Uma Análise Comparativa Entre Serviços SaaS (AWS-ECS) e IaaS (AWS-EC2)

Saulo Ferreira[1] orcid.org/0000-0002-3977-2361

Igor Farias[2] orcid.org/0000-0002-1373-0008

Petrônio Lopes[2] orcid.org/0000-0002-1802-8664

Gustavo Callou[1] orcid.org/0000-0002-7997-374X

[1] Departamento de Computação Universidade Federal Rural de Pernambuco, Recife, Pernambuco, Brasil,
[2] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil,

E-mail do autor principal: Saulo Ferreira saulo.gomesferreira@ufrpe.br

Resumo

Para fazer com que a computação em nuvem se torne mais performática, escalável e menos custosa financeiramente, há de se utilizar os serviços da cloud de forma eficiente e planejada, evitando sub e sob utilizar os recursos alocados. Neste contexto, os serviços baseados em infraestrutura (IaaS) e baseados em software (SaaS) surgem como alternativas no momento de implantar sistemas em cloud. Entretanto, é necessário entender quais os benefícios ou malefícios de se abstrair a camada de infraestrutura e lidar com serviços puramente baseados em contêineres. Nesse contexto, este trabalho busca comparar os dois tipos de serviços sob o ponto de vista de desempenho e escalabilidade, utilizando testes de carga concorrente para extrair métricas que avaliem o consumo eficiente dos recursos atribuídos, bem como suas capacidades de escalar seus serviços. Os resultados mostram que, ainda que serviços baseados em contêineres possuam uma latência superior em situações de pouco estresse, consegue lidar melhor com a alta carga de requisições por responder ao estresse 30% mais rápido, gerenciando a concorrência através de balanceamento de carga.

Palavras-Chave: Computação em Nuvem; Análise de Desempenho; Amazom AWS; IaaS; SaaS;

Abstract

In order to make cloud computing more performative, scalable, and financially less costly, it is necessary to use efficient and planned cloud services, avoiding under and overusing the allocated resources. In this context, the infrastructure (IaaS) and software-based (SaaS) services arise as alternatives at the moment to deploy systems in the cloud. On the other hand, it is necessary to understand the goods and bad things in abstracting the infrastructure layer and dealing with container-based services. Because of this, this work looks for comparing both kinds of services observing performance and scalability, using load testing to extract metrics that evaluate the efficient consumption of assigned resources, as well as their capability to scale their services. The results show that if container-based services get more latency in situations of lack of stress, they can deal better with high load of requisitions responding faster to stress of concurrence using load balancing aspects.

Key-words: *Cloud Computing; Performance Analysis; Amazom AWS;*

1 INTRODUÇÃO

O conceito 'computação em nuvem' vem provocando consideráveis mudanças de paradigma no planejamento e implantação das arquiteturas de software. Entre os principais pontos discutidos que se assume como vantagem ao dar preferência a um sistema baseado em nuvem é o custo inicial reduzido [1], assim como a alta escalabilidade e disponibilidade [2]. Essa abordagem de uma forma geral, pode dispensar ao usuário, a preocupação de prover, manter e custear uma infraestrutura física para disponibilizar os seus serviços, optando por uma nuvem pública. O custo sem dúvidas é um dos pontos mais abordados neste modelo de arquitetura, visto que é estimado, por conta da disponibilização de recurso sob demanda, um sistema em nuvem economize mais em virtude da elasticidade de hardware [3].

Entretanto, os pontos positivos não são incontestáveis, mas sim variáveis a uma série de fatores. Estes fatores estão geralmente relacionados à configuração e implantação da arquitetura em desacordo com as propriedades do sistema, como acessos simultâneos, variação de carga e uso de armazenamento. Desta forma, decisões equivocadas podem comprometer o desempenho e gerar custos até mais altos do que aplicações na tradicional arquitetura *On-Premises*. Por conta disso, adoção de computação em nuvem deve ser planejada considerando diversos pontos, como necessidade, custo, confiabilidade, segurança e disponibilidade [4], visando explorar com eficiência os recursos e principais vantagens do ambiente em nuvem.

A implantação de um sistema em nuvem requer um esforço de planejamento e teste, que pode envolver até reescrita de código e extração de dados [5], para não subutilizar recursos de alto custo ou comprometer o desempenho com aumento de carga acima do projetado. Além de todo processo de gerenciamento dessa infraestrutura, a computação em nuvem disponibiliza diversas abordagens para se prover um serviço, desde as tradicionais VMs (Maquina Virtuais) onde toda configuração faz parte da implantação a métodos de utilização de determinados recursos como serviço, onde nenhum esforço prévio de configuração de ambiente se faz necessário.

Com intuito de comparação de desempenho entre tipos de serviços em *Cloud Computing*, esse trabalho aborda de maneira sintetizada a execução de uma aplicação utilizando hora um serviço IaaS (Infraestrutura como um serviço) outrora SaaS (Software como um serviço) ambos na AWS (Amazon Web Services). A IaaS, é uma oferta de computação em cloud na qual um fornecedor prover recursos de computação, como armazenamento, redes e servidores. Já SaaS é uma oferta de computação em cloud que prover acesso a um software, ou seja, não há instalação dos serviços diretamente em seus dispositivos, mas apenas um acesso direto ao serviço disponibilizado, sem nenhuma preocupação com configuração de infraestrutura e/ou servidores [6].

