UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

HIGOR DUARTE DE OLIVEIRA
Orientador: Prof. Dr. Tiago Garcia de Senna Carneiro

ANÁLISE DE DESEMPENHO E PLANEJAMENTO DE CAPACIDADE DE MICROSSERVIÇOS ESPACIAIS

Ouro Preto, MG
2022

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

HIGOR DUARTE DE OLIVEIRA

ANÁLISE DE DESEMPENHO E PLANEJAMENTO DE CAPACIDADE DE MICROSSERVIÇOS ESPACIAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Tiago Garcia de Senna Carneiro

Ouro Preto, MG
2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

O48a Oliveira, Higor Duarte de.
Análise de desempenho e planejamento de capacidade de microsserviços espaciais. [manuscrito] / Higor Duarte de Oliveira. - 2022.
25 f.

Orientador: Prof. Dr. Tiago Garcia de Senna Carneiro.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Análise de desempenho. 2. Apache JMeter. 3. AWS. I. Carneiro, Tiago Garcia de Senna. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Higor Duarte de Oliveira

ANÁLISE DE DESEMPENHO E PLANEJAMENTO DE CAPACIDADE DE MICROSSERVIÇOS ESPACIAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 03 de Novembro de 2022.

Membros da banca

Tiago Garcia de Senna Carneiro (Orientador) - Doutor - Universidade Federal de Ouro Preto
Joubert de Castro Lima (Examinador) - Doutor - Universidade Federal de Ouro Preto
Saul Emanuel Delabrida Silva (Examinador) - Doutor - Universidade Federal de Ouro Preto

Tiago Garcia de Senna Carneiro, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 03/11/2022.

Documento assinado eletronicamente por **Tiago Garcia de Senna Carneiro**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/11/2022, às 14:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0416030** e o código CRC **93CA1451**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Agradecimentos

Agradeço a meus pais Eduardo e Elisangela por todo apoio. Agradeço aos professores pelos ensinamentos. Em especial, agradeço ao meu professor orientador Tiago, pela oportunidade de participar do laboratório Terralab, o que permitiu me desenvolver pessoalmente e profissionalmente.

Resumo

No presente trabalho, são realizados experimentos para a análise de desempenho de uma arquitetura de microsserviços espaciais implantada em instâncias EC2 na Amazon Web Services (AWS) contendo os projetos livres Pelias para o serviço de geocodificação e Open Route Service (ORS) para o serviço de cálculo de isócronas de tempo e distância. Os experimentos foram realizados considerando uma taxa constante de requisições por minuto e a ferramenta Apache JMeter foi utilizada para simular uma carga de trabalho. Após a execução dos experimentos concluiu-se que o serviço de geocodificação executado em uma instância EC2 t2.2xlarge na AWS suporta uma taxa de 10.500 requisições por minuto com o componente Interpolation ativado e uma taxa de 13.000 requisições por minuto com o componente Interpolation desativado. Os experimentos também permitiram estimar valores ideais para os parâmetros de configuração desse serviço. Para o serviço de cálculo de isócronas, concluiu-se que, considerando a execução em uma instância EC2 t2.2xlarge na AWS e isócronas de distância com intervalo único, 20 km é o alcance máximo suportado e para isócronas de distância com intervalos de 1 km e para uma instância EC2 t2.2xlarge, conclui-se que 11 km é o alcance máximo suportado.

Palavras-chave: Análise de desempenho. Apache JMeter. AWS.

Abstract

In the present work, experiments were carried out to analyze the performance of a spatial microservices architecture deployed in EC2 instances on Amazon Web Services (AWS) containing the free projects, Pelias for the geocoding service and Open Route Service (ORS) for the time and distance isochrones calculation service. The experiments were performed considering a constant rate of requests per minute and the Apache JMeter tool was used to simulate a workload. After running the experiments, it was concluded that the geocoding service running on an EC2 t2.2xlarge instance on AWS supports a rate of 10,500 requests per minute with the Interpolation component enabled and a rate of 13,000 requests per minute with the Interpolation component disabled. The experiments also allowed estimating ideal values for the configuration parameters of this service. For the isochrones calculation service, it was concluded that, considering running on an EC2 t2.2xlarge instance on AWS and distance isochrones with single interval, 20 km is the maximum supported range and for distance isochrones with intervals of 1 km and for an EC2 t2.2xlarge instance, it is concluded that 11 km is the maximum supported range.

Keywords: Performance Analysis, Apache JMeter, AWS.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

SaaS	Software as a Service
SaaP	Software as a Product
IaaS	Infrastructure as a Service
ORS	Open Route Service
AWS	Amazon Web Services
GCP	Google Cloud Platform
ms	milissegundos
mb	megabytes
gb	gigabytes
s	segundos
ES	Elasticsearch

Sumário

1 Introdução

Com o aumento da acessibilidade da internet em larga escala e o surgimento de empresas fornecedoras de *Infrastructure as a Service* (IaaS) [1] como Amazon Web Services (AWS) e Google Cloud Platform (GCP), muitos softwares estão migrando seu modelo de *Software as a Product* (SaaP) para *Software as a Service* (SaaS) [2]. Como exemplo, o Microsoft Office que antes era disponibilizado como um produto por meio de uma mídia física, agora é disponibilizado como um serviço através de uma página web.

No modelo SaaS, existem muitas arquiteturas de software possíveis de se utilizar. Para este trabalho vamos nos ater a *arquitetura de microsserviços*. Segundo (FOWLER, 2019), uma arquitetura de microsserviços é uma abordagem para desenvolver um único aplicativo como um conjunto de pequenos serviços, cada um executando seu próprio processo e se comunicando com *lightweight mechanisms*, geralmente uma API de recursos HTTP.

Neste texto, entendemos por *microsserviços espaciais* aqueles que oferecem a seus usuários informações extraídas a partir de computações realizadas de dados geográficos, como mapas digitais e imagens de sensores remotos. Exemplos destas computações são cálculos de rotas entre dois locais no espaço, determinação da localização de um determinado endereço textual ou a extração de feições a partir de imagens obtidas de satélites.

Antes de disponibilizar um software através do modelo SaaS, é necessário analisar a viabilidade econômica deste software e planejar sua capacidade em atender um determinado volume de trabalho, isto é, número de usuários e número de requisições por usuário. Para tanto, é preciso utilizar técnicas para análise de desempenho e planejamento de capacidade de sistemas de computação (JAIN, 1991). Essas técnicas serão explicadas em detalhes no capítulo 2.

Este trabalho apresenta a análise de desempenho e o planejamento de capacidade de uma arquitetura de microsserviços espaciais, especificamente, um serviço para *geocodificação de endereços* e um serviço para *cálculo de isócronas de tempo e distância*. Para isso, são realizados experimentos nos quais uma carga de trabalho planejada foi utilizada para determinar o custo e a carga de trabalho suportada de dois projetos livres e amplamente utilizados, *Pelias* e *Open Route Service* (ORS), implantados na AWS. A partir disso determinou-se os valores ideais dos parâmetros de configuração destes projetos.

O Pelias (PELIAS, 2014) é uma plataforma Javascript desenvolvida inicialmente pela Mapzen para geocodificação de endereços, que é um processo computacional no qual endereços

[1] Segundo (MELL; GRANCE, 2011) Infrastructure as a Service (IaaS) é a capacidade fornecida ao consumidor de fornecer processamento, armazenamento, redes e outros recursos computacionais fundamentais onde o consumidor é capaz de implantar e executar um software arbitrário, que pode incluir sistemas e aplicativos.

[2] Segundo (MELL; GRANCE, 2011) Software as a service (SaaS) é a capacidade fornecida ao consumidor de usar os aplicativos do provedor executados em uma infraestrutura de nuvem.

textuais são transformados nas coordenadas (latitude, longitude) onde estão localizados no espaço geográfico (RUSHTON et al., 2007).

O Open Route Service (ORS, 2014) é uma plataforma Java desenvolvida pela GIScience Research Group and HeiGIT para o cálculo de rotas entre pontos e isócronas de tempo e distância que é um processo computacional no qual se define uma região de alcance com determinada distância ou tempo a partir de um ponto (O'SULLIVAN; MORRISON; SHEARER, 2000). Na figura 1.1 é demonstrado uma isócrona de distância com alcance de 5 km dividida em intervalos de 1 km.

