André Pichinin Macuco

Implementação de Tolerância a Falhas em Simuladores de Nuvem

São José do Rio Preto

2021

André Pichinin Macuco

Implementação de Tolerância a Falhas em Simuladores 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

2021

André Pichinin Macuco

Implementação de Tolerância a Falhas em Simuladores 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 André Pichinin Macuco

Banca Avaliadora:

Profa. Dra. Carina Alexandra Rondini

Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto
2021

Resumo

A computação em nuvem é uma área que vem ganhando grande destaque com o passar dos anos, e cada vez mais tem se tornado indispensável para a vida de muitas pessoas, mesmo para aquelas que não trabalham ativamente criando sistemas em nuvem. Um grande desafio ao criar um sistema utilizando essa tecnologia envolve diminuir os custos de implementação e maximizar a eficiência. Tendo isso em vista, fazer simulações é extremamente vantajoso. Simuladores muitas vezes são gratuitos, o que facilita o acesso à qualquer tipo de pessoa, e, além disso, com simulações é possível identificar possíveis falhas no sistema de nuvem projetado, sem a necessidade de contratar efetivamente um provedor de nuvem – economizando, assim, tempo e dinheiro. Este trabalho se propõe a adequar as métricas já implementadas no simulador de nuvem ISPD para que usuários futuros consigam utilizar a ferramenta em seu máximo, com nenhum erro ou ambiguidade sobre as resoluções das simulações.

Palavras-chaves: Computação em nuvem, Simulação, Falhas, ISPD, Metricas.

Abstract

Cloud computing has conquer great prominence over the years and it has become increasingly indispensable for many people, even those who do not actively work on creating cloud systems. A major challenge when creating a system using this technology involves lowering implementation costs and maximizing efficiency. Bearing this in mind, running simulations is extremely advantageous. Simulators are often free of charge, which facilitates access for any kind of person; moreover, with simulations it is possible to identify possible flaws in the projected cloud system, not needing to hire effectively a cloud provider – thus, saving time and money. This work proposes to adapt the metrics already implemented in the ISPD cloud simulator so that future users can use the tool at its maximum, with neither error nor ambiguity about the resolutions of the simulations.

keywords: Cloud computing, Simulation, Faults, ISPD, Metrics.

Lista de ilustrações

Lista de abreviaturas e siglas

GSPD	Grupo de Sistemas Paralelos e Distribuídos
iSPD	iconic Simulator for Parallel and Distributed Systems
AWS	Amazon Web Services
IaaS	Infrastructure as a Service
PaaS	Plataform as a Service
SaaS	Software as a Service

Sumário

1 Introdução

Computação em nuvem é uma tecnologia emergente que oferece uma grande capacidade de processamento e armazenamento, por isso as pessoas estão mudando da computação tradicional para a computação em nuvem: por oferecer também maior confiabilidade, tolerância a falhas, amplo acesso à rede, uso sob demanda, etc.

Essa tecnologia visa superar muitos problemas surgidos devido à rápida evolução de empresas e à crescente quantidade de dados, visto que não seria viável tais dados em computadores pessoais por estes não suportarem a atual capacidade necessitada e também pelo crescente aumento no custo de manutenção de hardware.

Um fator importante para decidir a implementação de um serviço em nuvem é analisar a sua viabilidade e o seu custo. Para ajudar nessa análise, simuladores de nuvem estão sendo cada vez mais utilizados, a fim de avaliar os riscos de tais investimentos. Por isso, é necessário que esses simuladores, além de conseguirem executar os testes de simulação para detectarem possíveis falhas no sistema, tenham métricas claras e precisas.

Cada vez mais a computação em nuvem está sendo utilizada por empresas e instituições educacionais, portanto, os simuladores precisam entregar resultados claros e não ambíguos, para que pessoas sem experiência possam conseguir ver a viabilidade de seu projeto.

1.1 Objetivos

O objetivo deste trabalho é atualizar e melhorar as métricas do iSPD (MANACERO et al., 2012). Essas alterações são necessárias para que os usuários possam ter medidas mais claras e precisas dos resultados de suas simulações, visto que nem sempre estes são pessoas que estão habituadas a utilizá-lo - como por exemplo, novos alunos ou pesquisadores de outras instituições.

1.2 Motivação

O atual sistema de simulação de nuvem do iSPD foi construído a partir de um módulo já existente no sistema. As métricas foram juntamente importadas desse módulo, porém, elas se mostram confusas para alguns usuários, principalmente os que não tem familiaridade com o simulador, sendo assim essas métricas precisam ser adaptadas.

1.3 Justificativa

Quando se projeta 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. Usar simulações antes de uma implementação de fato é de extrema importância nos ambientes mais modernos. Utilizando injeção e tratamento de falhas podemos averiguar todos os contratempos eventuais que possam ocorrer no projeto real, evitando, assim, quaisquer desperdícios de recursos, sendo estes financeiros ou humanos. Por isso que é importante fazer melhorias constantes em simuladores e nuvens, uma dessas melhorias é atualizar as métricas das simulações, colocando métricas novas que auxiliam a obtenção de informações que antes o usuario nao tinha acesso, ou ainda a revitalização de métricas antigas, melhorando assim a sua legibilidade.

1.4 Metodologia

A metodologia adotada para a realização do trabalho se deu por uma série de etapas, consistida em uma pesquisa prévia sobre o simulador iSPD e os problemas apresentados pelas métricas do mesmo, na parte do sistema responsável pela simulação de nuvem. Em seguida, para que o trabalho fosse efetuado corretamente, foi feita uma outra pesquisa sobre computação em nuvem, simuladores de nuvem, falhas em nuvem e métricas dessas falhas; o material utilizado para a pesquisa se deu, em sua maioria, através de artigos científicos, documentação online, site, entre outros meios gratuitos e de fácil acesso.

1.5 Organização do texto

Este texto está dividido em cinco capítulos, incluso este, introdutório. O capítulo 2 contém a revisão bibliográfica e estado atual da arte, nele são detalhados os seguintes tópicos: computação em nuvem, descrição dos tipos de nuvem, serviços de nuvem, provedores de nuvem, simuladores de nuvem, uma descrição da organização e do funcionamento do iSPD, falhas em computação em nuvem, e, por fim, métricas utilizadas em simuladores de nuvem. O objetivo deste capítulo é abordar todos os tópicos necessários para o leitor ter o entendimento necessários dos assuntos discutidos neste trabalho. O capítulo capítulo 3 apresenta a descrição das atividades que serão desenvolvidas e como serão feitas, enquanto o capítulo 4 trará a apresentação dos resultados obtidos para validação do trabalho proposto. Finalizando, o capítulo 5 irá mostrar a conclusão do trabalho e a proposição de trabalhos futuros para a melhoria da ferramenta.

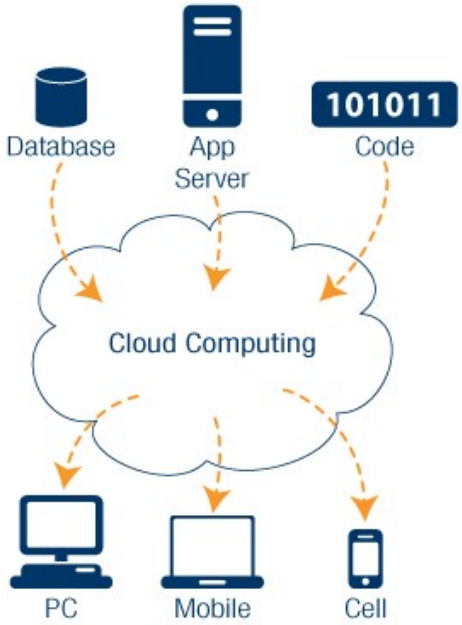
2 Revisão Bibliográfica

Neste capítulo serão apresentados e definidos alguns dos conceitos para a compreensão deste trabalho. Como, por exemplo, a definição de computação em nuvem, seus tipos e serviços, provedores de serviço em nuvem, simuladores de nuvem, apresentação do iSPD, falhas no ambiente de nuvem, e, por fim, métricas usadas em simuladores de nuvem.

