e utilizá-los a qualquer momento, o que traz mais comodidade e facilidade. O segundo componente *Amazon Virtual Private Cloud*(VPC) é o ponto de ligação entre a infraestrutura de uma empresa e a nuvem AWS (ZHANG Q.; CHENG, 2010). A Amazon VPC permite que empresas possam conectar sua infraestrutura com um conjunto de recursos isolados da AWS via conexão*Virtual Private Network* (VPN), e assim é possível expandir o gerenciamento da estrutura existente, segurança, *firewall*, e sistemas de detecção de invasores. Por último, para usuários da nuvem existe o *Amazon CloudWatch* uma ferramenta de gerenciamento a qual coleta dados "puros"dos serviços AWS e os processa de forma a torná-los legíveis para os usuários. Esses dados incluem, por exemplo, utilização de CPU, utilização de largura de banda, operações de leitura de disco, etc.

2.3.3 IBM Cloud

A plataforma de nuvem da IBM combina a plataforma como serviço (PaaS) com a infraestrutura como serviço (IaaS) para fornecer uma experiência integrada (IBMCLOUD, 2020). A IBM Cloud possui escalabilidade para suportar pequenas equipes de desenvolvimento e grandes empresas corporativas, além de alta disponibilidade durante todo o tempo. A IBM Cloud é formada por diversos módulos que trabalham juntos para oferecer serviços de nuvem confiáveis e consistentes para os usuários. Abaixo uma breve explicação de cada uma desses módulos:

- Console: Esse componente funciona como *front-end* da plataforma, usado para criar, gerenciar e visualizar os recursos da nuvem.
- Gerenciador de Acesso: Responsável por autenticar os usuários com segurança para ambos serviços de plataforma e controla o acesso aos recursos de forma consistente.
- Catálogo de produtos: Uma lista com todos os produtos suportados pela nuvem.

- Mecanismo de Busca: Uma ferramenta que permite busca e identificar os recursos disponíveis e em utilização pelos usuários e administradores.
- Gerenciador de Conta e Faturamento: Fornece o uso exato para os planos de precificação e proteção segura contra fraude de cartão de crédito.

Assim como os outros gestores de nuvem a IBM Cloud também utiliza virtualização para fornecimento e gerenciamento dos recursos oferecidos pela nuvem.

Na tabela 2.2 as principais caraterísticas e diferenças entre cada provedor de serviços de nuvem apresentado anteriormente.

Tabela 2.2: Comparação entre provedores de serviços de nuvem

Provedores de Nuvem	Microsoft Azure	Amazon EC2	IBM Cloud
Tipo de Serviço	IaaS	PaaS	SaaS
Aplicações alvo	Aplicações gerais e aplicações *Windows*	Aplicações gerais	Aplicações gerais
Computação	*Microsoft Common Language Runtime* (CLR) VM; *Predefined roles of app instances*	Nível de sistema operacional em uma máquina virtual	Controle de acordo com o serviço contratado pelo cliente
Armazenamento	Azure *storage service and* SQL; Serviço de dados	*Elastic Block Service*; Serviço Simples de Armazenamento; Amazon *SimpleDB*	IBM *Object Cloud Storage*; IBM *File Storage* IBM *Block Storage*
Escalabilidade	Dimensionameto automático com base em funções do aplicativo e um arquivo de configuração especificado pelos usuários	Altera automaticamente com base nos parâmetros fornecidos pelo cliente	Dimensionamento automático ou com base nos dados fornecidos pelo cliente

2.4 Gestores de Computação em Nuvem

Os gestores de nuvem são *softwares* que permite o gerenciamento das nuvens públicas, privadas e híbridas e tem o objetivo de prover agilidade e redução de custos. Esses gerenciadores facilitam o acesso ao *hardware* e *software* do provedor de serviços de nuvem, assim é possível

acessar todos os recursos de acordo com as específicações do cliente e o grau de disponibilidade de recursos do provedor.

2.4.1 OpenStack

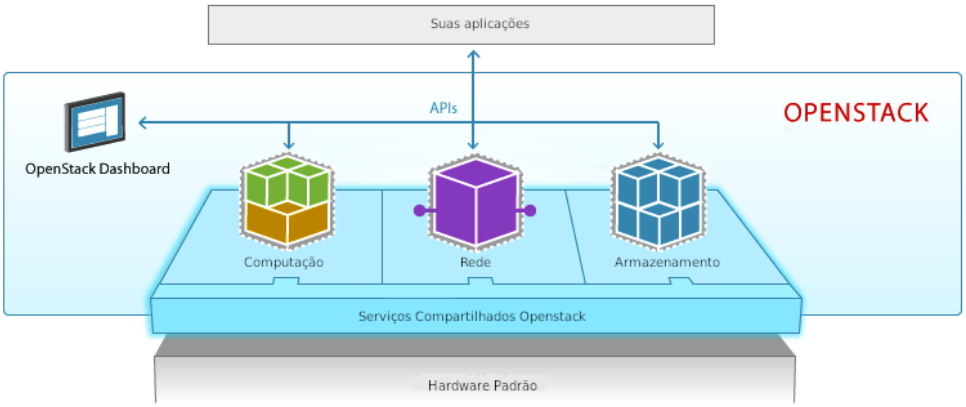
Openstack é um sistema de operação de nuvem que controla os recursos de armazenamento, computação, e rede através de um *datacenter*, todos são gerenciados e fornecidos por APIs com mecanismos de autenticação (OPENSTACK, 2020). Além disso, uma tabela com as informações sobre os recursos utilizados é fornecida aos administradores para que eles possam alocar os recursos via interface *web*.

O *Openstack* é dividido em diversos componentes entre os quais os mais importantes são (CODY, 2016):

- *Compute*: Gerencia os recursos, incluindo *CPU*, capacidade de armazenamento e interfaces de rede.
- *Networking*: Fornece recursos para a rede de máquinas virtuais, incluindo endereçamento IP, roteamento e redes definidas por software (RDS).
- *Object Storage*: Provê armazenamento de objetos, acesso permitido via *RESTful/API*
- *Identity*: Gerencia o controle de acesso para os componentes do *Openstack*, também permite a autorização de serviços.
- *Dashboard*: Fornece interface gráfica via *web* para trabalho com o *Openstack*.

Na figura 2.3 uma descrição de como os módulos do Openstack interagem para oferecer seus serviços.

Figura 2.3 – Arquitetura do Openstack



Fonte: Traduzido e adaptado de Openstack (2020)

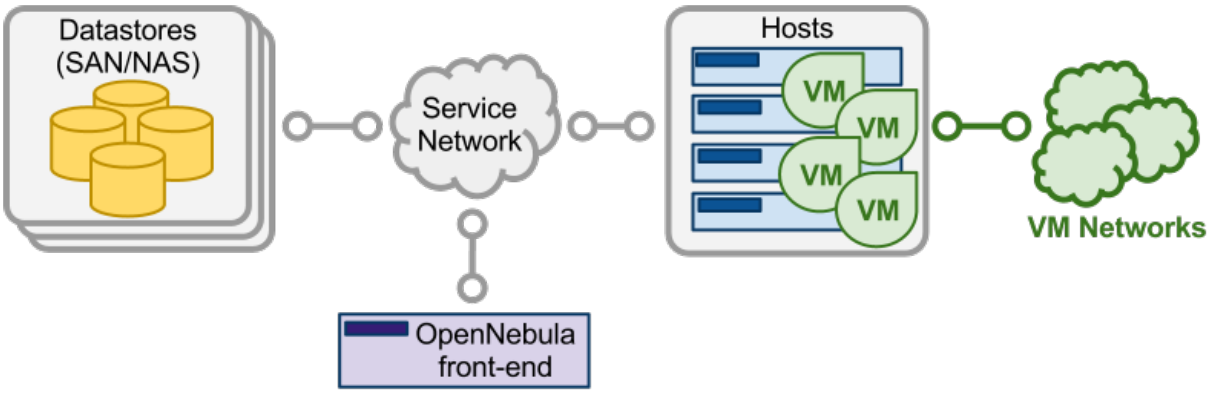
2.4.2 OpenNebula

O OpenNebula é uma plataforma de código aberto para gerenciamento de plataforma para criação de Iaas privadas, públicas e nuvens hibrídas (OPENNEBULA, 2020). Para otimizar o seu funcionamento o OpenNebula é divido em componentes, cada qual tendo uma função no funcionamento da ferramenta. Abaixo um descrição breve de cada componente:

- *Front-end*: É a máquina onde os serviços OpenNebula estão sendo executados.
- *Host*: Máquina fisíca onde um *hypervisor* é executado.
- *Cluster*: Conjunto de computadores que compartilham dados e redes virtuais. *clusters* são usados para fornecer balanceamento de carga, alta disponibilidade e alta performance.
- *Sunstone*: Interface *web* para gerenciamento do OpenNebula.
- *Self-Service*: Interface *web* para disponibilizada para o usuário final.
- *EC2 Service*: Servidor de gerenciamento que permite integração com o serviço EC2.
- *O*penNebula Clouds API(OCA): é um conjunto de bibliotecas que facilitam a comunicação entre o gerenciamento XML-RPC da interface.

