Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Auto-scaling em Uma Empresa de E-commerce: Um Estudo de Caso

Mariana Mendes e Silva

Campina Grande, Paraíba, Brasil

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Auto-scaling em Uma Empresa de E-commerce: Um Estudo de Caso

Mariana Mendes e Silva

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sistemas de Computação

Raquel Vigolvino Lopes
Thiago Emmanuel Pereira
(Orientadores)

Campina Grande, Paraíba, Brasil

S586a Silva, Mariana Mendes e.
Auto-scaling em uma empresa de e-commerce: um estudo de caso / Mariana Mendes e Silva. – Campina Grande, 2023.
87 f.: il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2022.
"Orientação: Profa. Dra. Raquel Vigolvino Lopes, Prof. Dr. Thiago Emmanuel Pereira".
Referências.

1. Engenharia de Programas de Computador. 2. Arquitetura de Computadores. 3. Auto-scaling. 2. E-commerce. 3. Serviços em Nuvens. I. Lopes, Raquel Vigolvino. II. Pereira, Thiago Emmanuel. III. Título.

CDU 004.41043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MARIA ANTONIA DE SOUSA CRB 15/398

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO CIENCIAS DA COMPUTACAO
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

MARIANA MENDES E SILVA

AUTO-SCALING EM UMA EMPRESA DE E-COMMERCE: UM ESTUDO DE CASO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 08/12/2022

Profa. Dra. RAQUEL VIGOLVINO LOPES, Orientadora, UFPB
Prof. Dr. THIAGO EMMANUEL PEREIRA DA CUNHA SILVA, Orientador, UFCG
Prof. Dr. FRANCISCO VILAR BRASILEIRO, Examinador Interno, UFCG
Prof. Dr. PAULO DITARSO MACIEL JÚNIOR, Examinador Externo, IFPB

Documento assinado eletronicamente por **RAQUEL VIGOLVINO LOPES**, **Usuário Externo**, em 09/12/2022, às 09:16, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Paulo Ditarso Maciel Júnior**, **Usuário Externo**, em 09/12/2022, às 10:52, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **FRANCISCO VILAR BRASILEIRO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 09/12/2022, às 19:37, conforme horário oficial de Brasília, com fundamento

no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **THIAGO EMMANUEL PEREIRA DA CUNHA SILVA**, **PROFESSOR 3 GRAU**, em 13/12/2022, às 10:20, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **2965216** e o código CRC **B3E4BCAE**.

Referência: Processo nº 23096.085621/2022-52 SEI nº 2965216

Resumo

O modelo IaaS ainda apresenta muitos desafios, um deles é o problema de provisionamento dinâmico de recursos. Esse provisionamento é feito geralmente a partir do mecanismo *auto-scaling*. Configurar e avaliar esse mecanismo não é uma tarefa trivial. Na literatura existe uma gama de trabalhos sobre o *auto-scaling* que em sua grande maioria propõem novos algoritmos. Apenas uma pequena parcela considera a configuração e avaliação, e ainda assim nenhum traz uma ênfase sobre como isso é feito na prática, considerando as dificuldades que provavelmente serão encontradas no contexto da indústria. Para entender melhor essa questão, realizamos um estudo de caso em uma empresa cliente de um provedor de nuvem. Construímos um questionário e analisamos seus resultados. Para avaliar a utilização da infraestrutura de fato, construímos uma ferramenta de coleta de dados para a infraestrutura e realizamos a análise desses dados. Com essas duas ferramentas - questionário e coletor -, conseguimos entender as maiores dificuldades na configuração e avaliação e correlacioná-las com as análises de dados da infraestrutura. Ao final, realizamos também um mapeamento dos pontos de melhoria encontrados. Portanto, a presente pesquisa dá os primeiros passos no sentido de sistematizar uma avaliação e configuração prática de serviço de *auto-scaling*.

Abstract

The IaaS model brings many challenges, one of which is the dynamic resource provisioning, which is made by an auto-scaling mechanism. Configuring and evaluating this mechanism is a complex task. There are many works about it in the literature, but most are about new algorithms. Only a few consider configuration and evaluation and don't emphasize the practical aspect, considering the difficulties that probably exist in the industry context. To better understand, we built a case study in a company client from a cloud provider. We wrote a questionnaire e analyzed the results. We created a tool to collect data to evaluate infrastructure utilization. We understood the most significant difficulties in configuration and evaluation with these two tools - questionnaire and collector - and then correlated them with infrastructure data analysis. In the end, we also made a mapping with improvement points. Therefore, this research takes the first steps towards systematizing the evaluation and practical configuration of an *auto-scaling* service.

Agradecimentos

Agradeço a minha família, que esteve e está ao meu lado me motivando, especialmente minha mãe Maria Aparecida.

Aos meu orientadores, Thiago Emmanuel e Raquel pela enorme paciência que tiveram durante toda a pesquisa e por todos os ensinamentos. À todos do time *Tinyverse* - inclusive coordenadores Talita, Fábio, Igor, Daniel, pela competência e dedicação num projeto que deu todo suporte a essa pesquisa.

Aos meus companheiros de sala no LSD (Laboratório de Sistemas Distribuídos) pela ótima companhia, especialmente nos dias difíceis. Também agradeço aos meus amigos que me acompanham desde a graduação, cuja convivência foi interrompida por causa da pandemia. Aos meus amigos de infância Caio, Ana Biatriz, e ao meu parceiro Pedro, pela paciência e motivação.

A todos, muito obrigada!

Conteúdo

Glossário

ADI *Auto-scaling Demand Index*. 7

CID *Continuous Integration and Delivery*. 14, 15

IaaS *Infrastructure as a Service*. 1

PaaS *Platform as a Service*. 1

QoS *Quality of Service*. 9, 10

SaaS *Software as a Service*. 1

SLA *Service Level Agreement*. 10

SLO *Service Level Objective*. 8

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A computação em nuvem é um paradigma que continua a atrair bastante a atenção da indústria. Segundo uma pesquisa do *Research and Markets* [1], o mercado nessa área tem uma expectativa de crescimento de US$371,4 bilhões em 2020 para US$832,1 bilhões em 2025. Neste paradigma existem três modelos principais: Plataforma como Serviço (PaaS, do inglês *Platform as a Service*), Software como Serviço (SaaS, do inglês *Software as a Service*), e Infraestrutura como Serviço (IaaS, do inglês *Infrastructure as a Service*).

A modalidade IaaS é bastante popular; oferece hardware (servidor, armazenamento e rede) e software associado, tudo isso como um serviço. É uma evolução da hospedagem tradicional, não exigindo compromisso de longo prazo e permitindo provisionamento de recursos sob demanda. Essa última característica do modelo é o que permite a elasticidade, que pode ser definida como a capacidade de adquirir ou liberar recursos, às vezes de forma automática, para aumentar e/ou diminuir gradativamente a capacidade de uma infraestrutura de acordo com a demanda [1]. Na visão do consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ser ilimitados e podem ser requisitados em qualquer quantidade a qualquer momento [22], além da possibilidade de seguir o modelo de tarifação "pago conforme utilização", onde os clientes pagam unicamente pelos recursos adquiridos.

Apesar de todos os benefícios do IaaS, existem alguns desafios relevantes na literatura atrelados à gerência de recursos nesse modelo. Um deles é o problema de **Provisão de Recursos** que diz respeito à capacidade do sistema de realizar um ajuste dinâmico, na sua

[1] `https://www.researchandmarkets.com/reports/5136796/cloud-computing-market-by-service-model`

infraestrutura, para atender a demanda [21]. Normalmente, esse ajuste é feito através do mecanismo de *auto-scaling*, definido como uma técnica de computação em nuvem para alocar e liberar dinamicamente recursos computacionais na forma de máquinas virtuais ou *containers*.

Esse mecanismo pode seguir três tipos diferentes de abordagem: reativa, proativa ou híbrida. A proativa tenta prever a carga da aplicação para tomar decisões sobre a capacidade adequada antes que a aplicação sofra perda de desempenho ou seus recursos fiquem subutilizados. A técnica reativa consiste em uma reação programada às mudanças percebidas na carga de trabalho da aplicação e/ou na sua infraestrutura [23]. Já a abordagem híbrida faz uso da técnica reativa e da proativa em conjunto. A abordagem reativa é bem difundida entre os provedores públicos, possivelmente devido a sua simplicidade e natureza intuitiva [10].

1.1 Motivação

Na literatura existe uma abundância de trabalhos envolvendo o *auto-scaling*. A maioria é voltada para o desenvolvimento de algoritmos de otimização [7, 18, 4, 17, 27, 3, 24] do próprio serviço que toma decisões sobre a capacidade dinâmica da infraestrutura. Outra parcela desses trabalhos é focada em estratégias mais específicas para configuração dos parâmetros do serviço [12, 13, 2]. Por último, em quantidade bem menor, temos os trabalhos que focam no processo de avaliação do *auto-scaling* [25, 8, 26] de maneira mais generalista.

Dentre os trabalhos citados existe uma limitação comum. Esses trabalhos costumam considerar cenários não diretamente aplicáveis na indústria. Na sua metodologia, consideram cargas que podem não representar bem a realidade por serem antigas, sintéticas, reduzidas ou com escopo limitado. Ou ainda, consideram estratégias de difícil implementação prática. Sendo assim, não sabemos se as conclusões discutidas na literatura são robustas para outros cenários.

Apesar desses trabalhos servirem como inspiração para as soluções práticas, existe uma lacuna na literatura: trabalhos que abracem especificidades dos sistemas reais e os problemas práticos envolvidos na gerência de recursos provenientes da nuvem. Este trabalho vai na direção de olhar este problema de forma muito atrelada a um estudo de caso prático de uma empresa real, que precisa lidar com a gerência de recursos provenientes da nuvem no seu dia

a dia, bem como extrair lições e ações importantes neste contexto.

1.2 Objetivos e Contribuições

A proposta de novas estratégias para melhorar o *auto-scaling* é um tema bastante explorado. Acompanhada dessas estratégias, os autores também propuseram métodos para avaliá-las. O objetivo destas avaliações é, em geral, comparar a nova estratégia com outra(s) e mostrar que a nova estratégia se sai melhor em alguns cenários importantes, seja por ser mais barata, mais simples ou mais eficiente em manter o bom desempenho das aplicações. Esse tipo de avaliação, apesar de importante no contexto da aceitação e entendimento de novos algoritmos, não é adequado quando levamos o problema para um contexto mais prático de um usuário de serviços públicos de nuvem, que pretende avaliar a estratégia de *auto-scaling* oferecida pelo provedor que está em uso. Esse usuário tem o objetivo de certificar-se que o serviço em uso, na forma como está configurado, está atendendo suas necessidades. Alguns estudos vêm olhando essa problemática de forma mais global, sem associar uma avaliação a novos algoritmos [25, 8, 26]. Apesar de apresentarem a questão da avaliação de forma mais genérica, esses estudos geralmente usam dados com abrangência limitada.

Portanto, este trabalho pretende investigar a avaliação do *auto-scaling* considerando o caso particular do uso do serviço no cenário de uma empresa de médio porte que usa nuvem pública. Alguns aspectos práticos do caso estudado guiaram a pesquisa para **dar os primeiros passos no sentido de sistematizar uma avaliação prática do serviço de *auto-scaling***.

Para atingir o objetivo principal do trabalho a metodologia possui duas atividades principais, que podem ser resumidas da seguinte forma:

1. Desenvolvimento de uma ferramenta de coleta, necessária para a obtenção de dados da infraestrutura em que o mecanismo de *auto-scaling* é utilizado. Os dados coletados a partir desse instrumento foram analisados a partir de métricas selecionadas, gerando um diagnóstico sobre a configuração atual do *auto-scaling.*

2. Desenvolvimento e aplicação de um questionário para que fosse possível entender melhor o contexto do uso de *auto-scaling* na prática e realizar uma análise de dados mais estruturada. As respostas desse questionário foram analisadas, gerando insumos

sobre como é o processo de configuração e avaliação do *auto-scaling*.

A principal contribuição deste trabalho é iniciar dentro da empresa um processo para avaliação do *auto-scaling* de forma sistemática, sendo essa experiência aplicável em outras empresas. Inicialmente, foi necessária a coleta de dados da infraestrutura onde estão as aplicações, para a qual desenvolvemos uma ferramenta. Portanto, outra contribuição consiste na ferramenta de coleta desenvolvida, descrita no Capítulo 5, bem como o conjunto de dados coletados, descritos no Capítulo 6. Para seguir entendendo melhor o contexto prático de configuração e avaliação e fazer análises alinhadas, foi necessária a elaboração e a análise de um questionário. Seus resultados são também uma contribuição, pois trouxe informações sobre como ocorre a configuração do *auto-scaling* em um cenário real e quais as maiores dificuldades nessa configuração, dada a limitação de tempo e treinamento específico desse serviço.

Além disso, os pesquisadores da área podem ter acesso a conhecimento que vem de uma abordagem prática da avaliação de *auto-scaling*, com dados reais, o que pode gerar novos conhecimentos e abrir novos planos de pesquisa a partir daqui. Nesse sentido, podemos citar alguns resultados secundários que foram alcançados através das contribuições deste trabalho no Capítulo 10.

1.3 Estrutura do Documento

O restante deste documento está estruturado da seguinte forma. No Capítulo 2 estão os trabalhos relacionados à avaliação e configuração do serviço *auto-scaling*. O Capítulo 3 traz uma contextualização mais detalhada sobre o estudo, descrevendo a empresa alvo, sua organização e tecnologias utilizadas, pontos cruciais para entendimento dos objetivos e análises da pesquisa. Já no Capítulo 4 apresentamos os materiais e métodos utilizados; caracterizando o estudo, descrevendo o método de questionário e coletor de dados. Nos capítulos seguintes, expandimos e detalhamos as ferramentas construídas e utilizadas. No Capítulo 5, descrevemos o desenvolvimento da ferramenta de coleta de dados e seu funcionamento e logo depois, no Capítulo 6, a análise descritiva dos dados obtidos. No Capítulo 7 estão as análises das repostas obtidas via questionário. No Capítulo 8 estão as descrições de todas as métricas usadas no diagnóstico do *auto-scaling* e em seguida, no Capítulo 9, a aplicação de

fato das métricas sobre os dados coletados. Por fim, no Capítulo 10 estão trabalhos futuros e resultados secundários.

Capítulo 2

Trabalhos Relacionados

Neste Capítulo apresentamos o estado-da-arte relacionado ao *auto-scaling*. É um tema bastante discutido na literatura de diversas formas, principalmente o desenvolvimento de novos algoritmos e otimizações [3, 7, 18, 4, 17, 27]. Existem alguns trabalhos que abordam o tipo de escala - horizontal ou vertical - [15], a proposição de algoritmos proativos [17, 27], estratégias proativas para decidir o tipo de instância mais adequado [23] e algoritmos híbridos [7]. Contudo, reduzimos o escopo dessa pesquisa para considerar apenas o *auto-scaling* horizontal reativo, que é o modelo utilizado atualmente na empresa alvo do estudo de caso. Além disso, como citado anteriormente, esse modelo é bastante difundido por seu caráter intuitivo. Portanto a seguir discutimos os artigos que se encaixam nesse contexto, divididos entre: Métricas, Estratégias e Avaliação do *auto-scaling*.

2.1 Métricas

Quando falamos de métricas aqui existem duas definições: 1. Métricas utilizadas para avaliar a eficácia do *auto-scaling*, configurações, estratégias, etc; 2. Métricas utilizadas na configuração do *auto-scaling*, ou seja, que serão monitoradas para decisões de provisionamento ou redução de recursos. Nessa seção estão os trabalhos que propõem novas métricas para avaliar o mecanismo estudado.

No trabalho de Herbst et. al, os autores exploram uma definição de métricas para avaliar o *auto-scaling*, considerando diferentes aspectos de uma aplicação, dentre eles a elasticidade [14]. Algumas métricas definidas nesse trabalho já foram utilizadas em outros experimentos,

como é o caso do *Tempo de Provisionamento Incorreto* e *Acurácia* que aparecem em [16]. Elas são de fácil medição, uma vez que métricas de sistema (como Utilização de CPU) são comumente monitoradas, portanto, podem ser incorporadas a análises sem tanta complexidade.

Em [25] os autores também trazem a definição de uma nova métrica, chamada de ADI (*Auto-scaling Demand Index*). Essa métrica tem como objetivo resumir, em um único número, a diferença entre a utilização real do sistema e o valor alvo de utilização configurado pelo usuário, e também pode ser facilmente incorporada.

As métricas utilizadas para avaliação estão presentes em todos as categorias de trabalhos desta revisão, portanto, na Tabela 2.1 temos um resumo de métricas encontradas na literatura que foram utilizadas para avaliar o *auto-scaling*.

Como foi dito no início da seção, também existem as métricas que são utilizadas na própria configuração do *auto-scaling*. Para essa categoria, não foram encontrados trabalhos que trazem uma definição totalmente nova de alguma métrica utilizada na configuração. Porém, existem algumas discussões sobre isso, como por exemplo a escolha entre usar uma métrica de aplicação ou do sistema. Os autores de [5] trazem uma discussão importante sobre a natureza das métricas (intrusivas ou não) utilizadas para desencadear ações de *scaling* e sua adequação para diferentes tipos de carga de trabalho. Posteriormente, no Capítulo 7, veremos que quase 100% das aplicações da empresa alvo, que foram analisadas, utilizam apenas a métrica de Utilização de CPU para desencadear ações de *scaling*.

2.2 Estratégias

Além das definições de métricas, outro tipo de trabalho também relevante para esta pesquisa é o que traz estratégias para mitigar problemas específicos ou definir valores de determinados parâmetros. No trabalho publicado por Netto et. al.[25] é mostrada uma estratégia para definir valores de *step size* (quantidade de máquinas adicionadas, ou removidas, a cada ação de *scaling*), além da definição de uma métrica avaliativa para o *auto-scaling*. Já em [4] os autores trazem uma estratégia direcionada a lidar com o sub-provisionamento durante pi-

Nome da Métrica	Breve Definição	Referências
Acurácia	Quantidade relativa de recursos que estão sub-provisionados ou super-provisionados durante o intervalo de medição.	[14, 5]
Tempo de Provisionamento	Tempo em que o sistema está sub-provisionado ou super-provisionado durante o intervalo de medição.	[14, 5]
Elastic Speedup	Calcula, para cada abordagem, o ganho entre essa abordagem e o cenário de não usar *auto-scaling*. Esse ganho é calculado combinando as duas métricas citadas acima (Acurácia e Tempo de Provisionamento).	[5]
Jitter ou Instabilidade	Compara a quantidade de ações de *scaling* da curva de provisionamento com a quantidade da curva de demanda. Captura quanto o mecanismo oscila até ficar estável.	[14, 16]
Violações de SLO	Quantidade de violações dado um SLO definido.	[5]
ADI	Soma de todas as distâncias entre a utilização do sistema e a utilização alvo definida pelo usuário num dado intervalo.	[25]
Quantidade de Máquinas Virtuais	Quantidade de máquinas virtuais alocadas em um dado intervalo.	[5]
Tempo de Resposta	Tempo gasto pelas máquinas virtuais para responder requisições.	[5, 2]
Custo	Custo dos recursos alocados.	[5]

Tabela 2.1: Resumo de métricas utilizadas para avaliar o auto-scaling.

cos na carga, utilizando instâncias *burstable* [1], que conseguem acumular créditos e utilizá-los para aumentar sua capacidade computacional durante alguns instantes, mascarando assim o tempo de "rampa", definido como o tempo entre o instante em que a regra de *auto-scaling* foi ativada até o instante em que o recurso fica disponível. Em [2], além da descrição do impacto da CPU em dois parâmetros do *auto-scaling*, também são mostradas estratégias para se determinar um valor ótimo para esses parâmetros. Essas análises que tentam mostrar a correlação entre as métricas e os parâmetros de configuração podem ser de grande valor na redefinição das configurações. Na prática, como veremos posteriormente, é comum que a configuração do *auto-scaling* seja feita de forma intuitiva ou fazendo pequenos testes com a aplicação sem um entendimento mais global de decisões de configuração deste serviço, de suas consequências para a aplicação e o custo de mantê-la.

Por fim, em [28] fala-se também da importância de talvez combinar os dois tipos de métricas (aplicação e sistema) para um *auto-scaling* mais eficiente. Estratégia ainda pouco experimentada, especialmente na indústria.

2.3 Avaliação do *auto-scaling*

Nessa última categoria, temos trabalhos que trazem metodologias ou estratégias de comparação e avaliação do *auto-scaling*, especificamente da configuração do mesmo. No trabalho de Papadoulos et. al. [26] é apresentada uma forma de avaliação através de um *framework* teórico, onde o objetivo é identificar o quão próximo o *auto-scaling* chega do cenário de super e sub-provisionamento para comparar diferentes estratégias e/ou configurações. A métrica derivada foi obtida através da utilização de CPU, mas pode ser obtida a partir de outras métricas de sistema, por exemplo a utilização de memória. Nesse *framework*, as métricas de alto nível (de aplicação) são descartadas completamente, o que pode não ser tão interessante para todos os tipos de aplicação.

