UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IAGO TITO OLIVEIRA

UMA AVALIAÇÃO DO MODELO FUNCTION AS A SERVICE PARA EXECUÇÃO DE ALGORITMOS DE PROCESSAMENTO DE LINGUAGEM NATURAL

CAMPINA GRANDE - PB

2022

IAGO TITO OLIVEIRA

UMA AVALIAÇÃO DO MODELO FUNCTION AS A SERVICE PARA EXECUÇÃO DE ALGORITMOS DE PROCESSAMENTO DE LINGUAGEM NATURAL

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Thiago Emmanuel Pereira

CAMPINA GRANDE - PB

2022

IAGO TITO OLIVEIRA

UMA AVALIAÇÃO DO MODELO FUNCTION AS A SERVICE PARA EXECUÇÃO DE ALGORITMOS DE PROCESSAMENTO DE LINGUAGEM NATURAL

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Thiago Emmanuel Pereira
Orientador – UASC/CEEI/UFCG

Reinaldo Cezar de Morais Gomes
Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro
Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de Setembro de 2022.

CAMPINA GRANDE - PB

RESUMO

O projeto ePol, parceria da Polícia Federal com a Universidade Federal de Campina Grande, tem como objetivo desenvolver tecnologias para execução de algoritmos de processamento de linguagem natural que manipulam grandes quantidades de dados provenientes de processos de investigação policial. Um dos requisitos das tecnologias é que os programadores dos algoritmos não se preocupem com a implantação de seus programas no ambiente de execução. Ainda, é importante que a execução dos algoritmos seja eficiente. Nesse contexto, o modelo arquitetural de Function as a Service (FaaS) pode ser uma opção viável por permitir que desenvolvedores criem algoritmos quaisquer sem se preocupar com a infraestrutura de execução. Considerando este problema e os requisitos para sua solução, este trabalho analisa uma abordagem com a ferramenta OpenFaas e compara com a abordagem criada durante o projeto, elencando vantagens e desvantagens e avaliando se existe uma melhor solução para o problema enfrentado. A comparação foi feita através da análise de métricas coletadas de protótipos do sistema, construídos utilizando as abordagens selecionadas. Ao final do estudo, foi constatado que utilizar o OpenFaas para gerenciar os processos oferece desempenho equivalente em relação ao sistema desenvolvido até então no projeto, além de trazer funcionalidades adicionais.

Uma avaliação do modelo Function as a Service para execução de algoritmos de processamento de linguagem natural

Iago Tito Oliveira
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
iago.oliveira@ccc.ufcg.edu.br

Thiago Emmanuel Pereira
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
temmanuel@computacao.ufcg.edu.br

RESUMO

O projeto ePol, parceria da Polícia Federal com a Universidade Federal de Campina Grande, tem como objetivo desenvolver tecnologias para execução de algoritmos de processamento de linguagem natural que manipulam grandes quantidades de dados provenientes de processos de investigação policial. Um dos requisitos das tecnologias é que os programadores dos algoritmos não se preocupem com a implantação de seus programas no ambiente de execução. Ainda, é importante que a execução dos algoritmos seja eficiente. Nesse contexto, o modelo arquitetural de *Function as a Service* (*FaaS*) pode ser uma opção viável por permitir que desenvolvedores criem algoritmos quaisquer sem se preocupar com a infraestrutura de execução. Considerando este problema e os requisitos para sua solução, este trabalho analisa uma abordagem com a ferramenta OpenFaas e compara com a abordagem criada durante o projeto, elencando vantagens e desvantagens e avaliando se existe uma melhor solução para o problema enfrentado. A comparação foi feita através da análise de métricas coletadas de protótipos do sistema, construídos utilizando as abordagens selecionadas. Ao final do estudo, foi constatado que utilizar o OpenFaas para gerenciar os processos oferece desempenho equivalente em relação ao sistema desenvolvido até então no projeto, além de trazer funcionalidades adicionais.