Na figura 2.4 um resumo da arquitetura do OpenNebula.

Figura 2.4 – Arquitetura do Opennebula



Fonte: Opennebula (2020)

2.4.3 VMware vRealize Suite

O VMware *vRealize Suite* é uma plataforma de gerenciamento de nuvem construída especialmente para fornecer serviços para centro de dados heterogêneos e nuvens hibrídas (VMWARESUITE, 2020). Sua principal função é gerenciar e oferecer a infraestrutura e os aplicativos para aumentar a velocidade dos negócios do cliente enquanto mantém o controle de

toda a tecnologia por trás dos serviços oferecidos. A arquitetura do *vRealize Suite* é composta pelos seguintes componentes:

- *Data center* definido por *software*: Esse componente fornece diferentes tipos de recursos em conjunto com as funcionalidades mais complexas tendo como base a infraestrutura oferecida pelo serviço. As configurações ficam a disposição do cliente.
- *vRealize Operations Manager*: Oferece o gerenciamento de operações automáticas no sistema englobando os aplicativos até a infraestrutura, o seu principal objetivo é otimizar as implantações dos ambientes conforme as especifícações do cliente.
- *vRealize Log Insight*: Fornece o gerenciamento de *logs* heterogêneos e outras técnicas para monitorar e analisar as entradas e saídas do sistema.
- *vRealize Automation*: É a plataforma de automação de infraestrutura que permite habilitar ambientes *multi-cloud* de autoatendimento, isso agiliza a implementação de ambientes e retira do usuário as dificuldades de configurações mais complexas do sistema.
- *vRealize Business for Cloud*: Esse componente permite a automatização da análise de custo da nuvem contratada, medição de seu consumo de recursos, a comparação e o planejamento da nuvem.

2.5 iSPD

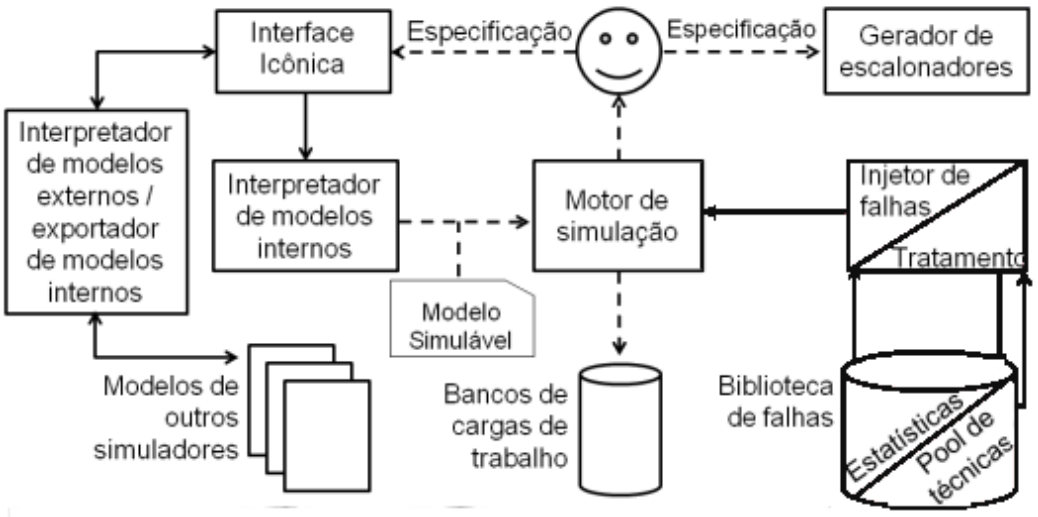
O iSDP (*iconic Simulator for Parallel and Distributed Systems*) é um simulador orientado a eventos discretos voltado para simulação de grades e nuvens computacionais escrito na linguagem de programação orientada a objetos Java. O principal objetivo do iSPD é prover a facilidade de uso, tendo como meio uma interface icônica de modelagem e módulos para gerar políticas de escalonamento e seleção de serviços de *cloud* (MANACERO A.; MENEZES, 2012). Com tais recursos é possível que os usuários mesmo sem conhecimentos de programação possam utilizar.

O iSPD é divido em diversos módulos, cada um tendo uma função específica no funcionamento da ferramenta (SILVA DIOGO.; MANACERO, 2015) e os novos módulos implementados e testados por (TAMASHIRO C. B. O.; LOBATO, 2020). Um diagrama de relacionamento entre os módulos constituintes do iSPD é mostrado na Figura 2.5 e abaixo cada um desses módulso será descrito de maneira breve:

- Interface Icônica: Módulo responsável pela modelagem através de ícones que representam os elementos que fazem parte da simulação conforme o serviço escolhido. Outra função desse módulo é a configuração da carga de trabalho.

- Interpretador de Modelos Internos: Esse módulo tem como função interpretar o modelo icônico, fazendo sua conversão para o modelo simulável que é utilizado pelo motor de simulação.
- Interpretador de Modelos Externos e Exportador de Modelos Internos: Sua função é interpretar modelos gerador a partir de outros simuladores, atualmente exporta modelos do Simgrid e GridSim para modelos icônicos do iSPD e exporta modelos icônicos do iSPD para modelos correspondentes desses simuladores.
- Motor de Simulação: Responsável por realizar a simulação de eventos discretos a partir da leitura de um modelo simulável. Outra função deste módulo é apresentar os resultados e métricas da simulação realizada.
- Gerador de Escalonadores: Tem como função gerenciar os escalonadores disponivéis para uso na simulação e gerar novas política de escalonamento de maneira facilitada, através de formulações matemáticas simples.
- Injetor de Falhas: Responsável por injetar as falhas na simulação que podem ou não ser escolhidas pelo usuário dependendo da situação, as falhas são armazenadas em uma biblioteca própria.
- Tratamento de Falhas: Responsável por armazenar as técnicas para o tratamento das falhas inseridas no sistema, atua em conjunto com o injetor sempre que uma falha é adicionada na simulação.