2.1 Computação em Nuvem

Com o avanço da computação, cada vez mais sistemas e serviços são desenvolvidos por empresas e instituições de ensino, porém, há um problema em relação a essa crescente produção de dados: muitos dos hardwares atuais podem acabar não acompanhando as necessidades dessas instituições, fazendo com que elas tenham que gastar cada vez mais em hardwares para adquirir equipamentos melhores e em mais quantidade para suprir todas essas necessidades. Assim, a computação em nuvem surge como uma melhor forma de suprir essas necessidades, pois provê diferentes serviços *on demand* para cada tipo especifico de necessidade do cliente, ou seja, o cliente pode contratar apenas o serviço que precisa pelo tempo necessário, fazendo o uso de tecnologias que muitas vezes por motivos financeiros estariam fora de seu alcance, Na Figura 2.1 temos um exemplo do que é a computação em nuvem.

Figura 2.1 – Exemplo de computação em nuvem



Fonte: https://www.infoescola.com/informatica/computacao-em-nuvem/

Devido às melhorias surgidas pela computação em nuvem, foram necessárias novas definições para o uso dessa tecnologia recém criada. Tais como: o surgimento dos tipos de nuvem (pública, privada, híbrida) (AZUREMICROSOFT, 2021) e serviços *Infrastructure as a Service* (IaaS), *Plataform as a Service* (PaaS), *Software as a Service* (SaaS) (AMAZONSERVICES, 2021), que estão expandindo muito rapidamente.

2.2 Características da Computação em Nuvem

A computação em nuvem tem como premissa oferecer serviços, dentre os quais, o serviço de armazenamento em banco de dados e os servidores por meio da internet, conforme a nescessidade do usuario; por esse motivo, essa tecnologia apresenta um conjunto de características próprias nescessárias para o bom funcionamento dos serviços oferecidos. As principais caracteristicas, portanto, são:

- Custo: A computação em nuvem elemina os gastos de capital com a aquisição e a manutenção de novos hardwares e os substituem por despesas variadas muito menores da TI consumida.
- Escalabilidade: É a capacidade do sistema de não precisar entregar em excesso para poder suprir possíveis picos de requisições no futuro. Essa capacidade de expandir e diminuir os recursos pode se adequar à demanda e ao recursos financeiros do contratante.
- Velocidade: É possível implementar serviços em questões de minutos com a computação em nuvem, mesmo que tais serviços necessitem de grandes quantidades de recursos computacionais.
- Confiabilidade: A computação em nuvem facilita e também reduz os custos com backup, aumentando a capacidade que o sistema tem de se recuperar de condiçoes anormais.
- Uso por demanda: A computação em nuvem permite que os usuários possam fazer o uso do serviço apenas pelo tempo desejado, ou seja, podem adequar os serviços conforme sua necessidade.

2.2.1 Tipos de Nuvem

Nenhuma nuvem é igual a outra, visto que os serviços da nuvem que serão usados dependem da atual necessidade da empresa contratante. Contudo, conhecer os tipos de nuvem ajuda a entender as limitações de cada uma e de seus serviços, e também a escolher aquela que melhor se adequa ao seu modelo de negócios. Sendo assim, os tipos de nuvem são:

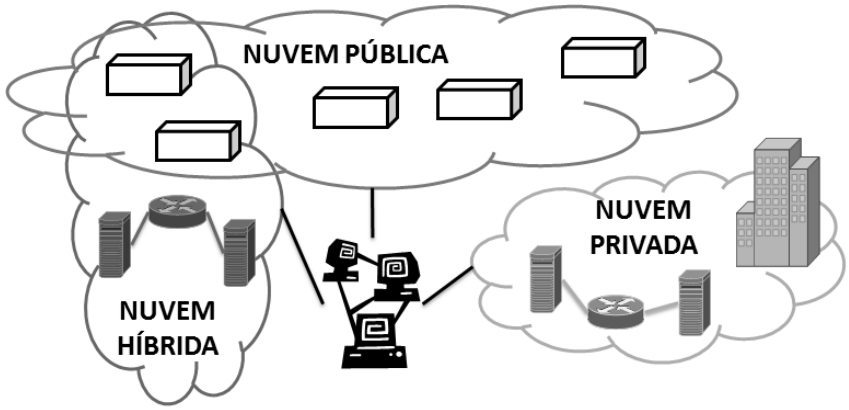
- Pública: As nuvens publicas são o modelo mais comum de implementação de computação em nuvem. Os recursos disponíveis na nuvem, como servidores e armazenamento, por

exemplo, pertencem ao provedor da nuvem e não ao contratante. Alguns dos maiores provedores de nuvens públicas são: Alibaba Cloud, Amazon Web Services (AWS), Google Cloud, IBM Cloud e Microsoft Azure.

- Privada: Uma nuvem privada consiste em recursos de cumputação em nuvem utilizados somente por uma única empresa ou instituição. A nuvem privada pode estar localizada fisicamente no local do contratante, ou hospedada em um provedor de serviços tercerizado. Com isso, há uma subdivisão de tipos de nuvens privadas:
- Gerenciada: Os contrataentes utilizam uma nuvem privada implantada e configurada por um fornecedor tercerizado.
- Dedicada: Uma nuvem dedicada dentro de outra nuvem, por exemplo: Uma nuvem exclusiva para um departamento de uma empresa, localizada dentro da nuvem privada da empresa em questão.
- Híbrida: É um tipo de computação em nuvem que combina uma nuvem pública com uma privada, permitindo, assim, que os dados e aplicativos se desloquem entre os dois ambientes.
- *Multicloud*: Multicloud é uma abordagem composta por mais de um serviço de um fornecedor de nuvem, sendo ela pública ou privada. Todas as nuvens híbridas são multicloud, porém, nem toda multicloud é uma nuvem híbrida. Para isso, várias nuvens devem estar conectadas.

A figura 2.2 é um exemplo simples dos tipos principais de nuvens e como elas estão organizadas.

Figura 2.2 – Arquiteturas de nuvem



Fonte: (AZEVEDO et al., 2012)

2.2.2 Serviços de Nuvem

Os serviços de nuvem são a forma pela qual os provedores de nuvem fornecem seus recursos aos seus clientes, de acordo com as necessidades e as demandas. Existem diversos tipos e modelos oferecidos, como por exemplo , o *Storage as a service* e o *Database-as-a-Service*, porém, normalmente, agrupamos todos em três categorias de modelos de serviços:(HöFER; KARAGIANNIS, 2011): *Infrastructure-as-a-Service* (IaaS), *Platform-as-a-Service* (PaaS) e *Software-as-a-Service* (SaaS). Esses três tipos de serviços podem ser implementados de forma única ou conjunta, de acordo com o desejo do cliente. Abaixo, cada serviço será explicado em maiores detalhes e exemplos de suas aplicações serão dados:

- IaaS: Nesse modelo de serviço, é oferecido uma plataforma virtualizada, que é uma evolução de um servidor virtual privado, já conhecido há anos. Os clientes compram os recursos, ao invés de servidores, *software*, e espaços em *data centers*, e, então, recebem uma cobrança pelo serviço consumido. Os clientes colocam seus próprios *softwares* em máquinas virtuais para que possam eles mesmos controlá-los e mantê-los. Exemplos desse tipo de serviço são: *Amazon Web Services* (AWS), Microsoft Azure, Digital Ocean e *Google Computing Engine* (GCE).

- PaaS: O PaaS oferece uma estrutura gerenciada de *software* de alto nível, na qual os clientes podem construir e implantar classes da aplicação por meio de ferramentas, ambiente e linguagens de programação suportadas pelo provedor. Os serviços de Paas são normalmente voltados para domínios especificos, tal qual o desenvolvimento de aplicativos da web, e dependem da linguagem de progrmação utilizada. Alguns exemplos de aplicações PaaS são: AWS, Windows Azure, Heroku e HPE; como pode-se observar, todos esses aplicativos citados estão prontos para o uso, fazendo com que o usuário tenha apenas a preocupação de usá-los da forma que melhor preferir.

