UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

FRANCISCO TIERRY BARROS OLIVEIRA

UMA ANÁLISE DA QUALIDADE DE UM SERVIÇO DE ARMAZENAMENTO BASEADO EM MÉTRICAS DE MONITORAMENTO DA AWS

QUIXADÁ

2022

FRANCISCO TIERRY BARROS OLIVEIRA

UMA ANÁLISE DA QUALIDADE DE UM SERVIÇO DE ARMAZENAMENTO BASEADO EM MÉTRICAS DE MONITORAMENTO DA AWS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Emanuel Ferreira Coutinho

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O47a Oliveira, Francisco Tierry Barros.
Uma análise da qualidade de um serviço de armazenamento baseado em métricas de monitoramento da AWS / Francisco Tierry Barros Oliveira. – 2022.
55 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2022.
Orientação: Prof. Dr. Emanuel Ferreira Coutinho.

1. Computação em Nuvem. 2. Benchmarks (Computação). 3. Amazon Web Services. I. Título.

CDD 005

FRANCISCO TIERRY BARROS OLIVEIRA

UMA ANÁLISE DA QUALIDADE DE UM SERVIÇO DE ARMAZENAMENTO BASEADO EM MÉTRICAS DE MONITORAMENTO DA AWS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em: ____/ ____/ ____.

BANCA EXAMINADORA

Prof. Dr. Emanuel Ferreira Coutinho (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Marcelo Uchôa de Alencar
Universidade Federal do Ceará (UFC)

Prof. Dr. Jefferson de Carvalho Silva
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus que me deu forças para vencer todas as dificuldades passadas nos últimos anos e por ter me mantido perseverante durante essa jornada.

A minha família, irmãos, tios, primos e especialmente a minha mãe Maria Barros por ter me dado todo apoio e confiança para nunca desistir.

Ao orientador deste trabalho, Prof. Dr. Emanuel Coutinho, por sua orientação, suporte e atenção, as quais foram essenciais para conclusão deste trabalho.

A todos meus amigos, em especial Jhonattan Nascimento e Vitória Gomes que me proporcionaram anos de convivência incríveis, os quais jamais esquecerei, além de todo apoio educacional, motivacional e afetivo.

A Universidade Federal do Ceará - Campus de Quixadá pela oportunidade de me proporcionar conhecimento e aprendizagem tanto técnica como pessoal, as quais me fizeram crescer profundamente.

“Se você não trabalha para construir os seus sonhos, você irá trabalhar para construir os de outra pessoa.”

(Desconhecido)

RESUMO

Considerando que estamos presenciando cada vez mais um crescimento explosivo de dados, a computação em nuvem se torna responsável por gerir boa parte dessas informações, principalmente quando falamos em sistemas de armazenamento. Porém, muitas vezes é necessário lidar com grandes volumes de dados perante a diversas situações de uso, as quais podem afetar negativamente o ambiente em que estão inseridos. Nesse contexto, a pesquisa realizada neste trabalho visa efetuar um *benchmark* no Amazon S3, utilizando o JMeter como ferramenta de teste de carga, o qual se comunica com uma aplicação local de armazenamento de imagens que está integrada com a API do S3. Desta forma o intuito é coletar elementos impactantes das métricas através do CloudWatch e verificar sua correlação diante dos atributos de qualidade da ISO/IEC 25010.

Palavras-chave: Computação em Nuvem. Benchmarks (Computação). Amazon Web Services.

ABSTRACT

Considering that we are witnessing an explosive growth of data, cloud computing becomes responsible for managing much of this information, especially when we talk about storage systems. However, it is often necessary to deal with large volumes of data in the face of different usage situations, which can negatively affect the environment in which they are inserted. In this context, the research carried out in this work aims to perform a *benchmark* on Amazon S3, using JMeter as a load testing tool, which communicates with a local image storage application that is integrated with the S3 API . In this way, the intention is to collect impacting elements of the metrics through CloudWatch and verify their correlation with the quality attributes of ISO/IEC 25010.

Keywords: Cloud computing. Benchmarks (Computing). Amazon Web Services.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Durante a última década, a Computação em Nuvem e os seus serviços tornaram-se um componente importante e uma ampla referência para a organização do trabalho que envolve um grande volume de dados (KHALIL *et al.*, 2021), tendo consequências que envolvem uma série de razões técnicas, nas quais é possível incluir a melhoria da eficiência energética, otimização da utilização de recursos tanto de *hardware* como de *software*, elasticidade de recursos, isolamento de desempenho, flexibilidade e utilização de serviço sob demanda das requisições (GIUSEPPE *et al.*, 2013).

Com a expansão global acelerada dos ambientes de nuvem e a partir de um grande volume de informações, é possível se obter dados estatísticos imensuráveis, como uso de processamento, uso de memória, capacidade de armazenamento, tempo de resposta, entre outros, já que ambientes de nuvem podem variar e armazenar seus elementos em diferentes regiões e dispositivos simultaneamente. Portanto, por ser uma tecnologia em constante evolução ainda há dúvidas e questionamentos específicas sobre plataformas, técnicas e ferramentas para monitorar infraestruturas de nuvem, serviços e aplicativos.

Para mensurar e organizar esses dados estatísticos são necessários serviços apropriados, os quais estão disponíveis de diversos modos e métodos, como o monitoramento, que é o processo de avaliação da integridade das estruturas de tecnologia da informação (TI), o qual auxilia as organizações em diversos segmentos como disponibilidade e otimização, além da identificação de padrões e descoberta de potenciais riscos de segurança na infraestrutura (HOWELL, 2018).

Apesar disso, existem alguns problemas envolvidos e estão diretamente interligados ou são consequências do ato de não monitorar, como soluções obsoletas a partir da análise dos dados, demanda de tempo para diagnosticar algo, e alto consumo financeiro. Porém, o mais significativo é o baixo desempenho, dado que ele impacta diretamente no funcionamento adequado de outros serviços da aplicação armazenada em nuvem, gerando assim um acúmulo de objeções que passam muitas vezes despercebidas pelos desenvolvedores, testadores e gestores de uma empresa.

Visando a necessidade dos serviços de nuvem que possam vir a ter aplicações com baixo desempenho ou problemáticas posteriores relacionados que causem instabilidade ou inatividade do mesmo, é possível notar que essas aplicações irão impactar diretamente no custo de sua manutenção, como mostra Infonetics (2015) em sua pesquisa com 205 empresas de TI

de médio a grande porte na América do Norte, onde essas empresas estão perdendo até $ 100 milhões por ano diante de seus clientes, devido ao tempo de inatividade da rede, do aplicativo ou servidor respectivamente. Isso demonstra que ainda existe uma alta demanda por serviços que impactem diretamente a qualidade do serviço, pois vai refletir diretamente no crescimento que uma organização deseja alcançar.

Portanto, para solucionar esses problemas, diversas alternativas de serviços surgem, que tornam o monitoramento mais funcional e também eficiente. Dentre eles o CloudWatch[1] da Amazon ganha destaque, pois conta com uma variedade de funcionalidades que complementam o seu uso, como sua coleta de dados em três formas diferentes e personalizadas, definição de alarmes, ações automatizadas entre outros aspectos que o tornam performático suficiente para analisar dados e aplicações de diversos tamanhos e tipos.

Na literatura existem diversas normas e padrões relacionadas a atributos de qualidade. Muitas vezes elas auxiliam a definir critérios de qualidade para os ambientes computacionais, possibilitando analisar a qualidade do serviço executado nos ambientes. Dentre as normas existentes, a ISO/IEC 25010[2] foca na qualidade de produto de *software*, definindo modelos de avaliação da qualidade de *software* e sistemas.

Diante disso, este trabalho tem o intuito de realizar uma análise de qualidade do serviço, tendo como referência a norma da ISO/IEC 25010 (ISO/IEC 25010, 2011). Para isso será construída uma aplicação web de armazenamento de imagens em pequena proporção para ser observada, onde a mesma irá utilizar serviços da Amazon Web Services (AWS), como o CloudWatch, para gerar as métricas de monitoramento, além de possuir visualização unificada do site, e o Amazon S3[3] para o armazenamento dos objetos da aplicação. A partir disso, informações vão sendo geradas, baseadas na utilização da aplicação pelo usuário. Assim, será possível visualizar a real necessidade dos serviços de monitoramento, e sua capacidade de mudança diante de situações cruciais para toda aplicação, como otimização de recursos, prudência sobre os gastos, prevenção e recuperação de falhas.

1.1 Objetivos

Nesta seção serão apresentados o objetivo geral e específico do trabalho.

[1] CloudWatch - https://aws.amazon.com/pt/cloudwatch/
[2] ISO/IEC 25010 - https://iso25000.com/
[3] Amazon S3 - https://aws.amazon.com/pt/s3/

1.1.1 Objetivo geral

O objetivo geral deste trabalho é analisar os recursos de uma aplicação em Computação em Nuvem visando a qualidade do serviço com foco no armazenamento.

1.1.2 Objetivo específicos

- Desenvolver uma arquitetura que permita organizar a aplicação na nuvem;
- Desenvolver uma aplicação a ser avaliada;
- Identificar serviços a serem avaliados;
- Identificar métricas a serem coletadas.

2 TRABALHOS RELACIONADOS

Nesta seção serão apresentados alguns trabalhos relacionados ao tema, junto a uma descrição de cada um deles, por fim é abordada uma comparação entre ambos, afim de demonstrar a correlação com o trabalho proposto.

2.1 Cloud Resource Management – Virtual Machines Competing for Limited Resources

O artigo de Grant e Eluwole (2013) procura destacar que um gerenciamento eficaz de recursos melhora a qualidade de serviço (QoS - *Quality of Service*) oferecidas pelos provedores de serviços em nuvem, como também reduz os custos de consumo, melhora o tempo de processamento e resposta, e afeta positivamente o desempenho final os usuários.

Em sua pesquisa foram utilizadas duas simulações de nuvem, o CloudSim e CloudAnalyst, com o objetivo de ver como as máquinas virtuais se comportam em diferentes cenários de uso e em diferentes ambientes de implementação. Assim foram utilizados os três sistemas operacionais mais nomeados atualmente, Mac, Linux e Windows rodando em instâncias do Amazon Elastic Compute Cloud (EC2), tendo os recursos analisados através do CloudWatch.

Para consolidar seu estudo, o autor usou o Facebook em três casos de uso diferentes e dois cenários, um com processador (CPU - *Central Process Unit*) mais lento e outro com processador mais rápido. O resultado foi surpreendente, embora a CPU mais forte tenha certamente sido mais rápida em execução que a fraca, ela chegou a ter o mesmo custo da CPU mais fraca em dois dos três casos de uso. Isso graças ao monitoramento que o CloudWatch oferece, sendo possível realizar um balanceamento de carga, assim demonstrando que o gerenciamento de recursos não é apenas fornecer serviços de pagamento conforme a utilização, mas sim atender a demanda do cliente e do acordo de nível de serviço (SLA - *Service Level Agreement*).

2.2 Metric selection and anomaly detection for cloud operations using log and metric correlation analysis

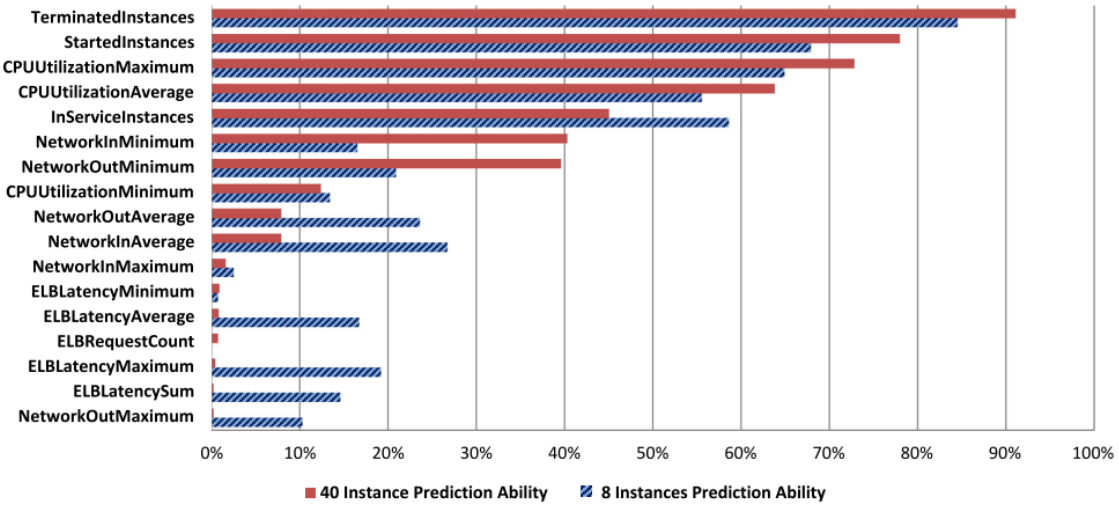
Na pesquisa de Farshchi *et al.* (2018) é desenvolvida uma abordagem para detectar erros em determinadas operações em registros de atividades nos recursos de nuvem, primeiro apresentando uma abordagem de seleção de métricas e um conjunto de experimentos com diferentes configurações na Amazon Web Services. Para os autores, o principal foco é realizar o monitoramento e garantir a confiabilidade em operações esporádicas, sendo elas a realização

de *backup*, atualização (contínua), migração em nuvem, reconfiguração e uma série de outras operações.