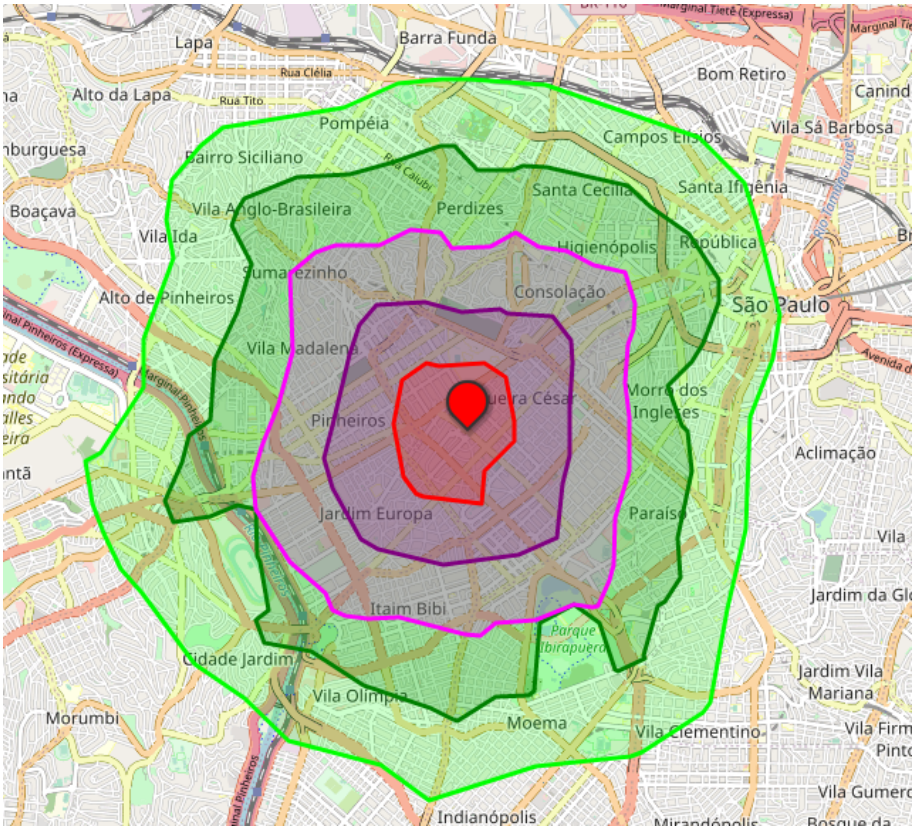

Figura 1.1 – Isócrona com 5 km de alcance e 1 km de intervalo

1.1 Justificativa

Devido à ausência de um trabalho para análise de desempenho das plataformas Pelias e ORS em um ambiente de IaaS, especificamente em uma instancia ec2 t2.2xlarge na AWS, se faz necessária tal avaliação para descobrir os limites dos serviços de geocodificação e cálculo de isócronas em um ambiente de IaaS adotado mundialmente.

1.2 Objetivos

Analisar o desempenho de uma arquitetura de microsserviços espaciais, formada inicialmente por um serviço de geocodificação e um serviço de cálculo de isócronas de tempo e distância. Assim este trabalho possui os seguintes objetivos específicos:

- Seleção de projetos livres e relevantes que oferecem o serviço de geocodificação e cálculo de isócronas
- Caracterizar carga de trabalho na qual os serviços serão submetidos
- Definir ferramentas para execução dos experimentos
- Planejar experimentos
- Realizar experimentos para análise de desempenho
- Analisar métricas coletadas durante os experimentos para determinar o custo e a capacidade de cada serviço, assim como determinar valores ideais de seus parâmetros de configuração.

1.3 Organização do Trabalho

Este trabalho está organizado nos seguintes capítulos:

- Capítulo 1: Introdução
- Capítulo 2: Revisão Bibliográfica, que apresenta a fundamentação teórica e os trabalhos relacionados.
- Capítulo 3: Desenvolvimento, que apresenta os procedimentos metodológicos.
- Capítulo 4: Resultados, que apresenta os resultados obtidos pelos experimentos computacionais apresentados no capítulo 3.
- Capítulo 5: Considerações Finais, que avaliam em que medida os objetivos específicos e gerais foram atingidos, além de propor os trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo apresenta uma revisão bibliográfica e seções de fundamentação teórica e trabalhos relacionados.

2.1 Fundamentação Teórica

Nesta seção são apresentadas as referências bibliográficas nas quais este trabalho é baseado.

2.1.1 Métrica

O termo *métrica* refere-se aos critérios utilizados para avaliar o desempenho do sistema. (JAIN, 1991). Alguns exemplos de métricas são:

- Throughput: Taxa (solicitações por unidade de tempo) na qual as requisições podem ser servidas pelo sistema.
- Uso de CPU: Tempo em percentual na qual a CPU está processando instruções.
- Uso de memória: Quantidade de dados contidas na memória em determinado instante.
- Uso de disco: Quantidade de dados lidos ou escritos no disco por unidade de tempo
- Uso de rede: Quantidade de dados transmitidos através de uma interface de rede por unidade de tempo

2.1.2 Caracterização da carga de trabalho

Para testar várias alternativas em condições idênticas, a carga de trabalho deve ser repetível. Um ambiente de usuário real geralmente não é repetível, é necessário estudar os ambientes de usuário real, observar as principais características e desenvolver um *Modelo de carga de trabalho* que possa ser usado repetidamente. Esse processo é chamado de *Caracterização da carga de trabalho* (JAIN, 1991).

Na literatura de caracterização da carga de trabalho, o termo *Unidade da carga de trabalho* é usado em vez de usuário (JAIN, 1991). As quantidades medidas, solicitações de serviço ou demandas de recursos, que são usadas para modelar ou caracterizar a carga de trabalho, são chamados de *Parâmetros da carga de trabalho* (JAIN, 1991).

2.1.3 Planejamento e gerenciamento de capacidade

O termo *planejamento de capacidade* significa garantir que os recursos computacionais adequados estarão disponíveis para atender às futuras demandas de carga de trabalho de forma econômica enquanto atende aos objetivos de desempenho. (JAIN, 1991)

O termo *gerenciamento de capacidade* é usado para denotar o problema de garantir que os recursos de computação atualmente disponíveis sejam usados para fornecer o melhor desempenho possível. (JAIN, 1991)

Segundo (JAIN, 1991) os passos para o planejamento e gerenciamento de capacidade são basicamente os mesmos:

- Instrumentar o sistema
- Monitorar o uso do sistema
- Caracterizar a carga de trabalho
- Predizer o desempenho em diferentes alternativas
- Selecionar a alternativa com o menor custo e melhor desempenho

2.2 Trabalhos Relacionados

Em (PRENER; FOX, 2021) é proposto o uso de três pacotes na linguagem R, postmastr2 que fornece a funcionalidade de normalização de endereços, gateway4 para geocodificação e censoxy para geocodificação em lote específico para a cidade de St. Louis, Missouri, EUA. Em seguida os compara a geocodificadores existentes no mercado.

Foi construída uma base de dados de homicídios registrados na cidade de St. Louis entre 2008 e 2018 utilizando-se do pacote R, compstair, para baixar todos os crimes relatados ao Departamento de Polícia Metropolitana de St. Louis nos anos entre 2008 e 2018, corrigir problemas comuns, remover registros infundados e combiná-los em uma única tabela representando apenas os crimes ocorridos. Um subconjunto com 1.822 endereços desses homicídios, foram normalizados usando o pacote postmastr para posterior geocodificação.

Para execução dos testes criou-se scripts R que realizam chamadas de API utilizando credenciais gratuitas de sete soluções existentes no mercado sendo estes: Bing, ArcGIS, Geocodio, Google Maps, HERE, TomTom e OpenCage. Através dos scripts R a base de dados é enviada um endereço por vez ou em lote para cada serviço e calcula-se o tempo de resposta de cada requisição para cada API pelo uso da função system.time() no pacote base R.

A execução dos testes mostra um resultado onde a geocodificação de linha única obteve um tempo médio de 650,6 s, em contrapartida, a geocodificação em lote obteve um tempo médio

de geocodificação de 224,4 s. Para geocodificação de linha única o serviço HERE obteve o melhor desempenho com 178,9 s, enquanto o pior desempenho foi para o pacote censoxy. Para a geocodificação em lote o pacote gateway interno obteve o melhor desempenho com tempo de aproximadamente 22 s, enquanto o pior desempenho foi para o serviço Bing com 536 s.