A estratégia experimental e comparativa usada, neste trabalho, corresponde a execução de planos de testes coordenados, através de requisições simultâneas de diferentes cargas, com o intuito de extrair dados e gerar gráficos de consumo de recursos e escalabilidade. O objetivo do estudo experimental é analisar como estas diferentes abordagens para disponibilização de serviços se comportam conforme o cenário explorado, observando propriedades relevantes, como elasticidade e disponibilidade.

Além da introdução, este trabalho se divide em 6 tópicos principais. Na Seção 2, serão apresentados os trabalhos relacionados à pesquisa, explorando como, no estado-da-arte, já se explorou o tema abordado a este estudo. Na seção seguinte é apresentado a fundamentação base para discussão dos tópicos explorados na pesquisa, introduzindo conceitos como Software-as-a-Service e Infrastructure-as-a-Service. Na Seção 4 há o detalhamento de como os experimentos foram conduzidos para análise e comparação da proposta aqui descrita. A Seção 5 faz uma análise dos resultados obtidos. Por fim, conclui-se o trabalho e são apresentados os futuros direcionamentos desta pesquisa.

2 TRABALHOS RELACIONADOS

Salah et al. [7] mostraram que serviços implantados através do uso de contêineres Amazon ECS (Elastic Container Service) surpreendentemente tem um desempenho significativamente pior quando comparado com

DOI: 10.25286/repa.v8i1.2190

serviços implantados usando VMs Amazon EC2 (Elastic Compute Cloud). Foi realizado e quantificado a diferença de desempenho em termos de taxa de transferência, tempo de resposta e utilização da CPU considerando diferentes cenários de implantação.

No estudo experimental [7], foi escolhido a Amazon Web Sevices (AWS) a plataforma da nuvem, já que a nuvem da Amazon tem sido, até o presente, o mais popular provedor de nuvem de infraestrutura como serviço (IaaS). De forma mais específica, este artigo foca em avaliar o desempenho dos serviços da Web usando: (1) ECS para implantação baseada em contêiner e (2) EC2 para implantação baseada em VM.

Os autores em [8] realizaram uma pesquisa das CSPs atuais e das ferramentas de orquestração, avaliando suas arquiteturas, componentes e detalhes da orquestração. Foi realizada uma revisão do estado da arte atual em ferramentas de orquestração de contêineres, utilizando as ferramentas de orquestração como um agendamento e mecanismos de implantação. Por fim, procurou obter uma visão geral de 4 ferramentas de orquestração proeminentes, ou seja, Amazon ECS, Kubernetes, Docker Swarm e Mesos em 2 CSPs Amazon.

Shi et al. [9], estudaram a replicação de aplicativos na nuvem e problemas de implantação, considerando o tempo de resposta do aplicativo, avaliando a execução do aplicativo, a latência da rede e o custo referente ao projeto. Para resolver o problema, foi proposta uma abordagem baseada em algoritmo genético (AG) com representação de solução sob medida para o domínio, medição de aptidão e inicialização de população.

Heilig et al. [10], analisaram o problema de implantação de serviços e corretagem que é semelhante a implantação de *multicloud*. Sendo assim, a latência de rede tem impacto significativo no desempenho de serviços em nuvem, o problema de corretagem de serviços foi explorado para lidar não apenas com o custo de implantação, mas também com a latência da rede entre usuários e *datacenter* em nuvem.

Para otimizar o posicionamento de dados de mídia social e replicação em nuvem em data centers, uma abordagem baseada em AG é apresentada para minimizar o custo monetário enquanto satisfaz os requisitos de latência para todos os usuários em [11].

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Software como Serviço – SaaS

Ultimamente sempre que o tópico "computação em nuvem" é abordado, uma sopa de letrinhas entra em discussão, entre as tantas, temos as definições de como os serviços cloud são oferecidos, uma delas são os famosos SaaS, ou seja, Software as a Service. SaaS é uma forma de disponibilizar softwares e soluções de tecnologia por meio da internet, como um serviço. Com esse modelo, nenhuma interação de configuração, instalação ou até mesmo atualização de hardwares ou softwares são necessários. O acesso é focando apenas na utilização de um serviço, por exemplo, banco de dados.

Como discutido por [12] no SaaS, um serviço completo é oferecido como uma nuvem de serviço. Assumindo a necessidade de instalação de um banco de dados, se faz necessário a configuração de diversos componentes de software para obter o correto funcionamento, mas quando se utilizando através um SaaS, basta instanciar sua estrutura de SGBD (Sistema de Gerenciamento de Banco de Dados), sem nenhuma configuração adicional para estabelecer o serviço. Preocupações com sistema operacional, bibliotecas ou até mesmo versões compatíveis entre aplicações não ocorrem durante a implantação.