Já Evangelidis et. al. [12] propõem um modelo de verificação probabilístico, que a partir de uma configuração passada como entrada, retorna um valor indicando se essa política viola ou não a QoS (*Quality of Service*), considerando Utilização de CPU e tempo de resposta

[1] `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-performance-instances.html`

como métricas de QoS. Nesse caso, são consideradas métricas de sistema e de aplicação na avaliação. Também Fé et. al.[13] trazem um modelo estocástico para tentar definir uma configuração que respeita o SLA *(Service Level Agreement)*, ao mesmo tempo que reduz custos. Os trabalhos citados avaliam o mecanismo em si e não necessariamente precisam de mais de uma estratégia para realizar uma avaliação e comparação.

Sabendo que o objetivo desta pesquisa é dar início a uma possível sistematização na avaliação do *auto-scaling* em um contexto prático, também procuramos alguns trabalhos que fazem avaliação de estratégias e políticas/configurações [16, 12, 26]. Esses trabalhos trazem métodos para avaliar a configuração realizada e também para compará-la à outras, sendo assim importantes para a pesquisa atual, pois além da metodologia para comparar ou avaliar, também mostram o uso de algumas métricas já definidas em outros artigos.

Os trabalhos relacionados à avaliação do *auto-scaling* de fato se aproximam do objetivo e contribuições da pesquisa atual. Contudo, além das diferenças, apresentam uma limitação quanto às cargas utilizadas em seus experimentos, que podem não representar bem a realidade, seja por serem antigas, sintéticas ou reduzidas. Sendo assim, não sabemos se as conclusões discutidas são genéricas o suficiente para outros cenários.

Além da limitação de carga, muitos desses trabalhos não consideram cenários reais, dando pouca ênfase à praticidade e clareza do que está sendo desenvolvido, fazendo com que sua aplicação seja complexa. Por fim, também não foi possível encontrar trabalhos que colocassem ênfase no cenário prático de configuração e avaliação de *auto-scaling* em provedores públicos e todos os fatores envolvidos, como tempo disponível para decidir configurações e como as pessoas tomam decisões.

Capítulo 3

Contexto do Estudo de Caso

Esta pesquisa descreve um estudo de caso sobre gerência de recursos, realizado em uma empresa brasileira de comércio eletrônico. O modo como se faz o gerenciamento de recursos nesta empresa é afetado por seus aspectos organizacionais e também tecnológicos, incluindo serviços desenvolvidos e tecnologias utilizadas. Para embasar nossas discussões e análises posteriores, neste Capítulo descrevemos o contexto da empresa onde o estudo de caso foi realizado.

3.1 A Empresa

Nosso estudo de caso envolveu uma multinacional brasileira que desenvolve tecnologias para implantação e gestão de lojas virtuais, negócios *e-commerce*, comércio unificado e *omnichannel* [1]. Um dos principais serviços oferecidos é uma plataforma de *e-commerce* ofertada no modelo *SaaS*. Essa plataforma é responsável por funcionalidades que permitem que os lojistas cadastrem seus produtos, integrem meios de pagamento, calculem frete, etc. Dessa forma, as empresas de comercialização, clientes da plataforma, podem focar apenas na venda de seus produtos no ambiente *online*.

A empresa tem pouco mais de 20 anos de trajetória e apresenta um grande crescimento ao longo do tempo. Em 2016, a consultoria internacional Deloitte [2] reconheceu a empresa como a PME (Pequenas e Médias Empresas) de maior crescimento no Brasil. Em 2019, a

[1] `https://www.zendesk.com.br/blog/omnichannel/`

[2] `https://www2.deloitte.com/br/pt.html`

empresa recebeu um aporte de R$ 580 milhões do Softbank [3], Constellation e Gávea. Já em 2020 e 2021, a Gartner [4] citou a empresa no principal relatório de tecnologia do mundo, o Quadrante Mágico Gartner, ao lado da Oracle e da IBM [5]. Ainda em 2021, a multinacional estreou na Bolsa de Valores de Nova York

Com esse crescimento continuo, é esperado que haja um uso mais intensivo de recursos computacionais para atender tanto uma maior demanda, como suportar novos serviços. A mudança também se dá no sentido de que inicialmente a empresa era uma *startup*, onde a maior preocupação estava relacionada à entrega do produto. Contudo, ao entrar na bolsa de valores, surge um novo desafio: manter a qualidade de serviços e reduzir os custos. De modo que, a gerência de recursos se torna então uma das áreas mais importantes na redução de custos.

Com um contexto melhor da empresa e da importância da gerência de recursos, entenderemos um pouco mais sobre a organização de times e dos recursos na nuvem, bem como esses dois pontos se relacionam, fatores que também afetam a gerência nesse estudo.

3.2 Organização Interna

A empresa segue uma estrutura de divisão organizacional baseada em Produto [6] e conta com centenas de funcionários na área de tecnologia. Esses funcionários se dividem em quase 50 times diferentes.

Para entender melhor a divisão interna, é importante termos em mente os termos *Produto* e *Aplicação*. Um Produto é a entidade maior de um serviço e pode ser composto de uma ou mais aplicações. Por exemplo, um Produto pode ser o serviço de *Checkout* ou de um Catálogo de Produtos. Já uma *Aplicação* está **contida** em um Produto, como por exemplo uma aplicação Carrinho de Compras que está contida no serviço de *Checkout*, ou a API do Catálogo de Produtos que está contida no Produto de Catálogo de Produtos.

Cada um dos times, geralmente, está ligado a um Produto e se subdivide para trabalhar nas Aplicações. Cada time possui uma pessoa gerente, que por sua vez responde aos dire-

[3] `https://www.softbank.jp/en/`

[4] `https://www.gartner.com.br/pt-br`

[5] `https://www.atwix.com/ecommerce/gartner-quadrant-digital-commerce-2022-vs-2021/`

[6] `https://www.lucidchart.com/blog/types-of-organizational-structures`

tores e diretoras de engenharia. Apesar dos objetivos alinhados, os times são independentes e geralmente as decisões de como lidar com a infraestrutura da aplicação são realizadas internamente. Ou seja, é comum que não existam ações conjuntas para definir pontos relacionados à gerência de recursos, como por exemplo a quantidade de instâncias.

3.3 Recursos

Os serviços da empresa são mantidos a partir de duas nuvens públicas. O foco desse estudo é apenas uma das nuvens, que opera no nível de infraestrutura como serviço. Para que houvesse um controle maior dos recursos, no final do ano de 2021, a empresa adotou a obrigatoriedade de usar um serviço de *Tags* [7], fornecido pela nuvem no momento da solicitação de recursos. Esse serviço de *Tags*, de forma resumida, permite adicionar rótulos a todos os tipos de recursos (máquinas virtuais, discos, etc) no formato chave-valor. Esses rótulos podem ajudar a gerenciar, identificar, organizar, pesquisar e filtrar recursos. É possível criar rótulos para categorizar recursos por finalidade, proprietário, ambiente, etc. Usando esse serviço a empresa pode ter um controle maior de custos por serviços oferecidos. Além disso, também possibilita análises mais direcionadas, pois é possível agrupar recursos de acordo com as *Tags*. As *Tags* obrigatórias, que são importantes para entendimento das análises posteriormente, são:

- *Product*: Nome do produto do qual o recurso é parte.
- *ApplicationName*: Nome do subcomponente de um Produto.
- *Environment*: Tipo de carga de trabalho que chega ao recurso.
- *Name*: Identificação do recurso.

Exemplo do uso de *Tags* para o Produto Catálogo de Produtos está descrito na Tabela 3.1.

[7] `https://docs.aws.amazon.com/pt_br/general/latest/gr/aws_tagging.html`

Chave	Valor
Product	Catálogo
ApplicationName	Busca-Catálogo
Environment	Produção
Name	Busca-Catálogo-versão-1

Tabela 3.1: Tabela de Exemplo do uso de *Tags*.

3.4 Tecnologias

Para entender um pouco mais alguns aspectos que afetam as análises da pesquisa, precisamos garantir um contexto sobre as tecnologias utilizadas pela empresa e como elas se relacionam para implantar aplicações.

Existem três tecnologias principais nesse processo de *deployment*:

1. **CID** (Continuous Integration and Delivery). Ferramenta de integração contínua desenvolvida pela empresa, que vai se comunicar com o serviço de orquestração de recursos e passar as configurações especificadas pelo usuário;

2. **Serviço de orquestração**. Usado para a implantação de Aplicações, que orquestra vários outros serviços, como Instâncias, Armazenamento, Balanceadores de Carga, etc.

3. **Serviço de escalabilidade**. Gerencia os recursos fornecidos pela nuvem. Isso inclui o controle de mercado de instâncias e também do mecanismo de *auto-scaling*.

Observando o diagrama da Figura 3.1 podemos ter uma visão geral de como é o fluxo de implantação de uma nova versão de uma Aplicação. Para exemplificar melhor, vamos descrever o passo a passo a seguir, que pode ser acompanhado também pelo diagrama a seguir:

1. As pessoas que desenvolvem a Aplicação definem um arquivo de configuração com informações como: número mínimo de instâncias, número máximo de instâncias, política de *auto-scaling*, métrica utilizada para a política, limiares da política, etc. Ou seja, configurações que dizem respeito à infraestrutura daquela Aplicação.

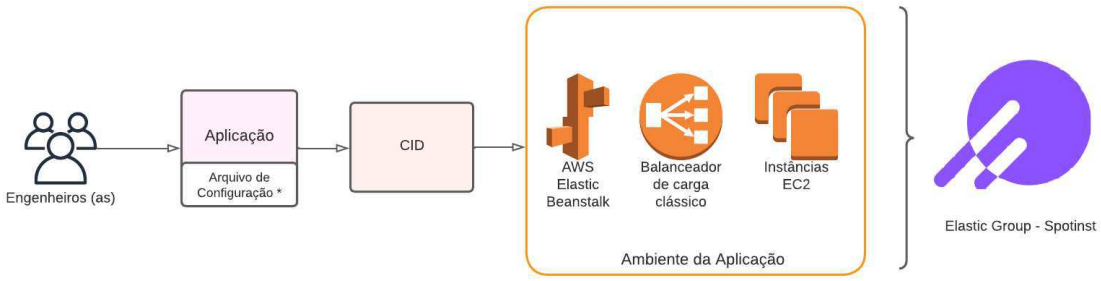

Figura 3.1: Fluxo de *deployment* de uma aplicação.

2. O CID entende esse **Arquivo de Configuração** e cria um novo ambiente de acordo com as informações nele contidas.

3. Em seguida, em um serviço externo à nuvem, será inicializado um agrupamento lógico de todos os recursos que foram criados anteriormente, respeitando o que foi passado no arquivo de configuração. Esse agrupamento é chamado *ElasticGroup*[8], serviço responsável pelo *auto-scaling*.

Esse método de implantação, utilizando essas três ferramentas permite que os times realizem mudanças nas Aplicações com frequência, de forma padronizada e independente. Todas as vezes que uma mudança na aplicação é realizada e uma nova implantação é requisitada, todo o processo descrito na Figura 3.1 é repetido. Esse processo gera um novo ambiente e é realizada uma migração gradual da carga para a nova versão da aplicação, que possui agora um novo ambiente, com novos recursos.

[8] `https://spot.io/products/elastigroup/`

Capítulo 4

Metodologia

Neste Capítulo estão descritos os métodos utilizados para a realização desta pesquisa. Inicialmente falaremos do método de Estudo de Caso, que caracteriza o estudo realizado, as técnicas utilizadas, etc. Posteriormente, sobre os dois métodos de coleta de dados e a metodologia utilizada para desenvolvê-los.

Esta pesquisa tem como objetivo principal investigar a avaliação do serviço de *auto-scaling* considerando o cenário de uma empresa de médio porte que utiliza nuvem pública. Com essa investigação, daremos os primeiros passos no sentido de sistematizar uma avaliação prática do serviço de *auto-scaling*.

Inicialmente investigamos e exploramos o problema da configuração e a avaliação do *auto-scaling* através da coleta e análise de dados da infraestrutura. Essa coleta foi realizada por uma ferramenta construída ao longo da pesquisa, permitindo a análise aprofundada da utilização de recursos. Para realizar esses passos, foi decidido que antes seria necessário fazer um levantamento das opiniões e impressões das pessoas que realizam essas configurações e avaliações dentro da empresa. Esse levantamento utilizou um questionário como instrumento de coleta, que serviu como base para a avaliação e posteriormente as sugestões de pontos de melhoria.

Com os dados coletados no período de 01/05/2022 a 30/05/2022 foi possível realizar uma série de avaliações do serviço de *auto-scaling* de várias aplicações.

4.1 Caracterização do Estudo

Para caracterizar esse estudo, baseados nesses objetivos e atividades, utilizamos as definições de Wazlawick et. al. [31] categorizando a pesquisa quanto a sua natureza, objetivos e procedimentos técnicos.

Quanto à natureza. Como vimos no Capítulo 1, um dos objetivos da pesquisa é inicializar uma sistematização da avaliação prática do serviço de *auto-scaling*. Ou seja, queremos gerar um novo conhecimento a partir de observações e teorias anteriores. Esse novo conhecimento se torna relevante pois traz luz sobre a forma de entendimento das configurações e da qualidade dessas configurações quando se trata de um cenário real. Portanto, consideramos que o estudo tem **natureza primária.**

Quanto aos objetivos. Para alcançar esses objetivos, duas grandes atividades foram realizadas. Inicialmente um levantamento das opiniões e impressões das pessoas que trabalham diretamente e diariamente com o mecanismo estudado, fazendo com que a pesquisa seja caracterizada, quanto aos objetivos, como **descritiva**. Posteriormente, tendo essas respostas do questionário base, foi realizada uma investigação e exploração do problema de configuração e avaliação do *auto-scaling*, onde buscamos nos dados da infraestrutura anomalias, padrões e bases para sugestões futuras de como realizar melhorias. Dessa forma, podemos considerar que esta pesquisa tem também um caráter **exploratório**, além do descritivo.

Quanto aos procedimentos técnicos. Este trabalho, de forma geral, procura entender com certa profundidade a configuração e a avaliação do *auto-scaling* em um cenário real, buscando, através de análises qualitativas e quantitativas, observar problemas e comportamentos associados à configuração do *auto-scaling*. Por esse motivo, podemos classificar o presente trabalho, quanto aos procedimentos técnicos, como um **estudo de caso exploratório**.

4.2 Questionário

Nesta seção está descrita o primeiro método de coleta de dados da pesquisa. O método de questionário foi aplicado por oferecer algumas vantagens em relação às entrevistas, por exemplo: manutenção do anonimato, obtenção de mais respostas, maior flexibilidade no

tempo de resposta e o formato padronizado para todos os respondentes. Essa coleta de dados tem como objetivo principal entender melhor o contexto da configuração e avaliação de *auto-scaling* inserido em uma empresa de comércio eletrônico. Inicialmente descrevemos como o questionário foi desenvolvido e em seguida como se deu sua análise.

4.2.1 Método

O método utilizado para definir o *survey* consiste em seguir 8 passos, indo da definição dos objetivos da pesquisa até como documentar e reportar os resultados. Os passos descritos são os seguintes [19]: **(i)** Definir os objetivos da pesquisa; **(ii)** Identificar o público alvo; **(iii)** Criar plano de amostragem; **(iv)** Definir instrumentos para realizar o *survey*; **(v)** Validar *survey*; **(vi)** Analisar Dados; **(vii)** Escrever Conclusões; **(viii)** Documentar e Reportar.

Esta seção contém uma descrição até o passo (v), que engloba tudo que diz respeito ao desenvolvimento do questionário em si. Os passos (vi), (vii) e (viii) estarão distribuídos nos próximos capítulos, onde estão a análise e as conclusões obtidas.

i **Definindo os objetivos do Survey**. Antes de iniciar o desenvolvimento do *survey*, algumas pesquisas foram realizadas em busca de possíveis pesquisas qualitativas dentro da Engenharia de Software, mais especificamente envolvendo o mecanismo de *auto-scaling*. Contudo, até o momento de início da realização do *survey*, os trabalhos encontrados que englobam o tema geralmente executam experimentação e simulação. Dessa forma, não foi possível utilizá-los como base para essa parte específica deste trabalho.

   Para definição do *survey*, foi utilizada uma abordagem *top-down*, ou seja, primeiro vem os objetivos e em seguida perguntas mais aprofundadas. O objetivo foi entender de que forma pessoas de engenharia fazem a configuração do mecanismo de *auto-scaling*, isso inclui: bases para tomar decisões de configuração, a configuração em si e de que forma avaliam as decisões. Além desse objetivo, muito possivelmente esse *survey* ajudará a entender as maiores dificuldades envolvendo o *auto-scaling*. O resultado final dessa pesquisa será usado para auxiliar times nas possíveis melhorias usando o mecanismo.

   Esse *survey* tem caráter exploratório, dado que seu objetivo foi lançar luz em uma área ainda pouco estudada na prática: a de configuração e avaliação de *auto-scaling*. O caráter

exploratório se deu justamente por não existirem trabalhos usando *surveys* para investigar aspectos do *auto-scaling*.

ii **Identificando público alvo**. A população alvo são pessoas funcionárias da empresa do estudo de caso, que trabalham na área de engenharia e já tiveram contato com o mecanismo de *auto-scaling*. A unidade de análise do *survey* consiste nas aplicações onde a adição e remoção de recursos é controlada pelo *auto-scaling*.

iii **Definindo instrumentos para Survey**. A área de engenharia da empresa possui uma quantidade de funcionários suficientemente grande para que fosse inviável utilizar entrevistas como método de *survey*, por isso foi optado usar questionário *online*. A ferramenta utilizada para coleta de dados foi o Google Forms, uma escolha arbitrária por questões de *compliance* da própria empresa. A chamada para responder a pesquisa foi feita através da ferramenta oficial de comunicação da empresa, o Slack [1]. A divulgação foi feita para a gerência de cada time, que por sua vez repassam para seus respectivos times, ficando disponível do dia 10 de junho de 2021 até o dia 20 de julho de 2021. O questionário contava com um texto de contextualização e motivação no início e 19 perguntas, sendo duas delas abertas e não obrigatórias e com tempo médio de 10 minutos para ser respondido. As 19 perguntas foram divididas em 3 seções lógicas:

   (a) **Perguntas gerais sobre aplicações.** O intuito dessas perguntas é captar o conhecimento que as pessoas possuem de características das aplicações e de que forma identificam recursos de cada aplicação na infraestrutura.

   (b) **Perguntas sobre *auto-scaling* (configuração e avaliação).** Após entender um pouco sobre a familiaridade das pessoas com as aplicações, a segunda etapa de perguntas tem o intuito de se aprofundar em como o *auto-scaling* é utilizado; como a escolha de parâmetros é feita, como são testadas as configurações e como é avaliada a eficácia de tais configurações.

   (c) **Pergunta de encerramento.** Por fim a pergunta de encerramento. Nessa última etapa foi elaborada uma única pergunta, não obrigatória, para entender quais são as maiores dificuldades de utilizar o *auto-scaling*.

[1] `https://slack.com/intl/pt-br`

iv **Validação do *Survey***. Como já citado anteriormente, a realização do *survey* tem como principal objetivo entender como as pessoas da engenharia estão configurando o *auto-scaling*. Para garantir que o questionário ajudaria a alcançar o objetivo, foi necessário realizar uma validação. A forma de validação utilizada foi o Piloto, o questionário em sua primeira versão foi enviado diretamente para 5 pessoas, que o responderam e deram algum *feedback* que serviu de base para a elaboração do questionário final. O questionário/*survey* aplicado está disponível no Apêndice A.

As pessoas voluntárias precisaram informar o e-mail de domínio da empresa para conseguir acessar o formulário por questões de segurança e privacidade, contudo esse e-mail não foi coletado, garantindo o anonimato. Um *e-mail* e um usuário do Slack, para um contato mais rápido e direto, foram disponibilizados no início do questionário para responder dúvidas ou receber *feedback*.

4.3 Coletor de Dados

A segunda forma de coleta de dados foi através de uma ferramenta construída durante a pesquisa. Dentro da empresa onde o Estudo foi realizado já existe uma coleta e monitoramento de dados, contudo algumas informações que consideramos importantes se perdem ou são agregadas num grão maior à medida que o tempo passa. Esse Coletor de Dados foi construído com o intuito de manter dados históricos num grão pequeno (1 minuto) e cruzar informações para possibilitar análises mais direcionadas e profundas sobre os serviços. O projeto e implementação dessa ferramenta está melhor detalhado no Capítulo 5.

