FOMA: Um *Framework* para Orquestração, Monitoramento e Automatização de Ambientes de Névoa

Sayonara S. Araújo[1], **Atslands R. da Rocha**[1], **Flávio R. C. Sousa**[1], **Nídia G. S. Campos**[2]

[1]Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará (UFC)
Campus do Pici – Fortaleza, CE – Brasil

[2]Instituto Federal do Ceará (IFCE)
Campus Fortaleza – Fortaleza, CE – Brasil

sayonarasantos@alu.ufc.br, atslands@ufc.br
flaviosousa@ufc.br, nidia@ifce.edu.br

Abstract. *Fog Computing brings computational power at the edge of the network to the Internet of Things solutions, providing processing and storage services. Implementing Fog environments at a large scale has the challenge of managing their nodes and services, together with the resource limitations of local machines. In this context, we present FOMA, a framework that aims to facilitate the management of Fog environments. The framework brings container orchestration functionalities, machine monitoring, and automation of tasks. The implementation of FOMA focused on easy deployment, open-source tools, and low consumption of physical resources. For this purpose, we use the tools K3s, kubectl, Telegraf, InfluxDB, Grafana, and Ansible. The performance evaluation results show that FOMA is a viable and interesting solution for environments with Fog nodes with a low capacity of physical resources.*

Resumo. *A Computação em Névoa proporciona às aplicações de Internet das Coisas um maior poder computacional na borda da rede por meio de serviços de processamento e armazenamento. Implementar um ambiente de Névoa em escala acarreta o desafio de gerenciamento da sua quantidade relevante de nós e serviços, somado às limitações de recursos das máquinas locais. Nesse contexto, apresenta-se o FOMA, um framework que visa facilitar o gerenciamento de ambientes de Névoa, trazendo funcionalidades de orquestração de contêiner, monitoramento de máquinas e automatização de tarefas. A implementação do framework teve como foco a implantação facilitada, o uso de ferramentas de código aberto e o baixo consumo de recursos. Para isso, foram utilizadas as ferramentas K3s, kubectl, Telegraf, InfluxDB, Grafana e Ansible. Os resultados da avaliação de desempenho mostram que o FOMA é uma solução viável e interessante para ambientes com nós de Névoa com baixa capacidade de recursos físicos.*

1. Introdução

Internet das Coisas (*Internet of Things* - IoT) é um conceito da computação referente à interconexão de objetos do cotidiano com a rede mundial de computadores, a Internet. No contexto atual, os objetos IoT podem ser variados tais como *smartwatches*, *smartphones*, automóveis, máquinas industriais, entre outros. Esses dispositivos com capacidade

computacional viabilizam sua comunicação com pessoas e outros dispositivos, tendo um potencial de uso nas mais diversas áreas das atividades humanas [Santos et al. 2016].

Os objetos de IoT podem gerar uma quantidade abundante de dados, tornando uma tarefa desafiadora processar e interpretar esses dados para o mundo real. Isso porque esses objetos possuem uma capacidade reduzida de memória, processamento, armazenamento, entre outras restrições. Para atender essa demanda de processamento dos dados, uma das medidas tomadas é o uso de infraestruturas com Nuvem, devido aos serviços sob demanda e a escalabilidade de recursos fornecida pelo paradigma de Computação em Nuvem [Naha et al. 2018].

A Computação em Névoa emergiu de um contexto de Internet das Coisas com soluções baseadas em Nuvem, no qual se fomentou uma mudança desse modelo para atender requisitos demandados em IoT, como mobilidade, localidade e baixa latência. Tem-se, então, o paradigma da Computação em Névoa estendendo os serviços de Computação em Nuvem para a borda da rede, em escala distribuída [Coutinho et al. 2016]. O termo Computação em Névoa (*Fog Computing*) foi proposto por pesquisadores da Cisco Systems e refere-se a um paradigma de computação distribuída, no qual o processamento é realizado na extremidade da rede, com a infraestrutura integrada à Nuvem [Bonomi et al. 2012].