PALAVRAS-CHAVE

FaaS, OpenFaaS, avaliação de desempenho

1. INTRODUÇÃO

A polícia federal do brasil está passando por um processo de transformação digital que facilita o processo de investigação de casos e inquéritos, através da criação automatizada de grafos de investigação ou a sumarização das informações mais relevantes dos documentos. Nesse contexto, o projeto ePol desenvolve tanto tecnologias de processamento de linguagem natural e inteligência artificial para a extração de tais informações, quanto o próprio sistema que dará suporte a essas tecnologias.

Alguns requisitos devem ser considerados, visto a natureza dos dados que serão processados. Por se tratarem de dados com propósitos diversos, inúmeros algoritmos podem ser desenvolvidos ou atualizados ao longo do tempo. Por isso, a implantação dos algoritmos deve ser facilitada para o programador. Esse requisito básico aproxima o sistema de uma plataforma *serverless*, especificamente com a dinâmica de *Function as a Service* (*FaaS*) [1]. Neste tipo de plataforma os desenvolvedores não precisam se preocupar com a complexidade de construir e manter a infraestrutura de processamento. A eficiência do sistema, que pode ser um desafio por si só para as plataformas *FaaS* [2], também é um importante fator para a escolha da tecnologia mais adequada, visto o grande número de processos que deverão executar.

Então, a fim de prover o melhor design para o sistema do projeto, este estudo compara a tecnologia utilizada na prática no projeto ePol para o gerenciamento e execução de funções com a ferramenta OpenFaas, uma escolha popular para desenvolvimento de plataformas *serverless* [3] [4]. A comparação foi feita a partir da construção de pequenos servidores que iniciam os processamentos de maneiras diferentes e da coleta dos tempos necessários para concluir o processamento de atividades do mesmo tipo.

Foi possível concluir que o uso do OpenFaaS para controle dos processos que precisam rodar apresenta eficiência equivalente ao sistema do ePol, ou seja, sem pioras no desempenho, mesmo contanto com ferramentas e camadas a mais para funcionar. O seu uso também provém outras vantagens, como ferramentas prontas, que podem ser úteis para o desenvolvimento do sistema.

O restante do artigo está dividido da seguinte forma. A seção 2 contextualiza as demandas da Polícia Federal e esclarece o que precisa ser feito. A seção 3 descreve o design dos experimentos feitos para analisar a eficiência que utilizar a ferramenta OpenFaaS pode proporcionar. A seção 4 apresenta os resultados obtidos. A seção 5 conta com as considerações finais sobre o impacto do uso da ferramenta.

2. CONTEXTUALIZAÇÃO

Anualmente, a Polícia Federal recebe, em média, um milhão de novos casos para investigar. Além dos casos novos, também existe toda a base de dados de anos passados que, apesar de inicialmente terem uma prioridade menor, também deverão ser processados. O projeto ePol surge com o objetivo de ajudar na investigação desses casos através do processamento dos mesmos por algoritmos de inteligência artificial com os mais diversos objetivos, tais como: extrair informações relevantes dos casos,

relacionar casos semelhantes ou procurar casos duplicados, ou seja, redigidos em momentos ou por pessoas diferentes, mas que tratam do mesmo ocorrido.

Processar esse volume de dados com desempenho adequado não é trivial. Um único caso pode exigir horas de processamento. Existe também a possibilidade de ser necessário reprocessar algum caso com o mesmo algoritmo, em uma situação que o algoritmo foi mal desenvolvido, acarretando em um bug, ou quando precisar ser atualizado para algo mais preciso.

Além disso, os programadores dos algoritmos têm muito trabalho para, além de desenvolver a lógica dos processos, implantar os algoritmos de maneira rápida, eficiente, segura e isolada, visto que diferentes algoritmos podem exigir versões diferentes de alguma dependência, no ambiente de execução.

2.1 ARQUITETURA DO SISTEMA

Para ter uma melhor noção das regras de negócio e de como os sistemas devem se comportar, primeiramente é preciso entender a arquitetura existente no sistema. A figura 1 ilustra como o sistema de processamento da equipe de grafos ePol está organizado.