Na nuvem da Amazon, o Amazon Container Service (ECS) que fornece este serviço, um gerenciamento de contêineres altamente escalonável e de alto desempenho, oferecendo suporte a Docker e permitindo que os usuários executem facilmente aplicativos em um cluster gerenciado.

3.2 Infraestrutura como serviço – IaaS

IaaS fornece infraestrutura como armazenamento ilimitado e poder de computação para desenvolvedores sem a necessidade de nenhum hardware físico local. É a camada de base para a computação em nuvem que basicamente lida com máquinas virtuais, armazenamento, servidores, redes, balanceamento de carga e provedores de nuvem IaaS fornece esses recursos sob demanda.

Os requisitos mínimos para se construir a nuvem IaaS são: Hypervisor - VMM (Virtual Monitor de máquina) e topologia de rede – pública

ou privada. IaaS atenua a necessidade de um datacenter e mantém o hardware no nível local. Como exemplo temos: Amazon Web Service (AWS); Rackspace; Windows Azure etc. A AWS é uma coleção de serviços de computação remota que compõem uma plataforma de computação em nuvem [13].

A nuvem se tornou uma plataforma popular devido às suas características de serviços por demanda, infinito recursos, disponibilidade onipresente e modelo de negócios pago sob demanda [14]. IaaS é um modelo de serviço que permite que os clientes economizem dinheiro removendo a necessidade de cada cliente configurar seus próprios computadores ou cluster de computação e permite que os dados utilizem de forma eficiente seus recursos de computação.

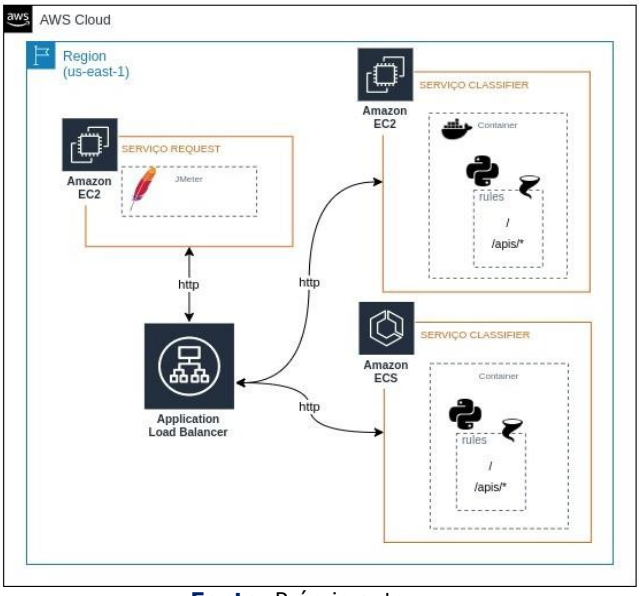
Na AWS, este serviço é oferecido pelo Elastic Compute Cloud (EC2), que fornece um ambiente de computação virtual por meio do qual o usuário pode inicializar Amazon Machine Image (AMI) que a Amazon denomina como "instância".

4 ARQUITETURA EXPERIMENTAL

No decorrer desta seção é apresentada em detalhes a arquitetura implementada no ambiente de execução dos experimentos. Para avaliação dos serviços, foram propostos testes de carga e performance com extração de métricas dos recursos utilizados. Para a execução foi utilizada a plataforma da Amazon Web Services (AWS), implantando todos em uma mesma região (us-east-1), com um serviço responsável por gerar a carga de requisições e em outra extremidade um serviço de classificação utilizando uma imagem Docker. A execução é realizada em um primeiro momento direto no EC2 (IaaS), outrora executada via ECS (SaaS). O diagrama apresentado na Figura 1 ilustra a arquitetura conforme foi executada.

Por lidar diretamente com o hardware, sem o gerenciamento e orquestração de contêineres da AWS, espera-se que o serviço de EC2 apresente melhores resultados em desempenho quando os testes são realizados sobre uma mesma máquina virtual com uma única instância de serviço executando. Entretanto, em um cenário de múltiplos contêineres simultâneos sendo iniciados pelo ECS, é possível que o balanceador de carga seja o diferencial para uma melhora de desempenho em relação ao EC2.

Figura 1 - Diagrama arquitetural de infraestrutura dos experimentos



Fonte: Próprio autor.

4.1 Serviço de Carga

O serviço utilizado para realizar o teste de carga foi o Amazon EC2, uma aplicação baseada em IaaS que provê máquinas virtuais com personalização abrangente de recursos – como memória, processador e disco – e sistemas operacionais (SOs). Neste serviço, foi utilizada instância tipo t3a.medium, esse perfil de máquina possui 2 vCPUs e 4GB de memória RAM. Toda a carga envolvida nos experimentos é baseada em requisições HTTP Rest para um endpoint disponibilizado pelos serviços analisados, enviando os dados em formato JSON através do body da requisição contendo uma imagem convertida em base64. A imagem utilizada é apresentada na Figura 2, essa imagem foi selecionada aleatoriamente assumindo apenas a necessidade de haver um veículo qualquer em cena, uma vez que o serviço de classificação usado é responsável por classificar objetos, entre eles, veículos.

