Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Solução de Elasticidade Automática Híbrida na Camada de Infraestrutura como Serviço em Ambiente de Nuvem Pública

Italo Gervásio Cavalcante

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof. Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten

Brasília
2020

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

GG386s

Gervasio Cavalcante, Italo
Solução de Elasticidade Automática Híbrida na Camada de Infraestrutura como Serviço em Ambiente de Nuvem Pública / Italo Gervasio Cavalcante; orientador Aletéia Patrícia Favacho de Araújo Von Paumgartten. -- Brasília, 2020.
94 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2020.

1. Computação em Nuvem. 2. Elasticidade Automática. 3. Provisionamento de Recursos na Nuvem. 4. Nuvem Pública. 5. Previsão. I. Favacho de Araújo Von Paumgartten, Aletéia Patrícia, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Solução de Elasticidade Automática Híbrida na Camada de Infraestrutura como Serviço em Ambiente de Nuvem Pública

Italo Gervásio Cavalcante

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten (Orientadora)
CIC/UnB

Prof. Dr.a Maria Clicia Stelling de Castro
UERJ

Prof. Dr.a Ana Paula Bernardi da Silva
UCB

Prof. Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 09 de Março de 2020

Dedicatória

Dedico a toda minha família, pelo carinho, pela paciência e pela capacidade de me trazerem paz no transcorrer deste mestrado. Em especial, dedico à minha mãe, Maria Helena, a qual me ensinou o justo e o correto, virtudes que me trouxeram até este momento, e também dedico à minha esposa, Tammy, a qual foi muito compreensiva durante os anos deste mestrado. Foi com parte do tempo que eu poderia tê-los dedicado que eu escrevi essa dissertação.

Agradecimentos

Agradeço a todo o corpo docente do Mestrado Profissional em Computação Aplicada (PPCA), em especial à minha orientadora, Professora Dra Aletéia Patrícia Favacho de Araújo, pelo esforço e dedicação na realização deste trabalho, e ao professor Marcos Caetano pelo ensinamento de como fazer uma leitura crítica de um trabalho científico.

Agradeço aos colegas do PPCA, pelos ensinamentos e pela disponibilidade, em especial ao meus colegas da linha de pesquisa de infraestrutura.

Agradeço à equipe que trabalha diariamente comigo no TCU (colegas do SIRED e agregados) que tanto me ajudou, equipe essa que compreendeu as minhas ausências nos momentos em que precisei e também assumiram carga maior de trabalho nos períodos em que estive ausente priorizando o estudo. Além disso, se dispuseram a avaliar a minha pré-apresentação de mestrado, fornecendo contribuições relevantes.

Agradeço a todos os amigos, que direta ou indiretamente, prestaram apoio para que esta pesquisa ocorresse da melhor forma possível.

Agradeço à minha mãe e aos meus irmãos pela inspiração, ensinamentos, e por compreenderem minhas ausências.

Agradeço à minha esposa Tammy pela paciência de ouvir todas as minhas lamentações, além do apoio incondicional e irrestrito. Agradeço também ao Nimbus e ao Bola de Neve pelos momentos de descontração.

Por fim, agradeço também aos membros da banca, por despenderem seu tempo para analisarem este trabalho e fornecerem comentários agregadores.

Resumo

O Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (*National Institute of Standards and Technology* - NIST) define elasticidade, *pool* de recursos e amplo acesso à rede como as principais características da computação em nuvem, a qual fornece serviços altamente disponíveis, confiáveis e elásticos aos clientes da nuvem. A natureza elástica dos recursos da nuvem, juntamente com o modelo de precificação permitem que os clientes na nuvem paguem apenas pelos recursos que realmente usem. Todavia, embora a elasticidade da nuvem e seu modelo de precificação sejam benéficos em termos de custo, a obrigação de manter qualidade de serviço com os usuários finais exige que os clientes da nuvem busquem soluções de elasticidade automática para equilibrar o compromisso entre o custo e o desempenho, provisionando automaticamente recursos para os serviços em nuvem. Esta tese propõe uma solução de elasticidade automática que supera as deficiências das soluções baseadas em regras com limites fixos. O sistema de elasticidade automática proposto consiste em uma abordagem híbrida, composta de um conjunto de previsões auto-adaptável e um componente reativo baseado em limites dinâmicos. O conjunto de previsões corrige a primeira falha (ou seja, a natureza reativa) das soluções baseados em regras, prevendo a carga de trabalho futura próxima do serviço em nuvem. Além disso, o componente reativo, baseado em limites dinâmicos, diminui a dificuldade de configuração da solução de elasticidade automática. Os resultados da avaliação mostram que a solução proposta reduz tempo de resposta médio em até 70% em comparação com o sistema de elasticidade automática da Amazon, além de reduzir o custo total em torno de 25%.

Palavras-chave: Elasticidade Automática; Provisionamento de Recursos na Nuvem; Previsão; Computação em Nuvem; Nuvem Pública

Abstract

The National Institute of Standards and Technology - NIST - defines elasticity, resource pooling and broad network access as the main characteristics of cloud computing, which provides highly available, reliable and resilient services to cloud customers. The elastic nature of cloud resources and the pricing model allow customers in the cloud to pay only for the resources that they use. Although the elasticity of the cloud and its pricing model are beneficial in terms of cost, the obligation to maintain quality of service with end-users requires that cloud customers look for auto-scaling solutions to balance the trade-off between cost and performance, automatically provisioning resources for cloud services. This thesis proposes an automatic elasticity solution that overcomes the shortcomings of rule-based solutions with fixed limits. The proposed automatic elasticity system consists of a hybrid approach, composed of a set of self-adaptive predictions and a reactive component based on dynamic limits. The forecast set corrects the first flaw (i.e. the reactive nature) of rule-based solutions, predicting the future workload near the cloud service. Besides, the reactive component, based on dynamic limits, reduces the difficulty of configuring the automatic elastic solution. The evaluation results show that the proposed solution reduces average response time by up to 70% compared to Amazon's auto-scaling system, in addition to reducing the total cost by around 25%.

Keywords: Auto-scaling; Cloud Resource Provisioning; Forecasting; Cloud Computing; Public Cloud

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ARIMA Modelo Autorregressivo Integrado de Médias Móveis.

AWS Amazon Web Services.

IaaS Infraestrutura como um Serviço.

LSTM Rede Neural Recorrente Baseada em Memória Longa de Curto Prazo.

MAE Erro Absoluto Médio.

MAPE Erro Médio Percentual Absoluto.

MAPE-K Monitoramento - Análise - Planejamento - Execução sobre o Conhecimento.

MLP Multi Layer Perceptron.

PaaS Plataforma como um Serviço.

QoS Qualidade de serviço.

RMSE Raiz do Erro Médio Quadrático.

RMSSE Raiz do Erro Médio Quadrático Escalonado.

RNA Redes Neurais Artificiais.

SaaS Software como um Serviço.

SLA Acordo de Nível de Serviço.

TCU Tribunal de Contas da União.

VM máquina virtual.

Capítulo 1

Introdução

A computação está sendo transformada em um modelo que consiste em serviços que são comercializados e entregues de maneira semelhante aos serviços públicos tradicionais, tais como água, eletricidade, gás e telefonia [1]. Nesse modelo, os usuários acessam os serviços com base em seus requisitos, sem levar em conta a localização onde os serviços são hospedados ou como são entregues. Assim, dentre paradigmas de computação que seguem o modelo distribuído, o mais recente é a computação em nuvem, na qual empresas e usuários podem acessar aplicativos de qualquer lugar do mundo sob demanda [1].

Considerando os avanços tecnológicos, a computação em nuvem se tornou uma realidade plenamente acessível às organizações, sendo mundialmente adotada por empresas e órgãos de governo, sendo por isso recomendada a adoção de sistemas de computação em nuvem pelo governo brasileiro, em vez de soluções de infraestrutura de TI individuais para cada órgão [2]. Dentre os benefícios da adoção deste modelo pelos órgão de governo, destacam-se: redução de custos, redução da ociosidade dos recursos, ambientes computacionais atualizados rapidamente, agilidade na implantação de novos serviços, foco nas atividades fim do negócio, uso mais inteligente da equipe de TI e elasticidade [2]. A elasticidade é o grau que um sistema é capaz de provisionar e desprovisionar recursos rapidamente e dinamicamente [3].

Diante do exposto, a computação em nuvem é um modelo de Tecnologia da Informação (TI) que permite um acesso fácil por meio de redes a recursos mutualizados e configuráveis, que podem ser rapidamente ativados e desativados. A computação em nuvem tornou-se uma tecnologia de ponta que oferece a mutualização de infraestruturas de TI como serviços em vários modelos, como Software, Plataforma, e Infraestrutura como serviço. Empresas como Amazon [4], Microsoft [5], IBM [6] e Google [7], para citar apenas algumas, oferecem esses serviços, que dependem de modelos de negócios de virtualização e de pagamento conforme o uso [8].

Dessa forma, a computação em nuvem busca a redução dos custos computacionais, o

aumento da confiabilidade, a flexibilidade e o provisionamento dos recursos e dos serviços sob demanda, na qual recursos e serviços estejam disponíveis de acordo com as necessidades de seus usuários, de forma transparente e dinâmica. Assim, na computação em nuvem o processamento, a manipulação de dados e o armazenamento são vistos como serviços [9].

Os recursos disponíveis nos ambientes de computação em nuvem parecem ilimitados para o usuário e podem ser adquiridos a qualquer momento, e em qualquer quantidade. A elasticidade automática é um requisito essencial das plataformas de nuvem atuais, e é alcançada usando técnicas de elasticidade automática. Para aproveitar isso, as empresas estão cada vez mais usando o ambiente de computação em nuvem para hospedar seus serviços [10].

Já há algum tempo, mais e mais empresas estão adotando o modelo de implantação de infraestrutura como serviço como sua principal forma de provisionar a capacidade de computação. A infraestrutura como serviço pode ser baseado em um modelo de implantação privado, em que o paradigma da nuvem é usado para reduzir os custos de execução da infraestrutura de TI, explorando a economia de escala na infraestrutura interna, ou com base em um modelo de implantação público, em que a capacidade é adquirida sob demanda de provedores externos, que normalmente cobram o uso de sua infraestrutura seguindo um modelo de tarifação *pay-per-use* [11].

Neste modelo de implantação de infraestrutura como serviço de modo público, a elasticidade é uma das principais vantagens oferecidas pelos provedores de nuvem aos seus clientes. Uma estratégia ideal para executar serviços com demandas variadas no tempo sobre os recursos na nuvem seria aquela na qual se poderia alocar, a qualquer momento, exatamente a capacidade necessária para manter os serviços em execução com o desempenho necessário. Essa abordagem ajustaria automaticamente a capacidade provisionada, alocando mais recursos quando a demanda aumentar e liberando recursos assim que eles não forem mais necessários. No entanto, a alocação eficiente de recursos é uma tarefa, geralmente, executada por uma solução de elasticidade automática [11]. Diante do exposto, este trabalho propõe um mecanismo de elasticidade automática para nuvens públicas que hospedam aplicações que devem lidar com alterações na intensidade da carga de trabalho dinamicamente. Para isso, este trabalho utiliza uma abordagem experimental para verificar a viabilidade do mecanismo de elasticidade automática, sendo esse experimento realizado por meio do Wikibench [12], um *benchmark* que reproduz solicitações reais dos rastreamentos da Wikipedia.

1.1 Motivação da Pesquisa

O Tribunal de Contas da União (TCU) é o órgão de controle externo do governo federal e auxilia o Congresso Nacional na missão de acompanhar a execução orçamentária e financeira da Administração Pública Federal. Uma das metas do TCU é ser referência na promoção de uma Administração Pública efetiva, ética, ágil e responsável. Para alcançar esta meta, o Tribunal promove regularmente ações para disseminar boas práticas por meio de treinamentos, cooperações nacionais e internacionais, as quais têm aumentando a demanda por recursos de TI do TCU. Para atender essa demanda de forma ágil, o TCU firmou um contrato administrativo com a empresa Primesys [13] para uso de serviços de computação em nuvem.

Nesse contrato, a Primesys funciona como uma intermediária entre TCU e os provedores de serviços em nuvem, papel conhecido na literatura como *broker* [14]. Uma das tarefas da Primesys nesse contrato é apoiar a equipe do TCU em decidir a quantidade ideal de recursos no ambiente de computação em nuvem. No entanto, como a taxa de solicitações às aplicações de uma organização pode variar com o tempo, geralmente, é difícil determinar a quantidade certa de recursos em nuvem [15]. Assim, uma opção seria observar manualmente a carga de trabalho de uma aplicação, mas essa opção é entediante e desperdiçadora de recursos, levando ao nascimento da elasticidade automática. Um mecanismo de elasticidade automática remove ou adiciona automaticamente uma certa quantidade de recursos com o objetivo de otimizar o desempenho e o custo da aplicação [16].

As soluções de elasticidade automática são apresentadas para equilibrar automaticamente a relação custo/desempenho e evitar as condições de subprovisionamento e superprovisionamento [17]. A condição de subprovisionamento é o resultado da saturação dos recursos e pode causar violação de qualidade de serviço desejada. Em contrapartida, a condição de superprovisionamento ocorre quando os recursos provisionados são desperdiçados, o que resulta em consumo excessivo de energia e alto custo operacional [18].

Nas primeiras tentativas da equipe do TCU em decidir a quantidade ideal de recursos no ambiente de computação em nuvem, a Primesys utilizou a solução de elasticidade automática da AWS, a qual é baseada em regras com limites para ações de elasticidade, as quais são normalmente baseadas na utilização de recursos em execução no provedor, como "Adicione algumas instâncias quando a média de utilização de CPU estiver acima de 70%". Embora essa solução de elasticidade automática seja fácil de usar e entender, nem sempre é fácil para os usuários dessa solução selecionar os limites para ações de elasticidade “adequados”, especialmente quando as aplicações que irão ser executadas ou em execução são complexas ou há pouco conhecimento do corpo técnico do cliente da nuvem. Em outras palavras, esses mecanismos baseados exclusivamente em regras de limites fi-

xos podem não resolver a tarefa de provisionamento de recursos para a aplicação, pois dependendo do modo como forem configurados os limites, podem nunca ocorrer ações de elasticidade ou ainda levar o sistema a uma condição de subprovisionamento ou superprovisionamento [17]. Além disso, essa solução de elasticidade automática negligencia o tempo de inicialização de uma máquina virtual, dado que uma ação de elasticidade é tomada com base exclusivamente no valor mais recente da demanda [19].

Por conta desse cenário, este trabalho se concentra em diminuir a complexidade de configuração da solução de elasticidade automática a ser usada no TCU, usando um mecanismo para propor os valores limites para ações de elasticidade da aplicação. Além disso, este trabalho trata de acrescentar a essa solução de elasticidade automática uma estratégia de antecipação à variação na demanda, predizendo a demanda futura com base em dados históricos.

1.2 Objetivos

O objetivo geral deste trabalho é propor uma solução de elasticidade automática em ambiente de nuvem pública, que possa ser implementada no Tribunal de Contas da União (ou em outros órgãos de governo similares), com a função atender a demandas variáveis dos usuários, utilizando uma abordagem para prever a variação na demanda, e que, ao mesmo tempo, essa solução abstraia a complexidade da tarefa de definir os limites de elasticidade.

Para cumprir o objetivo principal, este trabalho tem os seguintes objetivos específicos:

- Propor uma solução de elasticidade automática em nuvem que permita um uso massivo de computação em nuvem, garanta um consumo adequado (sem provisionamento exagerado ou insuficiente) de recursos e que seja independente de aplicação;
- Propor um mecanismo que reduza a complexidade de configuração da solução de elasticidade automática ao sugerir os valores limites para ações de elasticidade;
- Propor uma abordagem de antecipação da demanda na nuvem com base em dados históricos da aplicação;

1.3 Contribuição Esperada

A contribuição esperada com este trabalho é a proposição de uma solução de elasticidade automática híbrida (baseada em valores de demanda recentes e valores preditos) que seja independente de aplicação, e que seja também configurada de forma dinâmica e automática.

Essa solução proporá os limites para tomada de decisão sobre ações de elasticidade de maneira dinâmica, abstraindo a complexidade de configuração da solução pela equipe do TCU. Quanto às tarefas de previsão, os modelos para previsão deverão ser criados durante a execução das tarefas da aplicação (*on-line*), reduzindo assim o tempo de aprendizado da solução, quando comparado com as soluções que treinam os modelos com dados de execuções anteriores da aplicação. Dessa forma, para esta solução funcionar adequadamente, só será preciso monitorar a utilização de recursos de infraestrutura das máquinas virtuais nas quais os serviços são executados, sem nenhuma consideração adicional ou restritiva.

1.4 Estrutura do Trabalho

Este trabalho contém, além deste capítulo introdutório, outros cinco capítulos. No Capítulo 2 serão apresentadas a computação em nuvem, suas características e o seu modo de funcionamento. Além disso, serão apresentadas as características da elasticidade automática e o modelo de computação autônoma utilizado, assim como as técnicas usadas neste trabalho. Em seguida, no Capítulo 3 serão apresentadas as formas de abordar a elasticidade automática em ambientes de nuvem. Além disso, serão apresentados os principais trabalhos relacionados ao tema. No Capítulo 4 será descrita a solução híbrida de elasticidade automática proposta neste trabalho. O Capítulo 5 apresenta os experimentos realizados nesta dissertação com a solução de elasticidade proposta. Para finalizar, o Capítulo 6 apresenta algumas considerações finais deste trabalho e possíveis trabalhos futuros.

Capítulo 2

Referencial Teórico

Este capítulo apresenta uma revisão dos principais tópicos sobre a área de Computação em Nuvem, estando estruturado da seguinte forma: a Seção 2.1 apresenta o conceito, as características e os principais tipos e modelos de serviço em nuvem; a Seção 2.2 apresenta as principais definições de elasticidade em ambiente de nuvem, assim como os métodos e os modelos de elasticidade existentes na literatura; na Seção 2.3 é apresentado o Modelo de Computação Autônoma seguido neste trabalho; Por fim, na Seção 2.4 são apresentadas as técnicas de elasticidade, com destaque para as técnicas em previsões de séries temporais.

2.1 Computação em Nuvem

Computação em nuvem é um paradigma de computação que mudou a forma como os serviços de informação podem ser provisionados. As nuvens representam um passo na evolução da cadeia de desenvolvimento de tecnologias de computação e comunicação, introduzindo tipos de serviços e uma camada de abstração para a virtualização e a mobilidade dos serviços gerais [20].

A computação em nuvem surgiu como uma tendência para a provisão de recursos e de serviços de forma rápida, barata, escalável e flexível [21]. Assim, diversos tipos de recursos podem ser disponibilizados como serviços, tais como memória, capacidade de processamento, armazenamento, entre outros. Todavia, não há uma única definição de computação em nuvem na literatura. Algumas definições relevantes são apresentadas a seguir:

- Armbrust et al. [22] definem nuvem computacional como "a união de aplicações oferecidas como serviço pela Internet, com o hardware e o software localizados em *datacenters* de onde o serviço é provido";

- De acordo com Forster et al. [21], computação em nuvem é "um paradigma computacional altamente distribuído, direcionado por uma economia de escala, na qual poder computacional, armazenamento, serviços e plataformas abstratas, virtualizadas, gerenciadas e dinamicamente escaláveis são oferecidos sob demanda para usuários externos por meio da Internet";
- Buyya et al. [1] definem computação em nuvem como "um tipo de sistema paralelo e distribuído, que consiste em uma coleção de computadores virtuais interconectados que são provisionados dinamicamente, e apresentados como um ou mais recursos computacionais unificados, baseados em acordos de nível de serviço estabelecidos entre o provedor de recursos e o consumidor".

A primeira definição é bastante abrangente e, infelizmente, deixa margem para que haja confusão com outros paradigmas de computação distribuída. É possível, por exemplo, confundir com a definição de uma grade computacional [23]. A segunda definição já apresenta um elemento importante para a definição mais completa de nuvem, que é a expressão "oferecido sob demanda". Isto significa que na computação em nuvem o usuário terá tanto recurso quanto demandado, diferentemente de como ocorre nos outros paradigmas de computação distribuída, e isso é uma diferença fundamental. A terceira definição fala sobre os acordos de nível de serviço, que são contratos negociados entre o consumidor e o provedor, e que definem quais indicadores irão medir a qualidade do serviço oferecido na nuvem. Assim, a violação deste acordo pode levar ao pagamento de multas por parte de quem ocasionou a violação [1].

Além das definições para computação em nuvem apresentadas, existe a definição apresentada por Mell e Grance [24] que é a mais difundida. Nessa definição a nuvem é apresentada como um modelo para habilitar acesso onipresente, conveniente e sob demanda à rede para recursos compartilhados (por exemplo rede, servidores, armazenamento, aplicativos e serviços), que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação com o provedor de serviços.

Assim, a computação em nuvem possui diversas características que a difere dos outros sistemas distribuídos existentes. As principais características são apresentadas a seguir [23]:

- *Self-service*: na computação em nuvem, os serviços, tais como armazenamento e processamento, são contratados pelo usuário diretamente, sem a participação de algum administrador dos provedores de nuvem, e de forma que atenda às necessidades do usuário [24];

- Amplo Acesso: a ideia é que os recursos estejam disponíveis na Internet e possam ser acessados a partir de dispositivos heterogêneos, desde que estes possuam acesso à rede mundial de computadores;
- Agrupamento de Recursos: os recursos de um determinado provedor são organizados em um *pool* de recursos físicos e virtuais, de forma a facilitar o acesso de vários usuários e os ajustes de demanda;
- Elasticidade: caso seja necessário aumentar a utilização de determinado recurso, este aumento deve ocorrer de forma fácil ou até mesmo de forma automática;
- Serviços Mensuráveis: a utilização dos recursos deve ser monitorada a todo momento, de forma a garantir que seja cumprido o contrato de qualidade de serviço realizado entre o provedor de serviço e o usuário.

Dessa forma, as características de computação em nuvem podem ser ofertadas por diferentes modelos de serviços e tipos de nuvens que serão apresentados na Seção 2.1.1.

2.1.1 Modelos de Serviço e Tipos de Nuvens

Os modelos dos serviços livram o consumidor de gerenciar ou controlar a infraestrutura de nuvem do provedor. Os serviços oferecidos no paradigma de computação em nuvem podem ser divididos em categorias. Apesar de ser possível encontrar na literatura propostas com mais de três classes [25], o mais comum é dividir os serviços nas classes Infraestrutura como um Serviço, Plataforma como um Serviço e Software como um Serviço, as quais são descritas a seguir [24]:

- Infraestrutura como um Serviço (*Infrastructure-as-a-Service* - IaaS): os serviços que são oferecidos se caracterizam por serem de infraestrutura, podendo ser desde capacidade de processamento, de armazenamento ou até de máquinas virtuais completas. Um provedor que se destaca neste modelo de serviço é a Amazon, com o *Elastic Compute Cloud* (EC2) [26], e com o *Simple Storage Service* (S3) [27];
- Plataforma como um Serviço (*Platform-as-a-Service* - PaaS): o que caracteriza este modelo de serviço é a disponibilização de um ambiente no qual o usuário possa programar, testar e executar aplicações. O *Google App Engine* [28] é um exemplo desta categoria, pois é um ambiente no qual aplicações podem ser desenvolvidas sem terem nenhum tipo de programa instalado na sua máquina, tudo é feito por meio da Internet;
- Software como um Serviço (*Software-as-a-Service* - SaaS): são as aplicações disponibilizadas pelos provedores como serviços aos usuários comuns, de forma gratuita

ou cobrando pela sua utilização. Um exemplo é o *Google Docs* [29], no qual são disponibilizados editores de textos, editores de planilhas, e ferramentes para a criação de apresentações. Os usuários podem acessar estes serviços a partir de qualquer lugar, sem a limitação de ter o software instalado na sua máquina.

Além disso, as nuvens podem ser divididas em quatro tipos diferentes de implantação, que são nuvens públicas, privadas, comunitárias e híbridas, as quais são descritas a seguir [24]:

- Nuvem Pública: este tipo de nuvem é aquele mantido por um fornecedor, geralmente, uma grande companhia, para um usuário comum ou uma empresa. É a nuvem no sentido tradicional do senso comum, na qual os recursos são provisionados dinamicamente e através da Internet [25];
- Nuvem Privada: é a nuvem que está dentro de um contexto empresarial e não está acessível a todas as pessoas, sendo restrita apenas aos funcionários e aos parceiros da empresa. Pelo fato de não estar disponível na Internet, apresenta vantagens em termos de segurança e largura de banda da rede;
- Nuvem Comunitária: infraestrutura que é compartilhada por organizações que mantém algum tipo de interesse em comum (jurisdição, segurança, economia), e pode ser administrada, gerenciada e operada por uma ou mais destas organizações;
- Nuvem Híbrida: ambiente no qual ambos os tipos de nuvens anteriormente descritos podem ser utilizados em conjunto. É interessante para alguns modelos de negócios, pois arquivos sigilosos ou sistemas que manipulam dados sigilosos podem ser mantidos em uma nuvem privada, enquanto que os outros dados e sistemas podem ficar na nuvem pública, por exemplo.

Neste trabalho, o foco está na perspectiva do cliente IaaS em um provedor de nuvem pública. Um cenário típico pode ser uma organização que deseja hospedar uma aplicação e, para esse fim, arrende recursos de um provedor de IaaS público, como o Amazon EC2. Assim sendo, para alinhar a terminologia a ser usada neste trabalho, a partir de agora, os seguintes termos serão usados:

- Provedor: refere-se principalmente ao provedor de IaaS, que oferece recursos praticamente ilimitados na forma de máquinas virtuais;
- Cliente: refere-se ao cliente do serviço IaaS, que o utiliza para hospedar a aplicação. Em outras palavras, é o proprietário da aplicação;
- Usuário: refere-se ao usuário final que acessa a aplicação e gera a carga de trabalho ou a demanda que impulsiona o comportamento dessa aplicação.

2.2 Elasticidade em Nuvem Computacional

A elasticidade é um ponto chave para adicionar QoS aos serviços em nuvem. A elasticidade permite que os provedores adicionem ou removam recursos, sem interrupção e, em tempo de execução, para lidar com a variação de carga [30]. Como a elasticidade não possui uma definição única, algumas definições apresentadas para elasticidade na literatura serão apresentadas na Seção 2.2.1.

2.2.1 Definições de Elasticidade

A elasticidade da computação em nuvem foi definida em alguns trabalhos na literatura. Segundo Mell e Grance [31], elasticidade é permitir que os recursos em nuvem sejam adquiridos rapidamente, em alguns casos, até automaticamente, para atender aos aumentos e diminuições das cargas de trabalho. Para usuários da nuvem, os recursos disponíveis parecem ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento.

Além de Mell e Grance [31], Herbst et al. [3] apresentaram a seguinte definição para elasticidade: "Elasticidade é o grau em que um sistema é capaz de se adaptar às mudanças na carga de trabalho, provisionando e desprovisionando recursos de maneira autônoma, de modo que, a cada momento, os recursos disponíveis correspondam à demanda atual o mais rápido possível". As definições de elasticidade de Mell e Grance [31] e Herbst et al. [3] destacam a rapidez e adequação à demanda como aspectos relevantes. Outras definições de elasticidade são apresentadas a seguir:

- Li et al. [32]: a rapidez com que um sistema pode se adaptar às alterações na carga de trabalho que podem ocorrer em um curto período de tempo;
- Perez-Sorrosal et al. [33]: capacidade, em tempo de execução, para adicionar e remover recursos sem interrupção do serviço para lidar com a variação da carga de trabalho;
- Pandey et al. [34]: capacidade de um sistema para expandir e retrair sem problemas;
- Espadas et al. [35]: capacidade de criar um número variável de instâncias de máquinas virtuais, dependendo das demandas da aplicação;
- Cooper et al. [36]: o sistema pode adicionar ou remover mais capacidade a um sistema em execução implementando novas instâncias de cada componente e transferindo a carga para eles.

As definições de Li et al. [32] e Perez-Sorrosal et al. [33] destacam a importância do tempo como um aspecto central da elasticidade, enquanto a ausência de interrupção

durante a ação de elasticidade é destacada em Perez-Sorrosal et al. [33] e Pandey et al. [34]. As definições para elasticidade de Espadas et al. [35] e Cooper et al. [36] são específicas para ambientes de nuvem IaaS, assim como este trabalho.

Praticamente todas as definições apresentadas destacam a importância da elasticidade para adequar o ambiente em nuvem as demandas da aplicação em execução. Logo, o gerenciamento de ambientes em nuvem envolve a construção de estratégias para torná-lo elástico para garantir a QoS, e usar os recursos com eficiência.

Diferentes soluções implementam elasticidade, adicionando ou removendo recursos de acordo com políticas baseadas em carga de trabalho, por exemplo, quando o uso da CPU é maior do que um valor definido ou quando a QoS não é atendida. Na Seção 2.2.2 serão apresentados os métodos ou as ações de elasticidade utilizados para implementar a elasticidade em ambiente de nuvem, e será indicado também como esses métodos ou ações estão sendo tratados neste trabalho.

2.2.2 Métodos

Os métodos ou as ações de elasticidade comumente usadas na literatura são [37]:

- Elasticidade Vertical: é implementada com a elasticidade da nuvem em variar de nós pequenos até nós mais poderosos, equipados com melhores processador, memória ou armazenamento. Neste método os nós que compõem o *cluster* podem variar quanto ao tipo (Figura 2.1);
- Elasticidade Horizontal: permite adicionar mais instâncias de máquina ou nós de processamento do mesmo tipo, com base na cota acordada no contrato de nível de serviço. Obviamente, esta técnica foca no uso de *clusters* homogêneos com nós idênticos (Figura 2.1);
- Elasticidade Híbrida: permite escalonar ou reduzir o tipo de instância e ajustar a quantidade da instância por recursos de aumento ou redução ao mesmo tempo. O método híbrido é mais atraente com o uso de *clusters* heterogêneos.

Usando o método de elasticidade vertical, o sistema é expandido ou reduzido, aumentando ou diminuindo a capacidade da máquina virtual, como por exemplo alterando a quantidade de CPU ou de memória em uma única máquina virtual do sistema.

Usando o método de elasticidade horizontal, o sistema é expandido ou reduzido ao serem adicionadas ou removidas máquinas virtuais. Um serviço em nuvem em larga escala, geralmente, usa várias VMs executando aplicações e um balanceador de carga para distribuir as solicitações entre as máquinas virtuais. Esse tipo de sistema é capaz de ser elástico adicionando ou removendo máquinas virtuais sem parar o sistema [38].