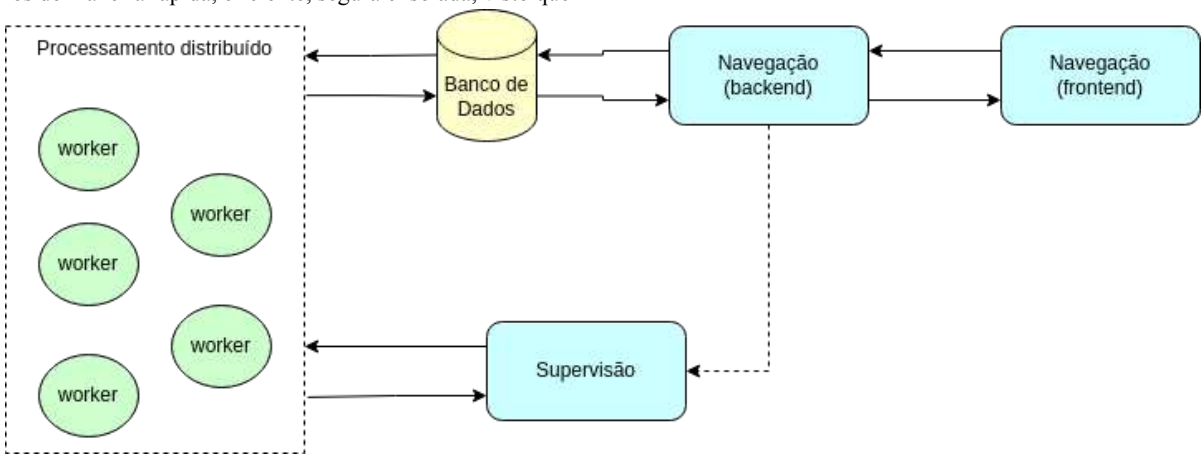

Figura 1: arquitetura do ePol grafos

O ePol grafos é dividido em 3 subsistemas principais:

1) Subsistema de Navegação: o sistema de navegação é onde o usuário interage com o sistema, visualizando dados ou solicitando o processamento de novos dados (não é o foco do estudo).

2) Subsistema de processamento: neste sistema, os programas, aqui chamados de *workers*, irão rodar. Pode ser visto como o próprio conjunto de todos os *workers* em execução, processando dados e salvando-os no banco de dados.

3) Subsistema de supervisão: parte do sistema que vai controlar o processamento e é o principal foco do estudo. O sistema de supervisão é responsável por receber demandas de processamento do usuário ou de outros *workers* e gerenciar o sistema de processamento.

O gerenciamento realizado pelo supervisor consiste em proporcionar alocação em máquinas que atendam aos pré-requisitos necessários para execução, controlar uso de recursos, despertar novos *workers* responsáveis pelo tipo de processamento solicitado, identificar *workers* problemáticos, interromper *workers* que não evoluem adequadamente e acompanhá-los até que tenham finalizado seu trabalho.

O Supervisor contém uma *API REST* com *endpoints* para criação de tarefas e operação do sistema e com um *daemon* executando para o gerenciamento dos *workers*. O *daemon*, inicialmente, observa as novas tarefas criadas e inicia uma instância do *worker* responsável por aquele tipo de processamento. Depois, realiza todos os procedimentos para gerenciamento de execução citados anteriormente.

O sistema também garante tolerância a falhas, através da alocação de tarefas pelo mecanismo de *leasing* [5]. Neste mecanismo um *worker* deve se responsabilizar em finalizar uma atividade em um certo período de tempo. Caso haja algum problema no *worker* ou na infraestrutura, como uma falta de energia, ao fim do tempo estipulado, a tarefa é realocada para outra instância do *worker* processá-la. Como consequência, os processamentos feitos devem ser pensados para serem idempotentes, já que há risco de que as mesmas tarefas sejam executadas mais de uma vez, ainda que o último passo não seja concluído.