Para conduzir esse experimento foi usado o Amazon EC2 e também o CloudWatch como fonte de monitoramento de métricas de recursos. Assim vai ser possível investigar se a adoção de uma técnica de análise de *log* com uma técnica baseada em regressão é prática para modelar a relação entre o comportamento da operação da nuvem e os estados variáveis dos recursos da nuvem. Com isso é seguido alguns passos iniciais como a realização de um mapeamento de log-métrico, visto que os eventos podem ser registrados com uma frequência de frações de segundos ou vários minutos, e em seguida esses eventos são agrupados. Após agrupá-las é realizado uma correlação estatística definida como uma ferramenta estatística para medir o grau ou a força da associação entre duas, ou múltiplas variáveis, e a partir disso é feita seleção de métrica alvo, que basicamente consiste em definir as métricas de monitoramento.

Após essas etapas foi realizado um resultado experimental que envolvia aprendizagem de *clustering* de *logs* por coeficiente de Pearson, aprendizagem de correlação e causalidade com regressão múltipla, entre outros fatores, realizados em dois casos de uso, um com 8 instâncias EC2, atualizando duas instâncias por vez e o outro executando atualização contínua de 40 instâncias, atualizando quatro instâncias por vez, gerando resultados conforme é demonstrado na Figura 1.

Figura 1 – Capacidade de previsão para cada métrica de monitoramento



Fonte: Farshchi *et al.* (2018).

Apesar de terem sido realizados outros testes como asserções de tempo de execução, conclui-se que esse trabalho é importante para quem buscar monitorar ambientes de nuvem, visto

que ajuda a encontrar o subconjunto com as métricas de monitoramento mais relevantes, bem como análise dessas métricas para a garantia confiável de execução, pois estudos futuros terão que passar por etapas como esta, de modo a demonstrar a aplicabilidade da sua abordagem.

2.3 Cloud monitoring for optimizing the QoS of hosted applications

Na pesquisa de Alhamazani *et al.* (2012) é conduzido um estudo com intuito de buscar uma abordagem e metodologia relacionada a desenvolver novas técnicas inteligentes e escalonáveis no monitoramento de nuvem, de modo a automatizar o gerenciamento de QoS (*Quality of Service*) de aplicativos hospedados, visto que a falha ou congestionamento na rede são inevitáveis, dada a escala, dinâmica e complexidade de um determinado ambiente, assim determinar como se adaptar a condições imprevisíveis é um problema inerentemente complexo para o qual várias implicações técnicas devem ser consideradas. Com isso, o foco da sua proposta é aplicar dinamicamente componentes de aplicativos monitorados e parâmetros de QoS para seu controle automático em condições de indeterminação.

Neste contexto, três atividades principais foram elaboradas. A primeira consiste em um monitoramento descentralizado onde a ideia é implementar um modelo de rede escalonável para coleta e criação de perfil de informações de QoS monitoradas. A segunda atividade é um monitoramento de QoS baseado em política e criação de perfil, onde a ideia consiste em desenvolver um serviço genérico para monitoramento eficiente de QoS e um perfil de estado dos serviços de nuvem em camadas múltiplas da rede, sendo um desafio, pois tem que identificar os parâmetros de QoS mais importantes e o modo de interação que as empresas e desenvolvedores gostariam de monitorar. Por último, a terceira atividade visa desenvolver novos modelos de previsão de desempenho de serviço de nuvem e carga de trabalho de aplicativo, tendo como base os avanços recentes em técnicas estatísticas computacionais.

Considerando os fatores apresentados foi possível tirar algumas contribuições como a de uma técnica escalonável baseada em tabelas *hash* distribuída para monitorar o comportamento de componentes, além de novas interfaces de usuário e linguagem de política, o que permitirá que usuários não especialistas em nuvem expressem seus parâmetros de QoS específicos de aplicativos de uma maneira contínua. Com isso, os autores concluíram que monitoramento em nuvem é um fator-chave, e consequentemente, descobrir quais parâmetros os afetam é muito importante.

2.4 Observing the clouds: a survey and taxonomy of cloud monitoring

No trabalho de Ward e Barker (2014) é realizada uma pesquisa das ferramentas de monitoramento contemporânea em nuvem, em que foi realizada uma taxonomia, a qual examina a eficácia com que as ferramentas e projetos existentes atendem aos desafios do monitoramento em nuvem. Primeiramente é levantado alguns aspectos que vão ajudar a trazer contexto a pesquisa, assim é apresentado um conjunto de requisitos para estruturas de monitoramento de nuvem, pois elas são usadas para estabelecer a eficácia das ferramentas de monitoramento atuais, como um ambiente escalável, autônomo, com tolerância a erro entre outros aspectos.

Com os aspectos estabelecidos o autor parte para uma apresentação de análises, na qual é demostrado ferramentas de monitoramento ou ferramentas que tentem atuar com monitoramento, sendo projetadas para Computação em Nuvem, seja ela como um serviço ou não, assim é apresentado todo um histórico de ferramentas, desde as mais antigas até as mais atuais, com o intuito de mostrar sua evolução e como essas ferramentas atuavam usando elementos em comum, e a partir desses elementos foi possível chegar na taxonomia, pois esses elementos em comum constam como os componentes são arquitetados e como eles se comunicam, assim é possível criar uma taxonomia que demonstre a motivação ou origem por trás da ferramenta e o caso de uso principal da ferramenta.

Essa taxonomia foi elaborada a partir da classificação da arquitetura, mecanismo de comunicação, mecanismo de coleta, origem e caso de uso de cada ferramenta, assim foi possível elencar todas as ferramentas de monitoramento que foram citadas, de modo a se criar uma classificação do que cada uma usa e posteriormente saber os requisitos atendidos por cada ferramenta, como é ilustrado na Figura 2.

A partir disso foi possível tirar algumas conclusões como análise do potencial futuro das ferramentas, onde com a diversificação de provedores de nuvem, APIs, aplicativos e outros fatores, se tornará cada vez mais difícil desenvolver ferramentas que abranjam todas as áreas da Computação em Nuvem, com isso é provável que as soluções de monitoramento futuras serão compostas de várias ferramentas que podem ser integradas e alternadas para fornecer um monitoramento mais abrangente.

Figura 2 – Requisitos atendidos por sistemas gerais de monitoramento

		Cloud monitoring requirements						
		Scalable	Cloud aware	Fault tolerant	Multiply granular	Comprehensive	Time sensitive	Autonomic
Monitoring systems	Astrolabe	✓		✓				
	Cacti					✓		
	collectd	✓				✓		
	Ganglia	✓		✓				
	GEMS	✓		✓				
	Icinga					✓		
	MonaLISA	✓				✓		
	Nagios					✓		
	OpenNMS					✓		
	Reconnoiter	✓						
	Riemann	✓						✓
	StatsD	✓			✓			
	sFLow					✓		
	visPerf	✓		✓	✓			
	Zabix					✓		
	Zenoss					✓		

Fonte: Ward e Barker (2014).

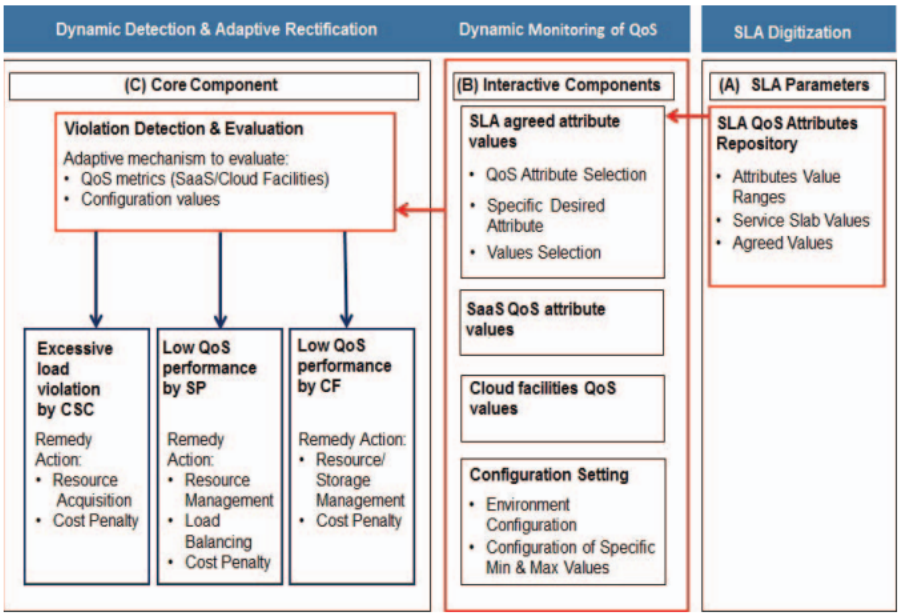
2.5 An adaptive monitoring framework for ensuring accountability and quality of services in cloud computing

No trabalho de Khan *et al.* (2016) é apresentado uma estrutura de monitoramento adaptável para monitorar métricas de QoS e medidas de desempenho, o que é equivalente ao trabalho proposto, pois conta com um conjunto de métricas que serão monitoradas visando neste caso a qualidade do serviço.

A estrutura para execução do trabalho consiste em três componentes principais, sendo digitalização de parâmetros de SLA, onde é definido um conjunto de parâmetros de QoS, sendo armazenados em um repositório. Em seguida possui os componentes interativos, que consiste em quatro unidades que contêm valores de atributos de QoS, sendo valores de atributo acordados com SLA, valores de atributos de QoS no Software como Serviço (SaaS – *Software as a Service*), valores de atributos de QoS das instalações em nuvem e definição de configuração. E por último o componente central, que vai reunir todas as métricas dos componentes interativos anteriores e avalia os valores, afim de encontrar violações dos SLAs, assim é criado um fluxo como é ilustrado na Figura 3.

Com isso são realizados alguns testes com foco em tempo de resposta e disponibilidade entre um consumidor e um provedor SaaS, e foi possível notar que algumas violações estavam sendo detectadas, seja do lado do consumidor ou provedor, porém o mais importante

Figura 3 – Estrutura de monitoramento adaptável para serviços confiáveis em nuvem



Fonte: Khan *et al.* (2016).

é que foi possível detectar e definir de qual lado a violação ocorre devido à execução continua dos componentes. Assim se conclui que a proposta apresentada detecta violações nos SLAs e que o objetivo futuro é criar um *framework* em uma plataforma *Cloud* que realize esse trabalho, pois assim vai ajudar o monitoramento a ter mais uma ferramenta que o auxilie na detecção de problemas.

2.6 Comparações

Os trabalhos apresentados são bastante extensos, trazendo perspectivas e situações diferentes as quais estão relacionadas com o trabalho proposto, como o trabalho de Grant e Eluwole (2013), que é o mais semelhante possível a este estudo, pois aborda plataformas de serviços, aplicações para serem testadas como caso de uso e serviço de monitoramento com foco na QoS, porém não tem como foco uma padronização internacional que vise melhorar a qualidade do produto. Já os estudos de Alhamazani *et al.* (2012) e Khan *et al.* (2016) são mais amplos, não envolvendo diretamente plataformas ou aplicações, pois tenta buscar novas formas de monitoramento que sejam mais eficientes ao decorrer do tempo, porém também com foco na QoS apenas.

Enquanto os trabalhos de Farshchi *et al.* (2018) e Ward e Barker (2014) apesar de terem as mesmas características são bem diferentes, ambos não envolvem uma aplicação de caso de uso, e não possuem objetivo na QoS e ISO, apesar disso eles servem como pesquisa para

projetos futuros, os quais envolvam o ambiente de nuvem e sua correlação, como é o caso deste trabalho proposto, com isso contribuem de diversas maneiras para evitar estudos já apresentados inicialmente, como por exemplo a escolha de métricas.

Com isso foi elaborado a Tabela 1 que traz a comparação entre alguns pontos que compõem a relação entre os trabalhos apresentados, sendo a plataforma de serviço, que está diretamente interligada aos provedores, seja eles Amazon, Microsoft, IBM ou Google. Aplicação de caso de uso, que visa usufruir de uma aplicação específica para realizar testes de monitoramento. CloudWatch como sendo o serviço principal usado no estudo proposto. A QoS como garantia de execução das aplicações em nuvem de forma confiável e rápida e por último a ISO como forma de padronização internacional para garantia da qualidade do produto.

Tabela 1 – Comparativo entre os trabalhos relacionados e o proposto

	Plataforma de serviço	Aplicação de caso de uso	CloudWatch	QoS	ISO
Grant e Eluwole (2013)	Sim	Sim	Sim	Sim	Não
Farshchi *et al.* (2018)	Sim	Não	Sim	Não	Não
Alhamazani *et al.* (2012)	Não	Não	Não	Sim	Não
Ward e Barker (2014)	Sim	Não	Sim	Não	Não
Khan *et al.* (2016)	Não	Não	Não	Sim	Não
Trabalho proposto	Sim	Sim	Sim	Sim	Sim

Fonte: elaborado pelo autor (2021).