3 Desenvolvimento

Este capítulo apresenta a metodologia, os dados e todas as ferramentas utilizadas para a realização deste trabalho.

3.1 Seleção de projetos livres que implementam geocodificação e cálculo de isócronas

Na seleção destes projetos, considerou-se somente projetos livres e de código aberto. Um segundo critério é a relevância destes projetos em um cenário global. Para isso, considerou-se a quantidade de estrelas no projeto do GitHub, 2.800 para o Pelias e 860 para o ORS.

Neste contexto, a plataforma Pelias foi selecionada para implementação do serviço de geocodificação e a plataforma Open Route Service foi selecionada para implementação do serviço do cálculo de isócronas.

3.1.1 Pelias

O Pelias (PELIAS, 2014) é uma plataforma Javascript de código e base de dados aberta, entre elas, *OpenStreetMap, OpenAddresses* e *Who's On First*. Desenvolvida inicialmente pela *Mapzen*[1] para geocodificação de endereços e baseada na ferramenta de pesquisa textual *Elasticsearch* [2]. O Pelias é dividido em componentes, entre eles:

- Pelias API: Comunica com todos os outros componentes (se disponíveis), Elasticsearch e fornece a interface para todas as consultas ao Pelias
- Placeholder: É usado especificamente para lidar com o componente relacional da geocodificação. O placeholder entende, por exemplo, que Paris é uma cidade em um país chamado França, mas que existe outra cidade chamada Paris no estado do Texas, EUA.
- Libpostal: É uma biblioteca que fornece um analisador de endereços usando um modelo estatístico de processamento de linguagem natural treinado em OpenStreetMap, OpenAddresses e outros dados abertos.
- PIP: Carrega na memória dados de polígonos que representam os limites de cidades, estados, regiões, países etc., e pode realizar cálculos nesses dados geométricos. É usado para determinar se um determinado ponto está em um determinado polígono.

[1] Disponível em: <https://www.mapzen.com/>
[2] Disponível em: <https://www.elastic.co/pt/>

- Interpolation: Combina geometrias de ruas com endereços e intervalos de endereços conhecidos, para permitir estimar a posição de endereços que podem existir, mas não estão em fontes de dados abertas existentes.

No presente trabalho, utilizou-se uma solução disponibilizada oficialmente em seu repositório no Github [3], em que cada componente é executado em seu próprio container Docker.

Para melhor entendimento da interação entre os componentes do serviço de geocodificação, na figura 3.2 é apresentado o diagrama de sequência do serviço de geocodificação.

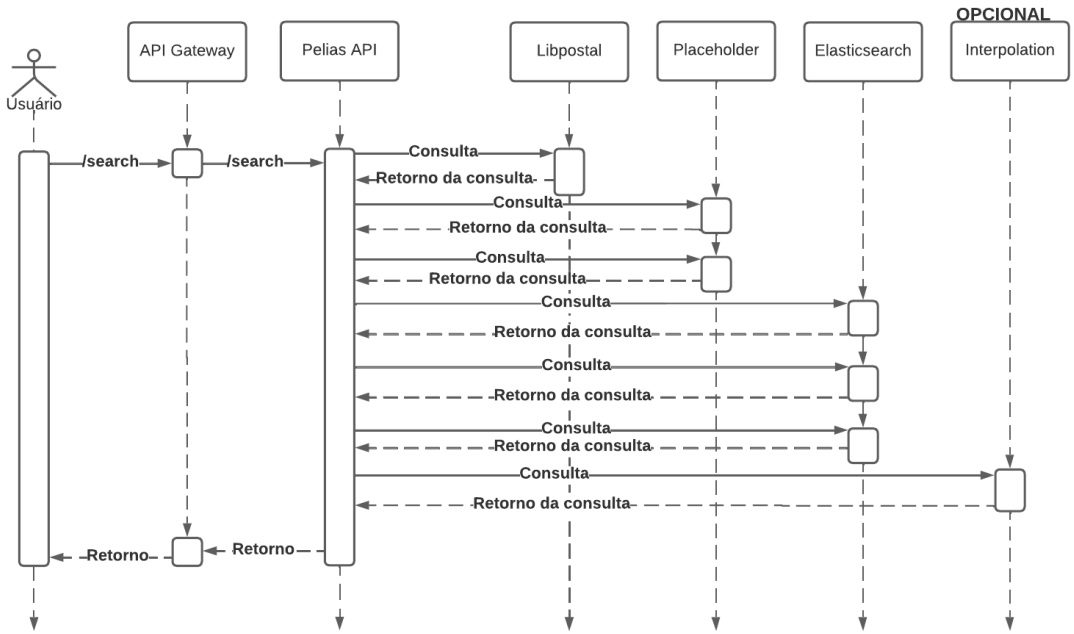

Figura 3.1 – Diagrama de sequência do serviço de geocodificação

3.1.2 ORS

O ORS (ORS, 2014) é uma plataforma Java de código aberto e de base de dados aberta, OpenStreetMap. Desenvolvida pela GIScience Research Group and HeiGIT e oferece as funcionalidades para cálculo de rotas entre pontos, cálculo de matriz tempo x distância e cálculo de isócronas de tempo e distância, mas para a finalidade deste trabalho somente a funcionalidade de cálculo de isócronas será considerada.

No presente trabalho, utilizou-se uma solução disponibilizada oficialmente em seu repositório no Github [4], em que é executado em um container Docker.

Para melhor entendimento do serviço de cálculo de isócronas, na figura 3.2 é apresentado o diagrama de sequência do serviço de cálculo de isócrona.

[3] Disponível em: <https://github.com/pelias/docker>
[4] Disponível em: <https://github.com/GIScience/openrouteservice/tree/master/docker>

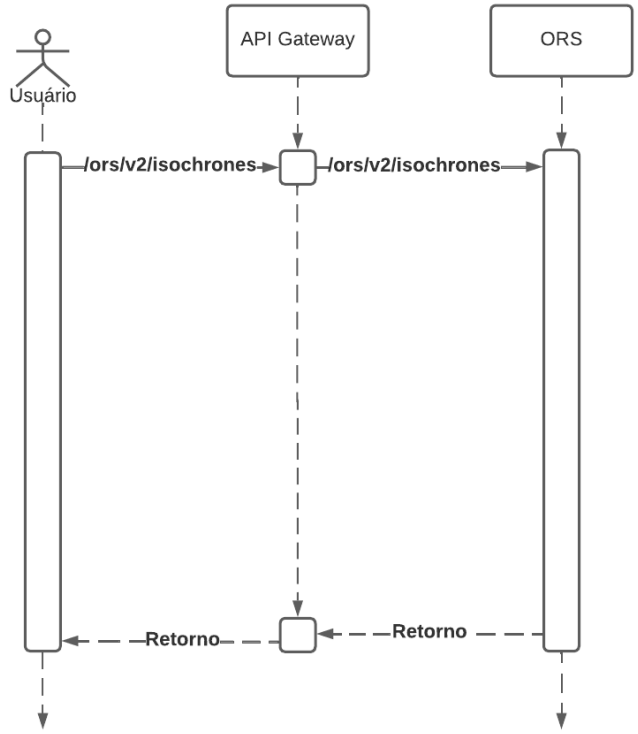

Figura 3.2 – Diagrama de sequência do serviço de cálculo de isócrona

3.2 Caracterização da carga de trabalho

A carga de trabalho foi definida como sendo uma taxa constante de requisições por minuto. Além disso, a fim de se evitar qualquer efeito de memória cache, cada requisição foi realizada a partir de um endereço não repetido.

3.3 Ferramentas utilizadas na coleta de dados

O *Apache JMeter* [5] (JMETER, 1998) é uma plataforma Java desenvolvida pela *Apache Software Foundation* para simular uma carga de trabalho e medir o desempenho de aplicações Web. Neste trabalho o JMeter foi estendido através do plugin *Perfmon*[6], ao qual recebe métricas de um agente em execução em uma máquina remota.

Além disso, para que gargalos pudessem ser identificados, o código fonte do Pelias foi instrumentado com instruções para coleta e registro de métricas de desempenho de cada macro funcionalidades, conforme detalhado na próxima seção de texto.

[5] Disponível em: <https://jmeter.apache.org/>
[6] Disponível em: <https://jmeter-plugins.org/wiki/PerfMon/>

3.4 Planejamento dos experimentos

Para melhor compreensão da arquitetura implantada na AWS durante os experimentos, a figura 3.3 apresenta o diagrama de componentes que demonstra a interação entre os serviços e ferramentas utilizadas.