Figura 2.5 – Diagrama de interação dos módulos do iSPD



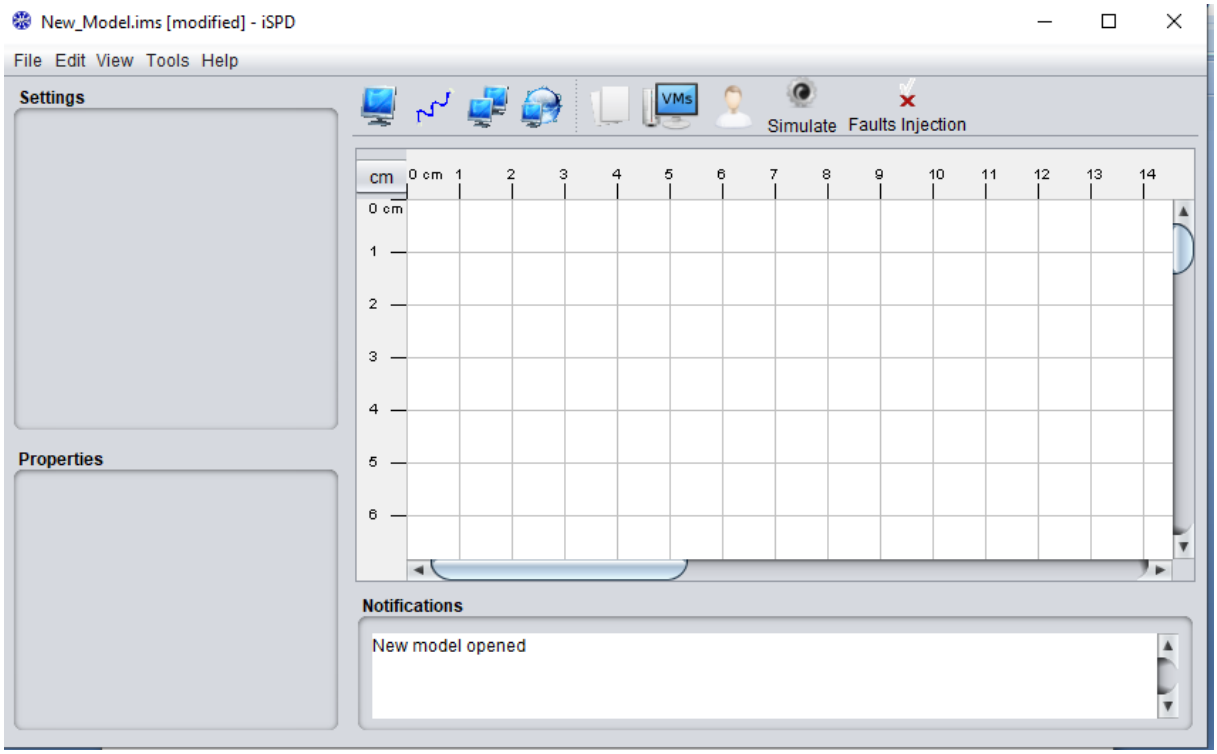
Fonte: Tamashiro (2020)

2.5.1 Estado atual

O iSPD é mantido pelo GSPD (Grupo de Sistemas Paralelos e Distribuídos) e seu *download* pode ser realizado na página *web* do grupo (GSPD, 2020). Pode ser usado para realização de testes por parte de profissionais ou por estudantes como um auxiliar em seus estudos.

A versão atual permite a escolha de três tipos de serviços para modelagem: *grid*, IaaS, PaaS. Após a escolha de uma das opções anteriores o usuário pode escolher através da interface icônica quais componentes farão parte da simulação sendo as opções: computadores, clusters, conexões, usuários, máquinas virtuais, rede. Na Figura 2.6 abaixo temos um exemplo da interface principal de simulação do iSPD.

Figura 2.6 – Interface principal iSPD



Fonte: https://www.dcce.ibilce.unesp.br/spd/downport.php

Após a escolha dos componentes que participarão da simulação é permitido ao usuário escolher as especificações de cada componente como, por exemplo: capacidade de disco, número de núcleos de cada processador, escalonadores, presença de máquinas virtuais, *hosts* mestres e escravos, carga de trabalho entre outras opções. Esse volume de escolhas permite que o usuário possa realizar a simulação com um grande detalhamento permitindo uma coleta mais precisa de dados conforme as diretrizes fornecidas. Além disso, o iSPD também permite exportar modelos Simgrid e GridSim, assim como escalonadores fornecidos pelo usuário.

Após a realização da simulação os seus resultados são mostrados na forma de um arquivo com extensão xml gerado pelo simulador e fica a disposição do usuário.

2.6 Falhas em Nuvem

Uma das principais preocupações no momento de projetar, implementar e manter um sistema de computação em nuvem e outros sistemas computacionais involve como entender, detectar e tratar as falhas que possam ocorrer e afetam o funcionamento de nossas aplicações. Nos grandes sistemas de nuvem a grande dificuldade para se obter informações sobre as falhas é a sensibilidade dos dados e o curto espaço de tempo em que são coletadas as infomarções sobre falhas, assim gerando uma baixa quantidade de dados para uma análise mais profunda e precisa de suas causas (SCHROEDER B.; GIBSON, 2010). O principal objetivo de se estudar falhas em um ambiente de nuvem é evitar que a indisponibilidade do sistema sabendo que uma das principais características dessa tecnologia é fornecer serviços a qualquer momento conforme as necessidades do cliente (MANI D.; MAHENDRAN, 2016).

Antes de um aprofundamento nos tipos de falhas possíveis em um ambiente de nuvem é necessário o esclarecimento de alguns termos que muitas vezes são intercambiáveis em outros contextos, contudo possuem significados diferentes dentro da computação. Os termos em questão são: falha, erro e defeito.

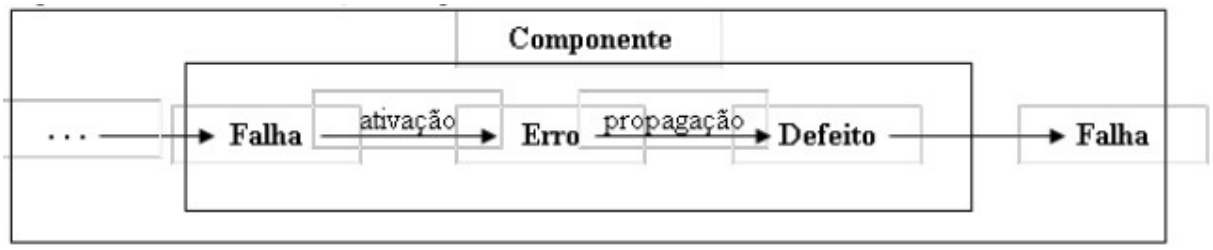
- Falha: É a incapacidade do sistema de executar uma tarefa devido a um estado anormal ou um *bug* presente em uma ou múltiplas partes do sistema. Falhas podem ser ou não ser a causa de erros.
- Erro: É a diferença entre a saída esperada e a saída obtida. Um erro pode fazer com que o sistema funcione de maneira errada podendo gerar um defeito.
- Defeito: A falta de capacidade do sistema ou de um componente de executar uma função requerida de acordo com suas especificações. Na maioria dos casos podem ser observados pelo usuário ou por outro sistema de computação. Existem diversos tipos de situações que podem dar errado dentro de um sistema, entretanto se um resultado errado não for fornecido isso não poderá ser classificado como defeito.

2.6.1 Modelos de falha

As falhas em sistemas costumam ocorrer de acordo com seu uso e organização do mesmo. As falhas levam o sistema para seu funcionamento incorreto ou até mesmo o desligamento, entretanto em sistemas distribuídos quando uma falha ocorre ela pode ser classificada como falha parcial, pois não afeta todo o sistema permitindo que o mesmo continue funcionando com capacidade reduzida de acordo com o problema ocorrido. Seja qual for a natureza de uma falha,

ela pode gerar um erro ou não. Uma falha que gera um erro é conhecida como falha ativa, caso contrário, é considerada uma falha latente. Para um erro tornar-se ativo, é preciso que ocorra um estado de defeito, caso não ocorra o erro é latente (DANTAS, 2005). Na figura 2.7 é mostrado uma generalização da ocorrência de uma falha até a geração de um defeito.