Capítulo 5

Coletor de Dados: Projeto e Implementação

Nesse Capítulo, consideramos a ferramenta de coleta de dados que desenvolvemos. Na Seção 5.1, explicamos a motivação para desenvolver a ferramenta. Em seguida, listamos alguns requisitos não funcionais que a ferramenta deve atender. Na Seção 5.3, detalhamos o projeto e a implementação do coletor e, logo em seguida, sua configuração. Na Seção 5.6, descrevemos que dados coletamos para apoiar nosso estudo de caso. Por fim, na Seção 5.7, discutimos alguns desafios encontrados ao construir a ferramenta.

5.1 Motivação

A empresa alvo de nosso estudo de caso já usa ferramentas de coleta de dados relacionados com uso de recursos e desempenho. Os dados coletados por essas ferramentas são usados para **monitorar** a situação dos serviços em operação. Por exemplo, usando essas ferramentas, é possível verificar se uma atualização de um determinado serviço provocou alguma degradação de desempenho ou aumento excessivo no uso de recursos, e portanto, a atualização precisa ser revertida. Ainda, é possível realizar análises comparativas entre os serviços, incluindo ranquear que serviços usam a maior quantidade de recursos em um determinado período.

Apesar destes dados serem úteis para os propósitos atuais da empresa, conforme listamos, eles não são para os propósitos que planejamos. Isso porque o sistema de armazenamento

de dados aumenta a granularidade dos dados com o passar do tempo; fazendo com que os dados fiquem cada vez mais resumidos. Isso impossibilitaria, por exemplo, realizar análises que necessitam de um grão de tempo menor, como é o caso do tempo de "rampa"(tempo que a máquina virtual leva até ficar em um estado funcional). Além disso, o atual sistema de armazenamento e consulta não mantém dados individualizados para cada máquina virtual que foi alocada, mas somente um agregado do conjunto de máquinas usadas por cada serviço, isso pode ser um problema em análise nas quais precisamos saber a quantidade de máquinas utilizadas e o comportamento de cada máquina individualmente para identificar problemas de balanceamento, por exemplo.

5.2 Requisitos

Nesta Seção listamos os requisitos obrigatórios que a ferramenta precisou seguir durante a sua construção. Consideramos os requisitos de **custo**, **facilidade de configuração**, **extensibilidade** para uso em outras nuvens computacionais e **confiabilidade**. A seguir, discutimos sobre esses aspectos.

Custo. Para coletar todas as informações necessárias, para uma análise detalhada da infraestrutura, a nossa ferramenta precisou utilizar recursos computacionais e acessar alguns serviços da nuvem. Em ambos os casos, há custos envolvidos. Foi importante ter em mente estes custos durante o projeto da ferramenta (inclusive, para justificar seu uso). Abaixo, resumimos os custos envolvidos:

- Custo de chamadas à API para recuperar informações de cada recurso (IDs, tags, etc);
- Custo de chamadas para recuperar métricas e estatísticas [1];
- Custo para manter o coletor rodando (servidor) [2];
- Custo com serviço de armazenamento [3].

Para exemplificar o custo descrito acima, vamos usar um cenário hipotético onde são feitas coletas de duas métricas diferentes, com a frequência de uma hora, em uma infraestrutura

[1] `https://aws.amazon.com/pt/cloudwatch/pricing/`
[2] `https://aws.amazon.com/pt/ec2/instance-types/t3/`
[3] `https://aws.amazon.com/pt/s3/pricing/`

que em média possui 1000 máquinas virtuais em uso todos os dias. Ou seja:

- A cada hora teríamos 2000 chamadas à API, pois existem 1000 máquinas e estamos coletando duas métricas. Então o custo por hora para recuperar as métricas é de US$ 0,02, o que resulta num custo de aproximadamente US$ 15/mês. A fórmula (Quantidade de Métricas x 0,02)/1000 nos dá o custo por coleta, o custo por mês depende da frequência da coleta.

- A cada hora, após a coleta, é gerado um volume de aproximadamente 70 MB. Como o custo de armazenamento é de US$ 0.023 por cada GB, a cada hora geraríamos um custo de US$ 0.00161, resultando em um custo de aproximadamente US$ 2/mês. A fórmula 0,07 GB x 0.023 nos dá o custo de armazenamento por coleta, o custo por mês também depende da frequência de coleta.

- Já o custo do servidor, onde este serviço de coleta irá executar, é fixo e independe da quantidade de métricas e máquinas virtuais. Verificamos que uma instância do tipo t2.large [4] consegue executar com uma boa performance as operações do coletor. O custo desse tipo de instância é de US$ 0,0928 por hora, gerando um custo de aproximadamente US$ 70/mês.

- O custo total mensal da coleta exemplo seria US$ 87/mês.

Configuração. A ferramenta construída deve ser de fácil configuração. Neste requisito, consideramos dois aspectos de configuração. O primeiro é relacionado com a configuração do ambiente de execução da ferramenta. O segundo aspecto é relacionado à configuração da ferramenta em si, após sua implantação no ambiente de execução. Para atender o primeiro aspecto existe uma automação, um *script* que configura o servidor, uma máquina virtual com todas as dependências necessárias. Já para atender o segundo aspecto o usuário pode tanto executar um *script* dentro do servidor e seguir a configuração via terminal, mas também pode optar por alterar diretamente o arquivo de configuração, como está melhor descrito na Seção 5.5. De forma resumida, o usuário da ferramenta precisa indicar somente quais as métricas que deseja coletar, quais as respectivas estatísticas e a frequência da coleta. Por exemplo, se o

[4] `https://aws.amazon.com/pt/ec2/instance-types/t2/`

usuário pretende coletar a métrica Utilização de CPU, deve indicar também quais estatísticas (soma, média, percentil 90, etc) deseja daquela métrica.

Uso com diferentes nuvens. A empresa considerada no estudo de caso usa um único provedor de nuvem. Entretanto, na escala desta empresa, não é raro o uso de múltiplos provedores. Considerando que este pode ser um cenário futuro, a ferramenta deve ser adaptável para coletar dados de outros provedores. Para atender esse requisito, o código da ferramenta que interage com serviços específicos do provedor de nuvem são organizados em módulos para reduzir o acoplamento, assim permitindo uma fácil substituição de tais módulos para operar com outros provedores.

Confiabilidade na coleta. Um requisito desejável é que todos os dados de interesse sejam efetivamente coletados. Consideramos duas abordagens para atender este requisito. A primeira envolveria manter uma alta disponibilidade do serviço de coleta. Por sua vez, a segunda envolveria a re-execução parcial de coleta de dados que não tenham sido coletados durante um período de indisponibilidade da ferramenta. Uma vez que o serviço do provedor de nuvem retém dados por um período suficientemente longo, adotamos a segunda abordagem. Complementamos essa abordagem com procedimentos para monitoramento do serviço de coleta. Desse modo, o usuário recebe alertas em caso de falha no serviço, para em seguida retomar o seu funcionamento e realizar a coleta dos dados em falta.

5.3 Projeto do Coletor

O projeto do coletor considera cinco componentes, representados na Figura 5.1. Destes cinco componentes consideramos dois serviços externos: o **Repositório** de métricas e o serviço de **Armazenamento**. Estes são serviços oferecidos pelo próprio provedor de nuvem para o qual nossa ferramenta coleta dados. O **Repositório** permite coletar métricas sobre os recursos fornecidos pelo provedor (p.ex métricas sobre máquinas virtuais). O serviço de **Armazenamento** provê confiabilidade na retenção de dados, bem como diferentes escolhas da qualidade de serviço e custo.

Como o serviço **Repositório** tem um período de retenção curto para o tipo de análise considerada neste estudo de caso, nossa ferramenta considera um componente **Coletor** que, periodicamente, se comunica com o serviço **Repositório** para que os dados sejam enviados

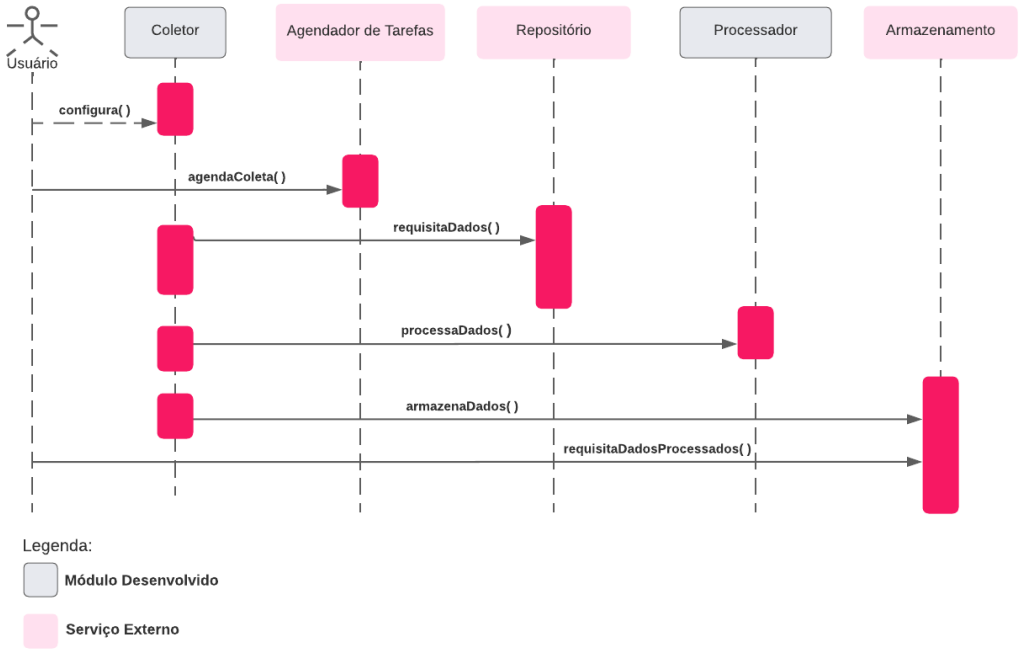

Figura 5.1: Diagrama de sequência do coletor.

ao serviço de **Armazenamento**.

Entre a coleta e o armazenamento permanente, os dados são armazenados temporariamente e transformados pelo componente **Processador**. Entre outras transformações, este componente converte os dados coletados para um formato que facilite a análise dos dados (o formato *CSV*) e os organiza em uma estrutura de diretórios que considera, por exemplo, agrupamentos por data e recursos monitorados.

Por fim, o módulo **Agendador de Tarefas** é responsável por executar periodicamente os módulos de coleta e armazenamento. A frequência com que o agendador executa esses módulos é configurada pelo usuário.

5.4 Implementação do Coletor

Dos cinco componentes citados na Seção anterior, temos dois desenvolvidos durante a pesquisa, o Coletor e o Processador. Todos os componentes foram desenvolvidos utilizando a linguagem de programação Python. A motivação para usar essa linguagem foi seu conhecimento prévio, a facilidade para manipular dados usando suas bibliotecas (pandas, numpy,

etc), e por último o fato de que já existe um *SDK* (Software Development Kit) Python que facilita a comunicação do código com as APIs dos serviços externos utilizados.

O Coletor e o Processador se comunicam com o **Repositório** e o **Armazenamento** respectivamente. Aqui, o serviço que representa o repositório de métrica é o CloudWatch e o serviço que representa o armazenamento é o S3, ambos da AWS. O acesso a esses serviços é realizado através de um *SDK* Python, chamado Boto3, [5] utilizado para criar, configurar e gerenciar serviços AWS.

Já o Agendador de Tarefas é um script Python que se comunica com o serviço Cron [6], que permite ao usuário agendar tarefas. O *script* vai agendar uma tarefa de acordo com a frequência de coleta configurada. Essa tarefa consiste em executar o código do coletor.

5.5 Fluxo e Configuração do Coletor

A partir do diagrama de sequência representado na Figura 5.1, podemos especificar as ações de cada componente e o fluxo de atividades. Os passos listados a seguir englobam todo o processo do Coletor, desde a configuração até o armazenamento dos dados coletados. Entretanto, o usuário só executará alguma ação no passo 1, onde deve especificar o que será coletado e com que frequência.

1. **Ação do usuário.** O usuário deve alterar o arquivo de configuração, um exemplo desse arquivo pode ser visto no Apêndice C, adicionando as seguintes informações para cada métrica que deseja coletar: o nome da métrica, a dimensão [7] (identificador único da métrica, por exemplo: *InstanceId* ou *LoadBalancerName*), e o nome das estatísticas (ex: média, máx, etc). Também deve ser descrita a granularidade da coleta em segundos, o nome do diretório onde os arquivos de coleta serão salvos de acordo com a dimensão e, por último, as informações específicas relacionadas ao serviço de armazenamento utilizado. No caso da AWS: a região de coleta e o nome do Bucket[8]

[5] `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`

[6] `https://www.redhat.com/sysadmin/automate-linux-tasks-cron`

[7] `https://docs.aws.amazon.com/pt_br/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html#Dimension`

[8] Bucket é uma definição do serviço S3 para identificar um conjunto de objetos armazenados. Similar a um diretório.

S3 para onde os dados serão encaminhados.

2. Após a edição do arquivo de configuração, o Agendador de Tarefas será utilizado para criação de uma nova tarefa que vai ficar responsável por executar o Coletor de acordo com a frequência escolhida pelo usuário. O Agendador de Tarefas também vai criar uma tarefa para executar o código responsável por enviar os dados para o serviço de Armazenamento. O *script* especificado no Agendador de Tarefas executa o Coletor, que por sua vez irá recuperar as métricas.

3. A recuperação de métricas é responsabilidade do Coletor, que executa as seguintes ações: Recupera os valores das dimensões via chamada à API do serviço daquele recurso (ex: recupera IDs das instâncias através da API do serviço de instâncias); Percorre os valores das dimensões e, para cada um, recupera as estatísticas da métrica que foram especificadas no arquivo de configuração.

4. Após essa coleta, o Coletor envia os dados para o módulo de processamento de dados. O módulo que processa os dados após a coleta recebe um formato JSON e o transforma em um CSV, com um formato definido na implementação e melhor descrito na Subseção de Dados Coletados. Após a formatação, os dados ficam armazenados localmente e depois são enviados para um serviço de armazenamento.

5. O *script* especificado no Agendador de Tarefas executa o código que se comunica com o serviço de Armazenamento, enviando os dados para serem armazenados. Este passo é, geralmente, executado numa frequência menor.

5.6 Dados Coletados

Para apoiar o estudo de caso descrito neste trabalho, nossa ferramenta coletou dados de três diferentes tipos de recursos: instâncias de máquinas virtuais, grupos de *auto-scaling* e balanceadores de carga.

Os dados relacionados às instâncias permitem identificar cenários de super e sub-provisionamento, localizar padrões temporais de utilização das mesmas, analisar como se dá o uso de cada tipo de instância e, de acordo com a aplicação, entender se determinados níveis de utilização fazem sentido.

As informações sobre os grupos de *auto-scaling* são necessárias para agrupar as instâncias usadas por cada aplicação da empresa. Note que as regras de *auto-scaling*, como já citado anteriormente, são definidas para cada aplicação. A forma para recuperar essas informações pode variar de acordo com a plataforma utilizada. Nesse caso, recuperamos via arquivos de configuração de cada aplicação.

Por último, é importante também analisar a carga à qual a aplicação está submetida. Para isso são necessárias informações dos balanceadores de carga. Com essa informação, por exemplo, podemos analisar quais são os padrões temporais das requisições que chegam na aplicação monitorada, qual a latência destas requisições e verificar se a aplicação degradou, através dos respectivos SLOs, que envolvem a latência. Assim como os grupos de *auto-scaling*, os balanceadores de carga também estão atrelados a uma aplicação.

Todos os dados foram coletados em um grão de um minuto. Os detalhes sobre quais dados foram coletados estão descritos a seguir em tabelas específicas para cada recurso. A Tabela 5.1 lista os dados coletados para instâncias de máquinas virtuais. A coleta que realizamos, a qual será descrita nos próximos Capítulos, considera somente a métrica Utilização de CPU. É possível que métricas de uso de memória sejam consideradas no futuro, a depender ainda de autorização da empresa[9]. As métricas coletadas para os balanceadores de carga foram a Contagem de Requisições e Latência. Os dados para este tipo de recursos estão descritos na Tabela 5.2. A Tabela 5.3 lista os dados da coleta para os grupos de *auto-scaling*. Coletamos seus identificadores e as respectivas máquinas que formam o grupo no momento da coleta.

5.7 Desafios

A tarefa de implementar nossa ferramenta de coleta apresentou alguns desafios. O primeiro desafio que encontramos dizia respeito à inadequação do serviço **Repositório** disponível na AWS, o CloudWatch, para o propósito da nossa coleta. Em particular, este serviço é incompleto. Ou seja, algumas informações não estão disponíveis no CloudWatch. Nesse caso, foi necessário consultar outros serviços para combinar as informações desejadas. Por

[9]No momento do desenvolvimento do coletor a empresa não autorizou a coleta de informações de memória pois o custo seria elevado.

Dado	Descrição
Timestamp	Data e horário da medição
Metric	Nome da métrica coletada
InstanceId	Identificador único da instância da qual a métrica foi coletada
InstanceType	Tipo da instância.[10]
Average	Valor médio da métrica agregado de acordo com a granularidade
Max	Valor máximo da métrica agregado de acordo com a granularidade
Min	Valor mínimo da métrica agregado de acordo com a granularidade

Tabela 5.1: Dados coletados para instâncias de máquinas virtuais.

Dado	Descrição
Timestamp	Data e horário da medição
Metric	Nome da métrica coletada
LoadBalancerName	Identificador do load balancer
Average	Métrica agregada pela média de acordo com a granularidade
Max	Métrica agregada pelo máximo de acordo com a granularidade
Min	Métrica agregada pelo mínimo de acordo com a granularidade

Tabela 5.2: Dados coletados para balanceadores de carga.

exemplo, para recuperar a utilização de CPU de uma instância específica, além da informação de Product, ApplicationName e o grupo de *auto-scaling* do qual a instância faz parte, é

Dado	Descrição
Timestamp	Data e horário da medição
InstanceID	Identificador da instância
AutoscalingGroupName	Identificador do grupo de *auto-scaling*
ApplicationName	Rótulo associado a uma instância que indica a qual aplicação a mesma está servindo
Product	Rótulo associado a uma instância que deve indicar a que produto a mesma está servindo

Tabela 5.3: Dados coletados para os grupos de *auto-scaling*.

necessário realizar chamadas a duas APIs dos serviços EC2 [11] e EC2 AutoScaling [12]. Essa necessidade adicionou complexidade não prevista, tanto na implementação do serviço de coleta, quanto no posterior processamento de dados e gerou outros desafios.

Um segundo desafio que encontramos envolvia o **ciclo de vida das instâncias e efemeridade de informações.** Como dissemos antes, o **Repositório** CloudWatch da AWS não era completo para o nosso propósito e os serviços adicionais que precisamos usar tinham pouco tempo de retenção para as informações de instâncias após elas terem sido removidas. Por exemplo, ao fim do ciclo de vida de uma instância, tipicamente, as informações sobre esta instância ficam indisponíveis após 30 minutos. Isso implica em risco de perda de dados caso a ferramenta seja configurada incorretamente (p.ex realizado coletas em períodos grandes).

Por fim, um último desafio que tivemos foi sobre como verificar se nossa coleta está correta. Por exemplo, como poderíamos verificar se foram coletados todos os dados disponíveis em determinado momento, uma vez que não temos uma outra fonte de informação com a qual comparar. A abordagem que adotamos para esse problema foi comparar nossos dados com o sumário de dados mantido pelo CloudWatch. Neste sumário, os dados são apresentados e mantidos de maneira agregada. Ou seja, o sumário do CloudWatch permite observar uma determinada métrica, digamos Utilização de CPU, como a média da utilização

[11] `https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html`

[12] `https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/autoscaling.html`

para todas instâncias de um grupo de *auto-scaling*. Em nossas verificações, agregamos os dados individuais obtidos com o nosso coletor e comparamos com os dados sumarizados do CloudWatch, com isso, verificamos a corretude da coleta.

Capítulo 6

Conhecendo os Dados

Neste Capítulo, realizamos uma análise descritiva dos dados coletados. Esta coleta abrange aproximadamente um ano. Apesar disso, em nossa análise consideramos parte dos dados coletados, apenas um mês que não possui tantos eventos extraordinários. O fragmento de um mês foi utilizado pois é suficientemente grande para os propósitos da pesquisa, sendo o total coletado um volume alto de dados, utilizá-los por completo agregaria mais complexidade às análises. Este mesmo fragmento foi considerado nas análises dos Capítulos posteriores. De forma resumida, identificamos milhares de diferentes instâncias utilizadas ao longo de um mês, algumas centenas de balanceadores de carga e pouco mais de uma centena de diferentes Produtos e Aplicações.

