
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Leandro Costa da Silva

# COSTA - Uma Solução Adaptativa Ciente de Custo para Migração de Aplicações entre Provedores de Nuvem

Recife

2023

Leandro Costa da Silva

# COSTA - Uma Solução Adaptativa Ciente de Custo para Migração de Aplicações entre Provedores de Nuvem

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

**Área de Concentração**: Redes de Computadores e Sistemas Distribuídos

**Orientador**: Nelson Souto Rosa

**Coorientador**: Robson Wagner Albuquerque de Medeiros

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

S586c Silva, Leandro Costa da
COSTA: uma solução adaptativa ciente de custo para migração de aplicações entre provedores de nuvem / Leandro Costa da Silva – 2023.
92 f.: il., fig., tab.

Orientador: Nelson Souto Rosa.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências e apêndices.

1. Redes de computadores e Sistemas distribuídos. 2. Sistemas adaptativos. 3. Gestão de custos. 4. Múltiplas nuvens. 5. Computação em nuvem. I. Rosa, Nelson Souto (orientador). II. Título

004.6 CDD (23. ed.) UFPE - CCEN 2024 – 006

**Leandro Costa da Silva**

**“COSTA - Uma Solução Adaptativa Ciente de Custo para Migração de Aplicações entre Provedores de Nuvem”**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovado em: 24/11/2023

**BANCA EXAMINADORA**

_____

Prof. Dr. Carlos André Guimarães Ferraz
Centro de Informática / UFPE

_____

Profa. Dra. Erica Teixeira Gomes de Sousa
Departamento de Computação / UFRPE

_____

Prof. Dr. Nelson Souto Rosa
Centro de Informática / UFPE
(**orientador**)

Dedico este trabalho à minha família e principalmente à minha mãe, que sempre esteve comigo e me incentivou em todos os momentos da minha vida.

## AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus por tudo em minha vida e minha saúde.

Agradeço à minha mãe, Conceição, e à minha família, por me apoiar todos esses anos e me ensinar que sem educação não se consegue nada.

Ao meu orientador, **Dr. Nelson Souto Rosa**, obrigado pela disponibilidade, compreensão, apoio e pelas ricas discussões e transferência de conhecimento que norteou esta dissertação de forma ética, profissional e, o mais importante, em um ambiente humano, buscando sempre o crescimento.

Meu co-orientador, **Dr. Robson Wagner Albuquerque de Medeiros**, por sua orientação, paciência, disposição, prestatividade e apoio.

Aos meus colegas pesquisadores do Centro de Informática, gostaria de agradecer a todos por suas contribuições diretas ou indiretas.

Gostaria de agradecer aos meus amigos de laboratório que me ajudaram muitas vezes durante meu programa de mestrado e tornaram meu dia mais agradável.

Gostaria de agradecer a todos os membros do grupo de pesquisa por suas ricas apresentações e discussões. Obrigado a todos os colegas do grupo **GFADS**.

Gostaria de agradecer aos **Professores do Centro de Informática da UFPE** por compartilharem todo o conhecimento que embasa minha formação de mestrado.

À **UFPE** pela infraestrutura para o desenvolvimento dessa dissertação.

"Nossa maior fraqueza está em desistir. A maneira mais certa de ter sucesso é sempre tentar mais uma vez." (THOMAS EDISON).

## RESUMO

A computação em nuvem mudou a forma como a TI funciona para a maioria das empresas que anteriormente utilizavam modelos baseados em compras de hardware e agora têm uma perspectiva de contrato de serviços especializados. A computação em nuvem tem permitido a implantação de aplicações com escalabilidade, flexibilidade, avanço tecnológico, maior segurança e alta disponibilidade. Aplicações escaláveis tem a capacidade de lidar com mudanças nas cargas de trabalho e na demanda de usuários, dimensionando automaticamente seus recursos computacionais. No entanto, fornecedores de nuvem não adotam uma padronização no desenvolvimento de seus produtos, dificultando a construção de soluções que fazem uso da migração de aplicações entre nuvens. Este trabalho apresenta uma solução que monitora despesas e extrai métricas de aplicações em nuvem, gerenciando a migração de aplicações entre múltiplas nuvens, levando em consideração o orçamento e os custos de execução, denominada ***COST*** *management of* ***A****daptive systems* (COSTA). COSTA realiza estimativas de custos das aplicações gerenciadas por ela, ajudando os usuários na fase inicial do planejamento e no processo de implantação das aplicações em nuvem. COSTA utiliza informações de custo para migrar aplicações baseadas em microsserviços entre diferentes provedores de nuvem. Para avaliar COSTA, foram conduzidos experimentos em que cenários de migração entre provedores de nuvem levassem à redução dos custos. Como resultado, COSTA reduziu efetivamente os custos financeiros das aplicações por meio de migrações entre provedores de nuvem, tirando proveito da diversificação existente dos esquemas de precificação dos provedores de nuvem.



**Palavras-chave**: sistemas adaptativos; gestão de custos; múltiplas nuvens; computação em nuvem.

## ABSTRACT

Cloud computing has changed how IT works for most companies that previously used models based on hardware purchases and now have a specialist services contract perspective. Cloud computing has allowed the deployment of applications with scalability, flexibility, technological advancement, greater security and high availability. Scalable applications can handle changes in workloads and user demand, automatically scaling their computing resources. However, cloud providers do not adopt standardization in developing their products, making it challenging to build solutions that use the migration of applications between clouds. This work presents a solution that monitors expenses and extracts metrics from cloud applications, managing the migration of applications between multiple clouds and considering the budget and execution costs, called COSTA - **COST** management of **A**daptative systems. COSTA performs cost estimates for the applications it manages, helping users in the initial planning phase and the deployment process of cloud applications. COSTA uses cost information to migrate applications between cloud providers based on microservices. To evaluate COSTA, experiments were conducted in which migration scenarios between cloud providers lead to cost reduction. As a result, COSTA has effectively lowered application finance costs through migrations between cloud providers, taking advantage of the existing diversification of cloud provider pricing schemes.



**Keywords**: adaptive systems; cost management; multi-cloud; cloud computing.

## LISTA DE FIGURAS

# LISTA DE CÓDIGOS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| **ABC** | *Artificial Bee Colony* |
| **API** | *Application Programming Interface* |
| **AWS** | *Amazon Web Services* |
| **AZ** | *Microsoft Azure* |
| **CAM** | *Cloud Automation Manager* |
| **COSTA** | ***COST** management of **A**daptive systems* |
| **CRIU** | *Checkpoint/Restore In Userspace* |
| **DevOps** | *Development and Operations* |
| **GCP** | *Google Cloud Platform* |
| **GKE** | *Google Kubernetes Engine* |
| **HTTP** | *Hypertext Transfer Protocol* |
| **IaaS** | *Infrastructure-as-a-Service* |
| **IaC** | *Infrastructure as Code* |
| **LLOOVIA** | *Load Level based OpimizatiOn for VIrtual machine Allocation* |
| **MAPE-K** | *Monitor, Analyzer, Planner, Executor, Knowledge* |
| **MBA** | *Microservice-Based Application* |
| **MODACLOUDS** | *MOdel-Driven Approach for the design and execution of applications on multiple Clouds* |
| **NIST** | *National Institute of Standards and Technology* |
| **OVMP** | *Optimal Virtual Machine Placement Algorithm* |
| **PaaS** | *Platform-as-a-Service* |
| **QoS** | *Quality of Service* |
| **REST** | *Representational State Transfer* |
| **SaaS** | *Software-as-a-Service* |
| **SIP** | *Stochastic Integer Programming* |

| | |
|---|---|
| **SRE** | *Site Reliability Engineering* |
| **VM** | *Virtual Machine* |

## SUMÁRIO

# 1 INTRODUÇÃO

A computação em nuvem tem possibilitado a criação de aplicações em larga escala. Além disso, outro benefício notável é a capacidade de tornar as aplicações em nuvem altamente disponíveis e flexíveis. Entretanto, tais vantagens também trazem consigo desafios na concepção de soluções que abordem os princípios de múltiplas nuvens, já que os provedores de nuvem não seguem um padrão uniforme na construção de seus serviços. Essa falta de padronização entre os serviços em nuvem cria dificuldades significativas na formulação de soluções que utilizam diversos provedores.

A utilização de uma abordagem de múltiplas nuvens pode ser benéfica em diversos contextos, como na redução do custo global da utilização desses serviços. Essa redução pode ocorrer quando o cliente consegue selecionar consistentemente o provedor que apresenta o menor custo, alinhando-se com as particularidades da sua aplicação. No entanto, fazer essa escolha pode ser um desafio considerável, pois os serviços de nuvem apresentam modelos de precificação que são complexos e diferem entre si, podendo variar entre estáticos ou dinâmicos (PORTELLA, 2021).

Os modelos de precificação estáticos oferecem aos usuários uma maior previsibilidade ao estimar os custos associados à utilização dos recursos computacionais durante o período contratado. Por outro lado, os modelos de precificação dinâmicas demandam um maior desafio para prever os custos finais do serviço contratado, devido às flutuações ao longo do tempo dos fatores de custo, denominados *Cost Drivers*. Inúmeros fatores podem influenciar na composição do preço mensal de um serviço de nuvem, *e.g.,* sistema operacional, localização geográfica dos servidores, forma de cobrança e período de utilização dos recursos (TRINDADE; SAWICKI, 2019). Além desses, podemos mencionar os fatores de custo como a transferência de dados de saída, armazenamento em disco e consultas para obtenção de dados em nuvem.

Na literatura, existem trabalhos que abordam a migração de contêineres ou máquinas virtuais entre diferentes provedores de nuvem (PELLEGRINI; ROTTMANN; STRIEDER, 2017; XAVIER; FERRAZ; SETTE, 2018; KEAHEY; RITEAU; TIMKOVICH, 2017; YAHYAPOUR, 2018; YAHYAPOUR, 2017). A motivação para essa migração surge das exigências dos recursos computacionais das aplicações e, em certos casos, da necessidade de lidar com o desafio clássico do *vendor lock-in* ou questões relacionadas ao desempenho. Entretanto, é importante notar que essas soluções não consideram os custos associados ao processo de migração entre plataformas de nuvem.

Para resolver tal problema, este provê uma solução chamada COSTA, que gerencia as migrações de *Microservice-Based Applications* (MBAs) entre diferentes provedores de nuvem, baseando-se em dados de custos. COSTA adota conceitos de sistemas adaptativos (IBM, 2005; LEMOS *et al.*, 2013; WEYNS, 2017; SALEHIE; TAHVILDARI, 2009) para monitoramento, análise e acionamento da migração das MBAs. Desta forma, a solução proposta tem como objetivo reduzir os gastos com nuvem explorando cenários de migração entre diferentes provedores de nuvem através de monitoramentos e análises nos fatores de custo das MBAs em execução.

## 1.1 MOTIVAÇÃO E JUSTIFICATIVA

Diante da diversidade de provedores de serviços em nuvem disponíveis no mercado e da ausência de padronização entre esses serviços, a tarefa de migrar MBAs ou VMs entre diferentes ambientes de nuvem apresenta-se como um desafio significativo para seus clientes. Os desafios se encontram no processo de instalação, manutenção e operação desses ambientes de várias nuvens. Para Buyya (2014), os desafios em um ambiente multinuvem incluem gerenciamento de rede (*e.g.,* física ou virtual), custos, seleção de planos, acordo de nível de serviço, segurança, autonomia, monitoramento dos recursos computacionais e portabilidade de dados.

Segundo Llorente (2011), a quantidade de provedores de nuvem vem aumentando em ritmo acelerado. Ao mesmo tempo, existe uma diversificação nos esquemas de preços adotados por cada proprietário de nuvem. No entanto, selecionar cuidadosamente o provedor de IaaS pode reduzir o custo (ASHWINI, 2012).

Para Trindade e Sawicki (2019), escolher os planos de serviços em nuvem é uma tarefa complexa. Isso é consequência do grande número de provedores de nuvem, somada à falta de transparência da política de preço dos provedores. Portanto, é de grande importância realizar uma análise cuidadosa nos regimes de preços antes de tomar qualquer decisão.

Diante disto, surge a necessidade de investigar os diferentes modelos de preços adotados pelos provedores de nuvem e como os usuários podem se beneficiar dessa característica por meio de migração de MBAs entre provedores de nuvem, levando em consideração os diferentes esquemas de precificação desses serviços.

## 1.2 OBJETIVOS

O objetivo principal deste trabalho é projetar e implementar uma solução para diminuir o custo de utilização de serviços de nuvem por meio de migração de MBAs em um ambiente de múltiplas nuvens. Para isso, deve-se levar em consideração os diferentes esquemas de precificação adotados pelos provedores desses serviços.

A fim de atingir o objetivo principal, o seguintes objetivos específicos foram definidos:

1. Realizar um estudo sobre os diferentes esquemas de preço adotados pelos provedores de nuvem;
2. Realizar um estudo sobre os conceitos de sistemas adaptativos na arquitetura de microsserviços;
3. Realizar um estudo sobre os tipos de migrações entre nuvens;
4. Definir uma arquitetura para a solução proposta baseada nos conceitos de sistemas adaptativos;
5. Implementar a proposta e seus cenários de simulação;
6. Avaliar a solução proposta nesta dissertação através de experimentos.

## 1.3 CONTRIBUIÇÃO

A contribuição principal desta dissertação é o desenvolvimento de uma solução para migração de MBAs entre diferentes infraestruturas de nuvens com o objetivo de minimizar os custos de utilização destes serviços. Para isto, foi proposto uma solução de migração de MBAs entre diferentes provedores de nuvem com base em informações de custo. Essa solução, denominada COSTA, adota conceitos de sistemas autoadaptativos *Monitor, Analyzer, Planner, Executor, Knowledge* (MAPE-K) para monitoramento, análise e acionamento das migrações de MBAs entre nuvens, tendo como principal objetivo a redução de custos.

COSTA reduz os custos pelo monitoramento e uso de técnicas de adaptação dinâmica para migrar MBAs entre vários provedores de serviços. Na verdade, COSTA escolhe o provedor de serviços de nuvem de menor custo para implantar as MBAs inicialmente, com base em seu

orçamento. Em seguida, procura oportunidades de reduzir custos enquanto verifica regularmente se há violação do orçamento. Quando surge uma oportunidade ou ocorre uma violação, COSTA transfere a *Microservice-Based Application* (MBA) para o provedor de serviços em nuvem recém-escolhido.

Por fim, COSTA gera estimativas de custos automaticamente entre diferentes provedores de nuvem e armazena o orçamento das MBAs para utilização em análises. Logo, foi necessário a construção de dois componentes para integrar a solução COSTA. O primeiro componente, chamado *Estimador de custo*, realiza estimativas de custos dos recursos computacionais necessários para executar uma MBA. O segundo componente, chamado *Elaborador de orçamento*, gerencia os orçamentos das MBAs.

## 1.4 ORGANIZAÇÃO DO TRABALHO

Os próximos capítulos estão organizados da seguinte forma:

- Capítulo 2 introduz os conceitos básicos sobre computação em nuvem, microsserviços, sistemas adaptativos e gestão de custos;
- Capítulo 3 descreve a solução COSTA, com apresentação do projeto e implementação;
- Capítulo 4 apresenta os experimentos realizados para a avaliação de COSTA;
- Capítulo 5 expõe os trabalhos relacionados e faz uma análise comparativa com COSTA;
- Capítulo 6 descreve as principais contribuições e limitações e sugere trabalhos futuros.

# 2 CONCEITOS BÁSICOS

Este capítulo introduz os conceitos necessários para entender o restante da dissertação. A Seção 2.1 apresenta os conceitos básicos de computação em nuvem. A Seção 2.2 fornece uma introdução aos conceitos de microsserviços. A Seção 2.3 apresenta os conceitos básicos de sistemas adaptativos, com ênfase na arquitetura MAPE-K. Por fim, a Seção 2.4 apresenta os conceitos de gestão de custos, com ênfase na computação em nuvem e seus sistemas de precificação.

## 2.1 COMPUTAÇÃO EM NUVEM

Os serviços em nuvem são caracterizados por um modelo de arquitetura distribuída para uso em larga escala de recursos computacionais. Esse modelo difere dos métodos tradicionais de computação, com a facilidade de provisionamento de recursos computacionais e o uso generalizado de virtualização para permitir o dimensionamento de aplicações em nuvem. A diferença também está no modelo de precificação adotado pelos provedores de serviços em nuvem, já que o pagamento é baseado no tempo de uso dos recursos disponíveis. Ou seja, o usuário de nuvem só paga pelo recurso computacional utilizado.

A definição do *National Institute of Standards and Technology* (NIST) sobre computação em nuvem é umas das mais aceitas e completas. Para Mell e Grance (2011), a computação em nuvem é um modelo para permitir acesso ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (*e.g.,* servidores, armazenamento, aplicações e serviços), que podem ser rapidamente provisionados e disponibilizados com o mínimo de esforço de gerenciamento ou de interação com o serviço.

Segundo Madadipouya (2015), a computação em nuvem é uma tecnologia emergente e sem precedentes que foi introduzida como a próxima geração de serviços sob demanda pela Internet. Este paradigma usa os conceitos de virtualização, conectividade, capacidade de processamento, armazenamento e compartilhamento para armazenar e compartilhar recursos de computação pela Internet.

Malathi (2011), por exemplo, estabelece que a computação em nuvem é um modelo para permitir o acesso conveniente a um conjunto compartilhado de recursos de computação configuráveis, que podem ser rapidamente provisionados e liberados com esforço mínimo de ge-

renciamento ou interação do provedor de serviços.

Para Sharma (2018), em vez de comprar e adicionar um novo *hardware* de armazenamento físico, é possível provisionar recursos pela Internet na nuvem, e isso ajuda a reduzir o custo de adição de novo armazenamento físico. Desta forma, torna-se possível armazenar e utilizar os serviços pela Internet.

De acordo com Armbrust (2010), a computação em nuvem refere-se tanto às aplicações fornecidas como serviços pela Internet quanto ao *hardware* e *software* de sistemas nos *data centers* que fornecem esses serviços. Os próprios serviços há muito são chamados de *Software-as-a-Service* (SaaS). Alguns fornecedores usam termos como *Infrastructure-as-a-Service* (IaaS) e *Platform-as-a-Service* (PaaS) para descrever seus produtos.

### 2.1.1 Principais características

A computação em nuvem fornece características importantes para compartilhar e provisionar recursos de infraestrutura de computação (MALATHI, 2011):

- **Autoatendimento sob demanda**: o consumidor de nuvem sempre pode solicitar mais recursos computacionais, como processamento e espaço de armazenamento, sem qualquer comunicação entre o consumidor e o provedor de serviços de nuvem. Portanto, essa eliminação da interação humana leva a uma maior eficiência e economia de custos para usuários e fornecedores de serviços de nuvem;
- **Amplo acesso à rede**: os recursos computacionais são acessados via Internet por diferentes dispositivos, *e.g., laptops*, impressoras, telefones celulares;
- ***Pool* de recursos independente de localização**: os recursos computacionais do provedor são combinados para atender a vários consumidores usando um modelo *multi-tenant* (ou multi-inquilino). Vários recursos físicos e virtuais são alocados e realocados dinamicamente com base na demanda do consumidor. Há um senso de independência de localização, em que o cliente normalmente não tem localização precisa ou conhecimento dos recursos entregues, mas pode especificar a localização em um nível mais alto de abstração (*e.g.,* país, estado ou *datacenter*). Exemplos de recursos incluem armazenamento, processamento, memória, largura de banda e máquinas virtuais;

- **Serviço medido**: os sistemas em nuvem gerenciam e otimizam automaticamente o uso de recursos usando medição no nível apropriado de abstração para o tipo de serviço, *e.g.,* armazenamento, processamento, largura de banda e contas de usuário ativas. O uso de recursos pode ser monitorado e gerenciado, o que traz transparência tanto para o provedor do serviço que está sendo usado quanto para o consumidor;

- **Elasticidade rápida**: os recursos da nuvem podem ser rapidamente reservados e liberados de acordo com os consumidores. Os recursos disponíveis para consumo podem ser facilmente dimensionados para qualquer quantidade a qualquer momento.

O compartilhamento de recursos de computação para obter eficiência no uso da infraestrutura fornecida é uma característica cada vez mais comum na computação em nuvem, porque os provedores estão interessados em aumentar seus lucros e ter um maior aproveitamento energético. Por outro lado, os consumidores estão procurando maneiras de reduzir os custos da nuvem.

### 2.1.2 Modelos de Serviços

Os produtos oferecidos pelos serviços em nuvem são classificados de acordo com o nível de abstração da infraestrutura da seguinte forma (MALATHI, 2011):

- **Software como Serviço (*Software-as-a-Service* ou SaaS)**: a opção oferecida ao consumidor é utilizar a aplicação do provedor de serviços que executa na infraestrutura de nuvem. As aplicações estão disponíveis em vários dispositivos clientes por meio de uma interface simplificada, como um navegador *Web*. O consumidor não gerencia ou controla a infraestrutura de nuvem subjacente. As vantagens de usar SaaS são inicialização rápida, manutenção e atualizações feitas pelo provedor. Os riscos, entretanto, incluem personalização mínima, integração de dados, riscos à segurança e privacidade, não gerenciamento de atualização e estratégia de saída;

- **Plataforma como Serviço (*Platform-as-a-Service* ou PaaS)**: a função fornecida ao consumidor é a implantação de aplicação de infraestrutura em nuvem, criada ou adquirida pelo consumidor, construída usando linguagens e ferramentas de programação suportadas pelo provedor. As ofertas de fornecedores de serviços PaaS incluem hospedagem completa de aplicação, desenvolvimento, teste e ambiente de implantação, bem

como serviços integrados extensivos, que podem incluir escalabilidade, manutenção e controle de versão. A vantagem de usar PaaS é o fornecimento de uma infraestrutura de rede flexível, escalável e segura. No entanto, os riscos incluem estratégia de saída, modelo de preços e problemas de atualização;

- **Infraestrutura como Serviço (*Infrastructure-as-a-Service* ou IaaS)**: a capacidade fornecida ao consumidor é prover processamento, armazenamento, rede e outros recursos básicos de computação, a parti dos quais o consumidor pode implantar e usar qualquer *software*, que pode incluir um sistema operacional e uma aplicação. As vantagens deste tipo de serviço incluem a escalabilidade e a inicialização rápida. Por sua vez, os riscos estão relacionados a modelo de preços, potencial bloqueio, segurança e privacidade.