A fim de criar uma interface única de comunicação do usuário para os serviços Pelias e ORS, foi implantada a *API Gateway*. API Gateway é um proxy reverso para serviços e atua como um único ponto de entrada no sistema (RICHARDSON, 2021). A plataforma *Kong*[7] em sua versão de código livre e configurado para executar em um container Docker, foi selecionada para implantação da API Gateway.

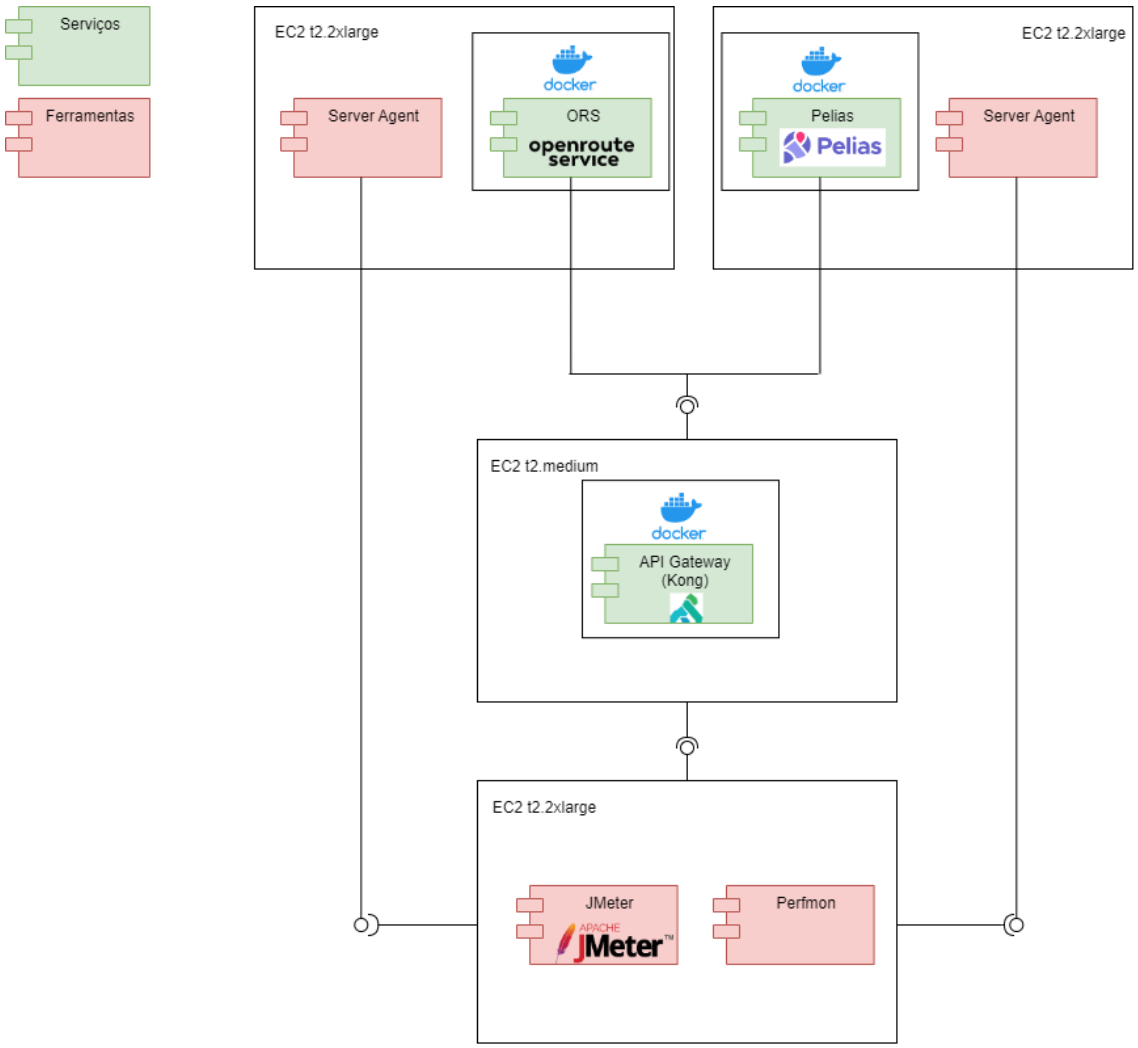

Figura 3.3 – Diagrama de componentes

[7] Disponível em: <https://github.com/Kong/kong>

Para realização dos experimentos foram utilizadas quatro instâncias EC2 instanciadas na AWS:

- t2.2xlarge (8vcpu, 32 gb de memória RAM, 50 gb de ebs, Ubuntu Server 20.04) - Para executar o serviço Pelias
- t2.2xlarge (8vcpu, 32 gb de memória RAM, 50 gb de ebs, Ubuntu Server 20.04) - Para executar o serviço ORS
- t2.2xlarge (8vcpu, 32 gb de memória RAM, 50 gb de ebs, Ubuntu Server 20.04) - Para executar o JMeter
- t2.medium (2vcpu, 4 gb de memória RAM, 8 gb de ebs, Ubuntu Server 20.04) - Para executar a API Gateway (Kong)

3.4.1 Métricas coletadas

Para coleta das métricas foi necessário realizar a instrumentação de código do componente Pelias API disponível neste repositório. As métricas coletadas pelas ferramentas foram:

- Uso de CPU: Obtido através do plugin Perfmon
- Uso de memória: Obtido através do plugin Perfmon
- Uso de disco: Obtido através do plugin Perfmon
- Uso de rede: Obtido através do plugin Perfmon
- Tempo de resposta do Pelias: Obtido através do JMeter
- Tempo de resposta dos componentes do Pelias: Obtido através de instrumentação de código
- Número de tarefas na fila: Obtido através do plugin Perfmon.

4 Resultados

Este capítulo apresenta os resultados obtidos e métricas coletadas através dos experimentos realizados nos serviços de geocodificação e cálculo de isócronas. Diversos experimentos foram realizados, no entanto, para objetividade do texto, discute-se somente aqueles resultados que permitiram alguma descoberta de conhecimento. Demais experimentos serviram tão somente para trazer os autores aos experimentos que são realmente relevantes ao estudo. Portanto, é preciso dizer que, em termos de velocidade e consumo de recursos computacionais, o desempenho do Pelias é totalmente modulado pelo componente Elasticsearch (ES), que se apresenta como principal gargalo do serviço de geocodificação. É importante relatar que, nas análises, todo cuidado foi tomado para ignorar as métricas coletadas nos instantes iniciais dos experimentos, consideradas outliers, devido ao comportamento diferenciado exibido pelo Pelias e ORS no momento inicial. Apesar do ORS e de todos os componentes do Pelias, incluindo o ES, estarem instânciados em memória e totalmente responsivos, o comportamento exibido nos instantes iniciais dos experimentos é diferenciado e as razões para isso continuam desconhecidas pelos autores, exigindo estudos mais aprofundados.

4.1 Experimentos no Pelias

4.1.1 Experimento com timeout de 1 segundo

Este experimento foi realizado com duração de 5 minutos e 5 repetições. As métricas coletadas são reportadas como o comportamento médio observado em todas as repetições. Inicialmente, o Pelias foi configurado com o componente Interpolation desativado, com a configuração padrão de 1.000 para o limite do número de tarefas na fila do ES e com a configuração padrão de timeout de 1 segundo para todos os componentes.

A tabela 4.1.5 apresenta a média do tempo de resposta do ES obtida das 5 repetições do experimento. É possível notar que ao se elevar a taxa de requisições por minuto para 10.000, a média se diverge dos resultados anteriores (10, 100 e 1.000 requisições por minuto) e ocorre um notório aumento na média do tempo de resposta. O coeficiente de variação alto, resultante do aumento significativo do desvio padrão com relação à média, indica a alta instabilidade do ES ao operar nestas condições.

REQ/MIN	**10**	**100**	**1.000**	**10.000**
Média	7,20	5,48	3,81	146,30
Desvio Padrão	1,26	1,85	1,00	255,60
Coef. Vari.	17,45%	33,70%	26,20%	174,71%

Tabela 4.1 – Tempo médio de resposta do ES (ms)

A figura 4.1.1 apresenta a média do tempo de resposta do ES, obtida das 5 repetições do experimento. Observa-se que o tempo de resposta para a taxa de 10.000 requisições por minuto é maior no início do teste até atingir um determinado pico e conforme o experimento continua o tempo de resposta é reduzido até se estabilizar.

