UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Modelagem e desenvolvimento de uma arquitetura distribuída para o gerenciamento de recursos híbridos em nuvem privada

Gabriel Tomiatti Andreazi

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Gabriel Tomiatti Andreazi

**Modelagem e desenvolvimento de uma arquitetura
distribuída para o gerenciamento de recursos híbridos em
nuvem privada**

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientador: Prof. Dr. Júlio Cezar Estrella

USP – São Carlos
Dezembro de 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

A557m Andreazi, Gabriel Tomiatti
 Modelagem e desenvolvimento de uma arquitetura
distribuída para o gerenciamento de recursos híbridos
em nuvem privada / Gabriel Tomiatti Andreazi;
orientador Júlio Cezar Estrella. -- São Carlos,
2020.
 93 p.

 Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2020.

 1. Computação em nuvem. 2. Provedor de Serviços.
3. Gerenciamento de Recursos. I. Estrella, Júlio
Cezar , orient. II. Título.

Gabriel Tomiatti Andreazi

**Modeling and development of a distributed architecture for
managing hybrid resources in a private cloud**

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Júlio Cezar Estrella

USP – São Carlos
December 2020

Este trabalho é dedicado a todas as pessoas que acreditaram em Mim.

AGRADECIMENTOS

Primeiramente gostaria de prestar meu mais sinceros agradecimentos a duas pessoas que não estão mais aqui, mas que em vida foram pessoas de muita importância para mim e e contribuíram no que sou hoje como pessoa. Meu Pai Antônio Golfi Andreazi que em vida me ensinou o valor das coisas e a nunca abaixar a cabeça para o que eu achasse errado; Meu Amigo Maurício Rocha Guimarães que me ensinou grande parte do que eu sei sobre computação e sempre me incentivou a nunca parar de estudar. Esse trabalho só foi possível por vocês estarem na minha vida, mesmo que por um curto período.

Agradeço aos meus grandes professores do Instituto Federal de São Paulo (IFSP) por me fornecerem um base acadêmica muito forte, contribuindo imensamente para minha formação e evolução acadêmica E pessoal. Minha passagem pela Universidade de São Paulo (USP) teve diversos desafios, os quais eu ansiava desde que eu entrei no programa de pós-graduação. Por esse caminho passaram na minha vida pessoas maravilhosas que acreditaram no meu potencial e o exploraram para me fazer evoluir. Professores como Paulo Sérgio Lopes de Souza, Sarita Mazzini Bruschi me apresentaram novos conceitos os quais eu não tinha muito conhecimento, mas me ensinaram com muita paciência e dedicação.

Reservo um agradecimento especial ao meu orientador e mentor Julio Cezar Estrella, que durante todo o meu caminho, me auxiliou e fez o melhor para me ajudar e me conduzir para o caminho certo. Minha evolução durante esse mestrado só foi possível graças a você acreditar em meu potencial além de compartilhar toda a bagagem que você tem sobre computação comigo, que certamente causou um crescimento acadêmico e pessoal incrível.

Agradeço também a minha mãe, a senhora Neusa Aparecida Tomiatti, que em momento algum da minha vida me desencorajou a nada e sempre me incentivou a seguir os meus sonhos, os quais eram seus próprios sonhos. Por fim, agradeço as amigas que eu consegui de durante a minha jornada pela USP, em especial a pessoas que eu tenho como verdadeiros irmãos e me ensinaram o valor da amizade Welington da Silva Martins, Antônio Marcos Almeida Ferreira e Guilherme Martins. Outras pessoas que fizeram a diferença são meus amigos Allan Ribeiro Polachini, Matheus Saldanha, Nathalia Lopes, Lucélia Santos, Davi Conte e Letícia Venditti.

*“Você precisa aceitar o fato de não ser o melhor
e ter o desejo de superar qualquer desafio que enfrente.”
(Eiichiro oda)*

RESUMO

ANDREAZI, G. T. **Modelagem e desenvolvimento de uma arquitetura distribuída para o gerenciamento de recursos híbridos em nuvem privada**. 2020. 92 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

A computação em nuvem é um paradigma revolucionário que engloba alta conectividade, elasticidade, serviços mensuráveis e de fácil gestão. Esses atributos colaboram com o impulsionamento da tecnologia e parte de sua eficiência baseia-se em realizar a alocação de recursos virtuais e atribuição de serviços. Por conta da alta demanda por processamento de dados em aplicações Web nos últimos anos, houve uma maior oferta de poder computacional pelos provedores de serviço e em função disso pode-se atender um número variado de demandas e clientes. Por conta disso a modelagem de uma arquitetura em nuvem, deve compreender desde o levantamento de tarefas, especificação dos módulos e funções de cada componente da arquitetura. MoHRiPA é um protótipo que tem como objetivo o gerenciamento de recursos computacionais e atendimento de requisições em uma nuvem privada. Assim o intuito do projeto foi desenvolver uma arquitetura de nuvem distribuída e orientada a serviços que tenha funções híbridas com módulos independentes que realizam funções específicas como alocação de recursos, atribuição de serviços, monitoramento e mapeamento da infraestrutura. O protótipo apresenta um diferencial em relação a alguns trabalhos da literatura que não abordam adequadamente os mecanismos de avaliação de desempenho, sendo esses fundamentais para estabelecer os pontos fortes e fracos da arquitetura. O método utilizado para a pesquisa compreende a modelagem da arquitetura, a especificação dos componentes, assim como uma ênfase nos módulos que possuem propriedades híbridas por suportar o uso de diferentes ferramentas de virtualização e monitoramento. A solução da arquitetura apresentada, permite avaliar sob o aspecto de desempenho o que vários trabalhos na literatura pouco exploram: a avaliação do desempenho geral da solução com técnicas formais, como o planejamento fatorial completo que ofereceram análises mais acuradas acerca das funcionalidades e peculiaridades dos módulos da arquitetura. Em conclusão o trabalho modelou uma arquitetura em nuvem, a fim de desenvolver um protótipo com componentes que desempenha-se as funções da computação em nuvem, a atribuição de diferentes modelos de serviço e a avaliação desses componentes como um todo. Em relação ao resultados foram modelados experimentos avaliando a atribuição de máquinas virtuais e avaliando os algoritmos de mapeamento de recursos que apresentaram resultados satisfatórios.

Palavras-chave: Computação em nuvem, Provedor de Serviços, Gerenciamento de Recursos.

ABSTRACT

ANDREAZI, G. T. **Modeling and development of a distributed architecture for managing hybrid resources in a private cloud** . 2020. 92 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Cloud computing is a revolutionary paradigm that encompasses high connectivity, elasticity, measurable, and easily managed services. These attributes help boost technology and part of its efficiency is based on the allocation of virtual resources and the allocation of services. Due to the high demand for data processing in web applications in recent years, there has been a greater supply of computational power by service providers and as a result of this, it is possible to meet a varied number of demands and customers. Because of this, the modeling of cloud architecture must comprise from the survey of tasks, specification of the modules, and functions of each component of the architecture. We present MoHRiPA, a prototype that aims to manage computing resources and fulfill requests in a private cloud. Thus, the aim of the project was to develop a distributed and service-oriented cloud architecture that has hybrid functions with independent modules that perform specific functions such as resource allocation, service assignment, monitoring, and mapping of the infrastructure. The prototype presents a differential in relation to works in the literature that do not adequately address the mechanisms of performance evaluation, which are fundamental to establish the strengths and weaknesses of architecture. The method used for the research comprises the modeling of the architecture, the specification of the components, as well as an emphasis on modules that have hybrid properties for supporting the use of different virtualization and monitoring tools. The architecture solution presented, allows to evaluate under the aspect of performance what several works in the literature little explore: the evaluation of the general performance of the solution with formal techniques, such as the complete factorial planning that offered more accurate analyzes about the functionalities and peculiarities of the modules of architecture. In conclusion, the work modeled a cloud architecture in order to develop a prototype with components that perform the functions of cloud computing, the assignment of different service models, and the evaluation of these components as a whole. Concerning the results, experiments were modeled evaluating the allocation of virtual machines and evaluating the resource mapping algorithms that presented satisfactory results.

Keywords: Cloud Computing, Service Provider, Resource Management.

LISTA DE ILUSTRAÇÕES

Figura 1 – Modelo de referência conceitual de computação em nuvem adaptado da visão do NIST.	29
Figura 2 – Modelos de Implementações de Nuvens	31
Figura 3 – Modelos de serviços da Computação em Nuvem	32
Figura 4 – Representação das arquitetura de virtualização completa.	33
Figura 5 – Representação das arquitetura de paravirtualização.	33
Figura 6 – Representação das arquitetura de Contêiner.	34
Figura 7 – Representação de arquitetura de monitoramento Funcional	36
Figura 8 – Estratégia de obtenção de índice de desempenho.	37
Figura 9 – Frame das palavras usadas na String de Busca.	42
Figura 10 – Etapas do processo de revisão da literatura	43
Figura 11 – Modelo conceitual da arquitetura.	52
Figura 12 – Modelo conceitual da MoHRiPA.	53
Figura 13 – Fluxo de interação do cliente.	54
Figura 14 – Processos internos da arquitetura.	54
Figura 15 – Diagrama de sequência do módulo alocador.	55
Figura 16 – Diagrama de sequência do seletor de propriedades.	56
Figura 17 – Diagrama de sequência do intermediador consumindo recursos para alocação.	57
Figura 18 – Diagrama de sequência do intermediador consumindo aplicações de serviço.	58
Figura 19 – Diagrama de sequência do wrapper do monitorAPI.	60
Figura 20 – Diagrama de sequência do connector dstat no monitorAPI.	60
Figura 21 – Diagrama de sequência do <i>logserver plugin</i> método.	61
Figura 22 – Diagrama de sequência do <i>logserver plugin</i> simplificado para método.	62
Figura 23 – Diagrama de sequência do <i>log4j</i>	63
Figura 24 – Fluxo Completo da Arquitetura	66
Figura 25 – Representação da Configuração do Ambiente de Testes	68
Figura 26 – Tempo Médio de Resposta Exponencial.	74
Figura 27 – Tempo Médio de Resposta Uniforme.	74
Figura 28 – Distribuição de tempo do ListScheduling	75
Figura 29 – Distribuição de tempo do HashRefresh	76
Figura 30 – Distribuição de tempo da Latência.	76
Figura 31 – Gráficos de Interação entre Algoritmos de Mapeamento e Distribuições	77
Figura 32 – Tempo Médio de Resposta Operação ShutDown	77

Figura 33 – Tempo Médio de Resposta Operação StartUp	78
Figura 34 – Distribuição de tempo Uniforme das Operações ShutDown StartUp	79
Figura 35 – Distribuição de tempo Exponencial das Operações ShutDown StartUp . . .	80
Figura 36 – Tempo Médio de Resposta Operação Resume	80
Figura 37 – Tempo Médio de Resposta Operação Stop	81
Figura 38 – Distribuição de tempo Uniforme das Operações Resume Stop	81
Figura 39 – Distribuição de tempo Exponencial das Operações Resume Stop	82
Figura 40 – Gráficos de Interação entre operações de Virtualização e Distribuições . . .	82

LISTA DE TABELAS

Tabela 1 – Palavras Chaves de Pesquisa	42
Tabela 2 – Quantidade de estudos selecionados por fase	44
Tabela 3 – Abordagens dos principais autores em Gerenciamento de Recursos	45
Tabela 4 – Métricas necessárias para a API.	59
Tabela 5 – Tabela dos recursos computacionais utilizados.	64
Tabela 6 – Representação dos principais fatores e níveis potenciais do projeto.	69
Tabela 7 – Tabela de Fatores e Níveis do Cenário 1.	70
Tabela 8 – Tabela de Experimentos do Cenário 1	71
Tabela 9 – Tabela de Fatores e Níveis do Cenário 2	72
Tabela 10 – Tabela de Experimentos Cenário 2A	72
Tabela 11 – Tabela de Experimentos Cenário 2B	72
Tabela 12 – Tabela de Latência	75

LISTA DE ABREVIATURAS E SIGLAS

CSB	<i>Cloud Service Broker</i>
DRL	Deep Reinforcement Learning
GER	Gerenciado Elástico de Recursos
HPC	<i>High Performance Computing</i>
IaaS	Infrastructure as a Service
IFSP	Instituto Federal de São Paulo
IoT	<i>Internet of Things</i>
LaSDPC	Laboratório de Sistemas Distribuídos e Programação Concorrente
PaaS	Plataform as a Service
QoS	Quality of Service
SaaS	Software as a Service
SO	Sistema Operacional
USP	Universidade de São Paulo
VM	Virtual Machines

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Contextualização	24
1.2	Motivação e Objetivo	24
1.3	Estrutura	25
2	REFERENCIAL TEÓRICO	27
2.1	Considerações Iniciais	27
2.2	Computação em Nuvem	27
2.2.1	<i>Desafios da Computação em Nuvem</i>	28
2.2.2	<i>Arquitetura da Computação em Nuvem</i>	29
2.2.3	<i>Modelos de Computação em Nuvem</i>	30
2.3	Plataformas de Virtualização	32
2.4	Monitoramento de Recursos em Computação em Nuvem	34
2.5	Carga de Trabalho	37
2.6	Considerações Finais	38
3	TRABALHOS RELACIONADOS	41
3.1	Considerações Iniciais	41
3.2	Revisão da Literatura	41
3.3	Análise Crítica dos Trabalhos Relacionados	44
3.4	Considerações Finais	48
4	MOHRIPA	51
4.1	Considerações Iniciais	51
4.2	Modelo conceitual da Arquitetura	51
4.3	Protótipo da Arquitetura	53
4.3.1	<i>Provedor da Nuvem - Alocador</i>	55
4.3.2	<i>Auditor da Nuvem - GER</i>	56
4.3.3	<i>Intermediador da Nuvem - Broker</i>	57
4.3.4	<i>Componentes Auxiliares</i>	58
4.3.4.1	<i>ClusterLoader</i>	58
4.3.4.2	<i>MonitorAPI</i>	59
4.3.4.3	<i>LogServer</i>	61
4.4	Infraestrutura Computacional	63

4.5	Considerações Finais	65
5	AVALIAÇÃO EXPERIMENTAL	67
5.1	Considerações Iniciais	67
5.2	Configuração do Ambiente	67
5.3	Planejamento de Experimentos	68
5.3.1	<i>Cenário 1</i>	<i>70</i>
5.3.2	<i>Cenário 2</i>	<i>71</i>
5.4	Análise dos Resultados	73
5.4.1	<i>Resultados obtidos no Cenário 1</i>	<i>73</i>
5.4.2	<i>Resultados obtidos no Cenário 2</i>	<i>77</i>
5.5	Considerações Finais	83
6	CONCLUSÃO	85
6.1	Considerações Iniciais	85
6.2	Publicações	85
6.3	Contribuições	86
6.4	Trabalhos Futuros	86
6.5	Limitações e Dificuldades	87
6.6	Considerações Finais	87
	REFERÊNCIAS	89

INTRODUÇÃO

A evolução da ciência só é possível graças a troca de informações e toda a informação é crucial para tomada de decisões em áreas como ciências, administração, matemática, computação, etc. Porém, certos problemas demandam muito processamento, e gera gastos exorbitantes em termos de custo X tempo caso não haja uma infraestrutura adequada. A fim desses dados serem explorados e calculados de forma eficiente e conveniente, o *High Performance Computing* (HPC) compreende auxiliar por meio da computação de alta desempenho, a diminuição do tempo de processamento de conjuntos de dados.

Muitas empresas, instituições e órgãos governamentais tem muito zelo por seus dados, seja para pesquisas, armazenamento de documentos, protótipos, etc; porém há um custo para manter um estrutura de HPC. Os equipamentos necessários para montar uma infraestrutura de HPC, variam de máquinas para processamento e armazenamento de dados, controladores de redes, etc. Esses equipamentos necessitam de suporte e manutenção para seu correto funcionamento, contudo nem todos estão dispostos ou podem arcar com o custo desse tipo de solução. Entretanto, em setembro de 2011 foi definido por (MELL; GRANCE *et al.*, 2011) um conceito chamado computação em nuvem e suas possibilidades viriam a mudar o mundo que conhecemos em relação ao HPC.

Por meio de determinadas características seria possível distribuir recursos computacionais sob demanda com alta conectividade, tendo também a possibilidade de serem provisionados e gerenciados sem necessidade da interação com o provedor de serviços. Sendo assim, o HPC poderia ser disponibilizado para a venda se tornando um serviço volátil que pode ser contratado através de termos pré estabelecidos sendo garantias a deveres, assinado por ambas as partes (ALHAMAD; DILLON; CHANG, 2010).

Vista magnitude de possibilidades que a computação em nuvem oferece, empresas e instituições ganharam um alternativa em relação a infraestrutura de HPC, tendo a possibilidade de usar o serviço sobre demanda de grandes *players* como a *Amazon*, *Microsoft* e *Google*. Após

sua disponibilização, houve uma grande adesão, afinal é um serviço que foi estabelecido para diferentes tipos de usuários, agradando desde o usuário avançado que irá comprar processamento ao usuário básico que vai hospedar seus dados na nuvem.

1.1 Contextualização

A computação em nuvem desde sua origem foi impulsionada rapidamente na computação. A *infraestrutura as a service* (IaaS) tem sido uma das áreas que mais cresceram (YOUNGE *et al.*, 2010), isso dado ao conceito de poder vender a infraestrutura sob demanda. Para um administrador abre oportunidade de armazenamento de algum serviço ou conjunto de dados, considerando um custo apenas de contrato sobre o serviço, sem a necessidade de se preocupar com energia, manutenção de equipamentos, refrigeração, conectividade e etc.

Em relação a esse contexto, tais centros de computação em nuvem acarretam em um investimento alto tanto para as empresas quanto para os usuários. Logo, é necessário haver uma gestão responsável da infraestrutura, afinal o cliente deve receber um serviço satisfatório e buscar por formas mais eficientes, consciente e econômicas de usar os recursos que resultaram em inúmeras pesquisas.