Devido à quantidade de dispositivos e serviços, o gerenciamento desses elementos faz parte dos desafios da Computação em Névoa, como expõem os autores [Yousefpour et al. 2019]. No mercado, existem diversas ferramentas que auxiliam o gerenciamento de máquinas e aplicações, como orquestradores de contêineres, *softwares* de automatização de tarefas, pilhas de monitoramento, entre outras tecnologias. Entretanto, para cenários de IoT, cuidados adicionais devem ser tomados na escolha dessas ferramentas. Afinal, esses cenários possuem limitações, como a pouca disponibilidade de recursos físicos nos dispositivos da Névoa.

Diante desse contexto, este trabalho apresenta o FOMA, um *Framework* de Orquestração, Monitoramento e Automatização para ambientes de Névoa, que pode contribuir para gerência de máquinas e serviços desses ambientes. A implementação do *framework* teve como foco o baixo consumo de recursos e o uso de soluções de código aberto. Ao longo deste documento, o FOMA e sua implementação serão apresentados. Para tanto, o artigo se divide em cinco seções: a presente introdução, Trabalhos Relacionados, *Framework*, Implementação, Avaliação e Considerações Finais.

2. Trabalhos relacionados

Diante do contexto apresentado, é pertinente abordar quais trabalhos atuais foram encontrados em bases de pesquisas acadêmicas com temas referente ao gerenciamento em Névoa. O recorte temporal encontra-se no período de 2017 a 2020. Dentre as pesquisas que formulam sistemas para ecossistema de Internet das Coisas com gerenciamento e provisionamento de serviços através da orquestração de contêiner na borda da rede, os trabalhos que se mostram relevantes são apresentados nos parágrafos seguintes.

[Hoque et al. 2017] realizaram uma análise técnica de diferentes ferramentas de orquestração para uma infraestrutura de Névoa e também uma avaliação sobre como os contêineres podem afetar o desempenho dos aplicativos nos nós da infraestrutura. Nesse

trabalho foi proposto um *framework* com orquestração de contêiner para Computação em Névoa, utilizando o orquestrador Docker Swarm[1] e um conjunto de ferramentas proprietárias voltado a IoT.

A pesquisa de [Wöbker et al. 2018] apresenta a Fogernetes, uma plataforma de Computação em Névoa baseada no orquestrador Kubernetes[2] que permite o gerenciamento e implantação de serviços em dispositivos heterogêneos com diferentes recursos. Fogernetes demonstra um esquema que combina os requisitos do serviço com os recursos do dispositivo, usando os componentes *labels* e *selectors* do Kubernetes. Com um estudo de caso, foram realizados testes na plataforma, examinando sua aplicabilidade prática na implantação e gerenciamento de serviços.

[Leskinen 2020] investigou a aplicabilidade da distribuição Kubernetes K3s[3] em um contexto industrial de borda de IoT, configurando um ambiente real e realizando testes de carga. O *cluster* K3s foi implantado de forma automatizada, utilizando o Ansible[4] e o *playbook* oficial do K3s. O autor configurou também uma ferramenta de entrega contínua no ambiente e acrescentou o monitoramento para medir o desempenho do *cluster*.

A pesquisa de [El Khalyly et al. 2020] apresenta um metamodelo genérico do ecossistema da IoT baseado em microsserviços. O modelo possui suporte do Docker[5] para criação dos contêineres, do Kubernetes para o provisionamento dos serviços e do Ansible para instalação dos componentes ao sistema e realização monitoramento.

Diferente dos trabalhos de [Leskinen 2020] e [El Khalyly et al. 2020], [Hoque et al. 2017] e [Wöbker et al. 2018] não apresentaram uma forma de implantação facilitada das ferramentas na infraestrutura. No entanto, [Leskinen 2020] demonstrou a implantação automatizada apenas para a ferramenta de orquestração, e [El Khalyly et al. 2020] não apresentaram uma avaliação prática do metamodelo. Nenhuma das pesquisas abordou uma alternativa de gerenciamento remoto do *cluster* do orquestrador. Em contrapartida, o *framework* FOMA proposto nesta pesquisa apresenta a implantação automatizada tanto das ferramentas de orquestração bem como das ferramentas de monitoramento e também permite o gerenciamento remoto do *cluster*. Além disso, o FOMA foi avaliado de maneira prática em um cenário real.