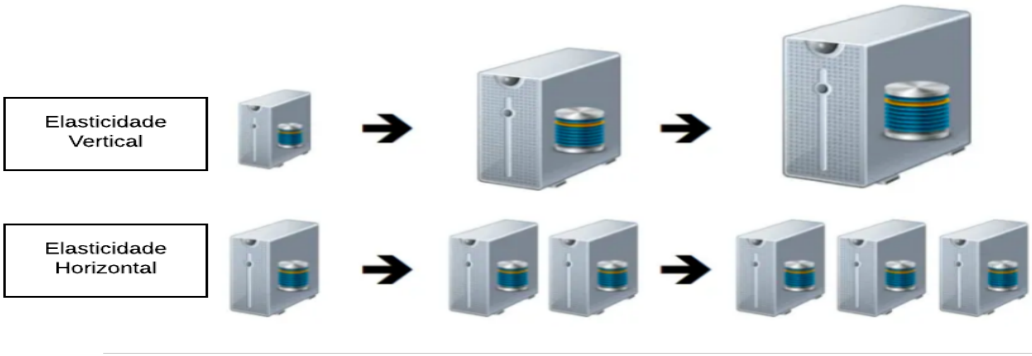

Figura 2.1: Elasticidade Vertical x Elasticidade Horizontal.

As elasticidades horizontal e vertical puras possuem melhores custos-benefício que o método híbrido [37]. Enquanto a elasticidade vertical possui maior desempenho no uso de *clusters* menores (poucos nós de instâncias de máquina muito poderosas), a elasticidade horizontal deve ser usada praticamente quando a frequência de elasticidade é alta, em razão da menor sobrecarga de reconfiguração [37].

A velocidade de elasticidade desempenha um papel vital na minimização das lacunas de super ou subprovisionamento. A elasticidade horizontal é mais rápida que as demais - além de ser mais robusta quanto a recuperação de falhas [39] e, em teoria, não causará nenhuma interrupção nos serviços do cliente. Esse é o principal motivo pelo qual a elasticidade horizontal é mais frequentemente praticada pelos provedores de nuvem do que os demais métodos de elasticidade [39].

Diante do exposto, e partindo-se da premissa de que o serviço em execução na nuvem não pode sofrer interrupção, este trabalho trata somente de elasticidade horizontal.

Na Seção 2.2.3 serão apresentados os modelos de elasticidade utilizados para implementar a elasticidade em ambiente de nuvem, e também será indicado como os modelos de elasticidade estão sendo tratados neste trabalho.

2.2.3 Modelos

Os modelos de elasticidade estão atrelados ao conceito de planejamento de capacidade, que é o processo de determinar e atender às demandas futuras dos recursos, produtos e serviços de TI de uma organização na nuvem. Nesse contexto, a capacidade representa a quantidade máxima de carga de trabalho que um recurso na nuvem é capaz de atender em um determinado período de tempo. Uma discrepância entre a capacidade de atendimento de um recurso na nuvem e sua demanda pode resultar em um sistema que se torna ineficiente (excesso de provisionamento), ou incapaz de atender às necessidades do

usuário (sub-provisionamento). Assim sendo, o planejamento da capacidade está focado em minimizar essa discrepância para obter desempenho e eficiência previsíveis [40].

O planejamento da capacidade pode ser desafiador, pois requer a estimativa de flutuações na carga de trabalho em execução. Existe uma necessidade constante de equilibrar os requisitos de pico de uso, sem gastos excessivos desnecessários em infraestrutura. Um exemplo é equipar a infraestrutura de TI na nuvem para acomodar cargas máximas de uso, o que pode impor investimentos financeiros irracionais, conforme representado na Figura 2.2.

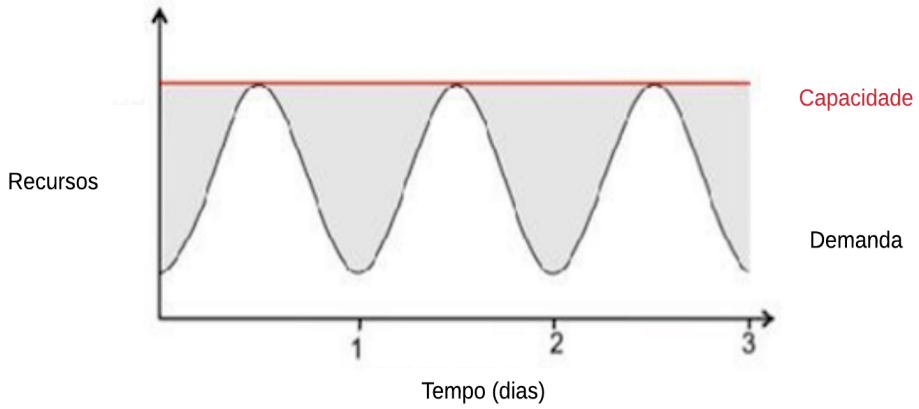

Figura 2.2: Provisionamento para Picos de Carga, adaptado de [24].

Em outros casos, a moderação dos investimentos pode resultar em subprovisionamento (veja a Figura 2.3) levando ao risco de perdas de transação e outras limitações de uso devido a limites de uso reduzidos, conforme apresentado na Figura 2.4, com a queda de demanda pelo não atendimento adequado em um tempo anterior [40].

Dentre as diferentes estratégias de planejamento de capacidade, há duas que orientam os modelos de elasticidade em nuvem, as quais são [40]:

- Estratégia de Atraso: adicionando capacidade quando o recurso da nuvem atinge sua capacidade total;
- Estratégia de Antecipação: adicionando capacidade a um recurso da nuvem em antecipação à demanda.

A primeira estratégia é a base para os modelos reativos, enquanto a segunda é a base para os modelos proativos ou preditivos. Nos modelos reativos, o sistema reage a

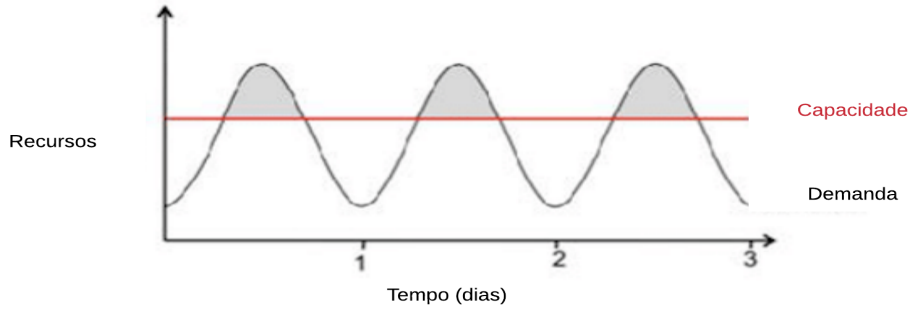

Figura 2.3: Subprovisionamento de Recursos, adaptado de [24].

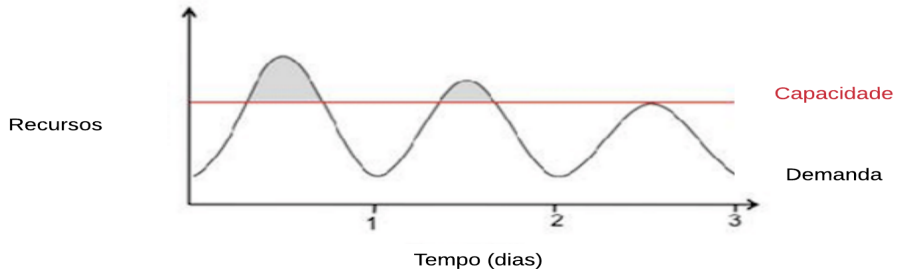

Figura 2.4: Redução na Demanda por Subprovisionamento de Recursos, adaptado de [24].

mudanças na carga de trabalho somente quando essas mudanças forem detectadas, usando os últimos valores obtidos do conjunto de variáveis monitoradas. Contudo, assim, como o provisionamento de recursos leva algum tempo, o efeito desejado pode chegar quando for tarde demais [15].

Os modelos proativos antecipam demandas futuras usando técnicas de previsão de carga de trabalho para determinar quando essas cargas futuras excederão a capacidade atualmente provisionada, e acionando um algoritmo para alocar recursos adicionais antes que essa capacidade seja excedida. Por outro lado, os modelos preditivos, geralmente, baseiam sua decisão no histórico da carga de trabalho [30].

Contudo, é possível criar uma solução híbrida combinando abordagens reativas e proativas para lidar com a imprecisão da previsão, e também para evitar variações na capacidade provisionada devido a oscilações na carga de trabalho. Este trabalho tratará de uma abordagem híbrida com o intuito de aproveitar as vantagens dos modelos reativo e

proativo.

Para que a solução híbrida proposta neste trabalho consiga fazer uso de modelos reativos e proativos de elasticidade de forma automática, esta solução segue um modelo de computação autônoma desenvolvido pela IBM que será apresentado na Seção 2.3.

2.3 Modelo de Computação Autônoma

O modelo de computação autônoma desenvolvido pela IBM [6] engloba as fases de Monitoramento (M), Análise (A), Planejamento (P) e Execução (E). Essas fases compartilham uma base de conhecimento (*Knowledge* - K), a qual é fundamental para a tomada de decisões no modelo conhecido como MAPE-K [41]. O modelo MAPE-K pode ser aplicado para implementar um sistema de aplicações em nuvem que conhece seu estado atual e reage a mudanças no seu estado. Este modelo detalha diferentes componentes que permitem que um gerente autônomo gerencie propriedades, como monitorar, analisar, planejar, executar e conhecimento da aplicação [42].

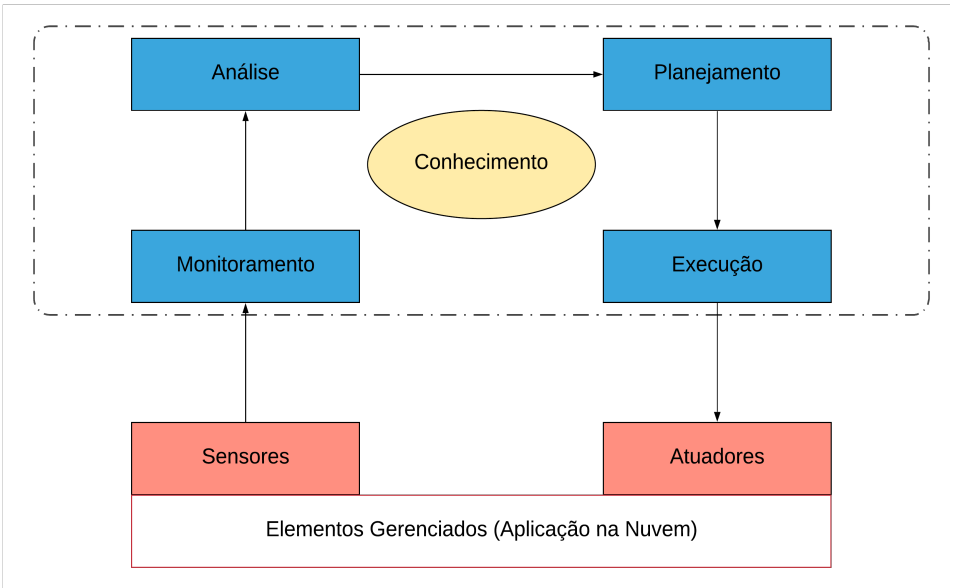

Figura 2.5: O Loop de Controle MAPE-K.

O processo de elasticidade automática para aplicações em nuvem neste trabalho corresponde ao modelo MAPE-K, confome apresentado na Figura 2.5. Este modelo adapta dinamicamente os recursos atribuídos as aplicações na nuvem, dependendo da carga de trabalho de entrada [43]. O gerenciador autônomo interage com o elemento gerenciado (por exemplo, recursos de aplicações na nuvem) por meio de duas interfaces que são os sensores e os atuadores para supervisionar e agir no sistema, respectivamente. O modelo MAPE-K consiste em quatro fases [44]:

- Monitoramento (M): a fase de monitoramento envolve a captura de propriedades do elemento gerenciado que podem ser componentes de software ou hardware usados para executar o monitoramento. Eles são chamados de sensores. O componente do monitor é responsável por coletar informações sobre as métricas de baixo nível (por exemplo utilização da CPU, uso de memória e tráfego de rede etc.), e/ou métricas de alto nível (por exemplo taxa de chegada de solicitação, tipo de solicitação, tamanho de solicitações etc.) durante a fase de monitoramento. Esses diferentes conjuntos de parâmetros de monitoramento são armazenados em uma base de conhecimento para uso por outros componentes;
- Análise (A): a fase de análise é responsável pelo processamento das informações coletadas diretamente da fase de monitoramento. Durante a fase de análise, o sistema determina se é necessário executar ações de elasticidade com base nas informações monitoradas;
- Planejamento (P): a fase de planejamento é responsável pela estimativa do número total de recursos a serem provisionados/desprovisionados na próxima ação de elasticidade. Ele segue regras que podem ser tão simples quanto uma política de ação - condição - evento, fácil de implementar e rápida de processar, ou assumem a forma de funções utilitárias, que tentam otimizar uma determinada propriedade dos sistemas gerenciados;
- Execução (E): a fase de execução é responsável pela execução das ações decididas na fase de planejamento. Ele pode ser implementado pelas APIs de automação (ou seja, atuadores) disponíveis para o ambiente e pela configurabilidade do tempo de execução do aplicativo em nuvem;
- Conhecimento (*Knowledge* - K): a base de conhecimento no modelo MAPE-K compartilha dados entre as fases MAPE reais.

Neste trabalho, aplica-se o modelo MAPE-K à adaptação autonômica de aplicações em nuvem. Um exemplo concreto de gerenciamento automático de infraestrutura é o *Auto Scaling* da Amazon [45], que gerencia quando e como os recursos de uma aplicação devem ser aumentados ou diminuídos dinamicamente. A solução apresentada neste trabalho será comparada com a proposta oferecida aos clientes do *Auto Scaling* da Amazon [45].

2.4 Técnicas de Elasticidade Automática

As técnicas de elasticidade automática de aplicações em nuvem costumam ser agrupadas de formas ligeiramente diferentes, dependendo do trabalho de revisão da literatura.

Segundo Lorido-Botran et al. [15], a classificação das técnicas de elasticidade automática é dada por:

- Regras baseadas em limites;
- Aprendizagem por reforço;
- Teoria das filas;
- Teoria de controle;
- Análise de séries temporais.

As técnicas de Teoria das filas e de Teoria de controle dependem da modelagem do sistema para determinar a necessidade futura de recursos de uma aplicação em nuvem [46]. No entanto, como o foco deste trabalho é apresentar uma solução independente da aplicação na nuvem, essas duas técnicas não são usadas neste trabalho. O aprendizado por reforço não usa nenhum conhecimento ou modelo a priori da aplicação, entretanto, essa técnica sofre de longas fases de aprendizado. O tempo requerido pela técnica para convergir para uma política ótima pode ser inviavelmente longo [15]. Em razão disso, as outras duas técnicas (Regras baseadas em limites e Análise de séries temporais) foram as escolhidas para serem usadas neste trabalho, e alguns conceitos relacionados a estas técnicas serão apresentadas nas próximas seções.

2.4.1 Regras Baseadas em Limites

Os provedores de nuvem públicas oferecem soluções de elasticidade usando regras baseadas em limites, como a Amazon EC2 [47] e a Flexera [48]. A técnica de regra baseada em limites está puramente relacionada ao planejamento, ou seja, o número de recursos alocados a aplicação, na forma de máquinas virtuais, deve variar de acordo com um conjunto de regras. As decisões de elasticidade são acionadas com base em algumas métricas de desempenho e limites predefinidos [46].

Essa abordagem se tornou bastante popular devido à sua (aparente) simplicidade. No entanto, para decidir o valor do limite, é necessário compreender profundamente todos os parâmetros do ambiente, e da carga de trabalho correspondente. Em razão disso, a eficácia das regras sob variações de cargas de trabalho abruptas é questionável [15].

As regras baseadas em limites são puramente uma técnica para auxiliar a tomada de decisão, equivalente a fase de planejamento do modelo MAPE-K (ver Seção 2.3). O número de VMs atribuídas à aplicação de destino variará geralmente com base em duas regras: uma para aumentar e outra para diminuir o número de VMs. As regras são estruturadas de forma similar ao Algoritmo 1.

Algoritmo 1: Algoritmo de elasticidade automática baseado em regras de limite

```
1  if x1 > limiteSup1 and/or x2 > limiteSup2 and/or ... then
2  |  if tempo > durSup segundos then
3  |  |  n ← n + r;
4  |  |  while tempo < resfSup segundos do
5  |  |  |  aguarde
6  |  |  end while
7  |  end if
8  end if
9
10 if x1 < limiteInf1 and/or x2 < limiteInf2 and/or ... then
11 |  if tempo > durInf segundos then
12 |  |  n ← n − r;
13 |  |  while tempo < resfInf segundos do
14 |  |  |  aguarde
15 |  |  end while
16 |  end if
17 end if
```

Como é possível observar do Algoritmo 1, cada regra possui duas partes: a condição, e a ação a ser executada, quando a condição for atendida. A parte da condição usa uma ou mais métricas de desempenho $x1$, $x2$, ..., como taxa de solicitação de entrada, uso da CPU ou tempo médio de resposta. Cada métrica de desempenho possui um limite superior ($limiteSup$) e um inferior ($limiteInf$). Se a condição for atendida por um determinado período ($durSup$ ou $durInf$), a ação correspondente será acionada (adicionar ou subtrair r recursos ao conjunto de tamanho n). Após executar uma ação, a solução de elasticidade automática se inibe por um pequeno período de espera, conhecido como tempo de resfriamento ($resfSup$ ou $resfInf$).

O desempenho da técnica baseada em regras depende muito desses parâmetros. No entanto, encontrar os valores apropriados para esses parâmetros é uma tarefa complicada. Um problema comum na elasticidade automática baseada em regras, que ocorre devido a um valor limite inadequado, são as oscilações no número de VMs concedidas. De fato, os parâmetros $durSup$ e $durInf$ são introduzidos para diminuir o número de ações de elasticidade, e reduzir as oscilações da VM, assim como os parâmetros $resfSup$ e $resfInf$ [15]. O significado de definir esses limites ($limiteSup$ e $limiteInf$) é determinar as tendências, pois as ações de elasticidade poderiam ser tomadas de acordo com essas tendências. As regras baseadas em limites são usadas por abordagens reativas, as quais se caracterizam pelos seguintes passos [15]:

1. as métricas ($x1$, $x2$, ...) são extraídas a partir de uma ferramenta de monitoramento (monitor);
2. a coleta dos dados, em tempo de execução, do monitor; e
3. a ação, conforme as regras.

Com o objetivo de tornar a solução de elasticidade proposta neste trabalho menos dependente da experiência e do conhecimento da equipe que estabelecerá os limites para as aplicações em nuvem, este trabalho tratará de limites definidos pelo histórico da aplicação.

2.4.2 Análise de Séries Temporais

A análise de séries temporais abrange uma ampla variedade de métodos para detectar padrões, e prever valores futuros em sequências de pontos de dados. A precisão no valor da previsão (por exemplo, número futuro de solicitações ou utilização média da CPU) dependerá da seleção da técnica adequada, além da configuração adequada dos parâmetros, especialmente, a janela do histórico e o intervalo de previsão. A análise de séries temporais é o principal facilitador de técnicas proativas de elasticidade automática [15].

Séries temporais são aplicadas na elasticidade automática para prever a futura carga de trabalho ou os recursos necessários. As regras predefinidas são projetadas na fase de planejamento [49], e otimizam a alocação de recursos na fase de análise do ciclo MAPE-K [50]. Na sequência serão apresentados os conceitos das abordagens de análise de séries temporais utilizadas neste trabalho.

Assim, neste trabalho é utilizado uma abordagens de análise de séries temporais conhecida como modelo Autorregressivo Integrado de Médias Móveis (*AutoRegressive Integrated Moving Average* - ARIMA) [51]. O modelo ARIMA é uma combinação de três partes. Primeiramente, o AR, ou parte autorregressiva, que é a parte que procura modelar a série com base em sua própria autocorrelação. Em seguida, a parte de média móvel, (*moving average* - MA) tenta modelar surpresas ou choques locais na série temporal, enquanto a parte I abrange a diferenciação [46]. A parte Autorregressiva será apresentada a seguir.

Autorregressão

Na autorregressão - AR , o valor futuro é previsto usando a soma ponderada linear da observação anterior p da série temporal, conforme indicado na Equação 2.1.

$$\hat{x}_{t+1} = b_1 x_t + b_2 x_{t-1} + ... + b_p x_{t-p+1} + \epsilon_t \tag{2.1}$$

Onde $\hat{x}_{t+1}$ representa o valor de previsão do próximo valor da série temporal, p representa o número de observações passadas na Equação de $AR(p)$, os valores indicados por

b_1, b_2, ..., b_p são os fatores de peso e o valor ϵ_t é o ruído branco adicionado na fórmula [46].

Seguindo com a descrição do modelo ARIMA, a parte MA tenta modelar surpresas ou choques locais na série temporal. Esta parte MA é conhecida como média móvel e será apresentado a seguir.

Média Móvel

A média móvel, *moving average* - MA, é um método amplamente utilizado para suavizar uma série temporal para filtrar o ruído da flutuação aleatória e fazer previsões. A fórmula geral da MA é descrita na Equação 2.2.

$$\hat{x}_t = \epsilon_t + a_1\epsilon_{t-1} + a_2\epsilon_{t-2} + ... + a_q\epsilon_{t-q} \tag{2.2}$$

Na qual $\hat{x}_t$ é um valor linearmente dependente de um finito número q de ruídos brancos (ϵ) anteriores. Além disso, fatores de peso iguais ou diferentes são atribuídos aos valores indicados por a_1, a_2, ..., a_q [46].

Um modelo hibrido de um AR da ordem p e um MA da ordem q é conhecido como modelo Autorregressivo e de Médias Móveis (*AutoRegressive Moving Average* - ARMA) e será apresentado a seguir.

Modelo Autorregressivo de Médias Móveis

O modelo Autorregressivo de Médias Móveis, ARMA , é um modelo híbrido descrito conforme a Equação 2.3.

$$x_t = b_1x_{t-1} + ... + b_px_{t-p} + \epsilon_t + a_1\epsilon_{t-1} + ... + a_q\epsilon_{t-q} \tag{2.3}$$

Onde $\hat{x}_t$ representa o valor de previsão do próximo valor da série temporal, p representa o número de observações passadas, os valores indicados por b_1, b_2, ..., b_p são os fatores de peso para esses valores passados, além de q ruídos brancos (ϵ) anteriores. Além disso, fatores de peso iguais ou diferentes são atribuídos aos ruídos brancos pelos valores indicados por a_1, a_2, ..., a_q.

O modelo $ARMA(p, q)$ pode ser usado puramente como modelo $AR(p)$ ou $MA(q)$ usando $ARMA(p, 0)$ e $ARMA(0, q)$, respectivamente. O modelo ARMA é mais adequado para séries temporais estacionárias [46]. A extensão do modelo ARMA adequada para séries temporais não estacionárias é o modelo ARIMA, o qual será descrito mais detalhadamente a seguir.

Modelo Autorregressivo Integrado de Médias Móveis

O modelo Autorregressivo Integrado de Médias Móveis (*Auto Regressive Integrated Moving Average* - ARIMA) é uma combinação de três partes. O AR, ou parte autorregressiva, a parte MA, ou médias móveis, enquanto a parte I abrange a diferenciação. Se a série temporal não for estacionária, uma das formas de transformá-la em série estacionária é por diferenciação, método usado neste trabalho [46].

Um comportamento homogêneo da série temporal não estacionária pode às vezes ser representado por um modelo que exige que a $d-ésima$ diferença do processo seja estacionária [51]. O modelo $ARIMA(p, d, q)$ pode ser gerado somando ou "integrando"o processo ARMA estacionário em peso, d vezes [51].

Uma série temporal w_t é uma série temporal estacionária se houver um número inteiro não negativo d que satisfaça a Equação 2.4. Caso contrário, ela precisa ser suavizada [52].

$$\nabla^d w_t = x_t \tag{2.4}$$

Definindo o operador autorregressivo $\phi(B)$ pela Equação 2.5, o polinômio $\varphi(B)$ pode ser definido pela relação com o operador autorregressivo, conforme apresentado na Equação 2.6.

$$\phi\left(B\right) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_P B^q = 1 - \sum_{K=1}^{q} \phi_K B^K \tag{2.5}$$

$$\varphi(B) = \phi\left(B\right)\left(1 - B\right)^{\mathrm{d}} \tag{2.6}$$

Os polinômios $\varphi(B)$ e $\phi(B)$ são relativamente primos, com $\varphi(p)\phi(q) \neq 0$. Portanto, ωt, com $t = 0,1,2,3\ldots$ deve satisfazer a Equação 2.7 na qual αi é a sequência de ruido branco, e $|B| \leq 1$, ωt, com $t = 0,1,2,3\ldots$ é uma sequência de média móvel autorregressiva com diferenciação, denotada como modelo $ARIMA(p, d, q)$, em que d é a ordem das diferenças.

$$\varphi\left(B\right) x_i = \phi\left(B\right) \alpha_i \tag{2.7}$$

Determinar parâmetros de regressão e parâmetros de suavização é a chave para o modelo ARIMA. O modelo $ARIMA(p, d, q)$ determina os valores dos parâmetros q e p com base, principalmente, na função de autocorrelação (*AutoCorrelation Function* - ACF) e na função de autocorrelação parcial (*Partial AutoCorrelation Function* - PACF), respectivamente. O modelo ótimo é determinado pelo critério de informação de Akaike, e pelo critério de informação bayesiano [52].

Um outro método estatístico para análise de séries temporais usado neste trabalho é o modelo Holt-Winters, baseado em suavização exponencial, o qual será apresentado a seguir.

Suavização Exponencial

A suavização exponencial é uma grande classe de modelos de previsão que usam séries temporais [53]. A técnica mais simples dessa classe é a Suavização Exponencial Simples (SES), a qual busca padrões nos dados históricos e tenta mapear o comportamento dos dados [54]. A suavização exponencial simples é adequada para a previsão de dados sem tendência clara ou padrão sazonal. A suavização exponencial simples precisa selecionar dois valores de parâmetros, sendo o primeiro deles o α, o qual é usado para diminuir a variação aleatória (ruído branco) dos dados históricos, e o α também representa a ponderação aplicada às amostras mais recentes da série temporal. Isso permite identificar melhor os padrões e os níveis que podem ser usados para estimar a demanda futura. Por outro lado, $L0$ é o nível (ou o valor suavizado) da série no tempo inicial, o que significa que é a primeira amostra da série temporal após a aplicação do peso calculado usando α [55].

Para os métodos a seguir, geralmente, há mais de um parâmetro de suavização e mais de um componente inicial a ser escolhido. Em alguns casos, os parâmetros de suavização podem ser escolhidos de maneira subjetiva, sendo o valor dos parâmetros de suavização definidos com base na experiência anterior da pessoa designada para essa tarefa. No entanto, uma maneira mais confiável e objetiva de obter valores para os parâmetros desconhecidos é calculá-los a partir dos dados observados. Os parâmetros desconhecidos e os valores iniciais para qualquer método de suavização exponencial podem ser estimados, minimizando-se a soma dos erros quadráticos [56]. A Equação 2.8 descreve este modelo, onde α é o coeficiente de suavização, y_{t-j} é o valor suavizado observado no momento $t-j$, $L0$ é o valor do nível da série no tempo inicial, e $\hat{y}_{t+1|t}$ é o valor da previsão.

$$\hat{y}_{t+1|t} = \sum_{j=0}^{t-1} \alpha(1-\alpha)^j * y_{t-j} + (1-\alpha)^t * L0 \tag{2.8}$$

O método de suavização exponencial simples foi estendido por Charles C. Holt [57], o qual criou o método de suavização exponencial dupla, também conhecido como método Holt [57]. Esse método permite trabalhar com a tendência dos dados, ao inserir um novo componente no método de previsão [56]. O método Holt é baseado em uma média móvel ponderada exponencialmente, que é um meio de suavizar flutuações aleatórias com as seguintes propriedades desejáveis: o peso em declínio é colocado em dados mais antigos, facilidade de calcular, e necessidade de pequena quantidade de dados históricos. O método

Holt é claramente um modo de comportamento sensível ao lidar com um simples problema de previsão. Os parâmetros desconhecidos dos métodos de Holt também são estimados pela soma dos erros ao quadrado [55]. A Equação 2.9 descreve o método de Holt, o qual é composto de dois componentes, sendo que o da esquerda é o responsável por suavizar as séries temporais, e o da direita é o responsável por detectar a tendência, na qual L_{t-1} é o componente de suavização no momento $t-1$, h é o horizonte de previsão, β é o parâmetro de suavização para tendência, α é o parâmetro de suavização, sendo que $0 \leq \alpha, \beta \leq 1$ e $\hat{y}_{t+1|t}$ é o valor da previsão [56]

$$\hat{y}_{t+h|t} = \alpha y_t + (1-\alpha)(L_{t-1} + \beta) + h * [\beta(L_t - L_{t-1}) + (1-\beta)] \tag{2.9}$$

A extensão do método Holt para capturar a sazonalidade é o modelo Holt-Winters [58], o qual adiciona um novo componente para tratar a sazonalidade. A Equação 2.10 descreve o modelo de Holt-Winters.

$$\hat{y}_{t+h|t} = L_t + h * T_t + S_{t-m+h_m^+} \tag{2.10}$$

Neste método estendido, a previsão para o instante de tempo h é $\hat{y}_{t+h|t}$ e esse valor é encontrado ao se realizar suavização exponencial tripla: para o nível L_t (Equação 2.11), para a tendência T_t (Equação 2.12), e para a sazonalidade S_t (Equação 2.13). Os parâmetros de suavização são α, β e γ, respectivamente. Para registrar o período de sazonalidade, m é usado, por exemplo $m = 24$ para dados diários, em que cada dado agrupa informações de 1 hora. O valor h_m^+ é igual a $((h-1) \mod m) + 1$, para garantir que as estimativas do índice sazonal sejam feitas usando o último período da amostra de dados [59].

$$L_t = \alpha(y_t - S_{t-m}) + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{2.11}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{2.12}$$

$$S_t = \gamma(y_t - L_t) + (1-\gamma)S_{t-m} \tag{2.13}$$

Os parâmetros do modelo Holt-Winters podem ser estimados de modo automatizado ao se maximizar a função de verossimilhança gerada a partir dos dados históricos [56].