O cliente, por realizar um investimento alto nesse tipo de serviço, sempre vai buscar o melhor custo benefício. O provedor de serviços, também vai buscar realizar um serviço da melhor forma possível para atender com maior qualidade. Atender melhor, é crucial para qualquer tipo de negócio, inclusive quando se tem um contrato da parte do provedor garantindo desde o provimento de serviços, alocação de recursos, mapeamento de recursos e adaptação de recursos.

Logo, é fundamental que a infraestrutura em nuvem tenha um gerenciamento de recursos eficiente para não quebrar contratos e não perder clientes. Em vista disso, esse projeto visa apresentar uma arquitetura para o gerenciamento de recursos que considera a alocação de recursos virtuais, provisionamento de recursos para o uso de um cliente, e o mapeamento de recursos disponíveis para o provisionamento e alocação. Assim como uma avaliação de desempenho acerca de alguns módulos da arquitetura.

1.2 Motivação e Objetivo

O intuito de elaborar uma arquitetura de nuvem privada que tivesse como foco o gerenciamento dos recursos se deu pelo seguinte questionamento: "não seria viável criar políticas financeiras e econômicas, sem ter um gerenciador de serviços?". Parte desse pressuposto poderia ser compensado com o uso de simuladores, mas a ideia de fazer um trabalho de prototipação, apresentaria resultados mais tangíveis diferente de simulações e com a devida orientação e recursos disponíveis.

Tendo em vista o gerenciamento de recursos, esse projeto teve que tratar desde a configuração de uma nuvem privada, configuração das Virtual Machines (VM), definindo IP das interfaces de rede para a comunicação entre os três *clusters* disponibilizados pelo Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC). Foi elaborado um ambiente de implantação totalmente controlado, no qual foram realizados os testes e experimentos do projeto.

Dentre os desafios deste projeto destacam-se:

- Elaborar e desenvolver uma arquitetura distribuída, no qual os módulos fossem desenvolvidos independentes, porém atuariam juntos para a solução final;
- Os módulos responsáveis pela alocação de recursos e monitoramento de recursos, devem se comunicar com diferentes ferramentas de virtualização e monitoramento;
- O provedor de recursos possui um módulo auditor de recursos que auxilia o mapeamento de recursos. O auditor deve ser capaz de calcular diferentes mapeamentos, independente do algoritmo usado na seleção de recursos.

Desta forma o projeto abordará as possibilidades que a arquitetura tem a oferecer, desde sua estrutura até suas aplicações. Em complemento, o trabalho realiza um mapeamento sistemático em relação ao gerenciamento de recursos, técnica de revisão bibliográfica que busca considerará mais critérios para a escolha de trabalhos que tenham mais relevância associada ao projeto.

1.3 Estrutura

O Trabalho está estruturado em 6 capítulos, os quais apresentam em síntese os seguintes assuntos:

- Na Seção 2 é apresentado o referencial teórico do trabalho, que engloba os conceitos técnicos respectivos a computação em nuvem, monitoramento e carga de trabalho;
- Na Seção 3 é descrito sobre os trabalhos relacionados. Para manter a seleção de forma mais criteriosa, foi utilizado o método de mapeamento sistemático que refina a qualidade dos estudos;
- Na Seção 4 é demonstrado como foi modelado o projeto, além de uma visão sobre o funcionamento da arquitetura como um todo;
- A Seção 5 expõe os resultados obtidos na pesquisa;
- Por fim, na Seção 6 são discutidas as conclusões que o projeto apresenta, além das publicações obtidas e trabalhos futuros.

REFERENCIAL TEÓRICO

2.1 Considerações Iniciais

Este capítulo apresenta o referencial teórico do presente projeto de pesquisa. Nas Seções 2.2 e 2.3 são discutidos os conceitos de Computação em Nuvem tais como os Desafios, Arquitetura base, Modelos de implantação, Modelos de Serviços e as Plataformas de Virtualização. Nas Seções 2.4 e 2.5 são discutidos os conceitos de monitoramento de recursos computacionais e carga de trabalho que agregam parte do presente trabalho e estão diretamente relacionados. Por fim encontram-se as considerações finais do capítulo explanando sobre todas as ressalvas importantes para o referencial teórico deste projeto.

2.2 Computação em Nuvem

A Computação em Nuvem propôs um novo paradigma baseado na prestação de serviços computacionais sob demanda. Uma das definições mais completas sobre computação em nuvem, foi divulgada pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST - *National Institute of Standards and Technology*), que diz o seguinte:

“A Computação em Nuvem é um modelo que permite o acesso à rede de forma ubíqua, conveniente e sob demanda, a um conjunto compartilhado de recursos computacionais configuráveis (tais como, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gestão ou interação com o provedor de serviços”. Este modelo de nuvem é composto de cinco características essenciais, três modelos de serviço e quatro modelos de implementação ([MELL; GRANCE et al., 2011](#))”

Dada a notoriedade dos *Cloud Service Broker* (CSB) devido a facilidade de utilizar recursos computacionais sob demanda via amplo acesso de rede por um custo adaptável ([ROGERS; CLIFF, 2012](#)), têm tornado as pesquisas em relação a administração e gerenciamento de recursos

virtuais, uma necessidade aparente afinal com a gama de serviços oferecidos e múltiplos tipos de usuários, há demandas variadas. Diversos estudos acadêmicos necessitando de processamento para calcular diversos problemas como escoamento de fluídos, predição de proteínas, sistemas meteorológicos, etc. Esse tipo de pesquisa demanda de poder computacional, e gera um custo enorme, mas sem isso não é possível avançar com a ciência pois certos cálculos computacionais demoram meses ou anos para serem calculados.

Alinhando a computação de alto desempenho com métodos de processamento paralelo, algoritmos genéticos e sistemas de otimização; tanto indústria quanto academia tem muito a ganhar. Porém, nem sempre é viável manter uma infraestrutura de grande porte (CHEN; SION, 2011) tendo em consideração a folha de pagamento de funcionários, uma estrutura redundante da rede elétrica, sistema de refrigeração (ZHOU *et al.*, 2019). Por consequência o investimento é elevado, uma vez que manter os equipamentos se torna outro desafio a ser considerado.

Visto a necessidade, empresas e instituições acadêmicas buscam nos CSB a possibilidade de utilizar seus recursos, apenas executando suas aplicações e assim facilitando suas pesquisas e desenvolvimento de produtos. Assim, além de economizar com gerenciamento e administração de infraestrutura, os CSB se tornaram um grande prestador de serviços e um trunfo para as comunidades acadêmicas e empresas.

2.2.1 Desafios da Computação em Nuvem

Os principais desafios em computação em nuvem cuja natureza são recorrentes, os quais sempre são tratados e abordados são escalabilidade (SHAW; SINGH, 2014), migração (KAPIL; PILLI; JOSHI, 2013), segurança (KHORSHED; ALI; WASIMI, 2012), Provisionamento de recursos (SINGH; CHANA, 2016) e gerenciamento de custos (AL-ROOMI *et al.*, 2013). Foi acrescentado na lista de desafios, a integração da computação em nuvem com os outros serviços como *Edge computing*, *Fog computing*, *Internet of Things* (IoT), *Blockchain*, etc. Dentre os conceitos de arquiteturas há seus próprios desafios, tornando a computação em nuvem uma base de processamento dentro de uma estrutura maior de arquiteturas que conversam e interagem para resolver diferentes problemas e desafios (DESHPANDE, 2020).

Em resumo, é importante sumarizar os principais desafios, pois alguns desafio impactam outros, tornando-se necessário quando tentar abordar um desafio se preocupar com outro desafio que esteja diretamente ligado. Em síntese alguns dos principais assuntos abordados são:

- **Escalabilidade:** O conceito escalável leva em conta a capacidade de uma infraestrutura eficiente de aumentar e diminuir os recursos sem perder desempenho, a contar o momento que a sua demanda de utilização aumente ou diminua.
- **Migração:** quando ocorre migração de dados, tem que se garantir a integridade e não pode ocorrer a perda de dados e a interceptação durante da comunicação desses dados. O

mesmo conceito se aplica a migração de máquinas virtuais como garantir que não vai ter perda de dados entre a transição de processamento de uma máquina para outra.

- **Provisionamento de recursos:** Para provisionar os recursos quando há uma infraestrutura com uma grande disponibilidade, os controladores e micro-serviços necessários para atribuir esses recursos para o cliente podem gerar gargalos na arquitetura, se não forem bem gerenciados pelo provedor podem ocasionar em atraso na entrega ou até mesmo em não entregar o recurso selecionado.
- **Gerenciamento de custos:** Tanto para o cliente quanto para o provedor é essencial a redução de custos. Para isso busca-se na literatura através de *green computing* ou políticas, reverter o alto consumo seja de gastos financeiros de um cliente e problemas administrativos de um provedor.

2.2.2 Arquitetura da Computação em Nuvem

A arquitetura em nuvem apresenta certos componentes que visam identificar como cada módulo deve ser estruturado em uma nuvem. Os termos: cliente, auditor, intermediador, provedor e transportador como demonstrado na Figura 1 são descritos da seguinte forma;

Figura 1 – Modelo de referência conceitual de computação em nuvem adaptado da visão do NIST.

Fonte: Liu *et al.* (2011).

- **Cliente da Nuvem (Cloud Consumer):** O cliente da nuvem é o usuário que usufrui da infraestrutura e seus serviços. Ele interage com o provedor da nuvem podendo acessar e gerenciar seus recursos conforme sua necessidade;

- **Provedor da Nuvem** (*Cloud Provider*): O provedor de nuvem possui a função de orquestração dos serviços, sendo assim responsável por organizar e coordenar a entrega de recursos virtuais e serviços ao cliente. Todavia também é responsável pelo gerenciamento da própria infraestrutura, assim como é responsável em prover segurança e privacidade aos clientes que consomem seus serviços;
- **Auditor da Nuvem** (*Cloud Auditor*): O auditor da nuvem tem um papel terciário dentro da arquitetura, realizando avaliações sobre os serviços de segurança, privacidade e desempenho de forma independente;
- **Intermediador da Nuvem** (*Cloud Broker*): O intermediador da nuvem é a entidade mais importante dentro da arquitetura, pois deve atuar como o gerente suas funções se caracterizam desde o acesso, prestação de serviços, intermediando as negociações do provedor de serviço com o cliente;
- **Transportador de Nuvem** (*Cloud Carrier*): O transportador da nuvem é responsável pela conectividade ao transporte de serviços do provedor. O transportador deve fornecer acesso aos clientes por meio da rede de comunicação.

2.2.3 Modelos de Computação em Nuvem

Os modelos previamente definidos e regulamentados por (MELL; GRANCE *et al.*, 2011) para computação em nuvem, são os modelos de implantação, os modelos de serviço, os quais são abordados nas próximas seções.

Modelos de implantação de Nuvens

Os modelos de implantação em nuvem são formas de implementar a nuvem em diferentes grupos de pessoas, sendo empresas, universidades e afins. O modelo de implantação é caracterizado em quatro diferentes implementações as quais foram previamente definidas pela NIST: a nuvem privada, nuvem pública, nuvem comunitária e por fim a nuvem híbrida. As implementações podem ser observadas na Figura 2.

Uma descrição sobre como esses tipos de nuvem, funcionam e interagem com o grupo de pessoas que a utilizam, são:

- **Nuvem privada** (*Private Cloud*): Essa implementação de nuvem é exclusivamente utilizada por uma organização particular, podendo ser terceirizada ou fazer parte da própria organização. Essa implementação é comumente usada quando a empresa vende um serviço para outras empresas, etc;
- **Nuvem pública** (*Public Cloud*): Essa implementação de nuvem está disponibilizada para o público geral. Podendo ser mantida por uma instituição acadêmica, organização governamental ou até mesmo pela união de ambas;

Figura 2 – Modelos de Implementações de Nuvens

Fonte: Buyya, Broberg e Goscinski (2010).

- **Nuvem Comunitária** (*Community Cloud*): Essa implementação compartilha o mesmo molde de uma nuvem pública, porém podendo estar disponível especificamente para uma comunidade com objetivos específicos.
- **Nuvem Híbrida** (*Hybrid Cloud*): Na implementação de nuvem híbrida a infraestrutura pode ser composta de duas ou mais implementações de nuvem, tais como pública, privada e comunitária. O principal detalhe é que essa implementação não exclui as entidades das implementações, pois apenas as interligam de forma padronizada permitindo a comunicação entre os modelos. Portanto, não é diretamente um tipo de nuvem, mas um conceito de interligar diferentes tipos de nuvem.

Modelos de Serviços

Os principais modelos de serviços são caracterizados por três entidades de prestação de serviço que são: **Software as a Service** (SaaS), **Plataform as a Service** (PaaS) e **Infrastructure as a Service** (IaaS). Esses modelos são formas alternativas de prover serviços aos diferentes usuários com diferentes propostas, em que cada uma delas seja direcionada a atender nichos específicos, como observado na Figura 3.

- **Software como um Serviço** (*Software as a Service*): Este modelo tem foco em atender o usuário final. Os serviços oferecidos neste modelo podem ser acessados via *browser Web*, devido a natureza dos serviços que são oferecidos. Como exemplo de conteúdo tem-se as redes sociais, serviços de *streaming* e jogos online;
- **Plataforma como um Serviço** (*Plataform as a Service*): Este modelo tem o foco de atender as demandas do desenvolvedor. Desta forma, o modelo se concentra em oferecer serviços voltados para o ambiente de desenvolvimento e que seja de fácil configuração e

Figura 3 – Modelos de serviços da Computação em Nuvem

Fonte: Buyya, Broberg e Goscinski (2010).

reconfiguração. Alguns dos principais serviços oferecidos atualmente são: *Google App Engine*¹, *Windows Azure*², *Facebook Developers*³, etc;

- **Infraestrutura como um Serviço** (*Infrastructure as a Service*): Este modelo tem o foco de oferecer a possibilidade de alugar a infraestrutura, implantando aplicações e desenvolvendo-as dentro do ambiente, etc. Estes serviços são oferecidos por empresas como AWS⁴, IBM⁵, etc.

2.3 Plataformas de Virtualização

A virtualização tem uma função essencial na Computação em Nuvem, pois é possível realizar a configuração e reconfiguração da arquitetura conforme a necessidade, sendo uma solução maleável tanto para a academia (ERCAN, 2010) quanto para a indústria (BUYYA; YEO; VENUGOPAL, 2008). Esse conceito traz a possibilidade de manter uma infraestrutura com escalabilidade e elasticidade em relação a configuração dos recursos, e principalmente na robustez que é diretamente uma consequência de ter infraestrutura otimizada para atender os serviços sob demanda (CHIEU *et al.*, 2009).

O principal componente de um virtualizador é o *Hipervisor* cuja função é gerenciar o sistema operacional hospedeiro variando seu funcionamento, dependendo do tipo de virtualização

¹ <<https://cloud.google.com/appengine/?hl=pt-br>>

² <<https://azure.microsoft.com/pt-br/>>

³ <<https://developers.facebook.com/>>

⁴ <<https://aws.amazon.com/>>

⁵ <<https://www.ibm.com/cloud/>>

que irá se utilizar. Há dois tipos de virtualizações apontados na literatura: a Virtualização Completa e a Para-virtualização.

Figura 4 – Representação das arquitetura de virtualização completa.

Fonte: Chowdhury e Boutaba (2010).

Virtualização Completa: Prevê o suporte total de uma virtualização desde o núcleo do ambiente do *hardware*. Com isso há a possibilidade de realizar uma virtualização sem necessidade de mudar o sistema operacional da máquina hospedeira, devido a interação que ocorre independentemente do *hardware* utilizado, como demonstrado na Figura 4;

Figura 5 – Representação das arquitetura de paravirtualização.

Fonte: Chowdhury e Boutaba (2010).

Paravirtualização: Diferente da virtualização completa, neste tipo de virtualização há necessidade da mudança do núcleo do ambiente de *hardware*. A consequência disso é a alteração de respectivos *drivers*, sendo necessário para cooperar com o funcionamento do *Hypervisor*, portanto mudando atributos do Sistema Operacional (SO) convidado, como exibido na Figura 5.

Todavia com os Contêineres surgiram novas possibilidades, pois essa pseudo-virtualização cria uma camada de abstração que permite a configuração de aplicações com mínimo esforço em relação a uma configuração padrão que envolve desde a instalação e configuração de um

serviço, ferramenta ou *framework*. Essa tecnologia beneficia principalmente a camada de PaaS, pois desenvolver com menos esforço em relação a configuração, auxilia o desenvolvimento ágil por facilitar o *deploy* e ser leve para o SO (MERKEL, 2014).

Figura 6 – Representação das arquitetura de Contêiner.

Fonte: Turnbull (2014).

Container: trabalha com um conceito totalmente diferente das outras virtualizações, através do *Docker Engine* que é uma camada de abstração do SO com as aplicações. É possível instanciar contêineres que contém aplicações pré instaladas apenas empilhando e subindo esse contêiner com o SO. Em relação aos outros tipos de virtualização o uso de um *hipervisor* é opcional, como representada na Figura 5.

Outro contexto muito importante dentro da virtualização são os conceitos de alocação vertical e horizontal, podendo ser usado para diferentes propósitos. A alocação vertical consiste em alocar mais recurso para a mesma máquina assim aumentando memória, processador e disco. Enquanto alocação horizontal liga novas máquinas virtuais conforme a demanda sendo útil em caso de balanceamento de carga e execução de códigos paralelos.

2.4 Monitoramento de Recursos em Computação em Nuvem

O monitoramento é fundamental em diversas áreas devido importância das informações coletadas em diferentes setores de aplicação como redes de sensores, sistemas computacionais, etc. Tendo esses dados coletados, os mesmos podem ser utilizados para diversos propósitos, tanto operacionais quanto gerenciais (HAZIR, 2015).