- SaaS: O modelo SaaS geralmente fornece o *software* já pronto e é executada uma estrutura de nuvem para o cliente. Os serviços SaaS mais conhecidos são aplicativos na web, muitas vezes acessados por muitos dispositivos. Os clientes desses serviços não gerenciam ou controlam a infraestrutura desses aplicativos, apenas configurações específicas e limitadas do usuario costumam ser possíveis. Desse modo, o público-alvo desse serviço, em geral, são organizações que querem fazer o uso mais rápido e proveitoso da aplicação sem se preocupar com o seu funcionamento interno. Os principais exemplos desse tipo de serviço são: Paypal, DropBox, Google Drive, Adobe Creative Cloud e Netflix.

Segue, 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 à 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

Provedores de nuvem são empresas que fornecem ambientes de nuvem, sendo eles públicos ou privados, além de tambem oferecer serviços de IaaS, PaaS e SaaS. Nestá seção serão apresentados os principais provedores de serviços de nuvem.

2.3.1 Microsoft Azure Service Plataform

Microsoft Azure Service Plataform(Azure), também conhecida como Windows Azure, é a plataforma de computação em nuvem da Microsoft. A plataforma tem suporte para nuvens tanto públicas quanto privadas, podendo haver, também, o desenvolvimento de soluções híbridas, para facilitar o armazenamento de dados para os clientes. Como boa parte dos provedores de serviços em nuvens, o Azure fornece seus serviços através de virtualização de hardware, para que seus clientes não precisem se preocupar com a manutenção do sistema. Segundo (ZHANG; CHENG; BOUTABA, 2010), o Azure consiste em três componentes e cada um deles provê um conjunto específico de serviços para os usuários da nuvem. A seguir, cada um dos componentes será explicado 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, disponibiliza serviços de banco de dados na nuvem, baseados no SQL Server.
- .NET Services: O terceiro 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 e mais abrangente atualmente, oferencendo mais de 200 serviços completos de *data centers*. Oferece serviços de infraestrutura, armazenamento de dados, entrega de conteúdo e diversas outras funcionalidades, de acordo com as escolhas do cliente (AMAZONAWS, 2021). Assim como a Azure, a AWS também utiliza de virtualização para fornecer seus serviços para seus clientes. A AWS é dividida em componentes, e cada um desses componentes oferece um tipo de serviço diferente, sendo utilizado apenas com a requisição do cliente. Um desses componentes, e também um dos mais utilizados, é o *Amazon Elastic Compute Cloud*(Amazon EC2), um serviço Web que disponibiliza capacidade computacional segura e redimencionavel na nuvem. Foi desenvolvido com o objetivo de facilitar a computação em nuvem para os desenvolvedores, podendo obter e configurar a capacidade da nuvem sem muito esforço. O segundo componente, *Amazon Simple Storage Service*(Amazon S3), é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados e segurança de grande qualidade, além de ofercer recursos de gerenciamento fáceis de usar. O *Virtual Private Network* (VPN) é um serviço que permite inicializar recursos da AWS em uma rede virtual isolada, criada pelo próprio cliente.

2.3.3 Google Cloud

Google Cloud é a plataforma de nuvem do Google. Apesar de ser a plataforma de nuvem mais recente no mercado, em comparação com as outras duas, vem crescendo bastante nos últimos anos. A plataforma foi criada em código aberto, o que ajuda a desenvolver *softwares* mais rapidamente, inovar com facilidade, escalonar com mais eficiência, além de reduzir os riscos tecnologicos(GOOGLECLOUD, 2021). A Google Cloud oferece também um serviço chamado BigTable, um banco de dados não relacional, gerenciável, e em escalas de petabytes capazes de lidar com milhares de requesições por segundo; o próprio Google utiliza o serviço. Além disso, Google Cloud é o serviço de nuvem mais barato disponível no mercado, tornando a tecnologia ainda mais acessível para todo tipo de público.

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, há alguns dos serviços prestados pelos provedores apresentados acima:

2.4 Simuladores de Nuvem

Apesar do início da computação em nuvem de mais de uma década atrás, ainda existem várias questões desafiadoras que precisam ser estudadas. Porém, é impraticável que instituições de ensino e companhias de pequeno e médio porte tenham uma nuvem física para realizar pesquisas sobre computação em nuvem. Dessa forma, uma solução seria utilizar simuladores de

Tabela 2.2: Comparação entre provedores de serviços de nuvem

Categoria do serviço	Microsoft Azure	Amazon AWS	Google Cloud
Computação	Máquinas Virtuais do Azure	Amazon Elastic Compute Cloud (EC2)	Compute Engine
PaaS	Serviço de app do Azure *Windows*	AWS Elastic Beanstalk	Aplicações gerais e App Engine
FaaS	Computação sem servidor do Azure Functions	AWS Lambda	Cloud Functions
Armazenamento	Armazenamento em disco do Azure	Amazon Elastic Block Store (EBS)	Persistent Disk
Redes virtuais	Rede virtual do Azure	Nuvem privada virtual (VPC, na sigla em inglês) da Amazon	Nuvem privada virtual

nuvens para simular uma nuvem real.

Um simulador de nuvem ajuda a modelar vaáios tipos de aplicações em nuvem, criando máquinas virtuais e outras utilidades que podem ser configuradas facilmente, tornando mais fácil a análise.(SURYATEJA, 2016).

2.4.1 CloudSim

CloudSim é a ferramenta de simulação mais popular para ambientes de simulação de nuvem. É um simulador orientado a eventos cujo código é escrito em Java, uma das mais famosas linguagens de programação; por isso, os módulos do *CloudSim* são fáceis de entender, além de seu simulador ser também escrito em código aberto. (CALHEIRO et al., 2010)

O *CloudSim* contém as seguintes recursos:

- Suporte de modelagem de simulações em ambientes de computação em grande escala
- Uma plataforma independente para modelar nuvens, corretores de serviços, políticas de provisionamento e alocação.
- Suporte para simulação de conexões de rede entre os elementos do sistema simulado.
- Facilidade para simulação de ambiente de nuvem federada, que contém recursos inter-redes de domínios públicos e privados.
- Disponibilidade de um mecanismo de virtualização que auxilia na criação e no gerenciamento de vários serviços virtuais

2.4.2 CloudAnalyst

Feito a partir do *CloudSim*, o *CloudAnalyst* foi projetado a partir da ideia de avaliação do desempenho de aplicações de nuvens distribuídas em grande escala, com alta carga de trabalho para o usuário, e que são distribuídas geograficamente em vários *data centers*. (WICKREMASINGHE; CALHEIROS; BUYYA, 2010).

Os recursos do *CloudAnalyst* são os seguintes:

- Interface gráfica do usuário fácil de usar.
- Capacidade de definir uma simulação com alto grau de configurabilidade e flexibilidade.
- Capacidade de repetir os experimentos com pequenas modificações.
- Gerar saída gráfica na forma de gráficos e tabelas.
- Uso de tecnologia consolidada e facilidade de adicionar extensões.

Ao usar o CloudAnalyst, os desenvolvedores de aplicações em nuvem serão capazes de determinar a melhor estratégia para alocação de recursos entre os *data centers*, além de selecionar o *data centers* ideal para atender as suas solicitações específicas e, assim, minimizar os custos dessas solicitações. No mais, terão uma maior facilidade em usar a ferramenta devido a sua interface gráfica, suas tabelas e gráficos, disponibilizados como saída de uma simulação.

2.4.3 GreenCloud

GreenCloud é um simulador de nuvem desenvolvido a partir do simulador de rede NS2 e considerando ambientes que priorizam o consumo consciente de energia. O *GreenCloud* é projetado para que se possa calcular o consumo de energia especifico de qualquer componente do *data centers*, como *switch*, *gateway*, etc. Além disso, o simulador também mostra a distribuição de carga de trabalho no sistema. (GREENCLOUD, 2021).

Os principais recursos do *GreenCloud* são:

- Foco em rede de nuvem e conscientização sobre energia.
- Simulação de CPU, memória, armazenamento e recursos de rede.
- Modelos de energia independentes para cada tipo de recurso.
- Suporte de virtualização e migração de máquinas virtuais.
- Alocação de recursos com base na rede.
- Complete TCP/IP implementation.

- Interface gráfica amigável.
- Código aberto.