Para a execução dos planos de teste, utilizou-se o Apache JMeter [15], uma ferramenta de automação de testes capaz de simular cargas com diferentes parâmetros, como número de usuários

DOI: 10.25286/repa.v8i1.2190

simultâneos, intervalo entre requisições e delimitação de tempo de stress. O software, além de executar o experimento, permite o monitoramento e exportação de diversas métricas extraídas das execuções, como tempo mínimo, médio e máximo de resposta, bem como taxa de erro e taxa de transferência envolvida na requisição.

4.2 Serviço de Classificação

Para comparação dos dois modelos de execução (Iaas e SaaS) foi utilizado um classificador implementado com a linguagem Python com utilização do OpenCV para processamento digital de imagem. Esse serviço é responsável por realizar a análise estática de um frame e identificar a presença de pessoas e/ou veículos na cena. Todo o processamento é realizado através de uma rede neural que tem como saída a porcentagem e a indicação na cena dos elementos citados.

Figura 2 – Imagem utilizada para request ao serviço de Classificação.

Fonte: https://www.abs.gov.au/statistics/industry.

Redes neurais são sistemas de computação com nós interconectados que funcionam simulando os neurônios do cérebro humano. Usando algoritmos, elas podem reconhecer padrões e correlacionados em contexto de forma classificatória [16].

4.2.1 Execução IaaS

Para o modelo com IaaS foi utilizado um Amazon EC2, instância tipo g4dn.xlarge, esse perfil de máquina possui 4 vCPUs, 16GB de memória RAM e uma GPU NVIDIA T4 de 16GB. A EC2 foi configurada com o sistema operacional Ubuntu na versão 20.04. Neste cenário, a aplicação foi implantada utilizando Docker para a conteinerização da aplicação, com a configuração dos contêineres variando conforme o experimento a ser executado. O sistema expõe endpoints acessíveis com APIs HTTP Rest pelo serviço. carga.

4.2.2 Execução SaaS

O serviço SaaS foi implementado utilizando Amazon ECS, uma aplicação modelo de arquitetura SaaS que disponibiliza, gerencia e monitora contêineres. Em tal cenário, a disponibilização do serviço abstrai a camada de recurso de hardware e gerenciamento de sistemas operacionais, utilizando aplicações baseadas contêineres para lidar diretamente com o software em si. Essa abordagem possibilita até, em determinados cenários, a disponibilização sob demanda, sem alocar recursos em momentos de ociosidade. Para os experimentos realizados, foi utilizado o tipo de inicialização com EC2, com o mesmo perfil g4dn.xlarge alocado para arquitetura com IaaS, mas de modo que a gerência e orquestração é feita através de ECS. Nesse contexto, o ECS é único responsável por alocar, monitorar e escalar o serviço, dada uma determinada máquina virtual. A utilização deste modelo se dá pela limitação da inicialização com *Fargate* – modelo que utiliza *serverless*, um mecanismo sem servidor para disponibilizar recurso sob demanda e independente de alocação direta de máquinas virtuais – no uso de GPUs, recurso necessário na aplicação utilizada para os testes [17].

5 EXPERIMENTOS

Esta seção descreve os parâmetros e configurações consideradas nos experimentos, bem como os cenários de execução realizados.

5.1 Parâmetros

Durante a execução dos testes, variaram-se os grupos de usuários envolvidos no experimento (quantidade simultânea de acesso ao serviço). Desta forma, foi possível analisar o comportamento do sistema de acordo com o número de usuários simultâneos realizando a mesma requisição no sistema durante um tempo determinado. Foram utilizados 4 grupos de requisições, contendo 10, 20, 30 e 50 usuários realizando requisições paralelas e sequenciais durante 30 minutos para cada grupo. Desta forma, iniciaram-se os experimentos com o grupo de 10 usuários fazendo requisições durante 30 minutos,

ao final da sequência foi realizado um intervalo de 5 minutos para em seguida o grupo de 20 usuários iniciar sua execução e assim sucessivamente até o grupo de 50 usuários simultâneos. Além de toda variação de carga citada, foi realizado para cada cenário a variação na configuração dos recursos computacionais utilizados, conforme apresentados na Tabela 1. O serviço utilizado para realizar o teste de carga.

Além da variação dos grupos de usuários executando nos dois tipos serviços a serem avaliados, alguns dados relevantes foram extraídos para análise, conforme descrito em 5.2, onde são apresentados os dados monitorados durante a execução. A partir das requisições, é possível extrair, através da ferramenta de automação utilizada, o tempo médio e a taxa de erro das requisições. Essas informações são bastante relevantes para a avaliação do quão performático são os serviços nos contextos avaliados pelo experimento, podendo-se chegar a conclusão até qual carga é suportada, através da taxa de erro, que pode crescer com aumento do número de usuários simultâneos.

Tabela 1 Parâmetros considerados nos experimentos

PARÂMETROS		
Tipo de Execução	Quantidade Usuários	Recurso Computacional
ECS EC2	10 20 30 50	Mem. RAM CPU GPU

Fonte: Próprio autor.