6.1 Demanda

Para ter uma visão geral da demanda que chega nas aplicações podemos observar a Figura 6.1. Nesta figura foram agregadas todas as demandas de todas as aplicações monitoradas ao longo do período de observação considerado. Os valores apresentados foram normalizados. O primeiro ponto importante a ressaltar é que a demanda é muito grande e varia bastante ao longo do tempo. Considerando períodos de 5 minutos, a menor quantidade observada foi na faixa de milhões, a maior também, porém aproximadamente 8 vezes a quantidade mínima observada. Esse número é medido através dos balanceadores de carga usados por cada aplicação, ou seja é a soma de requisições que chega em cada uma das aplicações. Devemos atentar para o fato de que essas quantidades não representam as requisições inicialmente sub-

metidas pelos clientes para a plataforma, pois uma única requisição pode ser contabilizada mais de uma vez devido ao possível fluxo que pode fazer entre os serviços internos.

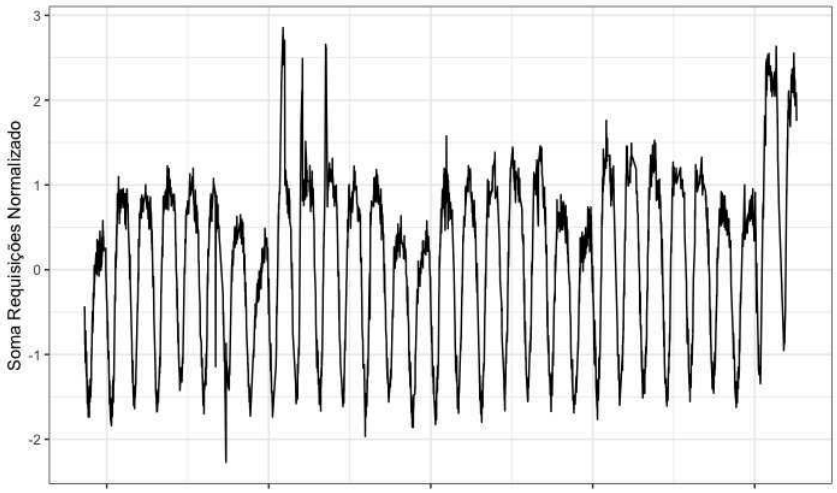

Figura 6.1: Quantidade de requisições ao longo do mês.

6.2 Tamanho da Infraestrutura

Como já foi visto no Capítulo 3, cada Produto pode conter mais de uma aplicação. No intervalo do mês observado, cada produto teve associado a si em média três aplicações. Observando o menor agrupamento lógico da divisão, a versão de uma aplicação, contabilizamos 577 versões diferentes durante o mês analisado. Cada Aplicação assumiu pelo menos duas versões durante esse tempo, como indica a Figura 6.2. Cada ponto relaciona uma aplicação e a quantidade de versões da mesma. Observamos que, tipicamente, as aplicações assumem sete versões. Contudo podemos ver que algumas Aplicações podem chegar a ter até 24 versões diferentes ao longo de um mês. As motivações das mudanças de versões não são conhecidas e podem ser correções de *bugs*, novas funcionalidades ou quaisquer outras mudanças no código.

Cada versão conta com diferentes instâncias para atender suas demandas. Ou seja, sempre que uma nova versão é gerada, as máquinas que atendem a versão atual serão removidas e outras novas serão adicionadas. Para ter uma visão geral, calculamos em média quantas instâncias cada versão da aplicação usa. Tanto a média de máquinas, quanto a quantidade

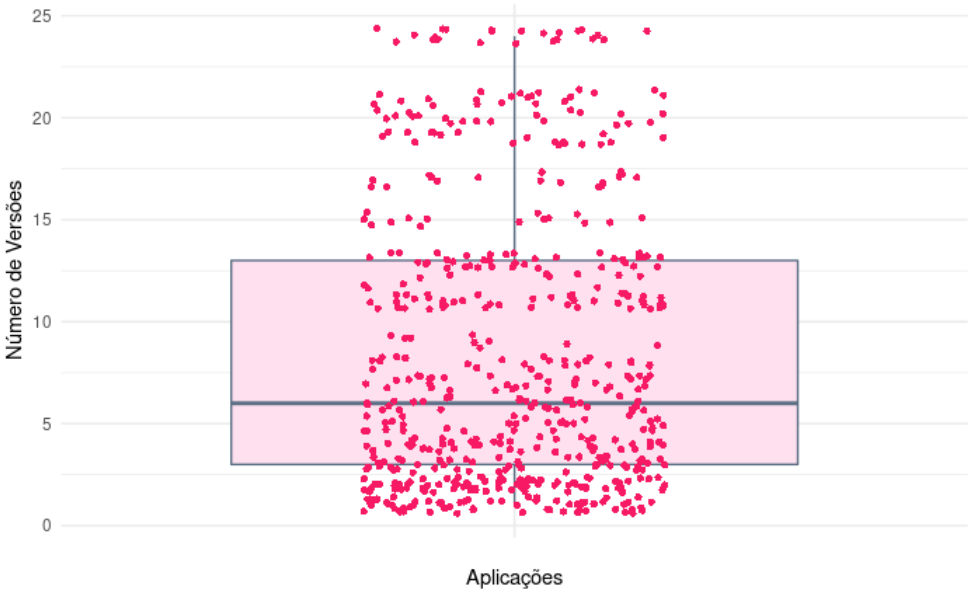

Figura 6.2: Quantidade de versões de cada Aplicação.

de versões mostram como a carga geral é heterogênea, como mostra a Figura 6.3. Os dados da Figura estão normalizados, mas o intuito nesse caso é observar o quanto as versões da aplicações são diferentes entre si e entre as outras aplicações.

Nesse mesmo sentido, podemos também observar como as Aplicações se diferenciam quanto à demanda. Na Seção 6.1, mostramos a demanda geral de todos as Aplicações observadas ao longo do tempo. Agora, podemos ver na Figura 6.4 como está variando a quantidade de requisições que chega para cada versão existente durante nosso período de observação. Essa Figura está numa escala logarítmica para facilitar sua visualização, além de estar normalizada. As aplicações que mais recebem maior carga podem chegar a atender um número de requisições que é 6 ordens de grandeza maior do que o número de requisições recebidas pelas menores aplicações.

Por fim, consideramos o uso dos recursos alocados para essas aplicações. A Figura 6.5 mostra um histograma das médias de Utilização de CPU em intervalos de 5 minutos. Podemos ver que há uma grande concentração em intervalos de baixa utilização de CPU.

Em resumo, entendemos que, durante o mês analisado, o sistema é dinâmico: há muitas aplicações e mudam com bastante frequência. Isso pode implicar em termos decisões sobre *auto-scaling* que perdem a qualidade ao longo do tempo (uma vez que as aplicações mudam).

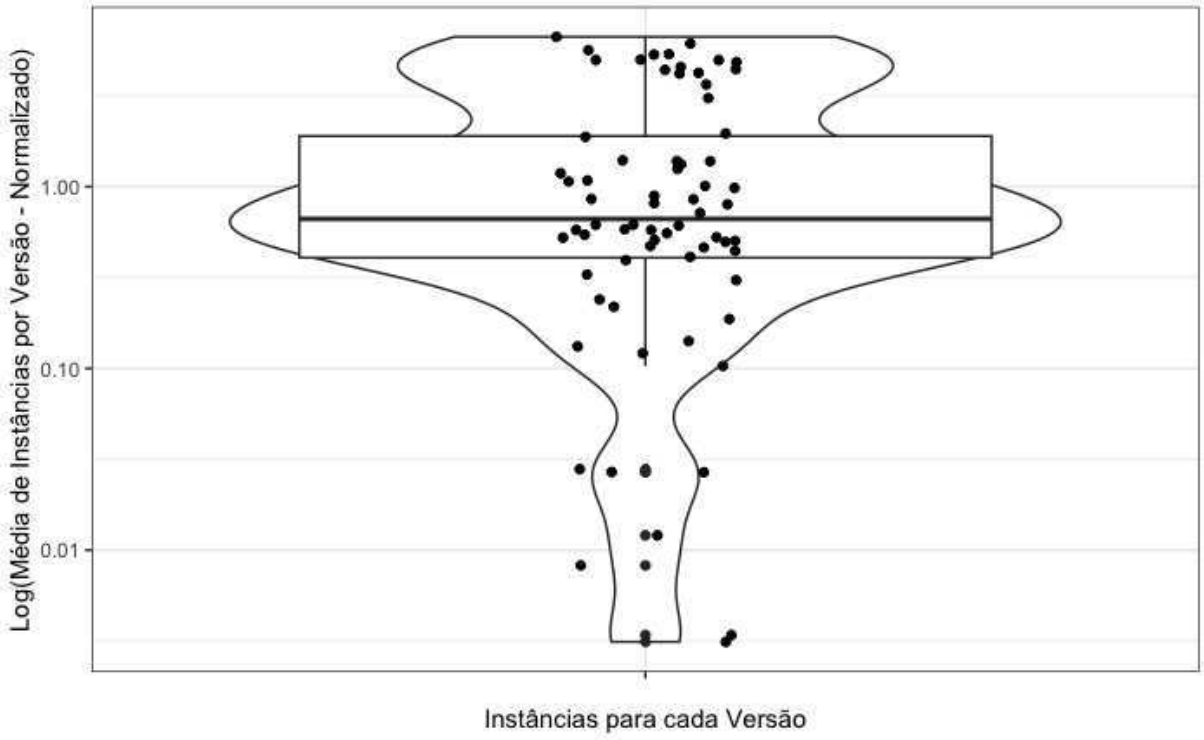

Figura 6.3: Média de instâncias para cada versão.

Ainda, podemos observar que algumas aplicações podem ser mais importantes que outras, tanto em termos de processarem uma parte maior da demanda total quanto usarem mais recursos que as demais, ou seja, melhorias em um conjunto pequeno de aplicações podem ter um grande impacto na empresa. Por último, também vimos que as aplicações parecem trabalhar em um baixo regime de utilização de CPU. Embora possam ser níveis corretos para manter um desempenho adequado, valores tão baixos podem também indicar super-provisionamento de recursos e oportunidades para sugestões de mudanças de *auto-scaling*.

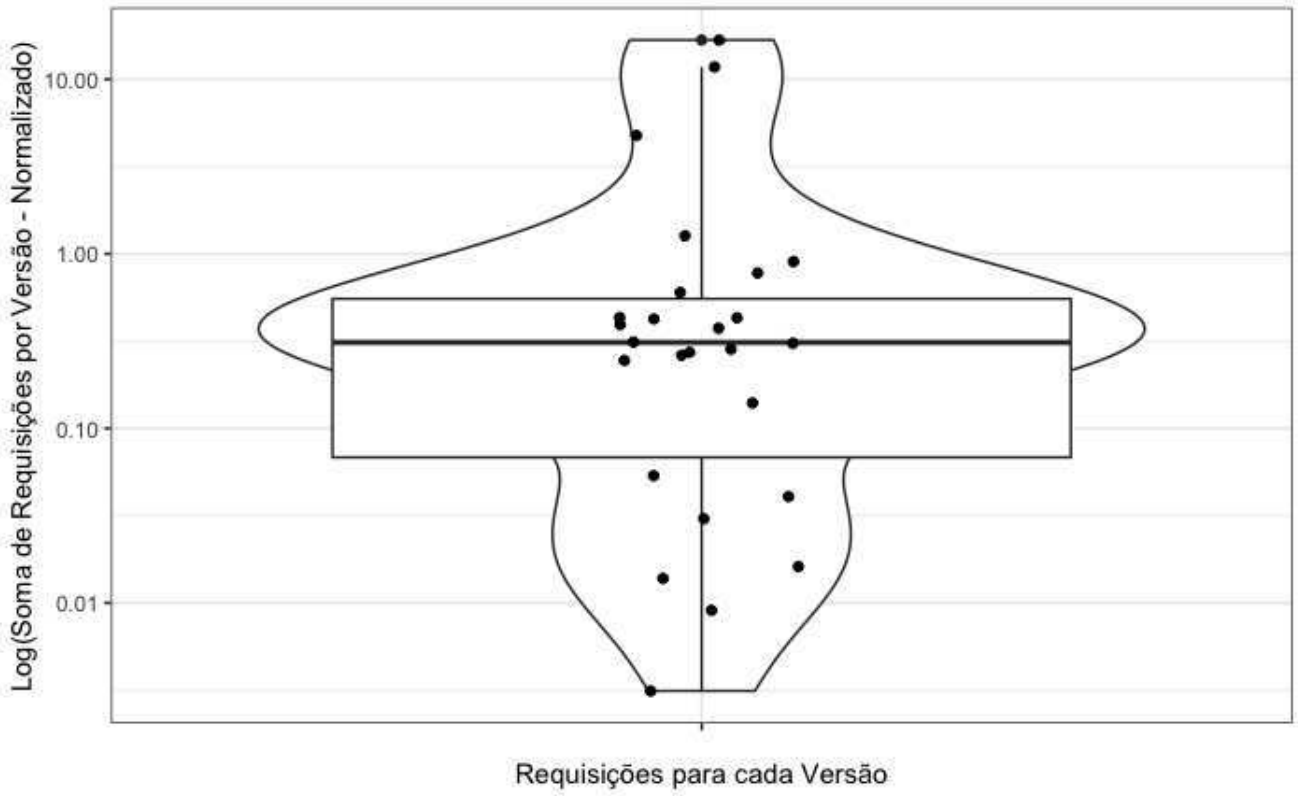

Figura 6.4: Total de requisições para cada versão.

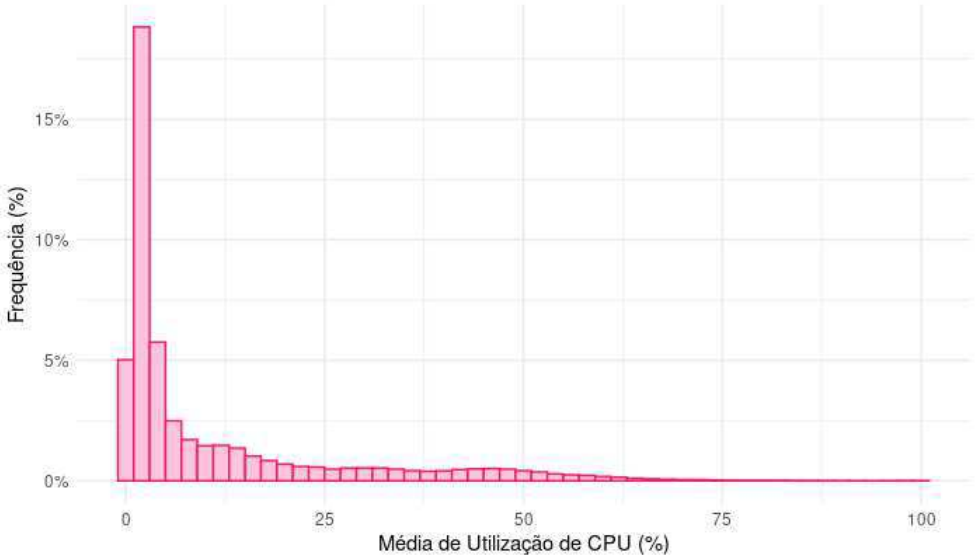

Figura 6.5: Distribuição da média de utilização de CPU.

Capítulo 7

Questionário

Neste Capítulo descrevemos motivação, aplicação e resultados da coleta de dados realizada através de um questionário/*survey*.

7.1 Objetivo

Os principais objetivos do questionário foram entender como as pessoas configuram e avaliam o *auto-scaling* e quais as suas maiores dificuldades nessas duas atividades. Para entender como a configuração e a avaliação são realizadas, é necessário saber o quão familiarizadas as pessoas estão com as aplicações, quais parâmetros acham que mais influenciam na eficácia do *auto-scaling*, qual a métrica que melhor descreve a carga que chega à aplicação, em que são baseadas as escolhas dos parâmetros, etc. Todos os objetivos listados consideram o contexto de uma empresa de médio porte em crescimento. Por se tratar de um cenário real, estabelecemos também um objetivo prático, que consiste em usar todo o conhecimento adquirido no questionário para propor soluções factíveis, que auxiliem na configuração, avaliação e em outros tópicos listados como dificuldades nas respostas obtidas.

7.2 Análise do Questionário

A análise dessa pesquisa foi realizada em duas etapas. Na primeira etapa foram avaliadas as respostas obtidas via questionário. Na segunda etapa foram analisados dados de monitoramento da infraestrutura, especificamente de aplicações que são citadas nessa primeira etapa.

O objetivo da segunda etapa é avaliar, a partir dos dados de monitoramento, a qualidade das configurações de *auto-scaling*. Essa segunda etapa está descrita no Capítulo 9.

Nesta primeira etapa, a análise do questionário busca compreender alguns aspectos do processo de configuração e avaliação do *auto-scaling*, como por exemplo o conhecimento das pessoas sobre as aplicações e suas cargas de trabalho, quais métricas consideram importantes, quais parâmetros acham que mais influencia na eficiência do *auto-scaling*, que métricas utilizam como parâmetro para desencadear ações de *auto-scaling* e quais as maiores dificuldades nessas duas atividades, configuração e avaliação.

No total foram recebidas 23 respostas. Destas, 2 foram descartadas por estarem associadas a aplicações cujas infraestruturas não estão sendo monitoradas ou não estão em ambiente de produção e, portanto, não poderiam participar da segunda etapa da avaliação. Sendo assim, foram analisadas 21 respostas. Estas 21 respostas válidas conseguem abranger o equivalente a aproximadamente 40% dos times existentes. Cada um dos times pode ser responsável por mais de um produto, assim nas respostas aparecem 17 produtos diferentes. As 20 respostas obtidas possuem informações sobre:

Padrão de Carga. É importante entender como as cargas de trabalho se comportam para que cada time possua uma ideia melhor de quais aplicações podem se beneficiar do *auto-scaling* e como podem fazer isso. Para capturar, em algum nível, a familiaridade dos times com as aplicações foram adicionadas algumas perguntas sobre padrão de carga. Três das 20 respostas afirmam que sua aplicação não apresenta padrão temporal na carga. Dos 17 voluntários que responderam que existe um padrão, a grande maioria indicou que existem um padrão ao longo das horas do dia e 6 apresentam mais de um padrão temporal na carga. Essa distribuição pode ser visualizada na Tabela 7.1.

Padrão	Quantidade
Ao longo das horas	13
Ao longo das horas e dias da semana	4
Ao longo das horas e dias do mês	2
Não apresenta padrão	2

Tabela 7.1: Respostas sobre padrão de carga das aplicações.

Esse resultado pode ser um indicativo de que geralmente existe um conhecimento sobre a demanda dos serviços, o que é um ponto importante para configurar de forma adequada o *auto-scaling*. Além disso, também foi observado que duas pessoas afirmaram que o seu serviço não apresenta padrão na carga, o que pode deixar a configuração mais complexa.

Métricas Importantes. Em geral, o serviço de *auto-scaling* toma decisões autônomas com base em monitoramento de certas métricas do serviço/aplicação sendo escalado. Assim, é importante entender sobre o conhecimento dos respondentes em relação às métricas que refletem bem a situação de carga das aplicações que eles gerenciam. Para tal, foram analisadas as respostas que indicam quais métricas são utilizadas para mensurar a "saúde" dos serviços. Os respondentes podiam escolher mais de uma métrica. Dentre as respostas aparecem as seguintes métricas:

- *Error Rate*: Representa a porcentagem de requisições com status 5xx, sendo citada por 80% dos participantes.
- *Response Time*: Tempo total entre uma requisição de serviço e a resposta para essa requisição, citada por 76% dos participantes.
- *CPU Utilization*: Porcentagem da média de utilização de CPU das máquinas virtuais, encontrada nas respostas de 61% dos participantes.
- *Throughput*: Citada por 41% dos participantes, indica a quantidade de requisições processadas em determinado período de tempo.
- *Database latency*: Tempo que leva para enviar, executar e receber uma resposta para uma consulta mínima ao banco de dados, citada por apenas 1 dos participantes.

É possível visualizar a frequência com que cada uma destas métricas aparece a partir da Figura 7.1.

A partir desses dados, podemos considerar *Error Rate* e *Response Time* como as métricas de maior importância. Erros do tipo 5xx, que aqui representam a métrica *Error Rate*, indicam problemas no servidor, que podem estar relacionados ou não à insuficiência de recursos naquele determinado momento. No contexto de *auto-scaling*, é difícil correlacionar essas duas métricas críticas citadas com a quantidade de recursos. Por isso, geralmente o seu propósito é monitorar outros aspectos relacionados à aplicação que não o *auto-scaling*.