Como os *workers* formam um conjunto de processos que podem se tornar muito pesados, seja pela quantidade e/ou complexidade, essa execução de processos é feita de forma distribuída. O supervisor desperta e gerencia os processos dos *workers* remotamente nas máquinas cadastradas para recebê-los.

Os *workers*, por sua vez, são basicamente módulos python, com uma função principal a ser executada. Com o *worker* cadastrado, ao receber uma solicitação do tipo do *worker*, o supervisor desperta uma instância dele em uma máquina e espera a confirmação de que ele finalizou corretamente, a sinalização de um erro crítico, ou se mais tempo do que o permitido for decorrido, assume-se que houve um erro, seu processo é finalizado e outro é criado para tentar novamente.

3. METODOLOGIA

Para testar o sistema proposto, foram desenvolvidos dois servidores com o mesmo comportamento, mas utilizando de

estratégias diferentes para iniciar os *workers*. O primeiro, seguindo a estratégia utilizada no projeto, utiliza o protocolo SSH para se comunicar com a máquina remota onde o código do *worker* está localizado e iniciar seu processo diretamente no sistema operacional da máquina em questão. O segundo utiliza o protocolo HTTP para se comunicar com uma instância do OpenFaaS, que por sua vez, gerencia a execução do *worker*.

Por simplicidade, apenas uma máquina remota foi utilizada, e a fim de obter resultados para operações genéricas que podem ser executadas no ambiente proposto, apenas um *worker* foi criado: o *Worker NoOp*.

O *Worker NoOp* consiste em uma função que apenas retorna um valor indicando sucesso para indicar ao servidor que ele finalizou corretamente. Dessa maneira é possível testar a inicialização de um processo e o tempo necessário para que a comunicação aconteça, ignorando o tempo do processamento da função criada, que pode variar de acordo com o que se deseja processar.

Para atingir o objetivo de analisar o processamento de tarefas em paralelo, foram criados testes para o *worker* processar 100, 1000, e 10000 tarefas, com no máximo 10 em paralelo de cada vez. Com o intuito de reduzir ruídos como latência de rede oscilante, cada teste foi executado 10 vezes para se analisar a média e o desvio padrão dos tempos necessários para cada teste. O tempo de processamento começa a ser calculado depois do servidor iniciar, mas antes dele começar a iniciar os processos dos *workers*, e termina ao fim do processamento do último *worker*, mas antes de desligar o servidor. O servidor foi finalizado e iniciado novamente a cada teste.

Os servidores tiveram seus *deploys* feitos em contêineres do Docker. Para o servidor que utiliza o OpenFaas, a ferramenta precisou ser alocada em um cluster do Kubernetes, que por sua vez também foi posto dentro de um contêiner Docker. As Figuras 2 e 3 mostram como os servidores, ferramentas e processos estão organizados. A figura 4 mostra em mais detalhes a arquitetura interna do OpenFaaS. Os processos dos *workers* no sistema do ePol são executados diretamente no sistema operacional da máquina remota que hospeda os processos, enquanto o OpenFaaS cria contêineres para executar as funções.

Os experimentos foram realizados em uma máquina com processador de 8 núcleos Intel(R) Core(TM) i7-1185G7 3.00GHz de 11ª geração e 16GB de memória RAM executando o Ubuntu 22.04.1 LTS e kernel do Linux 5.15.0-46-generic. Foram também utilizados: Python 3.10.6; Docker 20.10.17 build 100c701; Kubectl com Client v1.24.3, Kustomize v4.5.4 e Server v1.24.0; Kind versão 0.14.0; e OpenFaas com o faas-cli versão 0.14.5.