Figura 2.7 – Propagação de uma Falha



Fonte: Adaptado de Dantas (2005)

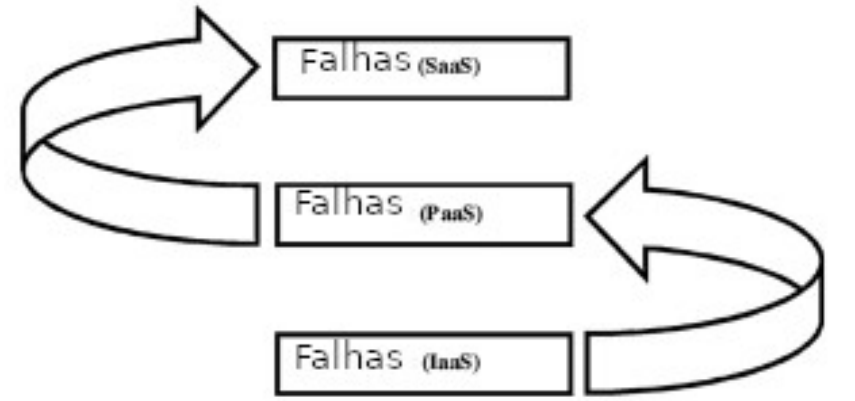
Na computação em nuvem os tipos de falhas assim como em outros sistemas dependem do seu local de ocorrência (KUMARI P.; KAUR, 2018) e dentre essas possíveis falhas existem aquelas que são mais recorrentes em todos os sistemas de nuvem. Abaixo uma breve descrição de cada uma dessas falhas:

- Falha de rede: Como o sistema de nuvem é acessado pela rede (Internet), a maior ocorrência de falhas são de rede. Essas falhas podem ocorrer de acordo com as partes da rede e aqui são algumas delas: perda de pacotes ou corrompimento, congestionamento, falha de entrega, tempo de vida do pacote, etc.
- Falhas físicas: Esse tipo de falha ocorre nos recursos de hardware, como processadores, memórias, dispositivos de armazenamento, queda de energia, entre outros dispositivos físicos.
- Falhas de processos: Ocorrem devido a falha de armazenamento, defeitos no *software*, falta de capacidade de processamento e outras causas.
- Expiração de serviço: Acontece quando um serviço expira enquanto uma aplicação faz uso do mesmo, isso leva a falha de serviço. Acontece na maioria das vezes por falta de desempenho.
- Falha de mídia: Ocorre quando um tipo de media (áudio, video, etc) falha ocasionando a queda do serviço.
- Falha de máquina virtual: Esse tipo de falha é predominante na nuvem, ocorre na maioria das vezes devido a falhas lógicas e erros do servidor quando este está alocado em um ambiente de nuvem (NIVITHA K.; PABITHA, 2020).

- Falha por omissão: acontece quando um servidor deixa de responder a uma requisição, ou seja, o servidor não tem conhecimento de que um pedido foi feito e não pode atendê-lo. Podem ser ocasionadas por erros de *software* ou gerenciamento inadequado do sistema.
- Falhas arbitrárias: podem ser chamadas de falhas bizantinas. Ocorrem devido a algum erro do sistema e ocasionam diversos outros erros podendo levar até mesmo causar a interrupção do sistema, ou seja, um defeito.

Na Figura 2.8 tem-se uma exemplificação do fluxo de uma falha dentro da arquitetura de nuvem.

Figura 2.8 – Fluxo de Falhas



Fonte: Traduzido e adaptado de Hasan (2018)

Uma falha pode ser categorizada com base nos recursos de processamento e tempo (SAIKIA P. L.; DEVI, 2018). As falhas podem ser classificadas das seguintes maneiras:

- Permanente: Esse tipo de falha ocorre quando algum tipo de acidente afeta algum equipamento físico, levando a grandes disturbios no sistema ou mesmo a interrupção do funcionamento em alguma parte durante um grande período de tempo.
- Intermitente: Ocorre de forma ocasional. A maioria dessas falhas não são percebidas quando o sistema é testado, e só aparecem durante o período de funcionamento.
- Transitória: É ocasionada por alguma falha inerente ao sistema. A maioria dessas falhas é corrigida retornando o sistema a um estado anterior, reiniciando ou reenviando uma mensagem. São tipos de falhas comuns em sistemas de computadores.

2.7 Considerações finais

Esse capítulo apresentou os principais conceitos envolvendo a computação em nuvem, destacando as principais características, serviços oferecidos, provedores de serviços e gestores de nuvem, além de apresentar o iSPD e o seu estado atual de desenvolvimento. Os principais conceitos sobre falhas e seus tipos foram apresentados para dar o embasamento necessário para realização do projeto.

3 Desenvolvimento

Neste capítulo serão apresentados com mais detalhes o funcionamento do iSPD, o estado atual do injetor de falhas, a biblioteca de falhas, todos componentes essenciais para o desenvolvimento do trabalho proposto.

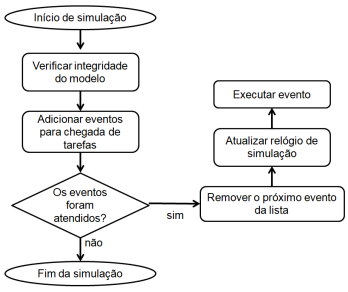
3.1 Apresentação e organização geral do iSPD

Para compreender melhor o trabalho de adição de novas as falhas ao simulador é, necessário aprofundar-se mais nas estruturas do programa e estudar os processos que o fazem funcionar.

O iSPD é organizado em classes separadas cada qual tendo sua função no funionamento do simulador, como por exemplo os algoritmos de escalonamento estão localizados em uma classe ou a interface icônica que tem sua classe própria entre outras funções do programa, entretanto não serão abordadas todas as classes que compom a ferramenta apenas as necessárias realização do trabalho.

Na seção 2.5.1 a interação com o usuário foi descrita de forma breve, entretanto sem mostrar como a simulação funciona internamente no simulador. A figura 3.1 apresenta o fluxo da simulação desde o começo até seu término. Primeiro temos o processo de validação de simulação onde os parâmetros fornecidos pelo usuário são analizados e verificados, depois a ocorrência da simulação dentro do motor e a adição de tarefas, após todo o processo a simulação é finalizada. Para o usuário será fornecido um relatório com resultados gerados e são exibidos o tempo de simulação, a distribuição de tarefas nas máquinas virtuais alocadas e a eficácia do algoritmo de escalonamento selecionado, assim o usuário pode verificar de maneira mais precisa como transcorreu a simulação. Os resultados gerados no relatório ao final do processo são referentes às métricas de desempenho, as quais são fornecidas pelo iSPD, sendo elas: tempo médio de espera e resposta, eficiência do sistema e satisfação do cliente (MANACERO A.; MENEZES, 2012).

Figura 3.1 – Fluxograma de Simulação do iSPD



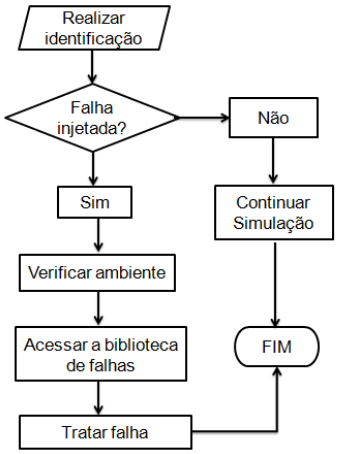
Fonte: Tamashiro (2020)

3.2 Injetor de Falhas iSPD

Com a aprimoração da interface icônica foi possível a inserção de falhas a partir da interação com a interface através da classe JSelecionarFalhas.java (TAMASHIRO, 2020). Durante o processo de simulação, as falhas são injetadas pelo sistema injetor e são detectadas pela biblioteca que as contém. Quando ocorre uma detecção de inserção, é realizado o processo de tratamento de falhas proativas ou reativas, a técnica usada vária de acordo com o ambiente em que a falha ocorreu.

Na figura 3.2 é fornecido um fluxograma do processo de identificação da ocorrência de falha.

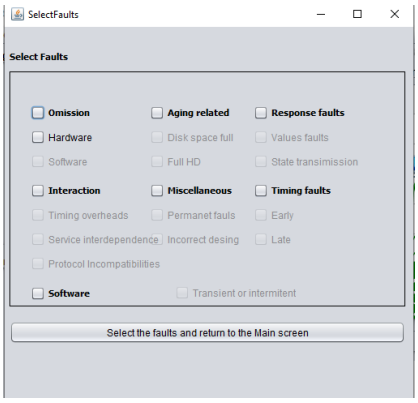
Figura 3.2 – Fluxograma da identificação e tratamento de falha



Fonte: Tamashiro (2020)

Na figura 3.3 pode-se ver uma tela de seleção de falhas que serão selecionadas pelo usuário na simulação. Nesta interface as opções de falhas são: omissão, evelhecimento, resposta, softwares, temporização, interação e diversas.