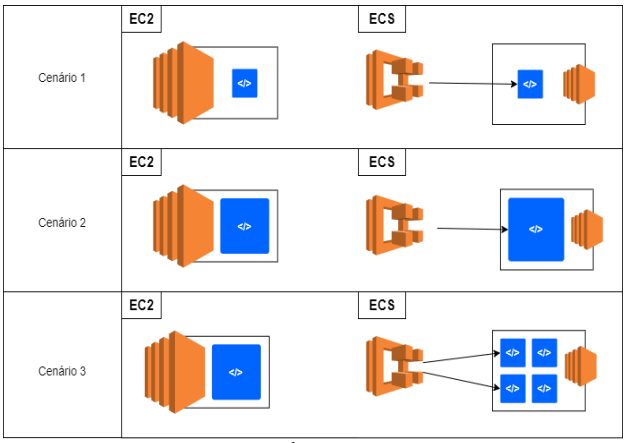
Adicionalmente, os recursos computacionais utilizados pelo serviço são monitorados durante os experimentos, de forma continua com saltos de 30 segundos entre cada leitura (limitação do serviço de monitoramento da AWS – *CloudWatch*). O intuito é avaliar como as duas aplicações gerenciam as requisições conforme a carga que será aplicada.

5.2 Cenários Analisados

Esse estudo busca estabelecer cenários para uma comparação mais abrangente da performance dos modos de operacionalizar o serviço. Esses cenários definem diferentes delimitações de recursos e de gerenciamento de escalabilidade para as aplicações, aplicadas tanto em contêiner gerenciado diretamente com Docker no EC2, quanto ao gerenciado pelo serviço ECS. Assim, define-se três cenários distintos para a execução dos experimentos, como mostrado na Figura 3 e descritos em seguida.

• **Cenário I**: Neste cenário, os dois contêineres possuem uma limitação equivalente em seus determinados ambientes de execução, definidos tanto pelo EC2 quanto para o ECS. Em tal caso, as máquinas foram configuradas para limitar os serviços ao uso de 1 vCPU, 3.5 GB de memória RAM e 30% da GPU disponível no modelo da instância, sem gerenciamento de escalabilidade pelo ECS ou EC2. Ou seja, não há resposta de quaisquer serviços à sobrecarga de uso, se mantendo indisponível nesta ocorrência.

Figura 3 – Cenários abordados nos experimentos



Fonte: Próprio autor.

• **Cenário II**: No segundo cenário estabelecido, aloca-se uma configuração mais robusta para os serviços, sendo em questão 4 vCPUs, 12 GB de memória RAM e 100% da GPU do modelo da instância. No entanto, apesar alta disponibilidade de recursos, não há nenhuma variação relacionada a escalar os serviços em caso de alta carga de execução. Os dois serviços ficam com recursos equivalentes, entretanto, um sendo gerenciando diretamente pelo Docker, sem nenhuma intervenção externa e o outro, pelo ECS. Por estar utilizando todo o recurso alocado, o ECS não consegue multiplicar os contêineres para atender a alta demanda. O EC2, por sua vez, para manter ambos os serviços equiparados, fica sem a configuração do AWS Auto Scaling que

DOI: 10.25286/repa.v8i1.2190

possibilitaria a inicialização de outras instâncias do serviço em resposta à sobrecarga.

- **Cenário III**: No último cenário, aloca-se o recurso com alto poder computacional, configuração equivalente ao cenário II para o serviço Docker no EC2. Enquanto isso, no ECS, o serviço terá recursos reduzidos igualmente ao cenário 1. Entretanto, podendo escalar até 03 contêineres em caso de alta carga aplicada. No cenário de maior carga, na prática, os dois utilizam recursos computacionais equivalentes, entretanto, o ECS fazendo múltiplos serviços com recursos reduzidos e o EC2 com puramente o Docker, executando um único serviço com toda o poder computacional da instância.

6 RESULTADOS

Esta seção objetiva explicitar os resultados obtidos através dos experimentos executados nos três cenários descritos anteriormente.

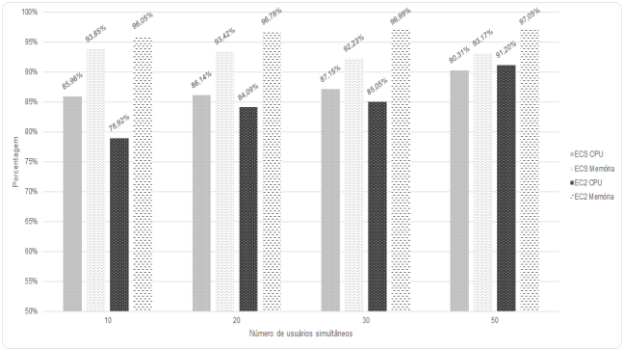
6.1 Cenário I

No primeiro cenário, onde os contêineres possuem recursos limitados e não conseguem escalar, observou-se que os recursos alocados são utilizados massivamente pelo serviço que mantém pouca variação ao longo dos grupos de usuários. A memória RAM e CPU são utilizados quase que por completo durante às quatro diferentes cargas aplicadas, não apresentando uma variação significativa que comprove algum conceito em específico.