Independentemente do modelo de serviço utilizado, é importante que prestadores de serviços de nuvem e usuários acompanhem e concordem quanto à disponibilidade, segurança, desempenho e qualidade dos serviços oferecidos.

### 2.1.3 Modelo de Implantação

O modelo de serviços de nuvem pode ser classificado como:

- **Nuvem Pública**: a nuvem pública, ou nuvem externa, descreve a computação em nuvem em que os recursos são fornecidos dinamicamente em uma base de autoatendimento e prestados pela Internet, por meio de serviços Web, por um provedor de serviços externo que cobra pelo uso exato dos recursos computacionais;

- **Nuvem Comunitária**: uma nuvem comunitária pode ser configurada quando várias organizações têm requisitos semelhantes, desejam compartilhar a infraestrutura e os benefícios da computação em nuvem. Como os custos são distribuídos por menos usuários do que na nuvem pública, essa opção é mais cara, mas pode oferecer um nível mais alto de privacidade, segurança e/ou conformidade regulatória;

- **Nuvem Híbrida**: os recursos de nuvem híbrida são a combinação de duas ou mais nuvens (*e.g.,* privada, comunitária ou pública) para fornecer um único serviço comum;

- **Nuvem Privada**: os recursos de nuvem privada são criados individualmente para organizações que atendem a vários consumidores, como empresas comerciais. Esse tipo de

serviço em nuvem pode ser gerenciado e usado pela própria organização, por terceiros ou por uma combinação de ambos.

A utilização de nuvens públicas e híbridas favorece a implementação do conceito de *cloud bursting*, que é uma expansão temporária da capacidade reservada de um serviço com demanda inesperada. Portanto, aplicar o conceito de *cloud bursting* oferece as vantagens de escalabilidade, uso eficiente de recursos e melhor disponibilidade das aplicações.

## 2.2 MICROSSERVIÇOS

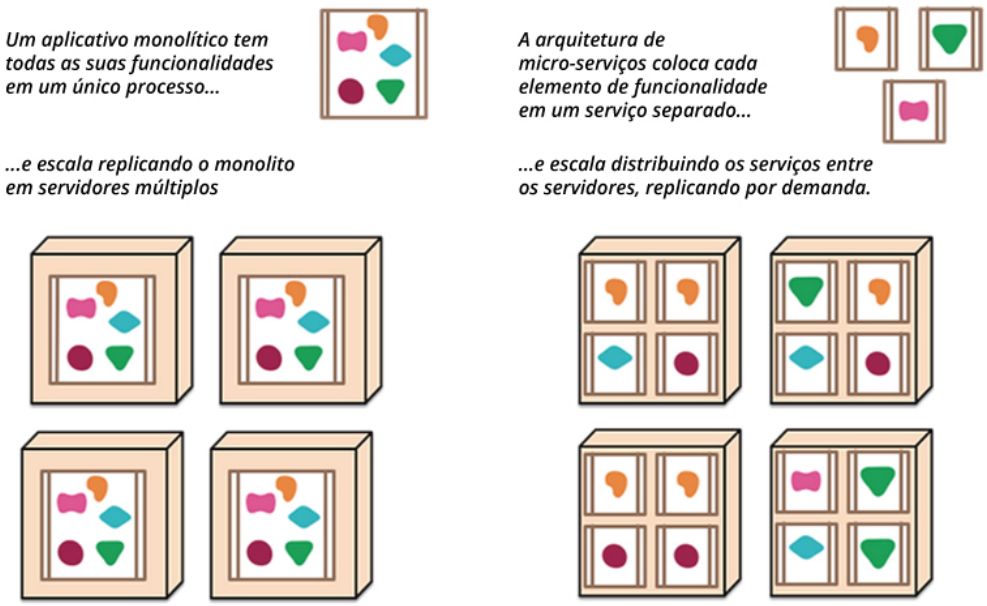
O estilo arquitetural de microsserviços é uma forma de construir aplicações como uma coleção de pequenos serviços, cada um executando seu próprio processo e interagindo através de mecanismos leves, geralmente uma *Application Programming Interface* (API) REST que faz uso de protocolo *Hypertext Transfer Protocol* (HTTP) (LEWIS; FOWLER, 2014). Microsserviços são construídos em torno de recursos de negócios e implantados de forma autônoma usando mecanismos de implantação totalmente automatizados. O gerenciamento centralizado desses serviços é mínimo e eles podem ser implementados em diferentes linguagens de programação e usar diferentes tecnologias de armazenamento de dados.

Para Lewis e Fowler (2014), o modelo de arquitetura de microsserviços difere do modelo de arquitetura tradicional, também conhecido como arquitetura monolítica. A Figura 1 mostra a diferença entre aplicação monolítica e uma MBA.

Uma aplicação monolítica é construída como uma única unidade composta de três partes: a interface do usuário, um banco de dados e uma aplicação do lado do servidor. No entanto, os microsserviços são independentes, podem ser implantados de forma independente e têm responsabilidades bem definidas, *e.g.,* um microsserviço não possui múltiplas responsabilidades.

Existem várias vantagens em usar microsserviços. Primeiramente, a aplicação é desenvolvida a partir de um conjunto de microsserviços, o que permite que cada módulo seja construído utilizando diferentes tecnologias. Em segundo lugar, cada microsserviço é responsável por implementar um requisito do sistema. Quando ocorre uma falha em um microsserviço, ela afeta apenas parte da aplicação. Finalmente, usando uma arquitetura de microsserviços, é possível aplicar escalabilidade individualmente a um componente que requer uma carga de trabalho maior. Como os microsserviços são desacoplados, você pode dimensionar o serviço para atender aos seus requisitos em tempo de execução.

Figura 1 – Monolitos e Microsserviços




**Fonte:** Lewis e Fowler (2014)

Por fim, o uso de diferentes tecnologias não é recomendado para aplicações monolíticas e, quando ocorrem falhas, isso afeta todo o sistema. Além disso, não é possível dimensionar parte do sistema porque ele é construído em um bloco. Por conta disto, os microsserviços se tornaram muito populares para criar aplicações complexas baseadas em nuvem.

### 2.2.1 Características de uma Arquitetura de Microsserviços

Daschner Daschner (2017) define um microsserviço como um sistema composto por várias aplicações com interfaces bem definidas. As interfaces devem ser tecnicamente agnósticas, com menos operações, coesas e bem delimitadas. A ideia por trás dessa arquitetura consiste em várias aplicações interdependentes, cada uma executando suas tarefas individuais para resolver um problema. No entanto, não existe uma definição sobre o tamanho de um microsserviço.

Não existe uma definição formal do estilo arquitetural de microsserviços, mas há um conjunto de características que ajudam a entendê-lo (LEWIS; FOWLER, 2014):

- **Componentização via serviços**: o componente é uma unidade de *software* que pode ser substituída e atualizada independentemente;
- **Organizado em torno de capacidades de negócios**: os microsserviços são organiza-

dos em torno de recursos de negócios. Esses serviços exigem *software* de pilha completa, incluindo interface do usuário, armazenamento persistente e todos os recursos externos de colaboração;

- **Produtos e não projetos**: a equipe é responsável pelo produto durante todo o seu ciclo de vida. Isso significa que a equipe de desenvolvimento assume total responsabilidade pelo *software* no ambiente de produção;
- ***Endpoints* inteligentes e *pipes* burros**: os dois protocolos mais comumente usados são *Request-Reply* via HTTP com APIs de recursos e mensagens leves;
- **Governança descentralizada**: as equipes são responsáveis por todos os aspectos do *software* que constroem, incluindo a operação do *software* 24 horas por dia, 7 dias por semana;
- **Gerenciamento de dados descentralizado**: cada serviço gerencia seu próprio banco de dados, sejam instâncias diferentes da mesma tecnologia de banco de dados ou sistemas de banco de dados completamente diferentes;
- **Automação de infraestrutura**: muitos produtos ou sistemas implementados com microsserviços estão sendo construídos por equipes com vasta experiência em entrega contínua. As equipes que criam *software* dessa maneira fazem uso extensivo de técnicas de automação de infraestrutura;
- **Projetado para falha**: uma consequência do uso de serviços como componentes é que as aplicações precisam ser projetadas para que possam tolerar a falha dos serviços;
- ***Design* evolutivo**: os desenvolvedores de microsserviços geralmente têm experiência em *design* evolutivo e veem a decomposição como uma ferramenta adicional para permitir que os desenvolvedores controlem as alterações em suas aplicações sem retardar a mudança.

Na construção de microsserviços, cada serviço pode ter suas particularidades, *e.g.,* mudar, evoluir, fracassar, exibir lentidão ou ser retirado em qualquer momento. No entanto, esses problemas não devem afetar o comportamento geral do sistema, pois o sistema deve ser capaz de aceitar alterações e lidar com falhas quando estas ocorrerem. Portanto, as seguintes propriedades são necessárias para garantir esse comportamento (ESCOFFIER, 2017):

- **Autônomo**: os microsserviços podem se adaptar à indisponibilidade dos serviços com os quais interagem;
- **Assíncrono**: a passagem assíncrona de mensagens permite que os microsserviços sofram falhas e se adaptem a comportamentos inesperados. Portanto, o dano não se propaga pelo sistema;
- **Resiliência**: como a resiliência trata da autocorreção e do tratamento de falhas, os microsserviços devem implementar uma estratégia de recuperação ou compensação em caso de falha;
- **Elasticidade**: os microsserviços devem ser flexíveis para acomodar o número de instâncias para suportar a carga de trabalho. Isso implica em um conjunto de restrições, como guardar sessão da aplicação em memória, compartilhar o estado da aplicação entre as instâncias, se necessário, ou ser capaz de passar mensagens para a mesma instância.

As vantagens e característica inerentes à arquitetura baseada em microsserviços são as seguintes (CARVALHO, 2016):

- Componentização via pequenos serviços independentes com limites e responsabilidades bem definidas;
- Serviços divididos em pequenos times por capacidades de negócios e estes são responsáveis por todas as camadas de tecnologia;
- Serviços pequenos que possuem um código menor são mais fáceis de se desenvolver, de manter e de serem entendidos por novos membros do time;
- Serviços autônomos que podem ser alterados e implantados rapidamente e independentemente de outros serviços;
- Conceito de produto ao invés de projeto. Portanto, times se tornam responsáveis por todo o ciclo de vida do produto;
- Utilização de protocolos simples síncronos ou assíncronos utilizados na Web: HTTP;
- Serviços que podem ser construídos utilizando tecnologias diferentes, de acordo com as necessidades de cada um deles;

- Adoção ou mudança para novas tecnologias ou melhores implementações são mais fáceis de administrar;
- Isolamento de falhas e resiliência;
- Descentralização de banco de dados; e
- Cada serviço pode ser escalado independentemente.

Por fim, a arquitetura de microsserviços não é uma "bala de prata", pois o seu uso deve ser feito com planejamento. Sua utilização aumenta a complexidade das regras de negócios e desenvolvimento. Portanto, conhecer as vantagens e desvantagens do estilo arquitetural de microsserviços pode ajudar a tomar as decisões certas nos estágios iniciais do projeto.

## 2.3 SISTEMAS ADAPTATIVOS

Em um sistema adaptativo típico (WILEY, 2021), um componente de gerenciamento supervisiona um elemento gerenciado (*e.g.,* um MBA) em tempo de execução para mantê-lo funcionando corretamente. Nesse contexto, execução adequada significa que o elemento gerenciado possui metas que devem ser satisfeitas durante a execução. Por exemplo, dada uma meta de desempenho de uso do processador abaixo de 90%, a supervisão garante que a parte gerenciada sempre atenda à meta. Se a meta for violada, o componente gerenciador age, *e.g.,* criando uma réplica do componente gerenciado para colocar o elemento gerenciado de volta à meta.

Adaptação é uma ação de mudança da estrutura ou função de um organismo para melhor se adaptar ao seu ambiente (DA; DALMAU; ROOSE, 2011). Os sistemas adaptativos são, portanto, capazes de se adaptar às mudanças que ocorrem no ambiente de sua implementação, a fim de continuar a atingir seus objetivos (WEYNS *et al.*, 2013). Para Salehie e Tahvildari (2009), essa adaptação decorre da complexidade de gerenciar a heterogeneidade existente entre os componentes que compõem uma aplicação, tratar erros e encontrar elasticidade para lidar com mudanças contínuas de metas/requisitos.

Para Huebscher e McCann (2008) as soluções adaptativas possuem quatro propriedades principais:

- Auto-organização: possibilita que o sistema adaptativo se auto-organize de acordo com objetivos de alto nível definidos;

- Auto-otimização: permite ao sistema a busca por oportunidades para melhorar o desempenho e a eficiência;
- Autocorreção: fornece soluções adaptativas com a capacidade de detectar e diagnosticar erros;
- Autoproteção: possibilita ao sistema autônomo se proteger de ataques mal-intencionados ou evita falhas em cascata.

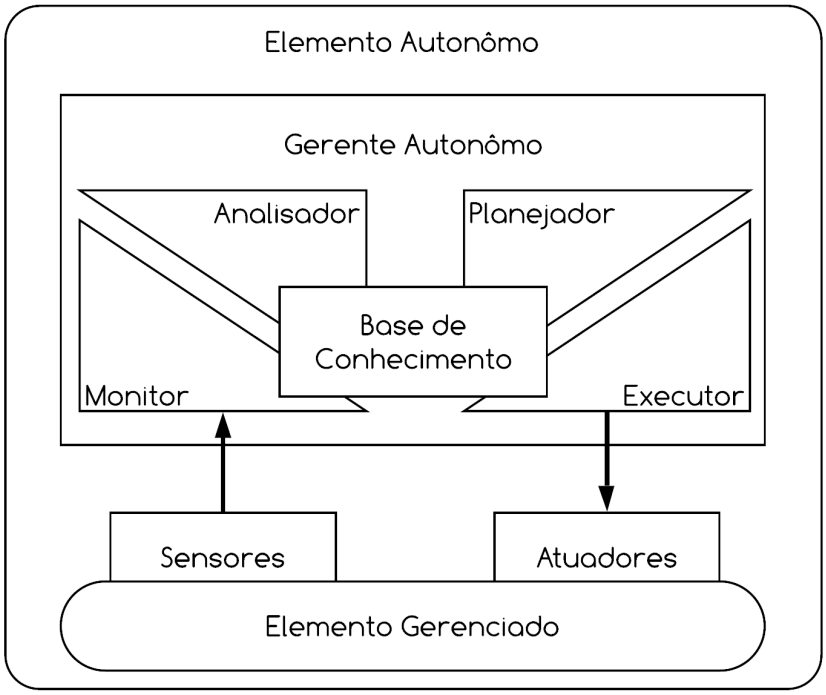
Por fim, Salehie e Tahvildari (2009) apresentam as seis perguntas para identificar os requisitos de um sistema adaptativo:

- Onde a adaptação ocorre? Esta pergunta está associada à localização do problema que será resolvido pela adaptação.
- Quando a adaptação ocorre? Esta pergunta é sobre os aspectos temporais das soluções adaptativas.
- O que é adaptado? Esta pergunta identifica as características que podem ser alteradas pelo comportamento adaptativo e quais características devem mudar em cada situação.
- Por que é adaptado? Esta pergunta determina a motivação para criar sistemas adaptativos.
- Como é a adaptação? Esta pergunta define como as soluções adaptativas mudam os sistemas e as ações usadas dependendo da situação.
- Quem adapta? Este ponto discute o nível de automação e intervenção humana.

### 2.3.1 MAPE-K

O MAPE-K (IBM, 2005) é uma arquitetura modular para o desenvolvimento de sistemas autônomos. Ela usa o princípio de separação de interesses para lidar com comportamentos dinâmicos de ambientes de execução e aplicações. Os módulos do MAPE-K são responsáveis por monitorar, analisar e tomar decisões sobre a necessidade de adaptação, criar planos de adaptação e executá-los. A Figura 2 apresenta os componentes do MAPE-K, cujas funções são descritas a seguir (HUEBSCHER; MCCANN, 2008).

Figura 2 – Arquitetura MAPE-K




**Fonte:** Huebscher e McCann (2008)

- **Elemento Gerenciado**: software que atua em tempo de execução e que é gerenciado pelo *Gerente Autônomo*.
- **Monitor**: a coleta de dados do *Elemento Gerenciado* é através de sensores, como taxa de transferência de rede, CPU e uso de memória. Existe uma grande diversidade de dados coletados e utilizados no processo de adaptação. Portanto, a natureza dos dados coletados depende do domínio da aplicação e do objetivo da adaptação.
- **Analisador**: este componente implementa mecanismos para examinar os dados coletados pelo *Monitor* e identificar se uma adaptação é necessária ou não. Quando uma adaptação é necessária, o *Analisador* gera uma solicitação de mudança ao *Planejador*. As adaptações devem sempre ocorrer quando um determinado objetivo do sistema é violado. Por exemplo, um servidor que deve processar 100 requisições/segundo (meta do servidor) deve sofrer uma intervenção (por exemplo, melhorar seus recursos) quando o monitoramento detectar que apenas 80 requisições/segundo são processadas.
- **Planejador**: o *Planejador* cria uma estratégia para realizar uma adaptação no *Elemento*

*Gerenciado*. O *Planejador* gera todos os passos necessários na adaptação e passa os comandos de alto nível para o *Executor*. Portanto, essa função pode assumir várias formas, desde um único comando até um fluxo de trabalho mais complexo.

- **Executor**: o *Executor* possui todos os mecanismos para agendar e fazer as alterações necessárias no *Elemento Gerenciado* por meio de *Atuadores*. Quando um plano de adaptação é gerado, ações são tomadas para modificar o estado de um ou mais recursos gerenciados. Na prática, comandos de alto nível gerados pelo *Planejador* são executados em baixo nível pelo *Executor*, por meio de uma série de ações.
- **Base de Conhecimento**: a *Base de Conhecimento* armazena as informações geradas e utilizadas pelos quatro componentes do *Gerente Autônomo* (Monitor, Analisador, Planejador e Executor).

## 2.4 GESTÃO DE CUSTOS

O gerenciamento de custos de nuvem envolve entender e gerenciar os custos e requisitos associados à tecnologia de nuvem. Isso significa encontrar maneiras econômicas de maximizar o uso e a eficiência da nuvem. Gerenciar os custos da nuvem é desafiador porque os custos são distribuídos e variáveis, dificultando o seu entendimento. No entanto, a implementação de uma estratégia de gerenciamento de custos de nuvem pode ajudar as organizações a analisar os custos, o uso, o *benchmarking* e o planejamento da computação em nuvem. Além disso, ao entender melhor o custo e o uso dos recursos computacionais, as empresas podem gerenciar melhor os registros contábeis e melhorar o desempenho e o rendimento das tecnologias de nuvem. Os custos da nuvem incluem máquinas virtuais, memória, armazenamento, tráfego de rede, suporte e licenças de *software*.

Em um ambiente de nuvem, os fatores de custo que normalmente causam aumento nas despesas incluem armazenamento, poder de processamento (*e.g.,* recursos como núcleos virtuais), bancos de dados e rede, *e.g.,* transferência de dados de saída da *Virtual Machine* (VM) para a Internet (TRINDADE; SAWICKI, 2019). Enquanto isso, os provedores de nuvem podem oferecer descontos que variam de acordo com o consumo de recursos computacionais para incentivar o uso dos serviços e se manterem competitivos no mercado. Portanto, selecionar o provedor de nuvem certo é fundamental antes de mover as aplicações para um ambiente de nuvem. Conhecer os recursos computacionais necessários para a execução de uma aplicação

ajudará a evitar gastos desnecessários. Além disso, os provedores de nuvem podem oferecer descontos que variam de acordo com o consumo para incentivar o uso de seus serviços.

Por fim, reduzir os recursos tecnológicos significa encontrar otimizações e identificar o uso excessivo sem impactar a infraestrutura da empresa. A chave para reduzir o uso excessivo e gerenciar custos com sucesso é monitorar e auditar cuidadosamente seu uso para encontrar itens subutilizados. A melhor maneira de atingir esse objetivo é considerar a infraestrutura de computação distribuída e as ferramentas analíticas incorporadas aos ambientes de nuvem.

### 2.4.1 Política de Precificação

Para Portella (2021), o objetivo da política de precificação da nuvem é definir padrões para o fornecimento de recursos de infraestrutura entre provedores de serviços em nuvem e usuários. Do lado do provedor de nuvem, essa política se alinha com o objetivo de aumentar a lucratividade e a eficiência no uso dos recursos de computação para evitar o tempo de inatividade. Para os usuários, essas políticas fornecem estratégias para usar recursos de computação (*e.g.,* ativos de infraestrutura e poder de computação) e custos. Logo, a política de precificação da nuvem define as regras para compra e venda de recursos de infraestrutura entre provedores de serviços e usuários.

### 2.4.2 Tipos de Precificação

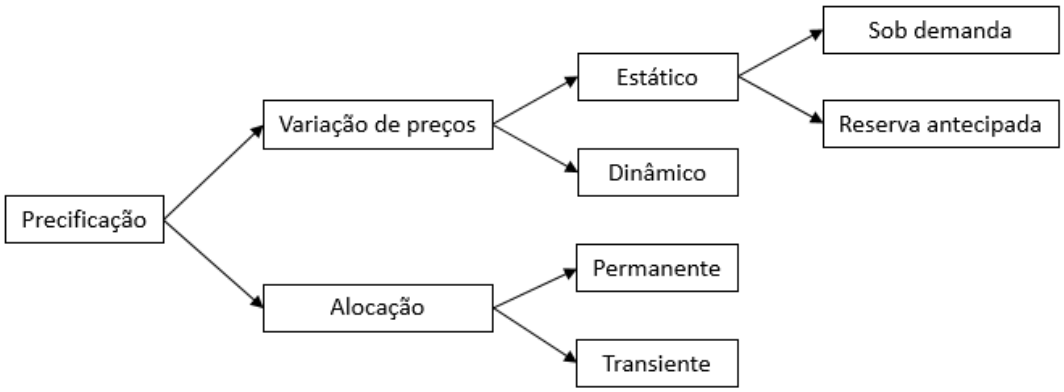
Os provedores de nuvem usam modelos de preços estáticos e dinâmicos para cobrar pelo uso de serviços/recursos (PORTELLA, 2021). No modelo estático, o provedor determina o preço a ser pago de forma fixa em função do tempo de uso da infraestrutura. Então, o pagamento ocorre após o consumo dos recursos computacionais. Este plano é conhecido como modelo de custo sob demanda (*on-demand*). Uma variação do modelo estático é a reserva avançada de recursos. Nesse caso, ocorre uma cobrança mesmo que o usuário não utilize os recursos computacionais reservados.