Na Seção 2.4.3 serão apresentadas as Redes Neurais Artificiais, uma alternativa aos métodos estatísticos para análise de séries temporais.

2.4.3 Redes Neurais Artificiais

O conceito de Redes Neurais Artificiais (RNA) consiste em uma analogia entre células nervosas vivas e um processo eletrônico binário [60]. As RNAs são utilizadas em soluções de problemas devido a sua capacidade de aprender. O treinamento é a capacidade da rede de ajustar os parâmetros para alcançar os resultados esperados, dado um conjunto de padrões específicos. Tais padrões de treinamento são constituídos de informações que se espera que a rede aprenda. Os ajustes dos parâmetros são realizados com a atribuição de pesos das conexões que interligam os neurônios [61].

Assim sendo, o treinamento de uma rede pode ser supervisionado ou não supervisionado. No treinamento supervisionado, o ajuste dos parâmetros é realizado de maneira que, dado uma entrada padrão, o valor é calculado para gerar uma saída. As entradas e a saída desejadas são fornecidas por um supervisor. Isso é feito para ajustar os parâmetros da rede a encontrar uma ligação entre os pares de entrada e de saída fornecidos. Ou seja, são fornecidos, previamente para a rede, os valores de entrada e de saída. No treinamento não supervisionado, o conjunto padrão de treinamento possui apenas entradas, cuja saída padrão não é previamente conhecida [60].

Uma RNA contém uma estrutura com pequenas unidades de processamento (ou neurônios) que são unidos entre si por conexões ponderadas. Ainda seguindo a analogia do modelo biológico, essas unidades de processamento representam os neurônios, e os pesos das conexões representam a força das sinapses entre os neurônios. A rede é ativada fornecendo uma entrada para alguns ou todos os neurônios. Essa ativação se espalha por toda a rede ao longo das suas conexões ponderadas, e a atividade elétrica dos neurônios biológicos atinge "picos". Em outras palavras, a RNA é uma combinação de neurônios artificiais, conexões e algoritmo de aprendizagem. A caracterização do agrupamento de neurônios é dada pela topologia da rede e deve considerar alguns aspectos, como o número de camadas da rede, o número de neurônios por camada, o tipo de conexão e o grau de conectividade entre os neurônios [61].

Os neurônios podem ter conexões diretas ou recorrentes. A conexão direta não permite que a saída de um neurônio seja utilizada como entrada em um neurônio de uma camada anterior a sua, diferentemente da conexão recorrente que permite a entrada a um neurônio de camada anterior à sua [60]. A RNA de conexão direta utilizada neste trabalho é Perceptron Multicamadas, o qual será apresentado a seguir.

Perceptron Multicamadas

A Rede Neural direta mais difundida é a Perceptron Multicamadas [62]. Uma das principais características da arquitetura do Perceptron Multicamadas (*Multi Layer Per-*

ceptron - MLP) se refere à função de ativação, que é responsável por ativar ou não a saída de um neurônio, de acordo com o valor da soma ponderada das suas entradas [60]. A função de ativação mais utilizada é a sigmoidal logística [63].

Uma MLP é composta por, no mínimo, três camadas, conforme estrutura apresentada na Figura 2.6. Essas camadas são a de entrada, a(s) oculta(s), e a camada de saída, que são descritas a seguir [64]:

- Camada de Entrada: composta por neurônios que recebem os valores de entrada;
- Camada(s) Oculta(s): composta por neurônios que dividem o problema em outros problemas menores, sendo que estes são linearmente separáveis;
- Camada de Saída: composta por neurônios computacionais, que fazem a rotulação ou mapeamento dos dados.

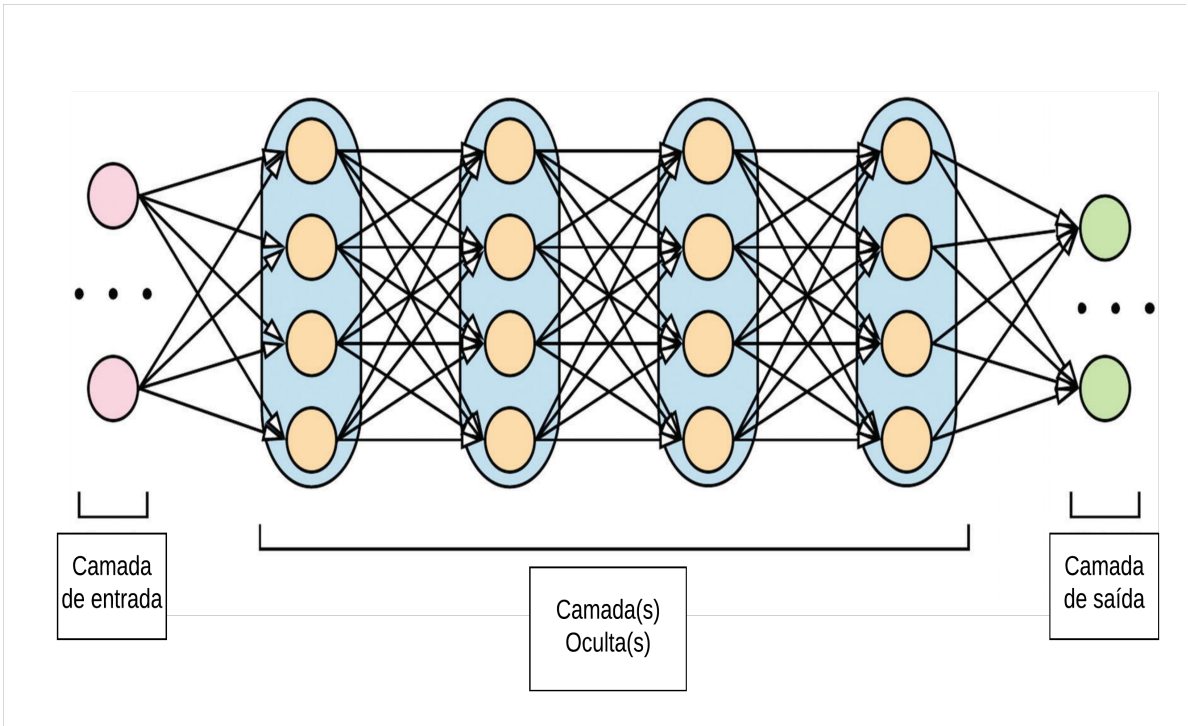

Figura 2.6: Estrutura de um *Perceptron* de Múltiplas Camadas, adaptado de [64].

Em uma MLP o processamento realizado por cada neurônio sofre influência do processamento dos neurônios anteriores conectados a ele. Devido a isso, conforme ocorre o processamento dos neurônios de cada camada em direção à saída, as funções tornam-se mais complexas. Assim sendo, pode-se dizer que as camadas ocultas desempenham a função de detectores de característica, pois geram uma codificação interna dos padrões de entrada e que servirão para gerar a saída da rede [60].

Esta rede é caracterizada por utilizar o *back-propagation*, que é o algoritmo de propagação retroativa de erro. O processo de aprendizado ocorre através dos ajustes dos pesos das sinapses da rede, de acordo com o erro existente entre o valor real e o predito [65].

Além da RNA de conexão direta, este trabalho faz uso de uma RNA de conexão recorrente, que é a Rede Neural Recorrente Baseada em Memória Longa de Curto Prazo, a qual será apresentada a seguir.

Rede Neural Recorrente Baseada em Memória Longa de Curto Prazo

Uma arquitetura padrão de RNA de conexão Recorrente tem uma capacidade limitada de acesso às primeiras camadas de informações da rede. Essa dificuldade de acesso ocorre devido à fuga de gradiente [62], conforme exemplificado na Figura 2.7.

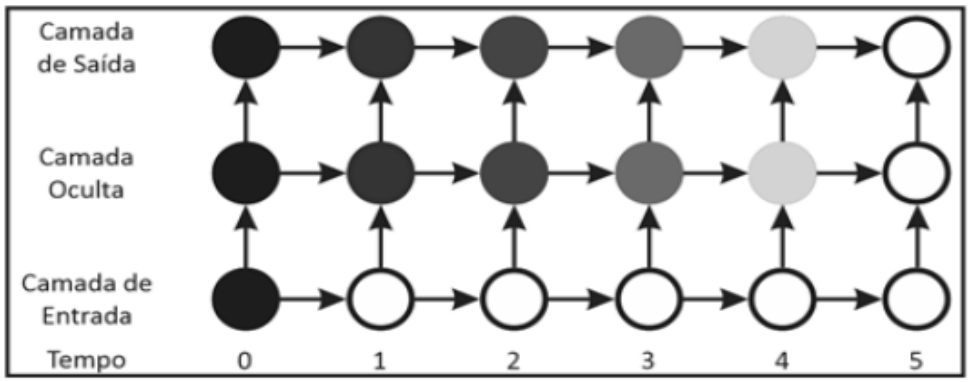

Figura 2.7: Fuga de Gradiente de uma RNA Recorrente, adaptado de [62].

Na Figura 2.7 as variações entre os círculos escuros para os círculos claros indicam a sensibilidade variando no tempo. A sensibilidade do neurônio representa a sua capacidade de armazenar informações na rede. Com isso, é possível perceber que, conforme o tempo avança, a sensibilidade diminui, sendo representada pelos círculos claros. Isso ocorre à medida que novas entradas substituem as ativações da camada oculta anterior, e assim, as primeiras entradas da rede são "esquecidas" [62].

Para atenuar o problema de fuga de gradiente, surgiu um novo tipo de RNA recorrente, baseada em blocos de Memória Longa de Curto Prazo (*Long Short-Term Memory* - LSTM). Os blocos LSTM são responsáveis por armazenar e acessar informações durante longos períodos de tempo. Uma rede LSTM é semelhante a uma RNA padrão, exceto que as unidades de soma na camada oculta são substituídas por blocos de memória [62].

Uma Rede Neural LSTM é uma arquitetura RNA de conexão recorrente que possui blocos LSTM. Um bloco LSTM pode ser considerado como uma unidade de rede complexa e inteligente, pois cada um deles é capaz de lembrar informações referentes a um longo período de tempo. A capacidade que um bloco de memória possui para lembrar

informações é possibilitada pela estrutura associada, a qual determina quando a entrada é significativa o suficiente para lembrar, quando deveria continuar a lembrar ou esquecer as informações, e quando deve produzir a saída de informação [63].

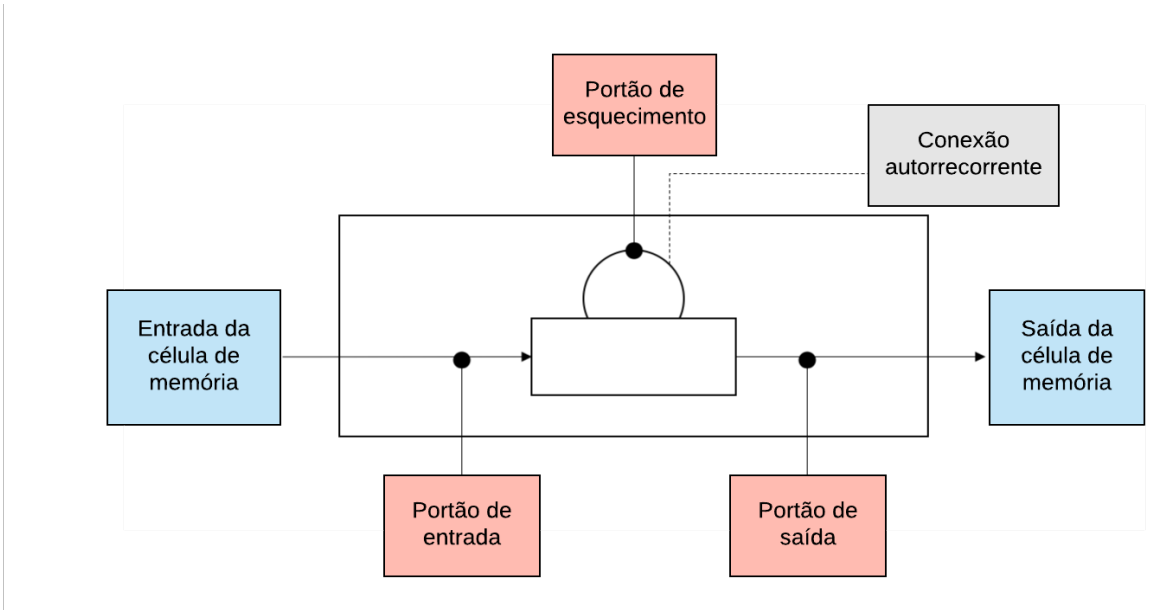

Figura 2.8: Célula de Memória LSTM, adaptado de [66].

Cada bloco de memória LSTM contém um ou mais blocos de memória autoligadas, e três unidades multiplicativas, representada por: portão de entrada, portão de esquecimento e portão de saída. Analogamente, essas unidades podem representar respectivamente as operações de gravação (entrada), reinicialização (esquecimento) e leitura (saída) [66], conforme apresentado na Figura 2.8.

As unidades multiplicativas de um bloco LSTM permitem o armazenamento e o acesso às informações durante longos períodos de tempo, reduzindo assim, o problema de fuga de gradiente. Por exemplo, enquanto o portão de entrada permanece fechado, possuindo ativação perto de 0, a informação contida no bloco de memória não será substituída por novas entradas que cheguem à rede. Com isso, a informação contida nesse bloco pode ser disponibilizada à rede, posteriormente. Essa informação é disponibilizada à rede ao abrir o portão de saída [62]. Essa preservação das informações de dados, ao longo do tempo em uma rede LSTM, é ilustrada na Figura 2.9.

Na Figura 2.9 os neurônios de cores escuras indicam máxima sensibilidade, enquanto os neurônios claros indicam ausência de sensibilidade. Os círculos escuros são neurônios que foram ativados, pois o seu portão estava aberto. Os círculos claros são referentes às células inativas. A sensibilidade da camada de saída pode ser ligada e desligada pelo portão de saída sem afetar o neurônio [62].

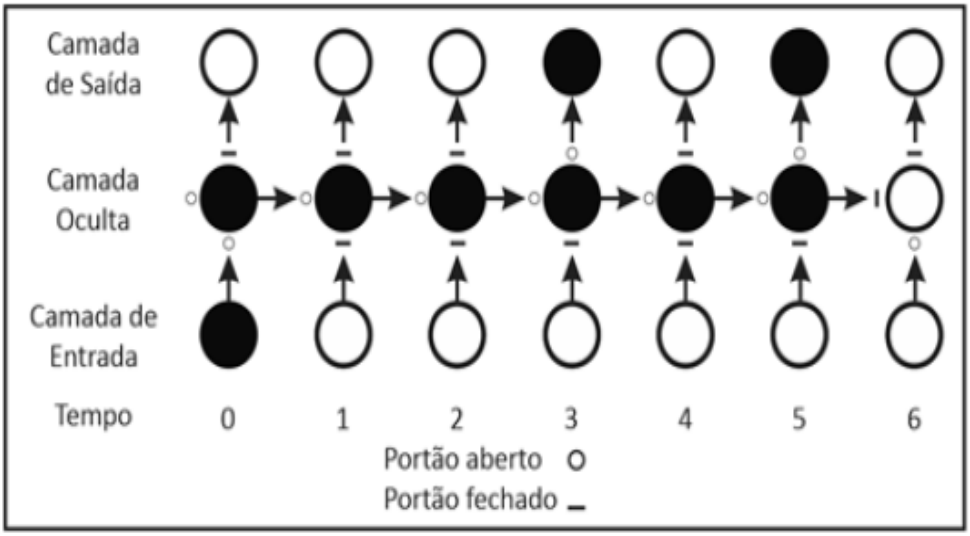

Figura 2.9: Preservação das Informações do Gradiente por Memória Longa de Curto Prazo, adaptado de [62].

2.5 Considerações Finais

Este capítulo forneceu fundamentos teóricos essenciais sobre os conceitos de computação em nuvem e elasticidade, com os quais se permite entender a proposta desta dissertação. Também foram apresentadas as técnicas de elasticidade, com foco em previsões de séries temporais, que serão usados posteriormente em um estudo de caso para apoiar a solução proposta.

O próximo capítulo apresentará o estado da arte das técnicas que estão sendo usadas na literatura para a elasticidade automática em ambientes de nuvem.

Capítulo 3

Trabalhos Relacionados

O objetivo deste capítulo é apresentar o estado da arte das técnicas que estão sendo usadas na literatura para a elasticidade automática em ambientes de nuvem. Para isto, a Seção 3.1 apresentará o processo de revisão sobre o tema elasticidade automática em ambientes de nuvem. A Seção 3.2 apresentará os trabalhos relacionados a elasticidade automática em nuvem utilizando o modelo reativo como parte da solução. A Seção 3.3 apresentará como os trabalhos relacionados buscam aumentar a precisão das previsões em modelos proativos. Por último, a Seção 3.4 apresentará uma comparação entre os trabalhos relacionados.

3.1 Processo de Revisão

Esta seção apresenta uma visão geral da literatura sobre o tópico de pesquisa elasticidade automática em computação em nuvem. A seguir será exposto o processo seguido para identificação dos trabalhos mais relevantes e relacionados a proposta deste trabalho.

A definição das questões de pesquisa é uma tarefa importante na criação do processo de revisão, uma vez que elas são usadas para orientar a revisão e ajudar a alcançar objetivos propostos. Assim, as questões de pesquisa definidas foram:

- Q1: Quais são as alternativas, baseadas em abordagens reativas, às soluções de elasticidade automática baseadas unicamente em regras de limites fixos? Essa questão identifica como são tratadas as variações da técnica baseada em limites fixos;
- Q2: Quais são os recursos utilizados para aumentar a precisão das previsões em soluções de elasticidade automática em nuvem?

Para responder as perguntas de pesquisa, os artigos foram pesquisados no ScienceDirect [67], IEEE Xplorer [68], ACM [69] e Scopus [70]. Essas bases foram selecionadas por

serem as maiores da área. Os termos chaves indicados nas questões de pesquisa como computação em nuvem, elasticidade automática, proativo, reativo (em língua inglesa), além de seus sinônimos foram utilizados como orientação para a pesquisa nas respectivas bases.

Como cada uma dessas bibliotecas possui um mecanismo de pesquisa próprio, logo as expressões de pesquisa utilizadas foram distintas. A Tabela 3.1 mostra as expressões usadas em cada motor de busca. A construção dessa sequência de pesquisa exigiu alguns cuidados, pois os resultados podem variar dependendo dos termos e dos operadores utilizados. Entretanto, vários testes foram realizados para definir a melhor sequência a ser aplicada nesta pesquisa, a fim de encontrar trabalhos representativos. As expressões criadas foram consultadas nos resumos das publicações.

Tabela 3.1: Expressões de Pesquisa Usadas nas Bibliotecas Digitais.

Repositório de Trabalhos	**Expressão de busca**
ScienceDirect	Title, abstract or author-specified keywords: “cloud computing”and auto and (scaling or scale or scaler)
ACM	recordAbstract:(+"cloud computing”auto scale scaler scaling) AND keywords.author.keyword: (scale scaler scaling auto cloud computing)
IEEE Xplore	((“Abstract": “auto”AND “Abstract": “cloud computing”AND (“Abstract": scaling OR “Abstract": scale OR “Abstract": scaler)))
Scopus	ABS (“cloud computing” AND auto AND (scaling OR scale OR scaler))

Como resultado da pesquisa realizada por meio das expressões indicadas na Tabela 3.1, foram encontrados 348 trabalhos, dos quais 39 no SciencDirect, 94 no IEEE Xplorer, 200 no Scopus e 15 na ACM.

Para selecionar os mais relevantes para responder as perguntas de pesquisa, alguns critérios de seleção e de exclusão foram aplicados. No primeiro momento foram excluídos quaisquer artigos duplicados (ou seja, artigos encontrados em mais de uma biblioteca digital) e os trabalhos indisponíveis. A busca eventualmente retornou resultados sem conteúdo que também foram removidos desta pesquisa. Com o primeiro filtro, removeu-se 121 artigos: 113 artigos duplicados, e 8 artigos indisponíveis ou resultados sem conteúdo. Assim, selecionou-se 227 artigos para a próxima etapa.

Em um segundo momento, o resumo dos 227 artigos, não excluídos da etapa anterior, foram lidos. Após a leitura do resumo desses trabalhos, foi observado que alguns artigos estavam fora deste escopo de pesquisa. Esses trabalhos não apresentavam indicação de resposta para nenhuma das perguntas de pesquisa. Nesta etapa, excluiu-se 173 artigos, e selecionou-se 54 artigos para a última etapa, na qual os artigos que fornecem respostas para as questões de pesquisa foram selecionados.

Para aplicar o último filtro, foi necessário ler o artigo completo. Nesta última etapa, os 54 artigos selecionados foram lidos, e 34 artigos foram removidos porque não responderam a nenhuma questão de pesquisa. Assim, 20 artigos selecionados ao final, ajudaram a responder a pelo menos uma das questões de pesquisa. Esses artigos serão discutidos nas seções seguintes.

A análise feita no segundo e no terceiro filtro foi subjetiva, em razão disso, quando houve dúvida se um artigo atenderia o respectivo filtro, esse artigo foi selecionado para análise, ou seja, só foram excluídos da seleção pelos filtros citados quando houve certeza de que não atendiam aos requisitos do filtro.

3.2 Soluções Baseadas em Abordagem Reativa

Soluções de elasticidade automática baseadas em abordagem reativa precisam de limites para ações de elasticidade ser cuidadosamente decididos, caso contrário, podem causar o problema de oscilação [71]. Para lidar com o problema de oscilação, são definidos certos parâmetros, como períodos de resfriamento, calma e inércia, para que nenhuma decisão de elasticidade possa ocorrer nesses intervalos de tempo [46].

Alguns pesquisadores propuseram técnicas alternativas para resolver o problema de oscilação na quantidade de VMs gerenciada pela solução de elasticidade automática. Por exemplo, o trabalho de Hasan et al. [72] usa um conjunto de quatro limites e duas durações. Desses quatro limites, dois são superiores e dois inferiores, enquanto os parâmetros durações contabilizam o tempo após cruzar o limite mais extremo (superior ou inferior), e é usado para verificar a persistência do valor da métrica acima ou abaixo dos limites superiores ou inferiores, respectivamente (Figura 3.1).

– A proposta Hasan et al. [72] de 4 limites fixos para uma aplicação conhecida pode ser útil, mas para uma solução de elasticidade automática genérica traz um aumento de complexidade com relação à solução mais conhecida de 2 limites fixos.

A RightScale [48] utiliza um processo de votação para elasticidade automática. Nesse processo, etiquetas de votação são usadas para indicar se um servidor está votando a favor ou contra em uma ação de elasticidade. O RightScale monitora as etiquetas em

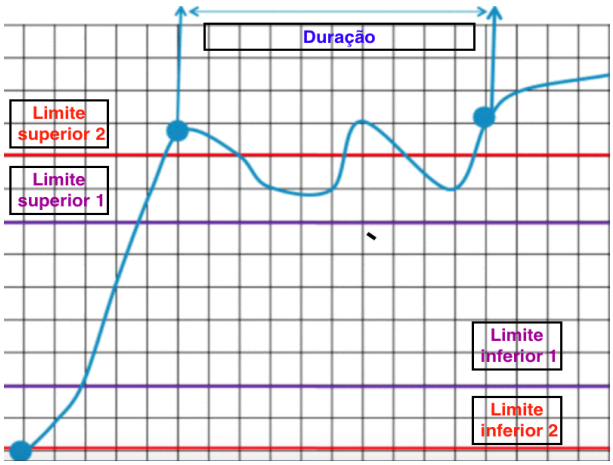

Figura 3.1: Proposta de 4 limites em um modelo reativo.

todas as instâncias de votação para determinar se uma quantidade suficiente de servidores estão votando ou não a favor de uma ação de elasticidade específica [73]. O sistema de votação RightScale é combinado com a abordagem reativa. Se a maioria das VMs concordar em uma decisão sobre aumentar ou diminuir recursos, será executada uma ação de elasticidade. O RightScale trabalha no sistema de votação, mas é altamente dependente dos valores limites definidos pelo gerenciador da aplicação, e da natureza da carga de trabalho da aplicação [46]. Esses limites são fixos ao longo da execução da aplicação.

Ghanbari et al. [74] apresentam uma implementação de uma política de elasticidade para uma aplicação usando abordagem reativa, com uma composição entre controle teórico e regras baseadas em limites fixos, em que a decisão utiliza a mecânica de votação da Rightscale [48].

– A proposta de Ghanbari et al. [74] apresenta uma composição de técnicas o que pode trazer um ganho com relação a uma técnica isolada. No entanto, o controle teórico não é recomendado para uma solução de elasticidade automática genérica.

Uma tentativa de superar esse problema foi a árvore da estratégia de Simmons et al. [75]. A árvore da estratégia proposta possui um nó pai, o qual representaria a política de nenhuma ação de elasticidade. Os três nós filhos representam três políticas de elasticidade diferentes, com base em várias heurísticas, para direcionar as ações de elasticidade automático na camada SaaS. Simmons et al. [75] apresentam uma estrutura e uma metodologia para gerenciar uma aplicação SaaS sobre a infraestrutura de um provedor de PaaS. Essa estrutura utiliza conjuntos de políticas de PaaS para implementar a política de elasticidade do provedor SaaS para sua camada de servidor de aplicações. Uma árvore de estratégia é utilizada na camada SaaS para orientar ativamente a seleção do conjunto

de políticas em tempo de execução, a fim de manter o alinhamento com o objetivo comercial do fornecedor de SaaS, especificamente para maximizar o lucro. Nesse trabalho foram apresentados resultados experimentais que refletiram positivamente a abordagem dos autores.

– A proposta de Simmons et al. [75] apresenta uma árvore da estratégia, no entanto, essas estratégias foram criadas anteriormente a execução da aplicação, com base em dados históricos. Em que pese a utilidade dessa técnica para uma solução de elasticidade automática genérica, a criação de uma árvore de estratégia deveria ser feita dinamicamente, o que demandaria um estudo mais apropriado, podendo ser abordado em um trabalho futuro.

Lorido-Botran et al. [76] sugeriram o uso de limites dinâmicos, os quais tentam superar a limitação do uso de limites estáticos. Os valores iniciais dos limites são estáticos, sendo os limites superior e inferior ajustados a medida em que se verifica se houve ou não violação do Acordo de Nível de Serviço (*Service Level Agreement* - SLA). Caso haja violação do SLA, a "janela" é diminuída (configuração 80-20 passa a ser 75-25), em caso de não violação, a janela é aumentada.

– A proposta de Lorido-Botran et al. [76] apresentou o uso de limites dinâmicos, com os quais a complexidade de configuração de uma solução de elasticidade automática é diminuída pela não necessidade de definição dos parâmetros $limiteSup$ e $limiteInf$ pelos administradores da aplicação em nuvem. No entanto, o modo de ajuste do tamanho da janela pode ser inadequado para mudanças repentinas na demanda.

Em um outro trabalho, Augustyn et al. [77] sugeriram uma abordagem reativa usando dois tipos de métricas, que são métricas de recursos (por exemplo, utilização da CPU), e métricas de aplicativos (por exemplo, tempo de resposta). Os autores afirmam que o usuário não conhece os valores ideais para os limites inferior e superior, considerando a utilização da CPU como a métrica para dimensionar o sistema. Então, eles sugeriram o uso combinado de tempo de resposta e utilização de CPU. A proposta de Augustyn et al. [77] utiliza estatísticas definidas nos valores dos T últimos tempos de execução da aplicação selecionada. O valores do quantil de ordem q (usado o de 90% nos experimentos) são usados como limites superiores, e o valores médios como limites inferiores para ação de elasticidade. Os citados valores são redefinidos após cada instância ser adicionada ou removida.

Beloglazov e Buyya [18] propuseram uma nova técnica para consolidação dinâmica de VMs com base em limites adaptáveis. Os autores partiram da premissa de que a utilização da CPU criada por cada VM pode ser descrita por uma variável aleatória com uma

distribuição específica, e a utilização da CPU de um *host*, sendo uma soma das utilizações das VMs alocadas para esse *host*, por uma distribuição aproximadamente normal, e logo podendo ser modelada pela distribuição *t-Student*. Usando essas informações e a função de probabilidade cumulativa inversa para a distribuição t, o limite superior é definido por meio dos intervalos de probabilidade ($P(cpu < 10\%)$ e $P(cpu > 90\%)$), além da média, e o desvio padrão da amostra. O mesmo vale para o limite inferior, que nesse trabalho não pode ser menor do que 30% do consumo de CPU.

– As propostas de Beloglazov e Buyya [18] e Augustyn et al. [77] apresentaram limites dinâmicos, com eles a complexidade de se configurar a solução de elasticidade automática é diminuída, pela ausência de definição dos parâmetros $limiteSup$ e $limiteInf$. O modo de ajuste dos limites baseado no histórico da aplicação parece promissor em conjunto com uma outra técnica que possa antecipar a demanda.

Assim, nota-se que é fácil implementar a elasticidade automática baseada em regras para uma aplicação específica. Logo, se a carga de trabalho e a natureza da aplicação puderem ser previstas com facilidade, é possível usar a técnica baseada em regras. Por outro lado, caso a aplicação apresente um padrão imprevisível, deve-se escolher outras estratégias de elasticidade mais adequadas [46].

Os valores fixos dos limites não são adequados para um ambiente com cargas de trabalho dinâmicas e imprevisíveis, nas quais diferentes tipos de aplicações podem compartilhar um recurso físico. O sistema deve automaticamente conseguir ajustar seu comportamento, dependendo dos padrões de carga de trabalho exibidos pelas aplicações [18]. Portanto, neste trabalho será usada uma variação da proposta de Beloglazov e Buyya [18] como técnica para o auto-ajuste dos limites para ação de elasticidade, com base em uma análise estatística dos dados históricos coletados durante a vida útil das VMs, sendo esses limites dinamicamente definidos/propostos.