Provedores de recursos como Amazon, Azure e Google contemplam o monitoramento de recursos em suas plataformas de forma a auxiliar o cliente e com isso tem-se uma dimensão do quanto está sendo utilizado de processamento em uma máquina virtual, assim sabendo se o recurso computacional está sendo o necessário, ou se precisa de mais capacidade de processamento para determinada operação. Tais informações são cruciais quando se utiliza um

serviço mensurável como a computação em nuvem, pois a partir do momento que o serviço ou arquitetura contratada não seja suficiente para seu propósito, a empresa ou pessoa estará perdendo seu investimento (ELSHAER, 2013).

Tendo em vista um exemplo prático, num provedor de serviços que disponibiliza processamento para a comunidade científica da USP e parceiros, o Euler ⁶, oferece sua infraestrutura para um número de usuários que realizam seus experimentos em diversas áreas de aplicação tanto da computação, matemática e física. Os diferentes experimentos executados pelos pesquisadores utilizam técnicas distintas que dependendo da aplicação, se for simulações pode estar a utilizar mais CPU. Se a aplicação tratar com grafos ou matrizes certamente será memória. Trabalhar com imagens exige mais GPU a ser utilizada e portanto esse uso variado dos recursos torna essencial o monitoramento para a melhor administração da infraestrutura (ALIVERDI; NAENI; SALEHIPOUR, 2013).

Não é trivial trabalhar com múltiplas métricas de desempenho como se tem em computação em nuvem (LAURAS; MARQUES; GOURC, 2010). É importante monitorar esse uso de recursos para que o administrador responsável possa gerenciar melhor a infraestrutura. Toda a informação pode ser crucial para o funcionamento de uma solução, que vão desde o gargalo dos equipamentos, taxa de utilização dos recursos, etc. Observado isso o administrador pode tomar decisões como a atualização de equipamentos, detecção de máquinas que estejam com desempenho inferior apontando a possível necessidade de troca de equipamento por defeito ou desgaste do equipamento (ANDRZEJAK; KONDO; YI, 2010).

A granularidade das informações e a qualidade das mesmas desempenham outro importante fator dentro da tomada de decisões, e quando se realiza o monitoramento de recursos é crucial o estudo sobre as informações que irão ser coletadas e como serão utilizadas posteriormente (TSANGARIS *et al.*, 2009). A Amazon CloudWatch ⁷ é um serviço que surgiu no último semestre de 2018, a fim de auxiliar esse tipo de monitoramento e análise de dados, e demonstra como é crucial, pois outras empresas que vendem soluções baseadas em nuvem utilizam esse serviço da Amazon, como *Rackspace* ⁸ que utiliza em suas máquinas virtuais, *SendGrid* ⁹ que tem suporte nativo, *CloudPassage* ¹⁰ que utiliza para monitorar seus micro-serviços.

Existe outras ferramentas de monitoramento, visto que há muitas particularidades dentro de cada solução de computação em nuvem. Temos ferramentas que facilitam desde a coleta das informações de uma máquina, quanto filtrar e exibir por meio de um gráfico as informações, ou monitorar o estado de um processo. A Figura 7 demonstra uma arquitetura funcional de monitoramento apresentada em (HASSELMAYER; D'HEUREUSE, 2010).

No contexto geral há agentes que atuam colhendo informações monitoradas e enviando

⁶ <<http://www.cemeai.icmc.usp.br/Euler/index.html>>

⁷ <<https://aws.amazon.com/cloudwatch/>>

⁸ <<https://www.rackspace.com/>>

⁹ <<https://sendgrid.com/>>

¹⁰ <<https://www.cloudpassage.com/>>

Figura 7 – Representação de arquitetura de monitoramento Funcional

Fonte: Hasselmeyer e d'Heureuse (2010).

para uma base de dados. Por outro lado um sistema que gerencia essa base de dados e apresenta esses dados, pode gerenciar os agentes também. Essa arquitetura se assemelha ao funcionamento do *Zabbix*¹¹ e *Nagios*¹². Portanto nem sempre essas ferramentas vão suprir todas as demandas, pois não oferecem suporte para a solução de casos específicos. Assim é fundamental implementar um módulo que auxilie tais casos.

Todavia quando se lida com computação em nuvem deve-se considerar que há diferentes modelos e camadas. Sendo assim certos componentes da arquitetura possuem configurações heterogêneas e por conta disso o monitoramento demanda uma maior precisão. As métricas de desempenho simples como CPU e memória são insuficientes em cenários heterogêneos como a nuvem e tais métricas geralmente são usadas em ambientes homogêneos.

O trabalho de (BRANCO, 2004) propôs uma métrica composta de desempenho em que tal métrica considera heterogeneidade de configurações. A Figura 8 destaca a estratégia para obtenção de índice de desempenho; A princípio é necessário colher N recursos do sistema operacional desde CPU, memória, rede ou disco(I/O). Sendo assim é realizada uma normalização pela seguinte Fórmula 2.1:

$$ID = \sqrt{I_{CPU}^2 + I_{MEM}^2 + I_{NET}^2 + I_{DISK}^2} \quad (2.1)$$

Então a fim de utilizar uma métrica mais confiável no contexto da nuvem que englobasse n tipos de recursos e se adaptando a cenários heterogêneos como o ambiente em nuvem, adotamos o uso dessa métrica composta no trabalho, pois a possibilidades se encaixam justamente a necessidade na proposta.

¹¹ <<https://www.zabbix.com/>>

¹² <<https://www.nagios.com/>>

Figura 8 – Estratégia de obtenção de índice de desempenho.

Fonte: BRANCO (2004).

2.5 Carga de Trabalho

Dentro de um provedor de recursos, um dos fatores que mais impactam na infraestrutura é a falta de disponibilidade de recursos (CHACZKO *et al.*, 2011), a principal causa dessa falta de máquinas é justamente a alta demanda imposta sobre o provedor. No pressuposto que um servidor atende um número variado de demandas, é essencial a filtragem dessas demandas em relação a disponibilidade, afinal o conceito de disponibilidade aplicado quando se atende o cliente sem interrupções (TCHERNYKH *et al.*, 2019), não se pode atender o que não se tem a oferecer. A fim de estabelecer limites de testes com cargas de trabalho, buscamos avaliar a infraestrutura como um todo. Após os devidos testes que buscamos regular a infraestrutura como mecanismos de balanceamento de carga, replicação de provedores, políticas de tolerância a falha, etc.

A carga de trabalho dentro de um provedor pode ser aplicada via três vertentes, os testes de performance, carga e stress:

- **Teste de Desempenho:** Este teste visa medir o desempenho de equipamentos, desde conectividade, velocidade da memória ou processamento. Consiste basicamente na ideia de um *overclock* tendo a sua solução como objeto de teste de desempenho;

- **Teste de carga:** Este teste tem como foco estabelecer uma carga que gera muita demanda no sistema, sendo possível determinar o quanto a solução consegue processar;
- **Teste de stress:** Este teste é um modo de tentar violar o sistema. Sendo feito por uma sobrecarga de recursos e tentativas de derrubar o sistema.

Há dois tipos de carga, real e sintética tendo características divergentes que mudam completamente a tradução de seus resultados em relação a conceitos como:

- **Replicação:** A carga real não pode ser repetida, enquanto a carga sintética pode ser replicada quantas vezes forem necessárias;
- **Distribuição Adequada:** A carga real não segue uma distribuição adequada para um teste de carga, porém a carga sintética deve ser configurada para uma distribuição adequada;
- **Modificação de Carga:** A carga real não permite modificação de carga, todavia a carga sintética permite a modificação de carga, sendo esta uma de suas principais vantagens;
- **Arquivos de Trace:** A carga real não utiliza arquivos de trace, em contrapartida a carga sintética é fundamental usar um arquivo de trace.

Em relação à carga de trabalho real obtêm-se melhores resultados, afinal caso não haja uma distribuição adequada nem modificadores de carga, os resultados se tornam menos controláveis e mais condizentes com o mundo real. Entretanto cargas sintéticas são muito utilizadas, afinal modificadores de carga e a definição de uma distribuição adequada contribuem na reprodutividade de experimentos, tornando também a avaliação mais controlada, defeito é gerar inúmeros testes que só é possível devido a customização dos experimentos.

A carga de trabalho pode ser aplicada para medir o desempenho de um provedor e é importante para averiguar riscos da arquitetura. Todavia, outros aspectos como testes de cargas sintéticas possibilitam que desenvolvedores possam avaliar novas estratégias de balanceamento de carga, replicação de serviços/micro-serviços e políticas de gerenciamento de recursos. Estratégias assim podem contribuir com um melhor atendimento para os clientes, tornando o provedor mais capacitado a atender altas demandas de requisições ([ARUNARANI; MANJULA; SUGUMARAN, 2019](#)).

2.6 Considerações Finais

Este capítulo destacou os principais tópicos de computação em nuvem, além de demonstrar a arquitetura e seus componentes, assim como os modelos de serviços oferecidos e quais são os tipos de nuvem disponíveis. Vale relatar que o capítulo apresenta uma discussão sobre as plataformas de virtualização que é um dos conceitos fundamentais dentro de computação em nuvem,

assim como o monitoramento de recursos que pode impactar diretamente na infraestrutura como um todo. Por fim, destacamos um conceito que gera muita influência dentro de um provedor que é a carga de trabalho, sendo que a mesma pode ser usada para regular a infraestrutura definindo pontos de interesses para os teste de desempenho.

TRABALHOS RELACIONADOS

3.1 Considerações Iniciais

Este capítulo apresenta os trabalhos da literatura encontrados na literatura que se encaixam no escopo do objetivo da dissertação de mestrado. Na Seção 3.2 é apresentado a seleção dos estudos a qual metodologia utilizada foi o modelo de revisão sistemática buscando os estudos mais relacionados ao contexto da dissertação. A Seção 3.3 possui uma análise crítica dos trabalhos da literatura cujo objetivo foi identificar pontos de convergência. Por fim, na Seção 3.4 são apresentadas as considerações finais.

3.2 Revisão da Literatura

Segundo (BRERETON *et al.*, 2007), o método de revisão sistemática tem como objetivo identificar problemas que possam ser resolvidos em algum contexto dentro de um trabalho. Portanto, através desta revisão o objetivo é identificar e analisar: o gerenciamento de recursos virtuais em provedores de serviços em nuvem. Para o deferimento da pesquisa, investigou-se apontar alguns dos principais trabalhos relacionados ao tema proposto. Cujas pesquisas foram realizadas nas bases de dados: ACM Digital Library¹, IEEE Xplore² e Scopus³.

A fim de extrair as informações de forma eficiente, foi composta uma *String*, por meio da definição de operadores lógicos que auxiliam na recuperação de trabalhos relacionados, facilitando a busca de estudos primários mais pertinentes a pesquisa, e com o auxílio de palavras-chaves e seus principais sinônimos, representada na Tabela 1, procurou-se abranger todos os trabalhos que apresentavam em seu título, *abstract* ou palavras-chaves, uma ou mais das palavras-chaves definidas, pois diferentes termos sobre a mesma palavra são utilizados por todo o mundo.

¹ <<http://dl.acm.org>>

² <<http://ieeexplore.ieee.org>>

³ <<https://www.scopus.com>>

Tabela 1 – Palavras Chaves de Pesquisa

Palavras Chave	Sinônimos	Abreviações
Cloud Computing	Cloud, Cloud-Based, Cloud Infrastructure.	Iaas
Management	Manage, Administration, Administer, Policy.	-
Financial	Financier.	-
Economic	Economy, Economical.	-

A elaboração da *String* de Busca contou com o auxílio de especialistas na área de gerenciamento de recursos da Universidade de São Paulo (USP) e o resultado é apresentado na Figura 9:

Figura 9 – Frame das palavras usadas na String de Busca.

("Cloud Computing" OR "Cloud-Based" OR "Cloud Infrastructure") AND ("Management" OR "Manage" OR "Administration" OR "Administer" OR "Policy") AND ("Financial" OR "Financier" OR "Economy" OR "Economic" OR "Economical")

Fonte: Elaborada pelo autor.

Para a realização da seleção de trabalhos relacionados foi realizado uma análise por meio de conceito sistemático, a fim de obter os trabalhos mais relevantes, e cujo contexto melhor enquadra a dissertação. Essa análise consiste em realizar uma seleção por meio de critérios, os quais auxiliam na exclusão e inclusão dos trabalhos relacionados, e também auxiliam na etapa de filtragem. Foi feito uma seleção dos estudos que seja realmente relevante para esta pesquisa.

Seleção de Estudos Primários

A seleção de Estudos é importante para filtrar os resultados, visto que o número de estudos retornados pelas bases de busca é crescente, e todos os anos a pesquisa sobre determinado assunto evolui e nem tudo o que a busca retorna caso possua relevância. Portanto, há necessidade de aplicar critérios de exclusão e inclusão de trabalhos os quais atuam como limitadores de interesse durante a filtragem dos resultados. No processo de seleção contamos com três etapas de filtragem, em que cada uma delas a seleção dos estudos tem um afinamento dos trabalhos mais relevantes.

1. **Leitura de Títulos e Resumos:** Essa etapa consiste na leitura dos títulos e resumos. Caso alguns desses estudos primários tenham algo relevante com a proposta, o mesmo será encaminhado para a segunda etapa e caso ocorra o contrário o estudo será descartado;
2. **Leitura das seções de Introdução e Conclusão:** Essa etapa ocorre a partir dos resultados da filtragem anterior. Para comprovar a possível relevância, é realizada a leitura das seções de introdução e conclusão buscando entender o que foi feito e o que se resultou disso;
3. **Leitura completa dos trabalhos:** A etapa final consiste em obter as informações desejadas e então é realizado a leitura completa. Dessa forma, espera-se a extração das nuances

do estudo a qual a pesquisa estima. A partir desta seção espera-se que todos os estudos resultantes sejam relevantes.

Ao considerar as etapas que foram descritas, é possível construir um fluxo de atividades para a realização do mapeamento da literatura conforme é demonstrado na Figura 10.

Figura 10 – Etapas do processo de revisão da literatura

Fonte: [Petersen et al. \(2008\)](#).

Em relação aos critérios de inclusão e exclusão de estudos primários foram considerados os seguintes:

IC 1: Estudos que abordam o gerenciamento de recursos em Computação em Nuvem;

IC 2: Estudos que abordem aspectos de um modelo, método, técnica e abordagem econômica em Computação em Nuvem;

IC 3: Estudos que abordem aspectos de um modelo, método, técnica e abordagem financeira em Computação em Nuvem;

CrITÉrios de Exclusão

EC 1: Estudos que não estejam na língua Inglesa;

EC 2: Estudos que não abordam Computação em Nuvem;

EC 3: Relatórios técnicos, Documentos que estão disponíveis na forma de resumos ou apresentações e também revisões de literatura secundárias;

EC 4: Estudos que não possuam versões completas disponíveis;

EC 5: Estudos que não abordam a gerência em Computação em Nuvem;

EC 6: Artigos que não sejam focados nas áreas de Ciência e Engenharia da Computação;

EC 7: Resultados correspondentes aos capítulos de livros (Estudos secundários);

EC 8: Resultados correspondentes a literatura cinza (Estudos terciários).

3.3 Análise Crítica dos Trabalhos Relacionados

Durante a seleção, foi utilizado a ferramenta Parsifal⁴ e essa possibilitou a classificação dos estudos primários, e conforme a seleção sendo aceitos para a análise crítica ou rejeitados. A ferramenta também auxiliou em filtrar trabalhos duplicados, esses que foram publicados em diferente conferências, porém com o mesmo conteúdo.

Tabela 2 – Quantidade de estudos selecionados por fase

Filtros	Seleção de Estudo			Total	Duplicadas
	1º Fase	2º Fase	3º Fase		
Quantidade	174	47	21	174	6

A Tabela 2 demonstra a quantidade de artigos filtrados por cada fase de seleção de estudos e dentre essas fases alguns estudos foram descartados conforme os critérios estabelecidos. A *String* de busca junta com os critérios e objetivos, visava filtrar estudos voltados para o gerenciamento de recursos, mas como esse desafio é muito relevante dentro da computação em nuvem, há diversos conceitos dentro da mesma problemática, como estudos que exploram otimização de recursos e custos, *trade-off* de desempenho, consumo de energia, impacto do uso de diferentes tecnologias de alocação, etc.

Os trabalhos selecionados para a análise crítica tem como foco discutir os conceitos já utilizados na literatura em relação a nossa proposta, buscando arquiteturas de gerenciamento de recursos que possibilitasse a expansão e integração de novas funcionalidades. No mundo real resolver um problema totalmente engessado sem a possibilidade de expansão não é viável. Afinal a tecnologia está em evolução constante e o surgimento de novas tecnologias (*BlockChain*, *IoT*,

⁴ <<https://parsif.al/>>

Smart Contrats, etc) trazem seus desafios para dentro da computação em nuvem(KELECHI *et al.*, 2020).

A Tabela 3 visa sumarizar os principais conceitos abordados dentro dos trabalhos dos autores, os quais foram estudados e de forma criteriosa foi realizado uma análise crítica em função de analisar e concluir os principais aspectos das pesquisas realizadas.