Na precificação dinâmica, os provedores oferecem recursos de infraestrutura a um custo variável, dependendo do preço oferecido pelos usuários. Assim, os usuários que pagam mais do que os demais têm maior probabilidade de contratar o serviço. Enquanto isso, os consumidores podem ter preços melhores em cenários de baixa demanda do que os praticados no modelo estático. Na precificação dinâmica, modelos de mercado baseados em leilões, teorias

econômicas e de jogos, técnicas de aprendizado de máquina e simulações são utilizados para a intermediação de contratos (NARAHARI; RAJU; RAVIKUMAR, 2005).

Os modelos de precificação ainda podem ser classificados como permanentes ou transientes, dada a probabilidade de interrupção da instância por parte do fornecedor de nuvem. A Figura 3 apresenta a classificação da precificação de nuvem.

Figura 3 – Tipos de precificação de nuvem




**Fonte:** Portella (2021)

A precificação dinâmica é praticada pelos provedores de nuvem ofertando instâncias *spot* (AMAZON EC2, 2023) e oferecendo instâncias preemptíveis (GOOGLE CLOUD., 2023), de modo a aproveitar a infraestrutura não utilizada. Ambos os modelos de precificação dinâmica podem ser classificados como modelos de precificação de instâncias transientes. O conceito transiente se refere à possibilidade do provedor poder interromper o fornecimento do serviço do consumidor, propiciando atendimento a requisições com maior prioridade ou que venham a proporcionar um maior lucro (SHASTRI; RIZK; IRWIN, 2016).

Independentemente da estratégia adotada, aplica-se o conceito de *pay-as-you-go*, pois o consumidor paga pelo tempo de uso de todo o recurso computacional de forma proporcional. Portanto, essa característica da computação em nuvem permite que as organizações que contratam serviços possam direcionar investimentos para o seu próprio negócio (ARMBRUST, 2010).

Com relação ao preço, a categoria VM sob demanda é oferecida por provedores de nuvem pública, por exemplo, Amazon Web Services *on demand* (AMAZON EC2 ON DEMAND, 2023), Microsoft Azure *pay-as-you-go* (MICROSOFT AZURE., 2023) e Google Cloud Platform *predefined* (GOOGLE CLOUD COMPUTE ENGINE, 2023). A categoria reservada é fornecida pelos provedores de nuvem pública como Amazon EC2 *reserved* (AMAZON EC2 RESERVED, 2023),

Google Cloud *reserved* (GOOGLE CLOUD COMPUTE ENGINE, 2023) e Microsoft Azure *reserved instances* (MICROSOFT AZURE RESERVED, 2023). VMs com reservas avançadas estabelecem um compromisso entre os usuários e o provedor de nuvem. O usuário concorda em pagar pela VM por um período fixo (*e.g.,* um ou três anos), independentemente de a VM ser usada ou não. Como compensação, o usuário recebe um desconto significativo do provedor de nuvem.

## 2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os conceitos e o referencial teórico relacionados a esta pesquisa. Na computação em nuvem são discutidos os conceitos e modelos. Em relação a microsserviços, foram apresentadas suas características e um comparativo com o modelo de arquitetura monolítica. Na área de sistemas adaptativos, foram detalhados seus conceitos e os elementos do MAPE-K. Por fim, discutiu-se o conceito de gestão de custos com foco na computação em nuvem.

# 3 COSTA

Este capítulo está organizado em 3 seções para apresentação de COSTA. A Seção 3.1 traz uma visão geral de cada componente que compõem COSTA. A Seção 3.2 apresenta os requisitos e arquitetura da solução COSTA, descrevendo as interações dos componentes. A Seção 3.3 detalha o projeto e sua implementação, apresentando as respectivas tecnologias (ferramentas externas) que integram COSTA.

## 3.1 VISÃO GERAL

COSTA[1] é um sistema de gestão de adaptação para MBAs em nuvem, que emprega metas de custo no processo adaptativo. Ao monitorar os custos relacionados à MBA e aos provedores de nuvem que hospedam a aplicação, COSTA tem a capacidade de reduzir os gastos das empresas junto aos provedores. A abordagem apresentada nesta dissertação possibilita a migração de MBAs entre diferentes provedores de nuvem, a fim de tirar proveito da diversidade nos esquemas de preços disponíveis. Na prática, depois de estabelecer um orçamento inicial, isto é, uma meta, são realizadas estimativas de custo para a implantação da MBA nas várias nuvens. COSTA pode então realizar a migração da MBA entre os provedores de nuvem para evitar a extrapolação do orçamento ou ao identificar oportunidades para redução de custos.

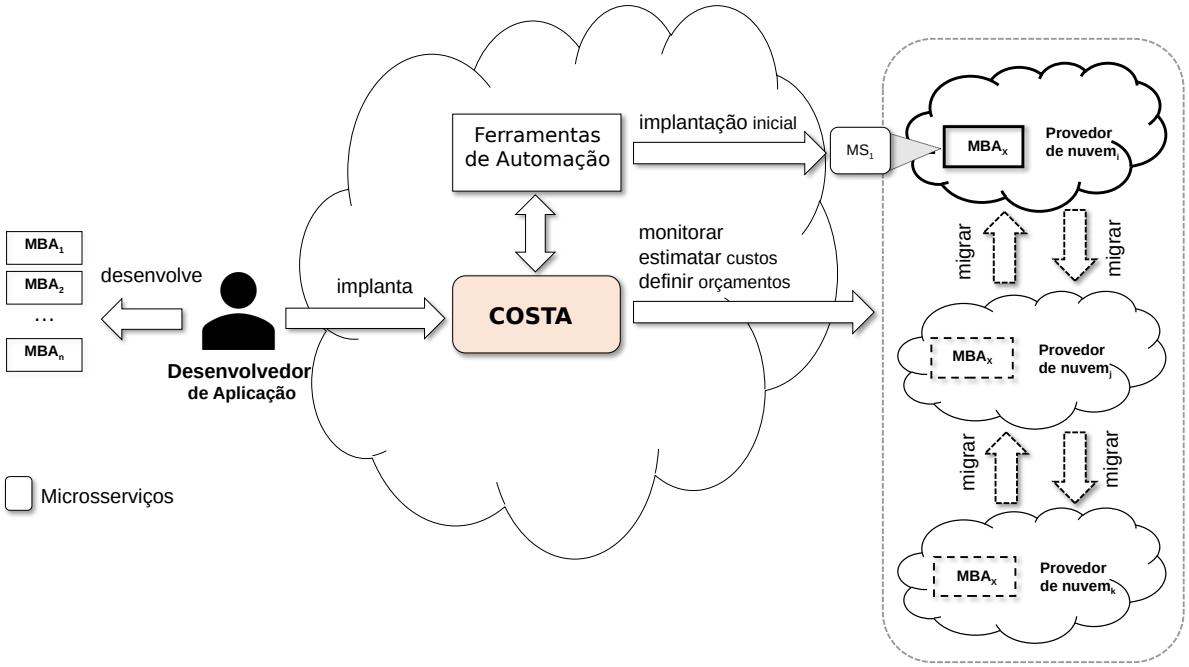
Na construção da solução apresentada neste trabalho, foram empregados os conceitos de sistemas adaptativos, especificamente o MAPE-K, devido à sua flexibilidade e clara divisão de responsabilidades entre os elementos constituintes. Isso viabiliza a incorporação de novos componentes conforme necessário. COSTA foi concebido com base no modelo arquitetural de microsserviços, uma vez que é composto por um conjunto de serviços independentes, cada um com um escopo bem definido e focado em uma única função.

A Figura 4 apresenta uma visão geral de COSTA. O *Desenvolvedor de Aplicação* implementa a aplicação a ser gerenciada e depois a implanta em COSTA. Além disso, o *Desenvolvedor de Aplicação* irá realizar uma estimativa de custo entre as nuvens para implantar a aplicação no provedor de nuvem mais barato. Em seguida, COSTA interage com as *Ferramentas de Automação* existentes para iniciar o processo de implantação da aplicação e migrar quando necessário. Após a implantação da MBA, inicia-se o monitoramento de custos e são realizadas análises para evitar violações do orçamento definido ou reduzir o custo de execução.

[1] https://github.com/lcs10-ufpe/costa

Quando uma violação de custo é detectada, a MBA é migrada para um novo provedor de nuvem. Vale observar que as migrações ocorrem a qualquer momento após a implantação inicial.

Figura 4 – Visão geral da solução COSTA




**Fonte:** Elaborada pelo autor (2023).

## 3.2 REQUISITOS E ARQUITETURA

Os requisitos de COSTA são motivados pela necessidade de reduzir os custos de MBAs em nuvem. Isso porque escolher o prestador de serviço certo é essencial para reduzir custos (ASHWINI, 2012). COSTA é uma solução gerenciadora de adaptação que utiliza metas de custos para migrar MBAs entre nuvens. Portanto, é necessário utilizar os conceitos de sistemas adaptativos (MOURA *et al.*, 2002). Um ambiente adaptativo pode detectar possíveis violações analisando os dados coletados por meio do monitoramento e, em seguida, gerar um plano de adaptação que inclua ações para resolver as violações detectadas.

Os microsserviços de COSTA devem ser capazes de Monitorar, Analisar, Planejar e Executar suas devidas tarefas. O Monitor será responsável pela coleta de métricas de MBAs em nuvem; o Analisador receberá as informações do Monitor para realização das análises e, caso seja necessário, migrar a MBA; o Planejador receberá uma solicitação do Analisador para a geração do plano de adaptação. Em seguida, os planos gerados deverão ser executados pelo Executor.

Outros requisitos, como definição de orçamento e geração de estimativas de custos, foram levados em consideração para ajudar na fase de planejamento das MBAs, *e.g.,* antes de implantá-las em nuvem. Portanto, se faz necessário a criação de dois novos microsserviços, Elaborador de orçamento e Estimador de custo, para compor COSTA.

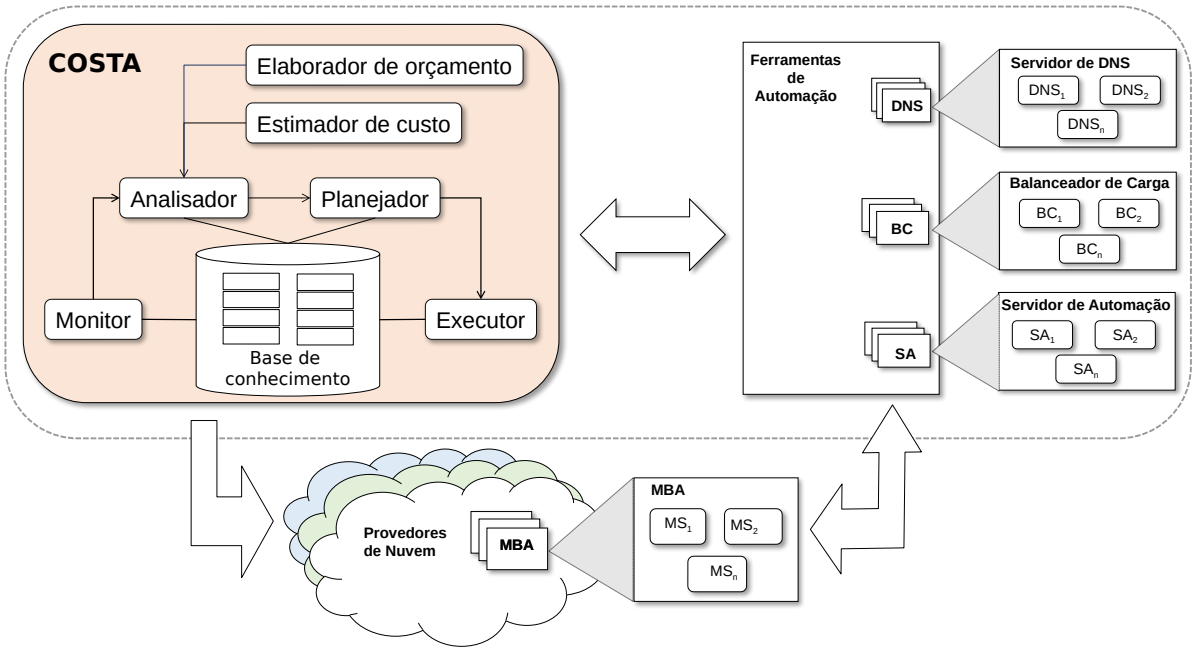
Desta forma, foram definidos os seguintes requisitos:

- **R1**: Monitorar os custos de execução em todos os provedores de nuvem onde as MBAs estão em execução;
- **R2**: Extrair métricas sobre recursos de nuvem usados pelas MBAs que influenciem diretamente nos fatores de custos. Por exemplo, armazenamento de dados em disco, tempo de execução da máquina virtual, transferência de dados de saída e troca de mensagens;
- **R3**: Monitorar o estado das MBAs nas nuvens integradas à COSTA para verificar se está ocorrendo alguma indisponibilidade do serviço de nuvem;
- **R4**: Gerenciar (*e.g.,* gerar, remover e armazenar) o orçamento das MBAs e notificar as partes interessadas em caso de violação desses orçamentos;
- **R5**: Gerar estimativas de custos onde as MBAs podem ser executadas;
- **R6**: Analisar os dados monitorados e decidir sobre a necessidade de migrar a aplicação para outro provedor de nuvem com base em violações de custos ou cenários de custos mais baixos;
- **R7**: Analisar os dados monitorados e decidir sobre a necessidade de migrar a MBA para outro provedor de nuvem com base nas informações de custos mais baixos em caso de indisponibilidade do serviço de nuvem;
- **R8**: Criar planos de implantação, remoção e migração das MBAs;
- **R9**: Executar implantação, remoção e migração das MBAs.

Com base nesses requisitos, a arquitetura de COSTA foi definida, conforme mostra a Figura 5. Em seguida, a arquitetura foi implementada por um conjunto de microsserviços que reforçam os elementos MAPE-K (ver Seção 2.3.1). Para atender os requisitos **R4** e **R5**, foi necessário criar os microsserviços Elaborador de orçamento e Estimador de custo. Na prática, esses microsserviços são responsáveis por monitorar os provedores de nuvem (Monitor) e analisar

os dados extraídos dos recursos computacionais utilizados pelas MBAs para verificar se houve violação do orçamento estabelecido inicialmente (Analisador). Além disso, os dados analisados podem evidenciar cenários de redução de custos com nuvem em que a MBA se encontra em execução. Planos de adaptação (Planejador) entre nuvens são criados e executados (Executor) com ajuda da *Ferramentas de Automação*. Por fim, o Elaborador de orçamento será responsável por armazenar os orçamentos definidos e o Estimador de custo irá gerar estimativas das MBAs antes de implantá-las em nuvem.

Figura 5 – Arquitetura da solução COSTA




**Fonte:** Elaborada pelo autor (2023).

### 3.2.1 *Monitor*

O *Monitor* (Figura 5) é o microsserviço responsável por três tarefas em COSTA. Ele obtém periodicamente os custos de execução em todos os provedores de nuvem em que as MBAs podem ser implantadas (**R1**, **R2** e **R3**). Os custos dos diferentes provedores de nuvem (*e.g.*, *Amazon Web Services* (AWS), *Google Cloud Platform* (GCP) e *Microsoft Azure* (AZ)) são obtidos por meio de consultas em APIs disponibilizadas pelos provedores e armazenadas (*Base de Conhecimento*) internamente em COSTA para uso em futuras análises. Essas consultas são feitas de forma periódica para minimizar os efeitos de possíveis falhas que podem ocorrer durante uma consulta.

Outra responsabilidade do *Monitor* é monitorar o estado das MBAs de forma periódica para verificar se a MBA está em execução ou indisponível. Caso seja detectada alguma indisponibilidade no serviço de nuvem, o *Monitor* envia uma mensagem ao *Analisador*, informando a indisponibilidade do serviço, para que este possa avaliar se deve ou não efetuar a migração da MBA para outra nuvem.

Por fim, o *Monitor* é responsável por coletar as métricas de uso dos recursos computacionais das MBAs. As métricas coletadas são: transferência de dados de saída, armazenamento em disco e tempo de execução da máquina virtual. Em seguida, essas informações são enviadas para o microsserviço *Analisador*. As métricas são obtidas por consultas nas APIs fornecidas pelos provedores de IaaS.

### 3.2.2 *Elaborador de orçamento*

O *Elaborador de orçamento* (Figura 5) é o microsserviço responsável por armazenar (*Base de Conhecimento*) orçamentos das MBAs implantadas em COSTA (**R4**). Primeiramente, o *Desenvolvedor de Aplicação* registra o orçamento em COSTA, notificações de violação desses orçamentos e ações em caso de violação. Em caso de violação, a MBA é removida ou seus recursos computacionais são reduzidos. Então, uma rotina recupera os orçamentos e notificações de violação armazenadas internamente em COSTA e os envia para os provedores de IaaS. Depois disso, é possível enviar notificações informando aos responsáveis sobre a situação do orçamento da MBA. Vale ressaltar que o *Elaborador de orçamento* está integrado com APIs fornecidas pelos provedores de serviços, o que permite a sincronização dos dados orçamentários armazenadas em COSTA com o provedor de serviços em nuvem.

As informações armazenadas no *Elaborador de orçamento* são usadas pelo *Analisador*, com o objetivo de analisar a necessidade de migração durante a vigência do orçamento. Se esse orçamento expirar, um alerta é emitido ao responsável da MBA para que ele tome as ações necessárias, *e.g.,* criar um novo orçamento com uma nova data de vigência.

### 3.2.3 *Estimador de custo*

O *Estimador de custo* (Figura 5) armazena e gera estimativas de custos onde as MBAs podem ser implantadas na nuvem (**R5**). Primeiro, para gerar estimativas de custos das MBAs, é necessário fazer uma requisição ao *Estimador de custo*. Após isso, a requisição é armazenada

e uma rotina recupera as requisições e as envia para o *Servidor de Automação* (*Ferramentas de Automação* na Figura 5) executar a tarefa responsável por gerar estimativas de custo para as nuvens integradas a COSTA. Quando a tarefa é concluída, o *Servidor de Automação* envia as informações da estimativa para o *Estimador de custo*. Recebidas as estimativas de cada nuvem integrada a COSTA, as informações são armazenadas e utilizadas em análises, por exemplo, para indicar qual nuvem é mais barata para implantar a MBA inicialmente.

A decisão de armazenar as informações de estimativas de custos internamente em COSTA foi devido ao processo assíncrono das estimativas de custos. Ou seja, a realização das estimativas de custos leva um tempo para ser processada no *Servidor de Automação*. Portanto, se os requisitos de infraestrutura forem pequenos, o tempo de publicação de dados para consumo é reduzido. O *Servidor de Automação* precisará de mais tempo para emitir estimativas de custo de consumo se os requisitos de infraestrutura forem maiores. Vale observar que os requisitos de infraestrutura mencionados anteriormente são a quantidade de recursos computacionais necessária para executar uma MBA na nuvem.

### 3.2.4 *Analisador*

O *Analisador* (Figura 5) é o microsserviço responsável por duas tarefas em COSTA (**R6** e **R7**). Ele analisa os dados coletados pelo *Monitor* (dos provedores de nuvem), informações de estimativas de custos (geradas pelo *Estimador de custo*) e orçamentárias (do *Elaborador de orçamento*) para verificar se o custo previsto está dentro do orçamento planejado. Em seguida, o *Analisador* decide por uma migração em três situações: o orçamento foi violado, há indisponibilidade do serviço de nuvem ou o custo de um recurso em outro provedor de nuvem é menor que o provedor atual.

As métricas extraídas pelo *Monitor*, geradas a partir do uso dos recursos de nuvem, são de grande importância para o *Analisador* avaliar possíveis migrações. Por exemplo, se o consumo de armazenamento em disco (um fator de custo) aumentar, mas em outro provedor de nuvem o custo de armazenamento é menor, então torna-se interessante migrar a MBA.

As métricas recebidas pelo *Analisador* são: transferência de dados de saída, armazenamento em disco e tempo de execução da máquina virtual para avaliar se está ocorrendo um aumento no consumo e, além disso, informações do atual custo da MBA em nuvem para verificar uma possível violação no orçamento. Outra informação analisada pelo *Analisador* é a disponibilidade do serviço de nuvem.

Todas as tomadas decisões do *Analisador* sobre o fluxo de migrações entre diferentes provedores de nuvem são armazenadas em COSTA para a realização de análises nos cenários executados. Por exemplo, se o cenário de migração de indisponibilidade de serviço de nuvem está funcionando corretamente ou está apresentando falhas em sua execução.

No final, o *Analisador* gera um relatório de análise e envia para o *Planejador*. O relatório de análise contém informações das métricas usadas na análise e também a métrica que foi avaliada e disparou o pedido de solicitação de migração para o *Planejador*.

### 3.2.5 *Planejador*

O *Planejador* (Figura 5) cria os planos de adaptação de alto nível para migrar um MBA do atual provedor de nuvem para outro (**R8**). Além disso, é possível criar planos de implantação e remoção das MBAs nas nuvens integradas a COSTA.

Todos os planos de adaptação das MBAs criados pelo *Planejador* são solicitações enviadas pelo microsserviço *Analisador*. No entanto, foi desenvolvida uma funcionalidade no *Planejador* a partir da qual é possível gerar planos de implantação e remoção da MBA sem executar o processo de análise realizado pelo serviço *Analisador*. A funcionalidade foi desenvolvida para facilitar o processo de implantação (*e.g.,* verificar se toda infraestrutura está configurada corretamente) e remoção (*e.g.,* desistir da execução de um experimento) de toda infraestrutura em nuvem. Vale observar que todos os planos criados são enviados para o *Executor*.

O *Planejador* é responsável por definir um conjunto de ações para implantar, migrar ou remover as MBAs. Por exemplo, no fluxo de migração das MBAs, ele define estes conjuntos de ações a partir da análise do relatório enviado pelo *Analisador*. O relatório de análise contém a nuvem de destino para implantar a MBA. Em seguida, o *Planejador* obtém o provedor atual da MBA, consultando a informação em sua base de dados (*Base de Conhecimento*). Como resultado, é gerado um plano de adaptação de migração de alto nível (*e.g.,* migrar a MBA X da nuvem de origem A para a nuvem de destino B) com as informações de nuvem de origem e nuvem de destino.

Por fim, o *Planejador* adota os seguintes parâmetros na geração dos planos de alto nível para realização de uma migração: nome da MBA, nuvem de origem e nuvem de destino. Para gerar um plano de alto nível para implantação ou remoção os parâmetros são: nome da MBA, nuvem de origem e ação (*e.g.,* implantar ou remover).