2.5 iSPD

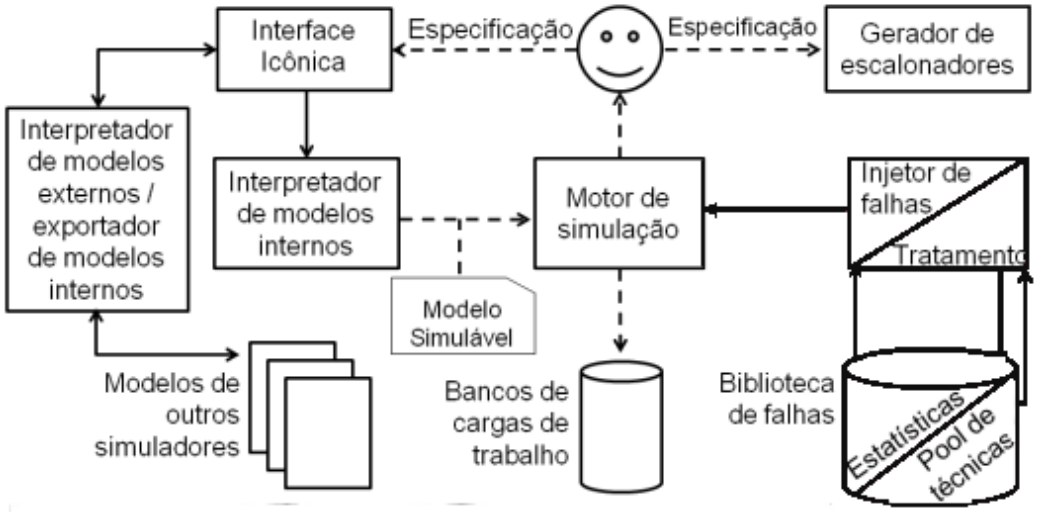
O iSDP (*iconic Simulator for Parallel and Distributed Systems*) é um simulador orientado a eventos discretos que foi construído pensando na simulação de grades, e posteriormente adaptado para, também, simular nuvens computacionais, escrito na linguagem de programação orientada a objetos Java. O principal objetivo do iSPD é a facilidade do uso, utilizando uma interface icônica de modelagem e módulos para gerar políticas de escalonamento e seleção de serviços de *cloud* (MANACERO et al., 2012). Com esses recursos, é possível que os usuários, mesmo sem conhecimentos de programação, possam utilizar o simulador.

O iSPD tem varios módulos, e cada um deles desempenha uma função específica para o funcionamento da ferramenta(SILVA et al., 2015). Um diagrama de relacionamento entre os módulos constituintes do iSPD é mostrado na Figura 2.3, e, abaixo, cada um desses módulos 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: Este 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 gerados 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 disponíveis para uso na simulação, e gerar novas políticas de escalonamento de forma 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, trabalha em conjunto com o injetor sempre que uma falha é adicionada na simulação.

Figura 2.3 – Diagrama de interação dos módulos do iSPD



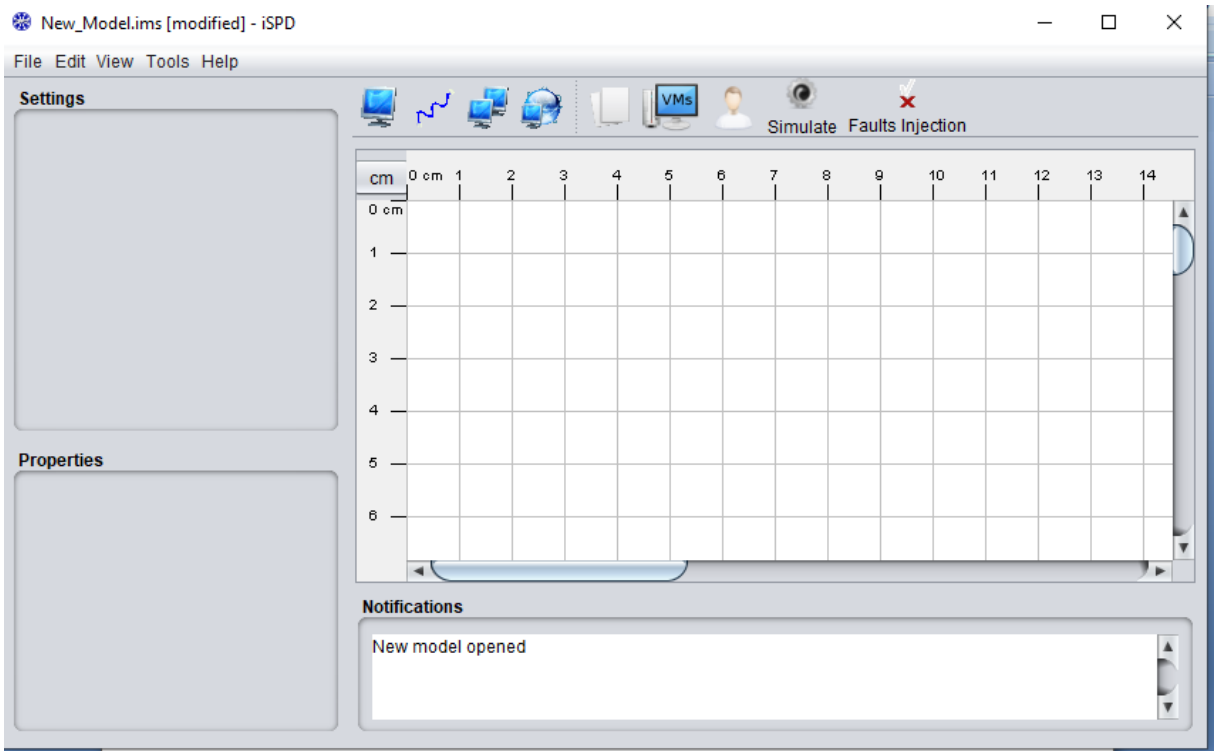
Fonte: Tamashiro (2020)

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, 2021). O simulador pode ser utilizado para fazer testes tanto por profissionais quanto por estudantes, para auxiliar em seus estudos.

A versão atual permite a escolha de três tipos de serviços para modelagem: *grid*, IaaS, PaaS. Depois de escolher entre uma das opções anteriores, o usuário pode escolher através da interface quais componentes farão parte da simulação, sendo as opções: computadores, clusters, conexões, usuários, máquinas virtuais, ou rede. Na Figura 2.4, temos um exemplo da interface principal de simulação do iSPD.

Figura 2.4 – 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 decidir 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. Com toda essa capacidade de escolha do usuário é possivel realizar uma simulação muito bem detalhada, permitindo, assim, uma coleta de dados ainda mais precisa. Além disso, o iSPD permite exportar modelos Simgrid e GridSim, bem como os escalonadores fornecidos pelo usuário.

Em seguida, os resultados são mostrados na forma de arquivo xml, gerado pelo simulador, e esse arquivo fica disponível para o usuário.

2.6 Falhas em Nuvem

Uma das grandes preocupações quando se projeta e se mantém um sistema de computação em nuvem, e tambem outros sistemas, em geral, é entender como detectar e tratar falhas que eventualmente podem ocorrer e que afetam o funcionamento correto da aplicação. Nos sistemas de nuvem de maior porte, há uma grande dificuldade para obter informações sobre as falhas, devido ao curto espaço de tempo necessário para coletá-las; assim, é gerada uma baixa quantidade de dados para uma análise mais profunda e precisa de suas causas (SCHROEDER; GIBSON, 2010). Um dos principais objetivos de se estudar falhas em um ambiente de nuvem é evitar a

indisponibilidade do sistema, pois uma das principais vantagens da computação em nuvem é poder fonercer os serviços ofertados a qualquer momento, conforme as necessidades do cliente (MANI; MAHENDRAN, 2016).

2.6.1 Métricas

A maioria das aplicações em computação em nuvem tem como foco de suas métricas a escalabilidade do sistema, a elasticidade e a eficiência. Contudo, existe uma imensa quantidade de métricas para esses valores, devido ao fato de não serem padronizados. Essa grande quantidade mostra a importância dessas propriedades e faz com que projetistas de sistemas em nuvem escolham as melhores métricas dentro do contexto do sistema projetado (SEBASTIAN LEHRIG.AND HENDRIK, 2015).