Tabela 3 – Abordagens dos principais autores em Gerenciamento de Recursos

Autor	Abordagem				
	Gerenciamento de Recursos	Monitor de Recursos	Políticas Econômicas	Escalonador	Ambiente
(SHAHRAD <i>et al.</i> , 2017)	Alocação	Sim	Sim	Não especificado	Protótipo
(Liu <i>et al.</i> , 2017)	Alocação	Sim	Sim	Deep Reinforcement Learning	Simulador
(Mhatre; Shree; Sharma, 2017)	Serviço	Não	Não	FIFO, Round-Robin	Simulador
(Arif; Sahoo, 2018)	Alocação/Migração	Sim	Sim	List Scheduling	Simulador
(ATIEWI; ABUHUSSEIN; SALEH, 2018)	Alocação	Não	Não	Energy Consume Scheduling	Simulador
(GOHER <i>et al.</i> , 2018)	Alocação/Migração	Sim	Sim	Eleição de Líder	Simulador
(SAYADNAVARD; HAGHIGHAT; RAHMANI, 2019)	Migração	Sim	Não	Cadeia de Markov	Simulador
(Hassan; Chen; Liu, 2018)	Alocação	Não	Não	Deep Learning	Simulador
(Monteiro; Henriques, 2018)	Migração	Sim	Sim	Energy Consume Scheduling	Simulador

Em (SHAHRAD *et al.*, 2017) foi proposto um modelo de incentivo ao uso de recursos com desconto de custos pela demanda de processamento, sendo bonificação feita a partir do momento em que a infraestrutura está apta a aumentar a carga de trabalho, contanto que não tenha escassez de recursos. O modelo é deveras interessante pois o mesmo, além de gerar uma vantagem na diminuição de custos dos clientes, combate a ociosidade do provedor devido ao incentivo de uso da arquitetura. Todavia o modelo não retrata com clareza o modelo arquitetural, gerando certas dúvidas sobre o funcionamento como: de que forma seria realizado o controle de escassez de recursos? Visto que, o monitoramento é essencial nesse modelo, o mesmo é pouco mencionado e dúvidas acerca da granularidade das informações do monitoramento, que teoricamente podem impactar no modelo, não são devidamente descritas. Assim o controle de escassez de recursos demanda de informações que corroboram com esse aprofundamento do método.

(Liu *et al.*, 2017) propõe um *framework* hierárquico para a alocação de recursos virtuais e gerenciamento de energia em nuvem, por meio de um sistema de tomada de decisão com Deep Reinforcement Learning (DRL). Essa trabalho dispõe de uma política que visa minimizar o consumo de energia, enquanto mantém o desempenho de um forma que seja viável dentro da infraestrutura. O modelo de aprendizado consiste em mapear todos os servidores disponíveis e através do tempo dos *jobs* atuais, assim ajuda a determinar qual *job* será submetido a um servidor que tenha menos processos executando ou que o tempo desses processos seja curto, assim não gerando sobrecarga dentro do servidor. O reforço via *deep learning* acontece a cada interação que mantém um consistência maior da qualidade e velocidade dos mapeamentos. Utilizando a rede neural como um índice, é estabelecido o que será feito baseado nas interações anteriores. Todavia esse método não abordou um modelo arquitetural, apenas o conceito do módulo atuador de *deep learning* em relação ao provedor de recursos.

Os autores (Mhatre; Shree; Sharma, 2017) propõem um método de escalonamento por priorização de tarefas de forma paralela, utilizando o conceito de mestre/escravo. O algoritmo proposto melhora o desempenho e utilização de recursos, além de reduzir o tempo de execução durante o balanceamento de carga. Por um lado o mestre recebe todas as tarefas pelo gerenciador de tarefas e as distribuem para as máquinas escravas que executam esses trabalhos em suas VM's. O escalonamento das tarefas pelo nó mestre pode utilizar dois algoritmos de escalonamento: Round-Robin e FiFo. Considerando apenas a entrega das tarefas e não o estados dos nós escravos, essa abordagem abre a possibilidade de sobrecarga de recursos. Afinal num ambiente real apenas considerar a distribuição de tarefas não é o ideal, visto que também é importante considerar a carga de trabalho imposta para as máquinas de processamento (nós escravos).

Em (Arif; Sahoo, 2018) o desafio abordado foi o *trade-off* de balanceamento de carga pelos provedores em relação ao Quality of Service (QoS). Por meio das políticas de alocação/migração de recursos, a prioridade do trabalho é realizar o balanceamento de carga sem comprometer operações em andamento. Tendo dois segmentos de gerenciamento. O primeiro segmento trabalha com a ordenação das tarefas, baseado no tamanho da tarefa e o limite que a infraestrutura pode fornecer sem perder o desempenho. O segundo segmento aplica a utilização de *hosts* físicos em demanda para realizar a migração, e diminuir a carga de trabalho para outras máquinas, pois assim um *host* físico sobrecarregado, migra seu processamento para outras vm's para diminuir sua carga. Ambas as metodologias mostram resultados contundentes e precisos, porém o ambiente carece de informações mais técnicas como o tempo de atualização do mapeamento de recursos, que é realizado pelo o algoritmo *ListScheduling* que pode ser ordenado com recursos heterogêneos e atualizado conforme a necessidade.

Em (ATIEWI; ABUHUSSEIN; SALEH, 2018) foi realizado um estudo empírico sobre o impacto da virtualização no consumo de energia e para isso foi utilizado dois algoritmos de escalonamento de tarefas voltados para controle de consumo de energia. Além disso o trabalho definiu 3 métricas para avaliar a eficiência dos algoritmos; O consumo total de energia, a carga de trabalho imposta aos provedores e a carga de trabalho imposta às máquinas virtuais. Apenas o escalonamento de tarefas nem sempre é o necessário para diminuir a carga de trabalho, seja essa carga imposta ao provedor ou a uma máquina virtual, visto que a técnica do estudo se concentrou em escalonar a partir do consumo de energia sem considerar o consumo de recursos computacionais, apenas o consumo de energia de cada máquina, não dando foco ao monitoramento do desempenho dos equipamentos. A proposta não se mostrou muito precisa em assumir apenas as métricas de consumo de energia, métricas de desempenho também são essenciais mesmo em um ambiente controlado. Afinal em um ambiente real o gerenciamento de recursos nunca seria inteiramente baseado na parte elétrica, mesmo com um ambiente totalmente redundante e equipamentos de ponta, as informações de desempenho dos equipamentos são muito confiáveis, pois fornece uma informação com maior granularidade dos *hosts*, assim os algoritmos de escalonamento pode demonstrar maior eficácia, afinal caso seja feita uma alocação errada pode impactar em todas as futuras alocações.

Em (GOHER *et al.*, 2018) propôs a implementação de um modelo arquitetural, o qual realiza a expansão de alocação de recursos em função da demanda via trocas automatizadas entre provedores. O principal fator do modelo é a colaboração entre os provedores, os quais faziam transações sobre recursos ociosos entre si, e caso fosse determinado um provedor com menos carga de trabalho, o mesmo era selecionado e condicionado a atender novas demandas, assim a infraestrutura permitia um balanceamento de carga entre os provedores. O processo de tomada de decisão era baseado em métricas do tempo de resposta, que foram pontuadas após uma eleição dos provedores considerando desempenho, elasticidade, disponibilidade e cumprimento do SLA. Contudo o modelo apresenta um sistema de tarifação baseado na transação de recursos entre provedores, visando variáveis como urgência do recurso, tempo de transação e consolidação dos recursos. O modelo se provou excelente durante o escalonamento de tarefas, além de aferir uma tarifa mais barata em relação aos provedores cobram para a transação de recursos. Como trabalho futuro os autores levantaram o seguinte questionamento, caso os provedores não tenham provedores candidatos para serem eleitos? como a arquitetura se comportará? Tais apontamentos ajudam a refletir sobre o que poderia vir a ferir o SLA da arquitetura, consequentemente caso essa predição de quando os provedores vão ter jobs finalizados, seja previamente realizada os mesmos poderiam ser pré-alocado em uma fila, assim não paralisando a arquitetura e aumentando a confiabilidade em relação aos recursos disponíveis.

Os autores (SAYADNAVARD; HAGHIGHAT; RAHMANI, 2019) estabelecem em seu trabalho a redução de manter máquinas físicas ligadas, pois provedores deixam ligadas às máquinas físicas e vão alocando vm's conforme a demanda. Entretanto como a demanda é elástica, pode ocorrer em certos períodos do dia, a diminuição demanda, deixando recursos ociosos, assim esse trabalho por meio da cadeia de Markov busca otimizar o uso da infraestrutura determinando por meio da utilização de CPU quais máquinas estariam hábeis a fazer uma migração de máquinas virtuais, a fim de diminuir o número de máquinas físicas ligadas simultaneamente. O ambiente utilizado foi o *CloudSim*⁵ e o conceito por trás da proposta apresenta algumas questões críticas. Como apenas a utilização de CPU como métrica de decisão, está não pode ser a única métrica necessária para determinar quais máquinas estejam consumindo mais ou menos recursos, já que há diferentes tipos de aplicações e algumas consomem mais memória RAM, do que CPU. Porém quando é considerando uma métrica composta como o índice de desempenho proposto por (ESTRELLA *et al.*, 2011), assim teria-se uma métrica mais volátil para a otimização de qualquer tipo de escalonamento considerando desempenho. Outro ponto é sobre desligar máquinas físicas no ambiente de simulação, pois isso não é considerado viável em soluções reais como *datacenter/cloud*, pois são projetadas para ficarem em operação 24/7 do tempo. Apenas se desliga uma máquina caso seja extremamente necessário, vista uma manutenção ou troca de componentes, certas soluções utilizam imagens virtuais que são descarregadas em seu disco volátil. Toda vez que são ligadas, portanto caso fique desligando e ligando essas máquinas podem ocorrer de se perder muito tempo no *boot*, além da latência gerada na rede, pois algumas soluções

⁵ <<http://www.cloudbus.org/cloudsim/>>

descarregam essas imagens via rede.

Em (Hassan; Chen; Liu, 2018) propõe um *framework* para a alocação de recursos automatizados, via escalonador de tarefas baseado em *Deep Learning*. O escalonador trabalhava em função dos *feedbacks* dos SLA's e dessa forma sempre que um novo *feedback* do SLA é adicionado, se tem uma atualização de predição para o módulo responsável pelo escalonamento de tarefas. Por função da forma que o escalonamento é realizado, a arquitetura não dá foco sobre as informações de monitoramento, ficando presa apenas ao *feedbacks* do SLA. A solução atende as requisições de forma eficiente, a arquitetura está limitada a sua capacidade total, mas essa capacidade fica determinada pelo *feedback* do SLA, que caso seja insuficiente pode gerar alocações não muito acertadas. Para a abordagem funcionar sem esse empecilho, a solução deveria contar com um módulo de reconfiguração da infraestrutura, a fim de caso as configurações não estejam disponíveis, haver um reconfiguração dos componentes e assim recommear a escalonar as tarefas. A rede neural fica dependente dos *feedbacks*, portanto uma solução auxiliar pode resolver esse problema.

Os autores (Monteiro; Henriques, 2018) apresentam uma abordagem voltada à modelagem e gerenciamento de máquinas virtuais, assim controlam o provimento de energia aos *web service cluster*. A abordagem utiliza um sistema linear de alta escalabilidade, que atende com muita qualidade e o modelo prioriza o *trade-off* de consumo de energia e desempenho. O modelo sempre busca reduzir o consumo de energia sem diminuir a qualidade do serviço, dessa forma quando é identificado que um *host* físico esteja consumindo muita energia, ocorre uma migração dos dados de processamento para uma máquina virtual pausada, e assim que a migração termina aquela nova máquina virtual reassume o processamento. Enquanto que o *host* físico que está sobrecarregado após a migração diminui seu consumo de energia, estabilizando a infraestrutura. O maior ponto de questionamento é acerca do tempo das operações que não fora devidamente exploradas pois uma máquina pausada demora um tempo considerável para reassumir os processos que estavam na fila e saberia qual seria o gargalo do modelo? as operações de migração ou mudança do estado das máquinas e retomada de tarefas.

3.4 Considerações Finais

Nesse capítulo foram abordados os conceitos relacionados ao mapeamento sistemático, demonstrando como foi realizado a técnica de filtragem dos trabalhos relacionados. Na fase de análise crítica dos trabalhos relacionados foi notado certas constantes que se mostram fundamentais as quais são discutidas a seguir:

- **Gerenciamento de Recursos:** Em relação ao gerenciamento de recursos, vale ressaltar que temos duas forma de abordar: alocação e migração. Alocação de recursos vai desde o provimento de serviços ou provimento de máquinas virtuais, enquanto a migração é uma função fundamental para o balanceamento de carga. No conceito geral a migração é

uma tomada para balancear uma infraestrutura que não está sendo bem gerida, seja por uma carga muito alta ou simplesmente por pura manutenção; Trabalhos abordaram esse conceito de forma variada, pois o gerenciamento de recursos começa na requisição e uma infraestrutura deve ser projetada para atender demandas variadas;

- **Monitoramento dos Recursos:** Durante a leitura sobre as obras da literatura, certos autores consideram apenas uma métrica de monitoramento, como o consumo de CPU ou energia. Decisões tomadas apenas em função de aplicações que estejam executando operações *cpu-bound*, tornam certas soluções ineficazes, considerando que há muitas aplicações com alto consumo de memória, ou alto consumo de memória e CPU. Métricas compostas agregam mais informações como rede, disco, etc; e assim tendem a fornecer informações gerenciais mais ricas, possibilitando a administração dos recursos da infraestrutura de forma mais profissional;
- **Políticas Econômicas:** As políticas econômicas tem o princípio de gerar uma redução de ociosidade e consumo de recursos. Todavia redução de ociosidade consiste em diversas técnicas, como atribuir tarefas de forma distribuída, e assim evitar a sobrecarga dos *hosts*, pois esse controle é fundamental para determinar que está sendo executado e onde está sendo executado. Outro exemplo é acerca das cargas, caso uma aplicação inicie sua tarefa carregando dados, sua carga irá aumentar com o processamento desses dados. Assim para combater a ociosidade requer-se uma granularidade das informações dos *jobs* executados e das informações coletadas pelo monitor de recursos. Então um dos principais pontos a serem analisados, para estabelecer uma política econômica é o consumo de utilização de recursos.
- **Estratégia de Escalonamento:** O escalonador de recursos é o fator mais crucial, durante a atribuição de tarefas a infraestrutura, pois caso não seja realizado um escalonamento das tarefas adequado, provavelmente a arquitetura não irá conseguir responder a todas às tarefas ou a infraestrutura estará sobrecarregada. Nos piores casos pode ocorrer de violar contratos de serviço com o usuário, ou diminuir a capacidade de processamento, gerar gargalos na rede, etc. Inúmeras técnicas utilizam algoritmos que fornecem a melhor informação para alocar da melhor forma possível. Em alguns casos que a migração é a etapa posterior a alocação, e é necessário migrar processamento de um lugar para outro.
- **Ambiente:** Por fim o fator crucial, é o ambiente de testes, que pode ser simulado, a qual se tem controle total sobre os componente necessários para o experimento, e equívocos podem ser evitados, caso tudo não seja devidamente configurado o mais próximo de um ambiente real.

A revisão da literatura permitiu identificar as principais abordagens, modelos e técnicas que devem ser aplicadas na concepção do presente projeto, além dos detalhes sobre quais

são os fatores que influenciam o gerenciamento de uma infraestrutura em nuvem. Os fatores determinam o motivo de certos conceitos serem amplamente reconhecidos e utilizados, os quais são fundamentais para a construção da solução para a execução dos testes e a obtenção das conclusões sobre o quanto determinada solução é viável para o adequado gerenciamento de recursos computacionais no contexto da nuvem privada.

MOHRIPA

4.1 Considerações Iniciais

Este capítulo apresenta a arquitetura MoHRiPA, como foi modelada e então desenvolvida. Na Seção 4.2 é discutido sobre o modelo conceitual da arquitetura no qual são expostos os principais componentes e como estes interagem entre si. Na seção 4.3 é discutido sobre o desenvolvimento do protótipo da arquitetura em que serão apresentados os detalhes da solução. Na seção 4.4 são apresentadas informações em relação a infraestrutura utilizada para os experimentos assim como as principais ferramentas utilizadas na execução e análise dos dados. Por fim, na seção 4.5 são apresentadas as considerações finais do Capítulo ao fazer uma síntese do que fora discutido com os principais pontos a serem considerados sobre a arquitetura.

4.2 Modelo conceitual da Arquitetura

A modelagem de uma arquitetura voltada a nuvem, iniciou-se a partir da definição dos núcleos e as funções de cada componente. Estas foram adaptadas da arquitetura **WSARCH** de (ESTRELLA *et al.*, 2011), uma arquitetura orientada a serviços com suporte à QoS. Modelar uma arquitetura em nuvem orientada a serviços possibilita controlar e gerenciar com mais eficiência serviços que são oferecidos aos clientes, porém, isso demanda uma estrutura robusta de como será realizada a atribuição de tarefas uma vez que tal aspecto impacta diretamente na qualidade do serviço.

Durante a elaboração da arquitetura foi definido como deveria ser a interação dos módulos e que estes seriam independentes. Isso porque partes individuais teriam funções bem definidas como alocação ou monitoramento. Logo, para facilitar a manutenção dos códigos, a implantação de novas tecnologias que englobam ferramentas de serviço e algoritmos de mapeamento, todos os módulos serão implementados e modelados de forma distribuída, pois estes ao serem desacoplados funcionam ao cumprir suas tarefas específicas.

Figura 11 – Modelo conceitual da arquitetura.

Fonte: Elaborada pelo autor.

Na Figura 11 é exibida a arquitetura conceitual composta por 7 módulos, os quais possuem funções específicas cujas características são:

- **Gerenciado Elástico de Recursos:** Essa entidade controla a audição dos recursos e realiza o escalonamento em função das métricas de desempenho coletadas pelo monitor de recursos. Outro ponto relevante é que essa audição de recursos é realizada com base em dois algoritmos: *ListScheduling* e *HashRefresh*; que buscam classificar as máquinas físicas e virtuais a serem atribuídas para o provedor;
- **Monitor de Recursos:** Também é definido como MonitorAPI, cujo papel é a realização do monitoramento de componentes como CPU, memória, disco e rede. Este módulo deve coletar as métricas de desempenho, pois o auditor de recursos(GER) precisa das métricas dos *hosts* disponíveis no provedor para calcular os índices de desempenho (BRANCO, 2004). Desta forma o GER gera um rank com os melhores *hosts* dentro do provedor, com o propósito de melhor atender a requisição dos clientes;
- **Intermediador da Nuvem:** Intitulado como *Broker* tem como função receber as requisições e responder essas requisições com base no mapeamento do GER. Essa entidade tem alguns auxiliares que são responsáveis pela alocação de recursos (atribuição de máquinas virtuais) e provimento de recursos (atribuição de serviços).
 - **Alocador:** O módulo alocador funciona de forma híbrida independente de ferramenta de virtualização, o qual permite a execução de comandos básicos como ligar, desligar, pausar e retornar.
 - **Transportador:** O módulo transportador de serviços é responsável pela disponibilização do link de algum serviço solicitado pelo usuário;

- **ClusterLoad:** Esse módulo comporta as aplicações de serviços, as quais são oferecidas ao cliente, que pode optar por 3 tipos de aplicações *CPU-Bound*, *Memory-Bound* e *I/O-Bound*;
- **Servidor de Log:** O servidor de *log* é crucial neste contexto, pois toda a informação necessária para identificar possíveis erros, gargalos e informações do funcionamento da arquitetura ficam nele armazenados, com informações desde *DEBUG*, *TRACE*, *FATAL*, *ERROR*, etc.