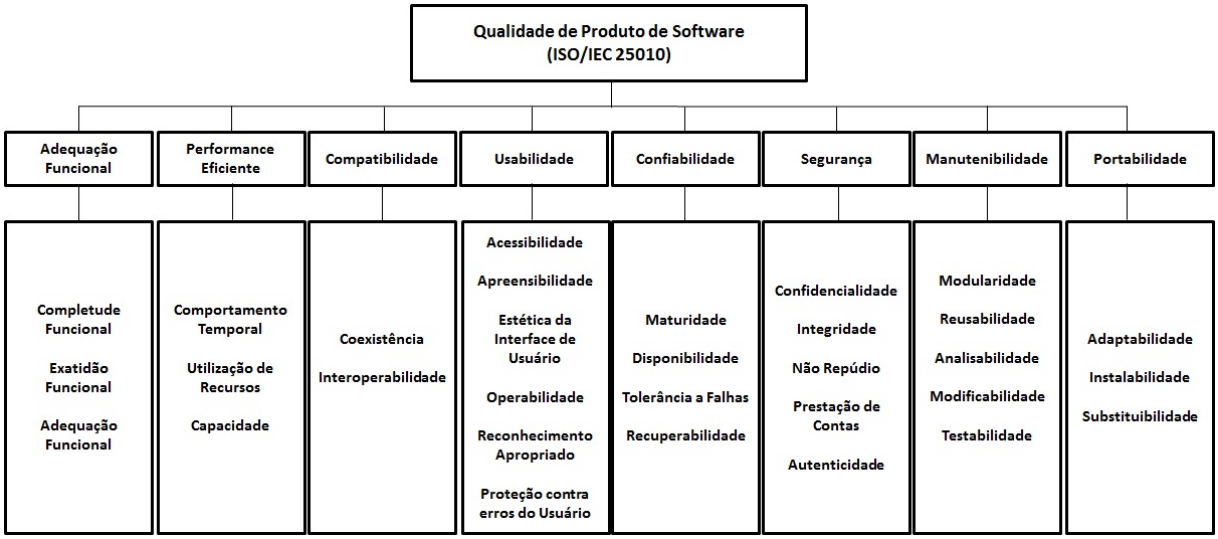
3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os principais conceitos que vão ser necessários no desenvolvimento do trabalho, começando pelo conceito da Organização Internacional para Padronização (*International Organization for Standardization* - ISO), sua subdivisão e suas características. Em seguida é apresentado o conceito de Computação em Nuvem. Logo após a apresentação sobre a Amazon Web Services, suas funcionalidades e características. E por último os serviços principais da nuvem que serão usados através da AWS, o CloudWatch, onde será apresentado sua funcionalidade e aspectos que o complementam, e também o Amazon S3 com sua arquitetura.

3.1 ISO/IEC 25010

Embora existam diversas padronizações e normas atualmente, a ISO é uma organização renomada, atuando desde 1947 de forma internacional promovendo a normatização de empresas e produtos. Desta forma, a ISO/IEC 25010 tem como foco a qualidade, pois ela é a base de um sistema de avaliação da qualidade do produto de software (ISO/IEC 25010, 2011). Com isso esta ISO possui um conjunto de oito características de qualidade apresentadas na Figura 4.

Figura 4 – Características de qualidade da ISO/IEC 25010



Fonte: ISO/IEC 25010 (2011).

Embora todas as características sejam termos bem amplos, todas envolvem subcaracterísticas, fazendo com que os termos mais gerais possam ser aprofundados de uma forma

concisa. Começando pela Adequação Funcional que representa o grau em que um produto ou sistema fornece funções que atendem às necessidades declaradas e implícitas quando usado sob condições especificadas. A Performance Eficiente que representa o desempenho em relação à quantidade de recursos usados nas condições estabelecidas. A Compatibilidade se refere ao grau em que um produto, sistema ou componente pode trocar informações com outros enquanto compartilha o mesmo ambiente de hardware, ou software. Já a Usabilidade também se refere ao grau em que um produto ou sistema pode ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso especificado. A Confiabilidade se refere ao grau em que um sistema, produto ou componente executa funções especificadas sob condições especificadas por um período de tempo especificado. Em seguida a Segurança com relação ao grau em que um produto ou sistema protege informações e dados de forma que pessoas, ou outros produtos ou sistemas tenham o grau de acesso aos dados apropriados para seus tipos e níveis de autorização. A Manutenibilidade representando o grau de eficácia e eficiência com que um produto ou sistema pode ser modificado para melhorá-lo, corrigi-lo ou adaptá-lo às mudanças do ambiente e dos requisitos. E por último a Portabilidade que é relacionada ao grau de eficácia e eficiência com que um sistema, produto ou componente pode ser transferido de um hardware, software ou outro ambiente operacional de uso para outro.

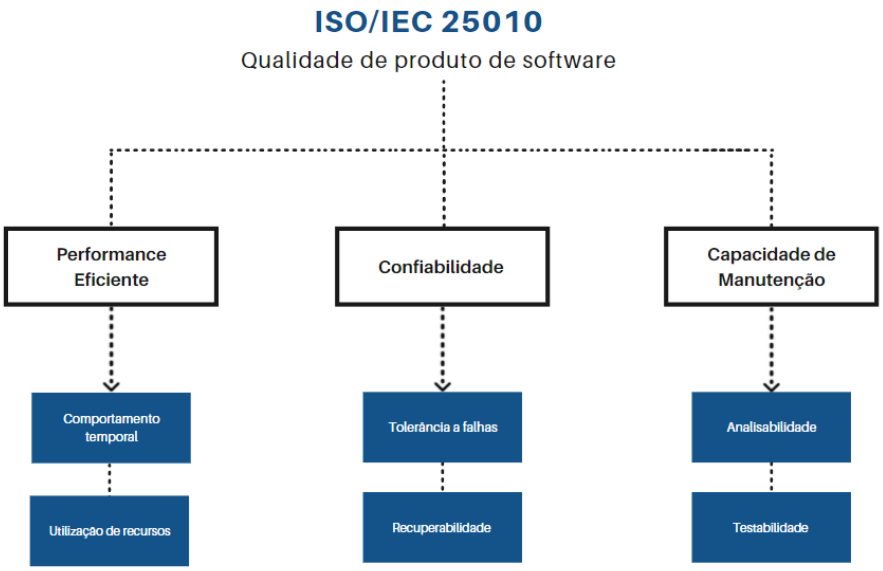
3.1.1 Características empregadas da ISO/IEC 25010

Considerando todo o contexto em que esse trabalho se propõe a fazer, e considerando que a qualidade de software é algo muito amplo e envolve muitas variáveis, este trabalho irá abordar apenas três das oito características citadas, conforme estão listadas na Figura 5. Foram escolhidas essas três características, pois são consideradas essenciais para atingir a eficiência em um sistema, bem como as que mais podem impactar na qualidade do *software*: Performance Eficiente, Confiabilidade e Capacidade de Manutenção (Manutenibilidade).

Assim começamos pela Performance Eficiente, tendo como subcaracterística o Comportamento Temporal, que é o grau em que os tempos de resposta, processamento e as taxas de transferência de um produto ou sistema atendem aos requisitos ao executar suas funções, e em seguida a Utilização de Recursos que é o grau em que as quantidades e tipos de recursos utilizados por um produto ou sistema, no desempenho de suas funções, atendem aos requisitos.

Como segunda característica temos a Confiabilidade, tendo como subcaracterística a Tolerância a Falhas, que se refere a um sistema, produto ou componente que opera conforme

Figura 5 – Características que serão usadas da ISO/IEC 25010



Fonte: Elaborado pelo autor.

planejado, apesar da presença de falhas de *hardware* ou *software*, e seguindo temos a Recuperabilidade que se refere a um produto ou sistema que pode recuperar os dados diretamente afetados e restabelecer o estado desejado em caso de interrupção ou falha.

E por último temos a Capacidade de Manutenção, possuindo como suas subcaracterísticas a Analisabilidade que tem foco em avaliar o impacto em um produto ou sistema de uma alteração pretendida em uma ou mais de suas peças, ou diagnosticar um produto quanto a deficiências ou causas de falhas, e a Testabilidade que examina a eficácia e eficiência com o qual os critérios de teste podem ser estabelecidos para um sistema, produto ou componente e determinar se esses critérios foram atendidos.

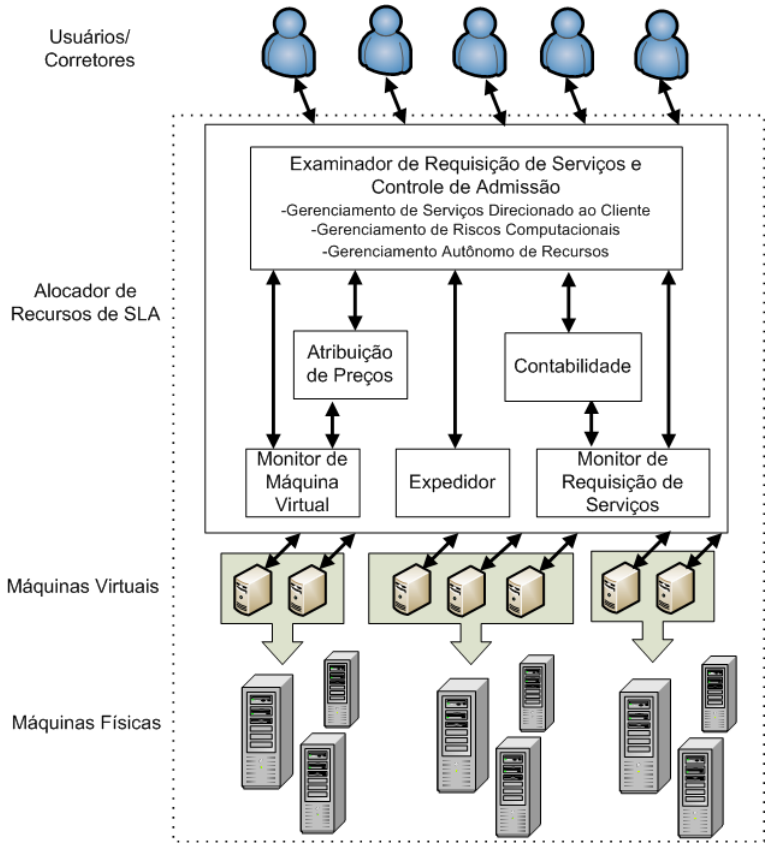
Todos esses critérios são importantes e impactam diretamente um sistema, como por exemplo o tempo de resposta, já que em um ambiente de nuvem essa subcaracterística é fundamental para que as requisições sejam rápidas e atendam a alta demanda que um sistema possa vir a ter, segundo Grant e Eluwole (2013), assim um dos intuitos do trabalho é utilizar serviços como CloudWatch, focando na aprendizagem e proteção do usuário contra erros, através da criação de eventos que a ferramenta possui, assim impactando a eficácia, eficiência e satisfação neste contexto de uso especificado, onde vai ser possível diagnosticar com precisão oque ocorreu e quando ocorreu. Com isso conseguimos atingir boa parte do objetivo do trabalho, além de ser possível analisarmos com maior precisão as métricas de nuvem que serão estabelecidas no

decorrer do estudo.

3.2 Computação em Nuvem

A Computação em Nuvem é uma mudança de paradigma na forma como os recursos de *hardware* e *software* são gerenciados e utilizados (SURURAH *et al.*, 2021). Logo se torna um termo amplo e a explicação para isso é a própria nuvem, sendo uma enorme rede de servidores ou mesmo computadores individuais interconectados, onde esses computadores funcionam em paralelo para combinarem recursos conforme a demanda dos usuários, onde realizam as requisições que passam pelo alocador de recursos, que funciona como uma *interface* entre um provedor de serviço na Nuvem e seus Usuários/Corretores para realizarem os mecanismos de verificação, como mostra a Figura 6.

Figura 6 – Arquitetura de Computação em Nuvem Orientada a Mercado



Fonte: Felipe *et al.* (2015)

Durante muito tempo a Computação em Nuvem teve muitas definições na comunidade acadêmica como cientifica, porém, a mais amplamente usada é a do Instituto Nacional de Padrões e Tecnologia (NIST - *National Instutite of Standards and Technology*)[1], onde expressa

[1] NIST - https://csrc.nist.gov/publications/detail/sp/800-145/final

que a Computação em Nuvem é um modelo para acesso à rede sob demanda, ubíquo e conveniente para um conjunto combinado de recursos computacionais configuráveis que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços.

Para melhorar seu entendimento e melhor subdividi-la, o instituto elencou cinco características que o compõe, que apesar de simples são essenciais, sendo a primeira o auto-serviço sob demanda, onde o consumidor pode suprir por conta própria os recursos de computação, seja armazenamento, processamento, tempo de servidor entre outros, de acordo com sua utilização e sem a necessidade de intervenção humana dos provedores de serviços. A segunda é o amplo acesso por rede, que basicamente consiste na acessibilidade, onde os recursos estão disponíveis na rede sendo acessados por diversas plataformas como *smartphones*, *tablets*, *laptops* ou *desktops*. A terceira característica é o agrupamento de recursos em que os provedores agrupam recursos para atender a múltiplos consumidores em modalidade da arquitetura de multilocação, com recursos físicos e virtuais diferentes dinamicamente atribuídos e reatribuídos conforme a demanda de seus consumidores. Já em quarto foi elencado a elasticidade rápida onde os recursos podem ser provisionados e liberados elasticamente, em alguns casos automaticamente, para rapidamente aumentar ou diminuir conforme a demanda do consumidor. Por último temos o serviço mensurado, que se baseia em controlar e otimizar automaticamente o uso dos recursos através de medições em um nível de abstração apropriado para a categoria de serviço nos sistemas de nuvem.

De modo a categorizar modos de entrega de serviços em nuvem o instituto também dividiu os modelos de serviços em três, onde cada uma possui sua particularidade, sendo a primeira o Software como Serviço (SaaS – *Software as a Service*) que consiste no uso de aplicações do fornecedor executando em uma infraestrutura na nuvem sendo fornecidas ao consumidor. A segunda sendo a Plataforma como Serviço (PaaS – *Platform as a Service*) que visa instalar na infraestrutura da nuvem aplicativos criados ou adquiridos pelo consumidor, desenvolvidos com linguagens de programação, bibliotecas, serviços e ferramentas suportados pelo fornecedor ou compatíveis. E por último a Infraestrutura como Serviço (IaaS – *Infrastructure as a Service*) que corresponde a provisionar processamento, armazenamento, comunicação de rede e outros recursos de computação fundamentais nos quais o consumidor pode instalar e executar *softwares* em geral, incluindo sistemas operacionais e aplicativos.