### 3.2.6 *Executor*

O *Executor* (Figura 5) implanta a MBA na nuvem e possui um banco de dados (*Base de Conhecimento*) com todas as informações de implantação, por exemplo, nome da MBA, nuvens de origem e destino (**R9**). O *Executor* é projetado para executar os planos de implantação, adaptação e remoção gerados pelo *Planejador*. O plano é uma sequência de comandos de alto nível, requisições ao *Servidor de Automação* (*Ferramentas de Automação* na Figura 5).

Logo, existe uma comunicação entre o *Executor* e a ferramenta *Servidor de Automação*. O *Executor* recebe uma informação de alto nível (*e.g.,* implantar a MBA na nuvem A), executa uma solicitação ao *Servidor de Automação*. O *Servidor de Automação*, por sua vez, aplica a implantação da MBA em nuvem. Além disso, o *Servidor de Automação* informa ao *Executor* o resultado da ação executada, *e.g.,* falha na implantação da MBA, sucesso na remoção da MBA, sucesso na implantação da MBA.

### 3.2.7 *Ferramentas de Automação*

O componente *Ferramentas de Automação* na Figura 5 inclui ferramentas externas necessárias para gerar estimativas de custo, implantar, migrar e remover MBAs: *Servidor de Automação*, *Balanceador de Carga* e *Servidor de DNS*. O *Servidor de Automação* é responsável por orquestrar toda a sequência de estágios necessários para gerar estimativas de custo, implantar, migrar e remover MBAs em diferentes provedores.

O *Balanceador de Carga* tem a responsabilidade de filtrar o fluxo de rede recebido da nuvem em que a MBA está em execução. No entanto, o *Balanceador de Carga* possui informações sobre todas a nuvens integradas a COSTA. A MBA, entretanto, só estará em execução em um provedor de nuvem por vez. Em seguida, o *Servidor de DNS* recebe o fluxo de rede filtrado pelo *Balanceador de Carga* e o seu IP de rede. Como resultado, o *Servidor de DNS* converte solicitações de nomes em endereços IP, controlando qual servidor será utilizado.

## 3.3 PROJETO E IMPLEMENTAÇÃO

Os microsserviços da solução COSTA foram desenvolvidos usando o estilo arquitetural *Representational State Transfer* (REST) (FIELDING, 2000). O *Spring Boot* (VMWARE TANZU INC., 2023a) foi usado na criação dos microsserviços da COSTA. Além disso, outras ferra-

mentas foram usadas na construção da solução COSTA. Por exemplo, *Spring Data* (VMWARE TANZU INC., 2023b) para acessar a camada de banco de dados, *PostgreSQL* (POSTGRESQL GLOBAL DEVELOPMENT GROUP., 2023) para o armazenamento dos dados, *Gradle* (GRADLE INC., 2023) para automatizar o *build* da aplicação e *Swagger UI* (SMARTBEAR SOFTWARE., 2023) para a documentação dos microsserviços. Foram considerados AWS, AZ e GCP por serem os provedores de nuvem mais populares (FLEXERA, 2023).

O *Monitor* tem as credenciais para acessar os custos e os fatores de custos relacionados aos MBAs implantados na nuvem. Para obter as informações dos fatores de custo de cada MBA, o *Monitor* usa APIs oferecidas pelos provedores de nuvem, como *Amazon CloudWatch* (AMAZON WEB SERVICES, INC., 2023b), API de monitoramento *Stackdriver* (GOOGLE INC., 2023) e *Azure Monitor* (MICROSOFT, 2023). O *Monitor* obtém informações de custo da AWS por meio da ferramenta *Cost Explorer* API (AMAZON, 2023), que permite consultar programaticamente os custos e dados de uso. O custo da GCP é obtido através do *BigQuery* (GOOGLE, 2023) que acessa os dados de faturamento. Em AZ, o custo é obtido por meio de consultas à API REST de cobrança do *Azure* (MICROSOFT, 2023), que nos permite visualizar e gerenciar detalhes de cobrança. O *Monitor* também monitora o estado da execução da MBA (*e.g., up* ou *down*) disponibilizados pelas aplicações.

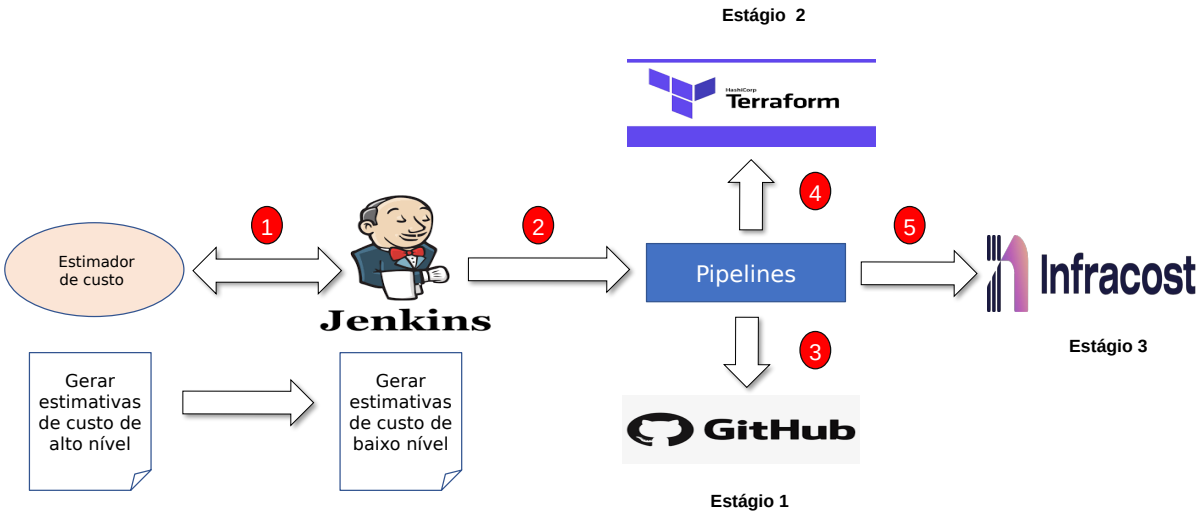
O Microsserviço *Elaborador de orçamento* possui todos os orçamentos das MBAs implantadas em COSTA. Ele tem as credenciais para criar os orçamentos e notificações de violação desses orçamentos em nuvem. Para criar o orçamento e notificação das MBAs, o *Elaborador de orçamento* usa serviços oferecidos pelos provedores, como a API AWS *Billing and Cost Management* (AMAZON WEB SERVICES, INC., 2023a), *Budgets* - REST API (*Azure Consumption*) (MICROSOFT, 2023). Em GCP, o orçamento é criado com a API *Cloud Billing Budget* (GOOGLE CLOUD, 2023b) e notificações com o serviço *Create and manage notification channels by API* (GOOGLE CLOUD, 2023a).

O *Estimador de custo* é responsável pela gestão das estimativas de custos das MBAs. A Figura 6 apresenta o fluxo de trabalho da geração de estimativa de custo do *Servidor de Automação* (*Ferramentas de Automação* na Figura 5).

O *Servidor de Automação* adotado foi o *Jenkins* (COMMUNITY JENKINS, 2023), que possui uma API REST utilizada para receber requisições para execução de *pipelines*, ou seja, sequência de estágios a serem executados. O *Estimador de custo* usa essa API para gerar estimativas de custo das MBAs (1).

*Terraform* (HASHICORP, 2023), *Infracost* (INFRACOST INC., 2023) e *GitHub* (GITHUB, 2023)

Figura 6 – Fluxo de geração de estimativa de custo do *Servidor de Automação*




**Fonte:** Elaborada pelo autor (2023).

foram integrados ao *Jenkins* para executar os comandos de geração de estimativas de custo de alto nível do *Estimador de custo*. Então, as seguintes etapas foram necessárias nos *Pipelines* do *Jenkins* (2):

1. ***Git Checkout*:** baixa o código fonte a partir do repositório *GitHub* (3).
2. ***Terraform Init*:** é usado para inicializar uma pasta de trabalho que armazena um arquivo de configuração do *Terraform*. Portanto, este é o primeiro comando a ser executado (4).
3. ***Infracost Generate Cost Estimation*:** gera uma estimativa de custo da nuvem com base no código do *Terraform* (5).
4. ***Send Cost Estimation*:** esta etapa é responsável por enviar a estimativa de custo para o *Estimador de custo*.

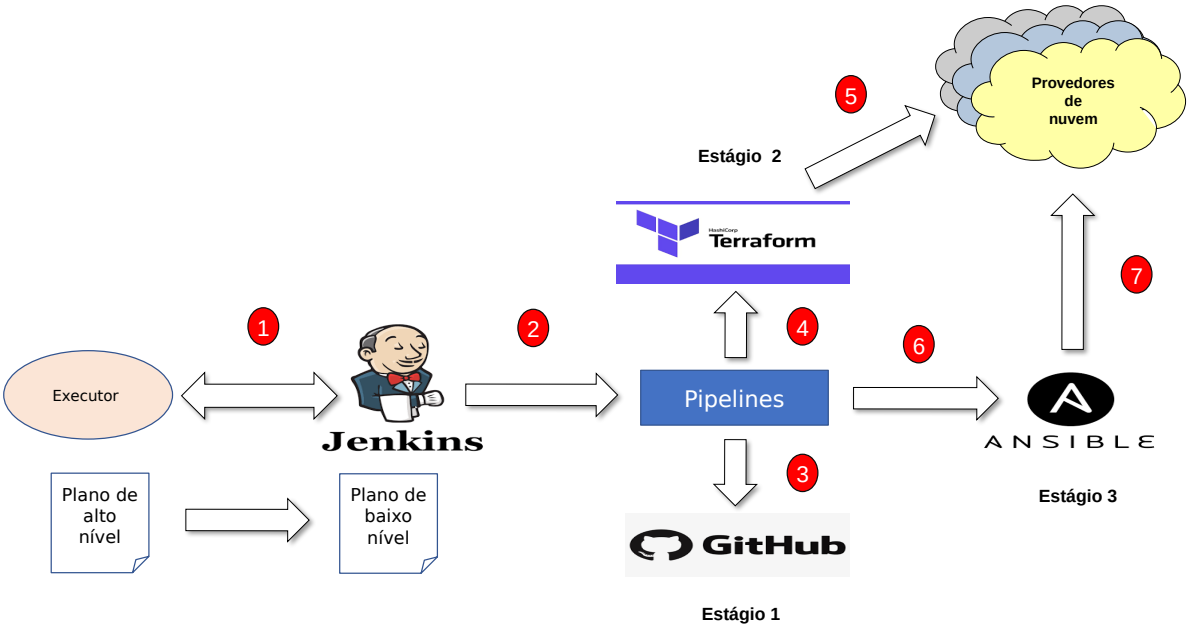
Essas etapas realizam a estimativa de custo de um MBA na nuvem. O *Infracost* mostra estimativas de custo de nuvem para projetos de *Infrastructure as Code* (IaC), como o *Terraform*. Ele ajuda *Development and Operations* (DevOps) (EBERT *et al.*, 2016), *Site Reliability Engineering* (SRE) (MADAMANCHI, 2021) e desenvolvedores a ver rapidamente uma divisão de custos e comparar diferentes opções antecipadamente.

O *Analisador* verifica os dados obtidos do *Elaborador de orçamento* e do *Monitor* para verificar se o orçamento previsto está sendo violado. Outra função do *Analisador* é migrar as MBAs por interrupção do serviço de nuvem, e, na tomada de decisão, aplicar as estimativas de custos (armazenadas pelo *Estimador de custo*). Ele também analisa o uso dos recursos computacionais das MBAs procurando cenários em que a migração entre nuvens diminuirá o custo. Caso seja necessária uma migração entre provedores de nuvem, o *Planejador* monta um plano de adaptação.

Como mencionado na Seção 3.2.5, o *Planejador* é responsável por especificar o plano de adaptação. Este plano contém uma sequência de comandos de alto nível: implantar, migrar e remover. O comando implantar possui dois parâmetros de entrada: o nome da MBA e a nuvem em que a MBA precisa ser implantada. O comando de migração possui três parâmetros de entrada: identificação da MBA, nuvem de origem (provedor onde a MBA está em execução) e nuvem de destino (novo provedor de nuvem). Por fim, o comando remover é parametrizado com a identificação da MBA e provedor de nuvem da qual a infraestrutura será removida.

A Figura 7 apresenta o fluxo de trabalho da implantação e migração da ferramenta externa: *Servidor de Automação* (*Ferramentas de Automação* na Figura 5).

Figura 7 – Fluxo de trabalho da implantação e migração do *Servidor de Automação*




**Fonte:** Elaborada pelo autor (2023).

O *Executor* usa a API REST do *Jenkins* para migrar as MBAs entre provedores de nuvem

(1). Vale ressaltar que essa API REST também é utilizada para o processo de implantação e remoção das MBAs em nuvem.

*Terraform*, *Ansible* (RED HAT, 2023) (ferramenta integrada ao *Jenkins*) e *GitHub* são usados para executar os comandos de alto nível do plano de adaptação. Portanto, as seguintes etapas foram necessárias nos *Pipelines* do *Jenkins* (2):

1. ***Git Checkout*:** baixa o código fonte a partir do repositório *GitHub* para o provisionamento de infraestrutura (*Terraform*) e configuração de máquina virtual (*Ansible*) (3).

2. ***Terraform Init*:** é usado para inicializar uma pasta de trabalho que armazena um arquivo de configuração do *Terraform*. Portanto, este é o primeiro comando a ser executado (4).

3. ***Terraform Apply*:** aplica todas as mudanças necessárias do plano de adaptação ou conjunto predeterminado de ações que executam todo o provisionamento da infraestrutura de nuvem (5).

4. ***Start Ansible*:** primeiramente, o *Ansible* é iniciado (6). Em seguida, realiza a configuração e preparação da máquina virtual para posteriormente baixar as imagens da MBA no *Docker Hub* (DOCKER INC., 2023) e executá-la (7).

5. ***Terraform Destroy*:** é responsável por destruir a infraestrutura gerenciada pela *Terraform*.

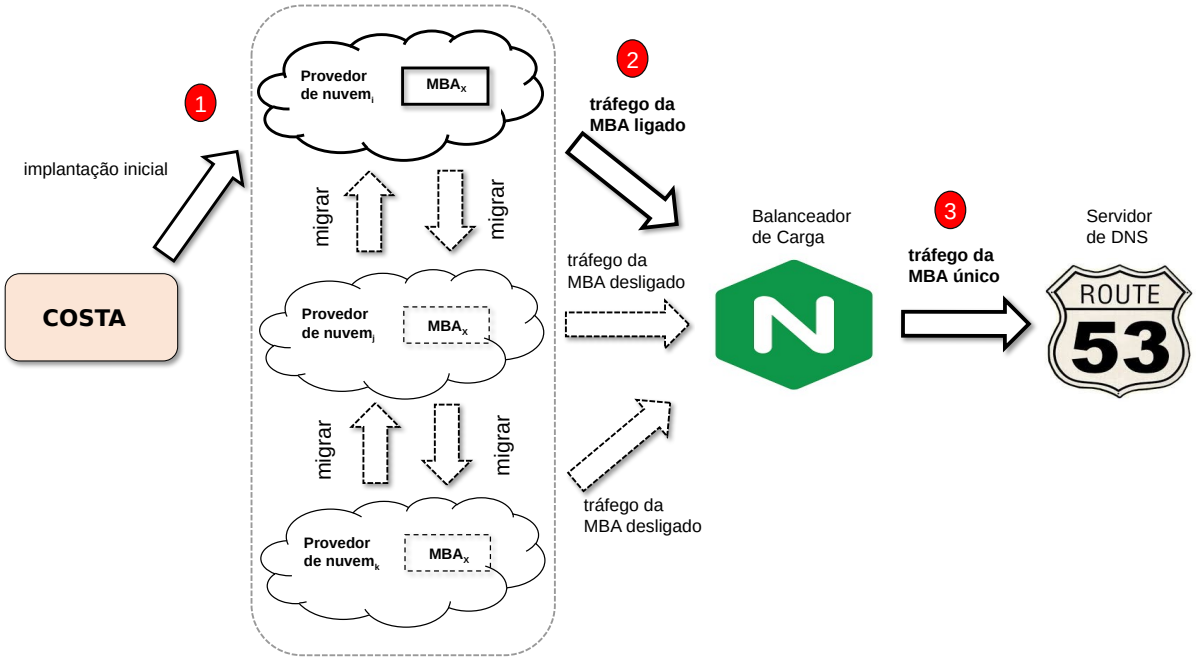
6. ***Send Execution Result*:** envia o resultado da execução (*e.g.,* sucesso ou falha) das tarefas de implantação, migração e remoção de um MBA para o *Executor*.

Essas etapas realizam a implantação, migração e configuração de máquinas virtuais de uma MBA na nuvem. Caso seja a remoção da MBA em nuvem, só será necessário executar a etapa ***Terraform Destroy***.

A Figura 8 apresenta o fluxo de trabalho das ferramentas externas: *Balanceador de Carga* e *Servidor de DNS* (*Ferramentas de Automação* na Figura 5).

Inicialmente a MBA será implantada em um dado provedor de nuvem (1). Nesse caso, podem ocorrer migrações entre nuvens para reduzir os custos da MBA, mas a MBA só será executada em uma nuvem por vez. A tecnologia utilizada para o *Balanceador de Carga* foi o NGINX (F5, INC., 2023). O NGINX recebe o tráfego de rede da MBA em execução na AWS, GCP ou AZ (2). *Amazon Route 53* (AMAZON WEB SERVICES, INC., 2023c) (tecnologia usada

Figura 8 – Fluxo de trabalho do *Balanceador de Carga* e *Servidor de DNS*




**Fonte:** Elaborada pelo autor (2023).

para o *Servidor de DNS*) recebe a saída do tráfego da MBA filtrado pelo NGINX (3). No final, o *Amazon Route 53* irá disponbilizar a MBA em um endereço (*URL*) para ser acessada na *Web*.

## 3.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os microsserviços e as ferramentas necessárias para execução da solução COSTA. Primeiramente, foram detalhados os requisitos e arquitetura com o auxílio de uma visão geral. Em seguida, foi detalhada cada tecnologia integrada a COSTA e também sua função no projeto. Por fim, foram apresentadas as ferramentas necessárias para gerar estimativas de custo, implantar, migrar e remover MBAs em nuvem.

# 4 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta as etapas da avaliação experimental de COSTA. Primeiramente, o capítulo apresenta uma descrição dos experimentos realizados. Em seguida, são mostrados os resultados obtidos e uma análise dos experimentos. Por fim, é mostrada uma análise da redução dos custos de cada avaliação experimental.

## 4.1 EXPERIMENTOS

A metodologia usada para a construção dos experimentos realizados nesse trabalho consiste dos seguintes passos (JAIN, 1991):

1. **Definição dos objetivos da avaliação experimental**: o objetivo da avaliação experimental é avaliar o impacto de COSTA na redução do custo das MBAs implantadas em ambientes de múltiplas nuvens. Vale pontuar que não foram considerados aspectos de disponibilidade na avaliação experimental. Além disso, o modelo de preço utilizado na avaliação dos experimentos foi o estático com planos de reservas avançadas (de um ano e três anos) e sob demanda. Por fim, o modelo dinâmico não foi considerado na avaliação experimental por causa do seu aspecto transiente, ou seja, a capacidade do provedor de nuvem de interromper o fornecimento do serviço no meio de uma avaliação experimental (ver Seção 2.4.2).

2. **Definição dos serviços a serem avaliados**: COSTA possui um conjunto de microsserviços com objetivos bem definidos. Os microsserviços avaliados incluem o *Monitor, Elaborador de orçamento, Estimador de custo, Analisador, Planejador* e *Executor*.

3. **Definição dos critérios (métricas) para a avaliação**: a métrica considerada na avaliação de cada experimento foi o *custo financeiro* (em real).

4. **Definição dos parâmetros da avaliação**: os parâmetros considerados na avaliação foram o estado do *Mecanismo de adaptação* que pode ser ativado ou desativado; os *Provedores de nuvem* considerados incluem AWS, GCP e AZ; o *Tamanho da VM* utilizada (grande e extra grande); as MBAs *Hipster Shop* (LIGHTSTEP, 2023) e *Sock Shop* (WEAVE, 2023) foram adotadas na execução dos experimentos; o *Armazenamento de dados* são os dados persistidos na base de dados das MBAs; a *Transferência de dados*,

ou seja, dados enviados da nuvem para internet; o *Tempo de execução da VM*, *i.e.*, a duração da execução dos experimentos; e o *Número de solicitações à aplicação* são requisições feitas as MBAs. A Tabela 1 mostra os parâmetros avaliados.

Tabela 1 – Parâmetros

| Parâmetro | Descrição |
|---|---|
| Mecanismo de adaptação | Ativado, Desativado |
| Provedores de nuvem | AWS, GCP, AZ |
| Tamanho da VM | Grande, Extra grande |
| MBAs | *Hipster Shop* , *Sock Shop* |
| Armazenamento de dados | 256 GB, 512 GB, 1024 GB |
| Transferência de dados | 50 GB, 250 GB |
| Tempo de execução da VM | 30 dias |
| Número de solicitações à aplicação | 10.000, 60.000 |

**Fonte:** Elaborada pelo autor (2023).

5. **Definição dos fatores**: Alguns parâmetros apresentados na Tabela 1 são conhecidos como fatores e variam nos experimentos, conforme mostrado na Tabela 2.

Tabela 2 – Fatores e níveis

| Parâmetro (Fator) | Níveis |
|---|---|
| Mecanismo de adaptação | Ativado, Desativado |
| Provedor de nuvem | AWS, GCP, AZ |
| Tamanho da VM | Grande, Extra grande |
| MBA | *Hipster Shop* (HS), *Sock Shop* (SS) |
| Armazenamento de dados | 256 GB, 512 GB, 1024 GB |
| Transferência de dados | 50 GB, 250 GB |
| Número de solicitações à aplicação | 10.000, 60.000 |

**Fonte:** Elaborada pelo autor (2023).

COSTA pode ser ativada (ocorrem migrações) ou desativada (não ocorrem migrações). Em relação aos provedores de nuvem, AWS, GCP e AZ foram adotados por serem amplamente utilizados e facilmente integrados ao *Terraform* e *Ansible* na execução dos experimentos. Outro fator considerado é o tamanho das VMs: Grande (2 vCPUs e 8 GB de RAM) e Extra grande (4 vCPUs e 16 GB de RAM). Vale observar que não foi possível utilizar as VMs Pequena e Média nos experimentos, pois suas respectivas configurações

eram insuficientes para a execução das MBAs. As MBAs utilizadas foram *Hipster Shop* (10 microsserviços) (LIGHTSTEP, 2023) e *Sock Shop* (15 microsserviços) (WEAVE, 2023). Essas MBAs têm sido amplamente adotadas em experimentos envolvendo o desenvolvimento de novas tecnologias/ferramentas sobre microsserviços.