4.3 Protótipo da Arquitetura

A arquitetura é dividida em processos que tem funcionalidades como monitoramento, virtualização, transporte de serviços, escalonador de requisições, módulo de decisão, lista de recursos e atendimento de requisições. Esses processos desempenham o funcionamento completo da arquitetura. Entre os atores que interagem dentro do protótipo, tem-se a arquitetura e o cliente. A Figura 12, apresenta fluxos distintos com diferentes abordagens de interação do cliente e processos internos da arquitetura:

Figura 12 – Modelo conceitual da MoHRiPA.

Fonte: Elaborada pelo autor.

O cliente é uma chave fundamental para ativação dos serviços da arquitetura, como demonstrado na Figura 13. O cliente inicia com uma requisição e assim que o *broker* recebe essa requisição, cabe ao escalonador detectá-la e respondê-la. Desta forma o cliente poderá ter acesso aos recursos virtuais ou serviço solicitado. O cliente recebe esse recurso de um transportador

que tem a missão de atribuir a resposta da requisição. Em razão disso, é atribuído acesso a uma VM ou serviço implantado no provedor de serviços.

Figura 13 – Fluxo de interação do cliente.

Fonte: Elaborada pelo autor.

Enquanto o cliente realiza uma requisição, a arquitetura executa alguns processos internos com o intuito de melhor atender ao cliente. A Figura 14 demonstra esses processos internos, tais como: o monitoramento dos provedores de serviço, audição de desempenho e decisão dos melhores candidatos (*hosts*) à alocação (VM ou serviço). Isso ocorre em *background* para haver a atualização do mapa de recursos, o que possibilita atribuir ao transportador o que será instanciado.

Figura 14 – Processos internos da arquitetura.

Fonte: Elaborada pelo autor.

A considerar os dois fluxos, a ação de cada módulo independente é fundamental para o funcionamento da arquitetura, uma vez que, toda a solução foi implementada para realizar operações que desempenham o papel da prestação de serviços e alocação de recursos virtuais. Esses dois fluxos tem funções de cliente-servidor, uma vez que, o cliente a função de realizar as requisições e a arquitetura com a função servidor de atender às requisições por meio dos seus diferentes módulos que desempenham essas tarefas. Nas próximas subseções serão detalhados todos os componentes de maneira isolada e os processos internos da arquitetura.

4.3.1 Provedor da Nuvem - Alocador

A virtualização é um dos principais componentes na computação em nuvem e sua variedade de usos para configuração e reconfiguração de ambiente são pontos muito importantes para uma infraestrutura escalável. Visto que, os virtualizadores fazem uso de um *hipervisor* para hospedar os sistemas operacionais convidados, alguns comandos devem ser executados no *host* físico (que hospeda as vms). Portanto, a demanda para a solução consiste em realizar uma conexão à máquina física, para então processar os comandos como por exemplo: ligar ou desligar uma máquina virtual.

O consumo de recursos utilizados para essa operação fim a fim seria muito custosa e demorada, afinal toda requisição seria enfileirada, gera um grande gargalo no atendimento das requisições o que compromete a infraestrutura. Para mitigar esse problema foi utilizado uma *thread pool* para auxiliar no atendimento, pois a cada requisição o *thread pool* aloca uma *thread* independente e responsável pela operação de conexão e execução do comando do virtualizador. Nesse caso, esse micro-serviço ao ser escalável, apenas amplifica a capacidade de atendimento da arquitetura.

Figura 15 – Diagrama de sequência do módulo alocador.

Fonte: Elaborada pelo autor.

A Figura 15 demonstra o funcionamento do módulo alocador e como citado o *thread pool* cria *threads* para executar as operações de forma dinâmica, ao criar *threads* conforme a necessidade. Quando a *thread* é instanciada, a mesma deve executar duas tarefas: a primeira pelo método *SSHDispatcher* que realiza a expedição daquela *thread* para a máquina física via conexão SSH. Após conectado a uma máquina, por meio do terminal *shell* interno, a *thread* realiza a segunda tarefa que consiste em executar o método *CommandDispatcher* responsável pela execução do comando de virtualização via terminal. Assim, a *thread* encerra seu ciclo de

vida ao avisar o alocador, o que encerra o ciclo com a finalização da conexão SSH e do canal de execução.

Figura 16 – Diagrama de sequência do seletor de propriedades.

Fonte: Elaborada pelo autor.

Para aplicação compreender um contexto híbrido (Figura 16) foram criadas funções das operações de vários virtualizadores, como: *Start*, *Shutdown*, *Pause*, *Resume*, *Destroy*; as quais funções são definidas por um atributo, que seria o nome da ferramenta de virtualização. Caso o atributo definido seja KVM¹ como foi o caso, esse atributo será utilizado para ler os comandos mapeados para o "KVM" em um arquivo de propriedades. Esse arquivo é expansível assim como os métodos empregados, e é necessário apenas inserir um comando de outro virtualizador. Quando um atributo é definido, pode ser utilizado o método daquele atributo e essa execução é feita pelo *broker* ao alocar uma máquina para o cliente.

4.3.2 Auditor da Nuvem - GER

O auditor de recursos realiza as avaliações e auditorias em relação aos recursos da infraestrutura e tais avaliações demandam dados de desempenho dos hosts pertencentes a infraestrutura. Esses dados são filtrados e selecionados por meio de algoritmos de seleção e ranqueamento.

Para a execução das etapas posteriores de alocação é necessário o pool de recursos o qual é formado pelas audições de desempenho realizadas. Dessa forma os algoritmos atualizam o pool com as informações das métricas dos *host*. Com essas métricas o pool de recursos pode ser consumido e assim as requisições podem ser atendidas adequadamente.

¹ <<https://www.linux-kvm.org/>>

Foram modelados dois algoritmos para determinar o rank na arquitetura: *ListScheduling* que é um algoritmo de ordenação em lista para ambientes heterogêneos (Arabnejad; Barbosa, 2014), o qual ordena a lista por um parâmetro específico. O *ListScheduling* tem um tempo de atualização que foi definido em 10 segundos. O *HashRefresh* o segundo algoritmo *ad-hoc* autoral é baseado em tabela hash, com desempenho superior, e que realiza atualização constante, uma vez que, atualiza o rank por requisição do cliente.

4.3.3 Intermediador da Nuvem - Broker

O intermediador da nuvem é o componente da arquitetura responsável por realizar as funções de gerência e atendimento. O *broker* contempla dois conceitos de atendimento:: o primeiro é a alocação de recursos virtuais(IaaS) e o segundo é a atribuição para aplicações(SaaS). Assim, dessa forma, dentro da camada de orquestração de serviços há maneiras variadas de lidar com as requisições.

A Figura 17 ilustra como o intermediador lida com requisições de alocação de recurso. Para toda a requisição que chega ao *Broker* é realizada uma audição de recursos para selecionar o melhor recurso em determinada requisição e no momento em que este recurso é selecionado o mesmo é repassado para o módulo de alocação, o qual realiza a virtualização daquele recurso para o cliente. Por fim, o transportador repassa instruções de acesso da VM ao cliente.

Figura 17 – Diagrama de sequência do intermediador consumindo recursos para alocação.

Fonte: Elaborada pelo autor.

A Figura 18 apresenta a forma com que o intermediador realiza o atendimento de serviços. No momento em que uma requisição é enviada ao *Broker* a mesma é submetida ao auditor de recursos que seleciona os melhores recursos em função do tipo de aplicação que o cliente deseja. Dessa forma, a audição de recursos busca encontrar a máquina virtual com recursos menos

ociosos em relação ao processamento desejado. Assim, será entregue pelo transportador uma VM com configurações mais adequadas para uma aplicação que consome mais memória ou então tenha muito processamento.

Figura 18 – Diagrama de sequência do intermediador consumindo aplicações de serviço.

Fonte: Elaborada pelo autor.

O intermediador da nuvem desse projeto atua como um prestador de serviço, pois realiza atribuição de recursos e aplicação. Esse orquestrador realiza apenas suas operações fundamentais e não se preocupa com a audição de recursos, provisionamento de recursos ou monitoramento de desempenho. A arquitetura foi projetada justamente para alcançar desempenho em relação a entrega dos serviços pertencentes aos modelos de *IaaS* e *SaaS*.

4.3.4 Componentes Auxiliares

Os componentes auxiliares se refere às soluções que complementam uma arquitetura em nuvem. Ainda que algumas de suas funções possam ser opcionais, é importante destacar que seu uso pode ser útil para a administração de uma nuvem. Tais componentes proveem desde informações gerenciais da infraestrutura, micro serviços que visam o monitoramento da qualidade do serviço e também ajuda na manutenção do sistema e infraestrutura. Três desses componentes auxiliares considerados e desenvolvidos no projeto são: *ClusterLoader*, *MonitorAPI* e *LogServer*.

4.3.4.1 ClusterLoader

O *ClusterLoader* é uma aplicação que é encontrada em todas as máquinas virtuais de serviço e essa aplicação usufrui de 3 tipos de aplicações com diferentes tipos de processamento, sendo *CPU-bound*, *Memory-bound* e *I/O-bound*. O acesso a essas aplicações é determinada pelo

cliente na requisição realizada ao intermediador da nuvem. O cliente define o tipo de aplicação que necessita e essa aplicação aloca o serviço para o cliente e retorna o resultado.

As aplicações foram definidas ao considerar algum nível de complexidade para assim ser possível atribuir operações que geram alguma sobrecarga na arquitetura. Para a aplicação *CPU-bound* foi definido uma sequência de Fibonacci, enquanto na aplicação *Memory-bound* é realizada uma multiplicação de matrizes. Por fim, a aplicação *I/O-bound* realiza uma ordenação de sequência de *Fibonacci* por meio do algoritmo de ordenação *insertion sort* feito propositalmente para gerar muita operação de entrada e saída.

4.3.4.2 MonitorAPI

O monitoramento de recursos computacionais fornece métricas de desempenho fundamentais para a gestão de uma infraestrutura e qualquer metodologia que conte com esse tipo de abordagem tende a aproveitar melhor os recursos disponíveis. Tais informações operacionais agregam muito na tomada de decisão durante a manutenção e gerenciamento da infraestrutura.

Tabela 4 – Métricas necessárias para a API.

	Métrica
CPU	usr, sys, idle, wait, stl
Memória	used, free , buff , cache

Neste contexto foi difundido a ideia de implementar uma API de monitoramento, a qual permitisse a agregação de ferramentas distintas. A Tabela 4 define as métricas de CPU e memória necessárias para esse projeto, as quais são necessárias para o cálculo do índice de desempenho discutido no Capítulo 2. O intuito de utilizar essa métrica composta é integrar uma métrica que sirva para o monitoramento de aplicações híbridas cujo consumo de recursos seja tanto *cpu-bound* e *memory-bound*.

A Figura 19 exibe o *Wrapper*, que possui entradas definidas para CPU e memória. Qualquer ferramenta adicional de monitoramento que utilize tais métricas está habilitada a ser utilizada. A partir disso, basta implementar um conector que forneça as informações para o *Wrapper*. Em seguida o *Wrapper* torna-se responsável por calcular o índice de desempenho e armazenar as métricas em uma base relacional.

O diagrama de sequência da Figura 20 destaca o fluxo do conector *dstat*, uma ferramenta de monitoramento utilizada na proposta da arquitetura. O *dstat*² realiza consultas das métricas de desempenho do sistema no */proc*. Para instanciar essa ferramenta foi implementado um método chamado *ProcessBuilder* que faz a chamada do *dstat* no terminal do *linux* e logo a seguir essa saída é tratada com o *StreamNormalizer* que deixa os dados mais limpos para serem repassados para o *connector*. Dessa forma, os dados são atribuídos a um construtor que pode organizá-los em variáveis simples e outras estruturas de dados mais complexas. Por fim, há o *DstatConector*

² <<http://dag.wiee.rs/home-made/dstat/>>

Figura 19 – Diagrama de sequência do wrapper do monitorAPI.

Fonte: Elaborada pelo autor.

Figura 20 – Diagrama de sequência do connector dstat no monitorAPI.

Fonte: Elaborada pelo autor.

que faz uma atribuição ao *Wrapper*, o que permite realizar uma requisição ao *DstatCollector* que salva as métricas de desempenho, para depois serem utilizadas no mapeamento dos recursos pelo GER.

4.3.4.3 LogServer

O *log* é um componente auxiliar muito importante dentro de qualquer tipo de sistema ou infraestrutura. Uma base de dados estruturada com informações gerenciais de todos os fluxos das informações na arquitetura é utilizada, e algumas informações ajudam a identificar gargalos, falhas, faltas e erros.

Este sistema de agrupamento de informações é imprescindível e por isso o servidor de *log* proposto visa o armazenamento de tempos de execução, execução de métodos e classes, assim como de mensagens das saídas do sistema. Por meio dos tempos de execução é possível ter uma dimensão da quantidade de dados que cada módulo pode gerar na arquitetura.

Por ser um sistema distribuído, caso uma das partes exceda o tempo, outros componentes podem gerar uma falta que pode ocasionar em uma falha e portanto um erro, que necessita ser registrado. Outro fluxo de informação armazenada nesse *log* são as mensagens de saídas do sistema, que direcionam quais são as ações tomadas pelos módulos bem como auxilia na identificação do comportamento do sistema.

LogTimer

O *LogTimer* é responsável pela coleta dos tempos e foi implementado em duas versões: uma voltada para testes e outra para a experimentação. Ambas as versões apresentam condições de coletar tempos de classes inteiras ou apenas de trechos específicos de algum método o que auxilia na depuração do código e identificação de gargalos.

Figura 21 – Diagrama de sequência do *logserver plugin* método.

Fonte: Elaborada pelo autor.

A Figura 21 apresenta a versão de testes e suas concepções são realizadas apenas se há permissão de sua execução. O *GetMethodName* recupera a *tag* daquele método e verifica a permissão em um arquivo de propriedades que contém todos os métodos e classes. A *tag*

de um método/classe pode ter duas marcações "*Enable*" para permissão habilitada a colher o tempo, e "*Disable*" para permissão negada a coletar o tempo. Esse controle de permissão evita a execução desnecessária do *GetSartTimer* e *GetEndTimer* que colhe os tempos iniciais e finais de um método ou classe. Após coletar esses tempos ainda há o *TimeCalc* que calcula os tempos, datas e etc, para então as informações serem salvas no banco de dados relacional por meio do *SQLManipulator*.

Para a depuração de desempenho essa versão de testes permite o controle de execução do servidor de *log*, o que no geral economiza operações I/O tanto por conta do arquivo de propriedades quanto pelo banco em modelo relacional. Todavia essa versão foi utilizada apenas para testes pela seguinte razão: o arquivo de propriedades para a recuperação das *tags*, recebe muita demanda sob um mesmo arquivo, o que gera um gargalo na recuperação dessas *tags*.

Figura 22 – Diagrama de sequência do *logserver plugin* simplificado para método.

Fonte: Elaborada pelo autor.

Em razão do gargalo com a execução de experimentos, foi realizado um clone do *Logtimer*, porém dessa vez sem o controle de propriedades de permissão de execução. A Figura 22 destaca os métodos *GetSartTimer*, *GetEndTimer*, *TimeCalc* e *SQLManipulator*. Sem a influência do controle de propriedades, o gargalo foi reduzido e assim a versão experimental é um clone que preserva a essência da versão de testes, porém com maior desempenho.

LogMessages

O componente *LogMessages* tem a função de obter as saídas do sistema e para isso foi utilizado a biblioteca em java *log4j* ³, a qual possibilita a implementação dessa função de

³ <<https://logging.apache.org/log4j/>>

log das mensagens. Para isso foi utilizado um arquivo de propriedades com uma configuração pré-estabelecida em tempo de execução.

O escopo das informações gravadas são representadas pelos seguintes níveis TRACE, DEBUG, WARN, INFO, ERROR e FATAL. Toda a aplicação foi projetada com essas *tags* e toda a mensagem ao ser executada corresponde a algum destes níveis. Dessa forma tem-se um *log* completo que contém informações separadas por tipo. Todas as mensagens engloba data, classe, método correspondente e nível de *logging*.

Figura 23 – Diagrama de sequência do *log4j*.

Fonte: Elaborada pelo autor.

A ferramenta pode ser configurada para salvar esses *logs* de diversas formas, tanto em arquivos de *log* quanto em banco de dados. Como as configurações são feitas em tempo de execução não há a necessidade de alterar o binário do sistema, mas apenas utilizar essa biblioteca em conjunto com o sistema para realizar o *logging*. A Figura 23 representa como o *log4j* intercepta a mensagem e salva no banco de dados como foi projetado para esse projeto.