3.3 Soluções Baseadas em Abordagens Proativas

A análise de séries temporais para soluções de elasticidade é bastante comum na literatura. Conforme será visto a seguir, as mais variadas técnicas são usadas e combinadas para alcançar a melhor predição possível. As técnicas passam do Modelo de Markov [78], que trata de uma cadeia de Markov oculta com os estados ocultos; de variações de Box-Jenkins (AR, MA, ARMA e ARIMA) [50] [79] [80] [81] [82] [83] [84]; Teoria das Filas [82] [85] [86]; e Regressão linear [80] [83] [85] [87] [88]. Técnicas de suavização também são bastante comuns (Holt, Holt-Winters) [80] [81] [89], e mais recentemente o uso de redes neurais [81] [90].

Antonescu e Braun [89] apresentaram uma arquitetura de gerenciamento de SLA, a qual é composta de cinco subsistemas que foram projetados para separar quatro etapas do ciclo de vida padrão de gerenciamento (processamento, planejamento, gerenciamento de informações e execução de SLA), bem como os mecanismos preditivos introduzidos pelos autores. O foco da arquitetura é suportar a abordagem dinâmica pela correlação das métricas de monitoramento. O subsistema que trata de predições prevê séries temporais correspondentes às métricas de monitoramento de SLA, detecta periodicidade de uma série temporal, determina correlações estatísticas de métricas sob demanda em tempo de execução e agenda ações de SLA. As previsões são realizadas usando o algoritmo Holt-Winters [58], que realiza uma suavização exponencial tripla dos dados, em combinação com o pacote de previsão do mecanismo de análise de dados R [91].

– A proposta de Antonescu e Braun [89] apresentou que correlação entre as métricas pode ser benéfica para aumentar a precisão das previsões, sendo essas previsões feitas por meio de métodos estatísticos. Esses métodos assumem uma relação linear entre os valores passados, e isso em alguns cenários não corresponde a realidade.

Akioka e Muraoka [78] propuseram um algoritmo que prevê a carga da CPU e da rede. Na abordagem dos autores as duas métricas de interesse (carga da CPU e carga da rede) são analisadas separadamente utilizando-se da variação sazonal na carga da CPU para uma previsão estendida da carga da CPU, assim como a variação sazonal da carga de rede para a previsão estendida da carga da rede. Além disso, Akioka e Muraoka [78] usam um meta-preditor baseado no modelo de Markov para selecionar o melhor preditor dentre quatro técnicas para previsão precisa de um passo à frente. Nos testes realizados, o método proposto previu as cargas da CPU e da rede ao longo de uma semana, e a taxa de erro média para a previsão um passo à frente foi de 6,2% para carga da CPU, e 9,4% para carga na rede.

– A proposta de Akioka e Muraoka [78] apresentou a análise de previsão para duas variáveis relacionadas a VM, tópico pouco explorado nos trabalhos deste tema. As quatro técnicas de previsão utilizadas são bem simples, sendo o modelo de Markov usado para selecionar a técnica de previsão. No entanto, para construir o modelo de Markov, foi usado o histórico de execuções antigas da aplicação.

Roy et al. [50] descreveram um algoritmo de alocação de recursos baseado em técnicas preditivas de modelo que alocam ou desalocam máquinas virtuais para a aplicação com base na otimização da utilidade da aplicação em um horizonte de previsão limitado. Esse algoritmo preditivo de modelo para previsão de carga de trabalho é usado para o *auto-scaling* de recursos. Os autores propuseram um algoritmo de alocação de recursos antecipada com base na técnica de previsão estatística ARMA [51].

– A proposta de Roy et al. [50] apresentou uma solução para antecipação da demanda por meio de previsões sendo feitas por métodos estatísticos. No entanto, esses métodos assumem uma relação linear entre os valores passados, o que em alguns cenários não corresponde a realidade.

Chen et al. [79] apresentam uma estrutura de garantia de QoS para serviços em nuvem e um método de previsão bayesiano é usado para prever a QoS do serviço em nuvem. Na estrutura proposta pelos autores, o sistema em nuvem pode monitorar e prever a QoS dos serviços em nuvem em tempo real durante a execução dos serviços. Quando os resultados da previsão de QoS mostrarem que algumas violações de QoS ocorrerão, o sistema em nuvem tomará medidas para evitar a ocorrência de violações de QoS. Os testes realizados utilizaram o software de simulação em nuvem chamado CloudSim [92] e em comparação com os métodos comuns de previsão de séries temporais, como ARIMA [51], MA [51] e Suavização Exponencial Única [93], o método de previsão bayesiano proposto apresentou uma precisão mais alta.

– A proposta de Chen et al. [79] apresentou uma solução que, de acordo com os testes realizados, superou previsões feitas por meio de métodos estatísticos. No entanto, um único método de previsão pode não ser a estratégia mais adequada para vários tipos de carga de trabalho.

Nikravesh et al. [90] propuseram um conjunto de previsões autonômicas para melhorar a precisão das previsões do sistema de elasticidade automático no ambiente de computação em nuvem. Para esse fim, o trabalho dos autores propôs que a precisão da previsão de sistemas de elasticidade automática preditivo aumentará se um algoritmo de previsão de série temporal apropriado, com base no padrão de carga de trabalho recebido, for selecionado. O conjunto proposto pode escolher automaticamente o algoritmo de previsão mais adequado com base no padrão de carga de trabalho recebida. Esse trabalho examina a influência dos padrões de carga de trabalho na precisão de três modelos de previsão de séries temporais: Máquina de Vetores de Suporte [94] e dois algoritmos de Redes Neurais Artificais: MLP [95] e MLP com redução de peso [96]. O ambiente de teste para os experimentos foi a nuvem pública da Amazon.

– A proposta de Nikravesh et al. [90] utilizou redes neurais artificiais, técnica que não assume nenhuma relação entre os dados passados para prever os valores futuros. A solução ainda utiliza um conjunto de 3 preditores, o que aumenta a probabilidade de previsão comparada com uma única técnica. No entanto, a seleção da técnica de predição é feita exclusivamente com base nos valores mais recentes, dessa forma, desconsidera o histórico de previsões.

Jiang et al. [85] propuseram um esquema de elasticidade automática que instancia as máquinas virtuais com base nas cargas de trabalho previstas para aplicações da web. Em outras palavras, o esquema proposto é usado para prever o número médio de solicitações da web em um período futuro, em que a unidade de tempo usada foi de uma hora. O principal objetivo de [85] era propor um esquema de elasticidade automática ideal no nível da VM, com compensação entre latência e custos. Os experimentos foram realizados na nuvem pública da AWS. Os dados de solicitação da web utilizados pelos autores foram uma série temporal, e a técnica de previsão utilizada por eles foi a regressão linear combinada com a teoria de filas.

– A proposta de Jiang et al. [85] utilizou uma combinação de técnicas para a solução de elasticidade automática. A solução apresenta uma proposta para o dilema desempenho (latência) x custos. Em que pese a teoria de filas ser bastante usada para modelar aplicações web, ela não é recomendada em uma solução independente de aplicação como a deste trabalho.

Yang et al. [88] propuseram um método que usa um modelo de regressão linear para prever cargas futuras de serviços baseados em nuvem. O mecanismo de elasticidade automático aumenta ou diminui o sistema de acordo com a carga de trabalho prevista. Vale ressaltar que o mecanismo de elasticidade automática usa as elasticidades horizontal e vertical.

– A proposta de Yang et al. [88] utilizou somente a regressão linear como método de previsão, assumindo dessa forma que a relação entre os valores passados podem ser modelados linearmente, sendo que essa suposição não cobre todas as cargas de trabalho. Essa proposta monitora, além das requisições dos usuários, o consumo de CPU, de memória e de armazenamento das VMs que compõem o *cluster* da aplicação.

Shariffdeen et al. [81] propuseram um método que pode ser treinado durante o tempo de execução do sistema para capturar as tendências mais recentes, e fornecer resultados suficientemente precisos para padrões de carga de trabalho drasticamente diferentes. Os autores também propuseram uma técnica de conjunto, que combina resultados do ARIMA [51], modelos exponenciais, como Holt [57] e Holt-Winters [58], e redes neurais [96].

– A proposta de Shariffdeen et al. [81] utilizou uma combinação de técnicas, sendo essas técnicas heterogêneas entre si, como técnicas de análise de séries temporais estatísticas, estatísticas com suavização e redes neurais. Assim, com um conjunto heterogêneo para a previsão, a probabilidade de se antecipar a carga de trabalho

diversificada é aumentada. Essa combinação parece ser promissora com o auxílio de um modelo reativo, dado que este pode aumentar a QoS quando as precisões das previsões estiverem baixas.

Niu et al. [84] propuseram uma solução de elasticidade automática de largura de banda que reserva dinamicamente recursos de vários *datacenters* para provedores de video sob demanda. A solução rastreia o histórico de demanda de largura de banda em cada canal de vídeo usando serviços de monitoramento em nuvem e estima periodicamente a expectativa, volatilidade e correlações de demandas para o futuro próximo, usando uma abordagem de série temporal por meio de ARIMA [51], SARIMA [51] e modelo customizado pelos autores. Eles também formularam um equilíbrio de largura de banda e uma redução nos custos de armazenamento.

Fernandez et al. [80] apresentaram um sistema de elasticidade automática que se beneficia da heterogeneidade das infraestruturas de nuvem para reforçar melhor os requisitos do cliente, mesmo sob grandes e temporárias variações de carga de trabalho. Os autores propuseram um preditor que previne violações do SLA antecipadamente, prevendo a carga de trabalho para estimar o tráfego recebido. Assim, o preditor toma os dados de monitoramento como entrada e usa diferentes técnicas de análise de séries temporais, como regressão linear para tendências lineares, o modelo ARMA [51] para pequenas oscilações, além da suavização exponencial com Holt-Winters [58], regressão automática [51] e, para tendências correlacionadas, usa o vetor autoregressivo [97].

– As propostas de Niu et al. [84] e Fernandez et al. [80] utilizaram uma combinação de técnicas estatísticas, sendo essas técnicas heterogêneas entre si, mas com todas elas assumindo uma linearidade entre os valores passados, sendo que essa suposição não atende a todas as cargas de trabalho.

Calheiros et al. [82] propuseram um módulo de previsão de carga de trabalho usando o modelo ARIMA [51], no qual aplica retroalimentação das últimas cargas de trabalho observadas para atualizar o modelo em execução. A carga prevista é usada para fornecer dinamicamente VMs em um ambiente de nuvem elástica, levando em consideração parâmetros de QoS, como tempo de resposta e taxa de rejeição.

– A proposta de Calheiros et al. [82] utilizou uma combinação de técnicas (ARIMA [51] e Teoria das Filas [98]), o que pode trazer melhores resultados do que uma única técnica. No entanto, a modelagem por meio de Teoria das Filas não é adequada para qualquer aplicação, dessa forma a combinação de técnicas deste trabalho não fará uso desse modelo.

Ali-Eldin et al. [83] introduziram dois controladores híbridos adaptativos, sendo um reativo e o outro proativo. A proposta dos autores detecta o crescimento da carga de trabalho e também não desaloca recursos prematuramente. Os resultados dos testes mostraram que o uso de um controlador reativo com um pró-ativo diminui a probabilidade de violações do SLA dez vezes em comparação com um mecanismo de elasticidade automática totalmente reativo. A abordagem proativa utiliza a regressão combinada com Teoria das Filas.

– A proposta de Ali-Eldin et al. [83] apresentou uma solução de elasticidade automática híbrida, absorvendo as vantagens das abordagens reativa e proativa. Em que pese a solução dos autores ter superado uma solução puramente reativa, em teoria, a solução poderia ser potencializada se passasse a usar limites dinâmicos ao invés de limites fixos. Contudo, a abordagem proativa utilizou teoria das filas, a qual limita a solução para algumas aplicações.

Iqbal et al. [87] propuseram uma metodologia e descreveram um sistema de protótipo para identificação automática de excesso de provisionamento em aplicativos multicamadas em ambientes de computação em nuvem. A proposta dos autores é um mecanismo híbrido de elasticidade automática, na qual ele usa uma abordagem reativa para aumento de recursos e uma proativa para redução de recursos. Os autores usaram o tempo de resposta como métrica principal, e o consumo de CPU como métrica secundária. Para a abordagem proativa, os autores escolheram a técnica de regressão para prever as necessidades futuras de um aplicativo da web. De acordo com os autores, é muito difícil identificar uma configuração com um mínimo de recursos de um aplicativo da web de várias camadas, que atenda aos requisitos de tempo de resposta para uma determinada carga de trabalho.

– A proposta de Iqbal et al. [87] apresentou uma solução de elasticidade automática híbrida, absorvendo as vantagens das abordagens reativa e proativa. No entanto, o foco dessa solução é evitar o excesso de provisionamento. Dessa forma, diferente deste trabalho em que o foco principal é evitar o subprovisionamento e, de modo secundário, evitar o excesso de provisionamento, acredita-se que utilizar a abordagem reativa e proativa nos dois sentidos (analisar o aumento ou a redução da demanda) pode ser mais benéfica para os objetivos deste trabalho do que a escolha feita por Iqbal et al. [87].

Urgaonkar et al. [86] propuseram uma técnica que emprega dois métodos que operam em duas escalas de tempo diferentes, preditiva e reativa. A abordagem preditiva aloca capacidade no período de horas ou dias. A abordagem reativa opera na escala de tempo

de minutos para responder aos picos da carga de trabalho. Os autores também propuseram um modelo analítico baseado na teoria de filas para capturar o comportamento de aplicativos com um número arbitrário de camadas. Os experimentos foram executados em uma plataforma de hospedagem baseada em Xen/Linux de 40 máquinas.

– A proposta de Urgaonkar et al. [86] apresentou uma solução de elasticidade automática híbrida, absorvendo as vantagens das abordagens reativa e proativa. O trabalho de Urgaonkar et al. [86] é exaustivo em testes para demostrar a vantagem da abordagem híbrida contra uma abordagem puramente reativa ou proativa. A proposta de Urgaonkar et al. [86] pode ser adaptada para uma solução de elasticidade automática independente de aplicação ao substituir a modelagem por meio de Teoria das Filas por uma modelagem de análise de séries temporais. O trabalho dos autores trata como entrada da solução de elasticidade automática o consumo de CPU, memória, rede, e disco, assim como este trabalho.

Diante do exposto, embora alguns autores utilizem técnicas de análise de séries temporais que possuem baixo consumo de recursos, e podem executar a previsão muito rapidamente, como AR, MA, ARMA, ARIMA e regressão, estas opções podem não ser as mais adequadas para prever carga de trabalho de certas aplicações por assumirem uma linearidade entre os valores passados.

Ao contrário dessas abordagens, os algoritmos de aprendizado de máquina, como as redes neurais, são adequados para séries temporais com tendências ou mudanças sazonais, em troca de um maior custo computacional comparada com as primeiras. Para extrair o benefício das duas abordagens, a proposta deste trabalho, visando aumentar a precisão da previsão, utiliza uma combinação de vários métodos de previsão, fazendo uma integração de redes neurais e métodos estatísticos para aproveitar a vantagem de cada abordagem, a um custo computacional intermediário.

3.4 Comparação entre os Trabalhos Relacionados

Este trabalho é comparado com os trabalhos relacionados citados em cinco aspectos diferentes, conforme resumo apresentado na Tabela 3.2. Por isso, inicialmente, compara-se qual é a abordagem utilizada pelas propostas (reativas, proativas ou híbridas), a qual é apresentada na coluna “Modelo”. Segundo, quais trabalhos utilizaram a técnica de regras baseadas em limites dinâmicos (coluna “Limites Dinâmicos”), sendo o símbolo ✓ indicando o uso da técnica e o símbolo χ indicando o não uso. Terceiro, quais trabalhos utilizaram alguma das técnicas de análise de séries temporais (coluna “Análise de Séries Temporais”), destacando-se o uso de métodos estatísticos (no qual agrupou-se os métodos

Tabela 3.2: Comparativo entre os Trabalhos Relacionados mais Relevantes.

Trabalhos	**Modelo**	**Limites Dinâmicos**	**Análise de Séries Temporais**	**Consumo**	**Avaliação**
[75]	Reativo	χ	χ	CPU	Lucro
[74]	Reativo	χ	χ	CPU	numero de sessões
[72]	Reativo	χ	χ	CPU e disco	tempo de resposta
[76]	Reativo	✓	χ	CPU	Tempo de Resposta
[18]	Reativo	✓	χ	CPU	Consumo de Energia
[77]	Reativo	✓	χ	CPU	tempo de resposta
[89]	Proativo	χ	Estatísticos	χ	tempo de resposta
[78]	Proativo	χ	χ	CPU e disco	vazão da rede
[50]	Proativo	χ	Estatísticos	χ	custo
[79]	Proativo	χ	χ	disco	tempo de resposta
[90]	Proativo	χ	Redes Neurais	χ	requisições/minuto
[85]	Proativo	χ	Estatísticos	χ	custo
[88]	Proativo	χ	Estatísticos	CPU, memória e rede	custo
[81]	Proativo	χ	Estatísticos e Redes Neurais	χ	tempo de resposta
[84]	Proativo	χ	Estatísticos	χ	vazão da rede
[80]	Proativo	χ	Estatísticos	CPU	tempo de resposta
[82]	Proativo	χ	Estatísticos	χ	tempo de resposta
[83]	Híbrido	χ	Estatísticos	χ	Custo
[87]	Híbrido	χ	Estatísticos	CPU	Tempo de resposta e vazão da rede
[86]	Híbrido	χ	χ	CPU, memória, disco, e rede	Tempo de resposta
Este trabalho	Híbrido	✓	Estatísticos e Redes Neurais	CPU, memória, disco, e rede	Tempo de resposta

que assumem a linearidade dos valores passados como o AR, o MA, o ARMA, o ARIMA, a regressão, o Holt-Winters, etc), e o uso de redes neurais artificiais. O símbolo χ indica que nenhuma técnica de análise de séries temporais foi usada no respectivo trabalho. Na quarta coluna compara-se quais trabalhos utilizaram como métrica de entrada variáveis de consumo de VMs, como CPU, memória, rede e disco (coluna "Consumo"). Para isso, considera-se aqui a métrica de entrada como as métricas que são monitoradas e avaliadas pelo decisor da solução de elasticidade automática. O símbolo χ indica que nenhuma variável de consumo de VMs foi usada no respectivo trabalho como métrica de entrada para o decisor. Finalmente, compara-se qual métrica de desempenho os autores avaliaram ao final dos respectivos experimentos para indicar o ganho das soluções propostas (coluna "Avaliação"). A Tabela 3.2 apresenta os trabalhos na ordem em que esses trabalhos foram citados neste capítulo.

Analisando a Tabela 3.2, pode-se observar que a previsão é amplamente estudada na literatura de elasticidade automática. Várias técnicas de previsão são propostas para executar a elasticidade automática. Assim sendo, a proposta deste trabalho fornece um conjunto mais heterogêneo de técnicas de previsão para aumentar a precisão da previsão. Logo, esta dissertação aborda a fraqueza de trabalhos relacionados, a fim de melhorar o mecanismo de elasticidade automática para ambientes de computação em nuvem.

Assim sendo, a solução proposta apresenta uma estratégia híbrida para provisionamento e desprovisionamento de máquinas virtuais, em que quatro modelos diferentes de previsão são apresentados como base da abordagem proativa desta proposta: ARIMA [51], Holt-Winters [58], MLP [95] e LSTM [99]. Os modelos utilizados por essa estratégia foram escolhidos com base em trabalhos relacionados, e também em trabalhos com foco apenas em predição em ambiente de computação em nuvem. Para a abordagem reativa que compõe esta proposta, utiliza-se limites dinâmicos, pois apresenta-se como uma opção viável a técnica reativa baseada em limites fixos utilizada pela AWS. Assim, a estratégia híbrida a ser apresentada no Capítulo 4 visa reduzir a latência do provisionamento de recursos na nuvem, e melhorar a qualidade do serviço.

3.5 Considerações Finais

Neste capítulo foram apresentados os trabalhos relacionados mais relevantes para a proposta deste trabalho. Embora existam muitos artigos na literatura sobre mecanismos de elasticidade automática usando a abordagem de séries temporais, a maioria deles usa apenas a abordagem proativa com algumas técnicas de previsão. Isso pode ser um problema porque nenhuma das poucas técnicas de previsão que compõem a solução de elasticidade automática pode ser a mais adequada para o momento, e o uso de mais técni-

cas de previsão e com características diferentes aumenta a probabilidade de uma melhor adaptação ao cenário de nuvem.

Outro problema é que, usando apenas uma abordagem reativa, o tempo para inicializar a máquina virtual é negligenciado, o que pode aumentar a probabilidade de violações de QoS. Portanto, o uso de uma estratégia híbrida de elasticidade automática de máquinas virtuais, usando séries temporais e limites dinâmicos, pode ser uma solução melhor, pois diminuirá a probabilidade de violação de QoS, enquanto não negligenciará o tempo para inicializar as máquinas virtuais.

Assim sendo, o impacto da melhoria do mecanismo de elasticidade automática deve ser bem relevante, pois reduz as taxas de rejeição de solicitações, o que a longo prazo poderá melhorar a satisfação dos usuários dos serviços oferecidos.

Capítulo 4

Solução de Elasticidade Automática Proposta

Neste capítulo será apresentada a solução para provisão de elasticidade automática na camada de infraestrutura como serviço em ambientes de nuvem pública proposta neste trabalho. A solução propõe um método de elasticidade automática que reage a variações nas métricas típicas de hardware de uma máquina virtual (CPU, rede, disco e memória) para executar a reconfiguração do sistema com eficiência. Para isso, é utilizado uma solução híbrida de elasticidade contendo um componente Proativo para previsão do consumo das métricas das VMs baseado em dados históricos, e também é usado um componente Reativo para reduzir o impacto de possíveis previsões incorretas, de modo a se obter as vantagens de cada uma das abordagens e manter o consumo de recursos dentro de uma faixa entre o subprovisionamento e o excesso de provisionamento.

4.1 Descrição da Estratégia

Os provedores de nuvem, como a AWS [26], que ofertam serviços de nuvem, usualmente oferecem mecanismos de elasticidade automática baseados em agendamento, ou baseados em regras com limites para tomadas de decisão. Essas regras são, normalmente, baseadas na utilização de recursos em execução no provedor, como por exemplo “Adicione algumas instâncias quando a média de utilização de CPU estiver acima de 70%”. Esses mecanismos são simples e também convenientes em alguns casos, no entanto, nem sempre é fácil para os usuários selecionarem os indicadores de escala “adequados”, especialmente quando as aplicações que irão ser executadas ou em execução são complexas ou de pouco conhecimento do corpo técnico do usuário da nuvem. Em outras palavras, esses mecanismos de gatilhos fixos não resolvem realmente o problema de dimensionamento da aplicação, pois dependendo do modo como forem configurados podem nunca gerar ações

de elasticidade ou gerar ações contraditórias, problema conhecido por oscilação. Essa dificuldade é encontrada hoje pela equipe técnica do TCU. Para superar a dificuldade de se indicar a *priori* os limites para a execução de ações de elasticidade, este trabalho propõe o uso de limites dinâmicos, com o objetivo de abstrair do usuário a tarefa de indicação dos limites.

Além dos limites dinâmicos, este trabalho ainda propõe uma estratégia de antecipação da demanda por recursos de uma máquina virtual, sendo essa previsão realizada com base em dados históricos da aplicação. As métricas da máquina virtual selecionadas para avaliação neste trabalho foram as regularmente monitoradas no ambiente computacional do TCU. Em seguida, com base na utilização prevista dos recursos da VM, os recursos são alocados pela solução de elasticidade automática proposta. A utilização prevista dos recursos informa que a carga de trabalho futura exigirá menos ou mais recursos computacionais das VMs.

Considerando que o objetivo do sistema de elasticidade automática é conceder a quantidade mínima de VMs de um determinado provedor de IaaS, para fornecer um certo nível de QoS para os usuários finais da nuvem [17], entende-se que a estimativa de recursos está no centro da elasticidade automática, pois determina a eficiência do provisionamento de recursos.

Dessa forma, um dos objetivos da solução proposta é identificar a quantidade mínima de recursos computacionais necessários para processar uma carga de trabalho para determinar se operações de elasticidade serão necessárias. A estimativa de recursos permite que a solução de elasticidade automática indique rapidamente a quantidade ideal de recursos. Por outro lado, os erros de estimativa resultam em provisionamento insuficiente - o que pode levar a um processo de provisionamento prolongado, ou ao provisionamento em excesso - o que causa aumento nos custos. Para diminuir o risco associado aos erros de estimativa, este trabalho faz uso da abordagem reativa de elasticidade. Assim sendo, para este trabalho, considera-se que:

- Um provedor de nuvem IaaS é composto por um conjunto de máquinas com alguma tecnologia de virtualização;
- Um provedor de nuvem IaaS define um conjunto de tipos de instância de VMs que podem ser instanciadas e iniciadas em um dos *clusters* disponíveis;
- Cada tipo de instância caracteriza univocamente uma VM em termos de capacidade de recursos (CPU, memória RAM, largura de banda da rede etc.);
- Todas as VMs que pertencem a um *cluster* estão executando a mesma aplicação e somente essa;

- O provedor de nuvem IaaS permite o aumento e a diminuição de recursos por meio de elasticidade horizontal;
- Apenas uma VM pode ser adicionada ou removida do *cluster* de recursos associados a aplicação de interesse;
- A aplicação em execução na nuvem pode apresentar cargas de trabalho variáveis no tempo, o que significa que a quantidade de instâncias de VMs necessárias para executar adequadamente esse serviço pode variar ao longo do tempo.

Assim, a aplicação em execução na nuvem é vista como um sistema a ser provisionado dinamicamente com recursos, logo, a solução de elasticidade automática proposta pode contribuir para que essa aplicação atenda adequadamente os seus usuários.

Por uma questão de simplicidade, assumiu-se que todas as VMs são de um mesmo tipo de instância. VMs diferentes podem ser executadas, mas cada serviço pode ser executado apenas em instâncias do mesmo tipo. Assim sendo, por acreditar que o ganho obtido pelo uso de recursos heterogêneos em uma aplicação não compensa a complexidade extra necessária para o gerenciamento [15], este trabalho parte do pressuposto que apenas um tipo de VM será provisionada pelo provedor de IaaS.

4.1.1 Custo da Violação

Este trabalho, segundo indicação da equipe que gerencia o contrato de nuvem pública do TCU, considera que uma maior satisfação com o uso da nuvem será alcançada se a utilização dos recursos das VMs alocadas no provedor for mantida com bastante frequência, sem excesso de provisionamento e também sem subprovisionamento. No entanto, para a equipe do TCU, as duas condições citadas possuem custos diferentes. Logo, quando esse intervalo entre as condições de excesso de provisionamento e subprovisionamento não for possível, evitar o subprovisionamento deve ser a prioridade, por ser considerado de maior custo pela equipe que gere o contrato de nuvem pública do TCU.

Esta solução de elasticidade automática é capaz, a cada unidade de tempo, de realizar uma das duas ações de elasticidade (aumentar uma VM ao *cluster* ou diminuir uma VM do *cluster*), ou ainda não realizar nenhuma ação. Essas decisões são tomadas com base nas demandas atuais ou previstas de recursos pelas VMs monitoradas, utilizando uma abordagem híbrida que é uma mistura das abordagens reativas e proativas.

A abordagem reativa da solução proposta é baseada em limites dinâmicos e é usada, principalmente, para o intervalo em que não há métodos de previsão treinados ou os erros na estimativa de recursos pela abordagem proativa não estiverem adequados. Em outras palavras, a abordagem proativa é priorizada, mas usa-se também a abordagem reativa para garantir que o sistema sofra ação de elasticidade se necessário.

4.2 Arquitetura da Estratégia

A proposta de elasticidade automática apresentada neste trabalho considera as limitações dos trabalhos relacionados, propondo um mecanismo de elasticidade automática híbrido que é:

- Independente de Serviço - a solução só precisa monitorar a utilização de recursos da infraestrutura em que os serviços são executados, sem nenhuma consideração adicional ou restritiva;
- Configurado de Forma Dinâmica e Automática - a solução define os limites para tomada de decisão sobre ações de elasticidade dinamicamente, e em relação às tarefas de previsão, ela cria os modelos para previsão durante a execução das tarefas da aplicação (*on-line*).

A solução de elasticidade automática proposta consiste em cinco componentes principais agrupados em dois módulos: Módulo Coletor e Módulo Híbrido de Elasticidade Automática. O módulo Coletor é composto do (i) monitor e de (ii) um repositório de dados. O módulo Coletor coleta informações de uso de recursos de máquinas virtuais em um intervalo de tempo regular. O módulo Híbrido de Elasticidade Automática (composto de um (iii) componente Reativo, de um (iv) componente Proativo e de um (v) decisor, que trata se o sistema será expandido ou não) usa o histórico de uso dos recursos armazenado dentro do módulo Coletor para prever o comportamento do consumo. A inteligência da solução proposta encontra-se no módulo Híbrido de Elasticidade Automática, enquanto o módulo Coletor é o módulo de apoio da solução.

O fluxo de informações entre os módulos e os componentes da solução estão representados na Figura 4.1. No início, o Monitor se comunica com um gerenciador do provedor de nuvem e busca periodicamente (por exemplo, a cada minuto) informações sobre o estado atual da aplicação em execução (1), sendo essa informação entregue por meio de filas de mensagens assíncronas. As informações coletadas incluem a utilização da CPU, a utilização da memória, a taxa de transferência de rede e a quantidade de leituras e de escritas em disco de cada nó pertencente a um *cluster* de VMs em que a aplicação de interesse está em execução. Em seguida, essas informações coletadas são armazenadas no Repositório de Dados (2). Com essas informações, o componente Reativo pode calcular os limites para as ações de elasticidade que irão subsidiar o componente Decisor sobre se o sistema precisa sofrer ação de elasticidade, com base na utilização média calculada do sistema e em uma abordagem padrão baseada em limites (3).