3. *Framework* de Gerenciamento de Névoa

O FOMA é um *framework* desenvolvido para ambientes de Névoa que proporciona a orquestração de serviços em contêineres, o monitoramento de recursos de máquina e a automatização de tarefas de tecnologia da informação, permitindo uma implantação facilitada. A Fig. 1 apresenta a arquitetura geral do FOMA constituída por quatro módulos, que apresentam os seguintes papéis e componentes:

- **Módulo Alocador**: tem como função essencial receber e executar os serviços que serão provisionados em contêiner por meio do Agente de Orquestração. Este é o módulo que deve ser implantado nos nós de Névoa para executar os serviços de

[1]https://docs.docker.com/engine/swarm/
[2]https://kubernetes.io/
[3]https://k3s.io/
[4]https://www.ansible.com/
[5]https://www.docker.com/

processamento e armazenamento. O módulo também possui o Agente de Monitoramento que coleta informação sobre os recursos físicos da máquina;

- **Módulo Intermediário**: controla o conjunto de Agentes de Orquestração através do Servidor de Orquestração. Bem como o Módulo Alocador, o Intermediário contém o Agente de Monitoramento, que coleta informação sobre os recursos;
- **Módulo Gerenciador**: oferece ao usuário uma forma de interagir via linha de comando com a infraestrutura. O módulo permite o gerenciamento remoto do *cluster* do orquestrador, utilizando a Interface de Usuário de Orquestração, e proporciona o acionamento das tarefas automatizadas, através do Servidor de Automatização;
- **Módulo de Monitoramento**: executa os componentes que mantêm o monitoramento centralizado de recursos. Esses componentes são o Serviço de Armazenamento, que possui as informações coletadas pelos Agentes de Monitoramento, e o Serviço de Visualização, que exibe essas informações.

Todos os módulos possuem o Agente de Automatização. Esse agente é necessário para a automatização de tarefas, incluindo a instalação e a configuração das ferramentas de orquestração e monitoramento nas máquinas.

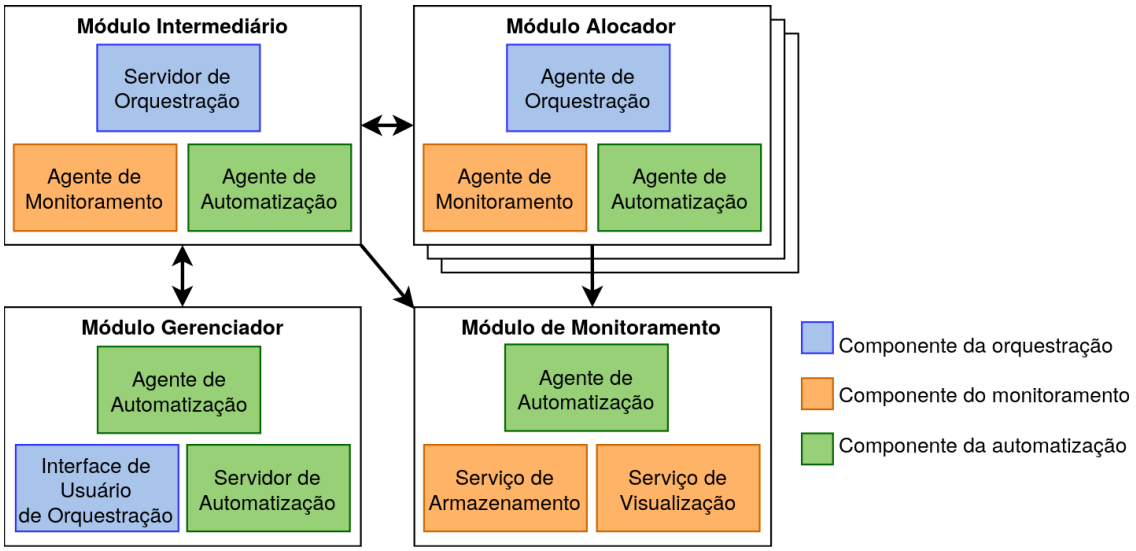

Figura 1. Arquitetura do FOMA.