4.4 Infraestrutura Computacional

Para a execução dos experimentos, assim como testes e ajustes da solução, foram utilizados os equipamentos computacionais do LaSDPC que dispõe atualmente de 3 *clusters*: o Cosmos⁴ com 19 *hosts* físicos que tem a capacidade total de 76 máquinas virtuais (4 por *host*), o Andromeda⁵ com 12 *hosts* físicos que tem a capacidade de 96 máquinas virtuais (8 por *host*) e o Halley⁶ com 13 *hosts* físicos que tem a capacidade de 104 máquinas virtuais (8 por *host*). Assim

⁴ <<http://infra.lasdpc.icmc.usp.br/cosmos/>>

⁵ <<http://infra.lasdpc.icmc.usp.br/andromeda/>>

⁶ <<http://infra.lasdpc.icmc.usp.br/halley/>>

com um total em ambiente de 276 máquinas virtuais cada uma, com 1vcpu e 4 GB de memória RAM, tal conjunto forma a Cloud privada LasDPC. A configuração das VM's foram baseadas nos modelos da Amazon⁷ que corresponde a uma m6g.medium, e no Google Cloud instances⁸ que corresponde a uma n1-standard-1. As configurações gerais dos *hosts* físicos são exibidas conforme a Tabela 5.

Tabela 5 – Tabela dos recursos computacionais utilizados.

Cluster	Cosmos	Andrômeda	Halley
Nº de Nodes	19	12	13
CPU	Intel Core 2 Quad 2.66 GHz	AMD Vishera 4.2 Ghz	Intel Core I7 3.60 Ghz
Memória	8 GB	32 GB	32 GB
Disco	SSD 240GB, HDD 160 GB	SSD 240GB, HDD 2 TB.	SSD 240GB, HDD 2 TB.

A infraestrutura utilizada possui uma rede estruturada e interconectada em 3 *clusters*, com configurações adicionais como por exemplo a existência de 3 interfaces de rede por *host* físico. Este recurso amplia a comunicação para uma estrutura com alta disponibilidade. Também dispõe de *nobreaks* e uma rede elétrica estável. Todas essas características tornam a infraestrutura utilizada apta para executar os experimentos do presente trabalho.

Ferramentas

Durante a elaboração da MoHRiPA foi necessário a utilização de diversas ferramentas, algumas com função gerencial e outras com propósito operacional ou de modelagem de sistema. A seguir são apresentados os itens que pertencem a arquitetura final, desde ferramentas de monitoramento, virtualização, banco de dados, servidores, etc.

- **Dstat:** Ferramenta de monitoramento utilizada, por se tratar de uma solução *open-source*. Seu uso foi pela praticidade e versatilidade em relação a outras ferramentas, pois funciona no mesmo molde de consulta ao */proc*, que contém as informações de desempenho;
- **KVM/libvirt:** Ferramenta de virtualização padrão no projeto, por ser um dos virtualizadores mais utilizados e por ter uma comunidade ativa que sempre busca aperfeiçoar a ferramenta, além de ser utilizada na infraestrutura do LASDPC;
- **Mysql:** O banco de dados utilizado foi o *Mysql* por conta da praticidade do uso de um banco relacional, além de ser fácil de agregar os dados, independente da linguagem utilizada;
- **Tomcat:** Servidor *Web tomcat*⁹. Foi utilizado durante a fase de implementação, assim como na fase de experimentação da solução, por ser um dos servidores mais utilizados e com uma excelente documentação;

⁷ <<https://aws.amazon.com/en/ec2/instance-types/>>

⁸ <<https://cloud.google.com/compute/docs/instances>>

⁹ <<http://tomcat.apache.org/>>

- **Log4J:** Biblioteca de *Log* utilizada para armazenar os possíveis erros, avisos entre outras informações que um sistema computacional pode apresentar;
- **JSch:** Biblioteca de conexão remota que foi utilizada para compor o módulo de alocação. Seus recursos de acesso remoto e execução de comandos foram fundamentais para a elaboração desse módulo;
- **Maven:** Ferramenta de manutenção de projeto usado durante a implementação. Possui diferentes bibliotecas para diversos propósitos, desde execução de comandos, conexões remotas, etc; O *maven*¹⁰ auxiliou no controle de dependências do projeto assim como o *deploy* da aplicação;
- **Spring:** *Framework Spring*¹¹ permitiu implementar certos componentes Web da arquitetura e auxiliou com os padrões de *Model-View-Controller*., possibilita tornar a solução mais profissional com um código limpo e de fácil manutenção;
- **Jmeter:** Ferramenta utilizada para o teste de carga, a qual oferece uso de grupos de *threads* e distribuições estatísticas. O *Jmeter*¹² possibilita simular diferentes comportamentos de carga de trabalho, o que é importante para avaliar a arquitetura na execução dos testes realizados;

Toda a solução foi desenvolvida com a linguagem java pela versatilidade de uso da linguagem assim como as diferentes tecnologias que podem ser agregadas. Isso expande o leque de soluções possíveis especialmente porque a arquitetura é composta de diferentes componentes. Tais componentes agregaram funções e soluções distintas, como no caso do alocador que precisou de uma biblioteca de conexão remota, conceitos de programação paralela e métodos dinâmicos.

As ferramentas apresentadas são as primárias do sistema, ainda que o mesmo tenha sido desenvolvido, já considera a possibilidade de ser acoplada novas ferramentas. Isso permite explorar a influência que um virtualizador pode gerar na alocação de máquinas virtuais, como um banco relacional pode ser o limitador na execução do fluxo, etc.

4.5 Considerações Finais

Neste Capítulo foram apresentados todos os 7 componentes da arquitetura: *Broker*, *GER*, *Transportador*, *LogServer*, *Alocador*, *MonitorAPI* e *ClusterLoader*. A seguir foram explorados em detalhes o funcionamento de cada componente, o fluxo de execução das tarefas, discussão sobre as ferramentas, descrição sobre as bibliotecas e os recursos computacionais que foram utilizados para o desenvolvimento deste projeto.

¹⁰ <<https://maven.apache.org/>>

¹¹ <<https://spring.io/>>

¹² <<https://jmeter.apache.org/>>

Dentre os tipos de atendimentos que a arquitetura possui, a atribuição de serviços e a alocação de máquinas virtuais foram exemplificados e devidamente descritos. Outros serviços complementares como o *LogServer* e o *MonitorAPI* auxiliam a gestão dos experimentos executados, uma vez que, apresentam-se dois diferentes fluxos para avaliar, os quais serão objetos de análise no próximo Capítulo.

Figura 24 – Fluxo Completo da Arquitetura

Fonte: Elaborada pelo autor.

Em relação ao fluxo completo da arquitetura, é apresentado o cenário (Figura 24) ao qual não conta com o alocador, visto que, o mesmo faz parte do *broker*. O fluxo compreende a operação fim a fim do cliente com o *broker*, que dispõe de duas formas de serviço, atribuição de serviços e alocação de máquinas virtuais. Ambos demandam de todos os componentes, diferenciado apenas pelo serviço que será oferecido ao cliente.

No próximo Capítulo será discutido a avaliação experimental do protótipo, a qual engloba cenários referentes a alocação de máquinas virtuais e atribuição de serviços. Também será apresentado os métodos formais utilizados, como foram executados os experimentos e a análise dos resultados ao considerar o contexto atual da arquitetura.

AVALIAÇÃO EXPERIMENTAL

5.1 Considerações Iniciais

Este capítulo compreende a avaliação experimental da arquitetura MoHRiPA e será descrito na seção 5.2 qual foi a configuração do ambiente de testes, assim como os fluxos de informações entre os hosts utilizados. Na seção 5.3 há uma discussão sobre o planejamento de experimentos realizando uma descrição dos métodos, objetivos das avaliações e como foi planejado o modelo fatorial completo dos respectivos cenários. Na seção 5.4 é realizado uma análise dos resultados dos respectivos cenários e por fim as considerações finais deste capítulo que corroboram com as conclusões do próximo capítulo e as contribuições do projeto de mestrado.

5.2 Configuração do Ambiente

A execução dos experimentos demanda dois tipos de recursos (hosts), os servidores que são alocados para os componentes da arquitetura como *Broker*, *GER*, *Logserver*; e os provedores de serviço que armazenam as aplicações ou serão gerenciadas pelo virtualizador. Tendo em vista que a infraestrutura computacional dispõe de 276 máquinas virtuais, advindas de *hosts* físicos heterogêneos entre si, foram atribuídos *hosts* físicos às máquinas servidores para não haver o compartilhamento de recursos, assim como tirando da trilha de atendimento, reduzindo o número de máquinas virtuais utilizadas, mas otimizando os servidores para apenas uma função.

Sendo assim foram alocadas 3 máquinas físicas exclusivamente para categoria servidores, e as demais sendo utilizadas para atribuição de serviços (provedores) e alocação de máquinas virtuais. Entretanto, como foi realizado a execução de testes de carga, uma máquina física foi reservada para tal finalidade. Assim a ferramenta *Jmeter* que auxilia com os testes de carga, terá um recurso independente não tendo outros fatores que possam influenciar sua tarefa.

A infraestrutura para a execução dos experimentos ficou organizada conforme demons-

Figura 25 – Representação da Configuração do Ambiente de Testes

Fonte: Elaborada pelo autor.

trado na Figura 25, o *LogServer* colhe o *logging* dos *Broker* e *GER*, assim como dos outros componentes como o alocador e o transportador. Dessa forma o fluxo geral em que o *Broker* toma todas as decisões e ações nos provedores, isso a partir do pool de recursos que o *broker* solicita do *GER*. Dessa forma a infraestrutura opera conforme o fluxo descrito na Seção 4.

5.3 Planejamento de Experimentos

A avaliação de desempenho faz o uso de técnicas formais, as quais devem obedecer certas regras, sendo a atribuição de fatores de avaliação, e os correspondentes níveis de cada fator. Um fator pode ser qualquer agente que possa impactar na avaliação, seja o tipo de serviço que será avaliado ou uma operação interna da arquitetura. Os fatores são divididos em níveis que representam o mesmo agente daquele fator, porém em diferentes intensidades ou propósitos. Dessa forma os níveis podem ser compostos por métricas numéricas (tempo entre requisições, quantidade de clientes) ou segmentos de aplicações (tipo de aplicação, funcionalidade de uma aplicação), etc.

A Tabela 6 exibe os principais fatores passíveis de modelar diferentes modelos de avaliação desde fatorial simples, parcial e completo. Tais fatores possuem níveis que são variantes das opções passíveis de experimentos dentro da arquitetura. Os fatores considerados possíveis são:

Tabela 6 – Representação dos principais fatores e níveis potenciais do projeto.

Fatores	Níveis
Distribuição Estatística	Constante, Gaussiana, Poisson, Uniforme, Exponencial.
Tipo de Atendimento	Atribuição de Serviços, Alocação de Máquinas Virtuais.
Serviço Prestado	Cpu-bound, Memory-bound, Hybrid-bound.
Algoritmos de Mapeamento	ListScheduling, HashRefresh.
Operações de Alocação	Start, Stop, Shutdown, Resume.
Quantidades de Clientes	1,2,3,n.
Fonte da Requisição	Local, Remoto.
Tempo entre Requisições	3,5,10,n.

- **Distribuição Estatística:** Por meio do uso das distribuições estatísticas a simulação de comportamentos da chegada de requisições pode ajudar na avaliação de pontos do sistema que não se adéquem a alguns números de requisições;
- **Tipo de Atendimento:** A arquitetura fornece diferentes tipos de provisionamento de serviços, como a atribuição de aplicações e a alocação de máquinas virtuais;
- **Serviço Prestado:** As aplicações oferecidas como serviço podem causar diferentes impactos na arquitetura por se tratar de aplicações com peso na decisão no escalonamento de recursos;
- **Algoritmos de Mapeamento:** A arquitetura apresenta diferentes mecanismos para atribuir o mapeamento de recursos e a diferença entre os algoritmos pode influenciar no atendimento das requisições;
- **Operações de Alocação:** o virtualizador dispõe de operações para a orquestração das máquinas virtuais, sendo fundamental para a administração da infraestrutura. As operações variadas apresentam diferentes fluxos operacionais na arquitetura;
- **Número de Clientes:** Qualquer arquitetura orientada a serviço depende da capacidade de mensurar a escalabilidade. Esta pode influenciar o atendimento de requisições e impactar negativamente no tempo de resposta do serviço.
- **Fonte da Requisição:** A relação da fonte da requisição por parte do cliente, indica problemas internos de conectividade da arquitetura;
- **Tempo entre Requisições:** tão importante quanto a distribuição, o que impacta diretamente na chegada das requisições são os tempos entre as requisições.

Após definir os principais fatores e seus respectivos níveis, nas próximas sub-seções seguindo o modelo fatorial, o planejamento de experimento nasce do objetivo do experimento. Isso também ocorre com os cenários que podem ser montados e assim avaliados nos experimentos: os comportamentos da arquitetura, os gargalos e o tempo de resposta.

5.3.1 Cenário 1

O atendimento de serviços da arquitetura depende de um *pool* de recursos atualizado, uma vez que toda requisição que chegue ao *broker*, depende anteriormente do mapeamento de recursos que o GER. O cenário 1 compreende a avaliação de desempenho referente aos algoritmos de mapeamento de recursos. Esse experimento foi idealizado a fim de determinar qual algoritmo teria o melhor desempenho, visto que ambos apresentam formas diferentes de funcionamento:

- **ListScheduling:** algoritmo de mapeamento para ambientes com configurações heterogêneas. Esse algoritmo realiza um mapeamento consistente e por conta de uma ordenação em lista com diferentes valores, é possível criar listas para diferentes atribuições. A abordagem do *ListScheduling* no MoHRiPA tem a peculiaridade do tempo de cadência do mapeamento. Sendo assim, a cada 10 segundos atualiza a lista garantindo um mapeamento mais apurado;
- **HashRefresh:** algoritmo de mapeamento *ad-hoc* autoral do MoHRiPA, com o intuito de realizar um mapeamento de alto desempenho, o *HashRefresh* realiza o mapeamento de recursos virtuais em uma tabela *hash*, e atualiza essa tabela por execução.

A Tabela 7 demonstra os fatores e níveis que serão executados no planejamento do cenário 1. Entre os fatores destaca-se a distribuição estatística, tempo entre as requisições e os algoritmos de mapeamento.

Tabela 7 – Tabela de Fatores e Níveis do Cenário 1.

Fator	Nível
Distribuição Estatística	Uniforme, Exponencial
Tempo entre Requisições	3 Segundos, 5 Segundos
Algoritmo de Mapeamento	Hashrefresh, ListScheduling

Estabelecendo esses fatores e variação dos níveis, as distribuições estatísticas também influenciam no comportamentos das requisições. O tempo entre as requisições pode influenciar no comportamento direto da distribuição utilizada e por fim o algoritmo de mapeamento que por suas peculiaridades de funcionamento, pode gerar diferentes tipos de atendimento pelo *broker*.

Os experimentos executados no cenário 1, correspondentes a Tabela 8 estão no escopo de atendimento de serviço. Assim será avaliado a capacidade da arquitetura atender as requisições, em variadas distribuições e tempos, utilizando os dois algoritmos. A variação de fatores e níveis resultou em 8 experimentos, como mostra a tabela 8.

O experimento será realizado com 10 repetições, sendo cada repetição correspondendo a 100 requisições de serviço que é a capacidade de máquinas estabelecida para o experimento. O número de repetições se demonstrou satisfatório e os dados apresentados na análise de dados corroboram com a integridade e qualidade do experimento.

Tabela 8 – Tabela de Experimentos do Cenário 1

Experimento	Distribuição Estatística	Tempo entre Requisições	Algoritmo de Mapeamento
A	Uniforme	3 Segundos	HashRefresh
B	Uniforme	5 Segundos	HashRefresh
C	Exponencial	3 Segundos	HashRefresh
D	Exponencial	5 Segundos	HashRefresh
E	Uniforme	3 Segundos	ListScheduling
F	Uniforme	5 Segundos	ListScheduling
G	Exponencial	3 Segundos	ListScheduling
H	Exponencial	5 Segundos	ListScheduling

Foram utilizadas 4 máquinas servidores 1 *broker*, 1 GER, 1 *Logserver* e 1 Jmeter Cliente. Em contrapartida será utilizada 100 máquinas virtuais para serviço. Pelo módulo *ClusterLoader* será instanciado a aplicação *Cpu-bound*. Dessa forma, a variação dos índices de desempenho foram definidas por aplicações *Cpu-bound*, e o mapeamento do pool de recursos do *broker* será composta por essa métrica de desempenho.

5.3.2 Cenário 2

A atribuição de máquinas virtuais da arquitetura depende da disponibilidade de máquinas virtuais, seja pela disponibilidade para executar a operação ou a atribuição de tarefas. O cenário 2 compreende a avaliação de desempenho do fluxo operacional da alocação de VMs e a atribuição dessas máquinas podem ter diferentes estratégias e formas de serem realizadas. Com o objetivo de estabelecer a diferença entre as operações de alocação e como a arquitetura se comporta em relação ao alocador, esse experimento foi idealizado a fim de determinar qual fluxo de operação teria o melhor desempenho, pressupondo quatro formas de operação:

- **Start:** Liga a máquina virtual, e demanda um tempo para montar todo o sistema;
- **ShutDown:** Desliga a máquina virtual. Por questões de interromper processos, pode levar um tempo que depende do tipo de virtualizador pra desligar;
- **Stop:** Pausa todos os processos e coloca a máquina num estado de *Stand-by*, sendo que há um tempo pra que esses processos sejam interrompidos;
- **Resume:** Retoma os processos de uma máquina virtual que estava parada. Essa retomada do estado *Stand-By* vai consumir tempo dependendo da carga que estava sendo empregada.

A Tabela 9 demonstra os fatores e níveis que foram executados no planejamento do cenário 2. Entre os fatores destaca-se a distribuição estatística, tempo entre as requisições e os tipos de operação do virtualizador.

Ambos os cenário 2A e 2B foram modelados para as mesmas distribuições estatísticas variando o tempo assim como apresentado no cenário 1. Em relação ao cenário 2A pela Tabela

Tabela 9 – Tabela de Fatores e Níveis do Cenário 2

Fator	Níveis
Distribuição Estatística	Uniforme, Exponencial
Tempo entre Requisições	3 Segundos, 5 Segundos
Operações do Virtualizador	Shutdown, Start up, Resume, Stop

10 podemos observar as variações dos tempos e distribuições, porém o principal fator a ser analisado será a operação de virtualização. Dado que o tempo entre desligar e ligar um máquina virtual pode ser ou não vantajoso para a arquitetura, buscamos medir os tempos e a influência que essas operações causam na arquitetura como um todo, por se tratarem de operações mais intensivas computacionalmente.