Visando completar a categorização o NIST ainda elenca quatro modelos de desen-

volvimento, sendo a nuvem privada, voltada ao uso exclusivo de uma organização com vários consumidores. A nuvem comunitária sendo para uso exclusivo por uma determinada comunidade de consumidores de organizações com interesses em comum. A nuvem pública que é a que usamos comumente, sendo aberta ao público. E por último a nuvem híbrida que compõe de duas ou mais infraestruturas na nuvem, seja ela privada, comunitária ou pública pertencendo a entidades distintas.

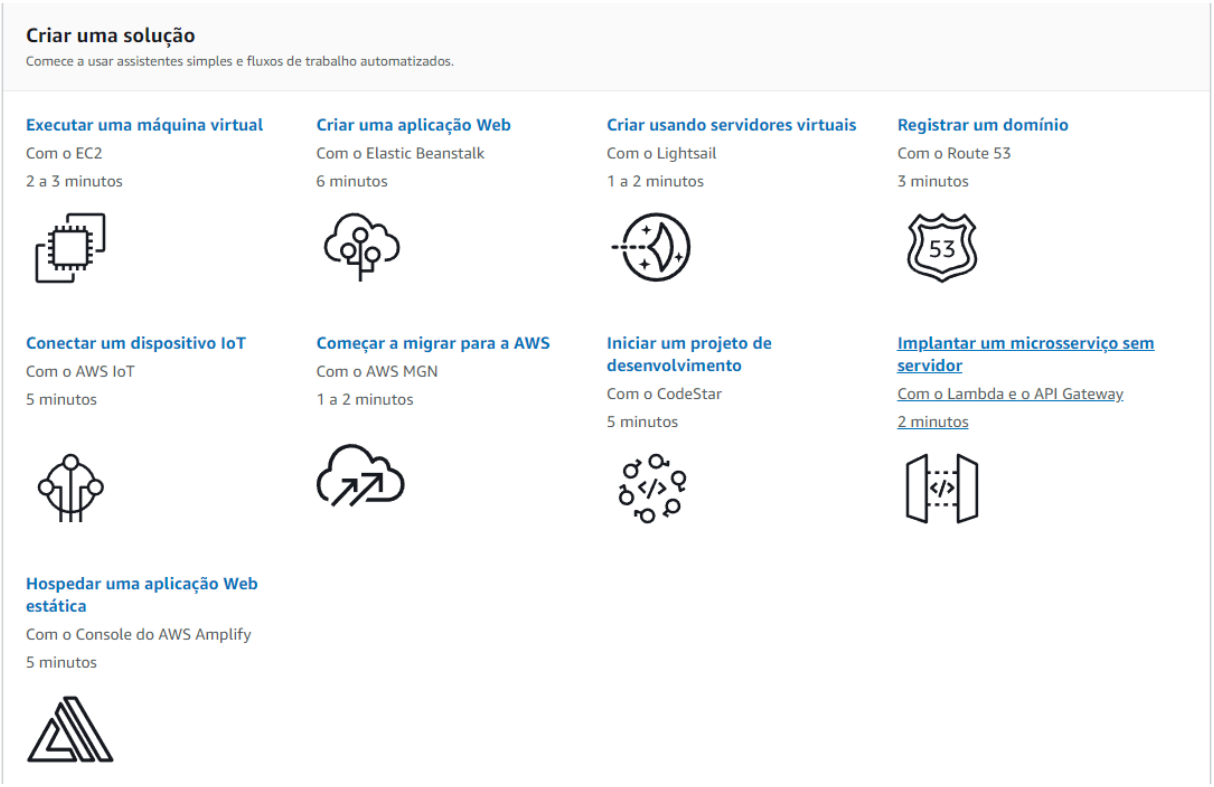
3.3 Amazon Web Services

Considerando todo o cenário atual em que vivemos de Computação em Nuvem, é impossível não destacar as plataformas de serviços para otimizar a rotina das empresas, e quando se fala em otimização desse serviço devemos destacar a *Amazon Web Services* (AWS), por algumas características, dentre elas a acessibilidade, desempenho e disponibilidade, já que além de possuir uma rede global. A AWS é usada por mais de 2062 empresas, dentre elas empresas grandes como Netflix, Coca-Cola, McDonald's, Ubisoft, entre outras, além de ser amplamente usada e citada em artigos acadêmicos, por esses e outros diversos benefícios são a plataforma escolhida para execução deste estudo.

Outro ponto bastante forte dessa plataforma é a ampla possibilidade de estruturas oferecidas e também de soluções como mostra na Figura 7, sendo muitas vezes algo prático e intuitivo para quem tem pouca familiaridade com os ambientes de nuvem, sendo possível realizar tarefas que seriam complexas em poucos minutos e de forma automatizada, dando destaque a serviços como EC2 que cria e gerência diversas maquinas virtuais simultaneamente, e o Elastic Beanstalk que realiza a implantação e escalabilidade de aplicações e serviços da web desenvolvidos com Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker em servidores familiares como Apache, Nginx, Passenger e IIS. Isso demonstra uma pequena parte do que essa plataforma consegue executar de forma performática e acessível.

Por outro lado, considerando o fator para quem busca soluções mais robustas e sofisticadas também é oferecido uma gama muito ampla de escolhas de acordo com cada necessidade, e um exemplo disso é o seu serviço de armazenamento multirregional, conhecido por Amazon S3 que será usado neste trabalho, pois além de oferecer alto desempenho e escalabilidade, vai fornecer recursos de gerenciamento fáceis de usar, de maneira que seja possível organizar os dados e configurar controles de acesso refinados para atender a requisitos específicos deste trabalho.

Figura 7 – Alguns dos serviços oferecidos pela AWS



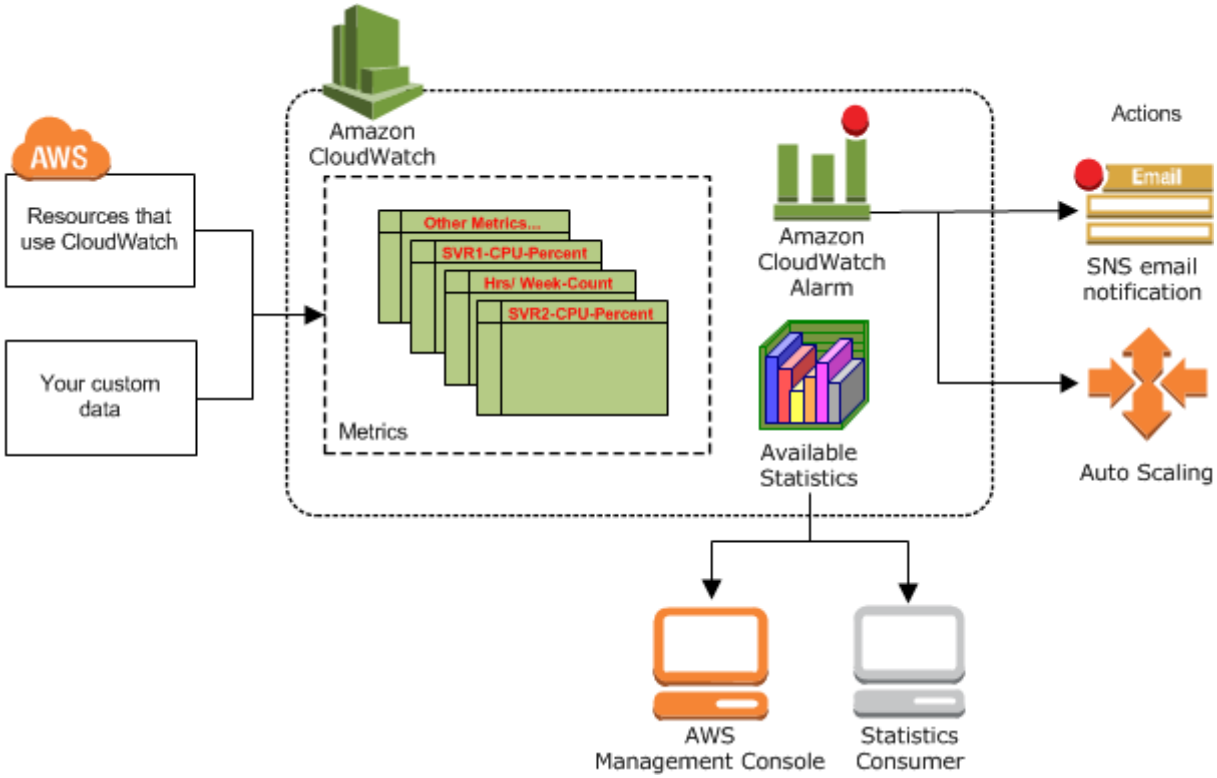
Fonte: AMAZON WEB SERVICES (2021).

3.4 CloudWatch

Com a alta demanda de aplicações executando simultaneamente em um ambiente de nuvem, surge a necessidade de gerenciá-los com intuito de criar um repositório de métricas, como a própria AWS nomeia, pois, com isso vai ser possível recuperar as estatísticas de forma rápida e organizada com base nessas métricas. Visando esse caso, a AWS oferece o CloudWatch que é um serviço de monitoramento e observação que aprovisiona dados e *insights* práticos para monitorar aplicativos, responder às alterações de desempenho em todo o sistema, otimizar a utilização de recursos e obter uma visualização unificada da integridade operacional.

A arquitetura do CloudWatch foi pensada de forma que os recursos ou seus dados sejam usados diretamente na AWS de forma que gere métricas, como tempo de resposta, tempo das requisições, uso de CPU, uso de memória entre outros, assim o usuário que utiliza o serviço do CloudWatch terá disponível todas as tarefas que podem ser executadas se baseado nas métricas geradas, conforme é demonstrado na Figura 8. Sua execução segue dois fluxos, sendo o primeiro de alarme que por sua vez executa ações pré-programadas como o *Auto Scaling* que monitora os aplicativos e ajusta automaticamente a capacidade para manter um desempenho constante e previsível pelo menor custo possível, e o outro fluxo é a avaliação das estatísticas formadas,

Figura 8 – Arquitetura do CloudWatch



Fonte: AMAZON WEB SERVICES (2021).

mostrando em formato detalhado os dados de consumo, no qual será o intuito desse estudo, assim será possível ter uma ampla visão de gerenciamento e análise da aplicação hospedada de forma unificada.

3.5 Amazon Simple Storage Service

O Amazon S3 é um serviço oferecido pela AWS de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e desempenho líderes do setor. Sua liderança é por possuir recursos muito eficientes como gerenciamento amplo e otimização de custos, além de escalabilidade de recursos. É um mecanismo de armazenamento de objetos desenvolvido para armazenar e recuperar qualquer quantidade de dados de qualquer local.

O Amazon S3 disponibiliza uma interface Web para armazenar e recuperar qualquer quantidade de dados, a qualquer momento, de qualquer lugar. Usando esse serviço, é possível criar aplicações que fazem uso de armazenamento nativo em nuvem. Como o Amazon S3 é altamente escalável e só se paga pelo que usa, é possível começar com uma aplicação pequena e expandi-la da forma desejada, sem comprometer a performance ou a confiabilidade.

Como é observado na Figura 9, sua arquitetura é bastante simples, podendo ser usado

Figura 9 – Arquitetura do Amazon S3

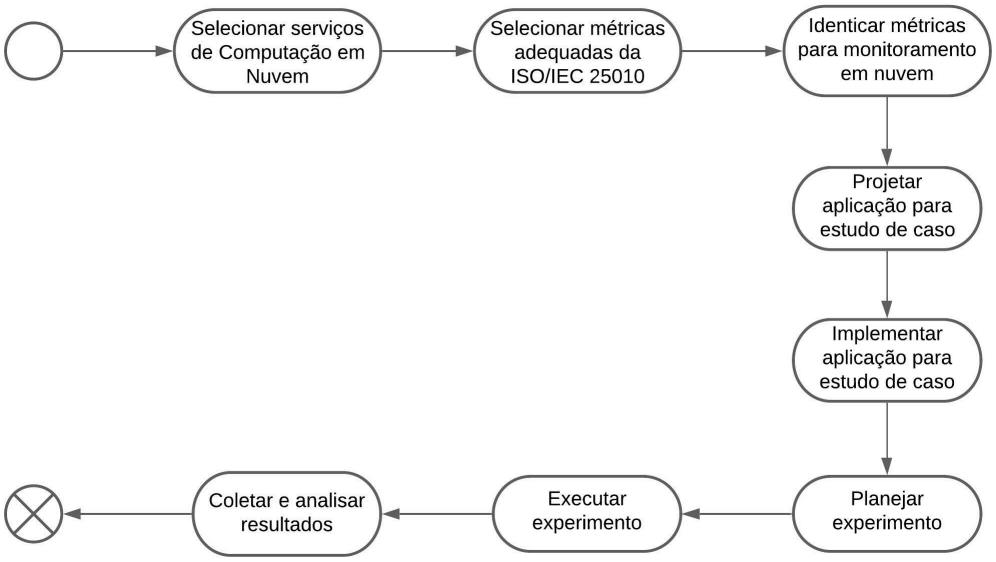
Fonte: AMAZON WEB SERVICES (2021).

para armazenar desde dados analíticos até *backup* de arquivos, onde serão armazenados dentro de um objeto denominado *bucket*, que possui uma série de funcionalidades como controle de acesso, proteção de dados, mudança de região entre outras que permitem ao *bucket* ser facilmente gerenciado pelo usuário. Por fim esses dados armazenados no *bucket* podem ser extraídos por serviços da própria AWS como CloudWatch ou serviço de terceiros de modo a analisar os dados e obter estatísticas sobre seu uso e armazenamento.