4. Implementação do *framework* voltada ao baixo consumo de recurso

Para a implementação do FOMA, realizou-se um levantamento de soluções que eram projetos de código aberto, direcionadas para ambientes de IoT e com especificações que mostrassem pouco consumo de recursos. Assim, as ferramentas selecionadas foram o K3s, o kubectl[6], o Telegraf[7], o InfluxDB[8], o Grafana[9] e o Ansible para proporcionar as três principais funcionalidades do *framework*.

O responsável pela orquestração é o projeto de código aberto K3s, uma distribuição leve do Kubernetes desenvolvida para IoT. O Kubernetes oferece gerência de

[6]https://kubernetes.io/docs/reference/kubectl/overview/
[7]https://www.influxdata.com/time-series-platform/telegraf/
[8]https://www.influxdata.com/products/influxdb/
[9]https://grafana.com/

tarefas como provisionamento de serviço em contêiner, escalonamento de réplicas, etc. O *cluster* do K3s é formado pelo K3s Agent, que executa os serviços em contêiner, e pelo K3s Server, que controla o conjunto de K3s Agent. Portanto, na arquitetura do FOMA, os papéis do Servidor e do Agente de Orquestração do FOMA são realizados pelo K3s Server e o K3s Agent, respectivamente. Para a Interface de Usuário de Orquestração, utilizou-se a ferramenta de linha de comando do Kubernetes, o kubectl.

O monitoramento de recursos das máquinas do *cluster* é realizado pela pilha de ferramentas Telegraf, InfluxDB e Grafana (TIG), que permite a coleta de informações de dispositivos, a visualização desses dados em painéis e a criação de alertas. Optou-se por essa pilha devido à demanda de pouco recurso do coletor de dados Telegraf e a facilidade do manuseio do banco, que utiliza uma linguagem parecida com SQL. O TIG funciona da seguinte maneira: o Telegraf captura as informações desejadas e as envia para um banco no InfluxDB, e o Grafana consulta esse banco no InfluxDB para exibir as informações em seus painéis com gráficos e tabelas. Conforme a arquitetura do *framework* FOMA, o Agente de Monitoramento é o coletor Telegraf, o Serviço de Armazenamento é feito pelo InfluxDB, e o Serviço de Visualização, pelo Grafana.

A automatização de tarefas de TI é proporcionada pelo Ansible, que pode configurar sistemas, implantar *softwares* e gerenciar ações mais avançadas (e.g. entrega contínua) a partir de um servidor central. Escolheu-se essa ferramenta por sua configuração simples e leve, sem a instalação de um agente próprio nos dispositivos gerenciados. Para o Ansible realizar o acesso remoto às máquinas e executar as tarefas, os dispositivos devem possuir acesso SSH configurado e o pacote Python instalado, programa nativo na maioria das distribuições Linux. Assim, na arquitetura do FOMA, o Servidor de Automatização é o serviço Ansible e o Agente de Automatização é formado pelo pacote Python e o acesso SSH.

A Fig. 2 apresenta a disposição das ferramentas na arquitetura do FOMA, seguindo a mesma ordem dos componentes na Fig. 1. As ferramentas estão destacadas com a cor que corresponde com sua contribuição nas principais funcionalidades do *framework*: orquestração de contêineres, monitoramento de recursos e automatização de tarefas.