Tabela 10 – Tabela de Experimentos Cenário 2A

Experimento	Distribuição Estatística	Tempo entre Requisições	Operações do Virtualizador
A	Uniforme	3 Segundos	Shutdown
B	Uniforme	5 Segundos	Shutdown
D	Uniforme	3 Segundos	Start Up
E	Uniforme	5 Segundos	Start Up
D	Exponencial	3 Segundos	Shutdown
E	Exponencial	5 Segundos	Shutdown
J	Exponencial	3 Segundos	Start Up
K	Exponencial	5 Segundos	Start Up

Da mesma forma como especificado no cenário 2A, no cenário 2B como representado pela Tabela 11, há as variações de distribuições estatísticas e de tempo entre as requisições. Porém essa cenário será focado nas operações de *resume* e *stop*, visto que interromper processos podem ou não ocasionar travamentos. A idéia é descobrir qual seria a vantagem da arquitetura utilizar essas técnicas, que são bem menos invasivas como desligar e ligar máquinas virtuais. Para isso ser analisado a influência dessas operações e por fim uma comparação entre os cenários 2A e 2B foram considerados.

Tabela 11 – Tabela de Experimentos Cenário 2B .

Experimento	Distribuição Estatística	Tempo entre Requisições	Operações do Virtualizador
A	Uniforme	3 Segundos	Pause
B	Uniforme	5 Segundos	Pause
D	Uniforme	3 Segundos	Resume
E	Uniforme	5 Segundos	Resume
D	Exponencial	3 Segundos	Pause
E	Exponencial	5 Segundos	Pause
J	Exponencial	3 Segundos	Resume
K	Exponencial	5 Segundos	Resume

O experimento realizado contou com 10 repetições, sendo cada repetição correspondente a 100 requisições de máquinas virtuais. O número de repetições se demonstrou satisfatório e os dados apresentados na análise de dados corroboram com a integridade do resultado apresentado.

Foram utilizadas 4 máquinas servidores 1 *broker*, 1 GER, 1 *Logserver* e 1 Jmeter Cliente. Em contrapartida foram utilizadas 100 máquinas virtuais para alocação. O intuito desse experimento é determinar se é vantajoso para a arquitetura ligar e desligar ou parar e retomar máquinas virtuais. Determinando tal procedimento pode-se inferir uma política de uso, que utilize os recursos *non-stop* ou instancie conforme a necessidade.

5.4 Análise dos Resultados

A análise de resultados compreende a interpretação dos dados, os quais foram coletados pela execução de experimentos. A fim de entender o conjunto de dados é necessário analisá-los e interpretá-los de forma clara e assim destacar comportamentos de um sistema computacional: como os principais gargalos do sistemas, a velocidade de processamento ou o tempo médio de resposta do servidor (TAKAGI, 1991). Tais métricas podem ajudar a determinar cenários críticos, experimentos indevidos, comportamentos anormais dentro da infraestrutura, ressaltando os módulos que podem funcionar de forma indevida ou com baixa rentabilidade (HUGHES; BARTLETT, 2002).

Cada cenário descrito na seção posterior, será analisado por meio de gráficos variados, tabelas e interpretação dos resultados. O intuito dessa seção é destacar a particularidade de cada cenário e discutir a fundamentação dos resultados obtidos. Desta forma, com essas análises são demonstradas as contribuições da arquitetura MoHRiPA.

5.4.1 Resultados obtidos no Cenário 1

Para realizar uma análise precisa dos experimentos, foram estabelecidos alguns objetivos relacionados às análises: identificar o gargalo do sistema e identificar a influência dos fatores. O delineamento fatorial 3^2 foi baseado em dois algoritmos de mapeamento, mas com algumas particularidades. A variação no tempo de chegada das solicitações e o comportamento das distribuições estatísticas aumentaram a necessidade de variação dos fatores.

Para comparar os diferentes experimentos, na Figura 26, mostramos um gráfico de barras referente à distribuição exponencial. Os tempos referentes ao *HashRefresh* em todos os módulos mostraram um tempo médio mais longo e concluímos que devido à solicitação exponencial e atualização constante do algoritmo houve aumento do tempo médio. Quanto ao *ListSheduling*, o tempo médio é o mais longo, apenas no cenário de 5 segundos entre as solicitações.

A Figura 27 compreende o gráfico de barras da distribuição uniforme. O comportamento dessa carga, sinaliza para a normalidade em relação ao tempo médio, e nota-se que o *HashRefresh* apresentou um tempo maior em todos os módulos, bem como observado ao utilizar a distribuição exponencial pelo motivo de funcionamento do algoritmo que gera mais processamento. Em relação ao *ListScheduling*, demonstrou-se que a distribuição Exponencial teve um tempo médio superior a todos os outros experimentos. Na distribuição uniforme, por sua vez, o *ListScheduling*

Figura 26 – Tempo Médio de Resposta Exponencial.

Fonte: Elaborada pelo autor.

Figura 27 – Tempo Médio de Resposta Uniforme.

Fonte: Elaborada pelo autor.

também demonstrou um tempo de resposta superior, reforçando assim que o algoritmo não se adapta devido ao tempo de cadência da lista.

As Figuras 28 e 29 representam o *boxplot* dos experimentos realizados em ambos os algoritmos *ListScheduling* e *HashRefresh*. Informações adicionais podem ser encontradas na Tabela 12, que contém a média, variação e desvio das soluções que corroboram com as análises acima.

A latência total representada pela Figura 30, indica que a medida que o tempo entre solicitações aumenta, o tempo médio tende a cair, o que reforça que o cliente é influenciado pelo módulo GER, uma vez que todo o mapeamento centralizado neste módulo gera um gargalo no tempo de resposta do cliente. No *ListScheduling* devido à taxa de atualização, quando o tempo entre as solicitações é de 5 segundos, podemos notar que a latência foi alta. Isso se deve à maneira como o algoritmo funciona, para concluir que a taxa de atualização da lista, pode ser muito alta, afinal, em alguns casos, solicitações podem ter sido realizadas durante a cadência da lista, aumentando assim o tempo para atender à solicitação. Enquanto isso, comparando-

Figura 28 – Distribuição de tempo do ListScheduling

Fonte: Elaborada pelo autor.

Tabela 12 – Tabela de Latência .

Experimento	Média	Variação	Desvio
LS Exp 3 sec	110	93,37150315	8718,2376
HR Exp 3 sec	140	305,5608252	93367,41792
LS Exp 5 sec	789,5	505,7696433	255802,9321
HR Exp 5 sec	150	515,926416	266180,0668
LS Uni 3 sec	144	371,0925005	137709,6439
HR Uni3 sec	155,5	275,1261872	75694,41886
LS Uni 5 sec	184	528,7296375	279555,0296
HR Uni 5 sec	153	440,8428065	194342,38

se com o algoritmo *HashRefresh*, não há uma latência alta no tempo de serviço, pois esse algoritmo é atualizado mediante a solicitação, mantendo o mapeamento sempre atualizado e, assim, contribuindo para aumentar a capacidade de serviço.

O cenário 1, apontou que o GER é o maior gargalo da solução e o melhor algoritmo é o *HashRefresh* em questão de tempo de resposta. A Figura 31 demonstra um gráfico de interação entre os principais fatores, que são as distribuições e algoritmos de mapeamento. Ao comparar os valores, é mostrado que o *ListScheduling* apresenta um desempenho melhor na distribuição uniforme, porém sofre muita influência quando utilizado a distribuição exponencial.

Figura 29 – Distribuição de tempo do HashRefresh

Fonte: Elaborada pelo autor.

Figura 30 – Distribuição de tempo da Latência.

Fonte: Elaborada pelo autor.

O *HashRefresh* por sua vez, apresenta menor influência e age independente da distribuição considerada. A conclusão é que a causa do *ListScheduling* ter esse desempenho inferior é justamente o tempo de atualização da lista.

Figura 31 – Gráficos de Interação entre Algoritmos de Mapeamento e Distribuições

Fonte: Elaborada pelo autor.

5.4.2 Resultados obtidos no Cenário 2

O cenário 2A retrata o experimento avaliando as operações *Shutdown* e *Start*. Os gráficos referente às Figuras 32 e 33 indicam que o alocador sofreu menor influência da distribuição referente a operação de *shutdown*, se comparado a operação *start*.

Analisando as distribuições especificamente, no caso da distribuição exponencial ocorreu um aumento em todos os módulos, tendo o maior impacto no *broker* pela taxa de chegada das requisições. Todavia na distribuição uniforme o tempo se manteve numa média com pouca variação. Visto que a carga uniforme não foi um problema para o alocador, pode-se concluir que o *broker* é o módulo mais afetado nesse experimento.

Figura 32 – Tempo Médio de Resposta Operação ShutDown

Fonte: Elaborada pelo autor.

A fim de concluir sobre a hipótese apontada acima, na Figura 34 há *boxplots* demonstrando a distribuição dos dados da distribuição uniforme. Nota-se que a dispersão dos dados é muito menor que na distribuição exponencial (Figura 35). Por outro lado, ao analisar a dispersão em relação às operações apenas com as gráficos de barra, não se pode afirmar claramente qual operação foi mais custosa.

Figura 33 – Tempo Médio de Resposta Operação StartUp

Fonte: Elaborada pelo autor.

Entretanto nos *boxplots*, entende-se que a operação de *shutdown* teve uma alta dispersão, isso sendo comparada com as requisições com menos tempo espaçados, o que faz sentido porque o *broker* recebe alta demanda de requisições, em relação aos dados apontados com um tempo maior entre as requisições.

Mesmo que o conceito do alocador seja *multi-thread*, ainda assim o primeiro módulo que recebe a requisição de alocação é o *broker*, o qual tem a função total de obter o recurso gerenciado e atribuí-lo de volta para o cliente, enquanto o alocador realiza sua função de orquestração de recursos.

A operação *start* representada nos gráficos (Figuras 34 e 35), demonstra que a dispersão dos dados aumentou em relação ao tempo entre as requisições, analisando a operação em si o tempo de instanciação de uma máquina virtual pode estar muito atrelado a forma que foi desligada anteriormente, observando o tempo de *shutdown* anterior, quando a máquina leva menos tempo para desligar, na operação de *start* esse tempo será maior provavelmente porque quando a máquina fora desligada rapidamente, e assim algum processo pode não ter sido devidamente finalizado.

Assim entende-se que as operações geram influências harmônicas entre si, uma vez que uma operação *shutdown* mal desenvolvida, acarrete em um maior tempo para a instanciação pela operação *start*. Sendo assim os experimentos com a uniforme em 3 segundos, exponencial em 3 segundos e exponencial em 5 segundos confirmam essa hipótese. Em contrapartida o experimento uniforme em 5 segundos em ambas as operações de *start* e *shutdown*, apresentaram uma dispersão semelhante corroborando que caso as operações, sejam as operações bem sucedidas, não ocorrerá um tempo maior para realizar as operações inversas, ou seja quando a máquina desliga corretamente a mesma será iniciada mais rápida e vice e versa.

A discussão do cenário 2B é referente ao fluxo operacional das operações *resume* e *stop*. Analisando as Figuras 36 e 37, pelos gráficos de barra apresentados, pode-se observar que a

Figura 34 – Distribuição de tempo Uniforme das Operações ShutDown StartUp

Fonte: Elaborada pelo autor.

variação entre os valores é simbólica, uma vez que operações são menos intensivas como as operações de *shutdown* e *start*; As operações *resume* e *stop* se mantêm em uma média bem próximas, não demonstrando forte influência independente dos fatores de distribuição estatística e tempo entre as requisições.

Operações que não mudam o estado físico da máquina (ligada e desligada) de orquestração como *resume* e *stop*, não tem muita influência sobre o recurso computacional. A fim de analisar a influência direta das operações, a Figura 38 apresenta o *boxplot* dos experimentos analisando a operação *stop* na distribuição uniforme. Observa-se que a dispersão dos dados na operação *resume* foi maior que a operação *stop*, visto que o *stop* apenas coloca a máquina virtual em estado de *stand-by*. Ao se considerar distribuição de requisições uniforme não houve muita influência. Em relação a operação *resume*, houve um aumento na dispersão dado a retomada de estado da máquina, sendo uma operação um pouco mais custosa por realocar processos dentro da CPU. A dispersão teve um aumento em relação ao tempo entre as requisições, dando a entender que quanto maior o tempo entre as requisições maior será o tempo para as máquinas virtuais retomarem seus processamentos.

O cenário 2B por sua particularidade de ser operações invasivas, tornam a estratégia de *resume* e *pause* superiores em tempo de alocação para um cliente, visto que apenas analisando a velocidade de atribuição, desconsiderando custo energético e outros fatores, esta técnica se

Figura 35 – Distribuição de tempo Exponencial das Operações ShutDown StartUp

Fonte: Elaborada pelo autor.

Figura 36 – Tempo Médio de Resposta Operação Resume

Fonte: Elaborada pelo autor.

Figura 37 – Tempo Médio de Resposta Operação Stop

Fonte: Elaborada pelo autor.

mostra promissora.

Figura 38 – Distribuição de tempo Uniforme das Operações Resume Stop

Fonte: Elaborada pelo autor.

Ao analisar a Figura 39 em relação ao *boxplot* exponencial das operações resume e stop, a dispersão dos dados foi contrária a distribuição uniforme. O aumento na dispersão dos dados na operação *stop*, é por conta das requisições exponenciais não serem uniformes, a operação de *stop* teve um maior número de operações simultâneas, pois o *host* físico estava realizando

Figura 39 – Distribuição de tempo Exponencial das Operações Resume Stop

Fonte: Elaborada pelo autor.

múltiplas operações, as quais corroboram com o aumento no tempo de resposta. Dessa forma podemos concluir que a operação *stop*, exerce uma grande influência em relação ao tipo de carga que chega no *broker*, visto que a concorrência dentro do *host* físico para realizar a operação *stop* teve uma dispersão maior do tempo.

Figura 40 – Gráficos de Interação entre operações de Virtualização e Distribuições

Fonte: Elaborada pelo autor.

Por fim a Figura 40 representa o gráfico de interação entre as operações de virtualização e as distribuições estatísticas. As operações *shutdown* e *start*, como detalhado acima exercem

influências entre si e quando o tempo de *shutdown* de uma VM é baixo, o *start* será maior, porque provavelmente a máquina não desligou corretamente. Além disso o gráfico de interação também reforça a afirmação da operação *stop*, que apresenta grande influência na máquina física, por conta da disputa dos processos serem colocados em *Stand-by*, mostrando que é custoso parar múltiplas VMs.

5.5 Considerações Finais

O Capítulo de avaliação experimental analisou 2 conjuntos de experimentos relacionados aos tipos de serviços oferecidos pela arquitetura, que são: atribuição de serviço e alocação de recursos.

Em relação ao Cenário 1, que é referente a atribuição de serviços, após as devidas análises foi identificado o maior ponto de gargalo da arquitetura que é GER, responsável pelo mapeamento dos recursos. Isso ocorre por este ser muito requisitado e gera muita latência pela sua resposta ao *broker*, o que pode representar o aumento no tempo do serviço. Foram analisados dois algoritmos de mapeamento de recursos: o *ListScheduling* cujo desempenho foi inferior ao *HashRefresh* devido a suas diferenças de abordagem. Enquanto o *ListScheduling* tem um lista com tempo para atualização do mapa, o *HashRefresh* realiza atualizações na sua tabela *hash* a cada requisição. Em relação desempenho geral o *HashRefresh* ganha, mas também gera muito *overhead* em relação ao *ListScheduling*.

O Cenário 2 considerou os experimentos de alocação de máquinas virtuais, tendo dois subtipos de avaliação das operações: *shutdown-start* e *resume-stop*. Durante a análise das operações *shutdown-start* concluiu-se que a maior influência desse cenário se deve ao estado da máquina. Como essas operações apresenta um caráter mais intensa de mudar o estado da máquina, uma vez que um máquina virtual não desliga corretamente durante sua próxima inicialização, a mesma apresentará um tempo maior para ligar. Portanto essas operações de *start* e *shutdown* mal sucedidas apresentam influências em relação ao tempo de resposta, pois em um cenário onde essas operação de *start* e *shutdown* não demonstram dispersões, reforçam que não houver desligamentos indevidos e não influenciam no tempo de resposta.

Em relação às operações *resume-stop*, analisando os gráficos, quando a operação *resume* acontece, esta tende a ser a com o melhor desempenho e a menor dispersão. Sendo assim retomar uma máquina virtual paralisada, se provou mais rápido do que ligar uma VM. Entretanto parte do gargalo gerado na operação *stop*, é por conta da concorrência dos processos de *stop* dentro do *host* físico, uma vez que a operação *stop* é mais invasiva por interromper a VM, pausando seu estado e processamentos. Quando a operação *stop* ocorre, o *host* físico está paralisando parte do seu processamento real e isso induz que mesmo reduzindo sua carga de desempenho, o custo computacional e o tempo para realizar essa operação é maior, o que indica maior dispersão, como fora analisado.

A avaliação experimental apontou os principais gargalos, dentro de cada componente da arquitetura. No cenário 1 analisamos os algoritmos de mapeamento, os quais geram muita influência no GER, que por ser um componente fundamental para o *broker* conseguir atender as requisições de serviço, gera uma maior sobrecarga na arquitetura e no atendimento das requisições. No cenário 2 analisamos as operações realizadas pelo módulo alocador e embora esse seja *multi-thread*, o mesmo está alocado no mesmo recurso físico, o que faz o *broker* apresentar sobrecarga em função das requisições de VM's e pelas operações de orquestração. Diante dos experimentos e análise dos resultados, é possível identificar pontos funcionais da arquitetura e diversas melhorias que podem ser desenvolvidas em trabalhos futuros.