O componente Proativo consulta o repositório de dados em busca de dados históricos disponíveis e com base nos modelos preditivos que fazem parte da estrutura deste componente. Esses modelos preditivos realizam as predições para cada uma das métricas de

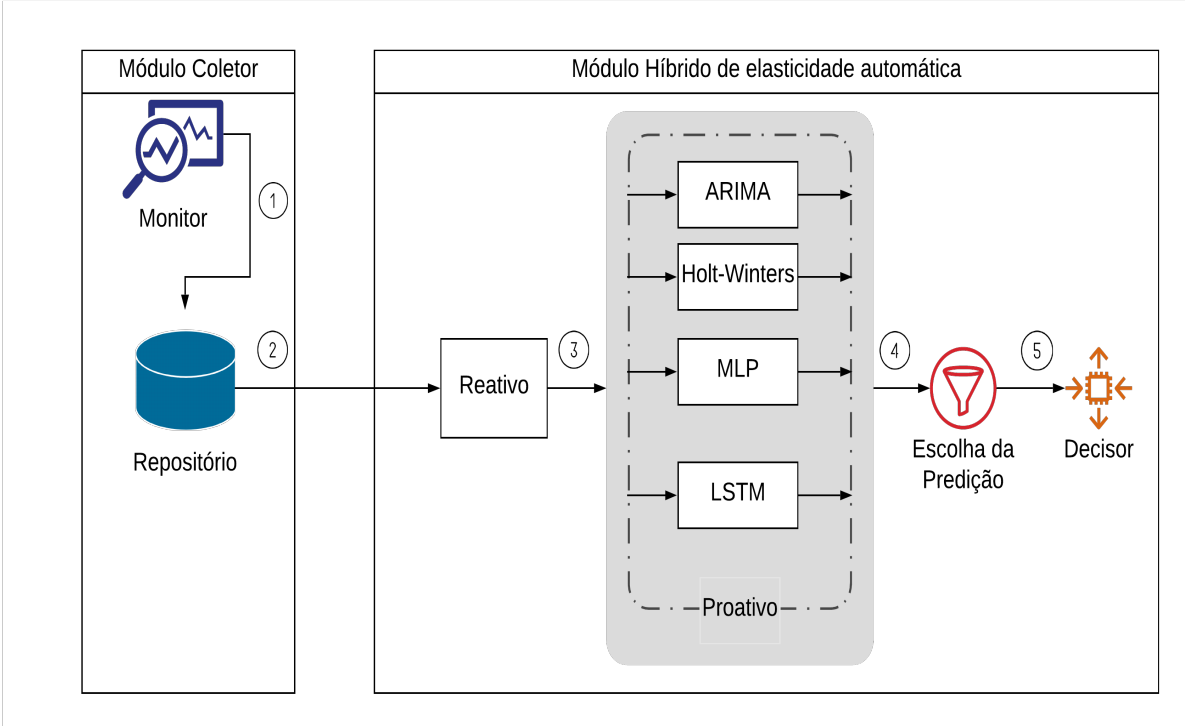

Figura 4.1: Visão geral da Solução Proposta.

interesse da aplicação (4). Em seguida, os novos valores de previsão disponíveis por cada modelo para cada métrica são enviados para o subcomponente de seleção de predição, para que se possa obter uma única predição mais precisa para o Decisor (5). Por último, o Decisor, para cada amostra coletada, verifica se o limite superior ou inferior foi atingido. Se a utilização do recurso atingiu um dos limites, o processo de elasticidade automática reativo é iniciado. Se o limite ainda não tiver sido atingido, o Decisor, com base nas informações repassadas pelo componente Proativo, decide se é necessário uma ação de elasticidade para atender uma demanda prevista ou se não fará nada.

Tendo o fluxo entre os componentes da solução sido descrito, as seções a seguir se concentram nos componentes da solução de elasticidade automática proposta.

4.2.1 Componente Monitor

Os recursos da nuvem IaaS podem ser gerenciados e utilizados com eficiência, prevendo a carga de trabalho futura ou o padrão de utilização futura de recursos [11]. A natureza e o tipo de carga de trabalho em uma nuvem pública não são determinísticos, portanto, algumas técnicas cognitivas são necessárias para prever o tipo e a natureza da carga de trabalho. [11]. A carga de trabalho, normalmente, é medida ou prevista na literatura como métricas de aplicações (por exemplo, tempo de resposta), enquanto os trabalhos

que possuem interesse em informações realistas sobre a memória e a CPU necessárias antes de se submeter essa aplicação em uma máquina virtual, buscam prever as métricas de recursos (por exemplo, utilização da CPU) [77].

O componente Monitor desta solução monitora métricas de recursos das máquinas virtuais como a utilização de CPU, de memória, quantidade de leituras e escritas em disco, além de entrada e saída de rede. Essa escolha tem explicação pelo fato dessas serem algumas das métricas monitoradas no ambiente computacional do TCU.

Todos os ambientes de computação em nuvem precisam oferecer suporte ao monitoramento de seus recursos, fornecendo medições sobre as demandas dos usuários [15]. O monitoramento dos recursos pelo provedor é feito por intermédio de sensores.

O componente Monitor interage com o sensor do provedor de nuvem AWS, chamado CloudWatch [100]. O Cloudwatch coleta diversas métricas das instâncias de VMs em execucão na AWS. Essa interação do componente Monitor com o CloudWatch é feita por meio do Boto3 [101], que é o *Kit* de Desenvolvimento de Software (*Software Development Kit* - SDK) da AWS para a linguagem Python. O Boto3 fornece uma API orientada a objetos, além de permitir acesso de baixo nível aos serviços da AWS [101].

O componente Monitor busca novos dados relativos as métricas das VMs junto ao Cloudwatch em um intervalo de 60 segundos. Esse intervalo pode ser alterado sem prejuízo para o funcionamento da solução de elasticidade automática proposta, pois entende-se que a carga de trabalho, dependendo da aplicação, pode aumentar e/ou diminuir em frequências distintas, levando a necessidade de se alterar o intervalo de tempo para a coleta de amostras. Em outras palavras, a escolha do intervalo de tempo depende da necessidade de cada aplicação [102].

Uma menção deve ser feita para a situação do componente Monitor buscar dados em um intervalo diferente do intervalo no qual o CloudWatch está coletando dados nas VMs. O CloudWatch somente coleta dados das VMs na nuvem da AWS em dois intervalos: 60 ou 300 segundos. Sendo assim, por exemplo, se for configurado para o componente Monitor buscar dados junto ao CloudWatch a cada 600 segundos, enquanto o CloudWatch está coletando dados junto as VMs a cada 300 segundos, a cada solicitação do componente Monitor, haverá dois dados disponíveis no CloudWatch. Nesse caso, deve-se especificar no Boto3 qual a forma de agregação (valor máximo, mínimo, média, 90 percentil, etc) dos valores coletados pelo CloudWatch. Neste trabalho utiliza-se a agregação por média [17].

Existem outras ferramentas úteis que acessam informações sobre a utilização dos recursos da infraestrutura de computação em nuvem. No entanto, considerando que os sensores dos provedores são instalados por padrão nas VMs, e estas já alteram o consumo de recursos dessas máquinas virtuais, optou-se por não usar uma ferramenta adicional para essa tarefa neste trabalho. Dessa forma, o monitoramento dos recursos neste tra-

balho é feito exclusivamente por meio do Amazon CloudWatch, assim como em outros trabalhos [19] [74] e [103], os quais também optaram por não usar uma ferramenta de monitoramento personalizada.

As decisões de elasticidade automática são baseadas nas informações fornecidas pelo componente Monitor, e o desempenho da solução de elasticidade automática depende da qualidade dos dados de uso de recursos. Um sistema de monitoramento ruim comprometerá a qualidade do desempenho de qualquer solução de elasticidade automática.

Vale a pena levar em consideração que pode haver várias máquinas virtuais em execução ao mesmo tempo no ambiente do provedor, fornecendo serviços diferentes. Assim, a interação entre o componente Monitor e o sensor CloudWatch é específica para capturar apenas o consumo de recursos das VMs pertencentes a um *cluster* em particular. Portanto, não importa se existem muitas aplicações diferentes em execução em outras máquinas virtuais, o componente Monitor captura apenas o consumo de recursos das VMs que estão executando a aplicação de interesse, e salva os valores capturados no componente Repositório, o componente que será apresentado na Seção 4.2.2.

4.2.2 Componente Repositório

O Repositório de Dados é um componente da solução de elasticidade automática proposta que independe do provedor de nuvem pública utilizado. Neste trabalho está sendo utilizado, por simplicidade, um conjunto de arquivos *.csv* para esta função. No entanto, também foi testado com sucesso o uso de Dynamo DB [104], como alternativa para o uso em produção desta solução proposta.

Assim sendo, o Repositório tem a função principal de guardar alguns dados salvos durante a execução da aplicação, poupando o uso de memória dos componentes que interagem com o Repositório. A interação dos outros componentes da solução com o Repositório estão descritas na Figura 4.2. Nessa figura é possível observar que o componente Monitor guarda os dados coletados do sensor do provedor no componente Repositório (1), enquanto o componente Reativo apenas consome os dados coletados pelo Monitor (2.1). Diferentemente dos demais, o componente Proativo salva os dados relativos das últimas predições (4.1), assim como consome dados salvos de predições anteriores (4.2).

4.2.3 Componente Reativo

Devido à simplicidade e ao desempenho adequado, a maioria das empresas e organizações ainda prefere usar abordagens reativas [43]. De acordo com essa visão, esta proposta também utiliza abordagem reativa como componente da solução de elasticidade automática. No entanto, o problema da abordagem reativa comumente usada é que os

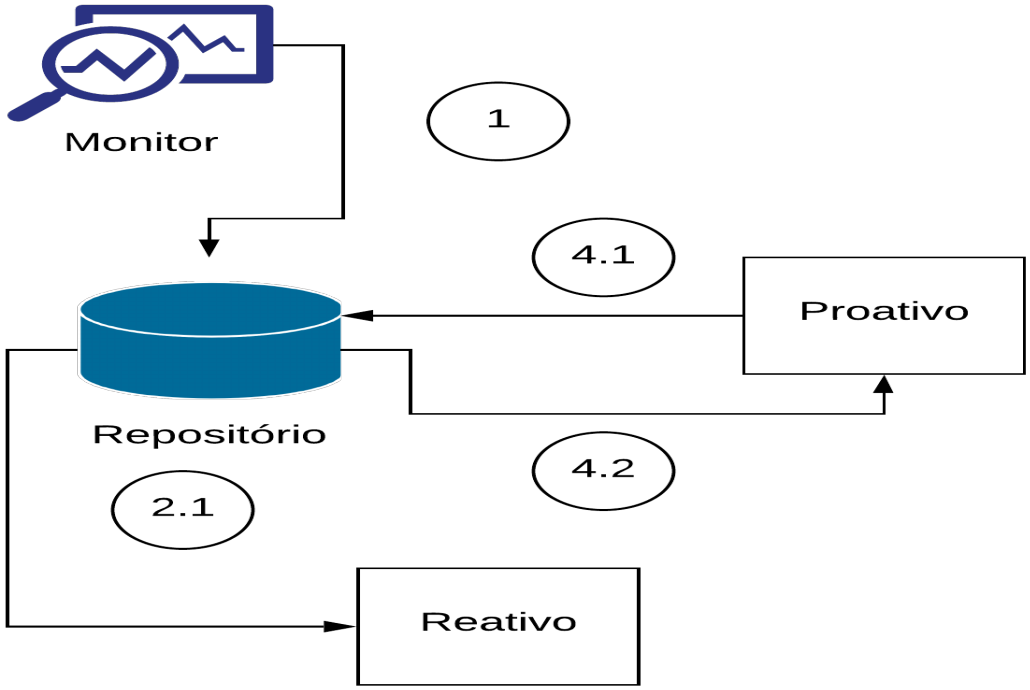

Figura 4.2: Interações dos Componentes com o Repositório.

limites para as ações de elasticidade não mudam de acordo com as cargas de trabalho dinâmicas e/ou o estado do *cluster* [16]. Portanto, as abordagens estáticas baseadas em limites frequentemente tem problema de oscilação, onde a solução de elasticidade automática frequentemente fica alterando o número de VMs (aumentando e diminuindo) para ajustar o tamanho do *cluster* [105]. Em razão disso, este componente Reativo visa a construção de limites dinâmicos para superar esses problemas. A abordagem proposta ajusta dinamicamente os limites com base no histórico das métricas coletadas da aplicação.

Primeiramente, o componente Reativo carrega os dados coletados mais atuais relativos a utilização da CPU e utilização de memória. Como os dados dessas duas métricas são apresentados em porcentagem, eles estão sempre dentro de um intervalo entre 0 e 1 ([0% - 100%]), logo nenhum tratamento adicional é feito nesses dados. No entanto, para as métricas de leitura e de escrita em disco, assim como entrada e saída de rede, que são medidos em *bytes*, só existe um valor mínimo, não existindo um valor máximo nominalmente indicado pela documentação da AWS. Dessa forma, o valor máximo a ser assumido deve ser indicado pelo proprietário da aplicação.

O consumo de disco e de rede da aplicação, para os experimentos apresentados no Capítulo 5, foram agregados da forma indicada nas Equações 4.1 e 4.2:

$$consumoDisco = qtdLeiturasDisco + qtdEscritasDisco \tag{4.1}$$

Onde *qtdLeiturasDisco* é a quantidade consumida por leituras em disco em *bytes*, *qtdEscritasDisco* é a quantidade consumida por escritas em disco em *bytes*, e a variável que agrega a soma dos dois valores anteriores é o *consumoDisco*.

$$consumoRede = qtdEntradaRede + qtdSaidaRede \tag{4.2}$$

Onde *qtdEntradaRede* é a quantidade de *bytes* no fluxo de entrada na interface de rede da VM, *qtdSaidaRede* é a quantidade de *bytes* no fluxo de saída na interface de rede da VM, e *consumoRede* é a variável que agrega a soma dos dois valores anteriores.

Com os dados atuais coletados pelo Monitor, o componente Reativo segue os passos apresentados no Algoritmo 2 para subsidiar o componente Decisor quanto as ações de elasticidade necessárias.

O processo seguido pelo Algoritmo 2 é bem comum das soluções de elasticidade automática, no qual a medição atual de cada métrica é comparada com o limite inferior (linha 1), e caso a medição seja menor do que o limite inferior, a ação indicada é uma ação de elasticidade para reduzir a quantidade de recursos (ação = -1, na linha 2). Caso a comparação da linha 1 não seja verdadeira, segue-se para comparar a medição atual de cada métrica com o seu respectivo limite superior (linha 4). Além disso, se a medição atual for maior do que o limite superior, a ação indicada é uma ação de elasticidade para aumentar a quantidade de recursos (ação = $+1$, na linha 5). No caso de nenhuma comparação ser verdadeira, o algoritmo indica que nenhuma ação deve ser tomada (ação = 0, na linha 8). Esse processo é feito para cada métrica, e os quatro valores são encaminhados para o componente Decisor.

Os valores dos limites inferiores e superiores são calculados dinamicamente a partir de dados históricos dos valores medidos, carregados do componente Repositório. O processo de geração dos limites dinâmicos será apresentado a seguir

Algoritmo 2: Abordagem reativa da solução de elasticidade automática

entrada: medição atual, limite superior e inferior
saida : Indicação para ação de elasticidade para a métrica

```
1 if medição atual < limiteInf then
2 |   ação ← −1
3 end if
4 else if medição atual > limiteSup then
5 |   ação ← +1
6 end if
7 else
8 |   ação ← 0
9 end if
```

Definição dos Limites Dinâmicos

Conforme apresentado no Capítulo 3, alguns trabalhos utilizaram características do histórico de dados como média e desvio padrão para a definição dos limites mínimo e máximo, como Augustyn e Warchal [77] e Beloglazov e Buyya [18], sendo que Beloglazov e Buyya [18] ainda caracterizaram a distribuição dos dados como aproximadamente normal. A curva normal possui características próprias como distribuição simétrica ou um valor de assimetria (grau de distorção em relação à distribuição normal) igual a zero, assim como o valor da curtose (medida de espalhamento dos valores extremos em relação à media/mediana) igual a três. Isso significa que partindo da média da curva normal, a distância máxima até o ponto mais extremo é aproximadamente 3 desvios padrões.

No entanto, a distribuição dos dados coletados pelo componente Monitor nem sempre será semelhante a curva normal. Assim, feita as considerações acima, neste trabalho serão usados nos cálculos dos limites, para cada métrica, os valores da mediana e do percentil de 90% do histórico de medições, da forma apresentada a seguir:

- Para o limite inferior:
  - mediana dos dados coletados na última hora.
- Para o limite superior:
  - percentil de 90% dos dados coletados na última hora.

A mediana e o percentil de 90% são calculados após os dados de consumo de cada métrica serem organizados em ordem crescente. Essa proposta de limites dinâmicos é similar a usada por Augustyn e Warchal [77], diferindo que nesta proposta utiliza-se como limite inferior a mediana em vez da média, pois a mediana é menos suscetível a valores extremos dos dados do que a média [106]. Além disso, no trabalho de Augustyn e Warchal [77] utilizou-se a média unicamente por limitação da ferramenta usada para monitoramento (Graphite [107]) pela solução de elasticidade automática dos autores.

O Algoritmo 3 apresenta a forma de cálculo para os limites inferiores e superiores para cada métrica. Como já citado anteriormente, os dados históricos são carregados a partir do componente Repositório (linha 1), sendo em seguida filtrados somente os dados da última hora, em razão da cobrança por VM ser feita por hora de uso (linha 2). Na linha 4, há o calculo do limite inferior ($limiteInf$) que será a mediana dos dados coletados na última hora. Caso haja um número par de valores, a mediana será a interpolação linear entre os dois valores centrais. Na linha 5, o limite superior ($limiteSup$) é calculado como o valor de percentil de 90% dos dados coletados na última hora.

Algoritmo 3: Calculo dos limites dinâmicos

entrada: histórico de dados da variável
saida : limites superior e inferior

```
1 carrega dados históricos do Repositório em dados históricos;
2 dados ⟵ Ultima Hora (dados históricos)
3
4 limiteInf ⟵ Calcula Mediana (dados);
5 limiteSup ⟵ Calcula 90Percentil (dados)
```

Esses limites dinâmicos são calculados a cada interação da solução proposta, e enviados para o componente Decisor. Este componente Decisor ainda faz uso de outra fonte de informação que é o componente Proativo, que será apresentado na Seção 4.2.4.

4.2.4 Componente Proativo

O componente Proativo é o elemento da solução de elasticidade automática proposta que trata da estratégia de antecipação à demanda. Essa antecipação é realizada ao se prever o consumo em tempo futuro com o auxílio de quatro métodos de previsão, que são: ARIMA, Holt-Winters, MLP e LSTM.

Para alcançar este fim, o componente Proativo busca junto ao componente Repositório os dados históricos de consumo das VMs que compõem o *cluster* de servidores que executam a aplicação de interesse. De posse desses dados históricos, o componente Proativo os divide em conjuntos de treinamento e teste. O treinamento dos modelos será apresentado a seguir.

Treinamento dos Modelos

A previsão acontece durante o tempo de execução do sistema, portanto, o treinamento e o teste de cada método de previsão não podem demorar muito tempo. Em razão disso, os modelos de previsão de cada um dos quatro métodos são criados em momentos distintos. O mesmo raciocínio é seguido para o treinamento dos modelos. O treinamento periódico dos modelos foi o escolhido para este trabalho. O conjunto de treinamento é composto por 80% da amostra total dos dados históricos, e o conjunto de testes é composto por 20% da amostra total dos dados históricos [56].

Com os modelos de previsão criados e treinados, a cada nova coleta de dados de consumo das máquinas virtuais no tempo t, a previsão do consumo de recursos no tempo $t + 1$ é realizada para cada um dos modelos do componente Proativo, e o mais adequado é escolhido como o melhor método para essa previsão, com base no processo de seleção do melhor modelo, descrito a seguir.

Seleção do Melhor Modelo

Para selecionar o melhor modelo de previsão, o componente Proativo usa os erros de previsão anteriores para cada modelo, assim como o erro da previsão mais recente para gerar um escore para cada modelo. Esse escore é calculado como uma soma ponderada entre o erro da previsão mais recente (Equação 4.3) e o erro médio ao longo do histórico de previsões (Equação 4.4).

O erro da previsão mais recente captura apenas a precisão com que a última previsão foi calculada pelo modelo. O erro absoluto da última previsão (e_t) pode ser definido como o módulo da diferença entre a previsão mais recente $\hat{x}_t$ e a medição mais recente x_t, conforme apresentado na Equação 4.3. Portanto, ele não captura a precisão geral. Assim, se o modelo cometeu repetidamente grandes erros em previsões anteriores, exceto a mais recente, os valores dos escores podem ser tendenciosos e levar a decisões suscetíveis a erros [81].

$$e_t = |\hat{x}_t - x_t| \tag{4.3}$$

Por outro lado, o erro médio ao longo do histórico de previsões captura o erro geral nas previsões passadas, onde todos os erros passados têm o mesmo nível de significância. O erro absoluto médio do histórico de previsões (*Mean Absolute Error* - MAE), conforme apresentado na Equação 4.4, é um média aritmética dos erros de previsões passadas, sendo o erro de previsão definido pela Equação 4.3.

$$\mathrm{MAE}_{t-1} = \frac{1}{n_{\mathrm{amostras}}} \sum_{i=0}^{n_{\mathrm{amostras}}-1} e_i. \tag{4.4}$$

Logo, se um método produziu erros maiores em previsões anteriores, o escore pode ser reduzido significativamente, mesmo que esteja produzindo as melhores previsões para valores mais recentes [81].

Ao calcular os escores com base em erros, os modelos cujos erros são baseados na última observação ignoram a precisão geral, e atribuem grande importância ao último erro de previsão. Por outro lado, os erros médios históricos pressupõem que todos os últimos erros de previsão tenham o mesmo nível de significância. Para fugir dos dois extremos, este trabalho utiliza a soma ponderada dos extremos para o cálculo do escore de cada modelo, conforme apresentado na Equação 4.5.

$$escore_t = \alpha * e_{(t)} + (1 - \alpha) * \mathrm{MAE}_{t-1} \tag{4.5}$$

O valor de α é fixo para todos os modelos e é calculado a cada interação por meio da Equação 4.6. Dessa forma, se a previsão mais recente aumentar a precisão ($e_{(t)} \to 0$), o

peso α aumentará ($\alpha \rightarrow 1$).

$$\alpha = \mathrm{MAE}_{t-1}/(e_{(t)} + \mathrm{MAE}_{t-1}) \tag{4.6}$$

O Algoritmo 4 apresenta como ocorre a seleção do modelo mais adequado para cada uma das métricas pelo componente Proativo, a partir dos valores dos escores de cada modelo. Assim sendo, os quatro modelos de previsão realizam suas predições para o

Algoritmo 4: Seleção do modelo mais adequado

entrada: escores para os modelos,
saida **:** predição do melhor modelo

```
1 for Cada modelo do
2 |  Lista Modelos ← (Modelo,escore)
3 end for
4 melhor modelo ← Menor Escore (Lista Modelos)
5 predição atual ← Predição Atual (melhor modelo)
```

momento futuro mais próximo, mas o componente Proativo seleciona somente um deles, o que possui o menor escore (linha 4), para predizer o próximo valor de utilização de cada métrica (linha 5). Isso quer dizer que o método de previsão escolhido para predizer o consumo de CPU pode ser distinto da métrica escolhida para predizer o consumo de rede, por exemplo.

A seleção do modelo mais adequado avalia continuamente as opções de previsão disponíveis e escolhe por meio da lógica apresentada a previsão de consumo para cada métrica em um futuro próximo. Com essa previsão calculada, o Algoritmo 5 indica a necessidade de ação de elasticidade para cada métrica de consumo.

Algoritmo 5: Abordagem proativa da solução de elasticidade automática

entrada: predição para a próxima janela, limite superior e inferior
saida **:** Indicação para ação de elasticidade para a métrica

```
1 if predicao de consumo < limiteInf then
2 |  ação ← −1
3 end if
4 else if predicao de consumo > limiteSup then
5 |  ação ← +1
6 end if
7 else
8 |  ação ← 0
9 end if
```

O processo seguido pelo Algoritmo 5 é bem comum das soluções de elasticidade automática que utilizam abordagem de antecipação, no qual a predição para o valor futuro da

métrica é comparada com o limite inferior (linha 1), e caso a predição seja menor do que o limite inferior, a ação indicada é uma ação de elasticidade para reduzir a quantidade de recursos (ação = −1, na linha 2). Caso a comparação da linha 1 não seja verdadeira, segue-se para comparar a predição para o valor futuro da métrica com o seu respectivo limite superior (linha 4). Assim, se a predição for maior que o limite superior, a ação indicada é uma ação de elasticidade para aumentar a quantidade de recursos (ação = +1, na linha 5). No caso de nenhuma comparação ser verdadeira, o algoritmo indica que nenhuma ação deve ser tomada (ação = 0, na linha 8). Esse processo é feito para cada métrica, e os quatro valores são encaminhados para o componente Decisor.

Como é possível observar, o processo seguido pelo Algoritmo 5 do componente Proativo é similar ao apresentado no Algoritmo 2 do componente Reativo, diferindo que no componente Proativo a comparação com os limites inferiores (linha 1) e superiores (linha 4) é feita com os valores preditos pelo modelo selecionado para cada métrica, enquanto no componente Reativo a comparação dos limites é feita com a medição atual. Dessa maneira, os resultados para as condições resultantes das comparações também são similares, isto é:

- Caso o valor predito seja menor do que o limite inferior, a ação indicada é uma ação de elasticidade para reduzir a quantidade de recursos (ação = −1, na linha 2);
- Caso a medição seja maior do que o limite superior, a ação indicada é uma ação de elasticidade para aumentar a quantidade de recursos (ação = +1, na linha 5);
- No caso de nenhuma comparação ser verdadeira, o algoritmo indica que nenhuma ação seja tomada (ação = 0, na linha 8).

O processo descrito no Algoritmo 5 é feito para cada métrica, e os quatro valores de *ação* são encaminhados para o componente Decisor, permitindo que a solução de elasticidade automática proposta se adapte às mudanças nos padrões de uso da aplicação.

Dessa maneira, a seleção de um modelo só é necessária porque há mais de um modelo sendo usado no componente Proativo. Em particular, considera-se o uso de múltiplos preditores, uma vez que não há um bom preditor para todos os serviços [108]. As características dos métodos de predição presentes no componente Proativo serão apresentadas a seguir.

Características dos Modelos Preditivos

De acordo com os resultados da avaliação relatados em [109], um único método não pode fornecer os resultados mais precisos para diferentes tipos de cargas de trabalho. A

direção de novos esforços de pesquisa tem sido sobre os métodos híbridos de previsão que mesclam a força de predição dos modelos de previsão individual [110].

A maioria dos trabalhos existente se concentra em um ou dois recursos (CPU e memória) e ignora a correlação entre os recursos. Investigar a correlação entre recursos pode fornecer resultados mais compreensíveis para a solução de elasticidade automática [111]. Em razão disso, foram escolhidas as duas redes neurais.

Os modelos de previsão podem ser usados para os diferentes tipos de cargas de trabalho e serem mais gerais, caso esses métodos de previsão não precisem fazer muitas suposições sobre o comportamento da carga de trabalho [111]. Os quatro modelos escolhidos atendem a essa característica [111].

O modelo ARIMA tenta capturar a correlação automática entre os dados [51], assim como o modelo Holt-Winters [58]. Entretanto, os modelos Holt-Winters e ARIMA são modelos lineares, o que em algumas situações pode não representar bem a demanda na nuvem. Para deixar o conjunto de modelos mais abrangente, também são usadas as redes neurais MLP e LSTM por poderem modelar bem o comportamento não linear da aplicação [111]. Por outro lado, as redes neurais são um método de "caixa preta", enquanto os primeiros possuem maior interpretabilidade ou seja, são mais transparentes [111].

4.2.5 Componente Decisor

O componente Decisor é elemento da solução de elasticidade automática proposta responsável por decidir se o sistema precisa aumentar ou diminuir o número de VMs, combinando um componente de elasticidade automática reativo e um componente de elasticidade automática proativo, apresentados nas seções anteriores.

O Algoritmo 6 apresenta a forma de interação entre o componente Decisor, e os componentes Reativo e Proativo. Primeiramente, as indicações para ações de elasticidade são encaminhadas ao Decisor na forma de listas, com cada lista contendo quatro valores, um valor entre $[-1, 0, 1]$ para cada métrica. A lista com as indicações para ações de elasticidade enviadas pelo componente Reativo é iterada (linha 1), e caso alguma métrica indique redução de recursos (linha 2), o contador de tempo para redução de recursos para essa métrica ($contadorInf_i$) é incrementado do intervalo de tempo de monitoramento (linha 3). Como a métrica indica um superprovisinamento, o contador de tempo para aumento de recursos para essa métrica $contadorSup_i$ volta ao valor zero (linha 4). No entanto, caso alguma métrica indique aumento de recursos (linha 6), o contador de tempo para aumento de recursos para essa métrica ($contadorSup_i$) é incrementado do intervalo de tempo de monitoramento (linha 7). Como a métrica indica um subprovisinamento, o contador de tempo para redução de recursos para essa métrica $contadorInf_i$ volta ao valor zero (linha 8). Depois disso, o maior valor dentre todas as métricas é aplicado ao contador de tempo

para aumento de recursos da solução (*contadorSup*) (linha 11) e ao contador de tempo para redução de recursos da solução (linha 12). Esses valores são avaliados pelo Decisor posteriormente junto a outros parâmetros da solução de elasticidade automática proposta (linha 13). Como é possível notar, o Decisor só levará em conta as informações repassadas pelo componente Proativo, após avaliar as indicações de ações de elasticidade enviadas pelo componente Reativo, similar a proposta de [112].

Algoritmo 6: Interação entre Decisor, e componentes Reativo e Proativo

entrada: contadores e indicações dos componentes Reativo e Proativo
saida : ação de elasticidade a ser avaliada pelo Decisor

```
1  for Lista Ações Reativas do
2      if Ação Reativa = −1 then
3          contadorInf_i+ intervalo
4          contadorSup_i = 0
5      end if
6      else if Ação Reativa = 1 then
7          contadorSup_i+ intervalo
8          contadorInf_i = 0
9      end if
10 end for
11 contadorSup ← máximo ( contadorSup_i)
12 contadorInf ← máximo ( contadorInf_i)
13 Decisor (contadorSup, contadorInf)
14 if componente Proativo já criou os modelos preditivos then
15     for Lista Ações Proativas do
16         if Ação Proativa = −1 then
17             contadorInf_i+ intervalo
18             contadorSup_i = 0
19         end if
20         else if Ação Proativa = 1 then
21             contadorSup_i+ intervalo
22             contadorInf_i = 0
23         end if
24     end for
25     contadorSup ← máximo ( contadorSup_i)
26     contadorInf ← máximo ( contadorInf_i)
27     Decisor (contadorSup, contadorInf)
28 end if
```

O componente Proativo só passará a interagir com o Decisor assim que os modelos preditivos tenham sido criados (linha 14), enquanto isso não ocorre, apenas o componente Reativo estará em operação. Esse é uma das vantagens do modelo híbrido adotado pela solução de elasticidade automática proposta, quando em comparação ao modelo pura-

mente proativo, dado que esse último não toma nenhuma ação de elasticidade enquanto o modelo de previsão não for criado.