6. **Escolher a carga de trabalho**: Dois tipos de carga de trabalho foram usados na avaliação experimental: fixa e variável. Com uma carga de trabalho fixa, o número de clientes e solicitações não mudam durante o experimento, *e.g.,* 3 clientes fazem 10.000 solicitações simultâneas ao MBA por 30 dias quando o mecanismo de adaptação está desativado. Quando o mecanismo de adaptação é ativado, 6 clientes simultâneos fazem 60.000 solicitações ao MBA por 30 dias. O número de clientes e solicitações foi definido para descrever as cargas de trabalho médias (3 clientes fazem 10.000 solicitações) e altas (6 clientes fazem 60.000 solicitações). Vale pontuar que as cargas de trabalhos foram diferentes quando o mecanismo está desativado e ativado. Os cenários com o mecanismo desativado serviriam para evidenciar os custos das nuvens, então foi utilizada uma carga de trabalho fixa. Quando o mecanismo foi ativado, foram utilizadas cargas de trabalho variáveis para evidenciar as diferenças nos custos dos recursos computacionais (fatores de custos) de cada nuvem utilizada nos experimentos;

   Estas cargas de trabalho foram usadas em seis cenários distintos:

   - **Cenário 1**: Este cenário foi definido para explorar o custo gerado pela instância executando no plano sob demanda com o tamanho da máquina virtual grande. Este cenário consiste na execução da MBA *Hipster Shop*, em que o mecanismo de adaptação inicia desativado (para avaliar os custos) e depois é ativado para avaliar a capacidade de COSTA em reduzir custos.
   - **Cenário 2**: Este cenário foi definido para explorar o custo gerado pela instância executando no plano sob demanda com o tamanho da máquina virtual extra grande. Este cenário consiste na execução da MBA *Sock Shop*, em que o mecanismo de adaptação inicia desativado (para avaliar os custos) e depois é ativado para avaliar a capacidade de COSTA em reduzir custos.
   - **Cenário 3**: Este cenário explora o custo gerado pela instância executando no plano com reservas de um ano e tamanho da máquina virtual grande. Este cenário consiste na execução da MBA *Hipster Shop*, em que o mecanismo de adaptação inicia

desativado (para avaliar os custos) e depois é ativado para avaliar a capacidade de COSTA em reduzir custos.

- **Cenário 4**: Este cenário avalia o custo gerado pela instância executando no plano com reservas de um ano e tamanho da máquina virtual extra grande. Ele consiste na execução da MBA *Sock Shop*, em que o mecanismo de adaptação inicia desativado (para avaliar os custos) e depois é ativado para avaliar a capacidade de COSTA em reduzir custos.
- **Cenário 5**: Neste cenário, explora-se o custo gerado pela instância executando no plano com reservas de três anos e tamanho da máquina virtual grande. Este cenário consiste na execução da MBA *Hipster Shop*, em que o mecanismo de adaptação inicia desativado (para avaliar os custos) e depois é ativado para avaliar a capacidade de COSTA em reduzir custos.
- **Cenário 6**: Este cenário foi definido para explorar o custo gerado pela instância executando no plano com reservas de três anos e tamanho da máquina virtual extra grande. Este cenário consiste na execução da MBA *Sock Shop*, em que o mecanismo de adaptação inicia desativado (para avaliar os custos) e depois é ativado para avaliar a capacidade da COSTA em reduzir custos.

Quando o mecanismo de adaptação é ativado, a nuvem de destino é selecionada por meio de análises de utilização de recursos computacionais que impactam o custo. Além disto, as cargas de trabalho são variáveis em todos os cenários com a configuração de 6 clientes realizando 60.000 solicitações. Quando o mecanismo de adaptação está desativado, são executadas cargas de trabalho fixas com a configuração de 3 clientes realizando 10.000 solicitações.

Em todos os cenários, o tempo de execução total foi de 30 dias. É importante observar que esse é o período de cobrança normalmente usado pelos provedores de nuvem. A Tabela 3 resume os diferentes cenários avaliados. A Tabela 3 possui as seguintes colunas: Cenário, Tipo de Cenário (TC), Carga de Trabalho (CT), Clientes, Solicitações e Tempo de Execução (TE);

Por fim, o *JMeter* (APACHE, 2023a) foi o *software* escolhido para geração de carga de trabalho nos experimentos. Ele realiza testes de carga e de estresse em recursos estáticos

ou dinâmicos oferecidos por sistemas computacionais. Para executar os cenários com o *JMeter* foram necessárias as informações de número de clientes e solicitações.

Tabela 3 – Informações sobre a carga de trabalho

| Cenário | TC | CT | Clientes | No. de Solicitações | TE (Dias) |
|---|---|---|---|---|---|
| 1 | Fixo | Média | 3 | 10.000 | 30 |
| 2 | Fixo | Média | 3 | 10.000 | 30 |
| 3 | Fixo | Média | 3 | 10.000 | 30 |
| 4 | Fixo | Média | 3 | 10.000 | 30 |
| 5 | Fixo | Média | 3 | 10.000 | 30 |
| 6 | Fixo | Média | 3 | 10.000 | 30 |
| 1 | Variável | Alta | 6 | $>$ 20.000 | 30 |
| 2 | Variável | Alta | 6 | $>$ 20.000 | 30 |
| 3 | Variável | Alta | 6 | $>$ 20.000 | 30 |
| 4 | Variável | Alta | 6 | $>$ 20.000 | 30 |
| 5 | Variável | Alta | 6 | $>$ 20.000 | 30 |
| 6 | Variável | Alta | 6 | $>$ 20.000 | 30 |

**Fonte:** Elaborada pelo autor (2023).

7. **Projetar o experimento**: cada experimento foi executado seguindo a sequência da Tabela 3. Vale ressaltar que, na execução de todos os cenários, as informações geradas são armazenadas em COSTA (Base de Conhecimento).

   Como mencionado anteriormente, a métrica *custo financeiro* (em real) é extraída após a finalização da execução de cada experimento por consultas feitas ao provedor de nuvem. É importante observar que o custo financeiro é composto pelo conjunto de fatores de custos (Capítulo 1). Ao final de cada experimento, são coletados os custos mensais de cada fator de custo, por exemplo, tempo de execução da VM, armazenamento em disco e transferência de dados. Em seguida, calcula-se a soma de cada fator de custo para se obter o custo total. Em cenários em que há migrações entre nuvens, os dados são coletados de cada provedor de nuvem e, finalmente, agregados para fornecer o custo mensal.

8. **Análise e interpretação dos resultados**: a análise e interpretação dos dados é feita por meio de tabelas e gráficos que apresentam os custos detalhados de cada fator de custo. Por fim, para obter o custo mensal de cada cenário experimental executado, é

realizada a soma de todos os fatores de custo, por exemplo, tempo de uso da VM em 30 dias, Transferência de dados (TD) e Armazenamento (AR).

É importante observar que, ao executar experimentos com o Mecanismo de Adaptação ativado, as VMs receberam endereços IP estáticos para facilitar a avaliação experimental. Se não se configurar uma estratégia de IP estático, os provedores de serviços usarão endereços IP dinâmicos. Outro local que contém os endereços IP estáticos é o *Balanceador de Carga* (ver Figura 8). Ele é responsável por filtrar o tráfego da nuvem onde a MBA está em execução.

A Tabela 4 mostra as configurações das VMs utilizadas nos experimentos. As regiões utilizadas nos provedores de nuvem foram *US East/N. Virginia* (AWS), *South Carolina/us-east1* (GCP) e *West US* 2 (AZ). Essas regiões tiveram os custos mais baixos no momento dos experimentos[1].

Tabela 4 – Configuração dos experimentos

| Cenários | MBA | Nuvem | OS | VM | vCPUs | RAM |
|---|---|---|---|---|---|---|
| 1, 3, 5 | HS | AWS | Linux | t2.large | 2 | 8 GB |
| 1, 3, 5 | HS | GCP | Linux | e2-standard-2 | 2 | 8 GB |
| 1, 3, 5 | HS | AZ | Linux | B2ms | 2 | 8 GB |
| 2, 4, 6 | SS | AWS | Linux | t2.xlarge | 4 | 16 GB |
| 2, 4, 6 | SS | GCP | Linux | e2-standard-4 | 4 | 16 GB |
| 2, 4, 6 | SS | AZ | Linux | B4ms | 4 | 16 GB |

**Fonte:** Elaborada pelo autor (2023).

## 4.2 RESULTADOS

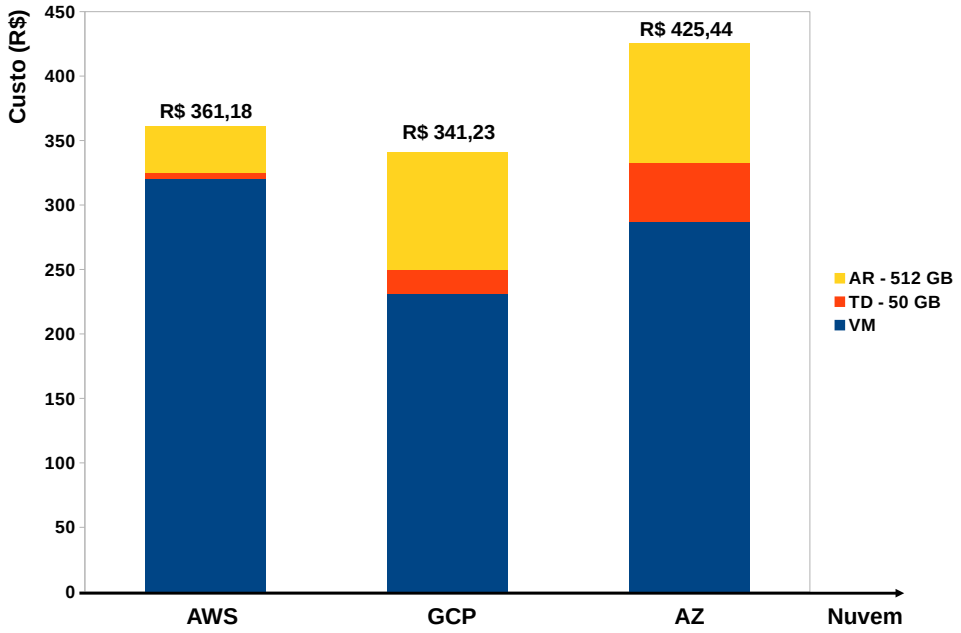
As análises dos resultados dos experimentos são apresentadas da seguinte forma: nas Seções 4.2.1 e 4.2.2, são apresentados os resultados da avaliação dos cenários sob demanda. Em seguida, as Seções 4.2.3 e 4.2.4 mostram os resultados da avaliação dos cenários de instâncias com reservas de um ano. Finalmente, as Seções 4.2.5 e 4.2.6 apresentam os resultados da avaliação experimental dos cenários de instâncias com reservas de três anos.

[1] O período das realizações dos experimentos foi entre 01/06/2023 e 30/06/2023.

### 4.2.1 Instâncias sob demanda (*Hipster Shop*)

Nos experimentos envolvendo a *Hipster Shop*, os custos para executar a MBA por um mês com o mecanismo de adaptação desativado são mostrados na Figura 9 e detalhados na Tabela 5. Além disso, a Figura 9 apresenta os custos das execuções dos experimentos com a *Hipster Shop* divididos por recursos computacionais (*e.g.,* VM, Transferência de dados e Armazenamento) nas nuvens AWS, GCP e AZ. A Tabela 5 destaca o provedor mais barato (GCP) e detalha o custo por nuvem de cada recurso computacional usado.

Figura 9 – Pagamento mensal (Desativado, *Hipster Shop, VM Large*, N = 10.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 5 – Detalhes de custo (R$) do cenário 4.2.1 e Mecanismo de migração Desativado

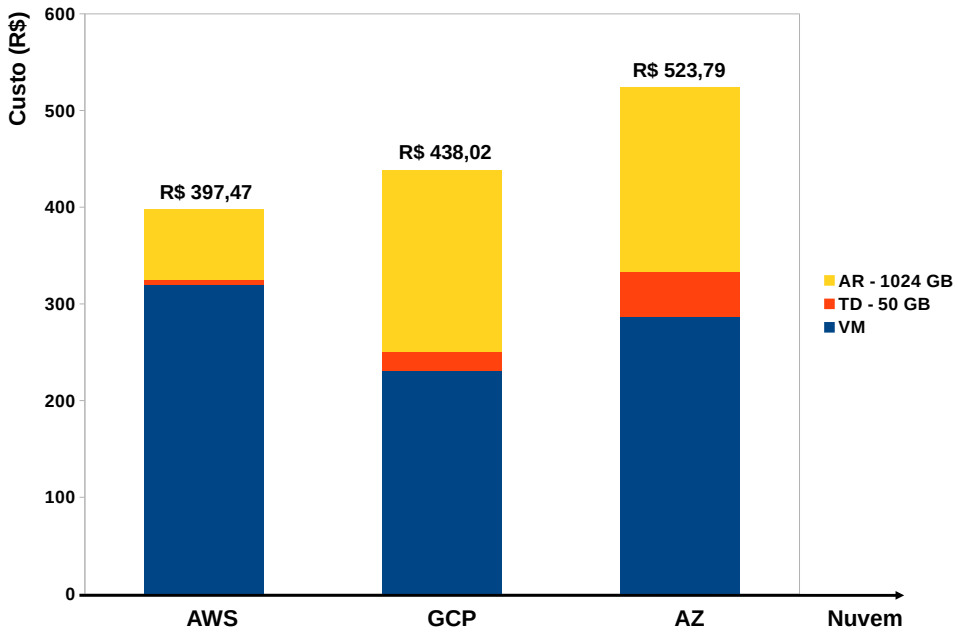
| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 320,15 | 4,73 | 36,30 | 361,18 |
| **GCP** | **231,21** | **18,90** | **91,12** | **341,23** |
| AZ | 287,07 | 46,08 | 92,29 | 425,44 |

**Fonte:** Elaborada pelo autor (2023).

Quando o mecanismo de adaptação foi ativado e a MBA foi implantada inicialmente no provedor mais barato (GCP), os custos mensais são os mostrados na Figura 9. À medida que o

número de solicitações aumentou, COSTA identificou um crescimento no uso de persistência de dados que elevou o custo da MBA. Em seguida, o Planejador foi ativado e criou um plano de migração para reduzir o custo movendo a MBA da GCP para um provedor mais barato (AWS). Com essa migração, o custo mensal da *Hipster Shop* foi de R$ 397,47, ou seja, uma economia de R$ 40,55, conforme Figura 10 e Tabela 6.

Figura 10 – Pagamento mensal (Ativado, *Hipster Shop, VM Large*, GCP → AWS, N = 60.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 6 – Detalhes de custo (R$) do cenário 4.2.1 e Mecanismo de migração Ativado

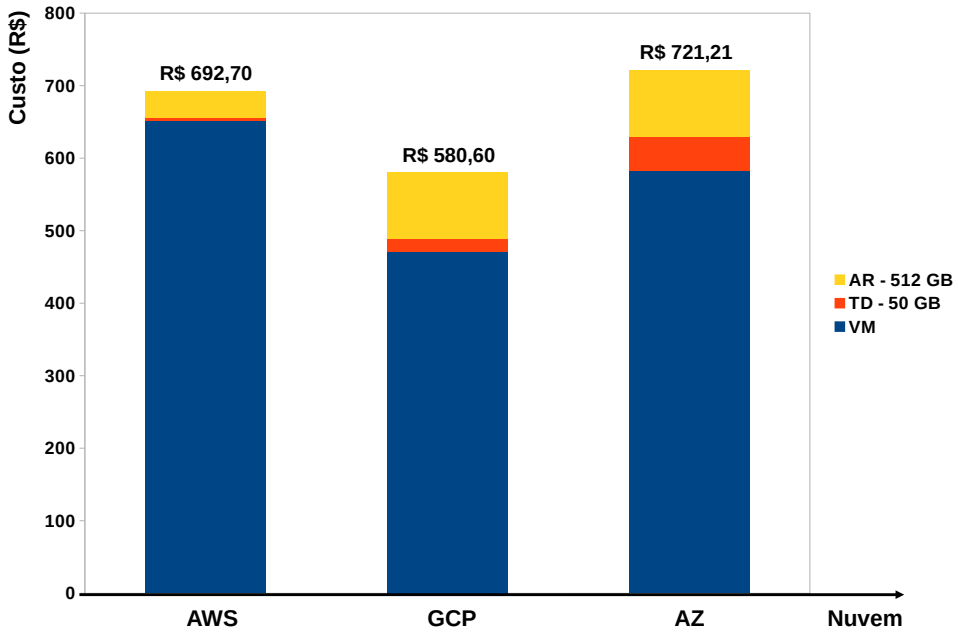
| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 320,15 | 4,73 | **72,59** | 397,47 |
| GCP | 231,21 | 18,90 | 187,91 | 438,02 |
| AZ | 287,07 | 46,08 | 190,64 | 523,79 |

**Fonte:** Elaborada pelo autor (2023).

### 4.2.2 Instâncias sob demanda (*Sock Shop*)

Os resultados com a *Sock Shop* foram semelhantes aos da *Hipster Shop*. A Figura 11 e a Tabela 7 apresentam os custos mensais quando o mecanismo de migração está desativado.

Figura 11 – Pagamento mensal (Desativado, *Sock Shop, VM XLarge*, N = 10.000)




**Fonte:** Elaborada pelo autor (2023).

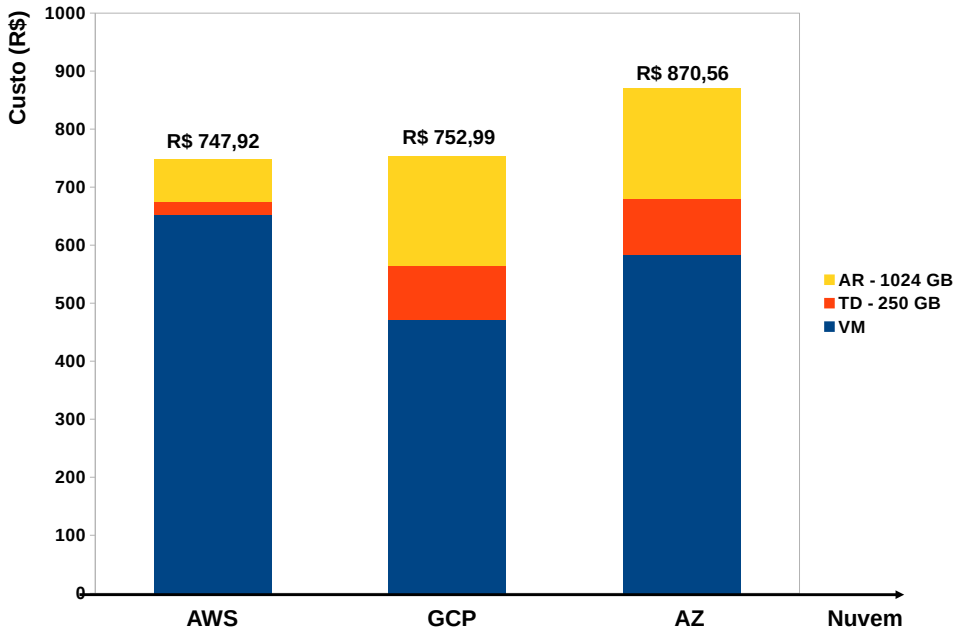
Tabela 7 – Detalhes de custo (R$) do cenário 4.2.2 e Mecanismo de migração Desativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 651,67 | 4,73 | 36,30 | 692,70 |
| **GCP** | **470,58** | **18,90** | **91,12** | **580,60** |
| AZ | 582,84 | 46,08 | 92,29 | 721,21 |

**Fonte:** Elaborada pelo autor (2023).

Quando o mecanismo de adaptação foi ativado e a MBA foi implantada inicialmente no provedor mais barato (GCP), os custos mensais são os mostrados na Figura 11. À medida que o número de solicitações aumentou, COSTA identificou um crescimento no uso de persistência de dados e na transferência de dados de saída que elevou o custo da MBA. Em seguida, o Planejador foi ativado e criou um plano de migração para reduzir o custo movendo a MBA da GCP para um provedor mais barato (AWS). Com essa migração, o custo mensal da *Sock Shop* foi de R$ 747,92, ou seja, uma economia de R$ 5,07, conforme a Figura 12 e a Tabela 8.

Figura 12 – Pagamento mensal (Ativado, *Sock Shop, VM XLarge*, GCP → AWS, N = 60.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 8 – Detalhes de custo (R$) do cenário 4.2.2 e Mecanismo de migração Ativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 651,67 | **23,66** | **72,59** | **747,92** |
| GCP | 470,58 | 94,50 | 187,91 | 752,99 |
| AZ | 582,84 | 97,08 | 190,64 | 870,56 |

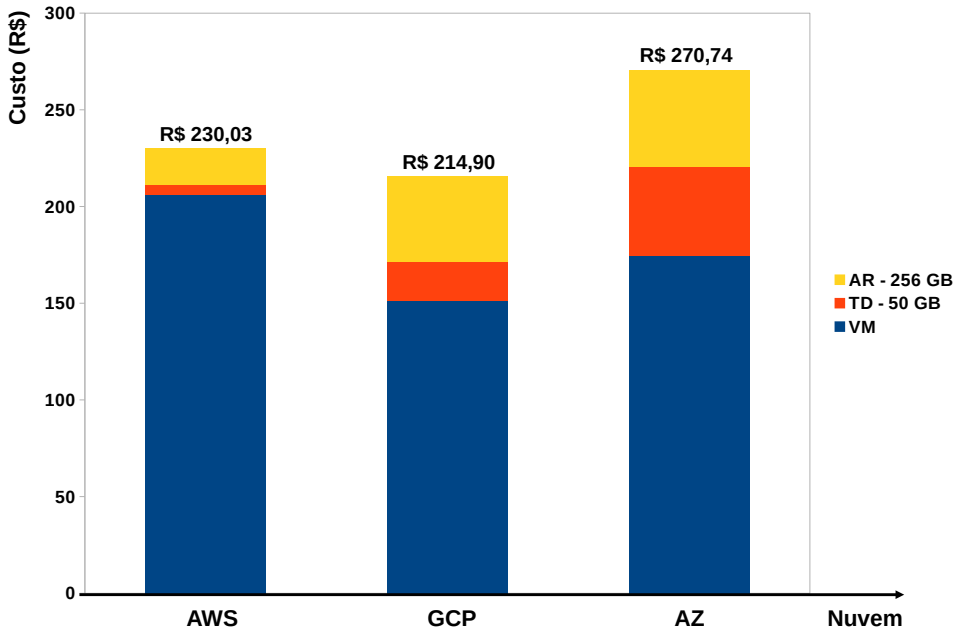
**Fonte:** Elaborada pelo autor (2023).