No campo dos simuladores de nuvem e nas falhas em nuvem, essas métricas também se mostram extremamente relevantes, principalmente a eficiência, pois elas estão ligadas diretamente à experiência do usuário no sistema, e muitas vezes são responsáveis por dar o retorno a este usuário para informá-lo se a sua simulção foi bem sucedida ou não.

É necessário que as métricas, principalmente em situações de falha, sejam claras e não ambíguas, para que até um usuário sem muita experiência consiga identificar se ocorreu algum erro, juntamente com sua causa, podendo, assim, adequar o sistema para que mais nenhum tipo de falha aconteça.

2.7 Considerações finais

Este capítulo apresentou os principais conceitos que abrangem a computação em nuvem, dando destaque nas principais características da tecnologia e dos seus serviços ofertados - os provedores de serviço, alguns dos simuladores de nuvem mais famosos, uma apresentação do iSPD e seu estado atual, e, por fim, a relevância de métricas adequadas para o caso de falhas em um sistema de nuvem.

3 Desenvolvimento

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

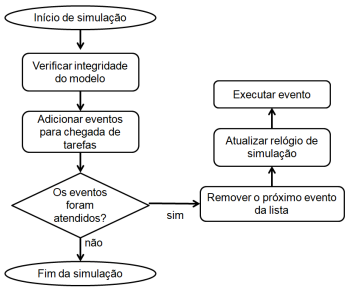
3.1 Apresentação e organização geral do iSPD

Para compreender melhor o trabalho de adição de métricas novas às falhas do 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 funcionamento 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 compõem a ferramenta, apenas as necessárias para a 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 analisados 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 et al., 2012).

Figura 3.1 – Fluxograma de Simulação do iSPD



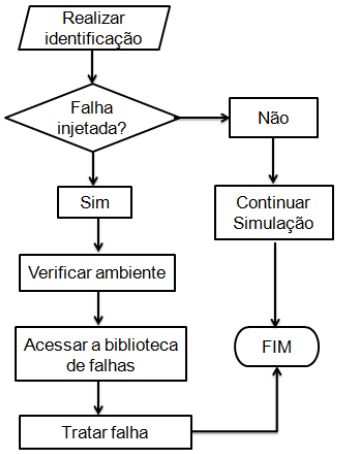
Fonte: Tamashiro (2020)

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.

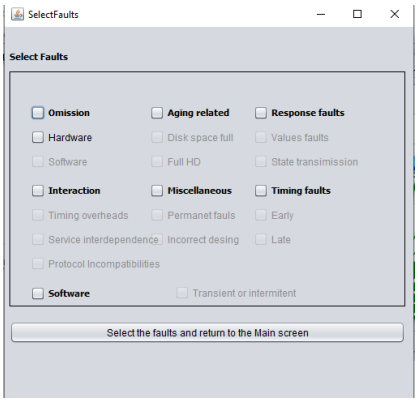
Figura 3.2 – Fluxograma da identificação e tratamento de falha



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, envelhecimento, resposta, softwares, temporização, interação e diversas.

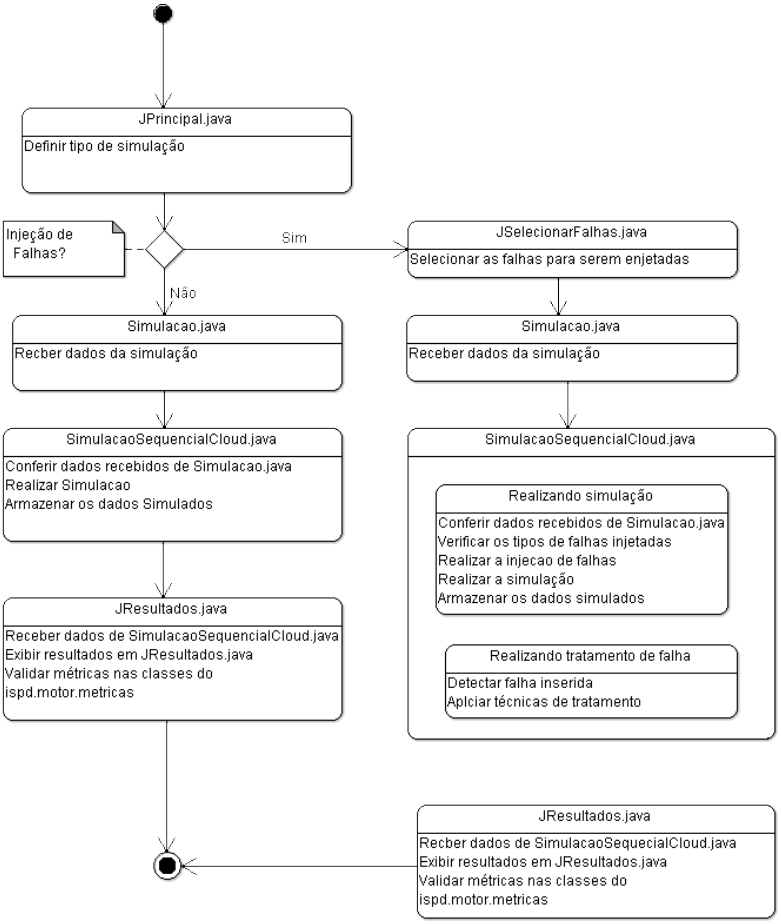
Figura 3.3 – Interface de inserção de falhas iSPD



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 há 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

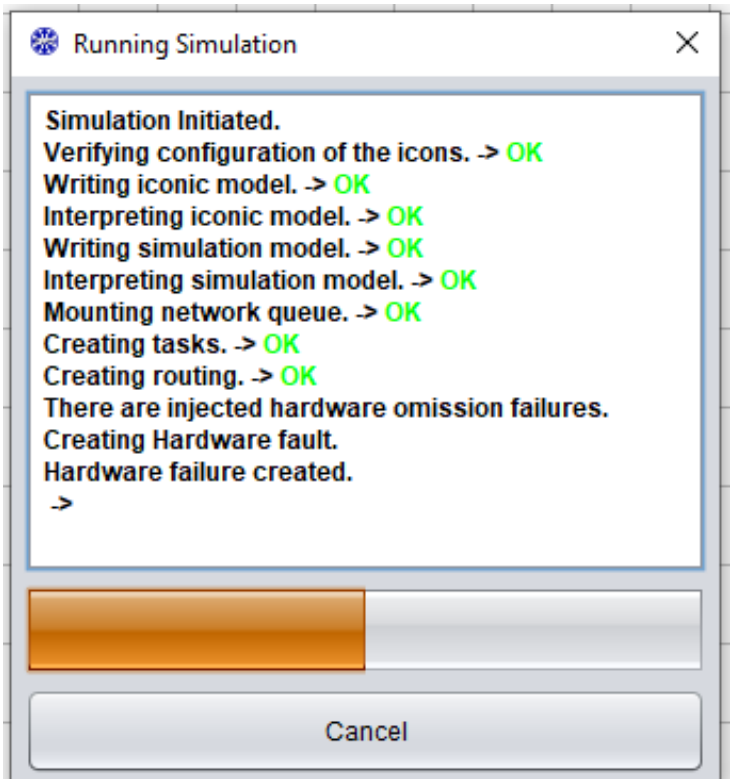


Fonte: Tamashiro (2020)

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 a simulação é finalizada uma mensagem indicando que a falha foi tratada é exibida. Na figura 3.5 é mostrado 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 biblioteca de falhas dentro do sistema. As técnicas de tratamento estão localizadas em classes específicas 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 (2021)

3.3 Métricas na 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 a implementação de uma métrica antes não existente no simulador que é o tempo em que se leva para tratar uma falha. É uma informação relevante visto que antes o simulador só contava o tempo total da simulação e não tinha como saber quanto do tempo da simulação foi utilizado apenas para corrigir a falha.

Essa métrica será implementada nos dois tipos de falhas apresentadas pelo simulador, as falhas de respostas e as falhas de estado de transmissão, elas apresentam as seguintes características.