Com os modelos preditivos já criados, a mesma sequência apresentada nas linhas (1-13) será seguida para a lista com as indicações para ações de elasticidade enviadas pelo componente Proativo, sendo essa lista iterada (linha 15), e caso alguma métrica indique redução de recursos (linha 16), o contador de tempo para redução de recursos para essa métrica ($contadorInf_i$) é incrementado do intervalo de tempo de monitoramento (linha 17). Como a métrica indica um superprovisinamento, o contador de tempo para aumento de recursos para essa métrica $contadorSup_i$ volta ao valor zero (linha 18). No entanto, caso alguma métrica indique aumento de recursos (linha 20), o contador de tempo para aumento de recursos para essa métrica ($contadorSup_i$) é incrementado do intervalo de tempo de monitoramento (linha 21). Como a métrica indica um subprovisinamento, o contador de tempo para redução de recursos para essa métrica $contadorInf_i$ volta ao valor zero (linha 22). Depois disso, o maior valor dentre todas as métricas é aplicado ao contador de tempo para aumento de recursos da solução ($contadorSup$) (linha 25) e ao contador de tempo para redução de recursos da solução (linha 26). Esses valores são avaliados pelo Decisor posteriormente junto a outros parâmetros da solução de elasticidade automática proposta (linha 27).

Contudo, se um dos dois componentes indicar a necessidade de ação de elasticidade, este componente Decisor concorda em fazer a reconfiguração do sistema (aumentando ou diminuindo o número de VMs) desde que sejam atendidos os planos de elasticidade automática estabelecidos, os quais serão apresentados a seguir.

Planos de Elasticidade Automática

Os planos de elasticidade automática são os planos de expansão e contração do *cluster* no qual a aplicação de interesse está em execução. Nestes planos há a indicação de variáveis importantes para todas as soluções de elasticidade automática, como contadores de tempo para aumento e redução de recursos, os quais possuem o papel de impedir que uma leve variação no ambiente exija uma ação de elasticidade, e no passo seguinte ocorra a indicação de uma ação de elasticidade contrária. Com papel similar está o contador de tempo de resfriamento, o qual indica o intervalo de tempo em que não haverá ação de elasticidade após a última ação de aumento ou redução do *cluster*.

Os planos de elasticidade automática também fazem referência a um contador para o número de VMs em execução no *cluster*, assim como o número mínimo e máximo de VMs no *cluster*.

O plano de expansão do *cluster*, apresentado pelo Algoritmo 7, trata das condições avaliadas pelo Decisor sobre a possibilidade de uma ação de elasticidade para aumento de

recursos, com base nos contadores enviados pelo componentes Reativo ou pelo Proativo (linha 1). Inicialmente avalia-se a duração de tempo no qual o valor medido ou a predição supera o limite superior (*durSup*, linha 2). Caso esse tempo seja maior, na sequência se avalia há quanto tempo houve a última ação de elasticidade (linha 3), sendo que se o tempo decorrido da última ação for maior do que o tempo de resfriamento configurado, avalia-se o tamanho do *cluster* (linha 4). Caso o *cluster* ainda suporte uma nova VM, a indicação do novo tamanho do *cluster* é feita ao Atuador do Provedor de nuvem (linhas 5-6).

Algoritmo 7: Plano de expansão do *cluster*

entrada: tempo de resfriamento, relógio, contadores, tamanho do cluster
saida : Número de VMs no Cluster da aplicação

```
1  Decisor (contadorSup, contadorInf):
2  if (contadorSup ≥ durSup) then
3  |  if (relógio ≥ tempo de resfriamento) then
4  |  |  if (TamanhoCluster < nMax) then
5  |  |  |  TamanhoCluster ←TamanhoCluster +1
6  |  |  |  Atuador (TamanhoCluster)
7  |  |  end if
8  |  |  relógio ← 0
9  |  end if
10 |  else
11 |  |  relógio ← relógio + intervalo
12 |  end if
13 end if
14 else
15 |  contadorSup ← contadorSup + intervalo
16 end if
```

Quando uma nova máquina é acrescentada ao *cluster*, o relógio que controla o tempo após a última ação de elasticidade é zerado (linha 8). Caso o tamanho do *cluster* não seja alterado, o relógio e o contador de tempo para aumento de recursos (*contadorSup*) são incrementados do intervalo de tempo de monitoramento (*intervalo*, linhas 11-15).

De forma similar funciona o Plano de contração do *cluster*, apresentado pelo Algoritmo 8, o qual trata das condições avaliadas pelo Decisor sobre a possibilidade de uma ação de elasticidade para redução de recursos, com base nos contadores enviados pelo componentes Reativo ou pelo Proativo (linha 1). Inicialmente avalia-se a duração de tempo no qual o valor medido ou a predição segue abaixo do limite inferior ($durInf$, linha 2). Caso esse tempo seja maior, na sequência avalia-se há quanto tempo houve a última ação de elasticidade, sendo que se o tempo decorrido da última ação for maior que o tempo de resfriamento configurado (linha 3), avalia-se o tamanho do *cluster* (linha 4). Caso o *cluster*

não esteja em seu tamanho mínimo (nMin), a indicação do novo tamanho do *cluster* é agendada ao Atuador do Provedor de nuvem (linhas 5-6). Esse agendamento posterga a contração do *cluster* até que uma das VMs do *cluster* esteja próxima de completar uma hora de uso.

Algoritmo 8: Plano de contração do *cluster*

entrada: tempo de resfriamento, relógio,contadores, tamanho do cluster
saida : Número de VMs no Cluster da aplicação

```
1  Decisor (contadorSup, contadorInf):
2  if (contadorInf ≥ durInf) then
3  |  if (relógio ≥ tempo de resfriamento) then
4  |  |  if (TamanhoCluster > nMin) then
5  |  |  |  TamanhoCluster ←TamanhoCluster −1
6  |  |  |  agenda Atuador (TamanhoCluster)
7  |  |  end if
8  |  |  relógio ← 0
9  |  end if
10 |  else
11 |  |  relógio ← relógio + intervalo
12 |  end if
13 end if
14 else
15 |  contadorInf ← contadorInf + intervalo
16 end if
```

Quando uma máquina é retirada do *cluster*, o relógio que controla o tempo após a última ação de elasticidade é zerado (linha 8). Caso o tamanho do *cluster* não seja alterado, o relógio e o contador de tempo para redução de recursos são incrementados do intervalo de tempo de monitoramento (linhas 12-16). Assim, o número mínimo e máximo de VMs no *cluster* deve ser indicado pelo gerente da solução em função da restrição de orçamento.

O componente Decisor será o responsável por interagir com o provedor para garantir que o tamanho do *cluster* indicado seja fornecido pelo provedor IaaS de nuvem. O Atuador é o elemento do provedor que interage com o componente Decisor para este fim, sendo essa interação será apresentado a seguir.

Interação entre Decisor e Atuador IaaS

Para aumentar ou diminuir o número de máquinas virtuais no *cluster* da aplicação, o Decisor precisa interagir com o provedor IaaS que hospeda na nuvem a referida aplicação. Provedores de computação em nuvem, como o Amazon EC2, fornecem acesso ao hardware,

que pode ser alocado ou desalocado a qualquer momento [50]. O Atuador é o responsável por realizar mudanças nos elementos gerenciados.

Para automatizar esse processo de interação entre cliente e provedor, realizado pelo Atuador, a Amazon EC2 fornece uma API a qual permite que imagens da aplicação cliente, criadas antecipadamente, sejam carregadas em novas máquinas virtuais que precisem ser criadas. Também é por meio dessa API que essas mesmas máquinas podem ser liberadas [50].

Assim sendo, neste trabalho, parte-se da suposição que o gerente da aplicação possua previamente:

- Imagem da aplicação disponível na AWS;
- Grupo de "*Auto-Scaling*".

Esse grupo de "*Auto-Scaling*" é o *cluster* de VMs no qual o cliente da nuvem estabelece o número mínimo e máximo de VMs no *cluster*.

A interação entre Decisor e Atuador é feita por meio do Boto3 [101], sendo que o Decisor ao solicitar ao Atuador o aumento ou a redução do tamanho do *cluster*, deve indicar a conta AWS, nome do grupo de "*Auto-Scaling*" e o novo tamanho do *cluster*. Quando há indicação para redução do número de máquinas no *cluster*, essa redução será postergada até que uma das VMs esteja próxima de completar uma hora, sendo que a VM a ser encerrada será a mais próxima de completar uma hora, em razão da cobrança do serviço ser por hora.

4.3 Considerações Finais

Em resumo, a solução de elasticidade automática híbrida proposta foi preparada para ser implantada no ambiente de produção de nuvem pública do TCU junto ao provedor Amazon AWS. A solução proposta pode ser executada de qualquer computador/VM que possua o sistema operacional baseado em Unix, como o MacOS em que parte dos testes foram executados, ou Ubuntu Linux em que outra parte dos testes foram executados. No entanto, deve haver a linguagem Python [113] instalada, caso contrário, esta solução não será capaz de prever o consumo de recursos. Vale ressaltar que foi escolhida a linguagem Python por esta linguagem de programação ser ensinada e difundida pelo TCU nos seus cursos de pós-graduação. Desse modo, espera-se que a manutenção e a evolução dessa solução no ambiente de produção do TCU seja facilitada.

Diante do exposto, é importante destacar que o componente Monitor utilizado nesta solução (que interage com o Cloudwatch) pode ser substituído por qualquer componente

equivalente, que busque os dados coletados em uma fonte diferente, como em outro provedor ou mesmo em ferramentas comerciais como o Zabbix [114], a qual é a ferramenta de monitoramento que se pretende implantar para coletar os dados nos provedores utilizados pelo TCU.

A solução de elasticidade automática híbrida proposta nesta dissertação é completamente independente do provedor de nuvem IaaS escolhido, com exceção do componente monitor como informado anteriormente. Isso significa que o tipo de balanceador de carga e o tipo de aplicação em execução nas máquinas virtuais não importam para esta solução proposta, desde que os requisitos anteriores sejam atendidos.

Capítulo 5

Experimentos

Este capítulo apresenta três experimentos, os quais foram realizados com o objetivo de avaliar a solução de elasticidade automática híbrida proposta. Nos experimentos verifica-se o comportamento das técnicas de elasticidade usadas na solução proposta (limites dinâmicos e análise de séries temporais), e como elas apoiam a solução de elasticidade automática proposta a reagir sob mudanças repentinas de carga de trabalho. Este capítulo também compara o desempenho da solução de elasticidade automática híbrida proposta com a versão emulada da solução de elasticidade automática da AWS. Assim, para avaliar a solução proposta, construiu-se um ambiente experimental na nuvem pública da AWS, o qual também será apresentado neste capítulo.

5.1 Ambiente Experimental

O ambiente experimental deste trabalho foi construído na nuvem pública da AWS, por ser essa uma das nuvens que o TCU utiliza no seu dia a dia. Como cenário representativo ao ambiente de nuvem do TCU, a aplicação MediaWiki [115] foi implantada no provedor de nuvem AWS. A MediaWiki é uma aplicação web do tipo *I/O bound*, ou seja, é intensiva em entrada/saída ao invés de CPU [16], assim como algumas aplicações de interesse do TCU que serão executadas na nuvem. Para executar a aplicação MediaWiki foi usado neste trabalho o *benchmark* WikiBench [12], o qual usa o MediaWiki como aplicação. O WikiBench usa dados reais do banco de dados da Wikipedia [116], e gera tráfego real com base nos traços de acesso da Wikipedia.

O ambiente criado na nuvem AWS para os experimentos deste trabalho foi montado seguindo experimentos anteriores apresentados por outros trabalhos [16] [80] [117] [118], os quais também utilizaram o Wikibench para avaliar as respectivas soluções de elasticidade automática. A Figura 5.1 apresenta o ambiente montado na nuvem AWS com o Wikibench para avaliar a solução de elasticidade automática proposta neste trabalho.

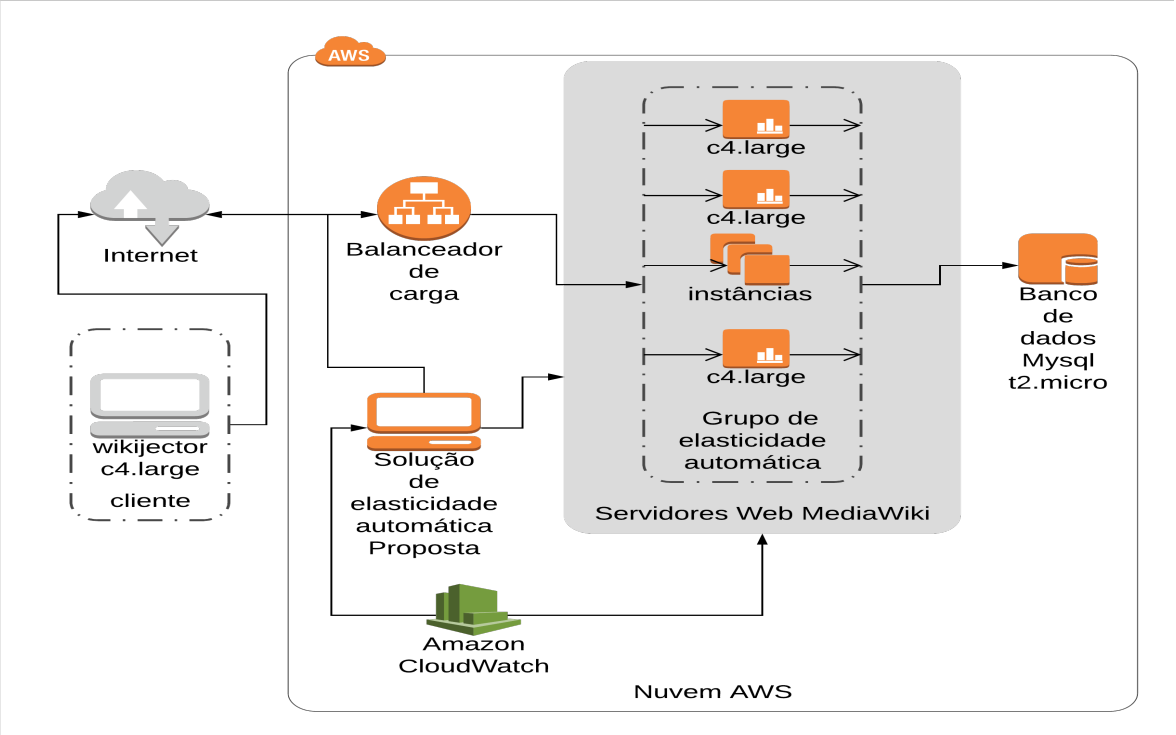

Figura 5.1: Ambiente Implantado para os Experimentos com WikiBench.

Como pode ser notado na Figura 5.1, a MediaWiki é uma aplicação de duas camadas (aplicação e banco de dados). Nos experimentos realizados a camada de banco de dados era composta por somente um banco de dados, enquanto a camada de aplicação era composta por um *cluster* de servidores web MediaWiki. Para esses experimentos, o foco foi na elasticidade da camada de aplicação e, em particular, em como o número de VMs do *cluster* hospedando servidores de aplicação MediaWiki era alterado sob demanda pelas soluções de elasticidade automática.

O banco de dados da MediaWiki, um banco MySQL, foi carregado com uma semana de rastreamento de 10% de solicitações da Wikipedia em inglês, de 19 de setembro de 2007 a 26 de setembro de 2007. Acredita-se que uma semana de rastreio é suficiente para a finalidade dos experimentos, pois oferece à solução proposta oportunidades para exercer os planos de elasticidade automática.

O Wikibench ainda é composto pelo wikijector, o qual é um agente usado para enviar solicitações dos arquivos de rastreamento da Wikipedia para a instância do balanceador de carga. É possível observá-lo na Figura 5.1 como sendo o cliente da aplicação MediaWiki.

O serviço de balanceamento de carga usado nos experimentos foi o balanceador de carga de aplicações [119], disponível na AWS, dado que a AWS o considera o mais adequado ao balanceamento de carga de tráfego HTTP e HTTPS. Neste trabalho assumiu-se que este balanceador de carga possui uma capacidade ilimitada e ele pode equilibrar ade-

quadamente a demanda dos usuários para cada VM que foi alocada para essa aplicação [17]. Esse balanceador de carga encaminha uma nova solicitação recebida para o servidor MediaWiki com menor número de solicitações no momento [119]. Mais detalhes das configurações usadas em cada experimento deste trabalho serão apresentados na Seção 5.2.

5.2 Descrição dos Experimentos

Esta seção destaca as configurações no ambiente que são comuns a todos os experimentos. Os experimentos conduzidos neste trabalho iniciam sempre da mesma forma, com o cliente wikijector enviando uma sequência de solicitações HTTP ao *cluster* de servidores MediaWiki. Essa sequência de solicitações segue a mesma ordem do rastreio de uma semana do Wikipedia populado no cliente. Certamente, se nenhuma modificação fosse realizada no arquivo original, o wikijector faria solicitações por uma semana. Logo, para atender as demandas deste trabalho, a ordem original das solicitações foi mantida, com isso a "forma" do rastreamento original é preservada, enquanto a dimensão do tempo é reduzida de 1 semana para o tempo indicado em cada experimento.

O cliente wikijector ao enviar uma solicitação ao conjunto de servidores da aplicação MediaWiki, aguarda a resposta dessa solicitação até o tempo limite de 30 segundos, sendo esse tempo definido para todos os experimentos.

O *cluster* de servidores de aplicação MediaWiki, conforme pode ser observado na Figura 5.1, fica por trás de um balanceador de carga, sendo ele o responsável por distribuir as solicitações enviadas pelo wikijector ao *cluster* MediaWiki. Esse *cluster* é um grupo de *autoscaling* [45] criado na AWS. Nos experimentos deste trabalho todas as máquinas virtuais da aplicação MediaWiki possuíam o sistema operacional Linux Ubuntu 16.04.

Os dados de consumo das VMs que executavam o MediaWiki são coletados e subsidiam a decisão sobre ações de elasticidade da solução de elasticidade automática proposta, e da solução de elasticidade automática da AWS. Quando o sistema é expandido, uma VM da camada de aplicação é criada a partir da solução de elasticidade automática e, quando o sistema é contraído, uma máquina virtual é destruída. Além disso, foi definido o número mínimo de VMs permitido e o número máximo de VMs permitido no grupo de *autoscaling* sendo 1 e 10 VMs, respectivamente. O número máximo foi definido com base no orçamento disponível para os testes. Essas VMs são alocadas sob demanda nos experimentos. Assim, em todos os experimentos, o número de VMs inicial é o número mínimo de instâncias permitido, o que atende a condição de pouca atividade da instância. Por fim, os servidores MediaWiki consultam o banco de dados que compõem o *benchmark*.

Nos 3 experimentos realizados, o cliente wikijector e os servidores MediaWiki estiveram em execução no tipo de instância c4.large. De acordo com [117], a instância c3.large é o tipo mais econômico para executar o aplicativo MediaWiki. No entanto, essa instância não estava mais disponível na AWS. Em função disso, a c4.large (2 VCPU e 3,75 GB RAM) foi escolhida por possuir especificações de recursos mais próximas que a citada instância. Enquanto isso, a instâncias EC2 do tipo t2.micro (1 VCPU e 1,0 GB RAM) foi usada para o banco de dados [16].

Para os experimentos, algumas condições do ambiente foram alteradas para se verificar o comportamento da solução proposta. A Tabela 5.1 apresenta um resumo das características de cada experimento. O primeiro parâmetro a ser definido no experimento foi a duração do Experimento, sendo esse parâmetro o que indica a compressão da janela do rastreio usado pelo wikijector. Esse tempo foi de 30, 60 e 120 minutos para os Experimentos 1, 2 e 3, respectivamente. A escolha da duração dos experimentos buscou apresentar o comportamento da solução de elasticidade automática proposta em um tempo menor, igual e maior do que o tempo de treinamento dos limites dinâmicos da solução (60 minutos).

O segundo parâmetro a ser definido foi o intervalo de coleta dos dados do *cluster* MediaWiki. Para os Experimentos 1 e 2, esse intervalo foi de 60 segundos; enquanto para o Experimento 3, esse intervalo foi de 120 segundos. O terceiro parâmetro foi a duração mínima de tempo no qual os limites devem ser superados ($durSup$, $durInf$) para se iniciar uma ação de elasticidade. Para os Experimentos 1 e 2 essas durações foram de 180 segundos; enquanto para o Experimento 3, esse intervalo foi de 360 segundos. O quarto parâmetro a ser configurado foi o intervalo mínimo entre as ações de elasticidade, conhecido como tempo para resfriamento ($resfSup$, $resfInf$). Para os Experimentos 1 e 2 esses tempos para resfriamento foram de 120 segundos; enquanto para o Experimento 3, esse tempo foi de 240 segundos.

Tabela 5.1: Parâmetros usados em cada Experimento.

Parâmetros	**Experimento 1**	**Experimento 2**	**Experimento 3**
Duração	30 min	60 min	120 min
Intervalo coleta	60 s	60 s	120 s
$durSup$, $durInf$	180 s	180 s	360 s
$resfSup$, $resfInf$	120 s	120 s	240 s

Cada um dos três experimentos foi repetido, no mínimo, por três vezes, conforme procedimento adotado por [16]. Na Seção 5.3 está relatado o melhor resultado obtido por cada solução de elasticidade automática.

5.3 Resultados dos Experimentos

O *benchmark* Wikibench permite avaliar desempenho de cada solução de elasticidade automática com relação a algumas variáveis que são registradas ao final de cada execução de um experimento. Os detalhes de cada uma das variáveis serão explicados a seguir.

O Wikibench, por meio do wikijector, registra todas as solicitações feitas ao *cluster* da aplicação, assim como o instante em que foi feita a solicitação, o tipo de solicitação HTTP enviada, o código HTTP retornado pelo servidor da aplicação, o tempo de resposta a essa solicitação, e o tamanho da resposta. Um extrato dos registros do Wikibench é apresentado na Tabela 5.2.

Tabela 5.2: Registros das Solicitações do Wikibench.

Registros da Solicitação	**Solicitação**
38 - GET - 194 - 200 - 7881	/MediaWiki/skins/monobook/headbg.jpg
13 - GET - 254 - 200 - 204	/MediaWiki/skins/common/images/magnify-clip.png
39 - GET - 206 - 404 - 324	/MediaWiki/Syria
31 - GET - 196 - 200 - 17466	/MediaWiki/skins/common/shared.css?99
10 - GET - 218 - 404 - 324	/MediaWiki/Dancing_with_the_stars
23000 - GET - 40 - 404 - 324	/MediaWiki/England_national_rugby_union_team
23024 - GET - 54 - 200 - 324	/MediaWiki/Main_Page

Para fins de ilustração, a última linha será usada para explicar o que cada campo da coluna "Registros da Solicitação" da Tabela 5.2 significa:

- 23024: registro de data e hora relativo da solicitação, em milissegundos, desde o início do rastreamento; ou seja, a primeira solicitação no arquivo de rastreamento teria um registro de data e hora relativo de 0;
- GET: tipo de solicitação http enviada, podendo ser GET ou POST;
- 54: tempo de resposta em milissegundos;
- 200: o código de *status* da resposta http [120];
- 324: tamanho da resposta http recebida em *bytes*.

Para processar esses registros do Wikibench foi criado um *script* em Python que escaneia os registros e retorna informações sobre o número de conexões realizadas, o tempo de resposta, e o tamanho das respostas. A seguir será apresentada a análise quanto ao número de conexões realizadas.

5.3.1 Número de Conexões Realizadas

As duas soluções de elasticidade automática foram avaliadas quanto ao número de conexões realizadas durante os três experimentos. Os dados que subsidiam essa análise foram coletados a partir dos registros do Wikibench. A Tabela 5.3 apresenta o número total de solicitações respondidas pelo MediaWiki ao wikijector, sendo que a coluna "AWS Emulado" apresenta o valor para cada experimento usando a versão emulada do AWS *Auto-Scaling*; a coluna "Solução Proposta" apresenta o valor para cada experimento usando a solução híbrida de elasticidade automática proposta; e a coluna "Impacto" apresenta o valor do impacto da solução proposta relativamente ao AWS *Auto-Scaling*, calculado conforme a Equação 5.1, na qual o valor negativo indica que houve uma piora relativa, e o valor positivo indica uma melhora relativa.

$$impacto = \left(\frac{solucaoProposta}{AWS} - 1 \right) * 100 \tag{5.1}$$

Na Tabela 5.3 é possível observar que nos três experimentos, a quantidade de conexões realizadas em cada experimento é muito próxima, usando a solução de elasticidade automática da AWS ou a solução de elasticidade automática proposta. Também é possível observar que a solução proposta apresenta melhora relativa em relação à solução da AWS a medida que a duração do experimento aumenta. Esse impacto negativo nos Experimentos 1 e 2 pode ser explicado pelo aprendizado dos limites dinâmicos, os quais estabilizam após uma hora, e também pela precisão reduzida decorrente da pequena quantidade de amostras para treinar os modelos de previsão.

Tabela 5.3: Número Total de Conexões durante os Experimentos.

Experimento	**AWS Emulado**	**Solução Proposta**	**Impacto**
Experimento 1	838.263	818.117	(↓) 2,4%
Experimento 2	1.629.697	1.649705	(↓) 0,66%
Experimento 3	3.288.033	3.304.350	(↑) 0,5%

5.3.2 Tempo de Resposta

As duas soluções de elasticidade automática foram avaliadas quanto ao tempo de resposta das solicitações durante os três experimentos. Os dados que subsidiam essa análise foram coletados a partir dos registros do Wikibench. A Tabela 5.4 apresenta o tempo médio de resposta das solicitações respondidas pelo MediaWiki ao wikijector, sendo que a coluna "AWS Emulado" apresenta o valor para cada experimento usando a versão emulada do AWS *Auto-Scaling*; a coluna "Solução Proposta" apresenta o valor para cada experimento usando a solução híbrida de elasticidade automática proposta; e

a coluna “Impacto” apresenta o valor do impacto da solução proposta relativamente ao AWS *Auto-Scaling*, calculado conforme a Equação 5.2, na qual o valor negativo indica que houve uma piora relativa e o valor positivo indica uma melhora relativa.

$$impacto = \left(1 - \frac{solucaoProposta}{AWS}\right) * 100 \tag{5.2}$$

A partir da Tabela 5.4 é possível notar que os tempos médios de resposta vão diminuindo a medida que é aumentada a duração do experimento, independente da solução de elasticidade automática usada. Isso indica que os maiores tempos de resposta concentram-se no início dos experimentos. É possível observar também na Tabela 5.4 que a solução proposta apresentou uma melhoria em comparação com a solução de elasticidade automática da AWS, a medida que a duração dos experimentos aumentava, fato que era esperado pois há um maior tempo de treinamento das técnicas de previsão usadas neste trabalho, além da estabilização do aprendizado dos limites dinâmicos.

Tabela 5.4: Tempo Médio de Resposta durante os Experimentos.

Experimento	**AWS Emulado**	**Solução Proposta**	**Impacto**
Experimento 1	471,53 ms	540,14 ms	(↓) 14,55%
Experimento 2	323,56 ms	243,04 ms	(↑) 24,88%
Experimento 3	207,23 ms	62,07 ms	(↑) 70,05%

O tempo de resposta da aplicação MediaWiki possui relação direta com o quantidade de VMs no grupo de elasticidade automática. A razão disso é que quanto maior for o número de VMs da aplicação MediaWiki disponíveis para atender as solicitações dos clientes, menores serão as filas para atendimento e, consequentemente, menores os tempos de respostas. As Figuras 5.2, 5.3 e 5.4 ilustram o número de servidores MediaWiki (VMs da aplicação no *cluster*) durante os Experimentos 1, 2 e 3, respectivamente.

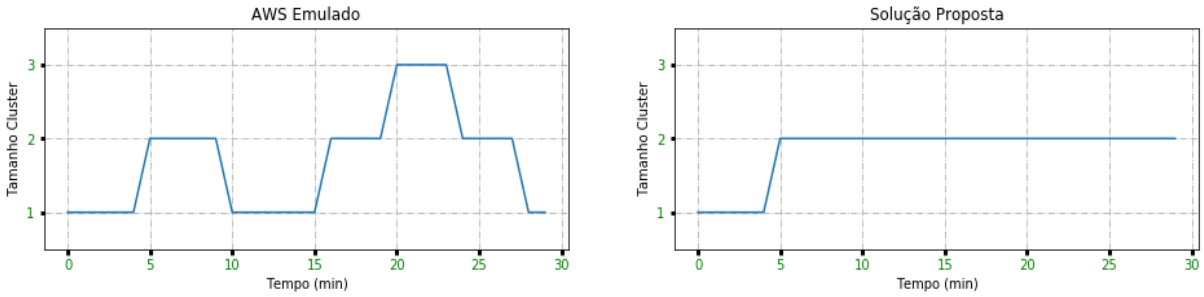

Figura 5.2: Número de Servidores MediaWiki durante o Experimento 1.

Essas figuras explicitam que, nos experimentos mais longos, a solução proposta conseguiu disponibilizar um maior número de VMs (tamanho do *cluster*) no *cluster* da aplica-

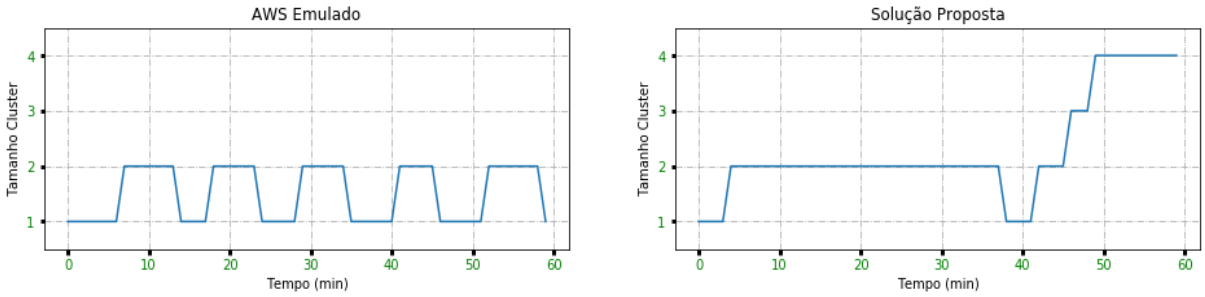

Figura 5.3: Número de Servidores MediaWiki durante o Experimento 2.