4 PROCEDIMENTOS METODOLÓGICOS

Com a finalidade de executar oque foi proposto, nesta seção capítulo é apresentado os passos a serem executados para conclusão deste trabalho. Com isso esta pesquisa foi dividida em oito etapas, como é apresentada na Figura 10.

Figura 10 – Fluxo das atividades que serão executadas



Fonte: Elaborado pelo autor.

4.1 Selecionar serviços de Computação em Nuvem

Para se iniciar o estudo é essencial definir quais ferramentas atendem a necessidade de uma análise de Computação em Nuvem, sendo essas ferramentas uma de monitoramento dos dados e outra para armazenar os dados. É necessário que ambas ferramentas sejam da mesma plataforma de serviço, pois essas ferramentas vão trabalhar em conjunto de modo a alcançar oque foi proposto. A princípio, será utilizada a plataforma de Computação em Nuvem da AWS, e seus serviços S3 e CloudWatch.

4.2 Selecionar métricas adequadas da ISO/IEC 25010

Em seguida, também se faz necessário definir quais métricas serão usadas para análise, pois a ISO/IEC 25010 (ISO/IEC 25010, 2011) engloba muitas características e variáveis que envolve a qualidade do produto de *software*, onde é fundamental definir quais delas são

realmente importantes e quais causam impacto no estudo. Considerando que o objetivo do trabalho é o monitoramento, serão definidas métricas que estejam diretamente relacionadas a essa característica e estejam atreladas aos efeitos que implicam na qualidade do *software*.

4.3 Identicar métricas da nuvem para monitoramento

Diferente dos atributos da ISO/IEC 25010, as métricas de monitoramento da nuvem são algo mais específico. Assim é identificado quais dessas métricas impactam os termos mais gerais da ISO, como por exemplo tempo de resposta, quantidade de solicitações, requisições HTTP, entre outras. Com isso serão definidas as métricas mais relevantes para se analisar dentro da AWS, juntamente ao CloudWatch.

4.4 Projetar aplicação para estudo de caso

Para se ter embasamento no que foi proposto é necessário algo para analisar, com isso é fundamental projetar uma aplicação WEB para executar o estudo. Assim será construído uma arquitetura que vise demonstrar desde o ambiente de nuvem e suas conexões até a aplicação, dessa forma será possível desenvolver um site de armazenamento de imagens que será hospedado na nuvem usando os serviços da AWS pré estabelecidos anteriormente, como o Amazon S3 para armazenar os objetos, na qual será feito a partir de requisitos funcionais e não funcionais estabelecidos. Através desse site será possível o usuário hospedar uma imagem até certo tamanho, obter um link de *download* da imagem, e excluir a imagem hospedada. Todos esses requisitos fazem com que o CloudWatch analise a aplicação constantemente de forma que se gere métricas que serão posteriormente analisadas.

4.5 Implementar aplicação para estudo de caso

Após projetar a aplicação é necessário implementá-la. Para isso será usado o ambiente de execução Javascript, que é o NodeJs sendo usado para construir o *backend* que estará integrado à nuvem através do Amazon S3. Para finalizar, será construído o *frontend* usando a biblioteca React e de modo a facilitar a efetivação dos testes, a aplicação será executada localmente, sem possuir interferência de ambientes externos.

4.6 Planejar experimento

Nesta etapa é necessário organizar o que sera executado, repassando por todos os pontos citados anteriormente, começando pela organização das métricas estabelecidas, tanto da ISO como da nuvem. Com isso estabelecido, é necessário integrar o site ao CloudWatch de forma que seja possível monitorar suas execuções, além de criar casos de uso específicos para executar a aplicação de modo que as métricas que impactem diretamente no funcionamento do *software* possam ser coletadas.

4.7 Executar experimento

Apos toda a elaboração dos passos e para validar o estudo, a aplicação será executada em alguns dispositivos diferentes e com conexões diferentes, de modo a ser o mais próximo da realidade e conforme os casos estabelecidos no planejamento para ser possível extrair dados necessários suficientes para análise.

4.8 Coletar e analisar resultados

Durante a execução do experimento, por meio do CloudWatch os dados são obtidos e armazenados na plataforma de nuvem. Assim, esses dados serão coletados e interpretados através da geração de gráficos ilustrativos que o próprio serviço do CloudWatch fornece. A análise consiste em verificar o desempenho de um ambiente computacional e sua qualidade diante de situações prejudiciais para execução da aplicação.

5 PROJETO DO EXPERIMENTO

Nesta seção serão apresentadas as configurações do experimento, ferramentas, implementações e análise dos resultados obtidos durante o processo de execução.

5.1 Estudo do cenário

Inicialmente antes de qualquer passo, foi necessário estudar sobre o processo de *benchmark*, bem como sua importância diante de aplicações em computação em nuvem. Percebeu-se que a análise computacional vem se tornando cada vez mais comum, visto que estão surgindo e sendo disponibilizados cada vez mais serviços em nuvem para os usuários, onde dependem de diversos fatores como região, categoria de armazenamento e valor. Diante disso torna-se necessário investigar sobre esses ambientes, para análise da qualidade, estudos e contribuições futuras ou testes aplicados a cenários reais, como de uma aplicação que será usada em grande escala.

Visando todos os fatores apresentados foi decidido realizar um *benchmark* sintético, o qual usa um determinado programa que estimula o serviço a ter um certo tipo de comportamento ou ação desejada pelo usuário. Para visualizar e analisar como se comportam seus recursos em determinados cenários de uso.

Tendo em vista esse cenário foi escolhido como ferramenta de *benchmark* o JMeter[1], que é um programa que realiza testes de desempenho, tornando-se referência nesse segmento. Desta forma foi possível testar diferentes configurações diante da infraestrutura disponível, visando definir a escala de execução que será feita, como também conhecer e configurar as diferentes possibilidades de execução que a ferramenta disponibiliza. Um exemplo foi o *Thread Group*, que foi essencial para realização do trabalho, pois possibilitou a geração de um encadeamento de execuções simultâneas sem afetar estritamente o *hardware* do ambiente de execução. Além de executar múltiplas requisições HTTP por meio IP, sendo fundamentais para interligarem a simulação local da aplicação de imagens.

5.2 Configuração do *benchmark*

Neste tópico será abordado sobre a configuração na AWS, como sua integração com a aplicação, além das configurações realizadas no Jmeter.

[1] JMeter - https://jmeter.apache.org/

5.2.1 Configurações realizadas na AWS e na aplicação

Considerando diversos fatores, entres os principais a possível cobrança de uso da nuvem e seu desempenho, a configuração e execução foi realizada através do ambiente da AWS Academy[2], que utiliza a região us-east-1, sendo a mesma configurada na aplicação. Considerando esse fator a primeira configuração realizada foi extrair as *key* de acesso do *Sandbox Environment* e setá-las manualmente na aplicação, sendo um passo essencial, pois as *key* que vão permitir a *lib* aws-sdk se conectar com a nuvem.

Objetivando extrair o máximo desempenho de execução e de tempo durante os testes, a aplicação foi executada localmente, evitando interferências provocadas por outros ambientes de nuvem, como um site de hospedagem, por exemplo. Desta forma restou apenas setar o nome do *bucket* na aplicação, sendo uma opção escolhida propositalmente, visando uma maior organização da execução e controle dos objetos que serão armazenados.

Diferente de uma conta pessoal, o *Identity and Access Management* (IAM) não precisa ser configurado dentro da AWS, pois, a conta da Academy já vem definido um usuário com as políticas anexadas de leitura e gravação no S3. Desta forma foi realizado a criação do *bucket* cujo o nome deve corresponder com o definido no *backend* da aplicação, além de conter o desbloqueio de acesso ao público, já que a AWS não consegue identificar quais programas serão executados para prover o armazenamento.

5.2.2 Configurações realizadas no JMeter

Como o Jmeter é uma ferramenta de desempenho, é comum que disponibilize uma série de configurações de modo que o usuário possa defini-la conforme necessário. Como o objetivo é executar requisições no S3, foi essencial usar o *Thread Group* para simular um número X de usuários simultâneos, ocorrendo um *loop* entre eles durante a execução.

Estabelecido o número de conexões foi necessário configurar de onde será extraído esse número. Com isso foi definido o *HTTP Request* como forma de execução, já que se está executando a aplicação em NodeJS, que cria uma determinado URL local. Desta forma são definidos o protocolo, nome do servidor, porta, além do caminho ou rota que a aplicação tenha que acessar com sua solicitação específica, ressaltando que deve ser extremamente essencial marcar o multipart/form-data que é o corpo da requisição na solicitação POST. Após esses passos,

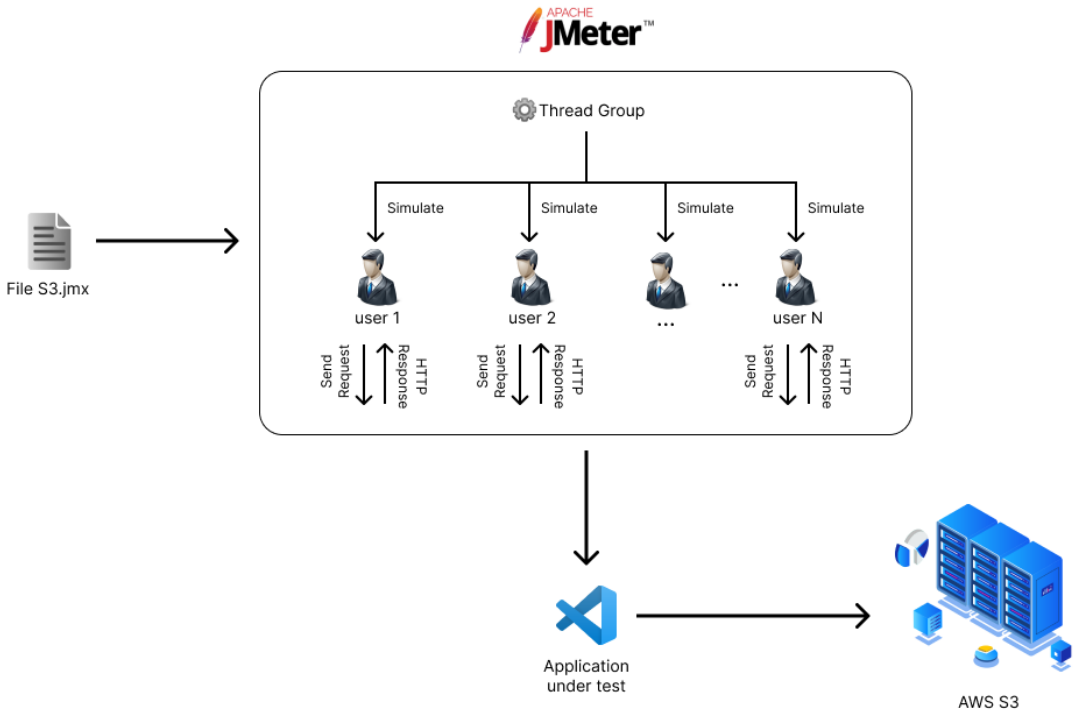
[2] AWS Academy - https://aws.amazon.com/pt/training/awsacademy/

foi definida uma imagem genérica no formato JPEG e com tamanho fixo de 51KB, visando tornar as requisições mais rápidas e precisas.

Embora não seja algo de extrema necessidade, em testes desse porte é recomendado usar os ouvintes do JMeter, que são os resultados parciais dos testes providos a partir do seu *benchmark*. Dessa forma é possível verificar e analisar caso uma requisição HTTP falhe ou tenha problemas relacionados a comunicação, além de fornecer um log de tudo que foi executado durante o processo. Com o ambiente pré configurado, todo ele foi salvo no formato .jmx, para que toda a configuração não precise ser refeita, além de auxiliar caso a execução seja sem "interface" gráfica ou outro ambiente computacional.

Com esses fatores estabelecidos todo o fluxo de execução foi definido conforme a Figura 11. Começando pela inserção do arquivo .jmx no Jmeter, que ao começar o processo de execução vai se comunicar com o *VSCode* que tem a aplicação rodando localmente através do terminal, assim a aplicação executa se comunicando com o S3 conforme o Jmeter solicita.

Figura 11 – Arquitetura de execução do projeto.



Fonte: Elaborado pelo autor.

5.3 Correlação com a ISO

A ISO/IEC 25010 possui muitas propriedades e características (ISO/IEC 25010, 2011). Porém neste trabalho, apenas três características foram definidas conforme exibido na Figura 5, pois são particularidades importantes para um ambiente de nuvem, e principalmente para um sistema de armazenamento como o S3.

Desta forma o primeiro atributo é a Performance Eficiente que relaciona tudo sobre recursos e utilização de tempo, estando relacionado a métricas como latência, tempo de execução ou taxa de transferência, que serão visualizadas facilmente através do JMeter no decorrer das execuções, pois são métricas que estão diretamente correlacionadas aos ambientes de nuvem.

A Confiabilidade diz respeito ao comportamento de um sistema, se o mesmo se comporta conforme o esperado em casos de falhas, por exemplo, e se essas falhas causam interrupções ou afetam o ambiente de forma negativa. Como neste *benchmark* o JMeter será usado, gerando um arquivo de log, será possível extrair informações necessárias para analisar as requisições que tiveram falhas ou não, e se perante essas falhas o sistema se comportou de forma esperada.

E por último a Capacidade de Manutenção que é a característica mais complexa de medir ou analisar, se vermos o significado da palavra, visto que ela diz respeito a diagnosticar o sistema perante as falhas, porém isso é muito complexo de investigar, pois diversos fatores podem influenciar um ambiente de armazenamento, como região, permissões, valores e tipo do *bucket*, restando apenas levantar suposições e possíveis acontecimentos que venham a ocorrer futuramente. Também nesse ponto tem-se a Testabilidade que vai examinar os critérios estabelecidos em testes, se encaixando no mesmo aspecto do anterior, onde pode ser medido com maior precisão e eficacia em acontecimentos futuros, os quais realizem testes de *benchmark* mais amplos e com uma maior variedade.