- 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 se divide 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 a métrica que será implementada no iSPD será: de tempo até tratamento de falha. A implementação dela será feita utilizando- se de dados já existentes no simulador e que apenas não eram direcionados para a partes de métricas de falhas, era previamente utilizado apenas para medir o tempo de execução de tarefas. A implementação das classes necessárias será feita seguindo o padrão já adotado pelo código fonte do simulador para que sua implementação possa ser feita rapidamente e também para que fique fácil de entender caso a implementação seja revisada em um trabalho futuro.

3.4 Considerações Finais

Ao longo deste capítulo foram apresentados os principais componentes do iSPD, uma explicação sobre as falhas inseridas nele e também uma descrição das modificações propostas por esse trabalho e o impacto delas no estado atual do simulador.

4 Resultados

Para a implementação da nova métrica foi pensado seguir como exemplo as métricas antigas do iSPD e fazer a implementação seguindo o modelo delas, contudo por falta de familiaridade com a linguagem que o sistema foi escrito e pela complexidade do sistema, o iSPD é um simulador bem antigo onde várias pessoas já fizeram alterações neles podendo até ser considerado um sistema legado, não foi possível fazer a implementação da métrica dentro do tempo planejado contudo os arquivos do simulador foram mapeados para a implementação, A seção a seguir irá apresentar o mapeamento das classes e arquivos.

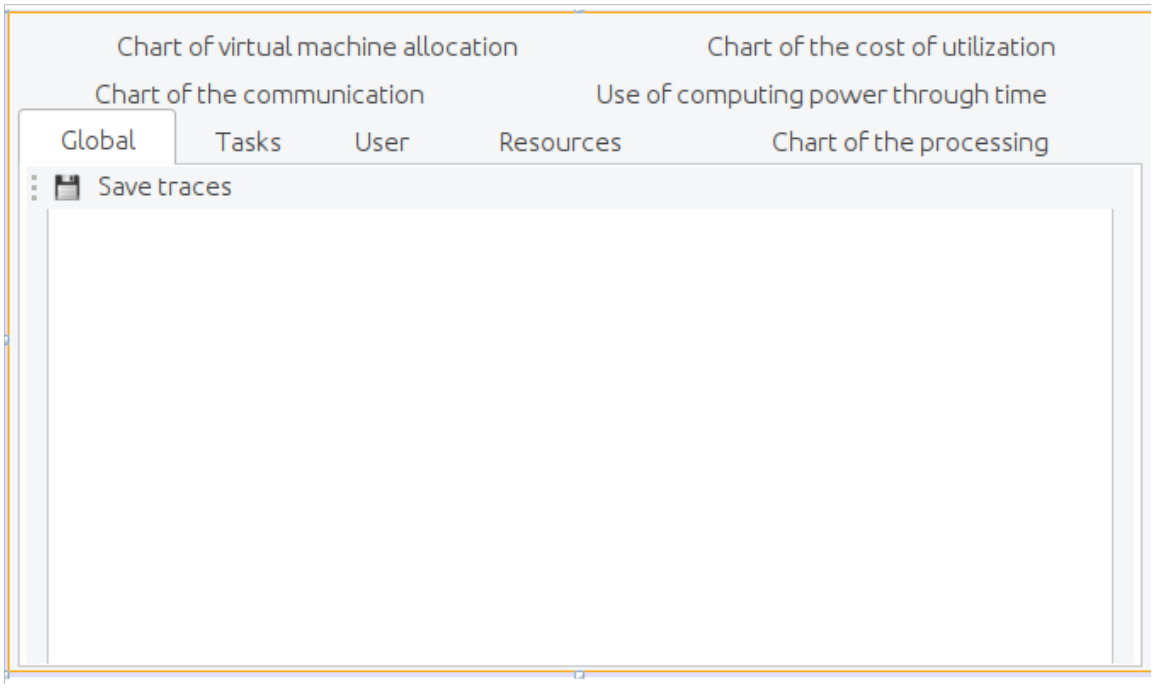
4.1 Mapeamento

As métricas utilizadas pelo simulador nas simulações de nuvem podem ser vistas no arquivo *JResultadosCloud.java*, localizado na pasta *iSPD.gui* do projeto, esse arquivo é responsável por fazer a tela se resultado do simulador no final da simulação, na parte do import do arquivo pudesse ver que as métricas são adquiridas de outros arquivos do projeto, os arquivos responsáveis por medir o tempo da simulação são os arquivos *Metricas.java* e *MetricasGlobais.java* os dois arquivos estão localizados na pasta *iSPD.motor.metricas*.

Os dois arquivos de métricas devem ser alterados para a adição da nova métrica, as alterações devem ser feitas seguindo como base os modelos das outras funções e variáveis para manter a padronização do código evitando assim aumentar ainda mas a complexidade do sistema, deve se usar como base de alteração funções e variáveis relacionadas ao tempo da simulação para auxiliar na implementação como por exemplo a classe *getTempoMedioProcessamento*.

Após a adição das métricas nesses arquivos devesse adicionar elas no *JResultadosCloud.java*, fazendo as alteraçoes para elas ficarem posicionadas no final da lista de métricas, pode-se usar as ferramenta de *Design* do *Apache NetBeans*, *software* que é utilizado para rodar o simulador, para conferir se a métrica adicionada está sendo mostrada no lugar devido.

Figura 4.1 – Janela de metricas do iSPD

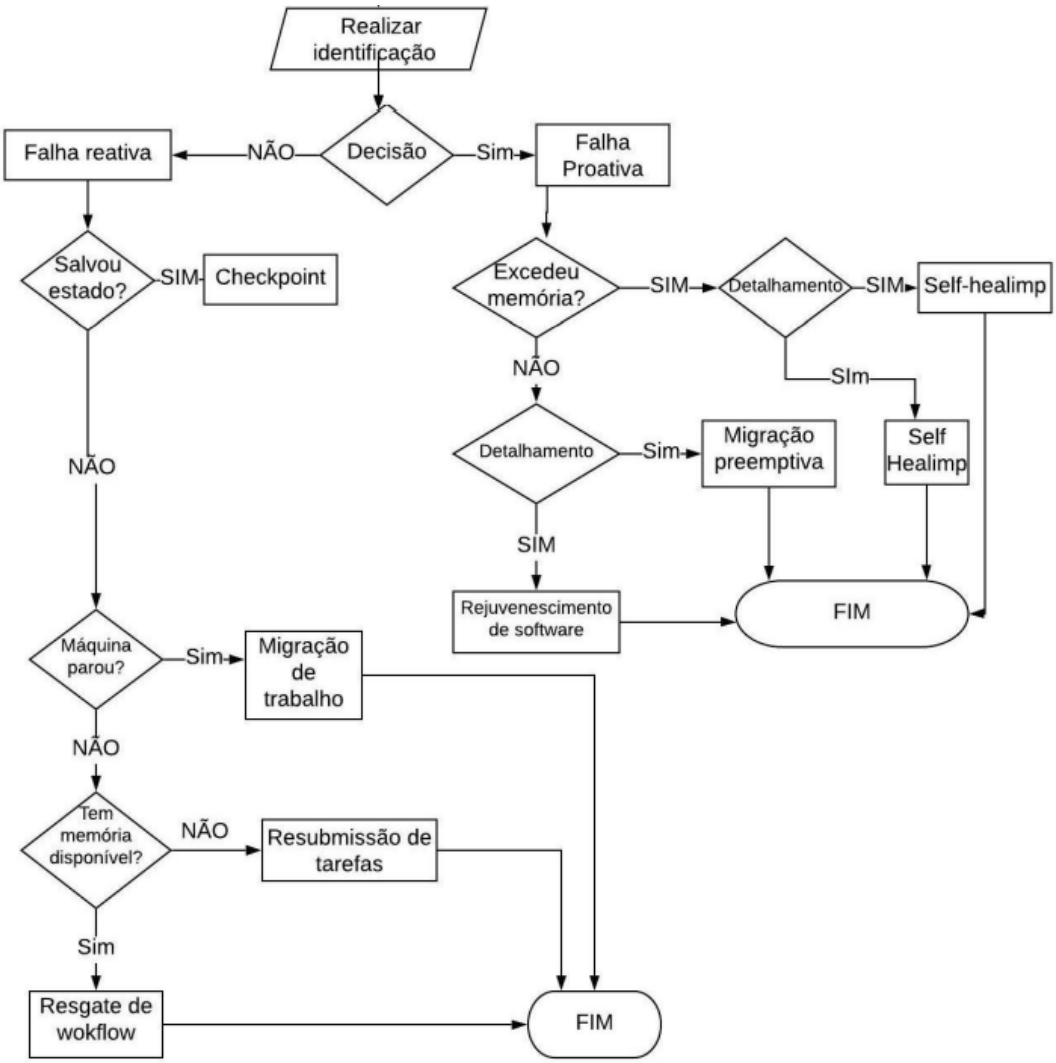


Fonte: Autor (2021)

4.2 Funcionamento e Implementação

As falhas no iSPD podem ser identificadas a partir de um sistema teste e um sistema injetor integrado a um sistema monitor e ao sistema de simulação. O registro dessas falhas é feito pelo sistema injetor, que por sua vez, está ligado às bibliotecas de falhas as quais têm a modelagem das principais falhas do simulador. Sua validação ocorre quando há uma simulação desta falha, representado pelo fluxograma quanto à detecção de ocorrências de falhas e erros na Figura 4.2.