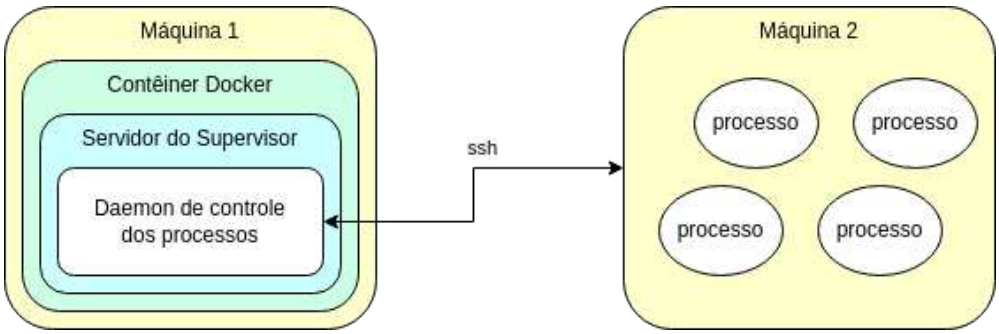

Figura 2: Projeto do servidor e dos processos utilizado no ePol

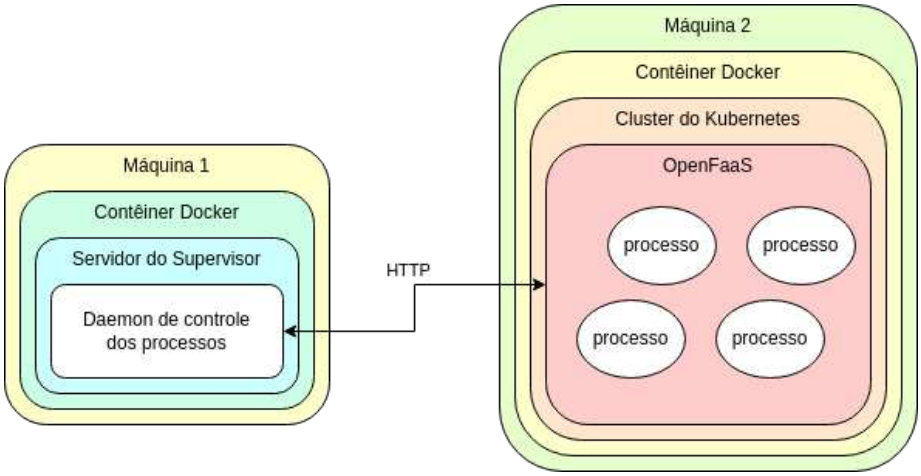

Figura 3: Projeto do servidor e dos processos utilizando o OpenFaaS

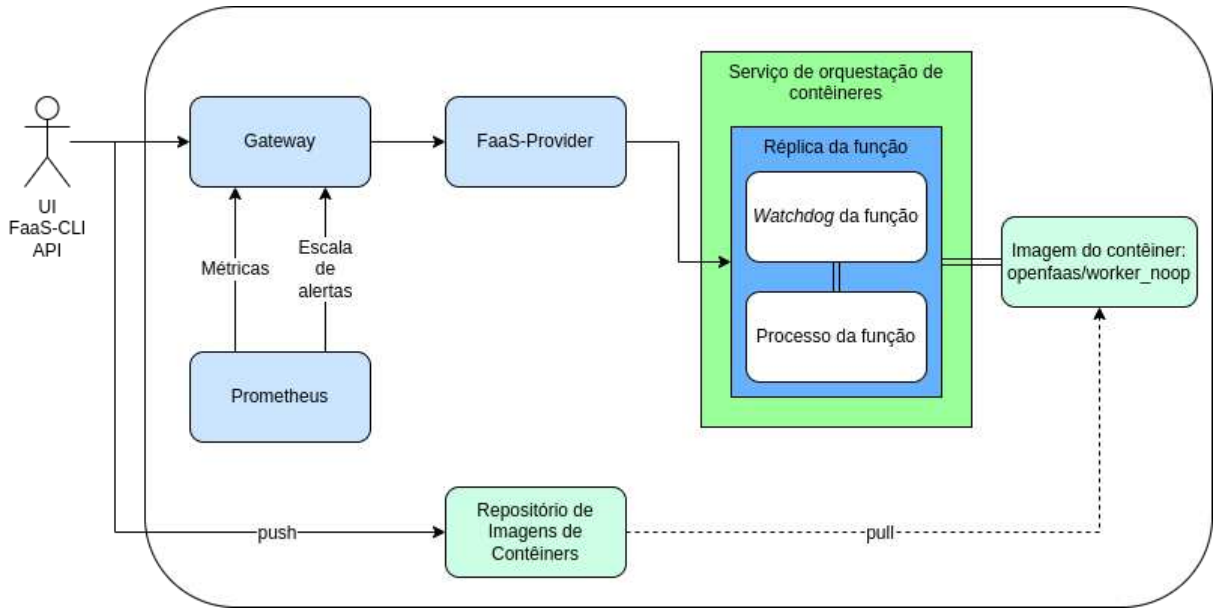

Figura 4: Componentes principais do OpenFaaS. As funções são armazenadas em um Repositório de imagens de contêiner, e posteriormente baixadas e usadas para criar novas Réplicas das Funções. Adaptado de [6]

Tabela 1: Tempo médio, em segundos, e desvio padrão de 10 testes do sistemas do ePol e do sistema utilizando o OpenFaas

	Tempo médio ePol	Tempo médio OpenFaaS	Desvio padrão ePol	Desvio padrão OpenFaaS
100 tarefas	3.1	1.9	0.08	0.08
1000 tarefas	37.7	26.2	2.67	2.99
10000 tarefas	402.7	291.5	2.50	5.12

4. RESULTADOS

Primeiro, vamos analisar se houve uma melhora nos tempos para executar os *workers*. A Tabela 1 mostra o tempo médio de execução dos testes, e o desvio padrão destes, utilizando o *Worker NoOp*. O objetivo é analisar os tempos necessários para iniciar e finalizar os processos que devem ser executados, ou que acabaram de executar, bem como a consistência dos mesmos.

É possível notar uma melhoria nos tempos necessários para o sistema iniciar e finalizar os *workers* quando utiliza a ferramenta OpenFaas para gerenciar os processos. Também podemos afirmar que o tempo de processamento ao utilizar o OpenFaaS é estatisticamente menor do que utilizar o sistema do ePol (*p-value* = 4.78e-09 para 1000 tarefas, e *p-value* < 2.2e-16 para 