6 RESULTADOS OBTIDOS

O projeto se baseou em cinco requisições principais, conforme mostrado na Quadro 1. As requisições foram divididas dessa forma devido a limitações de tempo e *hardware*, conforme serão explicadas no tópico de conlusão. Essas execuções foram realizadas em uma segunda-feira na parte da tarde, tendo intervalo de 20 minutos entre cada uma delas, para uma melhor visualização dos dados e para sua coleta.

Foi pensando nisso que o *benchmark* foi dividido em cinco etapas, onde a primeira executa por volta de 1000 requisições simultâneas, para início da execução, e a partir dela são feitas etapas com intervalos de 20.000, assim a segunda etapa contém por volta de 20.000 requisições, a terceira 40.000, a quarta 60.000 e a quinta e última 80.000 requisições.

Quadro 1 – Quantidade de requisições esperadas no Jmeter.

Execuções	Quantidade	Loops	Threads
Primeira	1000	1	1
Segunda	20.000	10	5000
Terceira	40.000	20	5000
Quarta	60.000	30	5000
Quinta	80.000	40	5000

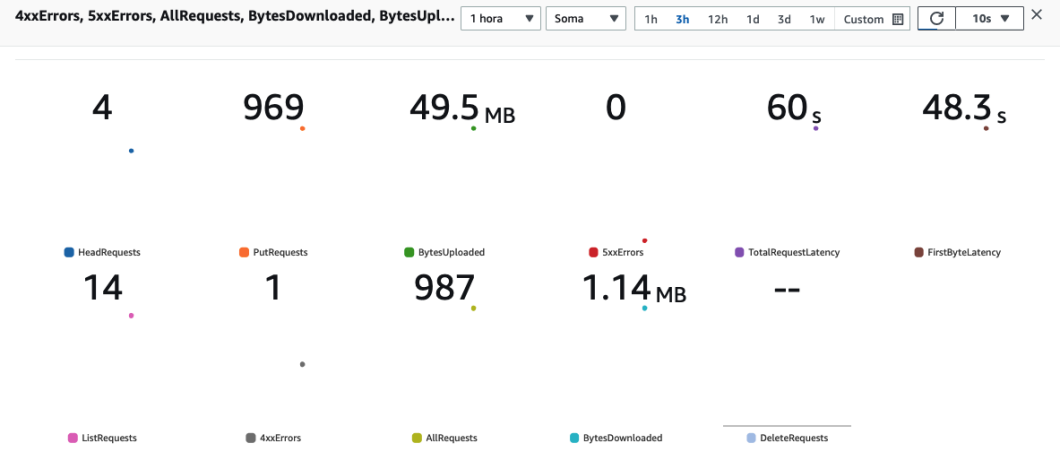
Fonte: Elaborado pelo autor.

É importante destacar que *Loops* e *Threads* são duas configurações do Jmeter, sendo *Thread* a simulação de um usuário e *Loop* o número de vezes que a *Thread* fará a solicitação. Porém, o *Ramp-Up*, que é a quantidade de tempo que o Jmeter deve levar para obter todos os *Threads* enviados para a execução, foi definido como 1 segundo, ou seja 1/5000 resultando na tentativa de execução de 10, 20, 30 e 40 requisições em 0,0002 segundos para o segundo, terceiro, quarto e quinto teste respectivamente. Desta forma as solicitações não serão distribuídas uniformemente pelo tempo de *Ramp-up*, ou seja as requisições serão iniciados de uma só vez sem qualquer atraso, resultando nas quantidades de solicitações presentes no quadro.

6.1 Primeira execução

O ponto de partida veio de uma requisição simultânea que buscou realizar 1000 inserções no S3, levando 22 segundos de execução no Jmeter. Dessas 1000 inserções, foram inseridas 969 com sucesso no *bucket*, conforme é exibido na Figura 12 do CloudWatch, mostrando a soma de tudo que ocorre no *bucket*. Também pode-se notar no gráfico da Figura 13, que exibe

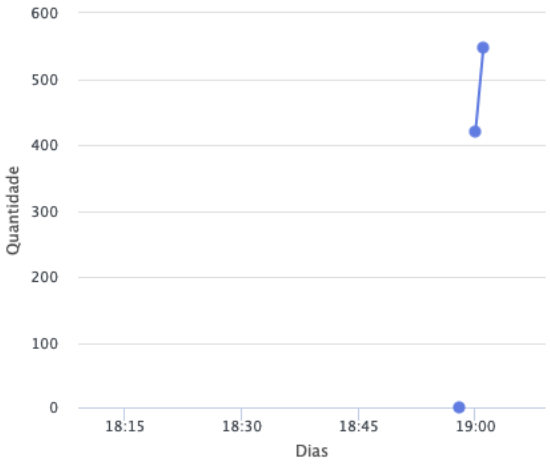
Figura 12 – Gráfico de soma do CloudWatch referente ao bucket do S3 - primeira execução.



Fonte: Elaborado pelo autor.

a requisição PUT, tendo a primeira solicitação por volta de 18:50, e outras duas posteriormente, resultando nas 969 inserções.

Figura 13 – Gráfico de requisições PUT feitas ao bucket do S3 - primeira execução.



Fonte: Elaborado pelo autor.

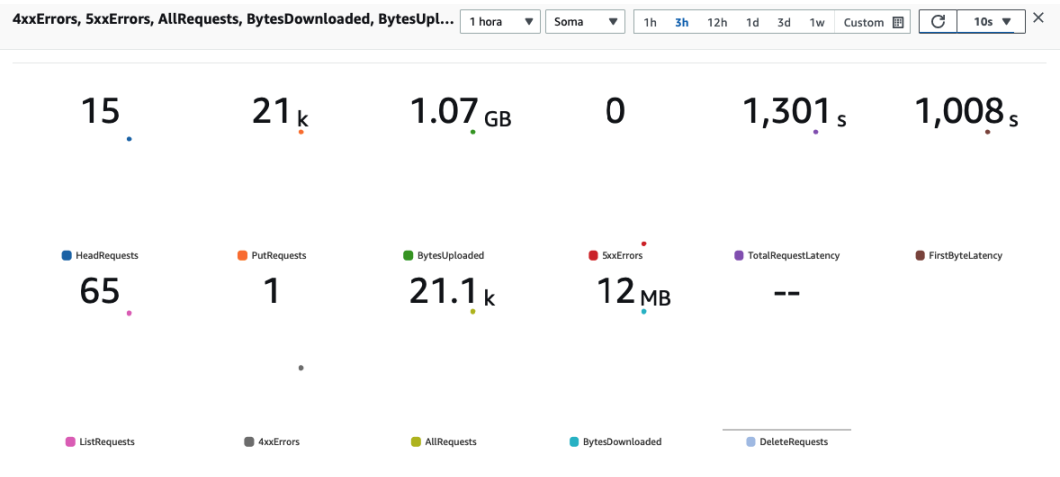
Também foi possivel notar através do Jmeter que esta execução teve uma taxa de transferência de 2,714.195 solicitações/minuto, em que esse número inclui quaisquer atrasos adicionados ao teste e o próprio tempo de processamento interno do JMeter, sendo de extrema importância pois apresenta algo real, comprovando que o servidor de fato lidou com essa quantidade de solicitações por minuto.

6.2 Segunda execução

A segunda requisição buscou realizar por volta de 20.000 inserções no S3, mais precisamente 20.111, levando 3 minutos e 50 segundos de execução no Jmeter, executada em 10 *loops* de 5000 *Threads*. Dessas 20.111 inserções, foram inseridas 20.026 com sucesso no *bucket*, tendo assim a somatória da requisição anterior com esta, conforme é apresentado na Figura 14 do CloudWatch, exibindo a soma de tudo que ocorre no *bucket*.

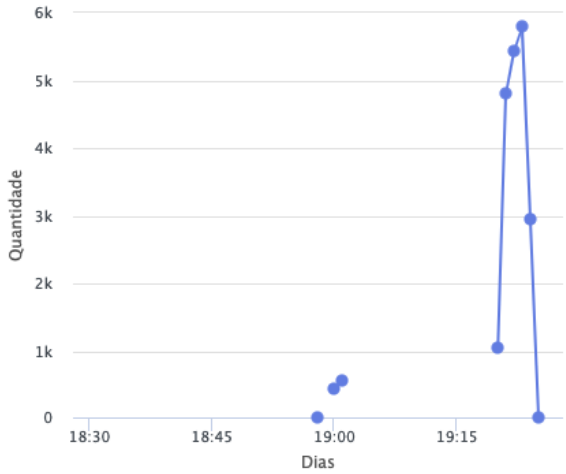
Também pode-se notar no gráfico da Figura 15 da requisição PUT, tendo a primeira solicitação por volta de 18:50, e outras duas posteriormente. Após o intervalo de 15 minutos, é possível notar a alta variação do gráfico, pois foi onde começou a segunda requisição, sendo distribuída em partes ate concluir a quantidade estabelecida, tendo assim uma taxa de transferência um pouco maior de 3,459.126 solicitações/minuto.

Figura 14 – Gráfico de soma do CloudWatch referente ao bucket do S3 - segunda execução.



Fonte: Elaborado pelo autor.

Figura 15 – Gráfico de requisições PUT feitas ao bucket do S3 - segunda execução.



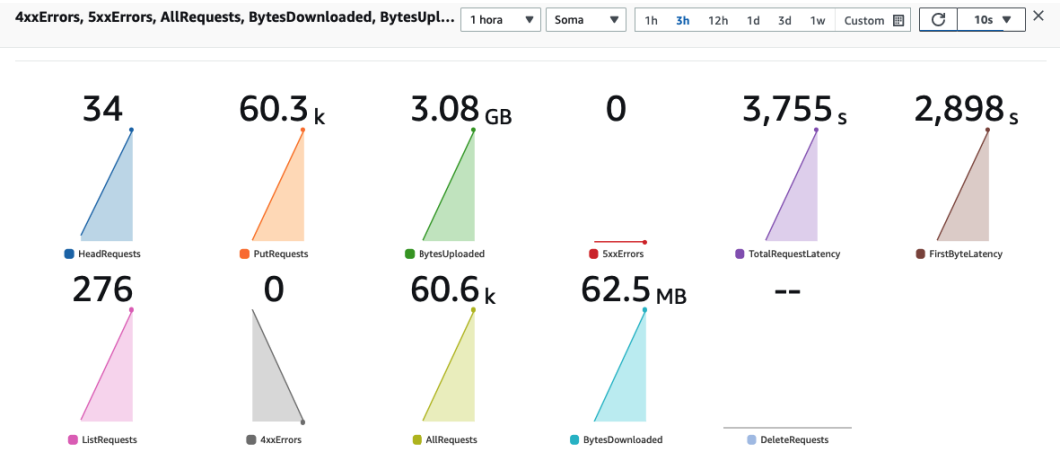
Fonte: Elaborado pelo autor.

6.3 Terceira execução

A terceira requisição buscou realizar por volta de 40.000 inserções no S3, mais precisamente 40.261, levando 6 minutos e 40 segundos de execução no Jmeter, executada em 20 *loops* de 5000 *Threads*. Dessas 40.261 inserções, foram inseridas 40.230 com sucesso no *bucket*, tendo assim a somatória das requisições anteriores com esta, conforme é exibido na Figura 16 do CloudWatch, mostrando a soma de tudo que ocorre no *bucket*.

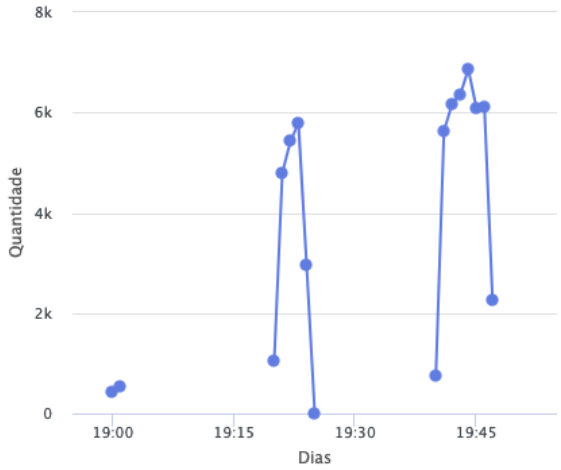
O gráfico da Figura 17 da requisição PUT exibe a continuidade das requisições, tendo o início da terceira etapa apos 15 minutos, também fazendo solicitações variadas em intervalos diferentes, de modo que a taxa de transferência foi de 6,112.704 solicitações/minuto.

Figura 16 – Gráfico de soma do CloudWatch referente ao bucket do S3 - terceira execução.



Fonte: Elaborado pelo autor.

Figura 17 – Gráfico de requisições PUT feitas ao bucket do S3 - terceira execução.