### 4.2.3 Instâncias com reservas de um ano (*Hipster Shop*)

Nos experimentos envolvendo a *Hipster Shop* com instâncias com reservas de um ano, os custos para executar a MBA por um mês com o mecanismo de adaptação desativado são mostrados na Figura 13 e detalhados na Tabela 9.

Quando o mecanismo de adaptação foi ativado e a MBA foi implantada inicialmente no provedor mais barato (GCP), os custos mensais são os mostrados na Figura 13. À medida que o número de solicitações aumentou, COSTA identificou um crescimento no uso de persistência de dados que elevou o custo da MBA. Em seguida, o Planejador foi ativado e criou um plano de migração para reduzir o custo movendo a MBA da GCP para um provedor mais barato

Figura 13 – Pagamento mensal (Desativado, *Hipster Shop, VM Large*, N = 10.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 9 – Detalhes de custo (R$) do cenário 4.2.3 e Mecanismo de migração Desativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 206,42 | 4,73 | 18,88 | 230,03 |
| **GCP** | **151,55** | **18,90** | **44,45** | **214,90** |
| AZ | 174,56 | 46,08 | 50,10 | 270,74 |

**Fonte:** Elaborada pelo autor (2023).

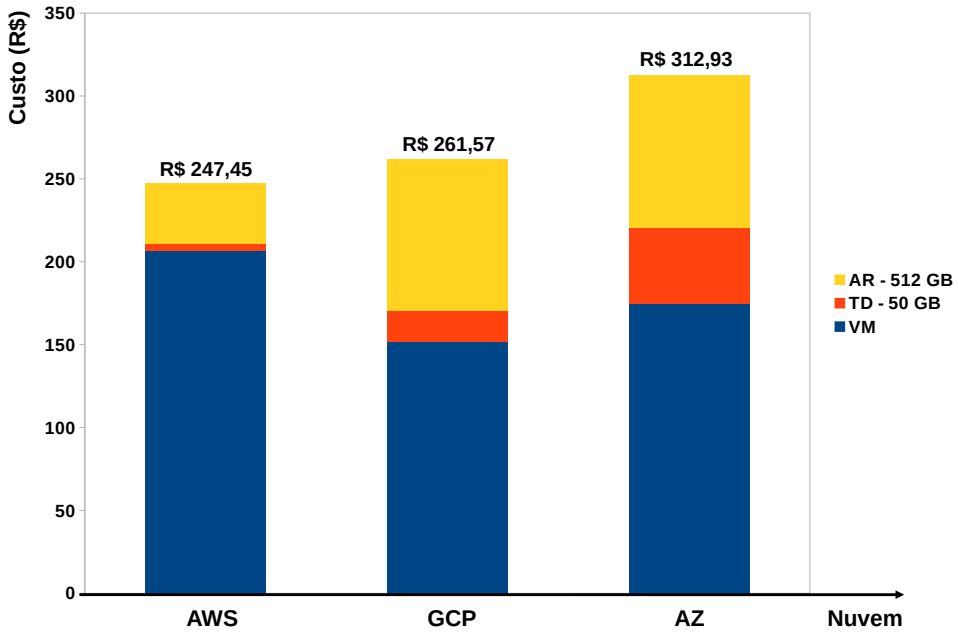
(AWS). Com essa migração, o custo mensal da *Hipster Shop* foi de R$ 247,45, *i.e.*, uma economia de R$ 14,12, conforme Figura 14 e Tabela 10.

### 4.2.4 Instâncias com reservas de um ano (*Sock Shop*)

Os experimentos com a *Sock Shop* foram semelhantes aos da *Hipster Shop*. A Figura 15 e a Tabela 11 apresentam os custos mensais quando o mecanismo de migração foi desativado.

Quando o mecanismo de adaptação foi ativado e a MBA foi implantada inicialmente no provedor mais barato (GCP), os custos mensais são mostrados na Figura 15. À medida que o número de solicitações aumentou, COSTA identificou um crescimento no uso de persistência

Figura 14 – Pagamento mensal (Ativado, *Hipster Shop, VM Large*, GCP → AWS, N = 60.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 10 – Detalhes de custo (R$) do cenário 4.2.3 e Mecanismo de migração Ativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 206,42 | 4,73 | **36,30** | 247,45 |
| GCP | 151,55 | 18,90 | 91,12 | 261,57 |
| AZ | 174,56 | 46,08 | 92,29 | 312,93 |

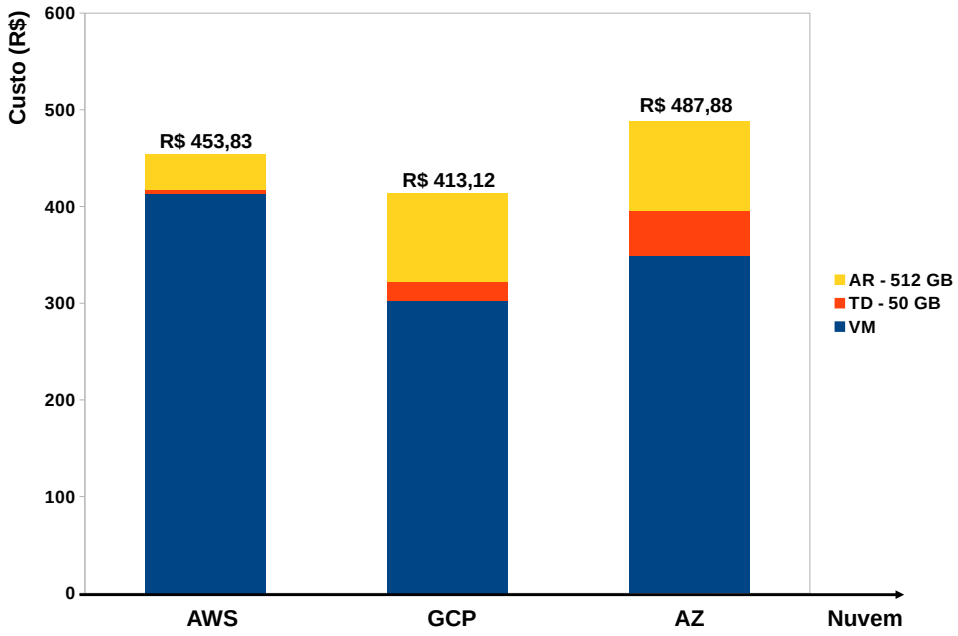
**Fonte:** Elaborada pelo autor (2023).

de dados que elevou o custo da MBA. Em seguida, o Planejador foi ativado e criou um plano de migração para reduzir o custo movendo a MBA da GCP para um provedor mais barato (AWS). Com essa migração, o custo mensal da *Sock Shop* foi de R$ 490,12, ou seja, uma economia de R$ 19,79, conforme Figura 16 e Tabela 12.

### 4.2.5 Instâncias com reservas de três anos (*Hipster Shop*)

Nos experimentos envolvendo a *Hipster Shop* com instâncias com reservas de três anos, os custos para executar a MBA por um mês com o mecanismo de adaptação desativado são mostrados na Figura 17 e detalhados na Tabela 13.

Figura 15 – Pagamento mensal (Desativado, *Sock Shop, VM XLarge*, N = 10.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 11 – Detalhes de custo (R$) do cenário 4.2.4 e Mecanismo de migração Desativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 412,80 | 4,73 | 36,30 | 453,83 |
| **GCP** | **303,10** | **18,90** | **91,12** | **413,12** |
| AZ | 349,51 | 46,08 | 92,29 | 487,88 |

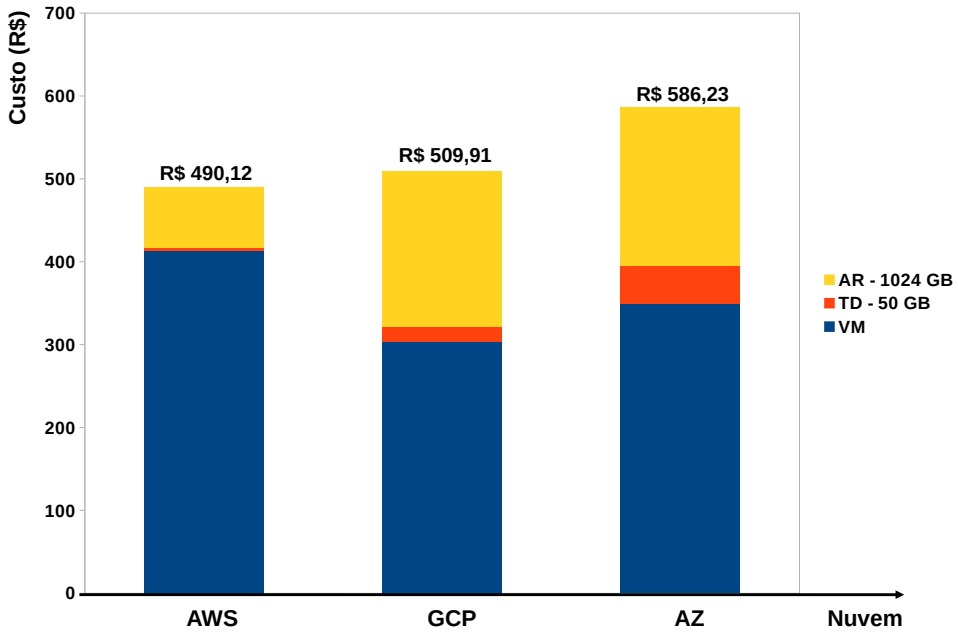
**Fonte:** Elaborada pelo autor (2023).

Quando o mecanismo de adaptação foi ativado e a MBA foi implantada inicialmente no provedor mais barato (AWS), os custos mensais são os mostrados na Figura 17. À medida que o número de solicitações aumentou, COSTA não identificou cenários onde a migração pudesse reduzir os custos. Portanto, foi vantajoso permanecer na AWS, como mostra a Figura 18 e a Tabela 14.

### 4.2.6 Instâncias com reservas de três anos (*Sock Shop*)

Nos experimentos envolvendo a *Sock Shop* com instâncias com reservas de três anos, os custos para executar a MBA por um mês com o mecanismo de adaptação desativado foram

Figura 16 – Pagamento mensal (Ativado, *Sock Shop, VM XLarge*, GCP → AWS, N = 60.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 12 – Detalhes de custo (R$) do cenário 4.2.4 e Mecanismo de migração Ativado

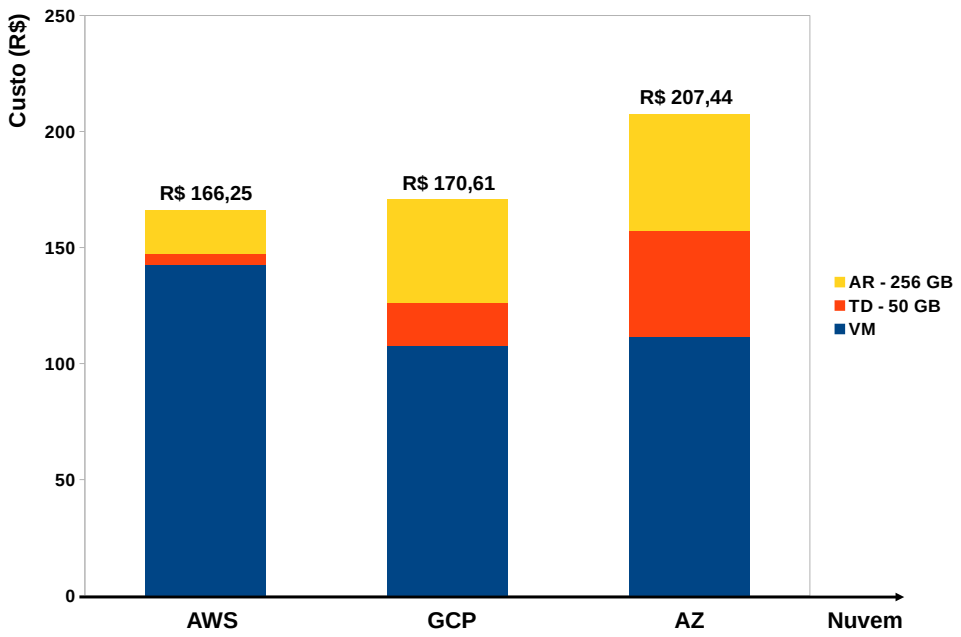
| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 412,80 | 4,73 | **72,59** | 490,12 |
| GCP | 303,10 | 18,90 | 187,91 | 509,91 |
| AZ | 349,51 | 46,08 | 190,64 | 586,23 |

**Fonte:** Elaborada pelo autor (2023).

os mostrados na Figura 19 e detalhados na Tabela 15.

Quando o mecanismo de adaptação foi ativado e a MBA foi implantada inicialmente no provedor mais barato (GCP), os custos mensais são mostrados na Figura 19. À medida que o número de solicitações aumentou, COSTA não identificou cenários onde a migração pudesse reduzir os custos. Portanto, foi vantajoso permanecer na GCP como mostra a Figura 20 e Tabela 16.

Figura 17 – Pagamento mensal (Desativado, *Hipster Shop, VM Large*, N = 10.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 13 – Detalhes de custo (R$) do cenário 4.2.5 e Mecanismo de migração Desativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| **AWS** | **142,64** | **4,73** | **18,88** | **166,25** |
| GCP | 107,26 | 18,90 | 44,45 | 170,61 |
| AZ | 111,26 | 46,08 | 50,10 | 207,44 |

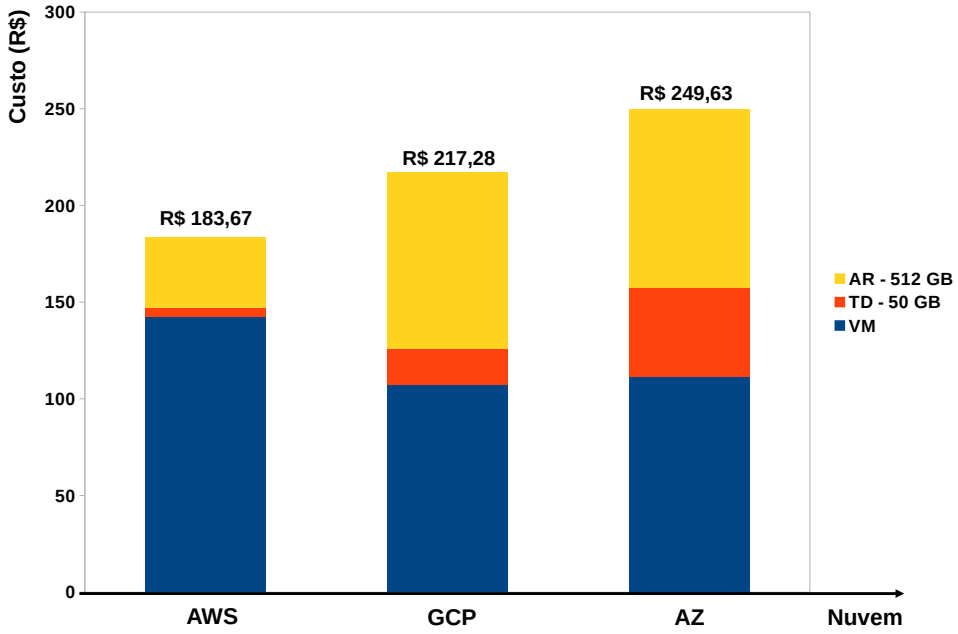
**Fonte:** Elaborada pelo autor (2023).

Tabela 14 – Detalhes de custo (R$) do cenário 4.2.5 e Mecanismo de migração Ativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| **AWS** | **142,64** | **4,73** | **36,30** | **183,67** |
| GCP | 107,26 | 18,90 | 91,12 | 217,28 |
| AZ | 111,26 | 46,08 | 92,29 | 249,63 |

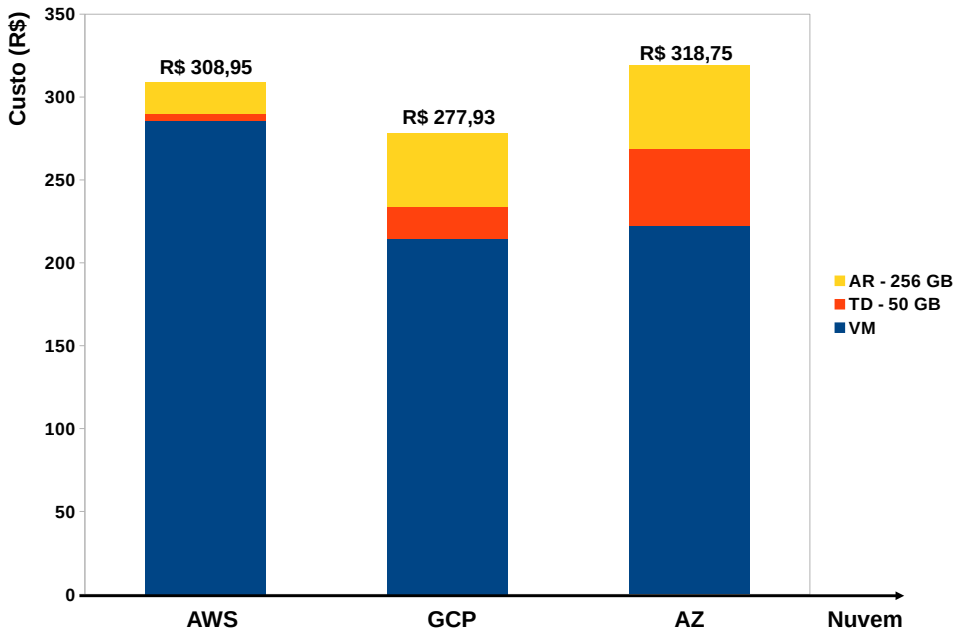
**Fonte:** Elaborada pelo autor (2023).

Figura 18 – Pagamento mensal (Ativado, *Hipster Shop, VM Large*, AWS, N = 60.000)




**Fonte:** Elaborada pelo autor (2023).

Figura 19 – Pagamento mensal (Desativado, *Sock Shop, VM XLarge*, N = 10.000)



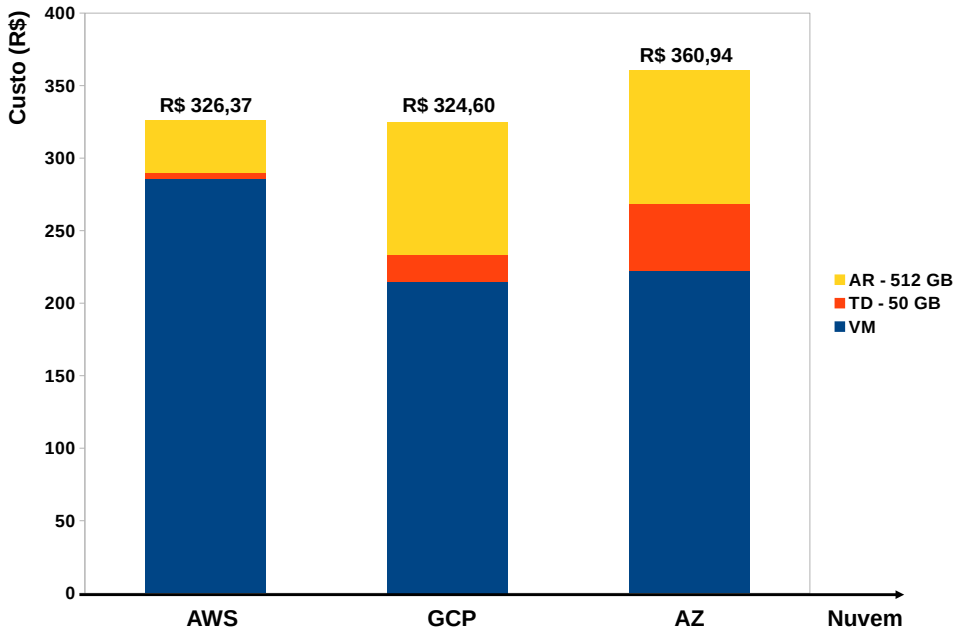

**Fonte:** Elaborada pelo autor (2023).

Tabela 15 – Detalhes de custo (R$) do cenário 4.2.6 e Mecanismo de migração Desativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 285,34 | 4,73 | 18,88 | 308,95 |
| **GCP** | **214,58** | **18,90** | **44,45** | **277,93** |
| AZ | 222,57 | 46,08 | 50,10 | 318,75 |

**Fonte:** Elaborada pelo autor (2023).

Figura 20 – Pagamento mensal (Ativado, *Sock Shop, VM XLarge*, GCP, N = 60.000)




**Fonte:** Elaborada pelo autor (2023).

Tabela 16 – Detalhes de custo (R$) do cenário 4.2.6 e Mecanismo de migração Ativado

| Nuvem | VM | Transferência de dados (TD) | Armazenamento (AR) | Total |
|---|---|---|---|---|
| AWS | 285,34 | 4,73 | 36,30 | 326,37 |
| **GCP** | **214,58** | **18,90** | **91,12** | **324,60** |
| AZ | 222,57 | 46,08 | 92,29 | 360,94 |

**Fonte:** Elaborada pelo autor (2023).

## 4.3 ANÁLISE DA REDUÇÃO DOS CUSTOS DE CADA AVALIAÇÃO EXPERIMENTAL

A Tabela 17 fornece os resultados obtidos de cada avaliação experimental. Como pode ser visto na Tabela 17, o **Cenário 1** iniciou o experimento na GCP e, em seguida, migrou para o provedor mais barato (AWS) para reduzir o custo. Portanto, a migração da GCP para AWS gerou uma economia de R$ 40,55, ou seja, uma redução em 9,26% do custo. O **Cenário 2** iniciou o experimento na GCP e, em seguida, migrou para o provedor AWS para reduzir o custo, logo, a migração da GCP para AWS gerou uma economia de R$ 5,07, isto é, uma redução em 0,67% do custo. O **Cenário 3** iniciou na nuvem GCP e migrou para AWS para diminuir o custo, logo, a migração da GCP para AWS gerou uma economia de R$ 14,12, uma redução em 5,40% do custo. O **Cenário 4** iniciou o experimento na GCP e migrou para AWS para reduzir o custo, logo, a migração da GCP para AWS gerou uma economia de R$ 19,79, uma redução em 3,88% no custo. No **Cenário 5**, o experimento iniciou na AWS e permaneceu na AWS, gerando uma economia entre R$ 33,61 (GCP) e R$ 65,96 (AZ), uma redução entre 15,47% (GCP) e 26,42% (AZ) do custo. Caso houvesse uma migração da AWS para outra nuvem, o aumento do custo da migração da AWS para GCP seria de R$ 33,61, e, caso houvesse uma migração da AWS para AZ, o aumento do custo seria de R$ 65,96. No **Cenário 6**, o experimento iniciou na GCP e permaneceu na GCP, gerando uma economia entre R$ 1,77 (AWS) e R$ 36,34 (AZ), uma redução entre 0,54% (AWS) e 10,07% (AZ) do custo. Caso houvesse uma migração, o aumento do custo da migração da GCP para AWS seria de R$ 1,77, e, caso houvesse uma migração da GCP para AZ, o aumento do custo seria de R$ 36,34.