Com o uso do Ansible e seus módulos, adaptou-se o *playbook* oficial do K3s e desenvolveram-se tarefas automatizadas referentes à implantação das ferramentas de orquestração e monitoramento na infraestrutura. O resultado desse desenvolvimento é um projeto, com *playbooks* e arquivos de configuração, disponível na Internet[10]. Depois da implantação com esses *playbooks*, o usuário terá a sua disposição:

I. O *cluster* do orquestrador instalado e configurado, com os K3s Agents e o K3s Server nas máquinas dos módulos Alocadores e Intermediário;
II. O kubectl instalado e configurado para gerenciar o *cluster* remotamente a partir do Módulo Gerenciador;
III. O InfluxDB instalado no computador do Módulo de Monitoramento e configurado com o usuário e o banco de dados para o Telegraf;
IV. O Telegraf instalado nos nós do *cluster* K3s, configurado com as informações do banco criado no InFluxDB;
V. O Grafana também instalado no computador do Módulo de Monitoramento e disponível na porta 3000.

[10]Repositório online: https://github.com/sayonarasantos/FOMA

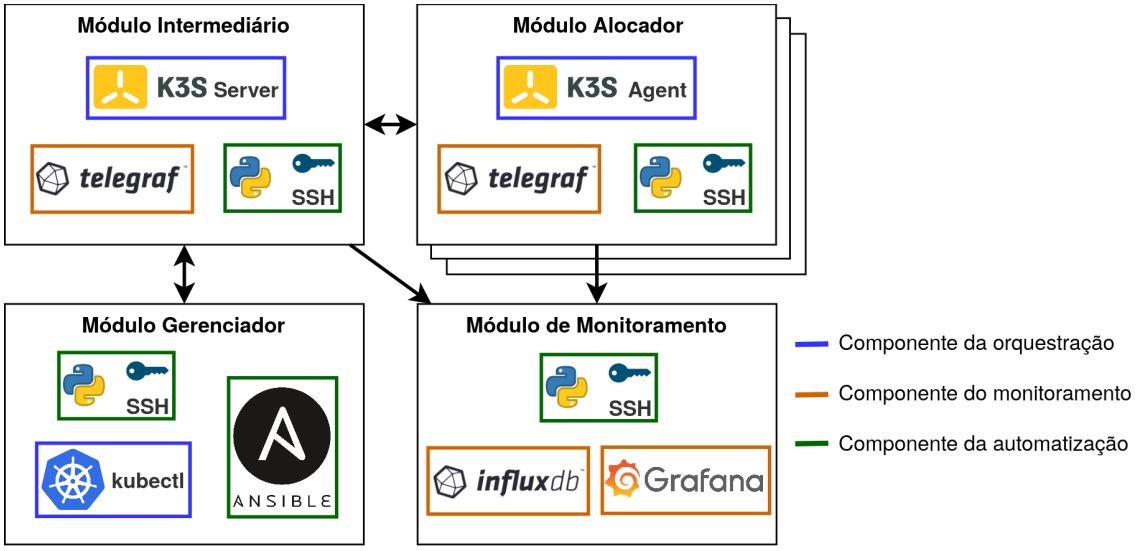

Figura 2. Arquitetura da implementação do FOMA.

Em resumo, com o FOMA, o usuário terá a sua disposição um conjunto de elementos que permite um gerenciamento facilitado da infraestrutura de Névoa, composto por orquestração de contêineres, monitoramento de recursos e automatização de tarefas.

5. Avaliação de Desempenho

Esta seção apresenta o uso e a análise do *framework* na prática. Para isso, foram realizados experimentos referentes ao desempenho da implementação desenvolvida.