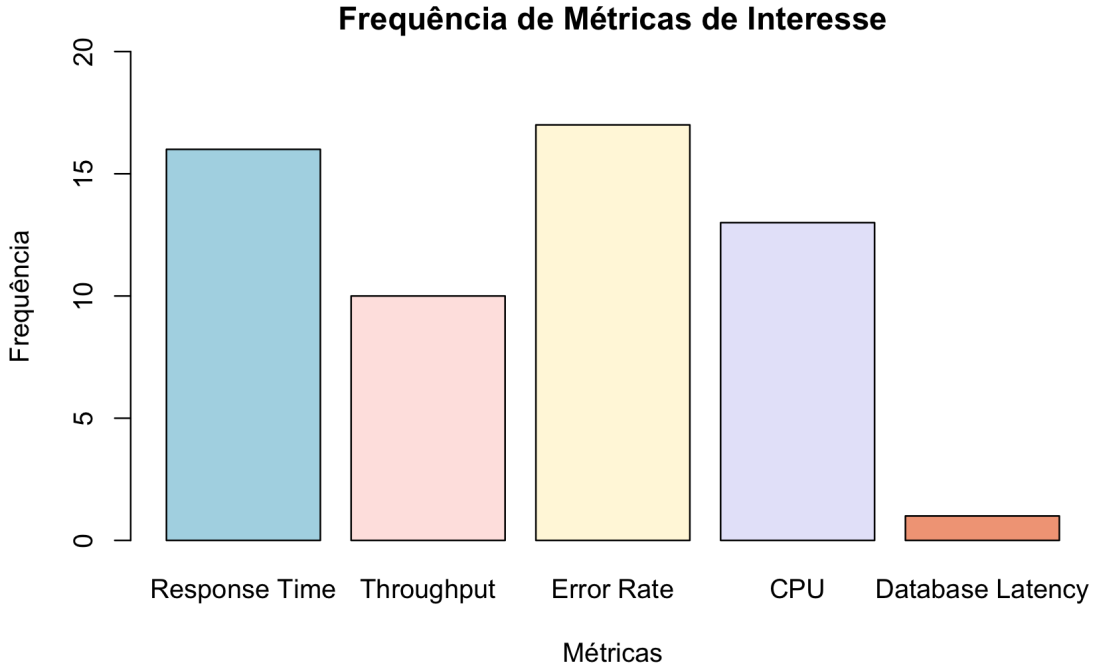

Figura 7.1: Frequência de métricas de interesse nas respostas.

Métricas do *Auto-Scaling*. Apesar de *Error Rate* ser a métrica crítica para quase todas as aplicações do questionário, ela não é usada para desencadear as ações de *auto-scaling*, pois não há uma função clara que relacione esta métrica com a quantidade de recursos ideal para a aplicação. Além disso, as métricas de interesse mais citadas geralmente servem para mostrar problemas mais recorrentes e imprevisíveis que o problema de sub-provisionamento. Portanto, das métricas citadas, é mais comum que a utilização de CPU seja usada para a configuração das regras de *auto-scaling* e isso pôde ser confirmado neste questionário. Quando questionados sobre que métrica utilizavam para controlar o *auto-scaling*, 20 indicaram *CPU-Utilization* e 1 indicou *ConsumedReadCapacityUnits*.

- ***CPUUtilization***: A porcentagem de unidades de computação alocadas que estão atualmente em uso na instância. Esta métrica identifica a quantidade de processamento da instância que está em uso no momento. Para instâncias dedicadas a rodar uma única aplicação, essa informação pode indicar a capacidade de processamento necessária para executar a aplicação na instância selecionada no momento da coleta.

- ***ConsumedReadCapacityUnits*:** O número de unidades de capacidade de leitura consu-

midas ao longo do período de tempo especificado, para que seja possível acompanhar quanto do *throughput* provisionado foi usado.

Na Figura 7.2 podemos observar a relação entre considerar *CPUUtilization* como métrica importante para saúde da aplicação e utilizar *CPUUtilization* como métrica de fato usada para controlar o serviço de *auto-scaling*.

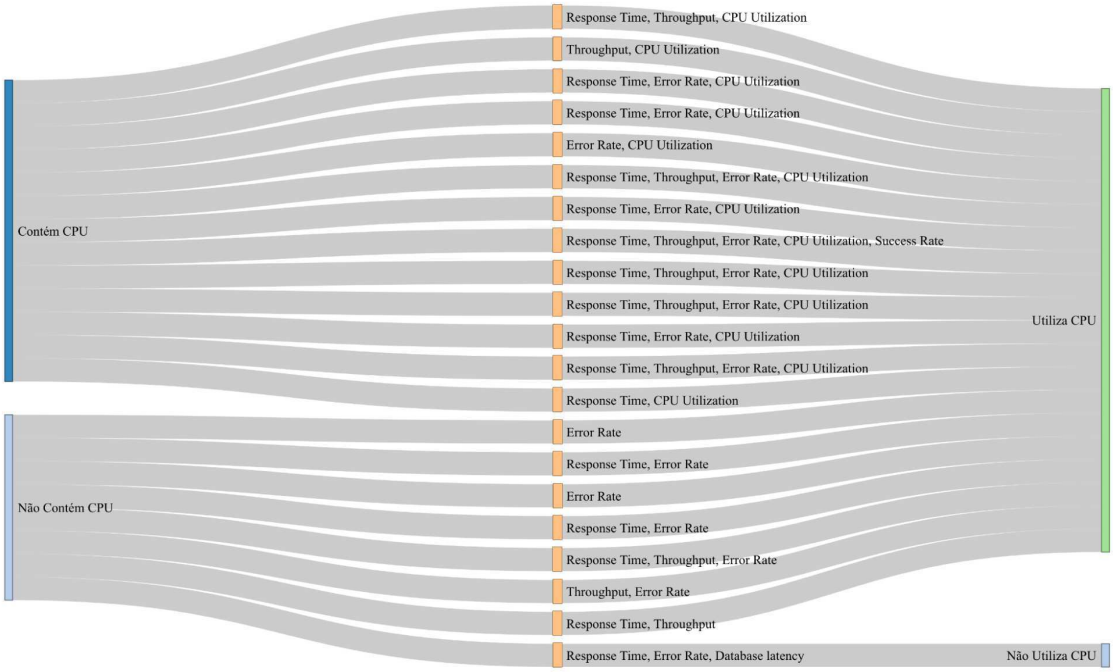

Figura 7.2: Relação de métricas consideradas importantes com métricas utilizadas para controlar o *auto-scaling*.

Na primeira coordenada da Figura 7.2 foram agregadas as respostas em dois grupos: As que citam CPU como métrica importante e as que não citam. Já na segunda coordenada foram instanciadas que respostas foram essas, explicitando todas as outras métricas contidas na resposta além da CPU. Por fim, foram agregadas novamente em dois grupos: As respostas que usam Utilização de CPU para desencadear ações de *auto-scaling* e as que não usam. A partir da Figura 7.2 podemos observar a divergência entre métrica de interesse e métrica usada no *auto-scaling*, o que é esperado, já que não existe uma função que relacione diretamente o uso de recursos, nesse cenário, além da utilização de CPU.

Outros parâmetros de configuração. Além da métrica escolhida para desencadear

ações de *auto-scaling*, o funcionamento desse mecanismo também é influenciado por outros parâmetros, que muitas vezes são escolhidos de forma intuitiva. Este questionário busca entender também quais parâmetros os respondentes acham que mais influenciam na configuração do serviço de *auto-scaling*. A frequência com que esses parâmetros aparecem podem ser vistos na Figura 7.3, bem como cada combinação escolhida pode ser visualizada na Tabela 7.2.

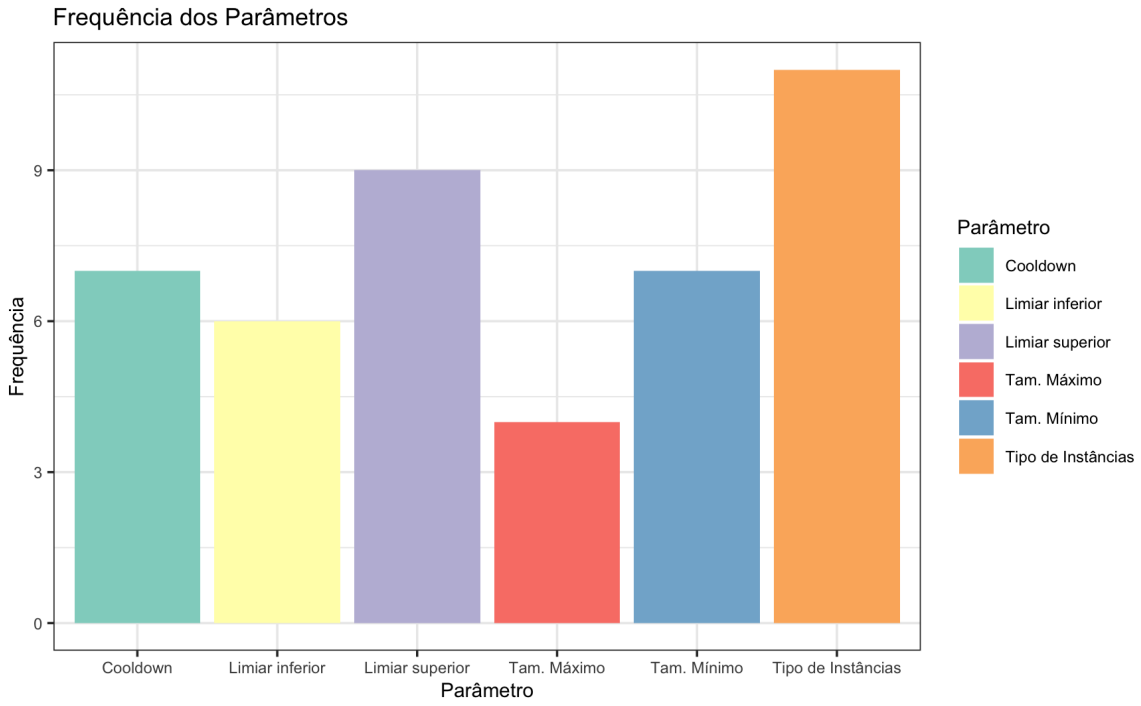

Figura 7.3: Frequência de parâmetros citados nas respostas.

- *Cooldown*: Tempo de espera entre o alcance de um limiar e o disparo de uma ação de *auto-scaling*.
- Limiar inferior: Valor mínimo que a métrica pode atingir antes de uma ação de *auto-scaling* ser disparada.
- Limiar superior: Valor máximo que a métrica pode atingir antes de uma ação de *auto-scaling* ser disparada.
- Tamanho Máximo: Quantidade máxima de instâncias que um grupo de *auto-scaling* pode alcançar.

Conjunto de Parâmetros	**Ocorrência**
Cooldown	1
Tipos de Instância	2
Tipos de Instância, Limiar superior	2
Tipos de Instância, Cooldown	1
Tipos de Instância, Limiar superior, Cooldown	1
Tipos de Instância, Limiar inferior, Cooldown	1
Tipos de Instância, Limiar superior, Limiar inferior	1
Limiar superior, Tamanho máximo do grupo de auto-scaling, Tamanho mínimo do grupo de auto-scaling	1
Tipos de Instância, Tamanho máximo do grupo de auto-scaling, Tamanho mínimo do grupo de auto-scaling	1
Tamanho máximo do grupo de auto-scaling, Tamanho mínimo do grupo de auto-scaling, Cooldown	1
Tipos de Instância, Limiar superior, Limiar inferior, Cooldown	1
Tipos de Instância, Limiar superior, Limiar inferior, Tamanho mínimo do grupo de auto-scaling	1
Limiar superior, Limiar inferior, Tamanho máximo do grupo de auto-scaling, Tamanho mínimo do grupo de auto-scaling	1
Limiar superior, Limiar inferior, Tamanho mínimo do grupo de auto-scaling, Cooldown	1
Não entendi a pergunta. É sobre o que a gente usa no nosso auto-scaling, o que prejudica o nosso sistema ou o que beneficia o nosso sistema?	1
Não sei opinar	2

Tabela 7.2: Respostas sobre métricas de interesse.

- Tamanho Mínimo: Quantidade mínima de instâncias que um grupo de *auto-scaling* pode alcançar.

- Tipo de Instâncias: Ou *flavor* da instância, diz respeito à configuração das instâncias que serão utilizadas. Exemplo: número de vCPUs, tamanho do disco, etc.

A importância de entender quais parâmetros mais influenciam e como eles influenciam é que existem estratégias específicas para definir cada um e melhorar a eficiência do *auto-scaling*. Contudo, nem sempre é possível adotar uma estratégia bem embasada devido ao tempo, dificuldade ou urgência dessa implementação. Por esse motivo, também é importante entender qual a base utilizada para tomar decisões de configuração.

Base para configurar. Para decidir todos esses parâmetros de configuração é preciso se basear em algo, do contrário a configuração pode prejudicar tanto a aplicação como aumentar o custo com infraestrutura desnecessariamente. Das 20 respostas, 4 respondentes indicam nunca ter configurado ou ajudado a configurar o *auto-scaling* e um que sua aplicação não usa o mecanismo, portanto, nesse contexto, vamos considerar 15 respostas. Os dois tipos de respostas que aparecem são as seguintes:

1. As decisões de configuração são intuitivas. (3 respostas)

2. As decisões de configuração são baseadas em experimentos com a aplicação. (12 respostas)

Para o segundo tipo de resposta podem existir algumas variações de como esse experimento é realizado de fato, mas como o contexto engloba serviços já em produção, tem-se a suposição que tais experimentos na realidade são testes rápidos realizados nos próprios serviços. Ainda assim, mesmo com testes mais rápidos, seria difícil testar, em um curto espaço de tempo, diversas configurações com diferentes combinações de valores de parâmetros. Esse tipo de experimento fatorial é caro, complexo e não é encorajado como importante para a configuração do *auto-scaling* dentro da empresa.

Dificuldades com *auto-scaling*. Por fim, além das dificuldades já citadas durante o processo de configuração, foi proposto às pessoas voluntárias que descrevessem quais as principais dificuldades em usar o *auto-scaling*. As respostas foram rotuladas pela semelhança das dificuldades e estão listadas no Apêndice B, aqui descreveremos apenas as quatro diferentes categorias identificadas a partir dos rótulos e sua descrição:

- **Momento de *trigger* de ações.** Nessa primeira categoria estão as respostas que apontam como maior dificuldade decidir o melhor momento para adicionar ou remover recursos.

- **Tempo de "rampa".** Nessa categoria estão as respostas que apontam que o tempo que a máquina demora para ficar disponível é um problema.

- **Configurar regras/conhecimento do sistema.** Nesse tópico as respostas estão relacionadas às dificuldades para configurar o *auto-scaling* por falta de algum conhecimento mais aprofundado do serviço.

- **Coleta/Análise.** Nesse último tópico foram agregadas as respostas que indicam maior dificuldade para experimentar e avaliar resultados de uma configuração.

7.3 Discussão

Conhecer as peculiaridades e padrões de uma aplicação auxilia na configuração adequada de *auto-scaling*. O primeiro tópico da análise se destina a entender o quão familiarizadas estão as pessoas em relação à aplicação. Esse resultado ajudou na verificação da visão de cada pessoa sobre a aplicação em contraste com o que os dados mostram.

No segundo tópico da análise falamos um pouco sobre métricas consideradas importantes, onde o foco é entender quais delas conseguem alertar de forma mais eficiente comportamentos anômalos na aplicação. Essas métricas apontadas, **error rate e response time,** geralmente não são utilizadas como métricas para desencadeamento de ações do *auto-scaling*, pois não existe uma relação direta entre erros ou comportamentos estranhos com a quantidade de recursos.

7.3.1 Configurando o *auto-scaling*

Métricas do *Auto-Scaling*. Sobre a configuração do *auto-scaling*, discutida a partir do terceiro ponto, se iniciam análises sobre aspectos mais práticos do *auto-scaling*, como a escolha de métrica para desencadear ações, configuração de outros parâmetros e a base para todas essas ações.

A métrica utilizada no *auto-scaling* deve ser a que melhor define a carga da aplicação. Nesse caso, quase 100% das respostas afirmam usar *CPU Utilization* como métrica para desencadear ações, que é a métrica que possui uma relação mais direta entre degradação do desempenho da aplicação, carga da aplicação e número de recursos. Ou seja, a complexidade é baixa na hora de tentar correlacionar essas métricas. É muito comum usar a utilização de CPU na configuração devido a sua simplicidade, porém existem já alguns trabalhos que comparam o uso de métricas de aplicação com métricas de infraestrutura e mostram que, dependendo da aplicação, a CPU talvez não seja tão eficiente quanto uma métrica de aplicação [6]. Além disso, existe a possibilidade de criação de métricas combinadas, que é uma abordagem um pouco mais sofisticada e exige mais conhecimento da aplicação e infraestrutura para serem definidas.

Contudo, mesmo que a métrica *CPU Utilization* seja de fato a mais adequada, ainda existem outros parâmetros que influenciam bastante na eficiência do *auto-scaling*, como os limiares, os tipos de instância, o *cooldown*, etc. Entender como eles devem ser refletidos na métrica utilizada é fundamental para um bom desempenho do *auto-scaling*.

Outros parâmetros de configuração. No que diz respeito aos demais parâmetros de configuração, os que mais aparecem como importantes são limiar superior e tipo de instâncias, que provavelmente são os que recebem maior atenção na hora da configuração. Para uma configuração mais eficiente, que equilibre desempenho da aplicação e custo, é preciso definir muito bem todos os parâmetros e isso implica entender a relação deles com a métrica observada. A escolha desses parâmetros deve ser individualizada por aplicação, não é de nosso conhecimento uma configuração que se adéqua a todas as aplicações e cargas de trabalho.

Base para configuração. Essa parte da análise serve para explicar como as pessoas estão realizando essas escolhas. A maior parte afirma que faz experimentos com a aplicação para configurar, e o restante afirma que faz essa configuração de forma mais intuitiva. Não podemos afirmar que tipo de experimentos são realizados, se são representativos, executados em ambiente isolado ou não, pois isso não foi informado no formulário. Mas, dado o caráter de urgência das aplicações, tem-se a hipótese de que tais experimentos são testes rápidos com a aplicação mudando alguma configuração e não experimentos de medição fatoriais que seriam os adequados para testar tais ambientes e identificar causa e efeito.

Por último, foram descritas dificuldades e problemas com o *auto-scaling* de forma mais explícita, onde foi possível agrupar por categorias os tipos de dificuldade que figuram. Sendo assim, foram compiladas as 4 maiores dificuldades:

- Melhor momento para adicionar/remover instâncias;
- Evitar e/ou mascarar o tempo de rampa;
- Falta de conhecimento de alguns aspectos da aplicação também dificultam a configuração;
- Experimentar e coletar informações para justificar uma configuração.

A partir do questionário foi possível ter uma visão geral sobre como as pessoas estão fazendo a configuração, a avaliação e quais são suas maiores dificuldades com o *auto-scaling*. A partir dessa análise, poderemos fazer a análise quantitativa dos dados da infraestrutura de forma mais direcionada.

Capítulo 8

Métricas avaliativas

Neste capítulo estão detalhadas as métricas utilizadas para diagnosticar o mecanismo de *auto-scaling*, algumas definições para entendê-las, bem como a motivação para utilizá-las.

Avaliando as métricas propostas na literatura, procuramos aquelas que mais se adéquam ao estudo, relacionando os resultados do questionário, os dados coletados e alguns objetivos da empresa de melhorar a elasticidade.

8.1 Definições

Demanda de Recursos. É definida como a quantidade mínima de recursos necessária para lidar com a intensidade da carga sem violar os SLOs (*Service Level Objectives*). Sendo assim, a demanda pode ser expressa como uma função da carga. Idealmente seriam realizadas algumas rodadas de experimentos com cada aplicação para testar os limites em relação a violações de SLO. Contudo, as aplicações analisadas e seus tamanhos são fatores proibitivos até o presente momento, além do fato das aplicações mudarem o tempo todo, como vimos no Capítulo 6. Por esse motivo, foi preciso pensar em outras abordagens para medir essa variável. A primeira forma representa a demanda de recursos em função da Utilização de CPU e a segunda em função do SLO:

1. **Demanda de Recursos em função da CPU** ($d_t cpu$). O valor alvo de utilização definido na configuração do *auto-scaling*, em teoria, é o valor ideal para aquela aplicação. Para calcular a demanda de recursos ideal aproximada para determinada aplicação,

selecionamos os instantes de tempo onde a utilização de CPU está no máximo 5% distante do alvo definido. Agora, para os instantes selecionados, calculamos a média das razões de quantidade de requisições por instância. Assim, teremos a aproximação de quantas requisições uma única máquina consegue atender. Assim, a qualidade desta abordagem está diretamente ligada à boa decisão sobre a utilização alvo configurada.

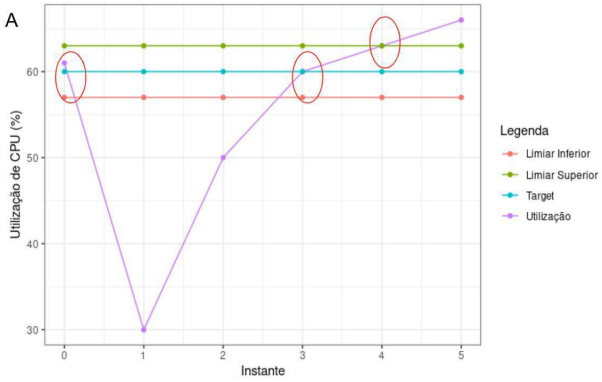

Figura 8.1: Valores ilustrativos de Utilização de CPU.