Tabela 17 – Apresentação dos resultados de cada avaliação experimental

| Cenário | Experimento | Migração | Economia (R$) | Redução (%) |
|---|---|---|---|---|
| 1 | 4.2.1 Instâncias sob demanda (HS) | GCP $\rightarrow$ AWS | De 40,55 | 9,26% |
| 2 | 4.2.2 Instâncias sob demanda (SS) | GCP $\rightarrow$ AWS | De 5,07 | 0,67% |
| 3 | 4.2.3 Instâncias com reservas de um ano (HS) | GCP $\rightarrow$ AWS | De 14,12 | 5,40% |
| 4 | 4.2.4 Instâncias com reservas de um ano (SS) | GCP $\rightarrow$ AWS | De 19,79 | 3,88% |
| 5 | 4.2.5 Instâncias com reservas de três anos (HS) | Sem (permaneceu na AWS) | Entre 33,61 e 65,96 | Entre 15,47% e 26,42% |
| 6 | 4.2.6 Instâncias com reservas de três anos (SS) | Sem (permaneceu na GCP) | Entre 1,77 e 36,34 | Entre 0,54% e 10,07% |

**Fonte:** Elaborada pelo autor (2023).

## 4.4 CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentada a avaliação experimental da solução COSTA. Inicialmente, foi apresentada uma metodologia para elaboração dos experimentos. Em seguida, os experimentos foram executados nos planos de instâncias com reservas (de um ano e três anos) e sob demanda. Os resultados mostram que COSTA conseguiu reduzir os custos das MBAs em nuvem efetivamente, avaliando os fatores de custos, por exemplo, de transferência de dados de saída e armazenamento de dados.

# 5 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados ao problema investigado nesta dissertação. A Seção 5.1 descreve soluções que lidam com a migração de aplicações entre provedores de nuvem. A Seção 5.2 apresenta soluções que migram VMs entre provedores de nuvem. A Seção 5.3 detalha soluções multinuvem, mas não aborda o conceito de migração entre nuvens. A Seção 5.4 mostra uma avaliação comparativa entre os trabalhos relacionados e o trabalho proposto nesta dissertação.

## 5.1 MIGRAÇÃO ENTRE NUVENS

A solução apresentada em Pellegrini, Rottmann e Strieder (2017) migra aplicações entre diferentes provedores de serviços em nuvem para evitar o bloqueio do fornecimento de serviço, *e.g.*, quando um serviço de nuvem apresenta problemas operacionais ou fica indisponível. Assim, a solução oferece suporte a recursos-chave, como portabilidade, interoperabilidade e independência de plataforma. A solução foi projetada seguindo o conceito de uma arquitetura de microsserviços para separar as responsabilidades dos serviços. Finalmente, os contêineres são usados para encapsular a aplicação e suas dependências.

O *MOdel-Driven Approach for the design and execution of applications on multiple Clouds* (MODACLOUDS) (ARDAGNA *et al.*, 2012) auxilia os desenvolvedores e operadores de sistemas, fornecendo ferramentas para todas as fases do ciclo de vida do sistema de computação em nuvem. Ele é usado para desenvolver, implantar, executar e adaptar aplicações em várias nuvens. O MODACLOUDS elimina a necessidade de os desenvolvedores comprometerem-se com as tecnologias de nuvem durante o desenvolvimento de *software* e aumenta as oportunidades de atingir as metas de negócios em termos de redução de custos, gerenciamento de riscos, *Quality of Service* (QoS) e flexibilidade do processo de desenvolvimento.

A *Apache jclouds* (APACHE, 2023d) é uma biblioteca Java de código aberto que permite criar aplicações portáteis entre nuvens, ao mesmo tempo em que oferece controle total para usar recursos específicos da nuvem. Atualmente, a biblioteca oferece suporte a mais de trinta provedores de nuvem, incluindo AWS, *OpenStack* e *Azure*.

O *Checkpoint/Restore In Userspace* (CRIU) (CRIU, 2023) e a solução proposta em Mirkin, Kuznetsov e Kolyshkin (2008) são implementados com base em verificação e reinicialização,

permitindo a movimentação de contêineres entre os servidores. A migração consiste em salvar o estado da memória de um processo em um arquivo e reiniciar o processo no *host* destino a partir de um ponto de verificação. Os contêineres são mais leves do que as máquinas virtuais, portanto, a migração do contêiner é menos problemática do que a das máquinas virtuais. Por exemplo, para os contêineres que executam serviços sem estado (*e.g.,* serviços da *Web RESTful*), é possível eliminar diretamente o contêiner no *host* de origem e criar um novo contêiner no *host* de destino. No entanto, este período é aceitável e apenas as requisições nas aplicações em andamento são afetadas.

## 5.2 MIGRAÇÃO DE VMS

Kumo (XAVIER; FERRAZ; SETTE, 2018) move máquinas virtuais entre diferentes provedores de nuvem para evitar o bloqueio do provedor. Esta solução resolve o problema de máquinas virtuais criadas em ambientes de nuvem, fornecendo a portabilidade de VMs entre diferentes nuvens de tecnologia heterogênea, expondo uma API *web* e *drivers* responsáveis pela migração. Na prática, tudo instalado na VM da nuvem de origem é migrado para a VM da nuvem de destino. A exposição da API *web* permite o registro de máquinas virtuais e ações, *e.g.,* migrar. *Drivers* é um código que executa as operações necessárias nas nuvens de origem e destino.

O agendador de nuvem (LLORENTE, 2011) aborda o desafio de implantação dinâmica. O agendador leva em consideração o histórico de preços dos provedores de nuvem disponíveis. Em seguida, calcula sua média, a tendência e prevê a melhor implantação na próxima hora. O objetivo do agendador é diminuir o investimento do usuário enquanto mantém um desempenho do *cluster* virtual gerenciado. Para atingir este objetivo, foi aproveitada a variabilidade de preços ofertados pelos provedores de nuvem, movendo parte do *cluster* ou o *cluster* inteiro para o provedor mais barato.

*Load Level based OpimizatiOn for VIrtual machine Allocation* (LLOOVIA) (GARCíA, 2017) é um método de alocação de máquinas virtuais que gerencia períodos de um ano para aproveitar os descontos fornecidos por máquinas virtuais reservadas em ambiente de várias nuvens. Esta solução é formulada como um problema de otimização que tem como entrada o nível de desempenho alcançável do serviço e produz a alocação de VM como saída. As alocações calculadas reduzem os custos do serviço, garantem o desempenho necessário e cumprem os limites estabelecidos pelos provedores de serviços. LLOOVIA reduz custos ao mover máquinas virtuais entre nuvens.

O algoritmo proposto em Niyato (2009) posiciona máquinas virtuais (*Optimal Virtual Machine Placement Algorithm* (OVMP)). Esse algoritmo possibilita a redução dos custos das máquinas virtuais em um ambiente de vários provedores de nuvem nos planos de reserva antecipada e sob demanda. O algoritmo OVMP toma decisões com base em soluções de programação inteira probabilística (*Stochastic Integer Programming* (SIP)), ideais para alugar recursos de provedores de serviços em nuvem. O desempenho do algoritmo OVMP é avaliado através de estudos numéricos e simulações. Os resultados mostram que o algoritmo OVMP proposto pode reduzir o orçamento dos usuários. Esse algoritmo pode ser usado para provisionar recursos em novos ambientes de computação em nuvem.

O modelo de previsão de falhas proposto por Alam (2020) é baseado em Cadeias de Markov para prever falhas de servidores em nuvem. Este modelo prevê a degradação do desempenho do servidor com base em dados históricos. A previsão de confiabilidade do servidor é então incorporada à abordagem de migração de VM (com base no algoritmo *Artificial Bee Colony* (ABC)) em configurações de várias nuvens para otimizar a tolerância a falhas, aumentar a confiabilidade da nuvem e reduzir a latência de comunicação.

## 5.3 SOLUÇÕES QUE NÃO MIGRAM

A plataforma de gerenciamento *LambdaLink* (KEAHEY; RITEAU; TIMKOVICH, 2017) permite armazenar e compartilhar imagens em vários provedores de nuvem. Ela também oferece suporte nativo a máquinas virtuais, *bare metal* e contêineres *Docker*. A proposta inclui os componentes portal, registro de dispositivo, gerente de recursos, agente do dispositivo, registro *lambda*, gerente *lambda* e agente *lambda*. Esses componentes permitem aos administradores da plataforma *LambdaLink* registrarem imagens de disco virtual que podem ser usadas para iniciar instâncias em vários serviços de nuvem. No entanto, esta plataforma não resolve o problema de mover máquinas virtuais e contêineres entre diferentes provedores de nuvem.

A *API Anthos* (GOOGLE, 2022) multinuvem permite o provisionamento e gerenciamento de *clusters* do *Google Kubernetes Engine* (GKE) em execução na infraestrutura da AWS e *Azure*, utilizando um plano de controle centralizado do *Google Cloud*. Isso significa que uma equipe pode criar, gerenciar e atualizar *clusters* do GKE, independentemente da nuvem pública que esteja usando.

*Multicloud Management Cloud Pak* (IBM, 2022) implementa o IBM *Cloud Automation Manager* (CAM) em *cluster OpenShift*. O CAM provisiona aplicações baseadas em máquinas

virtuais em várias nuvens híbridas, permitindo registrar provedores de nuvem pública adicionais, como *IBM Cloud, Amazon EC2, Microsoft Azure* e *Google Cloud*. Depois de adicionar um provedor de nuvem, é possível configurar modelos baseados em *Terraform* que definem como uma VM deve parecer no ambiente de destino.

*Apache Libcloud* (APACHE, 2023b) é uma biblioteca *Python* que oculta as diferenças entre diferentes APIs de provedores de nuvem e permite gerenciar diferentes recursos de nuvem por meio de uma API unificada e fácil de usar.

*Deltacloud* (APACHE, 2023c) fornece APIs e *drivers* necessários para conectar-se a provedores de nuvem. Ela fornece estabilidade e compatibilidade de longo prazo entre *scripts*, ferramentas e aplicações, permitindo a gerência de recursos em diferentes nuvens usando três APIs compatíveis. As APIs suportadas são a API *Deltacloud* clássica, a API DMTF CIMI ou API EC2. Assim, com o mesmo código, é possível iniciar instâncias em nuvem interna e pública ao mesmo tempo, *e.g., Amazon* EC2.

*EasyCloud* (ANGLANO; CANONICO; GUAZZONE, 2021) é um *kit* de ferramentas para criação e uso de sistemas multinuvem, fornecendo interoperabilidade, independência de plataforma, provisionamento de recursos e facilidade de uso. Ele oferece facilidade de uso por meio de uma API unificada que libera os usuários do aprendizado das diferentes APIs proprietárias expostas pelas várias plataformas de nuvem.

## 5.4 AVALIAÇÃO COMPARATIVA

A Tabela 18 fornece uma comparação entre os trabalhos relacionados citados neste capítulo. Esta tabela é baseada na proposta de Salehie e Tahvildari (2009) para analisar sistemas adaptativos (Seção 2.3). Além das questões sobre para onde migrar, quando migrar, o que migrar, por que migrar e como migrar, foi adicionada a coluna Gestão de Custo (GC) para destacar os trabalho que lidam com o gerenciamento de custo.

Como pode ser visto na Tabela 18, apesar de cinco propostas (ALAM, 2020; GARCíA, 2017; NIYATO, 2009; LLORENTE, 2011; XAVIER; FERRAZ; SETTE, 2018) lidarem com a migração de máquinas virtuais, apenas três levaram em consideração o custo. Além disso, três outras propostas (APACHE, 2023d; ARDAGNA *et al.*, 2012; PELLEGRINI; ROTTMANN; STRIEDER, 2017) promovem a portabilidade de aplicações entre as nuvens, mas sem considerar custos. Outras seis propostas (APACHE, 2023c; APACHE, 2023b; ANGLANO; CANONICO; GUAZZONE, 2021; GOOGLE, 2022; IBM, 2022; KEAHEY; RITEAU; TIMKOVICH, 2017) não oferecem suporte à transferência de

recursos entre provedores de nuvem, mas oferecem facilidade de uso ao tornar transparentes as várias APIs expostas pelos proprietários da nuvem. Duas propostas (MIRKIN; KUZNETSOV; KOLYSHKIN, 2008; CRIU, 2023) abordam a movimentação de contêineres entre *data centers*. Isso foi adicionado à Tabela 18 porque as aplicações podem ser encapsuladas em contêineres para transporte entre provedores de nuvem, como realizado em Pellegrini, Rottmann e Strieder (2017).

Dessas dezesseis propostas, apenas quatro (LLORENTE, 2011; GARCíA, 2017; NIYATO, 2009; COSTA) abordam o uso da gestão de custos, enquanto as demais não levam em consideração os custos gerados em nuvem. Ao avaliar as quatro propostas citadas anteriormente, percebe-se que três abordam a migração de máquinas virtuais (LLORENTE, 2011; GARCíA, 2017; NIYATO, 2009) e apenas uma solução oferece suporte à migração de MBAs entre provedores de nuvem (COSTA). Porém, apenas COSTA é um ambiente de sistemas adaptativos implementado usando o *feedback loop* MAPE-K.

Nesse contexto, COSTA se destaca por ser um ambiente de sistemas adaptativos e genérico que aborda o gerenciamento de custos em nuvem. Atualmente, COSTA é a única solução adaptativa que aborda a migração de MBAs entre diferentes provedores de nuvem, construída com base em um *feedback loop* comumente usado em soluções adaptativas. Por fim, é ainda uma proposta que aplica gerenciamento de custos para se beneficiar dos diferentes esquemas de preços oferecidos pelas plataformas de nuvem.

Tabela 18 – Sumário das soluções relacionadas

| Autor/Solução | Onde | Quando | O que | Por que | Como | GC |
|---|---|---|---|---|---|---|
| (PELLEGRINI; ROTTMANN; STRIEDER, 2017) | Nuvem | Limitado ou indisponibilidade | Aplicação | *vendor lock-ins* | Contêiner | Não |
| (ARDAGNA *et al.*, 2012) | Nuvem | Baixo desempenho | Aplicação | *vendor lock-ins* | REMICS | Não |
| Apache jclouds | Nuvem | - | Aplicação | - | Biblioteca | - |
| CRIU | *Data center* | - | Contêiner | - | - | - |
| (MIRKIN; KUZNETSOV; KOLYSHKIN, 2008) | *Data center* | - | Contêiner | - | - | - |
| (XAVIER; FERRAZ; SETTE, 2018) | Nuvem | - | VM | *vendor lock-ins* | API *web* e *drivers* | Não |
| (LLORENTE, 2011) | Nuvem | Tendência de custo | VM | Reduzir custo | *OpenNebula* | Sim |
| (GARCíA, 2017) | Nuvem | Descontos fornecidos por VMs reservadas | VM | Reduzir custo | Modelo | Sim |
| (NIYATO, 2009) | Nuvem | Probabilística (SIP) ideais | VM | Reduzir custo | Algoritmo | Sim |
| (ALAM, 2020) | Nuvem | Falhas e degradação | VM | Confiabilidade | Algoritmo ABC | Não |
| (KEAHEY; RITEAU; TIMKOVICH, 2017) | - | - | - | - | - | - |
| *Anthos* | - | - | - | - | - | - |
| *Multicloud Management Cloud Pak* | - | - | - | - | - | - |
| *Apache Libcloud* | - | - | - | - | - | - |
| *Deltacloud* | - | - | - | - | - | - |
| (ANGLANO; CANONICO; GUAZZONE, 2021) | - | - | - | - | - | - |
| COSTA | Nuvem | Violação, indisponibilidade e limite atingido | MBA | Reduzir custo | Sistemas adaptativos (*feedback loop*) | Sim |

**Fonte:** Elaborada pelo autor (2023).

## 5.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os trabalhos e soluções relacionados à migração de máquinas virtuais e aplicações. Soluções multinuvem que não lidam com migrações também foram apresentadas. Os trabalhos e soluções relacionados foram comparados com o trabalho proposto usando uma série de perguntas que capturam os requisitos dos sistemas adaptativos, como onde migrar.

# 6 CONCLUSÃO E TRABALHOS FUTUROS

Esse capítulo apresenta as contribuições do trabalho proposto, as suas limitações e o potencial para trabalhos futuros.

## 6.1 CONCLUSÃO

Esta dissertação focou na migração de MBAs entre diferentes provedores de serviços em nuvem, levando em consideração os fatores de custo que impulsionam a migração. Por esse motivo, soluções para evitar o bloqueio de fornecedores, ausência de serviços em nuvem ou economia de custos têm sido propostas na literatura. Porém, poucas soluções usam o gerenciamento de custo para tomar decisões durante o processo de migração entre provedores de nuvem. Além disso, nenhuma solução aplica conceitos de sistemas adaptativos.

Neste contexto, foi proposta COSTA, uma solução adaptativa para executar e gerenciar MBAs em ambientes de múltiplas nuvens. COSTA foi implementada visando a separação de responsabilidades entre os serviços que monitoram, analisam, planejam e executam a migração de aplicações entre vários provedores de nuvem baseados em critérios de custo. Ela realiza migrações sempre que detecta a possibilidade de redução do custo da aplicação ou quando o orçamento para executar a aplicação for atingido.

Vários experimentos foram realizados com as aplicações *Hipster shop* e *Sock shop*. Os experimentos foram executados em um ambiente de múltiplas nuvens para avaliar a capacidade de COSTA em reduzir custos. Como os modelos de precificação dos provedores de nuvem são diferentes, COSTA explorou a possibilidade de migração e conseguiu reduzir o custo de execução das MBAs.

## 6.2 CONTRIBUIÇÕES

A principal contribuição desta dissertação é a criação de COSTA, uma solução para executar e gerenciar MBAs em ambientes de múltiplas nuvens. COSTA implementa os componentes do MAPE-K como microsserviços que monitoram e migram aplicações entre vários provedores de nuvem usando critérios de custo. Na prática, COSTA pode realizar migrações periódicas sempre que detecta a possibilidade de redução do custo das MBAs ou quando o orçamento

para executar a MBA é violado.

A capacidade de COSTA em reduzir os custos foi avaliada com a execução de diversos experimentos nos planos sob demanda e em instâncias com reservas de um e três anos. Como os modelos de precificação dos provedores de nuvem são diferentes, COSTA explorou a possibilidade de migração e conseguiu reduzir o custo de execução das MBAs.

## 6.3 LIMITAÇÕES

COSTA possui algumas limitações:

- Atualmente é necessário codificar os arquivos *Terraform* e *Ansible* para cada nuvem integrada à COSTA. Além disso, é necessário implementar o *script* de orquestração do *Jenkins* para a realização do passo a passo do processo de migração. Portanto, é preciso conhecimento sobre as tecnologias utilizadas no processo implantação, migração e remoção das MBAs em nuvem;
- Não há banco de dados para armazenar o histórico de custo e uso. Ao analisar informações históricas, é possível descobrir novos cenários que levam à migração entre provedores de nuvem em tempo de execução. Portanto, é essencial que COSTA tenha um banco de dados (*e.g., Prometheus*) apto a coletar, minerar e processar métricas de uso e custo;
- COSTA é uma solução reativa, ou seja, para ocorrer uma migração entre nuvens é necessário um aumento do uso dos recursos computacionais (fatores de custos), *e.g.,* armazenamento em disco;
- Na versão atual de COSTA, as estimativas são geradas a partir da infraestrutura que será criada em nuvem. Portanto, seria muito útil se COSTA continuasse a gerar estimativas de custo após a implementação da MBA na nuvem, com base nas coletas de métricas e padrões de uso da MBA;
- A quantidade de nuvens integradas a COSTA para criar um ambiente experimental é limitada. À medida que COSTA integra mais nuvens, a variedade de esquemas de preços aumenta.

## 6.4 TRABALHOS FUTUROS

Após a conclusão desta dissertação, novos trabalhos podem dar continuidade a COSTA para torná-la uma solução mais robusta:

- Novas nuvens podem ser integradas à COSTA, diversificando ainda mais os esquemas de precificação;
- Pode haver a geração e envio de estimativas de custos das MBAs em nuvem após sua implantação;
- A avaliação experimental pode explorar novos cenários, por exemplo, cenários em que o custo é combinado com outros atributos de qualidade, como disponibilidade e segurança; e
- COSTA pode tornar-se proativa. Portanto, ao avaliar dados históricos de uso dos usuários, é possível identificar um padrão de utilização das aplicações e novos cenários de migração entre nuvens podem surgir.

# REFERÊNCIAS

ALAM, M. Z. T. H. A. H. A. B. M. B. Optimizing virtual machine migration in multi-clouds. In: *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. [S.l.: s.n.], 2020. p. 1–7.

AMAZON. *CE - AWS CLI 1.20.56 Command Reference*. 2023. Disponível em: <https://docs.aws.amazon.com/cli/latest/reference/ce/index.html>.

AMAZON EC2. *Recursos spot do Amazon EC2*. 2023. Disponível em: <https://aws.amazon.com/pt/ec2/spot/details/>.

AMAZON EC2 ON DEMAND. *Preço sob demanda do Amazon EC2*. 2023. Disponível em: <https://aws.amazon.com/pt/ec2/pricing/on-demand/>.