100 e 10000 tarefas), e que o *speed-up* foi de 1.6, 1.4 e 1.3 para 100, 1000 e 10000 tarefas respectivamente. A tabela 2 mostra o ganho de tempo que utilizar o OpenFaas garantiu em relação ao uso do sistema atual do ePol e a tabela 3 mostra o *p-value* de um teste-t de Student com 99% de confiança.

Tabela 2: Ganho de tempo médio ao utilizar o OpenFaas para iniciar processos

número de tarefas	*speed-up*
100	1.6
1000	1.4
10000	1.3

Tabela 3: p-value de um teste-t de Student, com 99% de confiança, sobre os dados

número de tarefas	*p-value*
100	< 2.2e-16
1000	4.786e-09
10000	< 2.2e-16

5. CONCLUSÃO

No cenário proposto, é válido notar que utilizar o OpenFaaS proveu um desempenho equivalente ao sistema desenvolvido no ePol, ou seja, sem perdas de desempenho. Tal equivalência de desempenho acontece mesmo com o OpenFaaS necessitando de muitos recursos do lado do servidor, como o Kubernetes, e possuindo uma pilha de camadas para funcionar. Ainda, essa equivalência acontece mesmo em um cenário na presença de *cold starts*, um atraso típico em plataformas FaaS [6][7]. Além disso, utilizar o OpenFaas pode trazer outras vantagens para o futuro do projeto e do sistema, como a presença de ferramentas de *Auto Scaling* e de observabilidade [4]. Usar o Kubernetes também pode oferecer vantagens como *Load Balancer* e outras. Utilizar as ferramentas já prontas livraria a equipe e o projeto de ter que se preocupar com tais funcionalidades, o que permitiria um foco maior em outras necessidades da Polícia Federal.