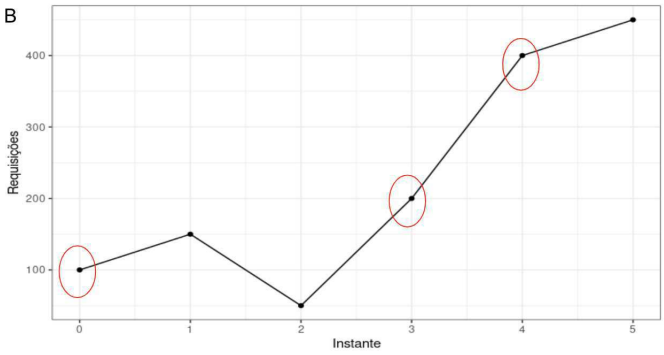

Figura 8.2: Valores ilustrativos de Requisições.

Para exemplificar melhor, na Figura 8.1 temos a Utilização de CPU variando ao longo do tempo. Podemos observar que os pontos destacados em vermelho são pontos onde a Utilização está no máximo 5% distante do *Target*/Alvo de utilização definido. Os valores observados nesses instantes serão utilizados para calcular a demanda (instantes $0, 3$ e 4). Em seguida, nos instantes selecionados calculamos a média da razão entre

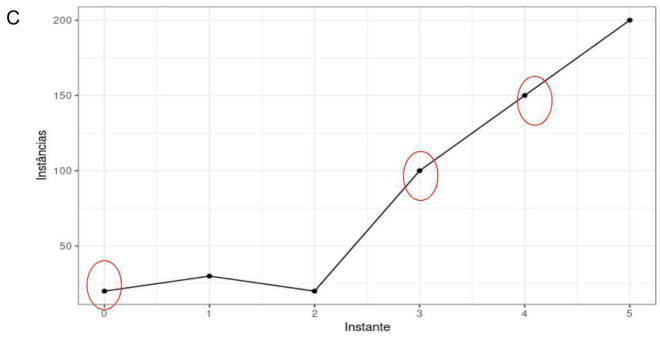

Figura 8.3: Valores ilustrativos de Instâncias.

Requisições e Instâncias. Os pontos usados no cálculo estão destacados em 8.2 e em 8.3, $\frac{1}{3}(\frac{100}{20}+\frac{200}{100}+\frac{400}{150})$, que resulta no valor de $3,2$. Ou seja, uma instância é capaz de atender de forma adequada 3,2 requisições por segundo. Então, a demanda calculada a cada instante terá como base que a proporção que a aplicação precisa é de 1 instância para 3,2 requisições.

2. **Demanda de Recursos em função do SLO** ($d_t slo$). Outra forma de calcular a demanda de recursos de uma aplicação foi em função do SLO - considerando SLOs de Latência. Nesse cálculo seguimos a mesma ideia anterior, contudo, o critério para selecionar os instantes de tempo é diferente. Nesse caso selecionaremos os pontos de dados onde a latência é menor do que o limiar de latência definido no SLO, mas 15 % menor que o valor de desse limiar. Por exemplo: Se o SLO definido é que o percentil 90 da latência deve ser menor do que 190ms, então os pontos de dados utilizados são aqueles onde a latência está entre 190ms e 161.5ms (que representa 85% de 190ms).

Essas duas formas de calcular a demanda foram incorporadas, quando necessário, às métricas selecionadas que apresentaremos a seguir:

Percentual do tempo em capacidade mínima (C_{min}): Se relaciona diretamente com parâmetro Capacidade Mínima da configuração do *auto-scaling*. Esse parâmetro especifica qual a menor quantidade aceitável de recursos que aquela aplicação pode atingir. O tempo em capacidade mínima, como o próprio nome indica, vai medir a porcentagem de tempo em que a aplicação usa a capacidade mínima e pode ser definida pela equação 8.1. O tempo, nesse

caso, é uma variável discreta e todos os intervalos de medição possuem a mesma duração.

- Q: Quantidade de instâncias alocadas em um determinado instante de tempo;
- M: Capacidade mínima de instâncias que a aplicação pode assumir. Definido pelo usuário;
- t: Instante em que a medição de quantidade de instâncias foi realizada;
- T: Tempo total do experimento;

$$C_{min}[\%] = \frac{100}{T} \sum_{t=1}^{T} C_t, \tag{8.1}$$

onde,

$$C_t = \begin{cases} 0, & \text{se } Q > M \\ 1, & Q = M \end{cases}$$

A porcentagem de tempo em capacidade mínima pode ser um indício de que a aplicação esteja super escalada, ou seja, menos recursos poderiam ser alocados, mas isso não acontece devido ao parâmetro de capacidade mínima. Essa métrica serve especificamente para identificar se a quantidade mínima de recursos configurada está muito elevada. Idealmente esta capacidade mínima deveria ser atingida apenas nos vales de carga e liberar de fato o serviço de *auto-scaling* para decidir sobre a capacidade da aplicação nos demais momentos. Quando a capacidade mínima é configurada com um valor mais alto do que o necessário pode causar super-provisionamento, uma vez que o serviço de *auto-scaling* pode estar constantemente indicando uma quantidade de recursos menor que a capacidade mínima.

***Timeshare* de provisionamento incorreto (t_{sub}, t_{super})**: O tempo de provisionamento incorreto captura o tempo em que o sistema está sub-provisionado ou super-provisionado. O intervalo dessa métrica é o intervalo [0, 100]. O melhor valor, nesse caso 0, é alcançado, quando o sistema não tem, durante a medição, super ou sub-provisionamento [16] [6]. Essa métrica gera duas variáveis, uma para indicar o sub provisionamento outra o super provisionamento, que podem ser calculadas a partir das equações 8.3 e 8.4, onde $(x)^+ := max(x, 0)$

e $sign(x)$ é a função sinal, essa última é uma função matemática que extrai o sinal de um número real. [1]

- s_t: Quantidade de recursos ofertados no instante t;
- d_t: Quantidade necessária de recursos para atender a demanda no instante t. Nesse caso, podemos substituir tanto pela demanda calculada em função da Utilização de CPU ($d_t cpu$), como pela demanda calculada em função do SLO ($d_t slo$);
- T: Tempo total da carga da qual foi extraída a métrica.

$$t_{sub}[\%] = \frac{1}{T}\sum_{t=1}^{T} sign(d_t - s_t)^+ \tag{8.3}$$

$$t_{super}[\%] = \frac{1}{T}\sum_{t=1}^{T} sign(s_t - d_t)^+ \tag{8.4}$$

Acurácia (θ_u,θ_o): A acurácia de provisionamento descreve a quantidade relativa de recursos que estão sub-provisionados ou super-provisionados durante o intervalo de medição [16]. A acurácia de sub-provisionamento é a quantidade de recursos faltantes necessário para atender a SLO em relação à demanda atual normalizada pelo tempo do experimento. Analogamente, a acurácia de super-provisionamento é quantidade de recursos fornecida acima da demanda atual, normalizadas pelo tempo. Essa é a definição original da métrica, que pode ser definida matematicamente pelas Equações 8.5 e 8.6. O tempo nesse caso é uma variável discreta e todos os intervalos de medição possuem a mesma duração.

- s_t: Quantidade de recursos ofertados no instante t;
- d_t: Quantidade necessária de recursos para atender a demanda no instante t. Nesse caso, podemos substituir tanto pela demanda calculada em função da Utilização de CPU ($d_t cpu$), como pela demanda calculada em função do SLO ($d_t slo$);
- T: Tempo total da carga da qual foi extraída a métrica.

[1] `https://www.statisticshowto.com/sign-function/`

$$\theta_u[\%] := \frac{100}{T} \sum_{t=1}^{T} \frac{\max(d_t - s_t, 0)}{d_t} \tag{8.5}$$

$$\theta_o[\%] := \frac{100}{T} \sum_{t=1}^{T} \frac{\max(s_t - d_t, 0)}{d_t} \tag{8.6}$$

De forma mais explicita, essa métrica vai nos retornar uma porcentagem da média de recursos faltantes ($\theta_u[\%]$) ou excedentes ($\theta_o[\%]$) em relação a demanda no intervalo total.

Essas métricas são calculadas para algumas aplicações da empresa usando os dados coletados durante esta pesquisa. A ideia é que estas métricas podem nos auxiliar a encontrar problemas, ou validar, a configuração do serviço de *auto-scaling*.

Capítulo 9

Resultados

Neste Capítulo, descrevemos os resultados da análise de uso da infraestrutura, aplicando as métricas descritas no Capítulo 8. A empresa alvo do nosso estudo de caso mantém um grande número de aplicações e estudar todas elas não seria produtivo. Por essa razão, selecionamos quatro aplicações representativas. Consideramos três critérios para a escolha das aplicações. O primeiro critério considera a quantidade de instâncias usadas. Aplicações que usam muitas instâncias são mais interessantes, pois identificar problemas (e posteriormente resolvê-los) trará maior impacto para a empresa. O segundo critério considerou a variabilidade na demanda e nas métricas de *auto-scaling* (no caso, utilização de CPU). Aplicações com pouca variação na demanda e no uso de recursos podem ter um uso simplório do mecanismo de *scaling* (ou mesmo nem precisar dele) e seriam pouco interessantes para o nosso estudo. Por fim, consideramos a importância das aplicações para o negócio da empresa. Todas as aplicações escolhidas pertencem ao nível mais alto de criticidade (nível 1), definido pela empresa como: *"... os mais críticos do nosso sistema. Um serviço é considerado nível 1 se uma falha do mesmo vai resultar num impacto significativo para clientes ou para a empresa."*

No que segue deste Capítulo, primeiro introduziremos algumas características importantes comuns à todas as aplicações. Estas características envolvem aspectos tanto do modo como as aplicações são implantadas quanto da maneira como o *auto-scaling* funciona e é configurado. Em seguida, mostramos uma visão geral sobre como é a demanda submetida para as aplicações analisadas. Estes dois primeiros aspectos são importantes para a leitura das análises descritas e discutidas mais ao fim do Capítulo.

9.1 Sobre as Aplicações e suas configurações

Um aspecto importante em nossas análises é o método seguido para implantar atualizações das aplicações. Qualquer nova funcionalidade ou correção em uma aplicação gera uma nova versão desta. Quando essa **mudança de versão** ocorre, a antiga versão é substituída pela nova de forma gradual, isso significa que, pelo menos em um primeiro momento, aquela aplicação vai usar o dobro de recursos que usaria normalmente e aos poucos vai atingir seu tamanho usual. Essa característica pode ser notada facilmente em todos os gráficos ao longo do capítulo.

A empresa usa dois tipos de **políticas de *auto-scaling*** dentre as disponibilizadas pelo provedor de nuvem: *Target Tracking Policy* e *Simple Scaling Policy*. Essas duas políticas são bem difundidas na empresa. A primeira tem como objetivo fazer com que o grupo de instâncias de uma aplicação mantenha o valor da métrica de *scaling* próxima a um alvo definido. Por sua vez, a segunda tem como objetivo fazer com que o grupo mantenha a utilização dentro da faixa de valores entre um limiar superior e inferior.

O provedor IaaS usado pela empresa implementa essas duas políticas, mas a empresa optou por terceirizar o controle do *auto-scaling* para um serviço externo, que vai ditar como os recursos da nuvem são provisionados. Esse serviço externo também implementa as mesmas políticas de acordo com a documentação, ou pelo menos bem similares. Através de discussões com suporte desses serviços, descobrimos que existem algumas diferenças na implementação da política *Target Tracking Policy*. Abaixo temos a definição de cada uma e suas diferenças.

- **Target Tracking Policy - Provedor de nuvem**: No uso dessa política, deve ser escolhida uma métrica e em seguida um valor alvo para essa métrica. Para usar os recursos de forma eficiente, o valor alvo deve ser suficiente para conseguir lidar com aumento repentino da demanda[1].

- **Target Tracking Policy - Serviço Externo:** Tal como na definição usada pelo provedor de nuvem IaaS, uma métrica e seu valor alvo devem ser escolhidos. O serviço vai gerenciar a quantidade de recursos, mantendo a métrica igual ou próxima ao valor de-

[1] `https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-target-tracking.html`

sejado. Adicional à essa definição, segundo o suporte da empresa que presta o serviço, foi explicitado que apesar dessa política se basear em um único valor, existe um *buffer* de 10% do valor alvo tanto para adicionar como para remover recursos. Por exemplo, se o valor alvo for de 65% de utilização de CPU, o *auto-scaling* só irá reagir quando a utilização estiver fora do intervalo [58.5%, 71.5%] [2].

Algumas aplicações além da configuração básica do *auto-scaling* também usam uma funcionalidade que permite mudar alguns parâmetros ao longo do tempo, através de **tarefas agendadas**. Sendo assim, aplicações podem ter mais de uma configuração ao longo do tempo.

9.2 Demanda

Aqui chamamos de demanda a quantidade de requisições recebida pelas aplicações ao longo do tempo. A partir disso, conseguimos identificar padrões e variações que podem contribuir para uma boa configuração de *auto-scaling*. Antes de analisarmos, é importante lembrar que consideramos apenas uma versão de cada aplicação, e como são independentes, a escala de tempo nos gráficos dessa seção é diferente para cada uma.

Nos gráficos mostrados na Figura 9.1, as áreas azuis em destaque indicam o horário de 8:00 às 20:00 (GMT-3). Esse horário é correspondente ao período em que normalmente estabelecimentos de comércio e serviços realizam atendimento ao consumidor. Apesar desse horário se aplicar a estabelecimentos físicos na maioria das vezes, desde o início da pandemia de COVID-19 os hábitos de compra online também se modificaram [3] e se assemelham ao padrão que observamos no gráfico.

Apesar de apresentar padrões parecidos ao longo do tempo, as aplicações se diferenciam na magnitude da demanda recebida. Por exemplo, comparando o máximo de requisições que cada Aplicação recebe, temos que a Aplicação 1 possui uma demanda mil vezes maior que a Aplicação 4.

Como foi visto no início desta seção, todas essas aplicações são de extrema importância

[2] `https://docs.spot.io/elastigroup/features/scaling/target-scaling`

[3] `https://www.ecommercebrasil.com.br/noticias/flexibilidade-horario-comportamento\-compra-brasileiros-coronavirus`

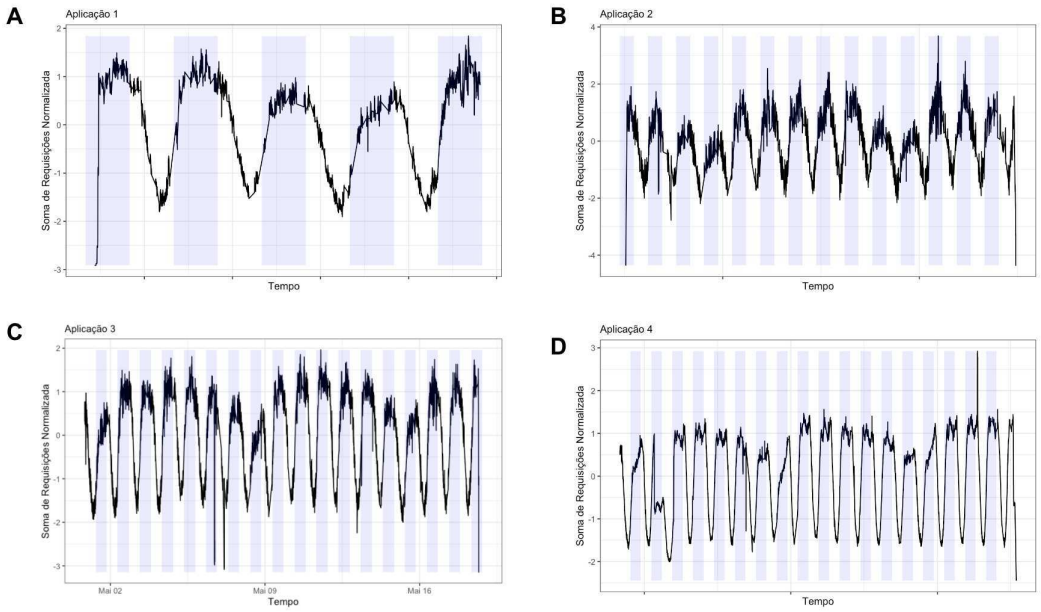

Figura 9.1: Soma de requisições ao longo do tempo para as quatro aplicações selecionadas.

para o funcionamento dos Produtos. Todas elas apresentam uma variação considerável na sua demanda. Dessa forma, a alocação de recursos torna-se mais eficiente e menos custosa caso seja dinâmica, ou seja, são casos onde usar *auto-scaling* é vantajoso.

Todas essas aplicações usam o mecanismo de *auto-scaling* com configurações realizadas pelos próprios times que as mantêm. Mesmo havendo monitoramento dos recursos utilizados, até o presente momento, não é comum realizar análises embasadas sobre como a configuração escolhida pode afetar a eficiência do mecanismo antes mesmo de aplicar as mudanças em ambiente de produção. Isso pode acontecer por vários motivos. Um deles é o fato de que, especialmente para as aplicações escolhidas, existem demandas de desenvolvimento de funcionalidades que afetam diretamente clientes, portanto essa tarefa de experimentação e análise pode acabar ficando em segundo plano.

Além disso, como foi reportado no Capítulo 7 com o Questionário, existe uma dificuldade em experimentar, coletar e justificar escolhas para o *auto-scaling* de forma estruturada. Por isso, na próxima Seção realizar um diagnóstico utilizando as métricas apresentadas no Capítulo anterior, que que podem ser incorporadas à análise da configuração do *auto-scaling* e auxiliar nas decisões de valores de parâmetros que afetam custo e eficiência.

9.3 Análise

A análise de dados a seguir tem como objetivo demonstrar o uso de métricas que consideram diferentes aspectos do *auto-scaling*, mostrando como elas podem explicitar pontos positivos e pontos de melhorias de configurações das aplicações.

9.3.1 Tempo em Capacidade Mínima

Com já descrito no Capítulo 8, o tempo em capacidade mínima vai medir a porcentagem de tempo em que a aplicação é provida com a capacidade mínima configurada.

Nas Figuras 9.2 e 9.3 observamos o número de instâncias ao longo do tempo, representada pela linha cinza, e o valor da capacidade mínima configurada na aplicação, representado pela linha vermelha tracejada. Esse padrão é usado em todos os gráficos usados para analisar o percentual do tempo em capacidade mínima

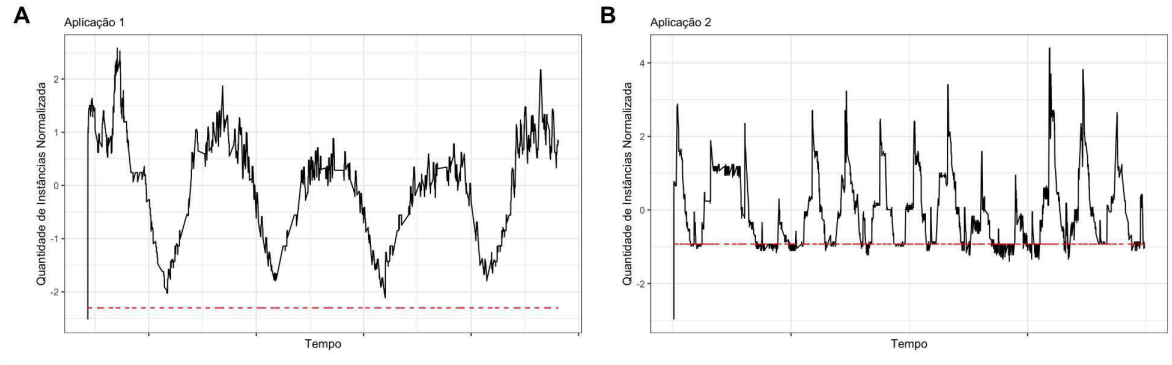

Figura 9.2: Quantidade de instâncias ao longo do tempo para as Aplicações 1 e 2.

Na Aplicação 1, apesar de a linha que representa a quantidade de instâncias atingir a linha vermelha (capacidade mínima) algumas vezes ao longo do tempo, podemos desconsiderar, pois, como foi explicitado no início, existem as mudanças de versões que causam esses comportamentos considerados estranhos, mas são comuns a todas as aplicações. Assim, essa primeira Aplicação se mantém pouco ou nenhum tempo em sua capacidade mínima. Isso significa que, se em algum momento identificarmos um super-provisionamento nessa versão, a causa não será o valor do parâmetro **capacidade mínima**.

Ao contrário da Aplicação 1, a 2 já apresenta um comportamento diferente. Ficando na

sua capacidade mínima durante quase 30% do período de tempo analisado. Isto pode ser um indício de um ambiente superescalado devido ao parâmetro de **capacidade mínima**. No caso apresentado não parece um indício grave, uma vez que a capacidade mínima é usada em geral em períodos de vales de carga.