Entretanto, as variações observadas no desempenho das requisições se mostram mais significativas para o teste aqui realizado. No gráfico da Figura 5, que apresenta a variação no tempo de resposta em decorrência dos diferentes grupos de usuários. Em ambos os serviços, o tempo médio de requisição aumenta de acordo com o aumento da carga aplicada, como esperado, implicando em um crescimento de cerca de 400% do tempo médio com a carga de 10 usuários para o tempo com a carga de 50 usuários concorrentes.

Porém, o EC2 demonstra melhor desempenho neste cenário, chegando a apresentar um tempo de resposta médio superior em aproximadamente 15% para a carga de 20 usuários simultâneos. Este comportamento pode ser explicado pelas camadas de infraestrutura a mais que o SaaS apresenta para fazer o gerenciamento dos contêineres, bem como a interface de rede aplicada e aplicações de *load balacing* que

Figura 4 – Cenário I: Utilização de recursos pelos serviços

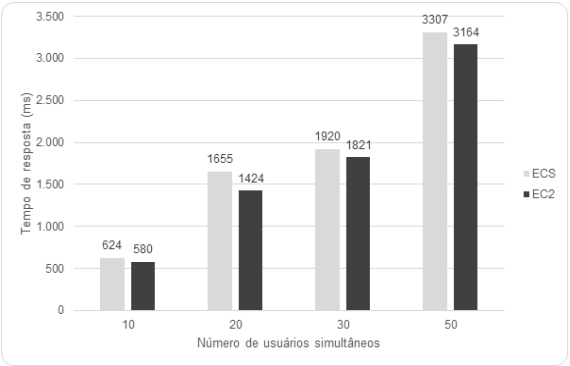


Fonte: Próprio autor.

compõem a estrutura ECS. Por isso, mesmo que com recursos equivalentes, o EC2 consegue performar melhor por conta da falta da aplicação intermediária provida pela AWS quando utilizando recursos limitados equivalentes e sem nenhuma escalabilidade.

Fonte: Próprio autor.

Figura 5 - Cenário I: Tempo médio de resposta das requisições



6.2 Cenário II

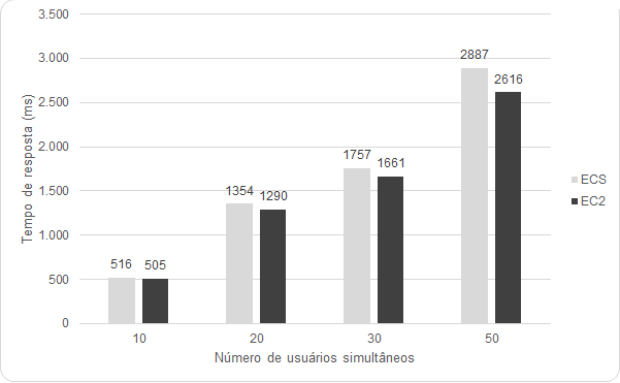
No segundo cenário, quando os contêineres estão configurados tendo acesso a quase totalidade dos recursos disponíveis no modelo da instância, é possível observar que o tempo médio de requisições reduz consideravelmente para ambos os serviços avaliados, por conta da maior disponibilidade de recursos para lidar com as

operações, como se observa na Figura 6. Neste caso, tem-se um desempenho melhor em cerca de 20% em relação ao tempo de resposta para carga de 50 usuários simultâneos aplicada ao serviço.

Tal fator impacta diretamente no consumo dos recursos providos pelo sistema aos contêineres, onde a utilização de CPU alcança os 90% e se mantém nesse nível durante toda a execução. No entanto, vemos uma utilização baixa da memória na Figura 7, isso se explica, pois o serviço utilizado para testes consome mais memória de vídeo (GPU), mantendo-se o nível de utilização de RAM atenuado no cenário estudado.

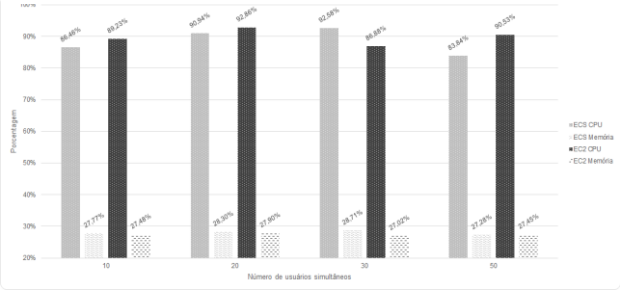
Neste cenário, ainda é possível identificar um desempenho superior do IaaS, podendo-se ainda se basear na mesma explicação do cenário anterior, onde, mesmo que existindo recurso computacional igual a ambos os casos, a camada de aplicação para prover o SaaS faz o serviço ter um desempenho inferior.

Figura 6 - Cenário II: Tempo médio de resposta das requisições



Fonte: Próprio autor.