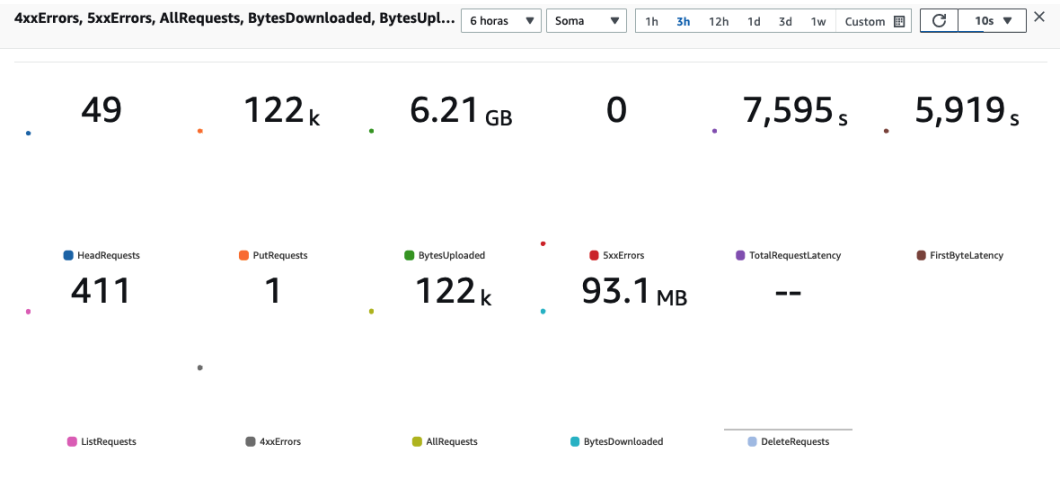
Fonte: Elaborado pelo autor.

6.4 Quarta execução

A quarta requisição buscou realizar 60.297 inserções no *bucket*, levando 9 minutos e 30 segundos de execução no Jmeter, executada em 30 *loops* de 5000 *Threads*. Dessas 60.297 requisições, foram inseridas 60.288 com sucesso no *bucket*, tendo assim a somatória das requisições anteriores com esta, conforme é exibido na Figura 18 do CloudWatch, mostrando a soma de tudo que ocorre no *bucket*.

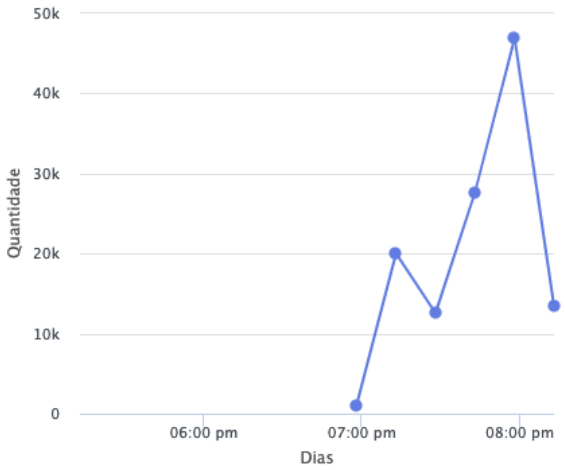
O gráfico da Figura 19 da requisição PUT exibe a continuidade das requisições, tendo o início da terceira etapa após 15 minutos, também fazendo solicitações variadas em intervalos diferentes e tendo uma taxa de transferência semelhante a anterior de 6,341.737 solicitações/minuto.

Figura 18 – Gráfico de soma do CloudWatch referente ao bucket do S3 - quarta execução.



Fonte: Elaborado pelo autor.

Figura 19 – Gráfico de requisições PUT feitas ao bucket do S3 - quarta execução.



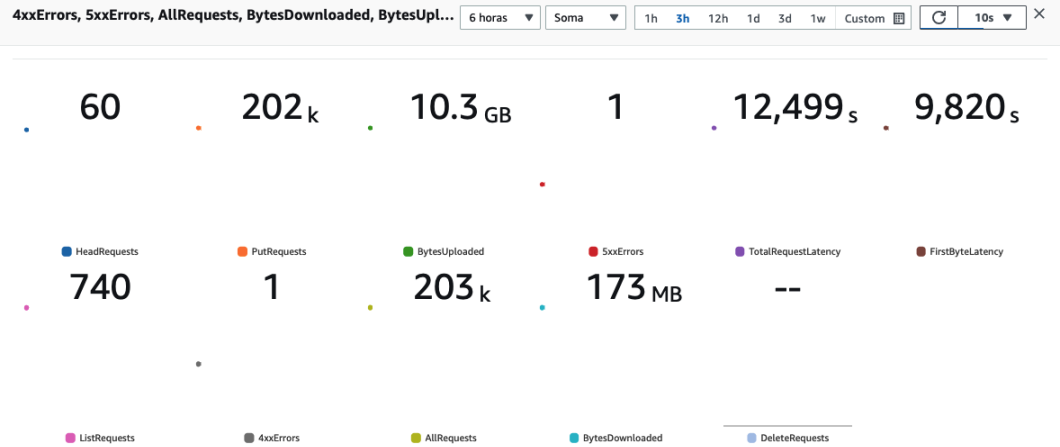
Fonte: Elaborado pelo autor.

6.5 Quinta execução

A quinta requisição tinha o objetivo de realizar 80.361 inserções no *bucket*, assim levando 13 minutos e 55 segundos de execução no Jmeter, executada em 40 *loops* de 5000 *Threads*. Dessas 80.361 requisições, foram inseridas 80.328 com sucesso no *bucket*, tendo assim a somatória das requisições anteriores, conforme é exibido na Figura 20 do CloudWatch, mostrando a soma de tudo que ocorre no *bucket*.

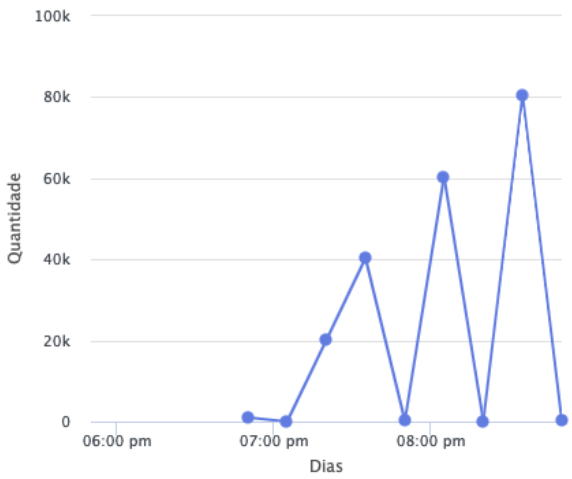
O gráfico da Figura 21 da requisição PUT exibe a continuidade das requisições para um intervalo de três horas, sendo possível notar o aumento das requisições conforme as etapas de execução, de modo que a transferência foi de 5,799.284 solicitações/minuto.

Figura 20 – Gráfico de soma do CloudWatch referente ao bucket do S3 - quinta execução.



Fonte: Elaborado pelo autor.

Figura 21 – Gráfico de requisições PUT feitas ao bucket do S3 - quinta execução.

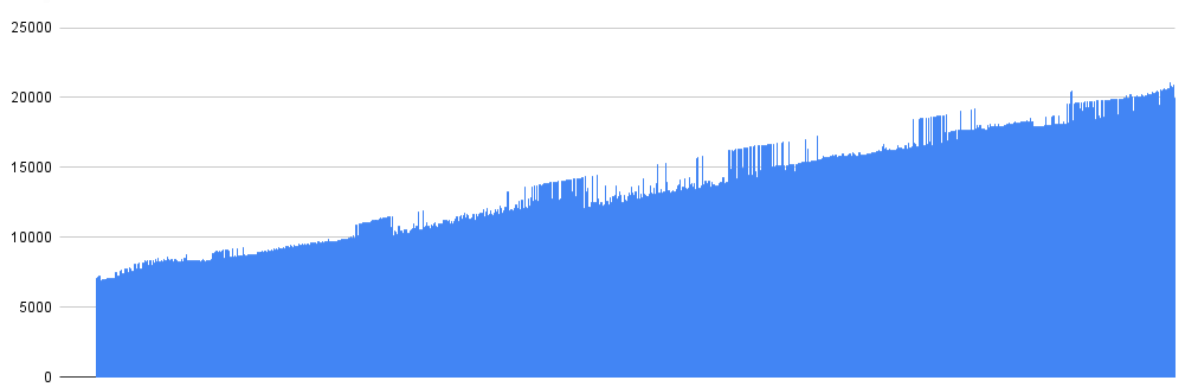


Fonte: Elaborado pelo autor.

6.6 Análises e discussões

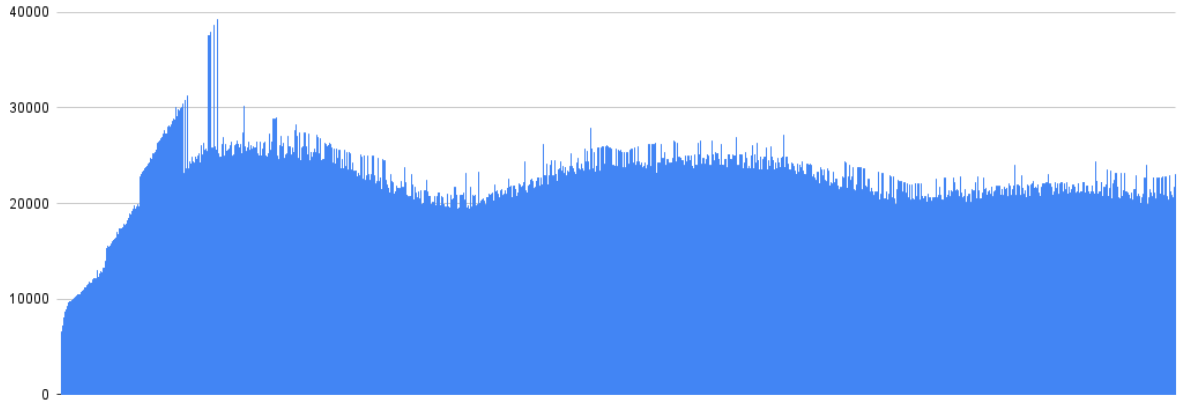
No decorrer dessas cinco etapas de execução, foram extraídos diversas informações relevantes através de um arquivo de *log* fornecido pelo Jmeter. Dentre essas informações tem-se a latência de cada requisição para todas as etapas, conforme exibido nas Figuras 22, 23, 24, 25 e 26.

Figura 22 – Gráfico de latência para as 1000 requisições.



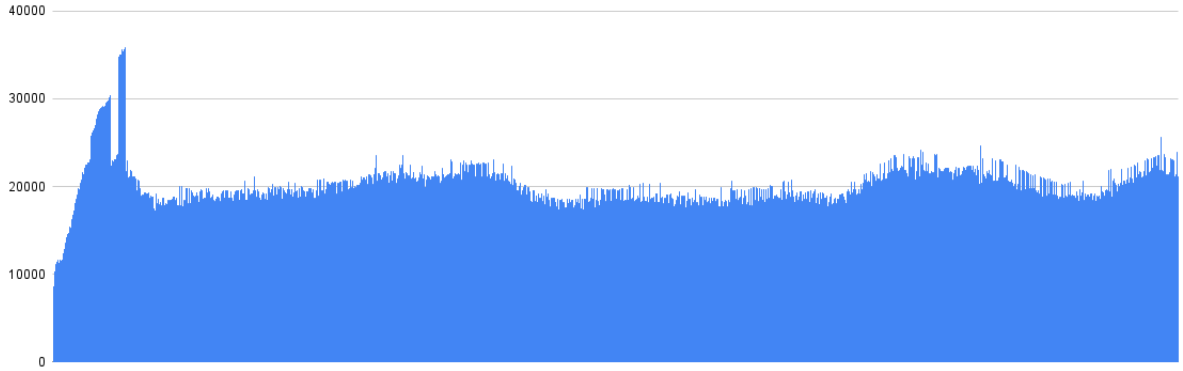
Fonte: Elaborado pelo autor.

Figura 23 – Gráfico de latência para as 20.000 requisições.



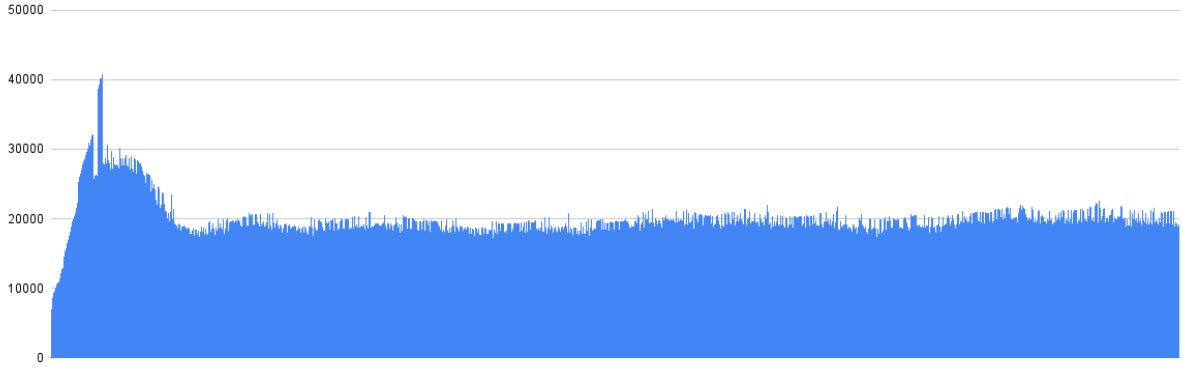
Fonte: Elaborado pelo autor.

Figura 24 – Gráfico de latência para as 40.000 requisições.