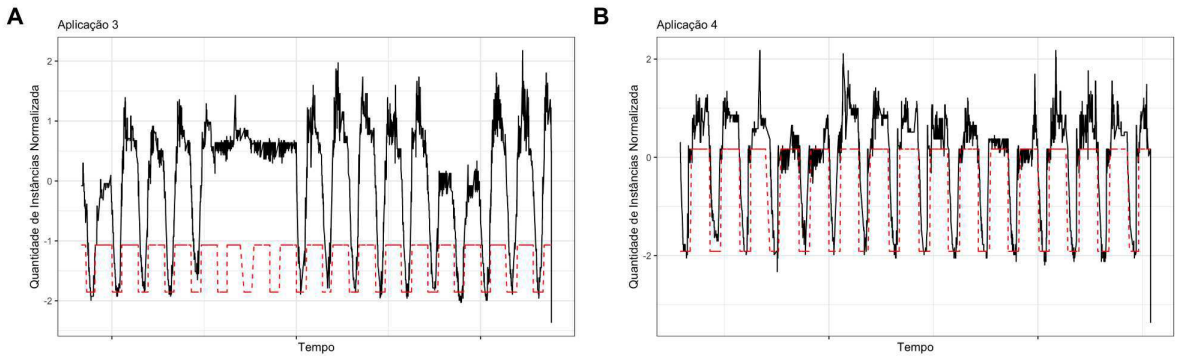

Figura 9.3: Quantidade de instâncias ao longo do tempo para as Aplicações 3 e 4.

Para apresentar os resultados das Aplicações 3 e 4 é preciso explicar sobre um conceito usado na configuração destas aplicações. Trata-se da configuração dinâmica do parâmetro capacidade mínima. Os resultados para estas aplicações são ilustrados na Figura 9.3. Quando o parâmetro de capacidade mínima é dinâmico, ele assume mais de um valor durante o dia. Essas duas aplicações ficam em capacidade mínima em 5.8% e 13% do tempo, respectivamente. Para entender melhor a razão do aumento de complexidade da configuração dessas aplicações (ou seja, a adoção de níveis dinâmicos de capacidade mínima), analisamos em mais detalhes os horário típicos em que a aplicação ficava em capacidade mínima, bem como verificamos as respostas ao questionário descrito no Capítulo 7.

Primeiro, a capacidade mínima tende a aumentar próximo ao horário comercial e diminuir na madrugada, por volta das 03:00h. Segundo, as aplicações possuem um tempo de *warm-up* muito alto e provavelmente não conseguem atender o pico matinal rápido o suficiente. Por isso a capacidade mínima é alterada algumas horas antes para que no horário do pico já existam máquinas suficientes para atender a demanda.

Essa estratégia funciona e atende aos objetivos atuais. Porém, os horários de modificação da configuração estão fortemente ligados ao horário local (GMT-3), o que torna esse tipo de configuração arriscada, dado que a empresa tende a um crescimento internacional (ou seja,

ter horários de pico que não coincidem).

9.3.2 *Timeshare* de provisionamento incorreto

Como já descrito, essa métrica consegue fornecer informações sobre a razão de tempo na qual a aplicação ficou super ou sub-provisionada durante o período de observação. Isso pode ser útil para avaliar tanto o algoritmo, como também a configuração. No caso dessa pesquisa, estamos realizando uma análise a posteriori, avaliando a configuração realizada pelos usuários.

Para a análise usando essa métrica optamos por calcular a demanda ideal de recursos de duas formas diferentes, já descritas no Capítulo 8. A primeira forma é calculando essa demanda de recursos como uma função da métrica de *scaling* (nesse caso a Utilização de CPU), utilizando os pontos de dados onde a utilização está no máximo 15% distante do limiar definido. Selecionando esses pontos extraímos a média da razão de requisições por instância, obtendo assim o número médio de requisições que cada instância daquela aplicação consegue atender sem se afastar muito do limiar determinado.

A Tabela 9.1 mostra os resultados da métrica de Tempo de Provisionamento Incorreto, quando calculamos a demanda ideal de recursos em função da utilização de CPU.

Aplicação	Tempo Super-provido	Tempo Sub-provido	Tempo Medição
Aplicação 1	78.62%	19.70%	105 horas
Aplicação 2	68.50%	26.95%	332 horas
Aplicação 3	76.82%	18.56%	77 horas
Aplicação 4	91.39%	8.68%	453 horas

Tabela 9.1: Tempo de Provisionamento Incorreto calculado em função da Utilização de CPU.

Podemos observar que os períodos de super-provisionamento são sempre maiores que 60% e os de sub-provisionamento menores que 30%. Esse resultado pode ser explicado pelo fato de que os cenários de sub-provisionamento são sempre mais críticos e causam mais impacto do que os cenários de super-provisionamento, onde o maior ponto negativo é o desperdício de recursos.

Portanto, pelo menos para essas aplicações, temos indícios de gastos desnecessários com

recursos. Isso pode ocorrer por diversos motivos, mas como relatado no questionário, uma das maiores dificuldades é definir o momento ideal de provisionamento e, como as aplicações são muito críticas, existe um receio de deixar a configuração menos conservadora. De forma simplificada, alguns parâmetros que podem influenciar diretamente nisso são o *step size* e a capacidade mínima. Para configurar isso da melhor forma, é necessário entender bem a relação entre a métrica de *scaling* e esses parâmetros [2], experimentos para entender tais relações teriam que fazer parte da rotina da equipe.

A segunda forma de calcular a demanda ideal de recursos é como uma função do SLO, no caso desta empresa, limiar máximo aceitável para a latência. Esta abordagem de cálculo foi definida no Capítulo 8. A partir da Tabela 9.2 conseguimos observar o resultado da métrica de Tempo de Provisionamento Incorreto quando calculamos a demanda ideal de recursos em função da latência.

Aplicação	Tempo Super-provido	Tempo Sub-provido	Tempo Medição
Aplicação 1	56.22%	42.19%	105 horas
Aplicação 2	23.99%	74.05%	332 horas
Aplicação 3	2.77%	97.14%	77 horas
Aplicação 4	91.8%	8.20%	453 horas

Tabela 9.2: Tempo de Provisionamento Incorreto calculado em função da Latência.

Observamos, na Tabela 9.2, que dessa forma os resultados quase invertem totalmente. A métrica aponta um maior sub-provisionamento em três das aplicações. Para entender melhor esse resultados, podemos também verificar a variação de Utilização de CPU nos pontos selecionados para calcular a demanda de recursos ideal, ou seja, onde o SLO está próximo de ser violado. Na Tabela 9.3 conseguimos observar o comportamento da CPU quando a latência está próxima de violar o SLO.

Isso é um indício de que as requisições que chegam na aplicação são diferentes na quantidade de recursos que demandam. Classificar as requisições seria importante para melhorar a configuração do *auto-scaling* e também para o balanceamento de carga entre as instancias. Contudo, avaliar com profundidade os tipos de requisição está fora do escopo desta pesquisa.

Aplicação	Desvio Padrão CPU	Min CPU	Max CPU	Coef. Variação
Aplicação 1	15.66%	33.28%	80.40%	24.49%
Aplicação 2	10.68%	19.25%	72.25%	24.54%
Aplicação 3	17.00%	6.72%	48.00%	68.00%
Aplicação 4	9.68%	0.34%	61.00%	24.86%

Tabela 9.3: Comportamento da Utilização de CPU quando próxima de violar o SLO.

9.3.3 Acurácia de Provisionamento

Acurácia de Provisionamento permite quantificar os recursos faltantes e excedentes de uma aplicação ao longo do tempo. Da mesma forma que o Tempo de Provisionamento Incorreto, o cálculo da acurácia também exige a variável demanda de recursos. Aplicando a mesma estratégia, calculamos a métrica tanto em função da Utilização de CPU, como da Latência (SLO). Na Tabela 9.4 vemos o resultado da Acurácia de Provisionamento em função da CPU.

Aplicação	Excedente	Faltante
Aplicação 1	62.98%	0.86%
Aplicação 2	20.48%	1.99%
Aplicação 3	22.64%	1.20%
Aplicação 4	75.05%	8.03%

Tabela 9.4: Acurácia de Provisionamento em função da CPU.

A quantidade de recursos excedentes é maior em todas as aplicações. Indicando um possível desperdício de recursos.

Para a Acurácia, também observamos um resultado quase invertido quando calculamos a métrica em função da Latência, na Tabela 9.5 podemos notar similaridades com a métrica anterior, onde algumas aplicações possuem mais sub-provisionamento do que super-provisionamento. Nesse caso, mais recursos faltantes do que excedentes.

Aplicação	Excedente	Faltante
Aplicação 1	54.71%	4.10%
Aplicação 2	8.01%	12.60%
Aplicação 3	2.72%	49.94%
Aplicação 4	246.68%	7.60%

Tabela 9.5: Acurácia de Provisionamento em função da Latência.

9.3.4 Violações de SLO

Além das métricas ligadas à elasticidade do sistema, também foi possível observar o comportamento das aplicações em relação aos seus respectivos SLOs. Consideramos apenas SLOs relacionados à latência. Realizamos duas análises nesse sentido. Na primeira, calculamos a porcentagem de violações dado o período de medição. Na segunda, calculamos a quantidade de períodos consecutivos de violação. Para isso foi necessário definir um grão de tempo para extrair a estatística definida. Nesta análise, escolhemos 15 minutos para todas as aplicações. Por exemplo, se um SLO é definido como percentil 90 (p90) da Latência < 190ms, então agrupamos os dados a cada 15 minutos e verificamos se o p90 da latência é menor ou maior que 190ms (classificando como uma violação ou não), e a partir disso calculamos, em média, quantos intervalos de 15 minutos consecutivos violam o SLO.

Tanto a porcentagem de violações no total, como a média de violações consecutivas podem ser observadas na Tabela 9.6 para cada a aplicação. No geral, as Aplicações 1, 2 e 3 apresentaram poucas violações de SLO. A Aplicação 4 apresentou um percentual maior de violações de SLO, e apresentou $3.3 períodos consecutivos de violação, que foi o máximo observado. Esses números reforçam os resultados das outras métricas, que apontam um super-provisionamento na maior parte do tempo, indicando que há espaço para redução de custos através de um melhor gerenciamento a curto prazo, mas ao mesmo tempo, dada a presença de violações de SLO, indica que um *tuning* mais fino na configuração do *auto-scaling* é necessário.

Aplicação	Média de Violações Consecutivas	Violações (%)	Tempo
Aplicação 1	1.5	0.9%	105 horas
Aplicação 2	1.3	0.6%	332 horas
Aplicação 3	1.5	0.4%	77 horas
Aplicação 4	3.3	11.9%	453 horas

Tabela 9.6: Violações de SLO.

9.4 Discussão

Problemas na configuração de *auto-scaling* podem acarretar duas consequências com implicações diretas no negócio da empresa. Primeiro, é possível que a infraestrutura esteja sub-provisionada e portanto o desempenho das aplicações e a experiência dos usuários podem ser comprometidas. Segundo, é possível que a infra-estrutura esteja super-provisionada e, por consequência, a empresa mantém desempenho aceitável mas paga em excesso por isso.

A última métrica usada em nossa análise, **Violações de SLO**, apresenta um bom panorama sobre o primeiro aspecto (relacionado com a qualidade dos serviços). A porcentagem de violações e de períodos consecutivos de violação se mostraram baixos para três aplicações das quatro analisadas.

As demais métricas permitem identificar oportunidades de melhoria, em particular no que diz respeito à redução de custos. Por exemplo, considerando mudanças de valores em alguns parâmetros, tais como capacidade mínima e *step size*, em horários do dia em que as métricas apontam problemas maiores. Um ponto negativo, principalmente com as métricas de ***Timeshare* de provisionamento incorreto** e **Acurácia de Provisionamento** é que se faz necessário calcular qual a real demanda para uma determinada carga. Isso não é simples de ser feito com base somente em dados de utilização tal como o que usamos; uma vez que a quantidade de recursos usada também é consequência da configuração do *auto-scaling* durante o período de observação. Para que as análises sejam mais acuradas é necessário que o cálculo da demanda de recursos seja realizado através de experimentos, por exemplo.

Capítulo 10

Conclusão

Neste Capítulo resgatamos o objetivo inicial da pesquisa, como esse objetivo foi alcançado e o que foi encontrado com as investigações realizadas, tanto na etapa do questionário como na etapa de diagnóstico do *auto-scaling*. Em seguida, discutimos sobre as limitações e ameaças à validade. Por fim, na última seção, levantamos possíveis trabalhos futuros.

10.1 Objetivos

O objetivo principal desta pesquisa foi investigar a avaliação de serviços de *auto-scaling* considerando o caso particular do uso do serviço no cenário de uma empresa de médio porte que usa nuvem pública. Essa investigação se deu em duas etapas. A primeira etapa da investigação foi entender o contexto em que ocorre avaliação e configuração do mecanismo de *auto-scaling*. Consideramos aspectos tais como: 1) o quanto as pessoas estão familiarizadas com a aplicação; 2) o quanto conhecem o mecanismo; 3) se entendem a relação entre métricas e parâmetros; e 4) quais seriam as principais dificuldades no uso do *auto-scaler*. A segunda etapa, realizada após obter conhecimento sobre o contexto, envolveu compilar métricas para avaliação e aplicá-las a dados reais. Com isso, foi possível gerar diagnósticos que podem ser usados para avaliar e auxiliar nas configurações do *auto-scaling*. Ao decorrer dessa investigação, foi possível mapear estratégias da literatura para problemas identificados no processo. Esse mapeamento e outras descobertas estão melhores descritos na seção seguinte.

10.2 Questionário

Estabelecemos inicialmente um objetivo de entender o contexto, que foi alcançado através do questionário e sua análise. Também estabelecemos um objetivo prático, que foi o de usar o conhecimento do questionário para propor soluções factíveis. Sabendo disso, aqui listaremos alguns mapeamentos possíveis, que foram encontrados a partir da análise das respostas.

Um primeiro mapeamento diz respeito às métricas de aplicação e sistema. No questionário vimos que as métricas consideradas mais importantes são duas relacionadas à aplicações, o *response time* e o *error rate*. Já a métrica mais utilizada para desencadear ações foi a Utilização de CPU. Esse resultado era esperado, dado que não existe uma função direta que correlacione métricas de aplicação com quantidade de recursos. No entanto, o primeiro mapeamento que encontramos na literatura para essa situação foi a possibilidade de combinar métricas [28] ou até mesmo usar a demanda [5]. Para realizar esse estudo, seria necessário implementar o suporte às métricas de aplicação em um simulador, ou até verificar se já existe na literatura. Essa iniciativa não é trivial devido à necessidade de implementação do simulador, portanto não pode ser aplicada na prática imediatamente.

Um segundo mapeamento considera a configuração de outros parâmetros que não sejam métricas de *scaling*. O único mapeamento foi o de começar a entender melhor como cada parâmetro afeta a métrica usada para avaliar e a métrica de *scaling*. Já existem trabalhos que observam esse fator, mas ainda não para todos os parâmetros [2]. Essa ação pode ser aplicada de forma imediata, pois consiste em um estudo para entender relação entre métrica e parâmetro.

Em seguida, consideramos um mapeamento sobre base para configuração. Sabemos que é inviável testar muitas configurações em ambiente de produção, pois seriam necessários vários ciclos de mudanças e implantações de novas versões da aplicação. Sabendo disso, o mapeamento direto que temos é o de simulação de uma configuração usando dados do passado da aplicação. Como veremos na seção 10.5, simular políticas de *scaling* é uma iniciativa que já está acontecendo na prática como auxílio da presente pesquisa.

Por fim, considerando as dificuldades relatadas, identificamos que o "tempo de rampa" é uma questão importante. Nesse caso, temos dois mapeamentos, o primeiro corresponde

à tentativa de medir de fato tal tempo de rampa e, em seguida, adaptar a configuração. Já o segundo mapeamento corresponde à tentativa de mascarar esse tempo usando instancias *burstable* [4]. Para esse segundo ponto, seriam necessárias outras interações de experimentos e análises até chegar em um formato prático, portanto pode ser pautada como sugestão, mas não pode ser aplicada imediatamente.

10.3 Diagnóstico do *Auto-scaling*

A análise de dados da infraestrutura teve como guia as respostas do questionário e a revisão da literatura. Um primeiro mapeamento tem relação com o uso da métrica **Percentual de Tempo em Capacidade Mínima**. Conseguimos identificar ocorrência de superprovisionamento causadas pelo parâmetro capacidade mínima. Isso por si já é um indício de que o seu valor pode ser melhor configurado. Além disso, observamos aplicações onde esse mesmo parâmetro é dinâmico, mudando de acordo com as horas do dia. Unindo tais observações com as informações do questionário, temos que possivelmente ocorre uma compensação do tempo de rampa utilizando uma capacidade mínima dinâmica, ou seja, o valor de capacidade mínima é incrementado para que a aplicação não degrade durante o aumento de carga. Um possível mapeamento para esse ponto é incorporar de fato a métrica às análises, pois é uma métrica simples e de fácil medição, podendo identificar valores inadequados no parâmetro de capacidade mínima. Outro ponto de melhoria, que pode ser implementado a longo prazo é a dedicação de estudo ao tempo de rampa das aplicações, algo que já foi citado na seção anterior.

Ao estudar os resultados obtidos com a análise da métrica ***Timeshare* de Provisionamento Incorreto**, encontramos indícios de que as aplicações estão frequentemente superprovisionadas. Fazendo um paralelo com o questionário, surgiu uma questão relacionada à dificuldade de definir o melhor momento para provisionar novos recursos, complementando os valores obtidos com a métrica em alvo. Para essa questão, o mapeamento é entender melhor a relação entre a métrica de *scaling* utilizada e os parâmetros que influenciam diretamente nisso, a exemplo dos parâmetros de *step size* e a capacidade de mínima. Já existem trabalhos que falam sobre essas relações [2].

Entendemos que é incomum acontecerem problemas de desempenho por sub-

provisionamento. Quando verificamos as violações de SLO, encontramos, para a aplicação com mais violações, no máximo 11.9% de tempo em violação. Além disso, a quantidade de violações consecutivas de SLO foram baixas, atingido no máximo 45 minutos para esta mesma aplicação. Combinando a informação de que a maioria dos cenários é de um super-provisionamento, com os dados de poucas violações de SLO, vemos que pode haver espaço para reduzir o super-provisionamento antes de atingir um nível de violação grave. Incorporar essa métricas às análises de configuração é simples e pode ser feita de forma imediata. A questão da configuração do *auto-scaling* é complexa, e nossos resultados demonstram isso: mesmo uma aplicação que fica super-provida na maior parte do tempo pode ter violações de SLO, especialmente se o serviço de *auto-scaling* não consegue reagir rapidamente às mudanças mais bruscas de carga crescente.

Por fim, ao relacionarmos as informações sobre SLO considerando latência e Utilização de CPU, encontramos uma outra evidência: existe uma variação considerável de Utilização de CPU - métrica de *scaling* utilizada - em um determinado intervalo de latência. Isso pode ser um indício de que a carga é heterogênea. Ou seja, pode haver diferentes tipos de requisição por aplicação, as quais consomem recursos de formas diferentes. Por esse motivo, consideramos que a classificação de requisições que chegam às aplicações é uma mapeamento importante. Entender essa divisão permite uma melhor distribuição da carga e configurações no geral. Essa iniciativa, entretanto, é complexa e precisaria de uma nova coleta de dados, análises e experimentações.

10.4 Limitações

Nesta pesquisa foram utilizados dados coletados de duas maneiras diferentes: os dados de recursos da infraestrutura, a partir de uma ferramenta também construída durante a pesquisa, e dados sobre como as pessoas configuram o *auto-scaling*, obtidos a partir de um questionário estruturado. Nessa seção, serão destacadas algumas ameaças à validade dos resultados obtidos [29].

Das ameaças à validade encontradas, classificamos as três como internas. Todas dizem respeito aos dados coletados da infraestrutura. A primeira tem a ver com a ferramenta de coleta, pelo fato de ter sido construída durante a pesquisa e não ter sido usada por outras

pessoas/organizações. Ao alterarmos seu código corremos o risco de perder dados, devido a erros na implementação. Uma das formas que encontramos para mitigar essa ameaça foi sempre manter duas instâncias dessa ferramenta em funcionamento, uma com a versão antiga funcional e outra com a nova versão do código. Para validar se o dado coletado pela nova versão está coerente, comparamos esse dado (agregando em um grão maior) com os dados do repositório de métricas do serviço de nuvem.

A segunda e a terceira ameaças estão relacionadas à forma como a empresa organiza e divide logicamente seus recursos na nuvem. Primeiro, aconteceram algumas mudanças na forma de usar Tags para recursos utilizados. Por exemplo, em um dado período, para requisitar um recurso não era necessário adicionar a Tag *ApplicationName* (a qual identifica a aplicação que irá utilizar o recurso). Ou seja, ainda não existia essa divisão por aplicação, apenas por produto. O outro ponto também é relacionado com a organização das Tags: algumas vezes os valores das Tags possuem grafias diferentes para identificar o mesmo recurso. Isso pode prejudicar as análises dos dados, quando se faz necessário agrupar os recursos de uma mesma aplicação. Essa última ameaça já está em processo de mitigação pela própria empresa, pois também pode atrapalhar seus processos gerenciais.