Durante o experimento, as primeiras requisições obtiveram erro de timeout no ES. Observando a figura 4.1.1, concluiu-se que a causa do erro de timeout no ES nas primeiras requisições é causado pelo tempo de resposta acima de 1 segundo. No entanto, ainda não conseguimos compreender as razões pelas quais o ES exibe comportamento diferenciado para as primeiras requisições. Visto que houve todo cuidado para que os endereços geocodificados fossem diferentes, não podemos concluir que esse comportamento se deve a um sistema de cache interna ao ES. Por outro lado, podemos apenas imaginar que ele possa ser resultante do tempo de instânciação de threads no pool de threads internas do ES.

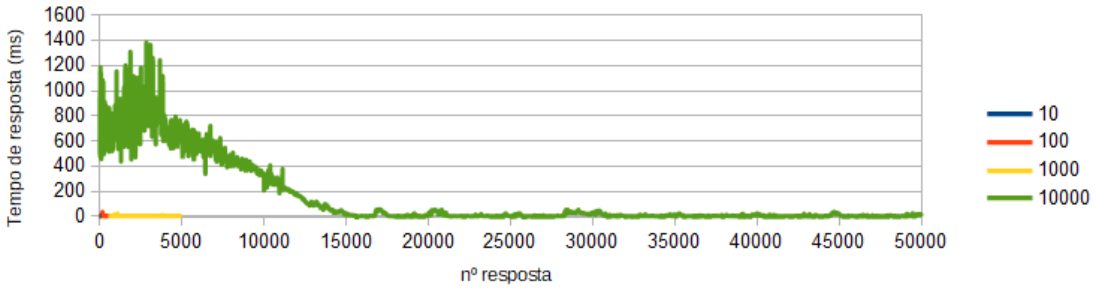

Figura 4.1 – Tempo médio de resposta do ES

Observa-se na figura 4.2 que o número de tarefas na fila do ES é baixo frente ao seu limite de 1.000 tarefas. No entanto, observa-se que existe um pico nos instantes iniciais do experimento.

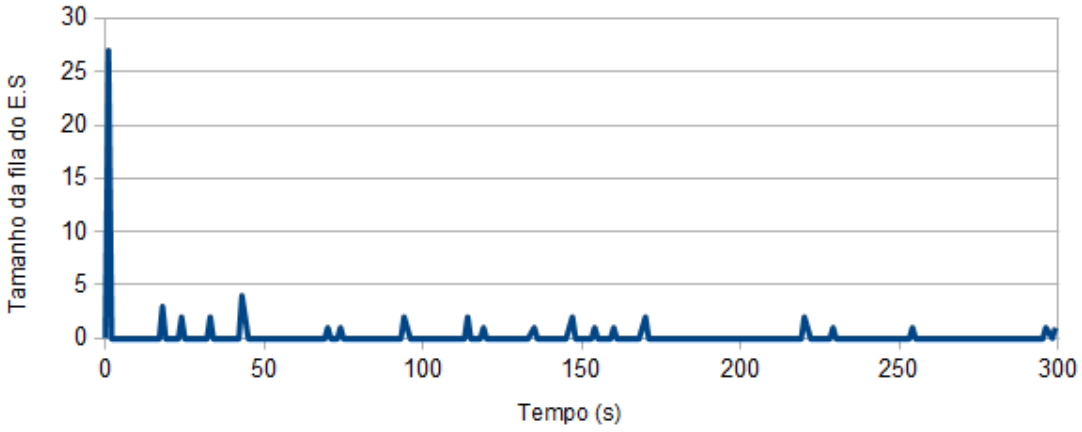

Figura 4.2 – Número de tarefas na fila do ES para 10.000 requisições/min

Devido ao erro de timeout um novo experimento foi realizado com o timeout configurado para 300 segundos.

4.1.2 Experimento com timeout de 300 segundos

Este experimento foi realizado com duração de 5 minutos e 3 repetições. O Pelias foi configurado com o componente Interpolation desativado, com a configuração padrão de 1.000 para o limite do número de tarefas na fila do ES e com a configuração de timeout de 300 segundos para todos os componentes.

Em comparação com o experimento anterior, neste não se obteve o erro de timeout. Porém ao executar este experimento, ocorreu o erro de máxima capacidade da fila no ES. Observando a figura 4.3, é notório que no início do experimento a fila atinge seu limite de 1.000 tarefas causando o erro de máxima capacidade da fila.

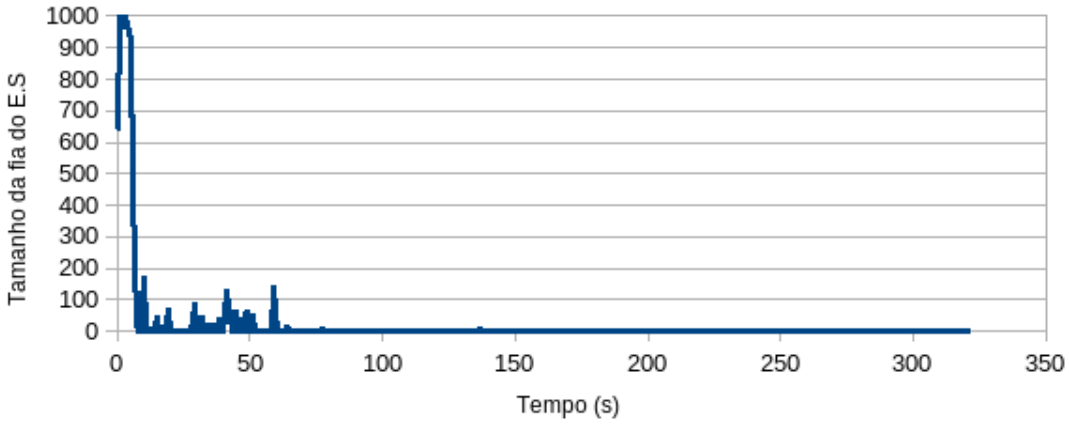

Figura 4.3 – Número de tarefas na fila do ES para 13.000 requisições/min

Até a taxa de 13.000 requisições por minuto o erro de máxima capacidade da fila no ES acontece somente no início do teste. Porém como demonstra a figura 4.4 a partir da taxa de 14.000 requisições por minuto o erro de máxima capacidade da fila no ES acontece não somente no início do teste, mas repetidas vezes durante o teste.

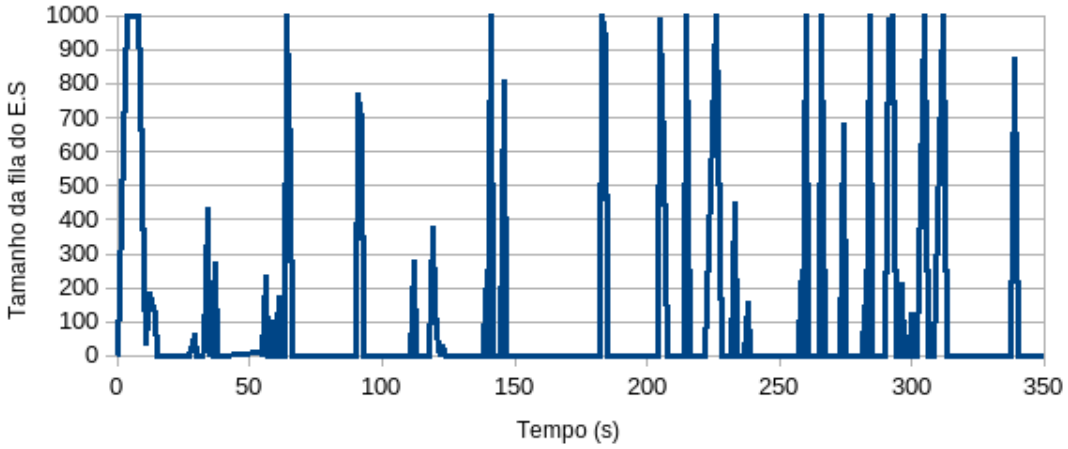

Figura 4.4 – Número de tarefas na fila do ES para 14.000 requisições/min

Como uma tentativa de resolver o erro de máxima capacidade da fila, um novo experimento foi realizado com o tamanho máximo da fila configurado para 3.000 tarefas.