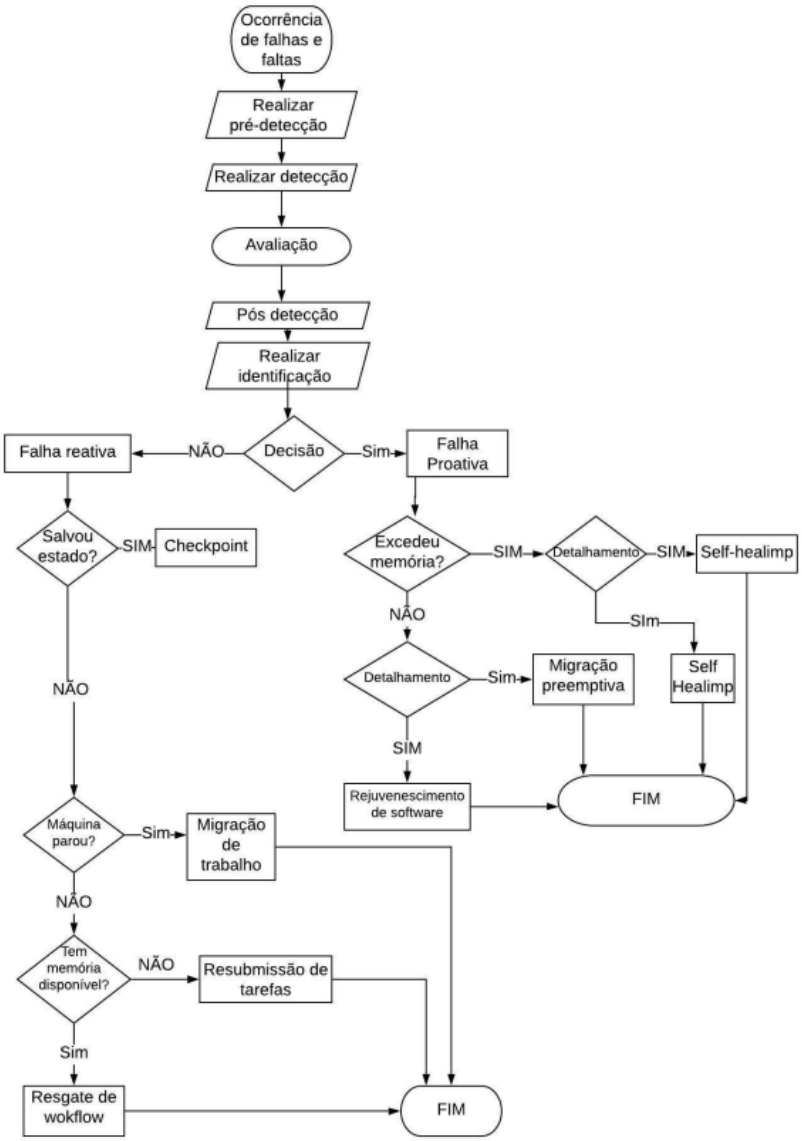
Figura 4.2 – : Fluxograma quanto à identificação de falhas



Fonte: Tamashiro (2020)

A Figura 4.2 refere-se apenas ao processo de identificação de falhas. Quando identificada há a continuidade do fluxo da informação para que haja sua reparação e recuperação do sistema caso necessário. No fluxograma apresentado na Figura 4.3 ilustra-se este processo de forma integral: de sua ocorrência até a recuperação do sistema.

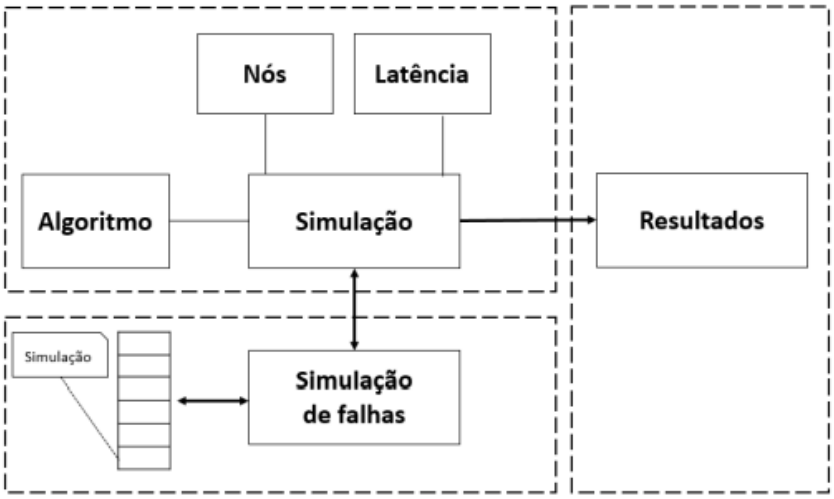
Figura 4.3 – : : Fluxograma do processo quanto à detecção de ocorrências de falhas



Fonte: Tamashiro (2020)

Durante o processo de simulação de falhas, o sistema monitor irá gerar uma simulação com uma árvore de chaves, para que esta faça testes aleatórios com atribuição de valores de números inteiros, injetando falhas nesta árvore e assim verificar se as simulações serão possíveis. Caso haja alguma simulação não possível, o sistema irá buscar a sua biblioteca de mensagens para reportar ao usuário o tipo de falha e como corrigi-la. No processo de simulação de falhas, o simulador utilizado necessita de dados para realizar a simulação, como por exemplo a quantidade de máquinas utilizadas na simulação, a quantidade de nós existentes, a latência, a granularidade. Estes dados representam os pré-requisitos para a simulação após a escolha do algoritmo que será simulado, apresentado no modelo conceitual, representado pela Figura 4.4.

Figura 4.4 – : :Modelo conceitual dos pré-requisitos para simulação



Fonte: Tamashiro (2020)

Nesse momento na parte de Resultados que são gerados as métricas da simulação, é nessas parte do simulador que as alterações precisam ser feitas para poder ser adicionado as métricas de tempo de falha na execução da simulação, essa parte do simulador se encontra especificamente no motor do simulador, Na Figura 4.5 podemos ver um exemplo no código onde se mostra na tela para o usuário o tempo total da simulação, o mesmo poderia ser feito com o tempo em que se ocorre uma falha adicionando essa métrica nesse pedaço do código, o arquivo mostrado é o JResultadosCloud.java localizado na pasta ispd.motor.metricas.