10.5 Resultados Secundários e Trabalhos Futuros

Como já citamos anteriormente, uma das contribuições desta pesquisa é o conjunto de dados coletado. Este conjunto de dados, descrito no Capítulo 6, continua a ser incrementado por meio da ferramenta de Coleta descrita no Capítulo 5. Esses dados, tais como aqueles publicados pela Google, Azure e Alibaba [30, 9, 20], podem apoiar outras pesquisas em gerência de recurso. Cabe ressaltar que esse conjunto de dados tem aspectos particulares que podem suportar pesquisas com bom potencial de impacto. Por exemplo, sua duração já é bastante longa, então podemos entender aplicações sob vários regimes de carga. O conjunto de dados representa um tipo de sistema, comércio eletrônico, e informações de demanda desses sistemas são geralmente considerados dados sensíveis pouco compartilhados. Outra característica é que temos ciência sobre a função de cada aplicação registrada nos dados (outros conjuntos de dados são alheios a isto). Ainda, no nosso conjunto de dados podemos relacionar as métricas de uso de recursos (p.ex Utilização de CPU) com métricas relacionadas

com a carga (p.ex quantidade de requisições).

Além dessa contribuição que os dados podem gerar para a comunidade, outras iniciativas conjuntas com outros pesquisadores da UFCG e engenheiros da empresa parceira já obtiveram resultados utilizando os mesmos. Dentre elas, podemos citar os seguintes resultados:

1. Em um projeto de iniciação científica realizado na UFCG (Universidade Federal de Campina Grande) e financiado pelo CNPq [1] o estudante bolsista desenvolveu um simulador de políticas de *auto-scaling* tendo como base o algoritmo da AWS. Para validar esse simulador, foram usados os dados provenientes da presente pesquisa;

2. Outra pesquisa, conduzida como parte do Trabalho de Conclusão de Curso de uma ex-aluna do curso de Ciência da Computação da UFCG, também usou os dados coletados e descritos nesse manuscrito. Desta vez, os dados foram usados para avaliar a possibilidade de uma das aplicações da empresa alvo desse estudo de caso substituir seu *auto-scaling* reativo por um preditivo. Os resultados foram bastante promissores, no sentido da predição ter tido alto grau de acerto [11].

Além dos resultados acadêmicos listados acima, também tivemos resultados com impacto na indústria. Esses resultados, descritos abaixo, foram obtidos no contexto de um projeto e parceria entre a UFCG e a empresa alvo do estudo de caso, financiado através da EMBRAPII:

1. Usando o simulador descrito anteriormente e os dados coletados, foram estudadas novas configurações de *auto-scaling* para as aplicações da empresa. Em uma dessas análises simuladas, consideramos mudanças no parâmetro *step-size* e nos limiares de utilização de CPU. Os resultados obtidos por meio dessas simulações embasaram mudanças implantadas em produção pela empresa. Como resultado dessas mudanças, a empresa conseguiu reduzir a quantidade de instâncias utilizadas pela aplicação; isso implicou em impacto financeiro direto;

2. Também considerando o simulador e os dados coletados, foi analisado o custo-benefício da mudança de limiares de utilização de CPU para uma aplicação cujo serviço tem um alto atraso para adicionar novas instâncias. Para essa aplicação, foi

[1] `https://www.gov.br/cnpq/pt-br/`

realizada uma análise retrospectiva do impacto da mudança do limiar na capacidade da aplicação atender os picos de demanda observados no passado. Como conclusão, notou-se que o aumento de custo seria proibitivo para incrementar, mesmo que de forma tímida, a cobertura para os picos de demanda;

3. Outra iniciativa, desta vez usando somente os dados coletados, envolveu uma análise de custo-benefício para a escolha de tipos de instâncias alternativas. No caso, estudo-se a troca de instâncias do tipo *x86* por instâncias do tipo *graviton*.

Dentre os trabalhos futuros, em cooperação com a empresa alvo do estudo de caso, podemos citar a continuação de análises de novas configurações de *auto-scaling*, incrementando métricas que capturam diferentes aspectos, como o custo e o SLO. Outro ponto que pode melhorar as próprias análises é ter uma forma mais segura de estimar a demanda de recursos da aplicação. Para realizar isso talvez seja necessário realizar experimentos para estimar a demanda de uma requisição e também começar a classificá-la. Seguindo os mapeamentos, outra iniciativa seria colocar em prática uso de instâncias *burstable* para mascarar tempo de rampa, e por fim testar a utilização de uma métrica de *scaling* diferente de CPU - testando combinações com métricas de aplicação. Ainda, outro, já em andamento, considera as consequências para a qualidade dos serviços de termos preempção de instâncias do mercado Spot. Os engenheiros da empresa alvo notam que o desempenho degrada quando isso acontece pois as aplicações ficam sub-provisionadas enquanto uma nova instância não é alocada para substituir a que foi removida.

Por fim, um trabalho já em andamento consiste em aprofundar o entendimento sobre as características da carga coletada. Em particular, pretendemos verificar se invariantes encontradas em cargas de serviços web ainda se mantêm. Essas invariantes para características de carga foram observadas em estudos realizados há mais de 20 anos [29]; como muitos fatores que afetam as características das cargas mudaram ao longo do tempo, é possível que as invariantes não sejam mais válidas.

Bibliografia

[1] Divyakant Agrawal, Amr El Abbadi, Sudipto Das, and Aaron J Elmore. Database scalability, elasticity, and autonomy in the cloud. In *International Conference on Database Systems for Advanced Applications*, pages 2–15. Springer, 2011.

[2] Fahd Al-Haidari, M Sqalli, and Khaled Salah. Impact of cpu utilization thresholds and scaling size on autoscaling cloud resources. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 2, pages 256–261. IEEE, 2013.

[3] Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. Auto-scaling web applications in clouds: A cost-aware approach. *Journal of Network and Computer Applications*, 95:26–41, 2017.

[4] Ataollah Fatahi Baarzi, Timothy Zhu, and Bhuvan Urgaonkar. Burscale: Using burstable instances for cost-effective autoscaling in the public cloud. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 126–138, 2019.

[5] André Bauer, Johannes Grohmann, Nikolas Herbst, and Samuel Kounev. On the value of service demand estimation for auto-scaling. In *International Conference on Measurement, Modelling and Evaluation of Computing Systems*, pages 142–156. Springer, 2018.

[6] André Bauer, Johannes Grohmann, Nikolas Herbst, and Samuel Kounev. *On the Value of Service Demand Estimation for Auto-scaling*, pages 142–156. 01 2018.

[7] Raouia Bouabdallah, Soufiene Lajmi, and Khaled Ghedira. Use of reactive and proactive elasticity to adjust resources provisioning in the cloud provider. In *2016 IEEE 18th International Conference on High Performance Computing and Communications;*

IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pages 1155–1162. IEEE, 2016.

[8] Maria Carla Calzarossa, Luisa Massari, and Daniele Tessera. Evaluation of cloud autoscaling strategies under different incoming workload patterns. *Concurrency and Computation: Practice and Experience*, 32(17):e5667, 2020.

[9] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[10] Rodrigo da Rosa Righi, Vinicius Facco Rodrigues, Gustavo Rostirolla, Cristiano André da Costa, Eduardo Roloff, and Philippe Olivier Alexandre Navaux. A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. *Future Generation Computer Systems*, 78:176–190, 2018.

[11] Marcella Medeiros Siqueira Coutinho de Almeida, Thiago Emmanuel Pereira, and Fabio Morais. A case study of proactive auto-scaling for an ecommerce workload, 2022.

[12] Alexandros Evangelidis, David Parker, and Rami Bahsoon. Performance modelling and verification of cloud-based auto-scaling policies. *Future Generation Computer Systems*, 87:629–638, 2018.

[13] Iure Fe, Rubens Matos, Jamilson Dantas, Carlos Melo, and Paulo Maciel. Stochastic model of performance and cost for auto-scaling planning in public cloud. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2081–2086. IEEE, 2017.

[14] Nikolas Herbst, Rouven Krebs, Giorgos Oikonomou, George Kousiouris, Athanasia Evangelinou, Alexandru Iosup, and Samuel Kounev. Ready for rain? a view from spec research on the future of cloud metrics. *arXiv preprint arXiv:1604.03470*, 2016.

[15] Kai Hwang, Yue Shi, and Xiaoying Bai. Scale-out vs. scale-up techniques for cloud

performance and productivity. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 763–768. IEEE, 2014.

[16] Alexey Ilyushkin, Ahmed Ali-Eldin, Nikolas Herbst, Alessandro V Papadopoulos, Bogdan Ghit, Dick Epema, and Alexandru Iosup. An experimental performance evaluation of autoscaling policies for complex workflows. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pages 75–86, 2017.

[17] Jing Jiang, Jie Lu, Guangquan Zhang, and Guodong Long. Optimal cloud resource auto-scaling for web applications. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 58–65. IEEE, 2013.

[18] Dhrub Kumar and Naveen Kumar Gondhi. A qos-based reactive auto scaler for cloud environment. In *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, pages 19–23. IEEE, 2017.

[19] Johan Linaker, Sardar Muhammad Sulaman, Martin Höst, and Rafael Maiani de Mello. Guidelines for conducting surveys in software engineering v. 1.1. *Lund University*, 2015.

[20] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2884–2892. IEEE, 2017.

[21] Sunilkumar S Manvi and Gopal Krishna Shyam. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of network and computer applications*, 41:424–440, 2014.

[22] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[23] Fábio Morais, Raquel Lopes, and Francisco Brasileiro. Provisionamento automático de recursos em nuvem iaas: eficiência e limitações de abordagens reativas. In *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2017.

[24] Fabio Jorge Almeida Morais, Francisco Vilar Brasileiro, Raquel Vigolvino Lopes, Ricardo Araujo Santos, Wade Satterfield, and Leandro Rosa. Autoflex: Service agnostic auto-scaling framework for iaas deployment models. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 42–49, 2013.

[25] Marco AS Netto, Carlos Cardonha, Renato LF Cunha, and Marcos D Assunçao. Evaluating auto-scaling strategies for cloud computing environments. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 187–196. IEEE, 2014.

[26] Alessandro Vittorio Papadopoulos, Ahmed Ali-Eldin, Karl-Erik Årzén, Johan Tordsson, and Erik Elmroth. Peas: A performance evaluation framework for auto-scaling strategies in cloud applications. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 1(4):1–31, 2016.

[27] J. Patel, V. Jindal, I. Yen, F. Bastani, J. Xu, and P. Garraghan. Workload estimation for improving resource management decisions in the cloud. In *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, pages 25–32, 2015.

[28] Chenhao Qu, Rodrigo N Calheiros, and Rajkumar Buyya. Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):1–33, 2018.

[29] Janet Siegmund, Norbert Siegmund, and Sven Apel. Views on internal and external validity in empirical software engineering. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, page 9–19. IEEE Press, 2015.

[30] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *Proceedings of the fifteenth European conference on computer systems*, pages 1–14, 2020.

[31] Raul Sidnei Wazlawick. *Metodologia de pesquisa para ciência da computação*, volume 2. Elsevier, 2009.

Apêndice A

Questionário

Entendendo o uso de auto-scaling em uma empresa de ecommerce.

Olá! Primeiramente, obrigado pelo interesse em contribuir com o nosso trabalho científico.
Objetivo principal deste questionário é entender como a os times da VTEX estão fazendo uso do mecanismo de auto-scaling e quais são suas possíveis dificuldades.

As respostas aqui obtidas serão utilizadas para como mapeamento inicial da pesquisa que Mariana Mendes, Engenheira de Software na VTEX e estudante de mestrado na Universidade Federal de Campina Grande, tem feito em pareceria com a VTEX utilizando-a como estudo de caso. Todos os dados sensíveis contidos nesse formulário foram cuidadosamente extraídos e serão cuidadosamente tratados e validados em termos de compliance e segurança antes de qualquer exposição pública ou para qualquer pessoa desprovida de um NDA com a VTEX. Além disso, apesar do formulário ser restrito para e-mails do domínio da VTEX, os dados fornecidos serão tratados com responsabilidade e ética e você não será identificado(a).

As questões são destinadas a pessoas que fazem parte dos times de engenharia na VTEX, mais especificamente as que fazem uso mais corriqueiro da conta cmc-vtex na AWS.
Se quiser saber um pouco mais sobre a pesquisa e seus objetivos, você pode acessar a proposta do mestrado através deste link:
https://drive.google.com/file/d/1VYDVwOhMTD4QbUuUV8uQg5pdReQn2JdE/view?usp=sharing

Algumas informações importantes:
O questionário é anônimo.
Responder esse questionário é um ato voluntário.
Você pode desistir de responder a qualquer momento.
O tempo estimado para responder é de 10 minutos!

Qualquer dúvida, entre em contato com Mariana:
email: mariana.mendes@vtex.com.br
slack: @mariana

*Obrigatório

Leia com atenção

Responda as perguntas deste formulário como membro do time atual informado acima e serviço desse time com o qual você possui maior familiaridade.

Perguntas relacionadas a nome de time e serviço serão utilizadas unicamente para um mapeamento. Ou seja, vai nos permitir analisar um grupo de instâncias e seu comportamento de forma mais aprofundada, pois teremos contexto de como o grupo foi configurado, dos padrões, de quais foram as dificuldades a configuração, etc. Reforçamos que os dados relacionados a nomes de time e serviço serão mantidos em total sigilo, bem como outras informações sensíveis que podem figurar nas respostas abertas do questionário.

Aplicação e Métricas

4. A carga do serviço escolhido apresenta algum padrão? (Exemplo: Pico de requisições durante a manhã; Queda de requisições das 5h às 10h)

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não
   - ◯ Não sei

5. Se sim. Qual(is) do(s) padrão(es) descreve melhor a carga que o serviço recebe? *

Marque todas que se aplicam.

- [ ] Padrão ao longo das horas do dia (Exemplo: Das 13h às 15h sempre ocorre um aumento de requisições)
- [ ] Padrão ao longo dos dias da semana (Exemplo: O sábado e o domingo geralmente ocorre um aumento de requisições)
- [ ] Padrão ao longo dos dias do mês (Exemplo: No início dos meses geralmente ocorre um aumento de requisições)
- [ ] Não apresenta nenhum padrão
- [ ] Não sei
- [ ] Outro: _____

6. Quais das métricas abaixo são as mais importantes para o seu time? *

Marque todas que se aplicam.

- [ ] Response Time
- [ ] Throughput
- [ ] Error Rate
- [ ] CPU Utilization
- [ ] Outro: _____

7. Considerando o conjunto de máquinas virtuais dedicadas ao serviço escolhido. Como você classificaria o estado típico desse conjunto? *

Marcar apenas uma oval.

- ◯ Sub provisionado.
- ◯ Adequado.
- ◯ Super provisionado.
- ◯ Outro: _____

8. Seu time usa políticas de auto-scaling para ajustar o número de instâncias na infraestrutura? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não *Pular para a pergunta 18*

Configuração do auto-scaling

9. Com que frequência a configuração dessas políticas são analisadas e atualizadas? *

Marcar apenas uma oval.

- ◯ Diariamente
- ◯ Semanalmente
- ◯ Mensalmente
- ◯ Apenas quando é notado algum comportamento anomalo
- ◯ Outro: _____

10. Qual a métrica principal que seu time considera para configurar o auto-scaling? *

Marcar apenas uma oval.

- ◯ CPUUtilization
- ◯ ProvisionedWriteCapacityUnits
- ◯ ConsumedReadCapacityUnits
- ◯ HTTPCode_Backend_4XX
- ◯ TargetResponseTime
- ◯ ReadLatency
- ◯ Outro: _____

11. Considerando a métrica escolhida, qual o valor máximo ou mínimo que essa métrica pode atingir antes da aplicação degradar? *

Caso não saiba informar responda com um 0 (zero).

_____

12. Você já configurou uma política de auto-scaling ou ajudou nessa configuração? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Outro: _____

13. Se sim, em que são baseadas as decisões de configuração?

Marcar apenas uma oval.

- ◯ Experimentos com a aplicação
- ◯ De forma intuitiva
- ◯ Baseado em decisão de outros times
- ◯ Não se aplica/Nunca configurei

14. Quais os parâmetros de configuração do auto-scaling você acha que interferem nas métricas de desempenho importantes para seu time? *

Marque todas que se aplicam.

- ☐ Tipos de Instância (instanceTypes)
- ☐ Limiar superior (upperThreshold)
- ☐ Limiar inferior (lowerThreshold)
- ☐ Tamanho máximo do grupo de auto-scaling (maxSize)
- ☐ Tamanho mínimo do grupo de auto-scaling (minSize)
- ☐ Cooldown
- ☐ Não sei opinar
- ☐ Outro: _____

15. Já ocorreu algum problema de desempenho devido a uma configuração de auto-scaling? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Outro: _____

16. Se sim, este é um campo livre para descrever a situação problema.

_____

_____

_____

_____

_____

17. O que considera difícil em relação ao uso do auto-scaling? *

_____

Custo e Dúvidas.

18. Existe demanda para redução de custos dentro do seu time? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Outro: _____

19. Campo livre para deixar quaisquer dúvidas sobre o questionário.

Apêndice B

Respostas do Questionário

Pergunta realizada no Questionário: "O que considera difícil em relação ao uso do *auto-scaling*"

- **Momento de *trigger* de ações.** Nessa primeira categoria estão as respostas que apontam como maior dificuldade decidir o melhor momento para adicionar ou remover recursos.
  - "Definir a relação tempo x *threshold* pra evitar escalar com muita antecedência ou muito atrasado"
  - "Acho que entender e otimizar as condições para escalar e desescalar (sic) as máquinas"superdimensionadas na maior parte do tempo, para durante o pico ficarem adequadamente provisionadas."
  - "Principalmente equilibrar o custo. Manter a aplicação com folga de processamento é importante para aguentar picos. Mas é preciso sempre considerar os custos ao tomar esse tipo de decisão."
- **Tempo de "rampa".** Nessa categoria estão as respostas que apontam que o tempo em que a máquina demora para ficar disponível é um problema.
  - "O tempo que a nova máquina demora para ficar online às vezes é um problema, mesmo sendo alguns minutos"
  - "Temos que considerar o tempo para a máquina inicializar e se ficar healthy."

  - "Demora para startar novas instâncias"
  - "Ele é muito raso e demora a provisionar máquinas. A load que temos aumenta em quase 10x a cada hora por alguns minutos, o que leva a uma variância razoável e as máquinas terem que ficar

- **Configurar regras/conhecimento do sistema.** Nesse tópico as respostas estão relacionadas à dificuldades para configurar o *auto-scaling* por falta de algum conhecimento mais aprofundado do serviço.

  - "Uma configuração inicial dessas regras sem ter experiência prévia com o sistema talvez fique meio na intuição, mas com a maturidade do sistema isso fica mais fácil"
  - "O uso do *auto-scaling* em si (como configurar e como funciona) acho tranquilo. Agora saber um valor exato de configuração para um sistema X é um problema difícil, já que sistemas podem ter comportamentos estranhos."
  - "*Fine tuning* dependendo da aplicação."
  - "O *tunning* de acordo com a *stack*, tipo de servidor e aplicação."
  - "Acredito que seja definir o perfil da aplicação e do uso, e com base nisso definir as métricas adequadas. "
  - "O *tunning* (sic) de qual o melhor valor considerando padrões de tráfego, tempo para escalar, e manter uso adequado de cpu."

- **Coleta/Análise.** Nesse último tópico foram agregadas as respostas que indicam maior dificuldade para experimentar e avaliar resultados de uma configuração.

  - "A Experimentação para definir parâmetros"
  - "A configuração em si é algo bem intuitivo, a coleta de evidências que ratificam a escolha que costuma ser algo mais complexo"
  - "Termos uma forma fácil de analisar/manter a configuração por dados em vez de por problemas/achismos"

Apêndice C

Exemplo de Configuração

```
{
    "metricsDescription": [
        {
            "metricName": "CPUUtilization",
            "namespace": "AWS/EC2",
            "dimension": "InstanceId",
            "statistics": [
                "Average",
                "Maximum",
                "Minimum"
            ]
        },
        {
            "metricName": "Latency",
            "namespace": "AWS/ELB",
            "dimension": "LoadBalancerName",
            "statistics": [
                "Average"
            ]
        }
    ],
    "endTime": "DEFINIDO-AUTOMATICAMENTE",
```

```
    "period": "60",
    "startTime": "DEFINIDO-AUTOMATICAMENTE",
    "storage": {
        "InstanceId": "instancias",
        "LoadBalancerName": "elbs"
    },
    "aws-config": {
        "region": "us-east-1",
        "bucket": "meu-bucket"
    }
}
```