CONCLUSÃO

6.1 Considerações Iniciais

O objetivo deste capítulo é apresentar as conclusões do projeto de mestrado. Na seção 6.2 é apresentada uma lista das publicações e submissões geradas durante a vigência do projeto. Na seção 6.3 são discutidas as contribuições geradas com o desenvolvimento do trabalho. Na seção 6.4 uma análise de potenciais trabalhos futuros são apresentadas. Na seção 6.5 é discutido as limitações e dificuldade enfrentadas ao longo do projeto de mestrado. Por fim, na seção 6.6 conclui-se o trabalho.

6.2 Publicações

Durante o vigência do trabalho destacam-se as seguintes publicações:

POLACHINI, A. R. ; **ANDREAZI, GABRIEL T.** ; NAKAMURA, LUIS H. V. ; ESTRELLA, J. C. . Gerenciamento de IaaS utilizando um Dashboard Inteligente baseado em Ontologia e Feedbacks. In: 36^a Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 2018, Campos do Jordão - São Paulo. v. 1 (2018): Anais do Workshop de Trabalhos de Iniciação Científica e Graduação (WTG - SBRC 2018), 2018 (**Publicação e Apresentação**).

ANDREAZI, GABRIEL T.; ESTRELLA, J. C.; BRUSCHI, S. M.; MARTINS, WELLINGTON SILVA.; FERREIRA, ANTONIO M. A. . Performance Evaluation in an Architecture which Instance Hybrid Resources in Private Cloud. In: IEEE Cloud Summit 2020, 2020, Harrisburg, Pennsylvania, United States of America. v. 1(2020): Main Conference, 2020 (**Publicação e Apresentação**).

MARTINS, W. S.; ESTRELLA, J. C.; BRUSCHI, S. M.; ;AZEVEDO, LEONILDO J. M. ; **ANDREAZI, GABRIEL T.** . Performance evaluation for signing JSON tokens in access

control for the cloud of things. In: IEEE Cloud Summit 2020, 2020, Harrisburg, Pennsylvania, United States of America. v. 1(2020): Main Conference, 2020 (**Publicação**).

6.3 Contribuições

Dentre os objetivos pautados na seção de introdução, destacam-se:

- Elaborar e desenvolver uma arquitetura distribuída, no qual os módulos fossem desenvolvidos independentes, porém atuariam juntos para a solução final;
- Os módulos responsáveis pela alocação de recursos e monitoramento de recursos devem se comunicar com diferentes ferramentas de virtualização e monitoramento;
- O provedor de recursos possui um módulo auditor de recursos que auxilia o mapeamento de recursos. O auditor deve ser capaz de calcular diferentes mapeamentos, independente do algoritmo usado na seleção de recursos.

Acerca dos objetivos, foi desenvolvido uma arquitetura distribuída com módulos independentes, alguns com funções híbridas e independentes de ferramenta. Como por exemplo os módulos alocador que permite o uso de ferramentas variadas de virtualização e o *monitorAPI* que pode ser complementado com diferentes ferramentas de monitoramento. A arquitetura conta com um auditor de recurso que dispõe de dois algoritmos de mapeamento, autoral *Hashrefresh* e *Listcheduling* da arquitetura, em que ambos algoritmos possuem diferentes mecanismos de atualização e ordenação em seus mapas de recursos.

6.4 Trabalhos Futuros

Algumas análises não foram realizadas dado o escopo amplo que a arquitetura possui. Com o protótipo em funcionamento novas abordagens podem ser consideradas e avaliadas. É importante enfatizar que entre os trabalhos futuros pode-se destacar:

- **Análise da fonte proveniente da requisição:** utilizou-se apenas 1 máquina Jmeter para simular vários clientes com o intuito de realizar as requisições. Os mesmos experimentos poderiam ter outros pontos a serem analisados, caso essa carga fosse gerada de forma distribuída;
- **Complementar as análises dos algoritmos de mapeamento de recursos:** não foi realizado um estudo aprofundado sobre o conjunto dos resultados. Logo, é preciso automatizar essa análise por envolver outras métricas;

- **Analisar a fonte da requisição:** todos os testes foram realizados em ambiente local. Desta forma, é importante salientar que uma análise sobre o impacto da arquitetura num ambiente remoto poderia mensurar pontos referente a rede que não foram analisados, embora esse aspecto seja relevante;
- **Tempo total de uma atribuição de máquina virtual:** um protocolo de várias fases possibilita concluir qual é o tempo entre a operação e o total funcionamento de uma máquina virtual.

6.5 Limitações e Dificuldades

Durante a vigência do projeto de mestrado foram encontradas algumas limitações e dificuldades.

- **Elaboração da proposta:** Durante o início do projeto de mestrado, estávamos considerando realizar um trabalho voltado a políticas financeira. Após diversas reuniões e uma revisão da literatura dedicada, encontramos as lacunas e começamos o planejamento do projeto;
- **Desenvolvimento:** Devido a um número moderado de módulos, o desenvolvimento das aplicações levou muito tempo. Todos os códigos foram organizados e modelados em conceitos de boas praticas;
- **Configuração da infraestrutura:** Uma das atividades que demandaram maior tempo, foi a elaboração e configuração de todos os *hosts* físicos e virtuais utilizados para os experimentos. Como o ambiente de testes era uma ambiente que não contava com mecanismos de auto-configuração, as configurações de pacotes, rede e etc; foram todas realizadas individualmente com auxílio do orientador;
- **Teste e avaliação:** Antes de qualquer teste e avaliação, era fundamental a infraestrutura estar configurada corretamente desta forma os testes antes de serem realizados, eram feitas conferencias acerca da infraestrutura computacional.

6.6 Considerações Finais

Esse projeto envolveu diferentes conceitos desde virtualização, arquitetura, algoritmos de mapeamento, rede e etc. Durante a elaboração da solução foi fundamental o levantamento dos trabalhos da literatura a fim de definir objetivos bem como identificar lacunas e determinar metas para a conclusão do trabalho. O desenvolvimento de uma arquitetura em nuvem não é trivial já que há vários componentes com atribuições de tarefas de complexidades variadas.

Para realizar as etapas de execução do trabalho houve a necessidade de modelar o ambiente de testes, considerar desde a configuração de rede, máquinas virtuais, serviços e manipulação de portas no *firewall*. O trabalho considerou dois tipos de serviços que a nuvem oferece, representados pelas camadas Iaas e SaaS. A primeira é voltada para a venda de infraestrutura como um serviço e a segunda voltada para o consumo de uma aplicação oferecida como serviço dentro da infraestrutura.

REFERÊNCIAS

AL-ROOMI, M.; AL-EBRAHIM, S.; BUQRAIS, S.; AHMAD, I. Cloud computing pricing models: a survey. **International Journal of Grid and Distributed Computing**, v. 6, n. 5, p. 93–106, 2013. Citado na página 28.

ALHAMAD, M.; DILLON, T.; CHANG, E. Conceptual sla framework for cloud computing. In: IEEE. **4th IEEE International Conference on Digital Ecosystems and Technologies**. [S.l.], 2010. p. 606–610. Citado na página 23.

ALIVERDI, R.; NAENI, L. M.; SALEHIPOUR, A. Monitoring project duration and cost in a construction project by applying statistical quality control charts. **International Journal of Project Management**, Elsevier, v. 31, n. 3, p. 411–423, 2013. Citado na página 35.

ANDRZEJAK, A.; KONDO, D.; YI, S. Decision model for cloud computing under sla constraints. In: IEEE. **2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems**. [S.l.], 2010. p. 257–266. Citado na página 35.

Arabnejad, H.; Barbosa, J. G. List scheduling algorithm for heterogeneous systems by an optimistic cost table. **IEEE Transactions on Parallel and Distributed Systems**, v. 25, n. 3, p. 682–694, 2014. Citado na página 57.

Arif, H.; Sahoo, P. K. Threshold scheme approach to balance virtual machines load in private cloud. In: **2018 International Conference on Applied Engineering (ICAE)**. [S.l.: s.n.], 2018. p. 1–6. ISSN null. Citado nas páginas 45 e 46.

ARUNARANI, A.; MANJULA, D.; SUGUMARAN, V. Task scheduling techniques in cloud computing: A literature survey. **Future Generation Computer Systems**, Elsevier, v. 91, p. 407–415, 2019. Citado na página 38.

ATIEWI, S.; ABUHUSSEIN, A.; SALEH, M. A. Impact of virtualization on cloud computing energy consumption: Empirical study. In: **Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control**. New York, NY, USA: Association for Computing Machinery, 2018. (ISCSIC '18). ISBN 9781450366281. Disponível em: <<https://doi.org/10.1145/3284557.3284738>>. Citado nas páginas 45 e 46.

BRANCO, K. R. L. J. C. **Índices de carga e desempenho em ambientes paralelos/distribuídos - modelagem e métricas**. Tese (PhD dissertation) — ICMC - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo USP - Universidade de São Paulo, 2004. Citado nas páginas 36, 37 e 52.

BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. **Journal of systems and software**, Elsevier, v. 80, n. 4, p. 571–583, 2007. Citado na página 41.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. M. **Cloud computing: Principles and paradigms**. [S.l.]: John Wiley & Sons, 2010. v. 87. Citado nas páginas 31 e 32.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In: IEEE. **2008 10th IEEE International Conference on High Performance Computing and Communications**. [S.l.], 2008. p. 5–13. Citado na página 32.

CHACZKO, Z.; MAHADEVAN, V.; ASLANZADEH, S.; MCDERMID, C. Availability and load balancing in cloud computing. In: **International Conference on Computer and Software Modeling, Singapore**. [S.l.: s.n.], 2011. v. 14. Citado na página 37.

CHEN, Y.; SION, R. To cloud or not to cloud?: musings on costs and viability. In: ACM. **Proceedings of the 2nd ACM Symposium on Cloud Computing**. [S.l.], 2011. p. 29. Citado na página 28.

CHIEU, T. C.; MOHINDRA, A.; KARVE, A. A.; SEGAL, A. Dynamic scaling of web applications in a virtualized cloud computing environment. In: IEEE. **2009 IEEE International Conference on e-Business Engineering**. [S.l.], 2009. p. 281–286. Citado na página 32.

CHOWDHURY, N. M. K.; BOUTABA, R. A survey of network virtualization. **Computer Networks**, Elsevier, v. 54, n. 5, p. 862–876, 2010. Citado na página 33.

DESHPANDE, P. Cloud of everything (clet): The next-generation computing paradigm. In: **Computing in Engineering and Technology**. [S.l.]: Springer, 2020. p. 207–214. Citado na página 28.

ELSHAER, R. Impact of sensitivity information on the prediction of project's duration using earned schedule method. **International Journal of Project Management**, Elsevier, v. 31, n. 4, p. 579–588, 2013. Citado na página 35.

ERCAN, T. Effective use of cloud computing in educational institutions. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 2, n. 2, p. 938–942, 2010. Citado na página 32.

ESTRELLA, J.; SANTANA, R.; SANTANA, M.; BRUSCHI, S. Wsarch: A service-oriented architecture with qos. **Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions**, p. 352–380, 01 2011. Citado nas páginas 47 e 51.

GOHER, S. Z.; BLOODSWORTH, P.; RASOOL, R. U.; MCCLATCHEY, R. Cloud provider capacity augmentation through automated resource bartering. **Future Generation Computer Systems**, v. 81, p. 203 – 218, 2018. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X17304867>>. Citado nas páginas 45 e 47.

Hassan, M.; Chen, H.; Liu, Y. Dears: A deep learning based elastic and automatic resource scheduling framework for cloud applications. In: **2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)**. [S.l.: s.n.], 2018. p. 541–548. ISSN null. Citado nas páginas 45 e 48.

HASSELMEYER, P.; D'HEUREUSE, N. Towards holistic multi-tenant monitoring for virtual data centers. In: IEEE. **2010 IEEE/IFIP Network Operations and Management Symposium Workshops**. [S.l.], 2010. p. 350–356. Citado nas páginas 35 e 36.

HAZIR, Ö. A review of analytical models, approaches and decision support tools in project monitoring and control. **International Journal of Project Management**, Elsevier, v. 33, n. 4, p. 808–815, 2015. Citado na página 34.

HUGHES, M. D.; BARTLETT, R. M. The use of performance indicators in performance analysis. **Journal of sports sciences**, Taylor & Francis, v. 20, n. 10, p. 739–754, 2002. Citado na página 73.

KAPIL, D.; PILLI, E. S.; JOSHI, R. C. Live virtual machine migration techniques: Survey and research challenges. In: IEEE. **2013 3rd IEEE International Advance Computing Conference (IACC)**. [S.l.], 2013. p. 963–969. Citado na página 28.

KELECHI, G.; AKUJUOBI, C. M.; SADIKU, M. N.; CHOUIKHA, M.; ALAM, S. Internet of things and blockchain integration: Use cases and implementation challenges. In: SPRINGER NATURE. **Business Information Systems Workshops: BIS 2019 International Workshops, Seville, Spain, June 26–28, 2019, Revised Papers**. [S.l.], 2020. v. 373, p. 287. Citado na página 45.

KHORSHED, M. T.; ALI, A. S.; WASIMI, S. A. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. **Future Generation computer systems**, Elsevier, v. 28, n. 6, p. 833–851, 2012. Citado na página 28.

LAURAS, M.; MARQUES, G.; GOURC, D. Towards a multi-dimensional project performance measurement system. **Decision Support Systems**, Elsevier, v. 48, n. 2, p. 342–353, 2010. Citado na página 35.

LIU, F.; TONG, J.; MAO, J.; BOHN, R.; MESSINA, J.; BADGER, L.; LEAF, D. Nist cloud computing reference architecture. **NIST special publication**, v. 500, n. 2011, p. 292, 2011. Citado na página 29.

Liu, N.; Li, Z.; Xu, J.; Xu, Z.; Lin, S.; Qiu, Q.; Tang, J.; Wang, Y. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In: **2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)**. [S.l.: s.n.], 2017. p. 372–382. ISSN 1063-6927. Citado na página 45.

MELL, P.; GRANCE, T. *et al.* The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011. Citado nas páginas 23, 27 e 30.

MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. **Linux journal**, v. 2014, n. 239, p. 2, 2014. Citado na página 34.

Mhatre, M.; Shree, P.; Sharma, S. K. Prioritized job scheduling algorithm using parallelization technique in cloud computing. In: **2017 2nd International Conference for Convergence in Technology (I2CT)**. [S.l.: s.n.], 2017. p. 576–581. ISSN null. Citado nas páginas 45 e 46.

Monteiro, A.; Henriques, F. d. R. Qmapper: Dynamic power and performance management in virtualized web servers clusters. In: **2018 Eighth Latin-American Symposium on Dependable Computing (LADC)**. [S.l.: s.n.], 2018. p. 37–46. ISSN null. Citado nas páginas 45 e 48.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **EASE**. [S.l.: s.n.], 2008. v. 8, p. 68–77. Citado na página 43.

ROGERS, O.; CLIFF, D. A financial brokerage model for cloud computing. **Journal of Cloud Computing: Advances, Systems and Applications**, SpringerOpen, v. 1, n. 1, p. 2, 2012. Citado na página 27.

SAYADNAVARD, M. H.; HAGHIGHAT, A. T.; RAHMANI, A. M. A reliable energy-aware approach for dynamic virtual machine consolidation in cloud data centers. **The Journal of Supercomputing**, v. 75, n. 4, p. 2126–2147, Apr 2019. ISSN 1573-0484. Disponível em: <<https://doi.org/10.1007/s11227-018-2709-7>>. Citado nas páginas 45 e 47.

SHAHRAD, M.; KLEIN, C.; ZHENG, L.; CHIANG, M.; ELMROTH, E.; WENTZLAFF, D. Incentivizing self-capping to increase cloud utilization. In: **Proceedings of the 2017 Symposium on Cloud Computing**. New York, NY, USA: Association for Computing Machinery, 2017. (SoCC '17), p. 52–65. ISBN 9781450350280. Disponível em: <<https://doi.org/10.1145/3127479.3128611>>. Citado na página 45.

SHAW, S. B.; SINGH, A. A survey on scheduling and load balancing techniques in cloud computing environment. In: IEEE. **2014 international conference on computer and communication technology (ICCT)**. [S.l.], 2014. p. 87–95. Citado na página 28.

SINGH, S.; CHANA, I. A survey on resource scheduling in cloud computing: Issues and challenges. **Journal of grid computing**, Springer, v. 14, n. 2, p. 217–264, 2016. Citado na página 28.

TAKAGI, H. **Queueing analysis: a foundation of performance evaluation**. [S.l.]: North-Holland Amsterdam, 1991. v. 1. Citado na página 73.

TCHERNYKH, A.; SCHWIEGELSOHN, U.; TALBI, E.-g.; BABENKO, M. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. **Journal of Computational Science**, Elsevier, v. 36, p. 100581, 2019. Citado na página 37.

TSANGARIS, M. M.; KAKALETRIS, G.; KLLAPI, H.; PAPANIKOS, G.; PENTARIS, F.; POLYDORAS, P.; SITARIDI, E.; STOUMPOS, V.; IOANNIDIS, Y. E. Dataflow processing and optimization on grid and cloud infrastructures. **IEEE Data Eng. Bull.**, v. 32, n. 1, p. 67–74, 2009. Citado na página 35.

TURNBULL, J. **The Docker Book: Containerization is the new virtualization**. [S.l.]: James Turnbull, 2014. Citado na página 34.

YOUNGE, A. J.; LASZEWSKI, G. V.; WANG, L.; LOPEZ-ALARCON, S.; CARITHERS, W. Efficient resource management for cloud computing environments. In: IEEE. **International Conference on Green Computing**. [S.l.], 2010. p. 357–364. Citado na página 24.

ZHOU, R.; BASE, C. E.; CADER, T.; WANG, Z. **Management of airflow provisioning to meet a cooling influence redundancy level**. [S.l.]: Google Patents, 2019. US Patent 10,228,668. Citado na página 28.