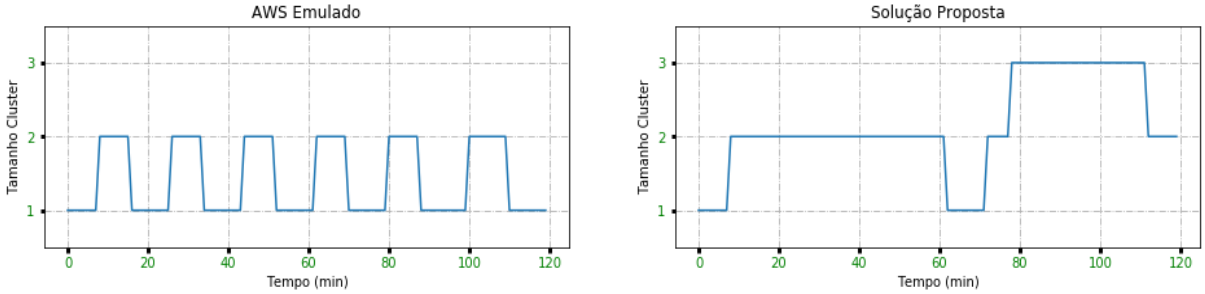

Figura 5.4: Número de Servidores MediaWiki durante o Experimento 3.

ção por um período de tempo maior do que a solução emulada da AWS, impactando na melhora do tempo médio de resposta.

5.3.3 Custo

As duas soluções de elasticidade automática foram avaliadas quanto ao custo durante os três experimentos. Os dados que subsidiam essa análise foram coletados a partir dos registros do Wikibench, e também da console do provedor de nuvem pública AWS. Os custos dos experimentos foram computados a partir de dois modelos de precificação, custo de cada VM por minuto ou por hora, apresentados nas Tabelas 5.5 e 5.6, respectivamente. Nas Tabelas 5.5 e 5.6, a coluna "AWS Emulado" apresenta o custo para cada experimento usando a versão emulada do AWS *Auto-Scaling*; a coluna "Solução Proposta" apresenta o custo para cada experimento usando a solução híbrida de elasticidade automática proposta; e a coluna "Impacto" apresenta o valor do impacto da solução proposta relativamente ao AWS *Auto-Scaling*, calculado conforme a Equação 5.3, na qual o valor negativo indica que houve uma piora relativa e o valor positivo indica uma melhora relativa.

$$impacto = \left(1 - \frac{solucaoProposta}{AWS}\right) * 100 \tag{5.3}$$

Para medir o custo total de um experimento, soma-se o custo de todas as VMs que estiveram disponíveis durante o experimento. Para medir o custo de uma única VM,

Tabela 5.5: Custo dos Experimentos para Cobrança por Minuto.

Experimento	AWS Emulado	Solução Proposta	Impacto
Experimento 1	51	55	(↓) 7,84%
Experimento 2	91	137	(↓) 50,55%
Experimento 3	170	256	(↓) 50,58%

Tabela 5.6: Custo dos Experimentos para Cobrança por Hora.

Experimento	AWS Emulado	Solução Proposta	Impacto
Experimento 1	240	120	(↑) 50,00%
Experimento 2	360	300	(↑) 16,67%
Experimento 3	420	300	(↑) 28,57%

supõe-se que o custo da VM é dado por uma taxa fixa (por minuto ou por hora, dependendo do modelo de precificação), e o custo da VM é calculado pelo tempo que a VM esteve disponível multiplicado pela taxa fixa. A taxa por minuto usada neste trabalho é de 1 unidade/min, enquanto a taxa por hora é dada por 60 unidades/h. Uma informação importante para o custo por hora é que a cobrança de 60 unidades é feita a partir do primeiro minuto da hora em que a VM estiver disponível. Portanto, se uma VM ficar ligada por 67 minutos, para o custo por hora, essa VM custará 120 unidades, enquanto para o custo por minuto, o custo ficaria em 67 unidades.

A partir da Tabela 5.5 é possível notar que ao se avaliar os custos dos experimentos em uma cobrança por minuto, a solução de elasticidade proposta teve maior custo do que a solução de elasticidade emulada da AWS em todos os experimentos. No entanto, ao se analisar a Tabela 5.6, que trata dos custos dos experimentos em uma cobrança por hora, observa-se uma inversão, tendo a solução de elasticidade proposta menor custo do que a solução de elasticidade emulada da AWS em todos os experimentos. As Figuras 5.5, 5.6 e 5.7 apresentam como os custos acumulados ao longo dos Experimentos 1, 2 e 3, respectivamente.

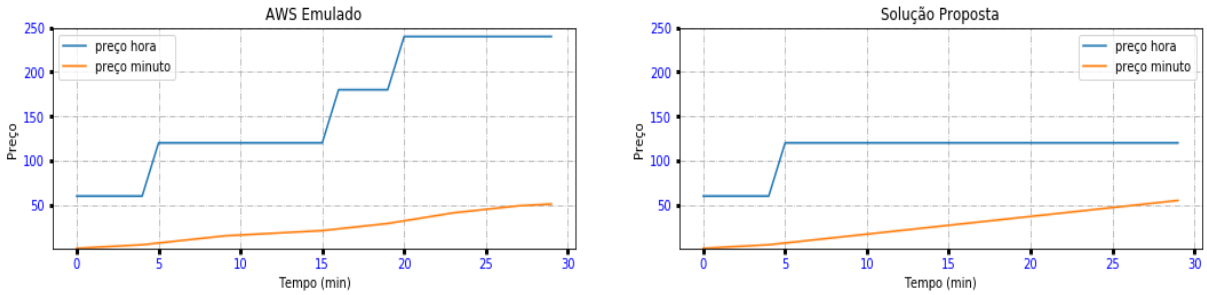

Figura 5.5: Custos do Experimento 1.

O custo dos experimentos, pelo provedor AWS, é feito por meio de uma taxa fixa por hora. Em razão disso, a solução de elasticidade híbrida proposta é mais benéfica em

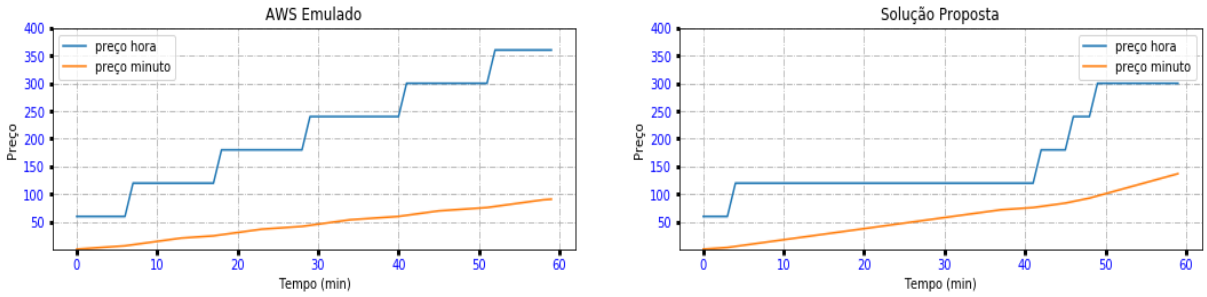

Figura 5.6: Custos do Experimento 2.

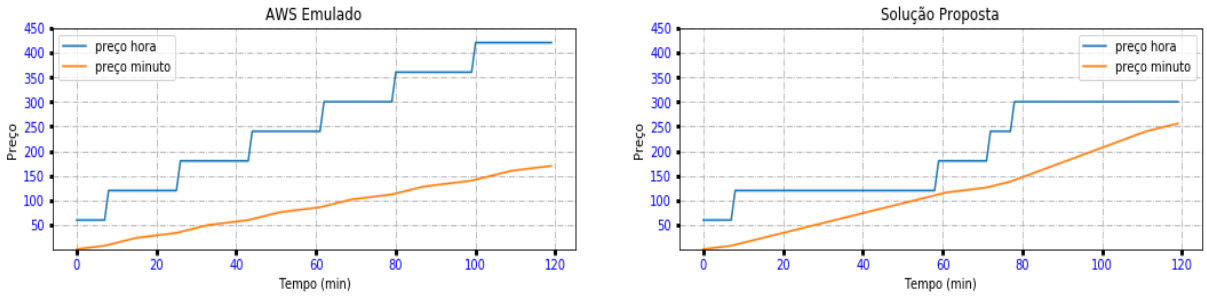

Figura 5.7: Custos do Experimento 3.

termos de custos para o cliente da nuvem do que a solução de elasticidade oferecida pela própria AWS. É importante ressaltar que caso a política de cobrança da AWS passe a ser por minuto, há a necessidade de se adaptar a solução proposta para essa política.

5.3.4 Limites Fixos x Limites Dinâmicos

Nesta seção serão analisados o comportamento das quatro métricas coletadas durante os experimentos, e também como são usados os limites dinâmicos neste trabalho.

As Figuras 5.8 e 5.9 apresentam o consumo das métricas para o Experimento 1, com o uso da solução de elasticidade automática da AWS e com o uso da solução de elasticidade automática híbrida proposta, respectivamente. De modo similar, as Figuras 5.10 e 5.11 apresentam o consumo das métricas para o Experimento 2, na solução de elasticidade automática da AWS, e na solução de elasticidade automática híbrida proposta, respectivamente. Por fim, as Figuras 5.12 e 5.13 apresentam o consumo das métricas para o Experimento 3.

As Figuras 5.8, 5.9, 5.10, 5.11, 5.12 e 5.13 apresentam, além dos consumos das métricas do Wikibench, os limites superiores e inferiores para a solução de elasticidade automática da AWS, e para a solução de elasticidade automática híbrida proposta.

Ao se analisar as figuras citadas, pode-se observar que as métricas coletadas para o Wikibench, normalmente, não alcançam o limite superior da solução de elasticidade

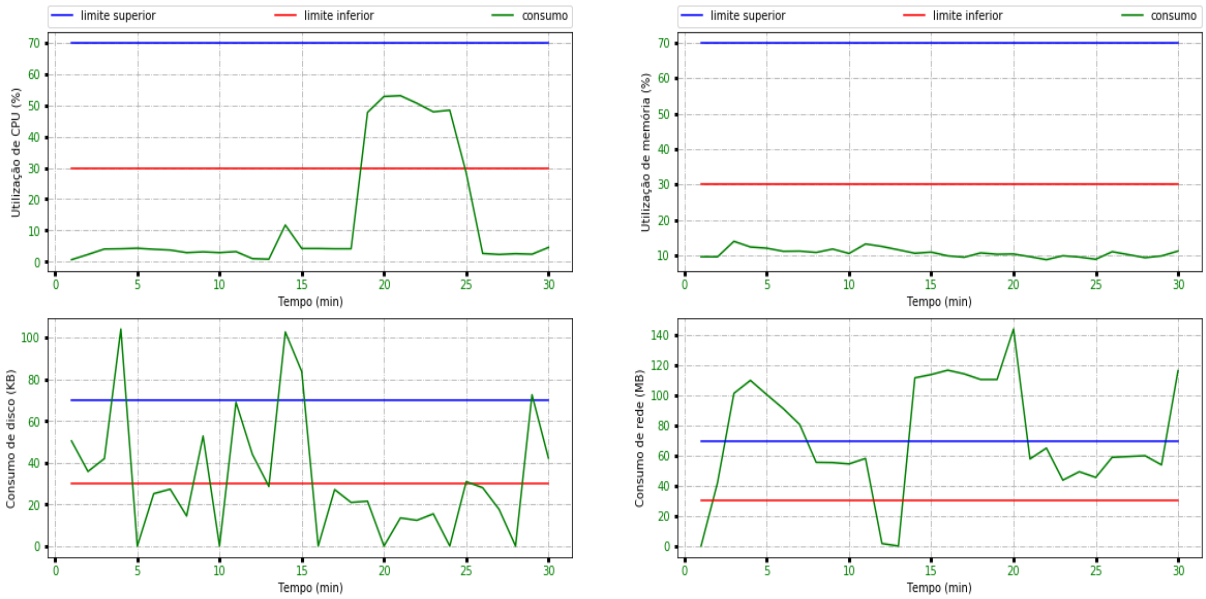

Figura 5.8: Consumos e Limites durante o Experimento 1 - AWS.

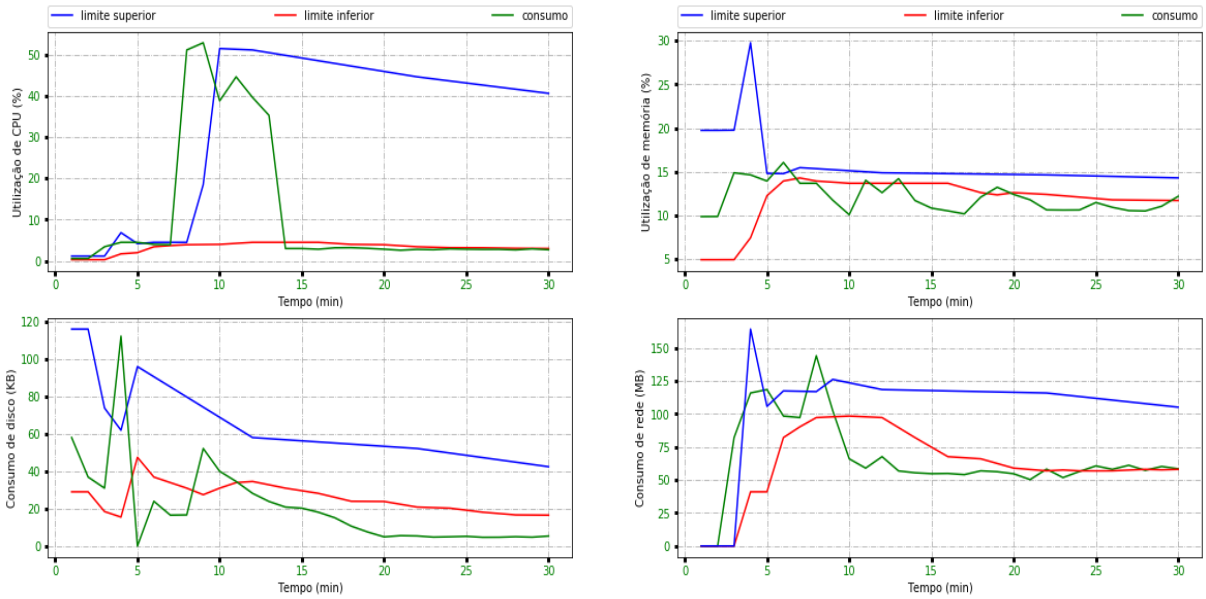

Figura 5.9: Consumos e Limites durante o Experimento 1 - Solução Proposta.

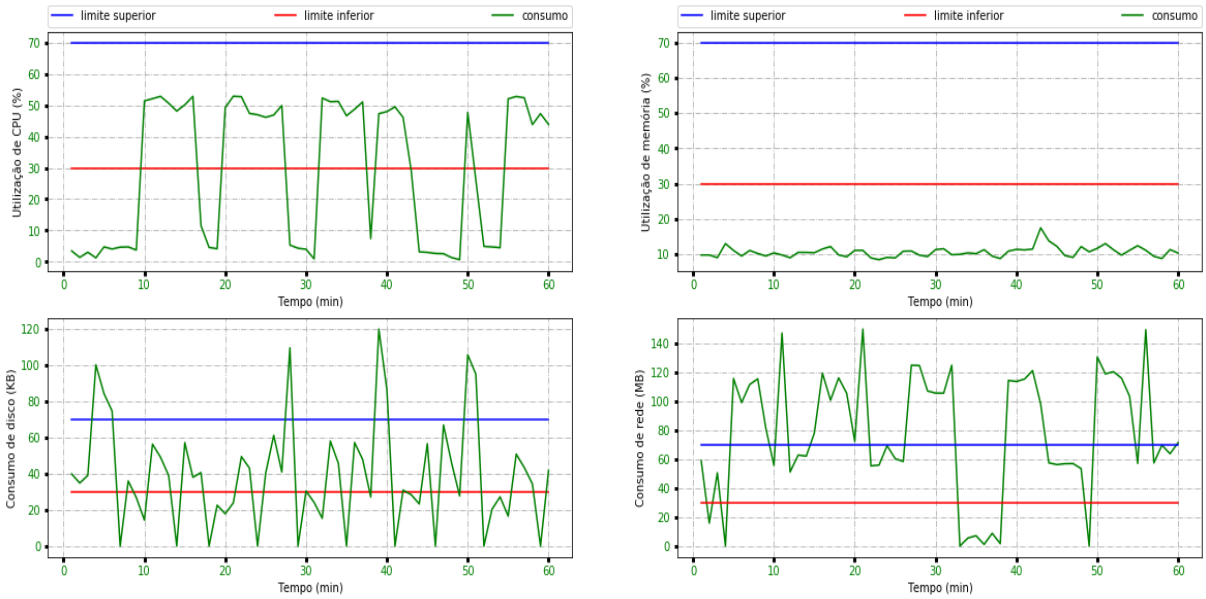

Figura 5.10: Consumos e Limites durante o Experimento 2 - AWS.

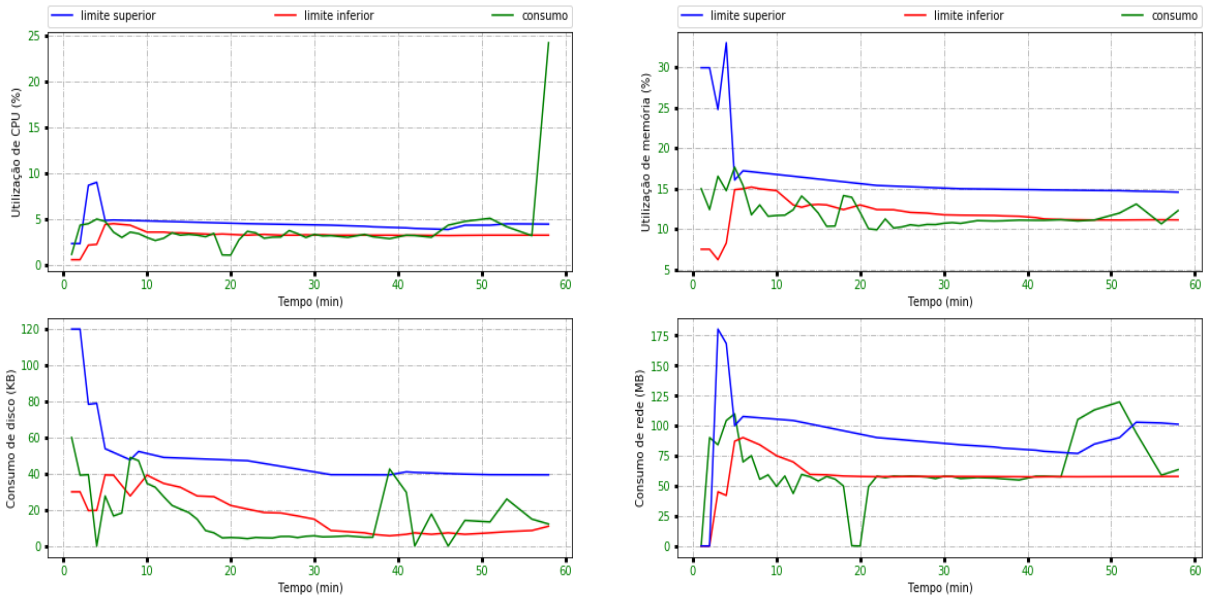

Figura 5.11: Consumos e Limites durante o Experimento 2 - Solução Proposta.

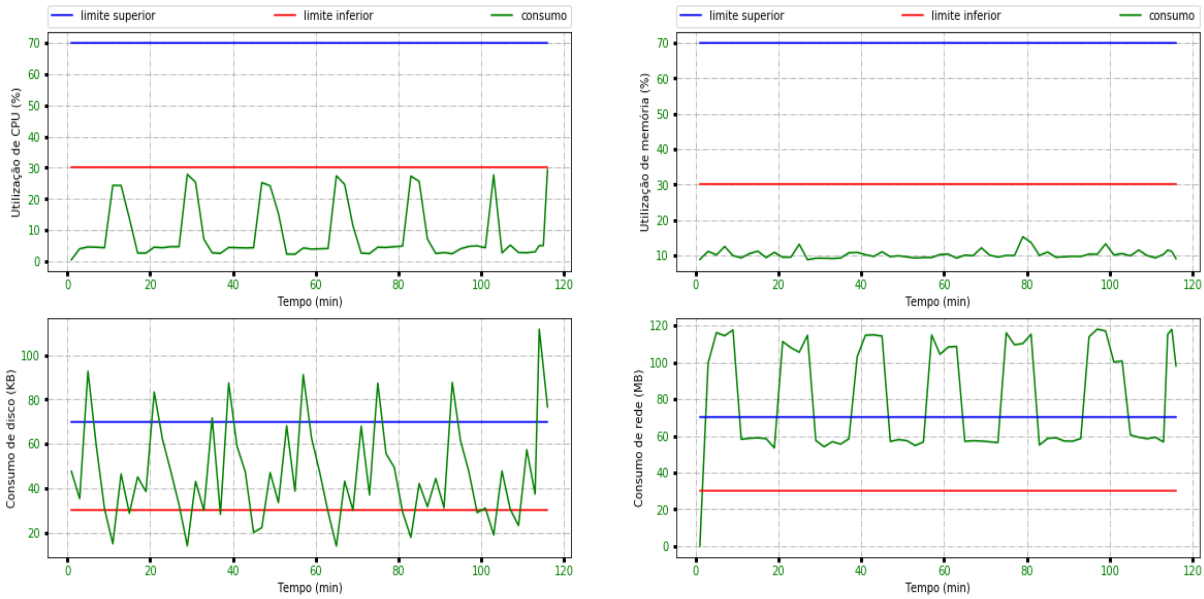

Figura 5.12: Consumos e Limites durante o Experimento 3 - AWS.

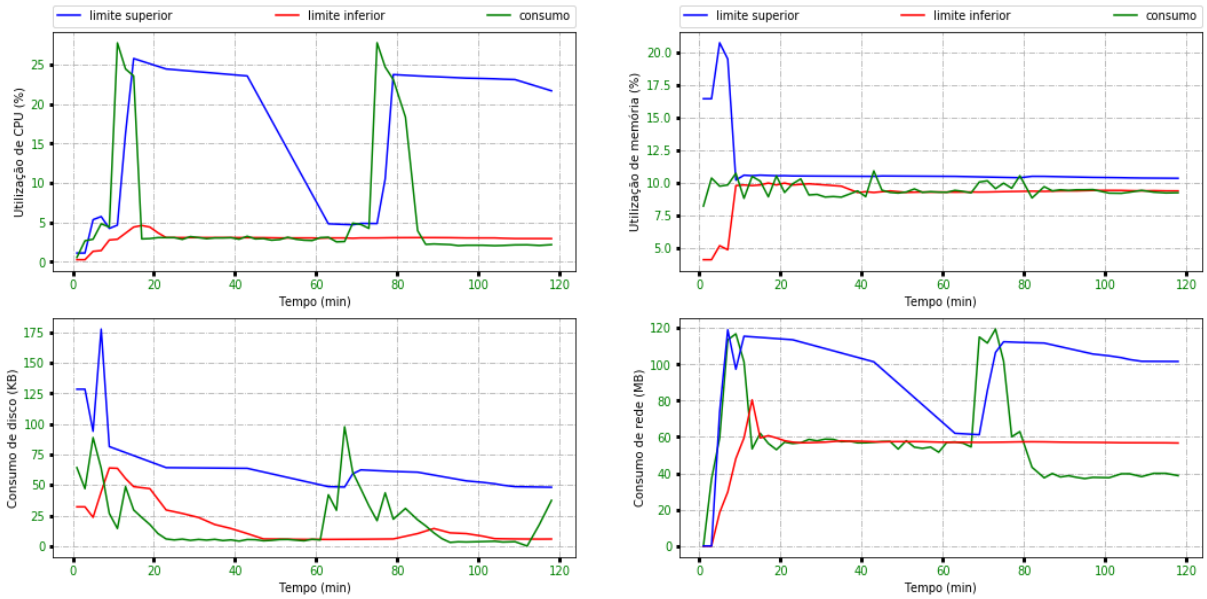

Figura 5.13: Consumos e Limites durante o Experimento 3 - Solução Proposta.

automática da AWS (70% de utilização de CPU e memória [11]), além de não passar muito tempo na faixa entre os limites inferior e superior para os consumos de disco e rede. Isso indica a dificuldade em se definir os valores desses limites, impactando no desempenho da aplicação. Por outro lado, a solução de elasticidade proposta consegue se adaptar para manter, durante grande parte do tempo dos experimentos, o consumo das métricas da aplicação na faixa entre os limites inferior e superior, sendo essa faixa a mais próxima do estado entre subprovisionamento e superprovisionamento.

As Figuras 5.14, 5.15 e 5.16 ilustram as ações de elasticidade durante os Experimentos 1, 2 e 3, respectivamente. Nas figuras citadas, o valor 1 indica que o *cluster* da aplicação foi expandido (acrescentada uma VM ao *cluster*), o valor -1 indica que o *cluster* da aplicação foi contraído (retirada uma VM do *cluster*), e o valor 0 indica que não houve ação de elasticidade.

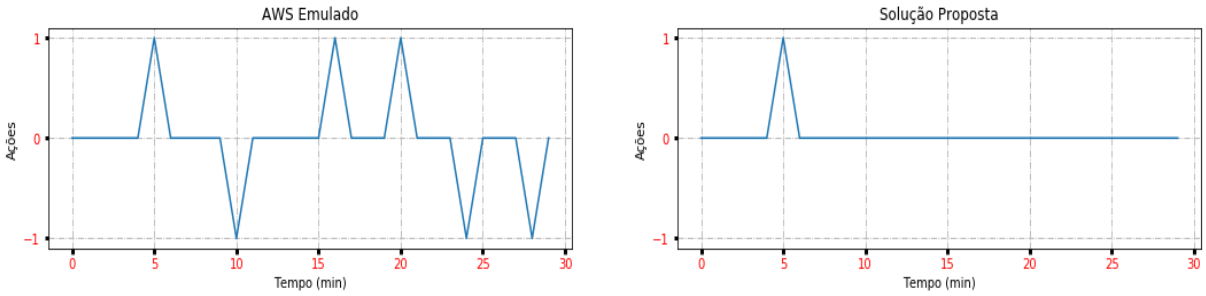

Figura 5.14: Ações de Elasticidade durante o Experimento 1.

Ao se analisar as citadas figuras, pode-se observar que a solução de elasticidade emulada da AWS entra em oscilação, expandido o tamanho do *cluster*, principalmente, por demanda de rede, e instantes depois, por indicação, principalmente, da utilização de memória, o *cluster* é contraído, enquanto na solução de elasticidade automática híbrida proposta, um aumento repentino no consumo de rede e de CPU é acompanhado da mudança do limite superior, resultando em menor quantidade de ações de elasticidade e sofrendo menos com o problema de oscilação.

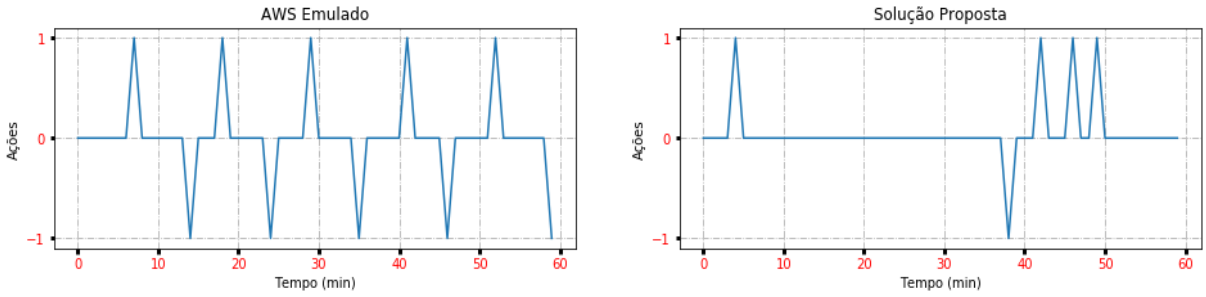

Figura 5.15: Ações de Elasticidade durante o Experimento 2.

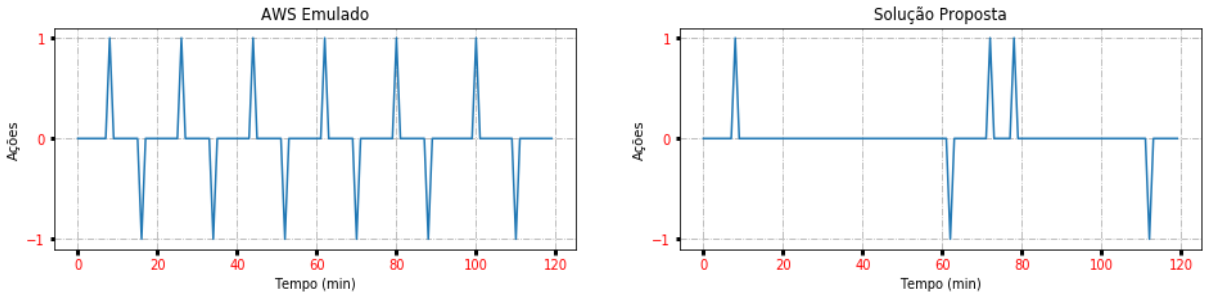

Figura 5.16: Ações de Elasticidade durante o Experimento 3.

A análise da precisão do método de predição que compõe a solução de elasticidade automática híbrida proposta é apresentada a seguir.

5.3.5 Precisão das Predições

A precisão das predições dos resultados experimentais pode ser avaliada com base em diferentes métricas, como Erro Absoluto Médio (*Mean Absolute Error* - MAE) e Erro Quadrático Médio Raiz (*Root Mean Square Error* - RMSE) [121], as quais são usadas para avaliar os resultados deste trabalho. A definição formal da RMSE é apresentada na Equação 5.4, na qual $\hat{x}_i$ é o valor previsto, x_i é o valor medido para a *i-ésima* observação e n é o número de observações para as quais a previsão é feita. A definição formal da MAE foi apresentada na Equação 4.3.

$$\text{RMSE} = \frac{1}{n}\left(\sqrt{\sum_{i=0}^{n-1}(\hat{x}_i - x_i)^2}\right) \tag{5.4}$$

A métrica do MAE é uma métrica popular na estatística, especialmente, na avaliação da precisão da previsão. A métrica RMSE representa o desvio padrão da amostra das diferenças entre os valores previstos e os valores reais. Os valores menores de MAE e RMSE indicam um modelo de previsão mais preciso.