4.1.3 Experimento com limite de tarefas na fila do ES igual a 3000 tarefas

Este experimento foi realizado com duração de 5 minutos e 1 repetição apenas, devido à observância de um desvio padrão muito pequeno nas métricas coletadas no experimento anterior. O Pelias foi configurado com o componente Interpolation desativado, com a configuração de 3.000 para o limite do número de tarefas na fila do ES e com a configuração de timeout de 300 segundos para todos os componentes.

Em comparação com o experimento anterior, como apresentado na figura 4.5, o erro de máxima capacidade da fila ainda persiste, apresentando alta variabilidade.

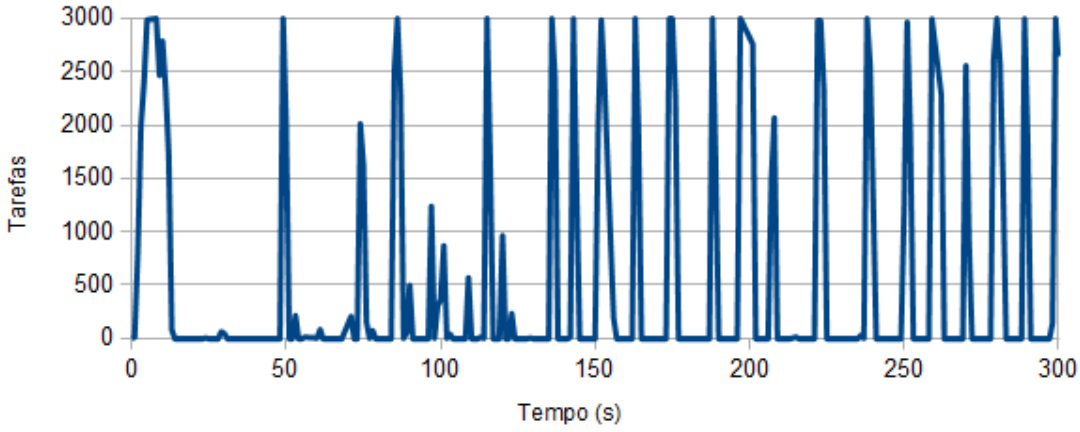

Figura 4.5 – Número de tarefas na fila do ES para 14.000 requisições/min

4.1.4 Experimento utilizando o componente Interpolation

Este experimento foi realizado com duração de 5 minutos e 3 repetições. O Pelias foi configurado com o componente Interpolation ativado, com o tamanho da fila no ES padrão de 1.000 tarefas e com a configuração de timeout de 60 segundos para todos os componentes.

Até o momento, os experimentos realizados não utilizavam o componente Interpolation. O intuito agora é justamente avaliar qual é o impacto sobre o desempenho geral do Pelias causado por esta funcionalidade que depende da aplicação de métodos numéricos realizados sobre a geometria da malha de estrada.

As figuras 4.6 e 4.7 apresentam o uso de CPU(%) para a taxa de 10.500 e 11.000 requisições por minuto, respectivamente. Comparando-se os resultados, observa-se na figura 4.7 que após um determinado período o sistema se comporta de forma instável.

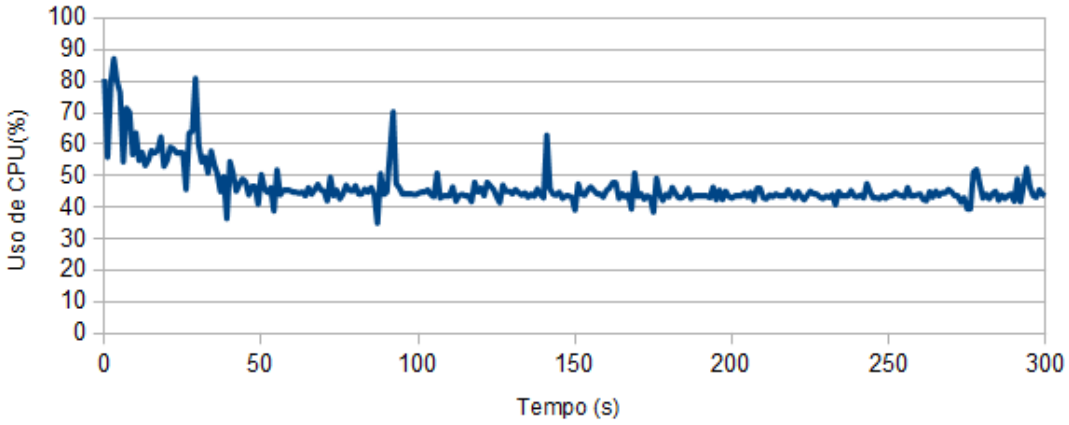

Figura 4.6 – Uso de CPU (%) para 10.500 requisições/min

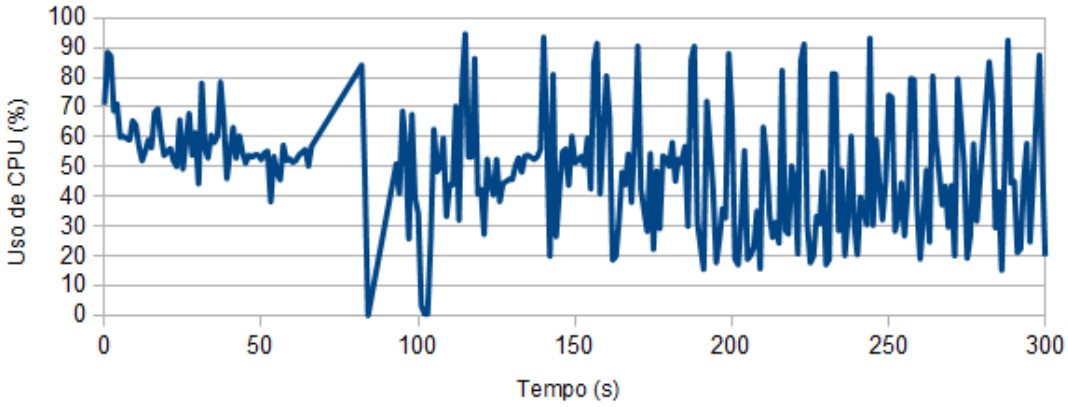

Figura 4.7 – Uso de CPU (%) para 11.000 requisições/min

A figura 4.8 apresenta o número de tarefas na fila do ES para a taxa de 11.000 requisições por minuto. Em comparação com os experimentos que não utilizam o componente Interpolation, neste experimento o erro de máxima capacidade da fila começa a ocorrer a partir da taxa de 11.000 requisições por minuto.

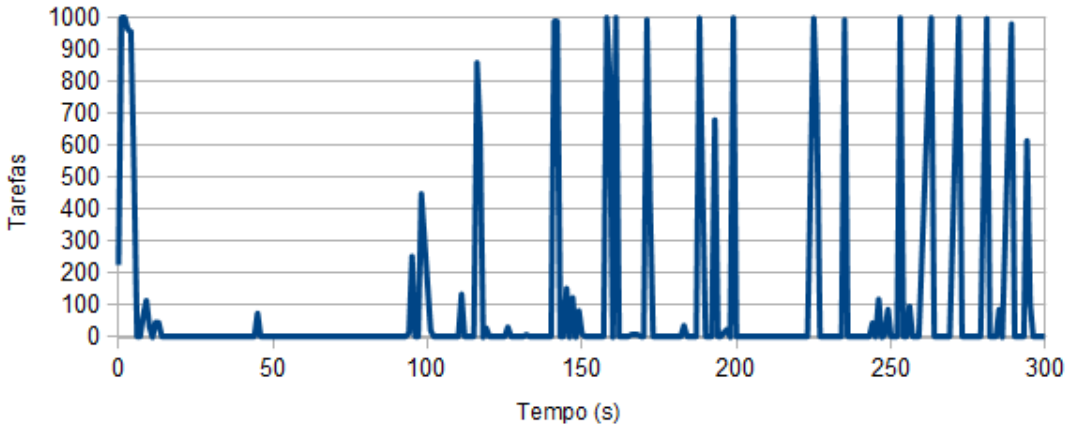

Figura 4.8 – Numero de tarefas na fila do ES para 11.000 requisições/min

4.1.5 Experimento variando o número de threads no pool do ES

Este experimento foi realizado com duração de 5 minutos e 3 repetições. O Pelias foi configurado com o componente Interpolation desativado, com o tamanho da fila no ES padrão de 1.000 tarefas e com a configuração de timeout de 60 segundos para todos os componentes. Utilizou-se a taxa de requisições por minuto de 13.000, o valor máximo antes do sistema se tornar instável, como demonstrado nos experimentos anteriores.