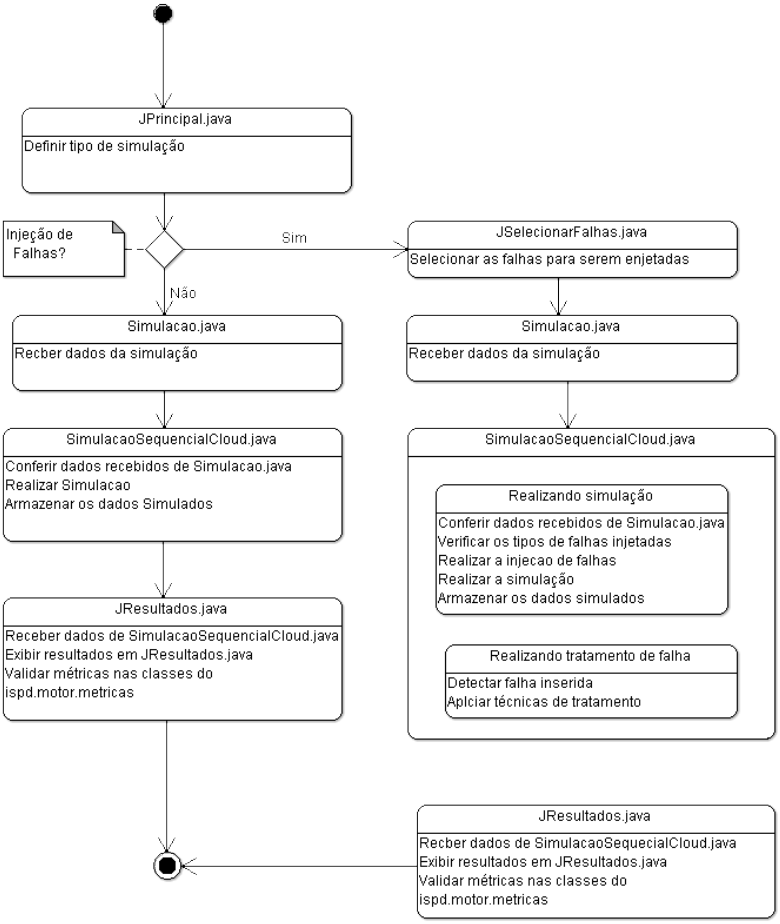
Figura 3.3 – Interface de inserção de falhas iSPD



Fonte: https://www.dcce.ibilce.unesp.br/spd/downport.php

O processo de injeção de falhas ocorre com a inserção de eventos no sistema de forma que um estímulo é gerado para que a falha possa ser detectada e assim o processo de tratamento possa ser iniciado e a simulação finalizada ao fim de toda execução (TAMASHIRO, 2020). Na figura 3.4 uma apresentação de um esquema detalhando como o sistema se comporta quando uma falha é colocada no sistema.

Figura 3.4 – Diagrama da Máquina de Estados da Modelagem das Falhas

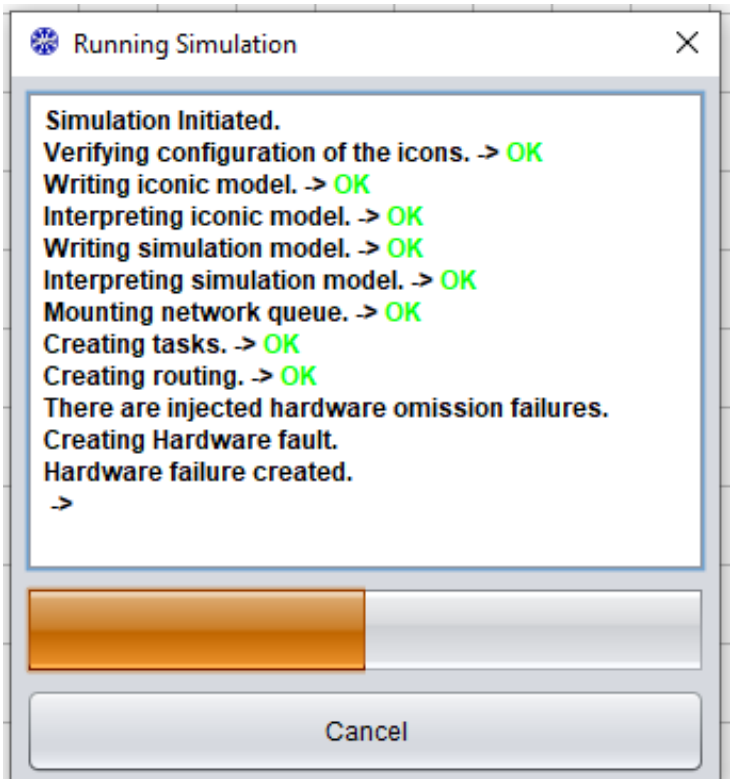


Fonte: Tamashiro (2020)

No atual sistema o gerador de falha inclui uma notificação durante o processo de simulação para o usuário saber se a falha foi injetada e o tipo da falha. Quando o simulação é finalizada uma mensagem indicando que a falha foi tratada é exibida. Na figura 3.5 é monstrado uma notificação de inserção da falha. Existem falhas que estão implementadas no sistema sendo, omissão por *hardware* e omissão por *software*, encontram-se na classe FaultLib.java que representa a bilbioteca de falhas dentro do sistema. As técnicas de tratamento estão localizadas em classes especifícas para cada falha sendo as de omissão *hardware* na classe FIHardware.java e a de omissão de *software* na classe FHSoftware.java. Cada falha tem mais de uma técnica de

tratamento caso a primeira selecionada não seja eficaz, ou seja, os processos são aplicados até a melhor solução disponível no simulador ser aplicada a cada situação definida pelo usuário.

Figura 3.5 – Inserção de falha durante da simulação



Fonte: Autor (2020)

3.3 Injeção e tratamento de falhas no iSPD

Nesta seção o planejamento da implementação da proposta do trabalho será apresentado utilizando as informações apresentadas nos capítulos anteriores.

Para a realização do trabalho foi escolhida uma falha que se divide em dois tipos para ser adicionada ao iSPD, sendo a falha de resposta. Essa opção foi adotada devido sua maior ocorrência e por estar ligada a comunicação em sistemas de computação em nuvem (ARGAWAL H.; SHARMA, 2015). Junto com as falhas de hardware elas representam aproximadamente metade das ocorrências (TAMASHIRO, 2020) de todas as falhas detectadas em ambientes de nuvem, portanto o estudo de seus efeitos tem maior importância para situações que possam ocorrer em sistemas reais. Abaixo uma descrição mais detalhada da falha escolhida.

- Falha de resposta: Essa falha ocorre quando um resultado esperado não é fornecido pelo sistema devido a algum comportamento anormal ou um resultado fica sem retorno. Esse tipo de falha por se dividir em falhas de valor e falhas de estado de transmissão.

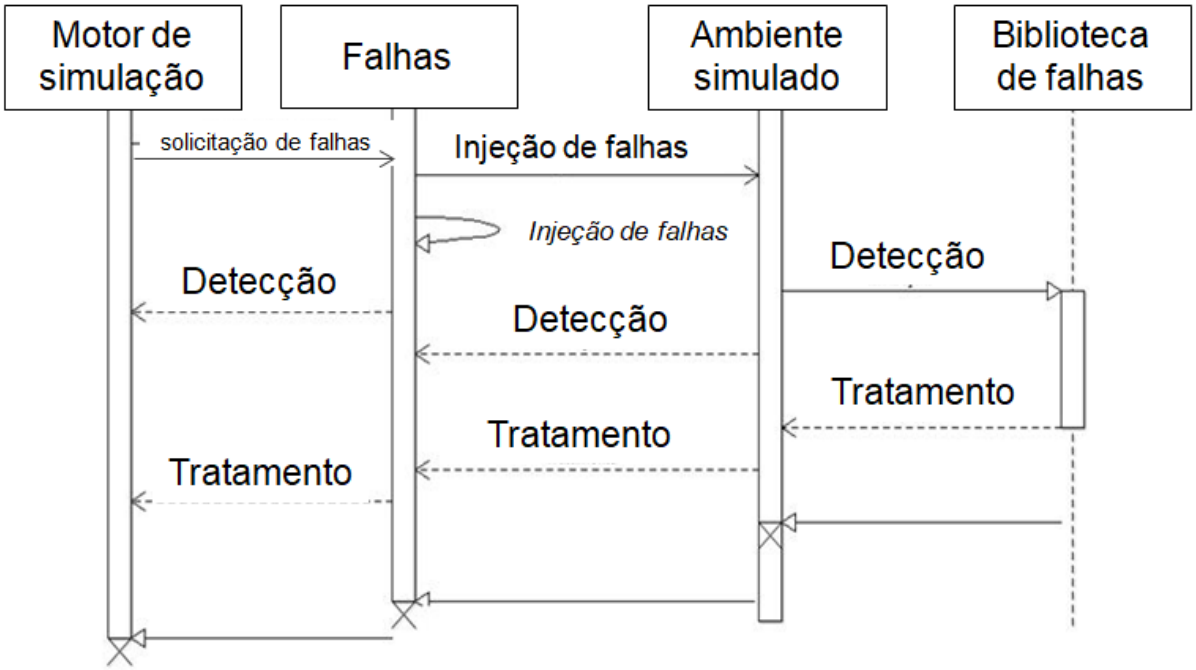
- Falha de estado de transmissão: A falha de estado de transmissão ocorre quando o sistema não está em um estado que permite a transmissão de uma mensagem impedindo a ocorrência de comunicação.