A métrica do MAE é uma pontuação linear que assume que todos os erros individuais são ponderados igualmente. Além disso, o RMSE é mais útil quando os erros grandes são particularmente indesejáveis. No domínio de elasticidade automática, um modelo de regressão que gera um número maior de pequenos erros é mais desejável do que um modelo de regressão que gera um número menor de erros grandes. O motivo é que os tomadores de decisão, baseados em regras, emitem as ações de elasticidade com base nos valores de previsão e, para gerar uma ação de elasticidade correta, a previsão deve estar próxima o suficiente do valor real. Em outras palavras, os tomadores de decisão baseados em regras não são sensíveis aos pequenos erros nos resultados da previsão. Portanto, os erros menores nos resultados da previsão são insignificantes. Como resultado, no domínio

de elasticidade automática da nuvem, o fator RMSE é mais importante do que o fator MAE. No entanto, considerando que as duas métricas (ou seja, MAE e RMSE) fornecem uma análise abrangente da precisão dos modelos de previsão, quanto maior a diferença entre RMSE e MAE, maior a variação nos erros individuais da amostra [121].

Uma outra métrica de erro bastante comum na literatura é o Erro Médio Percentual Absoluto (*Mean Absolute Percentage Error* - MAPE), definido pela Equação 5.5, na qual $\hat{x}_i$ é o valor previsto, x_i é o valor medido para a *i-ésima* observação, e n é o número de observações para as quais a previsão é feita.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|\hat{x}_i - x_i|}{x_i}\right) * 100 \tag{5.5}$$

Os erros de porcentagem como o MAPE têm a vantagem de serem independentes da escala e, portanto, são frequentemente usados para comparar o desempenho da previsão em diferentes conjuntos de dados. No entanto, o MAPE tem a desvantagem de se tornar infinito ou indefinido se $x_t = 0$ para qualquer t no período de interesse, além de ter uma distribuição extremamente distorcida quando qualquer x_t estiver próximo de zero [122].

Outra métrica para avaliar o erro é a Raiz do Erro Médio Quadrático Escalonado (*Root Mean Square Scaled Error* - RMSSE), que é calculada conforme a Equação 5.6, na qual $\hat{x}_i$ é o valor previsto, x_i é o valor medido para a *i-ésima* observação, e n é o número de observações para as quais a previsão é feita [123].

$$\text{RMSSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}\left(\frac{|\hat{x}_i - x_i|}{\frac{1}{(n-1)}\sum_{t=2}^{n}|x_t - x_{t-1}|}\right)^2} \tag{5.6}$$

A métrica RMSSE é adequada para quase todas as situações e é uma métrica que compara a RMSE do método avaliado com o RMSE da previsão de um *benchmark*, normalmente com esse *benchmark* sendo o método da caminhada aleatória. A previsão da caminhada aleatória prevê que para todo o horizonte de valores futuros, o valor será igual ao último valor do histórico. Ao se escalar o erro com base no RMSSE da amostra, a partir do método de previsão ingênuo (caminhada aleatória) obtêm-se um erro independente da escala dos dados [122]. Quando o RMSSE é menor do que 1 (RMSSE < 1), o método proposto apresenta, em média, erros menores do que os erros de previsão de um passo a frente do método caminhada aleatória. Por outro lado, se o RMSSE for maior do que 1 (RMSSE > 1), o modelo de previsão foi pior que a previsão um passo a frente do método caminhada aleatória [124].

A precisão dos resultados do Experimento 3 (o que apresentou os melhores resultados para a solução proposta), medida pelas métricas MAE, RMSE, MAPE e RMSSE são apresentadas na Tabela 5.7.

Tabela 5.7: Precisão das Predições para o Experimento 3.

Métrica de erro	**CPU**	**Memória**	**Disco**	**Rede**
MAE	1,926	0,296	2,984	4,998
RMSE	14,139	0,203	71,306	137,736
RMSSE	0,865	0,563	0,554	0,776
MAPE	17,541 %	3,089 %	6,732 %	10,173 %

De acordo com a Tabela 5.7, não há diferença significativa entre os valores do MAE e do RMSE para o consumo de memória, indicando que os valores de utilização de memória não variaram muito ao longo do experimento. A maior diferença entre MAE e do RMSE foi para o consumo de rede, o que indica a existência de muitos valores extremos.

Os altos valores de RMSE para os consumos de disco e de rede indicam uma dificuldade do mecanismo de previsão da solução proposta de prever um grande salto absoluto no valor futuro da métrica (utilização de CPU e de memória variam no intervalo entre $[0, 1]$). No entanto, os baixos valores relativo de MAE para os consumos de disco e de rede indicam que a solução se adéqua rapidamente ao novo cenário.

Os valores de MAPE encontram-se dentro do esperado, exceto o valor encontrado para o consumo de CPU. Uma possibilidade para o valor encontrado (17,54 %) pode ter relação com os baixos valores assumidos por essa variável na maior parte do experimento. Todavia, quando ocorre um aumento significativo na utilização de CPU, essa métrica de erro penaliza bastante o erro de previsão. O consumo de rede era o mais significativo no *benchmark* Wikibench, logo esperava-se que o maior valor de MAPE fosse para o consumo de rede.

Quanto aos valores encontrados para RMSSE, todos indicam que o mecanismo de previsão proposto apresenta, em média, erros menores do que os erros de previsão de um passo a frente do método caminhada aleatória, dado que todos os valores estão abaixo de 1, indicando que o método de previsão que compõe a solução de elasticidade automática atende adequadamente ao seu papel.

A solução de elasticidade automática híbrida proposta foi avaliada em relação a solução de elasticidade automática da AWS. Os resultados da avaliação, para as condições descritas, mostram que a solução de elasticidade automática híbrida proposta reduz o tempo de resposta médio do Wikibench percebido pelos clientes da nuvem em até 70% comparado com o modelo reativo padrão da AWS.

5.4 Considerações Finais

Todos os experimentos e resultados apresentados nesta tese devem ser investigados no escopo mencionado, e alterar o escopo de qualquer forma (como considerar diferentes

tipos de VMs, considerar outras camadas de serviço em nuvem, considerar outras métricas, etc.) podem alterar os resultados. Além disso, o conjunto de previsões proposto nesta tese encontra o modelo de previsão mais adequado entre os modelos ARIMA, Holt-Winters, MLP e LSTM, com base no padrão de carga de trabalho recebido. A precisão do modelo de previsão proposto é medida pelas métricas MAE, RMSE, MAPE e RMSSE. Assim, alterar as métricas de avaliação pode alterar os resultados experimentais relativos e alterar o mecanismo do tomador de decisão, como alterar seu conjunto de parâmetros ou sua lógica de elasticidade, pode alterar os resultados das ações de elasticidade.

Capítulo 6

Conclusões

O interesse da pesquisa neste tema de elasticidade automática para computação em nuvem surgiu da possibilidade de migração massiva de serviços do Tribunal de Contas da União para ambientes de nuvem pública.

A característica de elasticidade e o modelo de precificação de pagamento conforme o uso da computação em nuvem permitem que os clientes na nuvem reduzam seus custos pagando apenas pelos recursos que realmente usam. Os clientes de nuvem oferecem um serviço, por meio do provedor de nuvem, aos usuários finais e são obrigados a manter um certo nível de QoS, tarefa apoiada por soluções de elasticidade automática. Essas soluções ajustam automaticamente a quantidade de recursos de infraestrutura provisionado com base na carga de trabalho recebida do serviço em nuvem.

As soluções de elasticidade automática baseadas em regras com limites fixos, devido à sua natureza simples e fácil de entender, são populares, no entanto, essas soluções sofrem de duas deficiências principais: a natureza puramente reativa e a dificuldade de selecionar o conjunto correto dos parâmetros de configuração.

Esta tese propõe uma solução híbrida de elasticidade automática que supera algumas das deficiências dos sistemas baseados em regras com limites fixos. A solução de elasticidade automática proposta consiste em um conjunto de previsões e limites dinâmicos auto-adaptáveis.

O conjunto de previsões resolve a primeira falha dos sistemas baseados em regras (ou seja, a natureza reativa) prevendo a carga de trabalho futura do serviço em nuvem. Enquanto isso, os limites dinâmicos, diminuem a dificuldade de selecionar o conjunto correto dos parâmetros de configuração em modelos reativos.

A solução de elasticidade automática híbrida proposto reduz o tempo de resposta médio para o *benchmark* Wikibench em até 70% em comparação com a solução de elasticidade automática da AWS. Com a avaliação realizada, usando cargas de trabalho geradas pelo *benchmark* Wikibench, a viabilidade da solução proposta foi confirmada.

6.1 Contribuições

A proposta deste trabalho foi a provisão da elasticidade automática feita de acordo com a análise de várias métricas usualmente coletadas pelas ferramentas do TCU, com o propósito de melhorar métricas que afetam diretamente a percepção de um usuário, como o tempo de resposta da aplicação em nuvem, como ocorreu nos experimentos realizados. Desta forma, as contribuições diretas desta dissertação são:

- Proposição de uma solução de elasticidade automática independente de aplicação em execução;
- Proposição de uma solução de elasticidade automática que seja configurada de forma dinâmica e automática;
- Uma solução de elasticidade automática que propõem os limites para tomada de decisão sobre ações de elasticidade, abstraindo uma parte da complexidade de configuração da solução de elasticidade automática pela equipe do TCU;
- Quanto às tarefas de previsão, os modelos para previsão são criados durante a execução das tarefas da aplicação (*on-line*), reduzindo assim o tempo de aprendizado da solução, quando comparado com as soluções que treinam os modelos com dados de execuções anteriores da aplicação;
- Uma solução de elasticidade automática proposta só precisa monitorar a utilização dos recursos de infraestrutura das máquinas virtuais nas quais os serviços são executados, sem nenhuma consideração adicional ou restritiva;
- Aceite do artigo "*A Hybrid Automatic Elasticity Solution for the IaaS Layer based on Dynamic Thresholds and Time Series*" para a 15ª Conferência Ibérica de Sistemas e Tecnologias de Informação (CISTI'2020).

6.2 Trabalhos Futuros

A solução de elasticidade automática proposta faz uso da abordagem preditiva para gerar as ações de elasticidade antecipadamente, a fim de evitar as condições de subprovisionamento e super-provisionamento. Outra abordagem para evitar a condição de subprovisionamento é reduzir o tempo de provisionamento de recursos, substituindo as VMs por *containers*. Outra evolução interessante seria a adaptação desta proposta para elasticidade automática com VMs e *containers*, o que tornaria esta proposta um mecanismo genérico para provisão de elasticidade automática. Embora a redução do tempo de

provisionamento de recurso não possa impedir completamente a condição de subprovisionamento, o uso dos contêineres em vez das VMs pode aumentar o tempo de atendimento da QoS.

Além disso, a solução proposta assume que os estados de subprovisionamento e superprovisionamento possuem custos distintos, mas não há uma quantificação desse custo. Dessa maneira, em trabalhos futuros, a solução proposta pode ser aprimorada para calcular o custo desses estados com mais precisão, com base na QoS. No entanto, a avaliação do custo de não atendimento da QoS depende de diferentes fatores, como a duração do tempo de inatividade do serviço em nuvem, o número de usuários finais afetados e até os aspectos sociológicos do comportamento dos usuários finais. Portanto, para medir o custo de não atendimento da QoS, diferentes classes de usuários finais, bem como as características de cada classe, devem ser estudadas.

A sequência deste trabalho também pode ir no caminho da integração de novas métricas a serem consideradas pela solução de elasticidade automática, como requisições por segundo ou tempo de resposta. Um outro possível trabalho futuro pode ser a integração de um planejador de recursos a esta solução de elasticidade automática, pois esse planejador agregaria uma facilidade ao gerente de recursos da nuvem, dado que o pouparia de ter que indicar, previamente a execução da aplicação, a quantidade de recursos a ser adicionado ou removido.

Este trabalho utilizou um conjunto de técnicas de predição, no entanto, futuramente novas técnicas poderiam ser avaliadas. Por exemplo, poderiam ser exploradas somente técnicas de aprendizado de máquina, desde que o intervalo de monitoramento fosse aumentado. A implantação desta proposta em um ambiente de produção será importante para a verificação de possíveis questões que não ficaram evidentes durante os testes realizados com *benchmark*, ainda que as cargas de trabalho geradas por ele sejam criadas a partir de cargas reais.

A avaliação deste trabalho foi feita em um provedor de nuvem pública, portanto, seria interessante a realização de testes com esta proposta em um outro provedor de nuvem pública ou em uma nuvem híbrida. Neste caso, poderia ser feito um estudo que verificasse a economia financeira em termos quantitativos, que se pode ter utilizando esta proposta nos variados ambientes.

Referências

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for delivering computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009. 1, 7

[2] Ministerio do Planejamento, "Boas práticas, orientações e vedações para contratação de Serviços de Computação em Nuvem — Governo Digital." 1

[3] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013)*, p. 23, 2013. 1, 10

[4] AWS, "https://aws.amazon.com/pt/compute/sla/," *CoRR*, 2018. 1

[5] Azure, "Produtos, serviços e soluções de nuvem do Microsoft Azure | Microsoft Azure." 1

[6] "IBM Cloud | IBM." 1, 15

[7] "Google Cloud Platform | Google for Education." 1

[8] B. Gâteau, "A multi-agent system to monitor SLA for cloud computing," in *ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems*, vol. 2, pp. 647–652, 2014. 1

[9] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," in *Algorithms and Architectures for Parallel Processing* (C.-H. Hsu, L. T. Yang, J. H. Park, and S.-S. Yeo, eds.), (Berlin, Heidelberg), pp. 13–31, Springer Berlin Heidelberg, 2010. 2

[10] V. R. Messias, J. C. Estrella, R. Ehlers, M. J. Santana, R. C. Santana, and S. Reiff-Marganiec, "Combining time series prediction models using genetic algorithm to autoscaling Web applications hosted in the cloud infrastructure," *Neural Computing and Applications*, vol. 27, pp. 2383–2406, 11 2016. 2

[11] P. R. Pereira Da Silva, *A Hybrid Strategy for Auto-scaling of VMs: An Approach Based on Time Series and Thresholds*. PhD thesis, Universidade Federal de Pernambuco, 2019. 2, 48, 78

[12] E.-J. van Baaren, *WikiBench: A distributed, Wikipedia based web application benchmark*. PhD thesis, 2009. 2, 65

[13] Primesys, "Primesys." 3

[14] G. Venkat, R. Hayat, and T. M, "A Survey on the needs and issues of cloud broker in cloud environment," *International Journal of Development Research.*, pp. 1035–1040, 2014. 3

[15] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments," *Journal of Grid Computing*, vol. 12, pp. 559–592, 12 2014. 3, 14, 17, 18, 19, 46, 49

[16] B. Liu, *A Fuzzy logic based Auto-scaler for Web Applications in Cloud Computing Environments*. PhD thesis, UNIVERSITY OF MELBOURNE, 6 2018. 3, 51, 65, 68

[17] S. Y. Nikravesh, *A Self-Adaptive Auto-Scaling System in Infrastructure-as-a-Service Layer of Cloud Computing*. PhD thesis, 2016. 3, 4, 45, 49, 67

[18] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science - MGC '10*, (New York, New York, USA), pp. 1–6, ACM Press, 2010. 3, 33, 34, 41, 53

[19] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," in *Future Generation Computer Systems*, vol. 28, pp. 155–162, 1 2012. 4, 50

[20] S. Dubey and S. Agrawal, ""methods to ensure quality of service in cloud computing environment"," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, pp. 405–411, 2013. 6

[21] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *2008 Grid Computing Environments Workshop*, pp. 1–10, Nov 2008. 6, 7

[22] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, and A. Rabkin, "Above the Clouds: A Berkeley View of Cloud Computing," tech. rep., 2009. 6

[23] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 50–55, Dec. 2008. 7

[24] P. Mell and T. Grance, "The NIST Definition of Cloud Computing (NIST Special Publication 800-145)," *U.S Dept. of Commerce, Gaithersburg, MD (US)*, pp. 2–3, Set 2011. 7, 8, 9

[25] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, NCM '09, (Washington, DC, USA), pp. 44–51, IEEE Computer Society, 2009. 8, 9

[26] AWS, "Amazon Elastic Compute Cloud - EC2." `http://aws.amazon.com/pt/ec2/`, 2018. [Online; acessado em 19-Julho-2018]. 8, 44

[27] AWS, "Amazon Simple Storage Service (Amazon S3)." `https://aws.amazon.com/pt/s3/`, 2018. [Online; acessado em 19-Julho-2018]. 8

[28] Google, "Google App Engine." `https://cloud.google.com/appengine/docs/`, 2018. [Online; acessado em 19-Julho-2018]. 8

[29] Google, "Google Docs." `https://www.google.com/docs/about/`, 2018. [Online; acessado em 19-Julho-2018]. 9

[30] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: a survey," *annals of telecommunications - annales des télécommunications*, vol. 70, pp. 289–309, 8 2015. 10, 14

[31] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," tech. rep. 10

[32] J. Li, B. Li, T. Wo, C. Hu, J. Huai, L. Liu, and K. P. Lam, "CyberGuarder: A virtualization security assurance architecture for green cloud computing," *Future Generation Computer Systems*, vol. 28, pp. 379–390, 2 2012. 10

[33] F. Perez-Sorrosal, M. Patiño-Martinez, R. Jimenez-Peris, and B. Kemme, "Elastic SI-Cache: Consistent and scalable caching in multi-tier architectures," *VLDB Journal*, vol. 20, pp. 841–865, 12 2011. 10, 11

[34] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya, "An autonomic cloud environment for hosting ECG data analysis services," in *Future Generation Computer Systems*, vol. 28, pp. 147–154, 1 2012. 10, 11

[35] J. Espadas, A. Molina, G. Jiménez, M. Molina, R. Ramírez, and D. Concha, "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures," *Future Generation Computer Systems*, vol. 29, pp. 273–286, 1 2013. 10, 11

[36] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10*, pp. 143–154, 2010. 10, 11

[37] K. Hwang, Y. Shi, and X. Bai, "Scale-Out vs. Scale-Up Techniques for Cloud Performance and Productivity," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pp. 763–768, IEEE, 12 2014. 11, 12

[38] Y. Hirashima, K. Yamasaki, and M. Nagura, "Proactive-reactive auto-scaling mechanism for unpredictable load change," in *Proceedings - 2016 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016*, pp. 861–866, Institute of Electrical and Electronics Engineers Inc., 8 2016. 11

[39] K. Hwang, X. Bai, Y. Shi, M. Li, W.-G. Chen, and Y. Wu, "Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 130–143, 1 2016. 12

[40] T. Erl, R. Puttini, and Z. Mahmood, *Cloud computing : concepts, technology, & architecture*. 13

[41] IBM, "An architectural blueprint for autonomic computing," tech. rep., 2005. 15

[42] A. E. Rheddane, A. El, and R. Elasticity, "Elasticity in the Cloud," tech. rep. 15

[43] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," in *ACM Computing Surveys*, vol. 51, Association for Computing Machinery, 7 2018. 15, 50

[44] M. Sadegh Aslanpour, M. Ghobaei-Arani, and A. N. Toosi, "Auto-scaling Web Applications in Clouds: A Cost-Aware Approach," tech. rep. 15

[45] "AWS Auto Scaling." 16, 67

[46] P. Singh, P. Gupta, K. Jyoti, and A. Nayyar, "Research on auto-scaling of web applications in cloud: Survey, trends and future directions," *Scalable Computing*, vol. 20, pp. 399–432, 6 2019. 17, 19, 20, 21, 31, 32, 34

[47] "Elastic Compute Cloud - Amazon EC2 - AWS." 17

[48] "Flexera | Inform IT. Transform IT. Technology Insights, Spend Optimization, Tech Agility.." 17, 31, 32

[49] S. Khatua, A. Ghosh, and N. Mukherjee, "Optimizing the utilization of virtual resources in cloud environment," in *VECIMS 2010 - 2010 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, Proceedings*, pp. 82–87, 2010. 19

[50] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, pp. 500–507, 2011. 19, 34, 35, 36, 41, 63

[51] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis : forecasting and control*. John Wiley, 2008. 19, 21, 35, 36, 37, 38, 42, 58

[52] Z. Liu, H. Gu, K. Wang, B. Zhang, and K. Wang, "A novel laser source supply scheme for optical network on chip," *IEEE Access*, vol. 7, pp. 25872–25877, 2019. 21

[53] E. S. Gardner, "Forecasting the failure of component parts in computer systems: A case study," *International Journal of Forecasting*, vol. 9, no. 2, pp. 245–253, 1993. 22

[54] P. S. Kalekar, "Time series Forecasting using Holt-Winters Exponential Smoothing," tech. rep., 2004. 22

[55] P. Pereira, J. Araujo, and P. MacIel, "A hybrid mechanism of horizontal auto-scaling based on thresholds and time series," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 2019-October, pp. 2065–2070, Institute of Electrical and Electronics Engineers Inc., 10 2019. 22, 23

[56] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. 2018. 22, 23, 54

[57] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, pp. 5–10, 1 2004. 22, 37

[58] C. Chatfield, "The Holt-Winters Forecasting Procedure," *Applied Statistics*, vol. 27, no. 3, p. 264, 1978. 23, 35, 37, 38, 42, 58

[59] S. Rubab, M. F. Hassan, A. K. Mahmood, and S. N. M. Shah, "Forecasting volunteer grid workload using Holt-Winters' method," in *2nd International Symposium on Technology Management and Emerging Technologies, ISTMET 2015 - Proceeding*, pp. 422–426, Institute of Electrical and Electronics Engineers Inc., 12 2015. 23

[60] *Redes Neurais Artificiais : Teoria e Aplicações*. 24, 25

[61] H. A. OLIVEIRA-JUNIOR and H. Aguiar, *Inteligência Computacional Aplicada À Administração, Economia E Engenharia Em Matlab*. 24

[62] A. Graves, *Supervised sequence labelling with recurrent neural networks*. Springer, 2012. 24, 26, 27

[63] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012. 25, 27

[64] E. Bisong, "The Multilayer Perceptron (MLP)," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 401–405, Apress, 2019. 25

[65] T. F. d. Toledo, *Integrando Sistemas De Reconhecimento Automático De Fala Em Aplicações Web*. 2019. 26

[66] G. Bonaccorso, A. Fandango, and R. Shanmugamani, *Python advanced guide to artificial intelligence : expert machine learning systems and intelligent agents using Python*. 2018. 27

[67] "ScienceDirect - ScienceDirect.com | Science, health and medical journals, full text articles and books.." 29

[68] "IEEE Xplore Digital Library." 29

[69] "ACM Digital Library." 29

[70] "Scopus preview - Scopus - Welcome to Scopus." 29

[71] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From data center resource allocation to control theory and back," in *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, pp. 410–417, 2010. 31

[72] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, pp. 1327–1334, 2012. 31, 41

[73] RightScale, "Understanding the Voting Process." 32

[74] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai, "Exploring alternative approaches to implement an elasticity policy," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, pp. 716–723, 2011. 32, 41, 50

[75] B. Simmons, H. Ghanbari, M. Litoiu, and G. Iszlai, "Managing a SaaS application in the cloud using PaaS policy sets and a strategy-tree - IEEE Conference Publication," in *7th International Conference on Network and Services Management*, pp. 343–347, 2011. 32, 33, 41

[76] T. Lorido-Botrán, J. Miguel-Alonso, and J. Lozano, "Comparison of Auto-scaling Techniques for Cloud Environments.," 2013. 33, 41

[77] D. R. Augustyn and L. Warchal, "Metrics-based auto scaling module for Amazon web services cloud platform," in *Communications in Computer and Information Science*, vol. 716, pp. 42–52, Springer Verlag, 2017. 33, 34, 41, 49, 53

[78] S. Akioka and Y. Muraoka, "Extended forecast of CPU and network load on computational grid," in *2004 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2004*, pp. 765–772, 2004. 34, 35, 41

[79] W. Chen, J. Chen, J. Tang, and L. Wang, "A QoS Guarantee Framework for Cloud Services Based on Bayesian Prediction," in *Proceedings - 2015 3rd International Conference on Advanced Cloud and Big Data, CBD 2015*, pp. 117–124, Institute of Electrical and Electronics Engineers Inc., 3 2016. 34, 36, 41

[80] H. Fernandez, G. Pierre, and T. Kielmann, "Autoscaling web applications in heterogeneous cloud infrastructures," in *Proceedings - 2014 IEEE International Conference on Cloud Engineering, IC2E 2014*, pp. 195–204, Institute of Electrical and Electronics Engineers Inc., 9 2014. 34, 38, 41, 65

[81] R. S. Shariffdeen, D. T. Munasinghe, H. S. Bhathiya, U. K. Bandara, and H. M. Bandara, "Adaptive workload prediction for proactive auto scaling in PaaS systems," in *Proceedings of 2016 International Conference on Cloud Computing Technologies and Applications, CloudTech 2016*, pp. 22–29, Institute of Electrical and Electronics Engineers Inc., 2 2017. 34, 37, 41, 55

[82] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS," *IEEE Transactions on Cloud Computing*, vol. 3, pp. 449–458, 10 2015. 34, 38, 41

[83] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, pp. 204–212, 2012. 34, 39, 41

[84] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proceedings - IEEE INFOCOM*, pp. 460–468, 2012. 34, 38, 41

[85] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *Proceedings - 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013*, pp. 58–65, 2013. 34, 37, 41

[86] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier Internet applications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, pp. 1–39, 3 2008. 34, 39, 40, 41

[87] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," in *Future Generation Computer Systems*, vol. 27, pp. 871–879, 6 2011. 34, 39, 41

[88] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload Predicting-Based Automatic Scaling in Service Clouds," pp. 810–815, Institute of Electrical and Electronics Engineers (IEEE), 2 2014. 34, 37, 41

[89] A.-F. Antonescu and T. Braun, "Improving management of distributed services using correlations and predictions in SLA-driven cloud computing systems," in *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, pp. 1–8, 2014. 34, 35, 41

[90] A. Y. Nikravesh, S. A. Ajila, and C. H. Lung, "An autonomic prediction suite for cloud resource provisioning," *Journal of Cloud Computing*, vol. 6, 12 2017. 34, 36, 41

[91] R. R Core Team, "R: a language and environment for statistical computing." 35

[92] CloudSim, "The cloud computing and distributed systems (clouds) laboratory http://www.cloudbus.org/cloudsim/," *University of Melbourne*. 36

[93] R. G. Brown, R. F. Meyer, and D. A. D'Esopo, "The Fundamental Theorem of Exponential Smoothing." 36

[94] V. N. Vapnik, "An Overview of Statistical Learning Theory," Tech. Rep. 5, 1999. 36

[95] M. Minsky and S. Papert, *Perceptrons; an introduction to computational geometry*. MIT Press, 1969. 36, 42

[96] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. 36, 37

[97] J. D. Hamilton, *Time Series Analysis*, vol. 11. Princeton, NJ: Princeton University Press, 1994. 38

[98] A. K. Erlang, "Sandsynlighedsregning og telefonsamtaler," *Nyt tidsskrift for matematik*, vol. 20, pp. 33–39, 1909. 38

[99] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997. 42

[100] AWS, "GetMetricStatistics - Amazon CloudWatch." 49

[101] "Boto 3 Documentation — Boto 3 Docs 1.10.39 documentation." 49, 63

[102] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 1–12, 1 2019. 49

[103] A. Evangelidis, D. Parker, and R. Bahsoon, "Performance Modelling and Verification of Cloud-Based Auto-Scaling Policies," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 355–364, IEEE, 5 2017. 50

[104] DynamoDB, "Amazon DynamoDB – Visão geral." 50

[105] L. Yazdanov, *Towards auto-scaling in the cloud: online resource allocation techniques*. PhD thesis, Technische Universitat Dresden, 2017. 51

[106] F. Kane, *Hands-On Data Science and Python Machine Learning*. 2017. 53

[107] "Tools That Work With Graphite — Graphite 1.1.5 documentation." 53

[108] Y. Jiang, C. S. Perng, T. Li, and R. Chang, "Self-adaptive cloud capacity planning," in *Proceedings - 2012 IEEE 9th International Conference on Services Computing, SCC 2012*, pp. 73–80, 2012. 57

[109] C. Vazquez, R. Krishnan, and E. John, "Time Series Forecasting of Cloud Data Center Workloads for Dynamic Resource Provisioning," *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Ap plications (JoWUA) 6 (3)*, p. 87–110, 2015. 57

[110] C. Liu, Y. Shang, L. Duan, S. Chen, C. Liu, and J. Chen, "Optimizing Workload Category for Adaptive Workload Prediction in Service Clouds," in *13th International Conference on Service-Oriented Computing (ICSOC 2015)*, pp. 87–104, 2015. 58

[111] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *Journal of Network and Computer Applications*, vol. 82, pp. 93–113, 3 2017. 58

[112] Y. Kouki, *Approche dirigée par les contrats de niveaux de service pour la gestion de l'élasticité du "nuage*

. PhD thesis, Ecole des Mines de Nantes, 2013. 59

[113] Python, "Welcome to Python.org." 63

[114] Zabbix, "Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution." 64

[115] "MediaWiki." 65

[116] "English Wikipedia - Wikipedia." 65

[117] C. Qu, R. N. Calheiros, and R. Buyya, "A Reliable and Cost-Efficient Auto-Scaling System for Web Applications Using Heterogeneous Spot Instances," tech. rep., 2016. 65, 68

[118] C. Qu, *Auto-scaling and Deployment of Web Applications in Distributed Computing Clouds*. PhD thesis, 2016. 65

[119] "Elastic Load Balancing – Amazon Web Services." 66, 67

[120] "Status codes in HTTP." 69

[121] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems)*. 2011. 79, 80

[122] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, pp. 679–688, 10 2006. 80

[123] M. Theodosiou, "Forecasting monthly and quarterly time series using STL decomposition," *International Journal of Forecasting*, vol. 27, pp. 1178–1195, 10 2011. 80

[124] A. Bauer, N. Herbst, S. Spinner, A. Ali-Eldin, and S. Kounev, "Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, pp. 800–813, 4 2019. 80