Observando as tabelas da média do tempo de resposta abaixo conclui-se que o tamanho padrão do pool com 13 threads obteve o pior desempenho comparado ao pool com 8 e 18 threads, para os quais obteve-se resultados aproximados.

Threads	**8**	**13**	**18**
Média	1.375,16	2.458,86	1.396,38
Desvio Padrão	2.431,31	4.096,33	2.722,00
Coef. Vari.	176,80%	166,59%	194,93%

Tabela 4.2 – Tempo médio de resposta do Pelias API (ms)

Threads	**8**	**13**	**18**
Média	314,71	470,84	302,85
Desvio Padrão	733,57	1.049,91	745,43
Coef. Vari.	233,10%	222,99%	246,14%

Tabela 4.3 – Tempo médio de resposta do ES (ms)

Threads	**8**	**13**	**18**
Média	243,24	505,31	241,70
Desvio Padrão	584,71	1.537,96	674,69
Coef. Vari.	240,39%	304,36%	279,14%

Tabela 4.4 – Tempo médio de resposta do Placeholder (ms)

Threads	**8**	**13**	**18**
Média	330,99	634,95	369,18
Desvio Padrão	809,17	1.632,79	1.129,57
Coef. Vari.	244,47%	257,15%	305,96%

Tabela 4.5 – Tempo médio de resposta do Libpostal (ms)

4.2 Experimentos no ORS

Esta seção apresenta os resultados obtidos nos experimentos no ORS. Para todos os experimentos realizou-se com duração de 5 minutos e 5 repetições para cada alteração de variável, utilizou-se uma taxa constante de 600 requisições por minuto e entre cada repetição a instância ec2 foi reinicializada.

4.2.1 Experimento com variação no número de intervalos da isócrona

Este experimento foi realizado com isócronas contendo um intervalo de 1 km, ou seja, o número de intervalos aumenta à medida em que o alcance da isócrona também aumenta. O maior alcance de isócrona que conseguiu-se atingir através desse experimento foi de 11 km.

Observa-se na figura 4.9 que há a tendência de aumento no uso de cpu, à medida em que o alcance da isócrona e o número de intervalos aumentam.

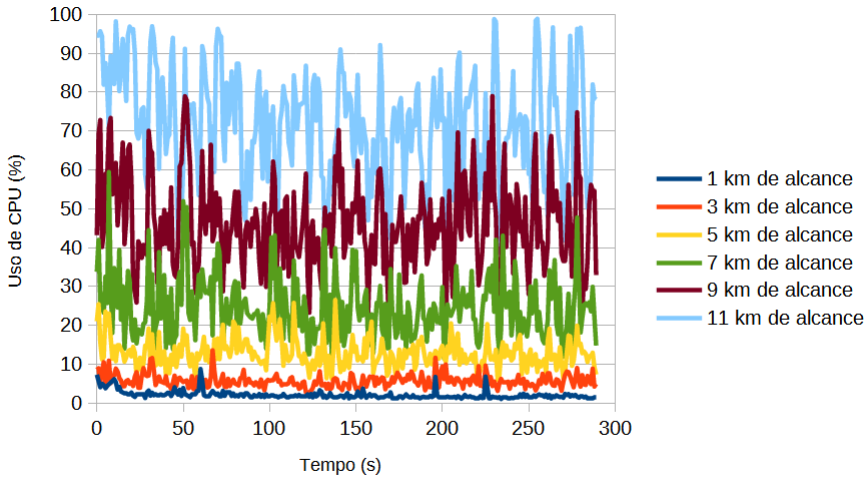

Figura 4.9 – Uso de CPU (%)

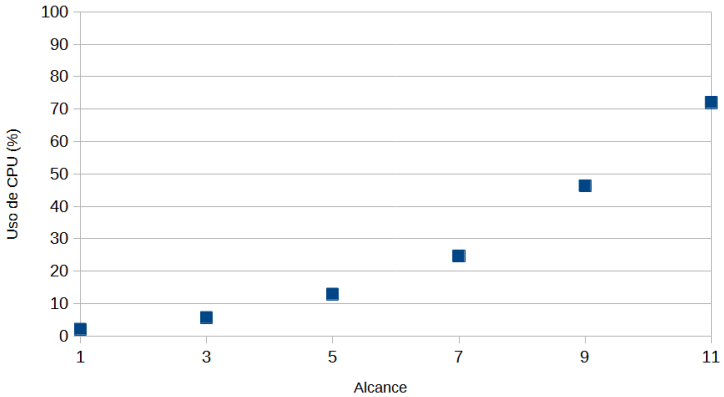

Figura 4.10 – Média do uso de CPU (%)

Observa-se na figura 4.11 que independentemente do alcance da isócrona, existe a tendência do uso de memória convergir para um determinado valor, aproximadamente 25 Gigabytes. E quanto maior o alcance da isócrona mais rapidamente o uso de memória converge para esse patamar assintótico.

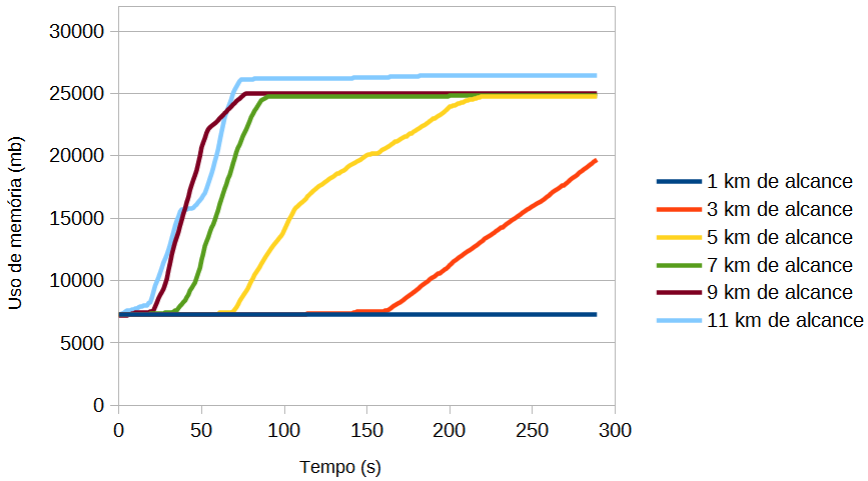

Figura 4.11 – Uso de memória (mb)

4.2.2 Experimento com variação no alcance da isócrona

Este experimento foi realizado com isócronas contendo um único intervalo, ou seja, o alcance da isócrona é igual à largura de seu único intervalo da isócrona. O maior alcance de isócrona que conseguiu-se atingir através desse experimento foi de 20 km.

Observa-se nas figuras 4.12 e 4.13 que há a tendência de aumento no uso de cpu, à medida em que o alcance da isócrona aumenta.

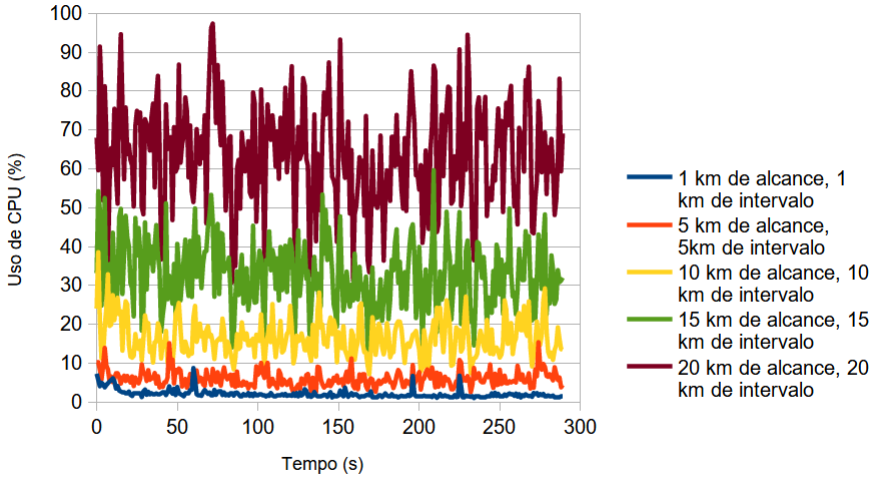

Figura 4.12 – Uso de CPU (%)