Portanto, utilizar o OpenFaas para inicialização de processos de maneira distribuída se mostrou consistentemente equivalente, em relação ao tempo necessário para iniciar os processos, em comparação a utilizar a solução desenvolvida até então pelo projeto ePol. Além disso, o uso do OpenFaas apresenta diversas outras vantagens no âmbito de ferramentas e funcionalidades para o controle de processamento de *workers*, sem comprometer em nada o desempenho do sistema.

6. AGRADECIMENTOS

Primeiro gostaria de agradecer, à minha mãe, Jaqueline Tito, meu pai, Itanajé Fernandes e minha irmã, Aísla Tito, por sempre me apoiarem e me incentivarem, bem como toda a minha família. Agradeço também a todos os amigos e colegas que conheci e com quem compartilhei a vida acadêmica, em especial Brener Quevedo, Diego Ribeiro, Edelly Brandão, Paulo Mateus e Raiany Rufino, e à minha namorada, Íris Almeida, que me acompanharam nos mais diversos momentos dentro e fora da graduação. Eu nada seria se não fossem vocês.

Segundo, agradeço à Universidade Federal de Campina Grande e a todos os professores com quem tive a oportunidade de aprender, por me proporcionar um ensino de alta qualidade na área da Ciência da Computação.

Quero também agradecer a todos os integrantes do projeto ePol, alunos e professores, com quem passei mais de 2 anos aprendendo e crescendo como pessoa, aluno e profissional, bem como à Polícia Federal por acreditar em nós e fazer esse projeto ser possível. Em especial, gostaria de agradecer ao professor Dalton Serey, responsável pela minha equipe, e com quem aprendi muito sobre computação e áreas relacionadas; e também aos integrantes da minha equipe, Débora Leda, Francicláudio Dantas, Francisco Igor, Franklin Régis, Vinícius Rodrigues e Thiago Lima, com quem interagi mais diretamente e pude aprender e compartilhar conhecimentos que agregaram muito à minha vida.

Por fim, agradeço ao meu orientador, o professor Thiago Emmanuel, por me ajudar a realizar este trabalho da melhor forma possível e me passar conhecimentos que me auxiliarão em pesquisas futuras.

7. REFERÊNCIAS

[1] Baldini, Ioana & Castro, Paul & Chang, Kerry & Cheng, Perry & Fink, Stephen & Isahagian, Vatche & Mitchell, Nick & Muthusamy, Vinod & Rabbah, Rodric & Slominski, Aleksander & Suter, Philippe. (2017). Serverless Computing: Current Trends and Open Problems. 10.1007/978-981-10-5026-8_1.

[2] Eyk, Erwin & Iosup, Alexandru & Abad, Cristina & Grohmann, Johannes & Eismann, Simon. (2018). A SPEC RG Cloud Group's Vision on the Performance Challenges of FaaS Cloud Architectures. 21-24. 10.1145/3185768.3186308.

[3] Li, Junfeng & Kulkarni, Sameer & Ramakrishnan, K. & Li, Dan. (2021). Analyzing Open-Source Serverless Platforms: Characteristics and Performance. 10.18293/SEKE2021-129.

[4] Balla, David & Maliosz, Markosz & Simon, Csaba. (2020). Open Source FaaS Performance Aspects. 358-364. 10.1109/TSP49548.2020.9163456.

[5] [5] Jain, Prashant & Kircher, Michael. (2000). Leasing.

[6] Silva, P., Fireman, D., & Pereira, T. E. (2020, December). Prebaking functions to warm the serverless cold start. In Proceedings of the 21st International Middleware Conference (pp. 1-13).

[7] QUARESMA, David; FIREMAN, Daniel; PEREIRA, Thiago Emmanuel. Controlling garbage collection and request admission to improve performance of faas applications. In: 2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). IEEE, 2020. p. 175-182