No trabalho as falhas que serão impletadas no iSPD serão: de resposta de estado e de valor. Cada uma dessas falhas possuirá técnicas de tratamento para recuperação do sistema. Abaixo uma apresentação da técnica de tratamento e sua correspondência com a falha na qual será aplicada, as correspondências foram feitas de acordo com Tamashiro (TAMASHIRO, 2020). Cada falha é representada por um tipo de classe, as quais são injetadas por chamadas de métodos, as falhas selecionadas estão localizadas na biblioteca de falhas FaultLib.java, apos a inserção são acionados as técnicas de tratamento conforme acordo com a falha escolhida (TAMASHIRO C. B. O.; LOBATO, 2020). Na figura 3.6 é apresentado um fluxograma exemplificando todo o processo desde a identifcação da falha até o seu tratamento. Os seguintes métodos de tratamento de falhas serão implementados no iSPD tendo em vista que são os mais adequados para as situações propostas anteriormente:

- Checkpoint: Usada para salvar o estado do sistema periodicamente, em intervalos de tempo fornecidos pelos administradores. No caso de uma falha em uma determinada tarefa, o processo é reiniciado com base no último estado salvo ao invés de iniciar do começo da simulação. Isso evita perda de dados importantes que foram gerados e não armazenados durante a execução do programa. Essa técnica será aplicada na falha de resposta por valor.
- Replicação: Usada para criar múltiplas cópias de tarefas e armazenar em diferentes localizações. Uma tarefa pode continuar sua execução na presenção de um malfuncionamento ou falhas até que as réplicas sejam destruídas. Essa técnica será aplicada na falha de estado de transmissão.

Para introduzir essas falhas alterações deverão ocorrer nas classes JSelecionarFalhas.java para permitir a seleção das novas opções. A classe JProgressoSimulacao.java recebe o parâmetro passado pela inserção e notifica o usuário sobre a injeção uma ou mais falhas. Nesse processo as classes JSimulacao.java e JSimulacoSequencialCloud.java sofrerão modificações tendo como objetivo suportar as mudanças que irão ser adicionadas para a adição das novas falhas. Além disso, classes específicas cada técnica de tolerância serão criadas para que o processo de tratamento possa ser efetivado por diversas falhas que possam fazer uso do mesmo e não só as que serão implementadas no trabalho proposto.

Figura 3.6 – Ciclo de deteccção e tratamento de falha no iSPD



Fonte: Tamashiro (2020)

3.4 Considerações Finais

Ao longo desse capítulo foram apresentadas as principais modificações que serão feitas no iSPD, falhas que irão fazer parte da bilbioteca e suas respectivas técnicas de tratamento. As descrições das modifições propostas procuraram descrever o funcionamento da ferramenta e como as novas funcionalidades serão introduzidas.

4 Testes e Resultados

A validação das novas falhas inseridas através do sistema injetor de falhas foi feita com base em alguns testes comparativos. Os testes envolveram 4 grupos que incluiram máquinas físicas e virtuais criadas durante as simulações realizadas no iSPD conforme especificado anteriormente no seu funcionamento. As seções a seguir irão apresentar descrições desses testes, tendo como ponto de início a injeção de falha por resposta e o seu método de tratamento *checkpoint*.

4.1 Grupos e testes

Para realizar os experimentos referentes a validação das inserções de falhas e métodos de tratamento foi necessária a efetuação de simulações para validar as implementações propostas no 3. Quatro grupos foram selecionados para a realização dos testes tendo cada 32 máquinas físicas e cada um 8, 16, 32 e 64 máquinas virtuais (VMs), respectivamente. Na tabela 4.1 temos a identificação cada um destes grupos de acordo com os números de máquinas físicas e virtuais.

Tabela 4.1: Grupos do experimento de validação

Grupos	M. Virtuais	M. Físicas
G1	8	32
G2	16	32
G3	32	32
G4	64	32

A escolha desses grupos foi feita com base em (TAMASHIRO, 2020) e na dissertação de (RODRIGUES, 2019) no qual são usadas máquinas virtuais de 32 processadores, máquinas físicas de 32 processadores, 32 GB de memória e com uso do sistema operacional Linux. Para cada grupo foram realizados 4 experimentos, sendo estes referentes as falhas de resposta, estado de transmissão, reposta e estado de transmissão, e todas as falhas implementadas no simulador até o momento deste trabalho. Comparações entre os tempos de simulação foram feitas para verificar as diferenças de desempenho em cada grupo com o objetivo de analisar o comportamento em cada caso. Os testes foram realizados em uma máquina local com sistema operacional Linux e a distribuição Ubuntu 20.04. Todos os testes obtiveram 100% de satisfação conforme as métricas fornecidas pelo iSPD ao final da simulação 2.5.1 mesmo com a injeção de falhas, quanto a eficiência todos os resultados foram ruins devido ao processo de tratamento de falhas que eleva o custo computacional de execução da simulação (TORQUATO M.; MACIEL, 2017). O primeiro experimento apresentado na tabela 4.2 é relacionado a falha de resposta e a falha de estado de transmissão tendo como alvo o G1.

Tabela 4.2: Tempo de simulação das falhas grupo G1

Falhas	Tempo em segundos (s)	Satisfação	Eficiência
Falha de Resposta	23.5510	100%	Ruim
Falha de Estado de Transmissão	29.4374	100%	Ruim
Falha de Resposta e Estado de Transmissão	34.3377	100%	Ruim
Falha de Resposta, Estado de Transmissão, Omissão de Hardware e Software	40.0458	100%	Ruim

Na tabela 4.2 podemos ver os resultados para o grupo 1 que apresenta 32 máquinas físicas e 8 máquinas virtuais. A falha de resposta foi a que apresentou o menor tempo 23.5510s de durante a simulação enquanto que a falha de estado de transmissão apresentou o maior tempo de maneira individual 29.4374s, a falha de resposta teve um teve cerca de 8% menor em relação a de estado de transmissão. Quando ambas as falhas são injetadas na mesma simulação o tempo de simulação aumenta 34.3377s conforme o esperado. Por fim, quando todas as falhas são injetadas na simulação o tempo de execução também aumenta para 40.0458s, ou seja, está dentro do que é esperado quando a simulação tem que injetar a falhas tratar todas de uma única vez. O tempo médio de simulação foi de 25.47438s.