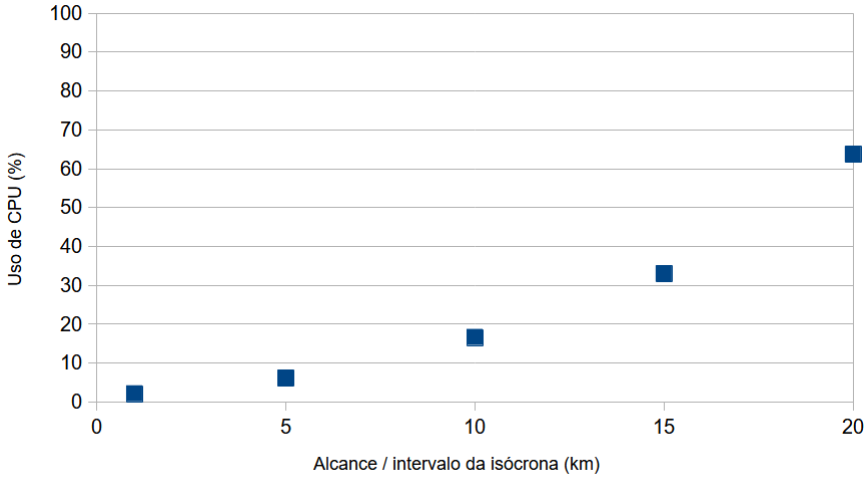

Figura 4.13 – Média do uso de CPU (%)

Observa-se na figura 4.14 que assim como no experimento anterior, existe a tendência do uso de memória convergir para um determinado valor assintótico, aproximadamente 27 Gigabytes. E quanto maior o alcance da isócrona mais rapidamente o uso de memória converge.

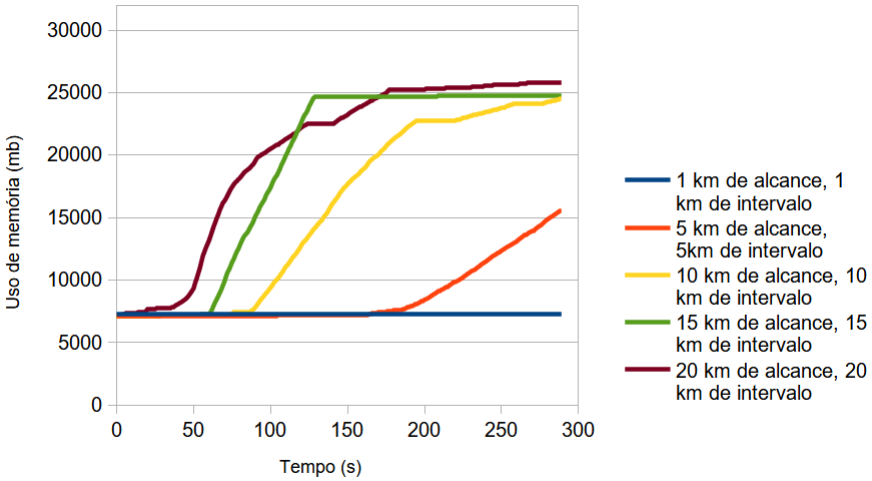

Figura 4.14 – Uso de memória (mb)

5 Considerações Finais

5.1 Conclusão

Inicialmente este trabalho propôs a seleção de dois projetos livres para compor uma arquitetura de microsserviços espaciais. Os projetos Pelias e ORS foram selecionados para implementação dos serviços de geocodificação e de cálculo de isócronas, respectivamente. Após a seleção dos projetos, foi caracterizada a carga de trabalho à qual esses serviços seriam submetidos, foram selecionadas métricas de desempenho a serem coletadas, foram planejados os experimentos a serem realizados e estudadas as ferramentas a serem utilizadas.

Diante das análises das métricas coletadas, conclui-se que para o serviço de geocodificação a taxa máxima de requisições por minuto que uma instância EC2 t2.2xlarge na AWS suporta, é de 13.000 requisições com o componente Interpolation desativado e 10.500 requisições com o componente Interpolation ativado. Após diversas mudanças de parâmetros no Pelias, conclui-se que aumentar o valor padrão do número máximo de tarefas na fila não faz diferença no desempenho do serviço, e que 8 ou 18 threads no pool do ES obtém-se desempenho superior comparado ao valor padrão 13.

Para os experimentos realizados no ORS, considerando uma taxa constante de 600 requisições por minuto, isócronas de distância com intervalos de 1 km e para uma instância EC2 t2.2xlarge, conclui-se que 11 km é o alcance máximo suportado. Considerando isócronas de distância com intervalo único, e para uma instância EC2 t2.2xlarge, conclui-se que 20 km é o alcance máximo suportado.

Segundo a ferramenta *AWS pricing calculator*[1] para estimativas de custo e conforme demonstra a tabela 5.1, no momento de realização deste trabalho, considerando instâncias EC2 do tipo *sob demanda* e a arquitetura de microsserviços demonstrada neste trabalho, mais especificamente formada por uma instância EC2 t2.medium para a API Gateway, uma instância EC2 t2.2xlarge para ORS e uma instância EC2 t2.2xlarge para o Pelias, o custo mínimo para manter a arquitetura hospedada na região de São Paulo (sa-east-1) disponível a qualquer momento é de 948,01 USD por mês. [2]

[1] Disponível em: <https://calculator.aws>
[2] Disponível em: <https://calculator.aws/#/estimate?id=6efa0ad972af7638571f4b5c4bf9df6476f69f75>

Serviço	**Custo Mensal**
Serviço de geocodificação (Pelias)	444,00 USD
Serviço de cálculo de isócronas (ORS)	444,00 USD
API Gateway (Kong)	60,01 USD
Total	**948,01 USD**

Tabela 5.1 – Estimativa de custo mensal da arquitetura de microsserviços com instâncias sob demanda

Este custo pode ser reduzido ao considerar-se a reserva destas instâncias por um período igual ou superior a 1 ano. Por exemplo, a tabela 5.1 demonstra o custo para a arquitetura de microsserviços considerando o modelo de definição de preço *EC2 Instance Savings Plans*[3], com 3 anos de reserva. [4]

Serviço	**Custo Mensal**
Serviço de geocodificação (Pelias)	198,35 USD
Serviço de cálculo de isócronas (ORS)	198,35 USD
API Gateway (Kong)	29,28 USD
Total	**425,98 USD**

Tabela 5.2 – Estimativa de custo mensal da arquitetura de microsserviços com instâncias reservadas

5.2 Trabalhos Futuros

Como trabalhos futuros é considerado realizar a análise de desempenho para isócronas de tempo.

[3] Disponível em: <https://aws.amazon.com/pt/savingsplans/>
[4] Disponível em: <https://calculator.aws/#/estimate?id=763415c6ef01f0a00e3d0c6264c67449f50397e1>

Referências

CARVALHO, M. de. *Construindo o saber: técnicas de metodologia científica*. [S.l.]: Papirus Editora, 1989. ISBN 9788530800710.

FOWLER, M. *Microservices Guide*. 2019. Disponível em: <https://martinfowler.com/microservices>. Acessado em 15/05/2022.

JAIN, R. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley, 1991. ISBN 978-0-471-50336-1.

JMETER. *JMeter*. [S.l.], 1998. Disponível em: <https://jmeter.apache.org/>.

MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. [S.l.]: Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2011.

ORS. *ORS*. 2014. Disponível em: <https://openrouteservice.org/>.

O'SULLIVAN, D.; MORRISON, A.; SHEARER, J. Using desktop gis for the investigation of accessibility by public transport: an isochrone approach. *International Journal of Geographical Information Science*, Taylor & Francis, v. 14, n. 1, p. 85–104, 2000.

PELIAS. *Pelias*. 2014. Disponível em: <https://pelias.io/>.

PRENER, C. G.; FOX, B. Creating open source composite geocoders: Pitfalls and opportunities. *Transactions in GIS*, Wiley Online Library, v. 25, n. 4, p. 1868–1887, 2021.

RAMPAZZO, L. *Metodologia científica*. [S.l.]: Edições Loyola, 2005. ISBN 9788515024988.

RICHARDSON, C. *API Gateway*. [S.l.], 2021. Disponível em: <https://microservices.io/patterns/apigateway.html>.

RUSHTON, G.; ARMSTRONG, M. P.; GITTLER, J.; GREENE, B. R.; PAVLIK, C. E.; WEST, M. M.; ZIMMERMAN, D. L. *Geocoding health data: the use of geographic codes in cancer prevention and control, research and practice*. [S.l.]: CRC Press, 2007.