AMAZON EC2 RESERVED. *Preço de instâncias reservadas do Amazon EC2*. 2023. Disponível em: <https://aws.amazon.com/pt/ec2/pricing/reserved-instances/pricing/>.

AMAZON WEB SERVICES, INC. *AWS Budgets*. 2023. Disponível em: <https://docs.aws.amazon.com/aws-cost-management/latest/APIReference/API_Operations_AWS_Budgets.html.>

AMAZON WEB SERVICES, INC. *What is Amazon CloudWatch?* 2023. Disponível em: <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html>.

AMAZON WEB SERVICES, INC. *What is Amazon Route 53?* 2023. Disponível em: <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html>.

ANGLANO, C.; CANONICO, M.; GUAZZONE, M. Easycloud: Multi-clouds made easy. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. [S.l.: s.n.], 2021. p. 526–531.

APACHE. *Apache JMeter*. 2023. Disponível em: <https://jmeter.apache.org/>.

APACHE. *Apache Libcloud*. 2023. Disponível em: <https://libcloud.apache.org/>.

APACHE. *Deltacloud API*. 2023. Disponível em: <https://libcloud.apache.org/>.

APACHE. *JClouds*. 2023. Disponível em: <https://jclouds.apache.org/>.

ARDAGNA *et al.* Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In: *2012 4th International Workshop on Modeling in Software Engineering (MISE)*. Zurich, Switzerland: IEEE, 2012. p. 50–56.

ARMBRUST, A. F. R. G. A. D. J. R. K. A. K. G. L. D. P. A. R. I. S. M. Z. M. A view of cloud computing. ACM Transactions on Autonomous and Adaptive Systems, v. 53, n. 4, 2010.

ASHWINI, M. K. M. M. H. A. S. J. P. Pricing models and pricing schemes of iaas providers: a comparison study. ICACCI '12: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, p. 143–147, 2012.

BUYYA, A. N. T. R. N. C. R. Interconnected cloud computing environments: Challenges, taxonomy, and survey. ACM Computing Surveys, n. 47, 2014.

CARVALHO, T. Micros-serviços: características, benefícios e desvantagens em relação à arquitetura monolítica que impactam na decisão do uso desta arquitetura. II SEMINÁRIO DE DESENVOLVIMENTO EM SOA COM CLOUD COMPUTING E CONECTIVIDADE INSTITUTO NACIONAL DE TELECOMUNICAÇÕES – INATEL, 2016.

COMMUNITY JENKINS. *Jenkins Handbook*. 2023. Disponível em: <https://www.jenkins.io/doc/book/>.

CRIU. *CRIU*. 2023. Disponível em: <https://criu.org/Main_Page>.

DA, K.; DALMAU, M.; ROOSE, P. A Survey of adaptation systems. *International Journal on Internet and Distributed Computing Systems*, International journal on Internet and distributed computing systems, v. 2, n. 1, p. 1–18, nov. 2011. Disponível em: <https://hal.science/hal-00689773>.

DASCHNER. *Architecting Modern Java EE Aplications*. nsr: Published by Packt Publishing Ltd., 2017.

DOCKER INC. *Overview of Docker Hub*. 2023. Disponível em: <https://docs.docker.com/docker-hub/>.

EBERT *et al.* Devops. *IEEE Software*, v. 33, n. 3, p. 94–100, 2016.

ESCOFFIER. *Building Reactive Microservices in Java Asynchronous and Event-Based Application Design*. nsr: O'Reilly Media, Inc., 2017.

F5, INC. *HTTP Load Balancing*. 2023. Disponível em: <https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>.

FIELDING, R. T. *Architectural Styles and the Design of Network-based SoftwareArchitectures*. Tese (Doutorado) — University of California, Irvine, 2000.

FLEXERA. *2023 State of the Cloud Report*. 2023. Disponível em: <https://info.flexera.com/CM-REPORT-State-of-the-Cloud-2023-Thanks?revisit>. Acesso em: 19 set. 2023.

GARCíA, J. L. D. J. E. M. G. J. G. D. F. Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. In: *Future Generation Computer Systems*. Oviedo, Spain: Elsevier BV, 2017. v. 71, p. 129–144.

GITHUB. *GitHub Docs*. 2023. Disponível em: <https://docs.github.com/pt>.

GOOGLE. *Anthos Modern Application Platform*. 2022. Disponível em: <https://cloud.google.com/anthos/>.

GOOGLE. *Export Cloud Billing data to BigQuery*. 2023. Disponível em: <https://cloud.google.com/billing/docs/how-to/export-data-bigquery>.

GOOGLE CLOUD. *Create and manage notification channels by API*. 2023. Disponível em: <https://cloud.google.com/monitoring/alerts/using-channels-api.>

GOOGLE CLOUD. *Get started with the Cloud Billing Budget API*. 2023. Disponível em: <https://cloud.google.com/billing/docs/how-to/budget-api-overview.>

GOOGLE CLOUD. *Preemptible VM instances*. 2023. Disponível em: <https://cloud.google.com/compute/docs/instances/preemptible>.

GOOGLE CLOUD COMPUTE ENGINE. *Compute Engine pricing*. 2023. Disponível em: <https://cloud.google.com/compute/all-pricing>.

GOOGLE INC. *Stackdriver Monitoring API. Introduction to Cloud Computing Monitoring API*. 2023. Disponível em: <https://cloud.google.com/monitoring/api/v3/>.

GRADLE INC. *Gradle Build Tool*. 2023. Disponível em: <https://gradle.org/>.

HASHICORP. *TerraForm*. 2023. Disponível em: <https://www.terraform.io/>.

HUEBSCHER, M. C.; MCCANN, J. A. A survey of autonomic computing—degrees, models, and applications. ACM Computing Surveys, n. 40, 2008.

IBM. *An Architectural Blueprint for Autonomic Computing*. [S.l.], 2005.

IBM. *How Red Hat OpenShift can support your hybrid multicloud environment - Servers & Storage*. 2022. Disponível em: <https://www.ibm.com/blogs/systems/how-red-hat-openshift-can-support-your-hybrid-multicloud-environment/>.

INFRACOST INC. *Get started*. 2023. Disponível em: <https://www.infracost.io/docs/>.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. [S.l.]: Wiley, 1991.

KEAHEY, K.; RITEAU, P.; TIMKOVICH, N. P. LambdaLink: An Operation Management Platform for Multi-Cloud Environments. In: *Proceedings of The10th International Conference on Utility and Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2017. (UCC '17), p. 39–46. ISBN 9781450351492.

LEMOS *et al.* Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In: _____. *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 1–32. ISBN 978-3-642-35813-5.

LEWIS; FOWLER. Microservices: A definition of this new architectural term. 2014. Disponível em: <https://martinfowler.com/articles/microservices.html.>

LIGHTSTEP. *Hipster Shop: Cloud-Native Observability Demo Application*. 2023. Disponível em: <https://github.com/lightstep/hipster-shop>.

LLORENTE, J. L. L. S. R. M.-V. R. S. M. I. M. Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In: *2011 International Conference on High Performance Computing & Simulation*. Istanbul, Turkey: IEEE, 2011. p. 1–7.

MADADIPOUYA, F. F. M. M. B. R. M. A. T. K. K. Cloud computing: Vision, architecture and characteristics. 2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC), n. 8, 2015.

MADAMANCHI, S. *Google Cloud for DevOps Engineers: A practical guide to SRE and achieving Google's Professional Cloud DevOps Engineer certification*. [S.l.]: Packt Publishing, 2021.

MALATHI, M. Cloud computing concepts. 2011 3rd International Conference on Electronics Computer Technology, 2011.

MELL, P.; GRANCE, T. The nist definition of cloud computing. NIST SP 800-145, n. 47, 2011.

MICROSOFT. *Azure Billing REST API*. 2023. Disponível em: <https://docs.microsoft.com/en-us/rest/api/billing/>.

MICROSOFT. *Azure Monitor Metrics - List*. 2023. Disponível em: <https://docs.microsoft.com/pt-br/rest/api/monitor/metrics/list>.

MICROSOFT. *Budgets*. 2023. Disponível em: <https://learn.microsoft.com/en-us/rest/api/consumption/budgets.>

MICROSOFT AZURE. *Linux Virtual Machines Pricing*. 2023. Disponível em: <https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/>.

MICROSOFT AZURE RESERVED. *Azure Reserved Virtual Machine Instances*. 2023. Disponível em: <https://azure.microsoft.com/en-us/pricing/reserved-vm-instances/>.

MIRKIN, A.; KUZNETSOV, A.; KOLYSHKIN, K. Containers checkpointing and live migration. In: *Proc. Linux Symp.* nsr: Proc. Linux Symp., 2008. p. 85–90. In Portuguese.

MOURA *et al.* Dynamic Support for Distributed Auto-Adaptive Applications. In: *Proceedings of the 22nd International Conference on Distributed Computing Systems*. USA: IEEE Computer Society, 2002. (ICDCSW '02), p. 451–458. ISBN 0769515886.

NARAHARI, Y.; RAJU, C. V. L.; RAVIKUMAR, K. Dynamic pricing models for electronic Business. In: *Sadhana 30*. Bengaluru, India: Springer, 2005. p. 231 – 256.

NIYATO, S. C. B.-S. L. D. Optimal virtual machine placement across multiple cloud providers. In: *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. Singapore: IEEE, 2009. p. 103–110.

PELLEGRINI, R.; ROTTMANN, P.; STRIEDER, G. Preventing vendor lock-ins via an interoperable multi-cloud deployment approach. In: *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. Cambridge, UK: IEEE, 2017. p. 382–387.

PORTELLA, G. J. *Cloud Computing Pricing for Permanent and Transient Instances: Modeling and Forecasting*. Tese (Doutorado) — Department of Computer Science, University of Brasília, 2021.

PORTELLA, G. J. *Precificação em computação em nuvem para instâncias permanentes e transientes : modelagem e previsão*. Tese (Doutorado) — Universidade de Brasília, Brasília, 2021.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. 2023. Disponível em: <https://www.postgresql.org/.>

RED HAT. *Ansible Documentation*. 2023. Disponível em: <https://docs.ansible.com/ansible/latest/index.html>.

SALEHIE; TAHVILDARI. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems, v. 4, n. 2, 2009.

SHARMA, N. S. C. D. Secure proof of retrievability system in cloud for data integrity. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), n. 1, 2018.

SHASTRI, S.; RIZK, A.; IRWIN, D. Transient guarantees: Maximizing the value of idle cloud capacity. SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, n. 18, 2016.

SMARTBEAR SOFTWARE. *Swagger UI*. 2023. Disponível em: <https://swagger.io/tools/swagger-ui/.>

THOMAS EDISON. *Frases de Thomas Edison*. nsr. Disponível em: <https://exame.com/ciencia/10-frases-de-thomas-edison/>. Acesso em: 04 ago. 2023.

TRINDADE, L. D. D. L.; SAWICKI, S. Precificação do modelo infraestructure-as-a-service (iaas): um mapeamento sistemático. Salão do Conhecimento UNIJUÍ, 10 2019. Disponível em: <https://publicacoeseventos.unijui.edu.br/index.php/salaoconhecimento/article/view/11694/10411>.

VMWARE TANZU INC. *Spring Boot*. 2023. Disponível em: <https://spring.io/projects/spring-boot>.

VMWARE TANZU INC. *Spring Data*. 2023. Disponível em: <https://spring.io/projects/spring-data>.

WEAVE. *Sock Shop : A Microservice Demo Application*. 2023. Disponível em: <https://github.com/microservices-demo/microservices-demo>.

WEYNS, D. Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges. In: TAYLOR, RICHARD. *Handbook of Software Engineering*. Trento, Italy: Springer, 2017.

WEYNS *et al.* On patterns for decentralized control in self-adaptive systems. In: _____. *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 76–107. ISBN 978-3-642-35813-5. Disponível em: <https://doi.org/10.1007/978-3-642-35813-5_4>.

WILEY (Ed.). *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. nsr: IEEE Press, 2021.

XAVIER, M.; FERRAZ, C. A. F.; SETTE, I. Kumo: A Service for Portability in Heterogeneous Multi-Clouds. In: *Brazilian Symposium on Multimedia and Web*. PORTO ALEGRE, BRAZIL: SBC, 2018. p. 25–30. In Portuguese.

YAHYAPOUR, F. Z. G. L. X. F. R. A survey on virtual machine migration: Challenges, techniques, and open issues. IEEE Communications Surveys & Tutorials, v. 20, n. 2, p. 1206–1243, 2018.

YAHYAPOUR, F. Z. X. F. R. Cbase: A new paradigm for fast virtual machine migration across data centers. 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017.

# APÊNDICE A – SCRIPT DO JENKINS

Código Fonte 1 – Passo a passo da automação no *Jenkins*

```
pipeline{
    agent any

    parameters {
        string(name: 'APPROVE', defaultValue: '', description: 'destruir
            infraestrutura sim ou nao?')
        string(name: 'COSTESTIMATION', defaultValue: '', description: 'estimar
            custos sim ou nao?')
    }

    options {
        retry(3)
    }

    stages{

        stage('Git Checkout') {
            when {
                expression { "${APPROVE}" == "" }
            }
            steps{
                git credentialsId: '########-####-####-####-############', url: '
                    https://{access_token_classic}@github.com/lcs10-ufpe/hipster-
                    shop-aws'
            }
        }

        stage('Terraform Init') {
            when {
                allOf {
                    expression { "${APPROVE}" == "" }

                    anyOf {
                        expression { "${COSTESTIMATION}" == "--yes" }
                        expression { "${COSTESTIMATION}" == "" }
                    }
                }
            }
            steps{
                sh label: '', script: 'terraform init'
            }
        }

```

```
        stage('Gerar estimativa de custos com infracost') {
41          when {
                allOf {
43                  expression { "${APPROVE}" == "" }
                    expression { "${COSTESTIMATION}" == "--yes" }
45              }
            }
47          steps{
                sh 'infracost breakdown --show-skipped --path /var/jenkins_home/
                    workspace/hipster-shop-aws'
49          }
        }
51
        stage('Terraform Apply') {
53          when {
                allOf {
55                  expression { "${APPROVE}" == "" }
                    expression { "${COSTESTIMATION}" == "" }
57              }
            }
59          steps{
                sh '''MY_PUBLIC_IP="$(curl -s ipinfo.io/ip)"
61                  echo "... Meu IP publico $MY_PUBLIC_IP"

63                  echo "Starting terraform..."
                    terraform apply -var "my_public_ip=$MY_PUBLIC_IP/32" --auto-
                        approve'''
65          }
        }
67
        stage('Start Ansible') {
69          when {
                allOf {
71                  expression { "${APPROVE}" == "" }
                    expression { "${COSTESTIMATION}" == "" }
73              }
            }
75          steps{
                sh '''cd key
77                  chmod 400 demo_key
                    cd ..
79                  cd ansible
                    ANSIBLE_HOST_KEY_CHECKING=false ansible-playbook -i ../hosts
                        --private-key ../key/demo_key playbook.yml -v'''
81          }
        }
83
```

```
        stage('Terraform Destroy') {
            when {
                allOf {
                    expression { "${APPROVE}" == "--auto-approve" }
                    expression { "${COSTESTIMATION}" == "" }
                }
            }
            steps{
                sh '''MY_PUBLIC_IP="$(curl -s ipinfo.io/ip)"
                    echo "... Meu IP publico $MY_PUBLIC_IP"

                    echo "terraform destroy ${APPROVE}"
                    terraform destroy -var "my_public_ip=$MY_PUBLIC_IP/32" ${
                        APPROVE}'''
            }
        }

    }
}
```

**Fonte:** Elaborado pelo autor (2023).

O *Script* do *Jenkins* é o mesmo para os provedores AWS, GCP e AZ. Portanto, só será necessário atualizar o atributo *url* que fica no estágio (*stage*) *Git checkout* para as opções da Tabela 19. Toda etapa do *Jenkins* é detalhado na Seção 3.3.

Tabela 19 – Repositório do *GitHub* dos *Scripts* do *Terraform* e *Ansible*

| MBA | Nuvem | *URL* |
|---|---|---|
| *Hipster Shop* | AWS | https://github.com/lcs10-ufpe/hipster-shop-aws |
| *Hipster Shop* | GCP | https://github.com/lcs10-ufpe/hipster-shop-gcp |
| *Hipster Shop* | AZ | https://github.com/lcs10-ufpe/hipster-shop-azure |
| *Sock Shop* | AWS | https://github.com/lcs10-ufpe/sock-shop-aws |
| *Sock Shop* | GCP | https://github.com/lcs10-ufpe/sock-shop-gcp |
| *Sock Shop* | AZ | https://github.com/lcs10-ufpe/sock-shop-azure |

**Fonte:** Elaborada pelo autor (2023).

# APÊNDICE B – CONFIGURAÇÃO DO NGINX

Código Fonte 2 – Configuração da função balanceador de carga do NGINX

```
upstream lcs10 {
   zone lcs10 64k;
   server 3.230.104.191:8080;
   server 34.72.122.128:8080 down;
   server 40.76.63.255:8080 down;
}

server {
   listen         80;
   server_name    localhost;

   location / {
       proxy_pass           http://lcs10;
       proxy_http_version   1.1;
       proxy_set_header     Connection          "";
       proxy_set_header     Host                $host:8000;
       proxy_set_header     X-Forwarded-For     $proxy_protocol_addr;
   }

   access_log /var/log/nginx/nginx-access.log;
}
```

**Fonte:** Elaborado pelo autor (2023).

A configuração acima foi usada para todas as execuções experimentais. As propriedades do servidor na opção *upstream*, são IPs estáticos obtidos dos provedores AWS, GCP e AZ. Se não usar um IP estático, cada migração vai gerar um novo IP e fica difícil lidar com essa dinâmica no código.

# APÊNDICE C – CRIAR VM COM *TERRAFORM* (AWS)

Código Fonte 3 – Código do *Terraform* para AWS

```
resource "aws_key_pair" "keypair" {
    public_key = file("key/demo_key.pub")
}

resource "aws_eip_association" "eip_assoc" {
  instance_id   = aws_instance.instances.*.id[0]
  public_ip = "23.20.12.31"
}

resource "aws_instance" "instances" {
    count = 1

    ami = "ami-0323c3dd2da7fb37d"
    instance_type = "t2.xlarge"
    subnet_id = element(aws_subnet.public_subnet.*.id, count.index)
    key_name = aws_key_pair.keypair.key_name
    vpc_security_group_ids = [aws_security_group.allow_ssh.id, aws_security_group
        .allow_outbound.id, aws_security_group.allow_application.id,
        aws_security_group.allow_kube.id]

    tags = {
        Name = "test_instances"
    }
}

data "template_file" "hosts" {
    template = file("./template/hosts.tpl")

    vars = {
        PUBLIC_IP_0 = aws_instance.instances.*.public_ip[0]
    }
}

resource "local_file" "hosts" {
  content = data.template_file.hosts.rendered
  filename = "./hosts"
}

output "public_ips" {
  value = join(", ", aws_instance.instances.*.public_ip)
}
```

**Fonte:** Elaborado pelo autor (2023).

Para visualizar o código completo do *Terraform* [1] (como criar infraestrutura) e *Ansible* (configuração da máquina virtual) para o provedor AWS, acessar a *url* do *GitHub* da Tabela 19, opção 1 e 4.

[1] https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/instance

# APÊNDICE D – CRIAR VM COM *TERRAFORM* (GCP)

Código Fonte 4 – Código do *Terraform* para GCP

```
resource "google_compute_instance" "vm_instance" {
  name         = var.name
  machine_type = var.type_machine
  zone         = var.zone

  boot_disk {
    initialize_params {
      image = var.image
    }
  }

  network_interface {
      network = google_compute_network.vpc_network.name
      access_config {
        nat_ip = "34.31.177.7"
      }
  }

  metadata = {
    ssh-keys = "${var.ssh_user}:${file(var.ssh_pub_key_file)}"
  }
}

data "template_file" "hosts" {
    template = file("./template/hosts.tpl")

    vars = {
        PUBLIC_IP_0 = "${google_compute_instance.vm_instance.network_interface.0.
            access_config.0.nat_ip}"
    }
}

resource "local_file" "hosts" {
  content = data.template_file.hosts.rendered
  filename = "./hosts"
}

output "ip" {
  value = "${google_compute_instance.vm_instance.network_interface.0.
      access_config.0.nat_ip}"
}
```

**Fonte:** Elaborado pelo autor (2023).

Para visualizar o código completo do *Terraform* [1] (como criar infraestrutura) e *Ansible* (configuração da máquina virtual) para o provedor GCP, acessar a *url* do *GitHub* da Tabela 19, opção 2 e 5.

[1] https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/compute_instance

# APÊNDICE E – CRIAR VM COM *TERRAFORM* (AZ)

Código Fonte 5 – Código do *Terraform* para AZ

```
resource "azurerm_linux_virtual_machine" "vm" {
    name                = "demo"
    # location              = azurerm_resource_group.terraform_group.location
    # resource_group_name   = azurerm_resource_group.terraform_group.name
    location                    = "East US"
    resource_group_name         = "demo_group"
    network_interface_ids = [azurerm_network_interface.adapter_network.id]
    size                = "Standard_F4"

    os_disk {
        caching              = "ReadWrite"
        storage_account_type = "Standard_LRS"
    }

    source_image_reference {
        publisher = "Canonical"
        offer     = "UbuntuServer"
        sku       = "18.04-LTS"
        version   = "latest"
    }

    admin_username = "lcs10"
    disable_password_authentication = true

    admin_ssh_key {
        username       = "lcs10"
        public_key     = file("key/demo_key.pub")
    }

    tags = {
        environment = "Terraform Demo"
    }
}

data "azurerm_public_ip" "public_ip_address" {
  # name                = azurerm_public_ip.terraform_public_ip.name
  name                = "demo_public_ip"
  resource_group_name = azurerm_linux_virtual_machine.vm.resource_group_name
}

data "template_file" "hosts" {
    template = file("./template/hosts.tpl")

```

```
    vars = {
45      PUBLIC_IP_0 = data.azurerm_public_ip.public_ip_address.ip_address
    }
47 }

49 resource "local_file" "hosts" {
  content = data.template_file.hosts.rendered
51  filename = "./hosts"
}
53
output "public_ip" {
55  value = data.azurerm_public_ip.public_ip_address.ip_address
}
```

**Fonte:** Elaborado pelo autor (2023).

Para visualizar o código completo do *Terraform* [1] (como criar infraestrutura) e *Ansible* (configuração da máquina virtual) para o provedor AZ, acessar a *url* do *GitHub* da Tabela 19, opção 3 e 6.

[1] https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux_virtual_machine