Tabela 4.3: Tempo de simulação das falhas grupo G2

Falhas	Tempo em segundos (s)	Satisfação	Eficiência
Falha de Resposta	19.7809	100%	Ruim
Falha de Estado de Transmissão	26.2555	100%	Ruim
Falha de Resposta e Estado de Transmissão	32.0647	100%	Ruim
Falha de Resposta, Estado de Transmissão, Omissão de Hardware e Software	60.9160	100%	Ruim

No G2 temos 32 máquinas físicas e 16 máquinas virtuais. Em relação a falha de resposta o tempo foi de 19.7809s sendo cerca de 17% mais rápido em comparação com o experimento do G1, algo muito satisfatório e antecipado tendo em vista que quanto maior o número de

máquinas virtuais o tempo de simulação é reduzido(TORQUATO M.; MACIEL, 2017). A falha de estado de transmissão teve uma melhora de rapidez de aproximadamente 11% em comparação com o G1, outro resultado esperado conforme dito acima em relação ao número de máquinas virtuais. Um ponto de distorção foi o teste com todas as falhas presentes que aumentou significativamente o tempo de simulação sendo este de 60.9160s, dentro dos testes realizados foi a maior incongruência encontrada. O tempo de simulação médio foi de 27.80342s, isto é 8.5% maior que o tempo em relação ao tempo de G1 e conforme dito anteiormente o resultado pode ser atribuído a distorção encontrada no teste onde todas as falhas foram injetadas na simulação.

Tabela 4.4: Tempo de simulação das falhas grupo G3

Falhas	Tempo em segundos (s)	Satisfação	Eficiência
Falha de Resposta	18.8935	100%	Ruim
Falha de Estado de Transmissão	29.2609	100%	Ruim
Falha de Resposta e Estado de Transmissão	42.2591	100%	Ruim
Falha de Resposta, Estado de Transmissão, Omissão de Hardware e Software	48.7797	100%	Ruim

No G3 temos 32 máquinas físicas e 32 virtuais. O tempo de simulação da falha de resposta foi reduzido sendo agora de 18.8935s um ganho aproximadamente 4.5% de velocidade em relação do experimento anterior. A falha de estado de transmissão teve um pequeno aumento de pouco mais de 11% no tempo de simulação em comparação com G2, assim temos outra distorção nos resultados, entretanto com um baixo impacto na totalidade dos experimentos realizados apesar de ser uma porcentagem significativa para esse teste. O tempo de simulação entre as duas falhas implementadas apresentou a última distorção entre os testes realizados, tendo durado 42.2591s um crescimento de 25% em comparação com G2, um aumento significativo. Este experimento foi o que mais diferenciou-se dos outros e a provavel causa é o tratamento da falha de transmissão, replicação, como envolve a criação de cópias dentro do sistema em alguns casos o tempo demandado para executar o processo pode ser mais elevado que o normal(TORQUATO M.; MACIEL, 2017). O tempo médio da simulação para esse grupo foi de 27.83864s, isto é um aumento de menos de 1% em relação ao experimento de G2.

O G4 apresenta 32 máquinas físicas e 64 máquinas virtuais. A falha de resposta teve o menor tempo dentre os experimentos realizados 17.9206s uma diminuíção de aproximadamente 6% comparado ao G3. A falha de estado de transmissão também teve o o seu menor valor 25.0675s sendo mais rápida cerca de 15% comparativamente ao experimento anterior, um

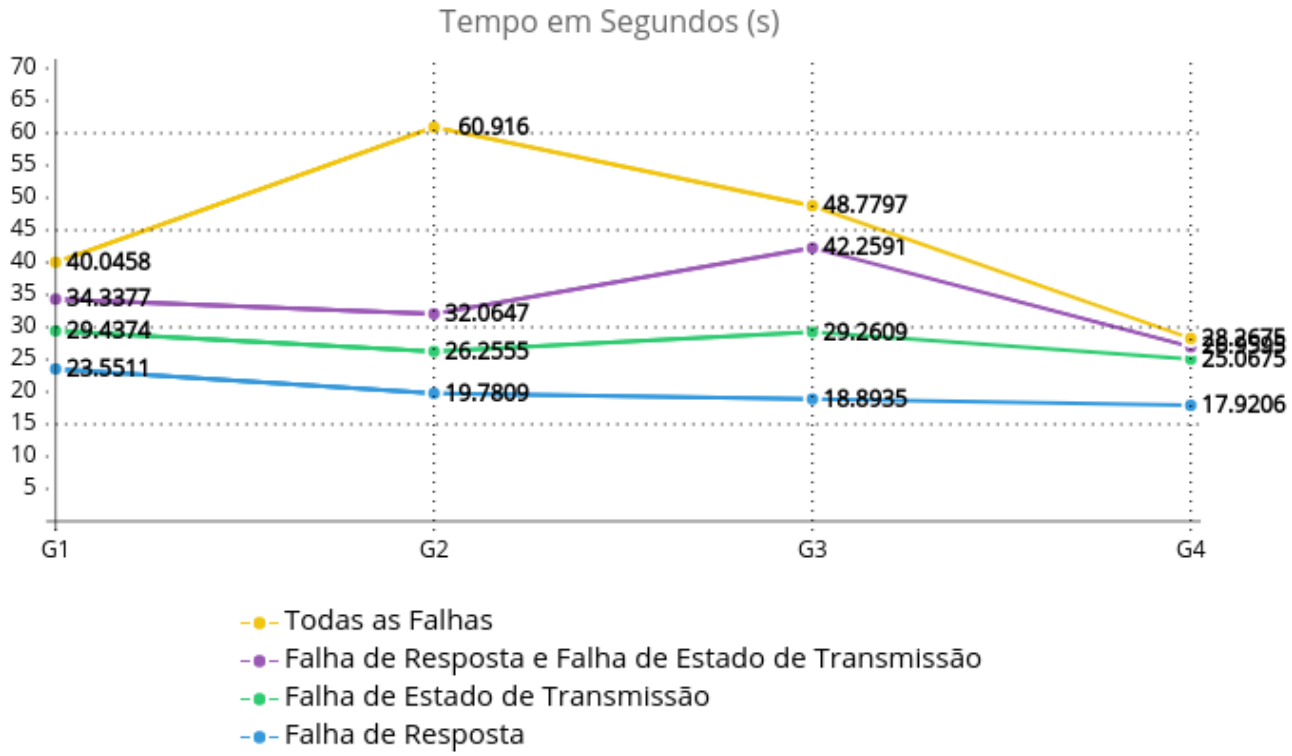
Tabela 4.5: Tempo de simulação das falhas grupo G4

Falhas	Tempo em segundos (s)	Satisfação	Eficiência
Falha de Resposta	17.9206	100%	Ruim
Falha de Estado de Transmissão	25.0675	100%	Ruim
Falha de Resposta e Estado de Transmissão	26.9595	100%	Ruim
Falha de Resposta, Estado de Transmissão, Omissão de Hardware e Software	28.2675	100%	Ruim

considerável ganho de desempenho considerando o método de tratamento utilizado por esta falha (replicação). Quando ambas as falhas foram injetadas o ganho em desempenho foi de mais de 17%, algo esperado se considerando que o número de máquinas virtuais foi dobrado. Por fim, o tempo de execução quando todas as falhas são inseridas foi de 28.2675s, ou seja, 17% mais rápido que o teste em G3, um ganho muito significativo em desempenho. O tempo médio de simulação neste grupo foi de 19.58431s, sendo 30% mais veloz que G4, algo muito significativo considerando que nos outros testes a média de crescimento não passou de 17% entre médias de tempo e nas comparações individuais entre as falhas. Esse resultado pode ser atribuído ao aumento de máquinas virtuais que passaram de 32 em G2 para 64 em G4, ou seja, um aumento considerável no poder de processamento acarretando nesse ganho notável no último experimento. Em todos os experimentos realizados em todos os grupos obtivemos uma satisfação de 100% e eficiência ruim, a eficiência deve-se aos métodos de tratamento serem custosos e a satisfação é baseada nas métricas de usuário como já especificadas em capítulos anteriores 2.5.1.

As comparações feitas ao longo dos experimentos são resumidas na figura 4.1, e como já citado anteiormente podemos ver claramente que no geral o aumento de máquinas virtuais ocasionou na queda no tempo de simulação e, caso mais teste fossem efetuados com uma maior quantidade de máquinas virtuais ocasionando no aumento na capacidade de processo total teríamos uma chance muito grande de observar descréscimo ainda maior no tempo de execução na maioria das situações, mesmo levando em conta as possíveis distorções que podem ocorrer conforme analisamos anteriormente nos experimentos realizados e levando-se em consideração que em qualquer ambiente diferenças dos resultados esperados podem ocorrer de acordo com as circunstâncias. O gráfico resume os experimentos realizados neste capítulo de forma que facilite à visualização dos resultados.