Figura 7 - Cenário II: Utilização de recursos pelos serviços



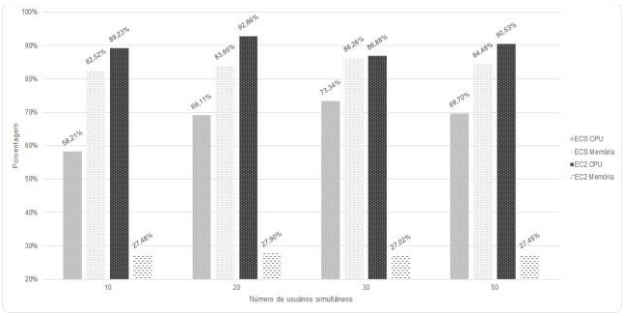
Fonte: Próprio autor.

6.3 Cenário III

O terceiro cenário, que apresenta configurações diferentes para os serviços, avalia um ECS com recursos limitados, mas com escalabilidade do serviço dentro da mesma instância e o EC2 com os recursos totais fornecidos pela infraestrutura selecionada, mas sem escalabilidade do serviço. Vale ressaltar que mesmo o EC2 tenha todos os recursos disponíveis para processamento, o ECS consegue escalar horizontalmente o serviço inicial limitando-se aos recursos totais equivalentes aos do EC2. Para isso, o ECS gerencia o escalonamento, iniciando outros contêineres na medida que for necessário para a carga que vai lidar.

Ao observar a utilização dos recursos do contêiner apresentada na Figura 8, vê-se o mesmo comportamento do cenário anterior para o caso do EC2 que, na prática, repete as configurações e conseguinte, os resultados apresentados na

Figura 8 - Cenário III: Utilização de recursos pelos serviços



Fonte: Próprio autor.

subseção anterior. Portanto, a partir da análise dos recursos, é possível ver que o ECS consegue fazer melhor a gestão dos recursos do SaaS, ao se observar que a utilização se mantém constante até mesmo com o escalonamento e inicialização de novos contêineres, mantendo o recurso em cerca de 65% de uso durante os experimentos.

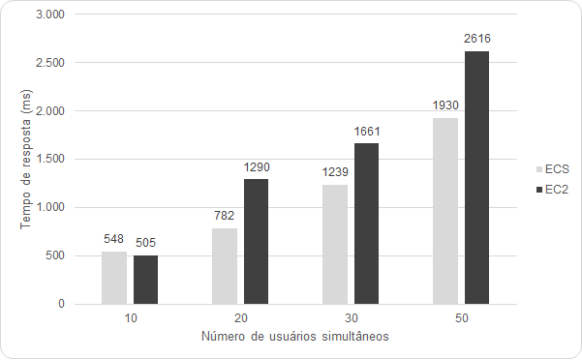
É importante descrever como o escalonamento dos recursos se comportou para o ECS neste terceiro cenário, onde, a partir da carga de 20 usuários, o ECS já observou sobreutilização dos recursos e inicializou outro contêiner. Assumindo 30 e 50 usuários foram inicializados os três possíveis contêineres. Desta forma, com até 3 serviços executando em paralelo e processando

DOI: 10.25286/repa.v8i1.2190

requisições, o ECS conseguiu melhor no desempenho que o EC2, mesmo que este, possuísse recursos superiores disponíveis durante todo o experimento, porém, com apenas um serviço para lidar com as requisições simultâneas.

Como pode-se ver na Figura 9, ainda que o EC2 tenha começado com desempenho superior, o ECS consegue lidar melhor quando começa a escalonar e iniciar novos contêineres. Na maior carga, que por sinal é a que apresenta a diferença mais expressiva em termos de tempo médio de resposta, o ECS consegue ser 30% mais rápido que o EC2 (2616 ms para EC2 vs 1930 MS para o ECS).

Figura 9 - Cenário III: Tempo média de resposta das requisições



Fonte: Próprio autor.

6.4 Limitações do Trabalho

O experimento, entretanto, possui alguns pontos relevantes para se considerar, apresentando limitações em seu objetivo e em sua execução. Sendo assim, é importante salientar que o framework utilizado para implementação de APIs HTTP REST utilizadas no experimento pode ter um impacto no resultado, visto que pode lidar com assincronia e concorrência de uma forma diferente de outras ferramentas com este fim. O serviço de carga, por sua vez, também pode ter o seu impacto no resultado, restringindo ocasionalmente a execução threads concorrentes que o JMeter utiliza, por conta de escassez eventual de recursos ao longo dos testes. Apesar da configuração de um ambiente igualitário para todos os cenários, até mesmo os meios de execução dos serviços podem utilizar componentes não controláveis, como o serviço de gerenciamento de contêineres do ECS ou o balanceador de carga da AWS. Por isso, é importante explicitar estes pontos.

7 RESULTADOS

Existem diferentes vantagens na utilização de cada serviço, uma vez que, em diferentes cenários, eles apresentam comportamentos distintos. Cabe ao engenheiro de Cloud observar o comportamento esperado de sua aplicação e o uso devido que se deseja fazer da estrutura ambientada na nuvem. Isso é relevante em diferentes contextos, pois, no caso do ECS, é possível utilizar uma mesma infraestrutura para inicialização e gerenciamento de diferentes serviços, sem manter subutilizado algum recurso por um período onde não ocorre tantas requisições. Desta forma, é possível manter, em um mesmo host, diferentes serviços em contêineres gerenciados pelo ECS, que podem ser escalados a qualquer momento pela plataforma evitando gasto financeiro desnecessário.