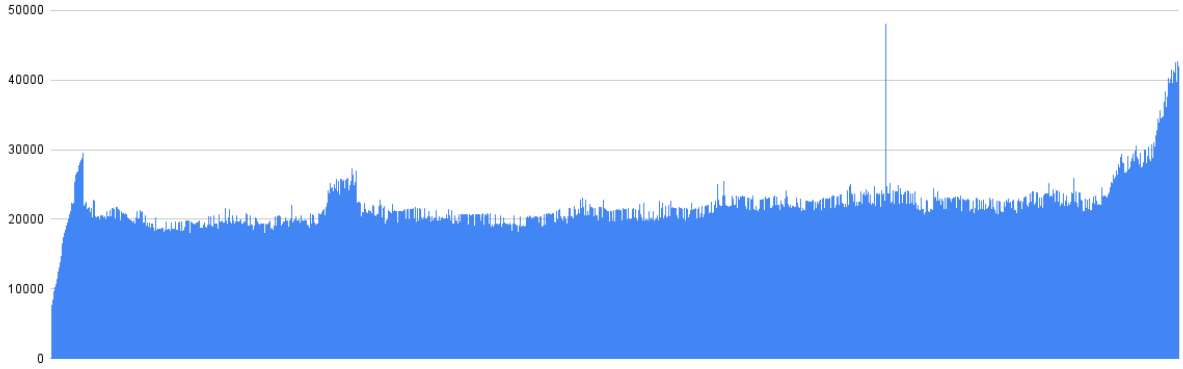
Fonte: Elaborado pelo autor.

Figura 25 – Gráfico de latência para as 60.000 requisições.



Fonte: Elaborado pelo autor.

Figura 26 – Gráfico de latência para as 80.000 requisições.



Fonte: Elaborado pelo autor.

Como se pode observar a latência para 1000 requisições é gradativa e escalável, com tendência de possivelmente continuar subindo caso ultrapassasse as 1000 solicitações. Isso se deve a diversos fatores como, por exemplo, região e horário, os quais não é possível saber com exatidão a problemática, porém é importante notar que foi a primeira pilha de execução realizada após criar o *bucket*, também a primeira vez que a solicitação viaja do remetente ao receptor, além de ser um teste simultâneo com várias *Threads*. Desta forma é possível mostrar que uma primeira execução com um grande volume de dados poderia afetar negativamente um serviço ou aplicação que use o S3 como forma de armazenamento, seja com demora na solicitação ou falha na retenção dos dados, como ocorreu nessa primeira etapa de execução.

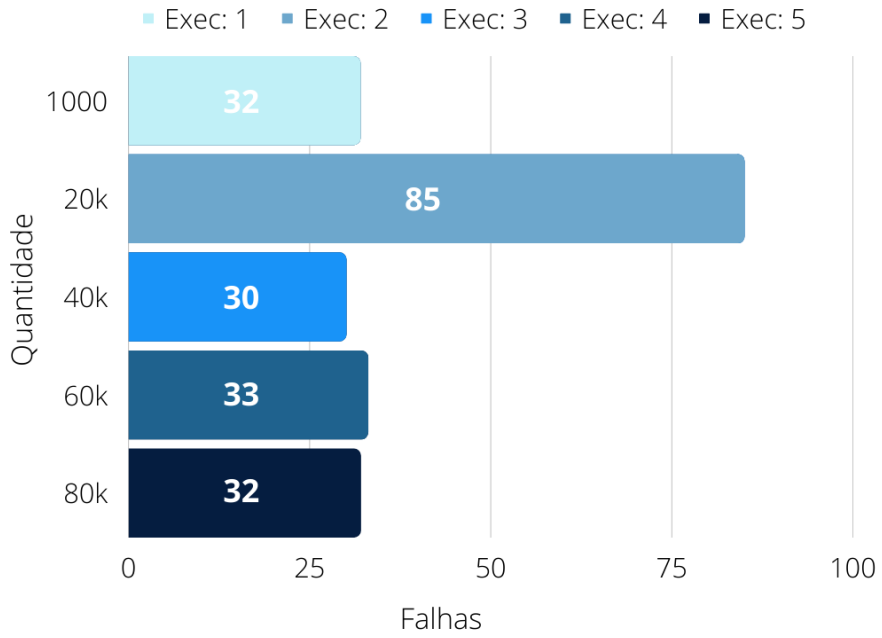
Os gráficos das figuras 23, 24, 25 e 26 para as 20.000, 40.000, 60.000 e 80.000 solicitações respectivamente mostram uma maior estabilidade nas requisições, visto que não é possível notar grandes variações ou variações crescentes como no gráfico da Figura 22 de 1000 requisições. Apesar de um aumento significativo na latência, elas acontecem de forma similar, não tendo aumento conforme o uso, mostrando que o S3 tenta se adaptar de uma forma linear mesmo tendo cada vez mais solicitações.

Por outro lado, é possível notar um maior aumento nas latências desde o início, além de estarem começando dando alguns picos que possivelmente são interferências no serviço ou na rede, visto que não ocorre novamente no decorrer do teste. Outra explicação para esse fenômeno é que o S3 pode ter um mecanismo de análise por IP, para verificar de onde e como ocorre as solicitações a ele, funcionando como um mecanismo de defesa e aplicando restrições com base no endereço de origem, como é dito em sua própria documentação. Dentre essas restrições o aumento no tempo de solicitação é totalmente viável e possível de se acontecer. Podemos notar essa restrição acontecendo principalmente na Figura 26, pois é a maior execução e maior gráfico desse *benchmark*, sendo possível notar evidentemente um grande aumento na latência no final da sua execução, dando a entender que aumentaria cada vez mais conforme continuasse sua concretização.

Apesar dessas variações serem impactantes, é importante ressaltar que nem todas requisições foram bem sucedidas, conforme já mencionado e à medida que mostra a Figura 27. É possível notar que as cinco execuções tem uma variação parecida, entre 30 a 33 falhas, apesar de a segunda execução ter tido um aumento significante, tendo uma correlação com a grande variação apresentada no gráfico da Figura 23.

Embora esses fatores apresentados possam parecer prejudiciais, é notório que o S3

Figura 27 – Gráfico da quantidade de requisições que falharam.



Fonte: Elaborado pelo autor.

se comporta de forma performática mesmo com um grande volume de dados e em uma região de armazenamento em outro país. Mesmo se analisarmos o pior cenário que é o de 20.000 requisições, pois foi oque teve mais falhas e maior instabilidade, o número de irregularidades é relativamente baixo considerando o número de solicitações. Se considerarmos o termo quantidade, podemos ver que a característica Confiabilidade da ISO/IEC 25010 manteve um comportamente adequado, tendo em consideração a baixa quantidade de falhas, porém, em um cenário de caso de uso específico em que todos os objetos precisão ser armazenados sem falhas, é explícito que o S3 pode trazer um impacto negativo a um sistema, pois são falhas que podem afetar todo um contexto de uso.

A característica de Performance envolve muitas vertentes para explorar. Porém, ao analisar dados gerais como a média de latência que possui por volta de 20.000 ms, incluindo toda execução do Jmeter e do código, é um número considerado aceitável visto que o *bucket* é criado e efetivado em outro país. Ao extrair dados do maior cenário de execução, de 80.000 requisições, é possível notar na Figura 20 do CloudWatch que a latência total das requisições para o S3 foi de 12,499 segundos, sendo um valor também aceitável considerando que foi realizado o número excessivo de 80.361 inserções diretas no *bucket*.

Por último na característica Capacidade de Manutenção que foi possível ver indícios também através dos gráficos de latência, pois mostram certa normalização e semelhança, como

o número de falhas presentes do gráfico da Figura 27, entrando muito no fato da testabilidade também, pois seria interessante executar mais algumas pilhas de testes, de modo a encontrar correlações entre as inserções, os erros, latência e taxa de transferência, desta forma seria possível ter uma maior precisão em dizer como funciona o mecanismo de verificação do S3 e como ele se comporta perante a múltiplas requisições.

Vale ressaltar que essa é uma análise inicial sobre o comportamento do S3, não considerando múltiplos e variados fatores como região, preço e tipo de conta. Para uma análise mais eficiente devem ser conduzidos inúmeros testes em diferentes cenários, conforme serão descritos no tópico de trabalhos futuros.

7 CONCLUSÃO

Este capitulo aborda o encerramento do trabalho que teve objetivo de realizar um *benchmark* no Amazon S3, como forma de analisar seu impacto e desempenho perante a múltiplas requisições, através de uma aplicação de armazenamento de imagens com testes de carga no Jmeter. Ressaltando os benefícios, dificuldades e trabalhos futuros.

7.1 Considerações Finais

Este trabalho analisa o maior serviço de armazenamento de objetos baseado na nuvem pública do mundo, realizando grandes cargas de dados e execuções, atingindo seus três pilares que são resiliência, disponibilidade e escalabilidade. É fato que os ambientes de armazenamento estão cada vez se tornando mais eficientes e performáticos, principalmente diante de situações adversas, porém se faz necessário comprovar e analisar sua eficiência, dado que relatórios e dados são gerados, para que com eles os desenvolvedores, engenheiros e usuários possam ter a garantia do seu perfeito funcionamento.

Este trabalho contribui para agregar conhecimento e dados para quem busca monitorar ambientes de nuvem, fornecendo as métricas relevantes voltadas a requisição, como a sua análise para ter uma garantia transparente do que é executado, dado que outros estudos iniciais de armazenamento em nuvem, terão que passar por estudos como este. Além disso, o trabalho demonstrou o quão operativo o Amazon S3 é diante de múltiplas interações, contando com um armazenamento de baixo custo e acessível mundialmente para qualquer público recorrente.

7.2 Benefícios e Dificuldades

Este trabalho traz como beneficio o conhecimento técnico voltado a *benchmark*, bem como sua análise diante de diversos dados, demonstrando métricas e passos essenciais para um início de estudo futuro. Além disso, contribui de forma positiva e de ampliação do desenvolvimento sobre a qualidade dos serviços em nuvem, pois esse *benchmark* fornece elementos para que usuários futuros possam estudar e compará-los diante de diversas situações presentes em outros *benchmarks*.

Contudo, surgiram dificuldades que poderiam afetar estritamente os resultados finais, como a definição de número de *Threads*, pois cada hardware tem um limitação diante do Jmeter, mais precisamente a quantidade de memória RAM. Além disso, a execução do *benchmark* em

um ambiente acadêmico, no caso o AWS Academy, não podendo configurar de forma abrangente as permissões do *bucket*, como também a sua limitação de execução por apenas três horas, restringindo algumas pretensões que seriam postas em prática.

7.3 Trabalhos Futuros

Como trabalhos futuros existem alguns seguimentos, entre eles uma expansão desta pesquisa, realizando o mesmo *benchmark* aplicado a diferentes dias e horários, cabendo também ser aplicado a diferentes regiões e objetos de variados tamanhos. Desta forma se teria uma análise mais ampla de como funciona o Amazon S3.

Outro trabalho futuro seria a comparação de um teste de carga executado no Amazon S3 a um mesmo teste de carga feito em outro serviço de armazenamento, como por exemplo, o *Google Cloud Storage*, *Microsoft Azure*, *DigitalOcean*, entre outros. Desta forma seria possível medir sua eficiência perante as outras infraestruturas.

Por último, buscar diminuir a distância entre o remetente e o receptor, ou seja, armazenar a aplicação dentro de algum serviço da própria Amazon, ou criar um teste de carga executando algum algoritmo através do AWS Lambda. Desta forma poderia se visualizar o real impacto e comportamento da nuvem perante a múltiplas execuções.

REFERÊNCIAS

ALHAMAZANI, K.; RANJAN, R.; RABHI, F.; WANG, L.; MITRA, K. Cloud monitoring for optimizing the qos of hosted applications. **4th IEEE International Conference on Cloud Computing Technology and Science Proceedings**. [S.l], p. 765–770, 2012.

AMAZON WEB SERVICES. **Serviços de nuvem - Amazon Web Services**. [S.l], 2021. Disponível em: https://aws.amazon.com/pt/. Acesso em: 23 nov. 2021.

FARSHCHI, M.; SCHNEIDER, J.; WEBER, I.; GRUNDY, J. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. **Journal of Systems and Software**. [S.l], v. 137, p. 531–549, 2018.

FELIPE, B.; LUIZ, F.; RENATO, S.; RODRIGO, C. **Computação em Nuvem**. [S.l], 2015. Disponível em: https://www.gta.ufrj.br/grad/10_1/nuvem/arquitetura.html. Acesso em: 28 nov. 2021.

GIUSEPPE, A.; ALESSIO, B.; WALTER, D.; ANTONIO, P. Cloud monitoring: A survey. **Computer Networks**. [S.l], p. 2093–2115, 2013.

GRANT, A.; ELUWOLE, O. Cloud resource management – virtual machines competing for limited resources. *In*: **Proceedings ELMAR-2013**, Croácia, p. 269–274, 2013.

HOWELL, K. **What Is Cloud Monitoring? Why Do You Need It?** [S.l], 2018. Disponível em: https://www.whatsupgold.com/blog/what-is-cloud-monitoring. Acesso em: 26 out. 2021.

INFONETICS. The cost of server, application, and network downtime. **Infonetics Research**. [S.l], 2015.

ISO/IEC 25010. ISO/IEC 25010:2011. **Systems and software engineering** — systems and software quality requirements and evaluation (square) — system and software quality models. [S.l], 2011. Disponível em: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. Acesso em: 26 nov. 2021.

KHALIL, M.; KHOMONENKO, A.; MATUSHKO, M. Measuring the effect of monitoring on a cloud computing system by estimating the delay time of requests. **Journal of King Saud University**. [S.l], 2021.

KHAN, H.; CHAN, G.; CHUA, F. An adaptive monitoring framework for ensuring accountability and quality of services in cloud computing. *In*: **International Conference on Information Networking** (ICOIN). [S.l], p. 249–253, 2016.

SURURAH, A.; LUKUMON, O.; OLUGBENGA, O.; MUHAMMAD, B.; JUAN, M.; LUKMAN, A.; ANUOLUWAPO, O.; HAKEEM, A. Cloud computing in construction industry: Use cases, benefits and challenges. **Automation in Construction**. [S.l], v. 122, 2021.

WARD, J.; BARKER, A. Observing the clouds: a survey and taxonomy of cloud monitoring. **Journal of Cloud Computing**. [S.l], v. 3, 2014.