Figura 4.5 – : :Imagem da classe getResultadosGlobais

```
723     private String getResultadosGlobais(MetricasGlobais globais) {
724         String texto = "\t\tSimulation Results:\n\n";
725         texto += String.format("\tTotal Simulated Time = %g \n", globais.getTempoSimulacao());
726         texto += String.format("\tSatisfaction = %g %%\n", globais.getSatisfacaoMedia());
727         texto += String.format("\tIdleness of processing resources = %g %%\n", globais.getOciosidadeComputacao());
728         texto += String.format("\tIdleness of communication resources = %g %%\n", globais.getOciosidadeComunicacao());
729         texto += String.format("\tEfficiency = %g %%\n", globais.getEficiencia());
730         //texto += String.format("\t"
731         if (globais.getEficiencia() > 70.0) {
732             texto += "\tEfficiency GOOD\n\n ";
733         } else if (globais.getEficiencia() > 40.0) {
734             texto += "\tEfficiency MEDIA\n\n ";
735         } else {
736             texto += "\tEfficiency BAD\n\n ";
737         }
738         texto += "\t\tCost Results:\n\n";
739         texto += String.format("\tCost Total Processing = %g $\n", globais.getCustoTotalProc());
740         texto += String.format("\tCost Total Memory = %g $\n", globais.getCustoTotalMem());
741         texto += String.format("\tCost Total Disk = %g $\n\n", globais.getCustoTotalDisco());
742         texto += "\t\tVM Alocation Results:\n\n";
743         texto +=String.format("\tTotal of VMs alocated = %d \n", globais.getNumVMsAlocadas());
744         texto +=String.format("\tTotal of VMs rejected = %d \n", globais.getNumVMsRejeitadas());
```

Fonte: Proprio autor (2021

Apesar das métricas serem mostradas no arquivo Resultados Globais elas não são calculadas nele, sendo assim alterações devem ser feitas em outros arquivos e importados

nesse respectivo arquivo. Na figura 4.6 podemos ver a mesma variável de tempo de simulação em outro arquivo, Métricas Globais.java, nesse arquivo deve se fazer também as alterações devidas para que se possa mostrar o resultado final, pode-se seguir como base o próprio método setTempoSimulação para se fazer a implementação devida.

Figura 4.6 – : :Definição das funções no arquivo MetricasGlobais.java

```
274
275     public void setTempoSimulacao(double tempoSimulacao) {
276         this.tempoSimulacao = tempoSimulacao;
277     }
278
279     public void setSatisfacaoMedia(double satisfacaoMedia) {
280         this.satisfacaoMedia = satisfacaoMedia;
281     }
282
283     public void setOciosidadeComputacao(double ociosidadeComputacao) {
284         this.ociosidadeComputacao = ociosidadeComputacao;
285     }
286
287     public void setOciosidadeComunicacao(double ociosidadeComunicacao) {
288         this.ociosidadeComunicacao = ociosidadeComunicacao;
289     }
290
291     public void setEficiencia(double eficiencia) {
292         this.eficiencia = eficiencia;
293     }
```

Fonte: Proprio autor (2021

Como a métrica de tempo de falha na simulação é calculado de um modo parecido com o tempo total da simulação, para fazer a implementação dessa métrica pode-se usar como base o tempo total de simulação, mostrado nas imagens 4.5 e 4.6, fazendo as alterações necessárias para apenas contar o tempo em que uma falha está ativa no simulador e não o tempo inteiro da simulação

Após as implementações deve se fazer os teste para ver se as métricas estão adequadas para o sistema, para os testes pode-usar como base os testes em (TAMASHIRO, 2020) e fazer uma comparação dos resultados para saber se a adição das métricas de tempo de falha contribuíram positivamente para o *feedback* do usuário ou se não interferiram na experiência dele.

Contudo utilizando de outros trabalhos como base (YIFEI, 2016) podemos ver o quão relevante é saber o tempo de falha na hora de uma simulação e também as quantidades de infomrções adicionais que pode ser extraída a partir desse simples dado, podendo auxiliar na melhora do modelo simulado e além disso na melhora do simulador.

4.3 Considerações finais

Ao longo do capítulo foi discutido o mapeamento do iSPD e a relevância da adição da métrica de tempo de simulação para o simulador iSPD, pode-se atestar a relevância dessa métrica a partir de trabalhos citados e o mapeamento do simulador se mostra importante devida a complexidade de seu código.

5 Conclusões e Trabalhos Futuros

5.1 Conclusões

Conforme foi apresentado ao longo do trabalho o objetivo foi o estudo e implementação da métrica de tempo de falha em um simulador de nuvem, no caso o iSPD pertencente ao Grupo de Sistemas Paralelos 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.

Contudo por falta de familiaridade da linguagem e complexidade do código fonte do simulador, não foi possível fazer a implementação da métrica como desejado no começo do projeto, contudo mesmo sem a implementação avanços foram feitos no entendimento do funcionamento do simulador e na necessidade da adição de tal métrica e na atualização de métricas já existente no simulador.

O maior desafio encontrado na realização do trabalho foi compreender internamente o funcionamento do iSPD, como é um sistema muito grande e complexo e também bem antigo foi difícil compreender plenamente o funcionamento do simulador, a falta de pessoas que ainda trabalham ativamente na manutenção e adição de novas funcionalidades do simulador ajudaram ainda mais na complexidade do projeto, pois a falta de pessoas para discutir e orientar sobre o funcionamento do simulador e suas classes foi também um fator que ajudou a somar na complexidade do projeto.

O principal objetivo deste trabalho foi adicionar a métrica de tempo de falha no simulador para melhorar a visualização das falhas no ambiente de nuvem do simulador, mesmo esse objetivo não sendo alcançado no final o trabalho ainda se mostra relevante por ajudar na compreensão do funcionamento do simulador de nuvem iSPD e na nescessidade da melhorias de suas métricas para que todos os tipos de possíveis usuários dele, alunos, professores, desenvolvedores ou apenas interessados a área de computação em nuvem possam receber medidas mais precisas de suas simulações.

5.2 Trabalhos Futuros

Para os trabalhos futuros além da implementação dessa métrica apresentada no trabalho pode-se pensar em implementar outras métricas dependente dessa, como por exemplo tempo médio de falha, tempo máximo de falha e tempo mínimo de falha, a adição dessas métricas podem auxiliar ainda mais na observação das simulações.

Referências

AMAZONAWS. O que é aws?como funciona amazon web services. Página web disponível em https://aws.amazon.com/pt/what-is-aws/. 2021.

AMAZONSERVICES. Tipos de cloud computing. Página web disponível em https://aws.amazon.com/pt/types-of-cloud-computing/. 2021.

AZEVEDO, E.; SILVA, C.; SIMõES, R.; DANTAS, R.; FERNANDES, S.; SADOK, D.; KAMIENSKI, C. Nuvem pública versus privada: Variações no desempenho de infraestrutura para elasticidade. 2012.

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/. 2021.

CALHEIRO, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. F. D.; BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software – Practice AND Experience*, 2010.

GOOGLECLOUD. Por que o google cloud? Página web disponível em https://cloud.google.com/why-google-cloud. 2021.

GREENCLOUD. Greencloud - the green cloud simulatorídos. Página web disponível em https://greencloud.gforge.uni.lu/index.html. 2021.

GSPD. Grupo de pesquisa em sistemas paralelos e distribuídos. Página web disponível em https://www.dcce.ibilce.unesp.br/spd/downport.php. 2021.

HöFER, C.; KARAGIANNIS, G. Cloud computing services: taxonomy and comparison. *Internet Serv Appl*, n. 4, p. 81–94, 2011.

MANACERO, A.; LOBATO, R.; OLIVEIRA, P.; GARCIA, M.; GUERRA, A.; AOQUI, V.; MENEZES, D.; SILVA, D. D. ispd: an iconic-based modeling simulator for distributed grids. *Proc. of the 45th Annual Simulation Symposium*, p. 5:1–5:8, 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.

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.

SEBASTIAN LEHRIG.AND HENDRIK, E. S. B. Scalability, elasticity, and efficiency in cloud computing: a systematic literature review of definitions and metrics. *International ACM SIGSOFT Conference on Quality of Software Architectures*, v. 11, p. 83–92, 2015.

SILVA, D.; MANACERO, A.; MENEZES, D.; ARTHUR, J.; LOBATO, S. R.; SPOLON, R. Simulação de sistemas de computação em nuvem para o ispd. *Interciência & Sociedade*, v. 4, n. 1, 2015.

SURYATEJA, P. S. A comparative analysis of cloud simulators. *I.J. Modern Education and Computer Science*, v. 4, p. 64–71, 2016.

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

WICKREMASINGHE, B.; CALHEIROS, R. N.; BUYYA, R. 2010 24th ieee international conference on advanced information networking and applications. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, p. 446–452, 2010.

YIFEI, Z. *Estudo e simulação de algoritmos de escalonamento para grades móveis voltados à conectividade dos dispositivos móveis*. [S.l.], 2016.

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