Porém, abdicar da responsabilidade de gerenciar os contêineres tem um custo, pois, como foi observado, a camada de aplicação responsável pelo monitoramento e gerenciamento automatizado dos serviços tem um impacto no desempenho do sistema em alguns cenários, implicando em uma maior espera nas mesmas requisições quando comparado a um serviço IaaS utilizando o EC2, por exemplo. Assim, o estudo de caso é necessário para o uso eficiente da Cloud, por que mesmo com a perca diminuta de desempenho, a otimização do uso de recursos pode ser um ponto chave em questão de custos financeiros e gerenciais da nuvem.

Este trabalho apresenta algumas destas respostas, ainda que com algumas limitações usuais em seu con- texto. Entre as limitações do trabalho, cabe observar a falta de análise de custo real da utilização dos serviços da plataforma em diferentes contextos. Também é importante pontuar a utilização da aplicação nos testes, um sistema de classificação de imagens que utiliza GPU e pode possuir suas particularidades em gerenciamento de processos concorrentes, podendo impactar diretamente nos resultados e impossibilitando a generalização necessariamente proporcional para outros casos de uso com outros serviços distintos.

AGRADECIMENTOS

Os autores gostariam de agradecer à empresa Avantia Tecnologia e Segurança SA[1], que financiou a infraestrutura necessária para a realização dos experimentos propostos em um caso de uso real da companhia, a qual trabalha com análise e detecção inteligente de ações em imagens de câmeras ao vivo para fins de segurança.

REFERÊNCIAS

[1] WEST, Darrell M. **Saving money through cloud computing**. Governance Studies at Brookings, 2010.

[2] GAJBHIYE, Amit; SHRIVASTVA, Krishna Mohan PD. Cloud computing: Need, enabling technology, architecture, advantages and challenges. In: **2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence)**. IEEE, 2014. p. 1-7.

[3] NETO, Paulo. Demystifying cloud computing. In: **Proceeding of doctoral symposium on informatics engineering**. 2011. p. 16-21.

[4] BRUMEC, Slaven; VRČEK, Neven. Cost effectiveness of commercial computing clouds. **Information Systems**, v. 38, n. 4, p. 495-508, 2013.

[5] SUN, Kewei; LI, Ying. Effort estimation in cloud migration process. In: **2013 IEEE seventh international symposium on service-oriented system engineering**. IEEE, 2013. p. 84-91.

[6] MOHAMMED, Chnar Mustafa; ZEEBARE, Subhi RM. Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review. **International Journal of Science and Business**, v. 5, n. 2, p. 17-30, 2021.

[7] SALAH, Tasneem; ZEMERLY, M. Jamal; YEUN, Chan Yeob; AL-QUTAYRI, Mahmoud; AL-HAMMADI, Yousof. Performance comparison between container-based and VM-based services. In: **2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)**. IEEE, 2017. p. 185-190.

[8] AWADA, Uchechukwu. Application-Container Orchestration Tools and Platform-as-a-Service Clouds: A Survey. **International Journal of Advanced Computer Science and Applications**, 2018.

[9] SHI, Tao; MA, Hui; CHEN, Gang; HARTMANN, Sven. Location-Aware and Budget-Constrained Application Replication and Deployment in Multi-Cloud Environment. In: **2020 IEEE International Conference on Web Services (ICWS)**. IEEE, 2020. p. 110-117.

[10] HEILIG, Leonard; LALLA-RUIZ, Eduardo; VOß, Stefan. A cloud brokerage approach for solving the resource management problem in multi-cloud environments. **Computers & Industrial Engineering**, v. 95, p. 16-26, 2016.

[11] KHALAJZADEH, Hourieh; YUAN, Dong; GRUNDY, John; YANG, Yun. Improving cloud-based online social network data placement and replication. In: **2016 IEEE 9th international conference on cloud computing (CLOUD)**. IEEE, 2016. p. 678-685.

[12] PAWAR., K. K. Cloud computing: For a better tomorrow. In: **Information Technology Department, St. Fran- cis Institute of Technology** (SFIT) College of Engineering, Borivli (W), 2011.

[13] VARIA, Jinesh. Amazon Web Services-Architecting for the cloud: best practices. **Amazon Web Services, January**, 2011.

[14] MELL, Peter; GRANCE, Tim. The NIST definition of cloud computing. 2011.

[15] HALILI, Emily H. **Apache JMeter**. Birmingham: Packt Publishing, 2008.

[16] VALENÇA, Mêuser. Fundamentos das redes neurais: exemplos em Java. **Livro Rápido**, 2010.

[17] AMAZOM AWS. Amazon Elastic Container Service - Guia do Desenvolvedor. Disponível em:https://docs.aws.amazon.com/pt_br/AmazonECS/latest/developerguide/ecs-gpu.html. Acesso em: 20 de novembro. 2021

1 www.avantia.com.br

DOI: 10.25286/repa.v8i1.2190