Figura 4.1 – Desempenho dos grupos



Fonte: Autor (2021)

No gráfico podemos constatar que no último grupo (G4) todos os valores tornaram-se mais próximos, mas todos diminuíram significativamente conforme observado anteriormente nos experimentos. Assim, podemos verificar mais uma vez que quanto maior a quantidade de máquinas virtuais, maior é o poder de processamento e o tempo é reduzido.

4.2 Considerações finais

Ao longo do capítulo diversas comparações foram feitas para a validação das falhas e métodos de tramento implementados no iSPD sendo as falhas de resposta e estado de transmissão com os métodos de tramento *checkpoint* e replicação, respectivamente. Pode-se verificar que conforme aumentamos o número de máquinas virtuais nos experimentos o tempo total é reduzido, mas devemos observamos que em como qualquer simulação algum compartamento anômalo pode ocorrer como visto em alguns experimentos. Assim podemos atestar a validade das falhas e métodos de recuperação inseridos no simulador.

5 Conclusões e Trabalhos Futuros

5.1 Conclusões

Conforme foi apresentado ao longo do trabalho o objetivo foi o estudo, implementação de falhas e métodos de tratamento em um simulador de nuvem, no caso o iSPD pertencente ao Grupo de Sistemas Parallelos e Distribuídos. Além do estudo dos assuntos relacionados ao assunto computação em nuvem foi necessário também o estudo do próprio simulador sendo este um sistema legado, o entendimento de ambas as partes foi essencial para a conclusão de forma satisfatória do que foi proposto no início do trabalho.

Com as novas falhas implementadas e os métodos de tratamento, agora é possível realizar simulações mais próximas de ambientes reais onde problemas podem ocorrer e acontecem a todo momento, sendo muitas vezes difícil prever o resultado. Com essas novas funcionalidades os usuários poderão antecipar-se de maneira mais eficiente e adequada com base nos dados fornecidos pelas simulações que realizarem. Além disso, os usuários poderão comparar os resultados obtidos com outras ferramentas como Openstack e OpenNebula para analisarem diferenças e desempenho algo necessário para se obter mais dados úteis e ajudar na tomada de decisões.

O maior desafio encontrado na realização do trabalho foi compreender internamente o funcionamento do iSPD, como é um sistema muito grande e complexo foi necessário muito tempo de dedicação até que o nível de entendimento do programa fosse o suficiente para a realização de alterações sem prejudicar a harmonia da sua execução ou mesmo a alteração de funcionalidades já existentes podendo causar algum tipo de interferência e prejudicar o sistema. E ainda levando em conta que estamos trabalhando com falhas e o sistema legado torna-se muito fácil tomar alguma decisão errada na hora de implementar novas funcionalidades dentro da ferramenta.

O principal objetivo deste trabalho foi fornecer simulações mais próximas de ambientes que os usuários vivem no seu dia a dia sendo eles professores, alunos, desenvolvedor ou apenas interessados na área de computação em nuvem e simulação de nuvem.

5.2 Trabalhos Futuros

Para os trabalhos futuros é recomendado implementar as falhas que estão presentes em (ARGAWAL H.; SHARMA, 2015) e ainda não foram inseridas no simulador, por exemplo: negação de serviço, disco cheio, temporização, entre outras. No momento de escolher uma falha a ser implementada é necessário verificar se o simulador não irá necessitar de muitas alterações

e a possibilidade de inserção sem problemas.

Referências

AMAZONAWS. O que é aws?como funciona amazon web services. Página web disponível em https://aws.amazon.com/pt/what-is-aws/. 2020.

AMAZONSERVICES. Tipos de cloud computing. Página web disponível em https://aws.amazon.com/pt/types-of-cloud-computing/. 2020.

ARGAWAL H.; SHARMA, A. Comprehensive survey of fault tolerance techniques in cloud computing. *Conference on Computing and Network Communications*, n. 15, p. 16–19, 2015.

ARMBRUST M.; FOX, A. G. R. A view of cloud computing. *Journal of the ACM (JACM)*, v. 53, n. 4, p. 50–58, 2010.

AZUREMICROSOFT. Public cloud vs private cloud vs hybrid cloud. Página web disponível em https://azure.microsoft.com/en-us/overview/what-are-private-public-hybrid-clouds/. 2020.

CODY, B. V. K. *Openstack in Action*. 1th. ed. 20 Baldwin Road PO Box 761 Shelter Island NY 11964: Manning Publications Co., 2016. ISBN 9781617292163.

DANTAS, M. *Computação Distribuída de Alto Desempenho, Redes, Clusters e Grids Computacionais*. 1th. ed. [S.l.]: Axcel Books do Brasil Editora, 2005. ISBN 8573232404.

ERI T.; MAHMOOD, Z. P. R. *Cloud Computing Concetps, Technology Architecture*. 1th. ed. Upper Saddle River, New Jersey 07458: Prentice Hall, 2013. ISBN 9780133387520.

GSPD. Grupo de pesquisa em sistemas paralelos e distribuídos. Página web disponível em https://www.dcce.ibilce.unesp.br/spd/downport.php. 2020.

IBMCLOUD. O que é a plataforma ibm cloud? Página web disponível em https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform. 2020.

KUMARI P.; KAUR, P. A survey of fault tolerance in cloud computing. *Journal of King Saud University – Computer and Information Sciences*, 2018.

MANACERO A.; MENEZES, D. L. S. R. P. H. M. A. O. e. a. ispd: an iconic-based modeling simulator for distributed grids. 2012.

MANI D.; MAHENDRAN, A. Availability modelling of fault tolerant cloud computing system. *International Journal of Intelligent Engineering Systems*, v. 10, n. 1, p. 154–165, 2016.

NIVITHA K.; PABITHA, P. Fault diagnosis for uncertain cloud environment through fault injection mechanism. *Proceedings of the International Conference on Intelligent Computing and Control Systems*, v. 4, 2020.

OPENNEBULA. Documentation - opennebula. Página web disponível em http://docs.opennebula.io/. 2020.

OPENSTACK. O que é openstack? Página web disponível em https://www.openstack.org/software/. 2020.

REDHAT. Tipos de nuvens. Página web disponível em https://www.redhat.com/pt-br/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud. 2020.

RODRIGUES, M. A. J. *Dissertação de Mestrado, Universidade Estadual Paulista 'Júlio de Mesquita Filho, Otimização de alocação de máquinas virtuais em datacenter heterogêneo de sistema de computação em nuvem*. [S.l.], 2019.

SAIKIA P. L.; DEVI, L. Y. Fault tolereane techniques and algorithms in cloud computing. *International Journal of Computer Science Communication Networks*, v. 4, n. 1, p. 1–8, 2018.

SCHROEDER B.; GIBSON, A. G. A large-scale study of failures in high-performance computing systems. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, v. 7, n. 4, 2010.

SILVA DIOGO.; MANACERO, A. M. D. A. J. L. S. R. S. R. SimulaÇÃo de sistemas de computaÇÃo em nuvem para o ispd. *Interciência Sociedade*, v. 4, n. 1, 2015.

TAMASHIRO, C. B. O. *Dissertação de Mestrado, Universidade Estadual Paulista 'Júlio de Mesquita Filho, Modelagem de falhas em nuvem*. [S.l.], 2020.

TAMASHIRO C. B. O.; LOBATO, S. R. S. R. C. A. M. Modelagem de falhas em nuvem. *15th Conferência Ibérica de Sistemas e Tecnologias de Informação*, p. 1–6, 2020.

TORQUATO M.; MACIEL, P. A. J. U. I. M. An approach to investigate aging symptoms and rejuvenation effectiveness on software systems. *12th iberian conference on Information systems and technologies (CISTI)*, n. 17, p. 1–6, 2017.

VMWARESUITE. Documentation - vmaresuite. Página web disponível em https://docs.vmware.com/br/vRealize-Suite/index.html. 2020.

ZHANG Q.; CHENG, L. B. R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, v. 1, n. 1, p. 7–18, 2010.