5.1. Cenário

Para a avaliação, três computadores foram configurados de forma a receber os módulos do FOMA. Para o Módulo Intermediário, foi utilizada uma placa Raspberry Pi 4 com 1 GB de RAM, 32 GB de armazenamento e processador Broadcom BCM2711, Quad core. Para o Módulo Alocador, empregou-se um Raspberry Pi 3 com 1 GB de RAM, 16 GB de armazenamento e processador Broadcom BCM2837, Quad core. Os Módulos Gerenciador e de Monitoramento foram aplicados em um notebook Lenovo IdeaPad Z400 com 8 GB de RAM, 350 GB de armazenamento e processador Intel® Core™ i7-3632QM, Quad core. A rede Wi-Fi do cenário possuía as velocidades de conexão de 15 Mbps para *download* e 2 Mbps para *upload*. Para avaliar o provisionamento de um serviço IoT no K3s, escolheu-se o Eclipse Mosquitto[11], um servidor de código aberto de mensagens que implementa o protocolo MQTT. O Mosquitto foi selecionado por ser uma aplicação bastante utilizada em projetos de Internet das Coisas e por possuir uma imagem oficial no registro Docker Hub[12].

Com esse ambiente montado, foram realizados os seguintes experimentos:

I. Consumo de recursos físicos das máquinas com o FOMA: dispositivos de Névoa podem apresentar uma quantidade reduzida de recursos físicos. Logo, é importante avaliar o *framework* FOMA medindo o consumo de CPU, memória RAM e

[11]https://mosquitto.org/
[12]https://hub.docker.com/

armazenamento em disco nas máquinas, tanto antes como depois da implantação de todas as ferramentas;

II. Tempo de entrega do serviço IoT por meio do K3s: as ferramentas de orquestração realizam inúmeras funções referentes ao provisionamento de serviços em contêiner. Assim, diferentes aspectos podem ser avaliados na orquestração do FOMA por meio do K3s. Procurou-se investigar um dos pontos em que o K3s pode impactar: a facilidade de entrega de serviços. Para isso, foi avaliado o tempo que o *cluster* leva para criar instâncias do Eclipse Mosquitto, escalonando o serviço de 2 a 14 réplicas;

III. Tempo da implantação automatizada do FOMA com Ansible: o FOMA possui uma implantação automatizada que realiza ações que podem demandar tempo e esforço relevantes do usuário. Por isso, foi analisado também o tempo de implantação das ferramentas da infraestrutura através dos *playbooks* desenvolvidos.

Para cada métrica, foi realizada a captura de trinta e três amostras, com o intuito de reduzir o impacto de aleatoriedades e calcular a média para cada valor. Na subseção a seguir, os resultados obtidos dos testes de desempenho aplicados ao *framework* são detalhados.

5.2. Resultados

I. Consumo de recurso físico das máquinas com o FOMA

A Tabela 1 mostra o consumo médio de CPU, de memória RAM e de armazenamento em disco, antes e depois da implantação do FOMA no ambiente. Analisando as medições dos três recursos, pode-se constatar que o Módulo Intermediário demanda mais recursos, se comparado aos outros módulos. Nota-se também que os consumos do Módulo Intermediário de CPU e memória se mostram relevantes em um computador de baixo poder computacional, o Raspberry com processador de quatro núcleos e memória de 1 GB. Contudo, considerando que o dispositivo desse módulo ficará dedicado ao controle do *cluster* K3s, conclui-se que o consumo do Módulo Intermediário é aceitável em uma infraestrutura com pouco recurso.

Tabela 1. Porcentagens médias do consumo de recursos.

Módulo destinado ao dispositivo	FOMA está presente?	CPU (%)	RAM (%)	Disco (%)
Módulo Intermediário	Não	0,29	6,00	8,00
	Sim	19,49	42,97	9,00
Módulo Alocador	Não	0,00	7,57	14,27
	Sim	4,83	14,09	17,27
Módulo Gerenciador e de Monitoramento	Não	2,30	21,84	35,43
	Sim	4,00	21,90	35,53

Em relação aos demais módulos, o consumo se mostrou baixo e satisfatório. Nos computadores dos módulos Gerenciador, Alocador e de Monitoramento, os acréscimos no uso de CPU e de memória não ultrapassaram o valor de 7%. Em todos os módulos, incluindo o Intermediário, o acréscimo no uso de disco não superou 4%, mesmo no Raspberry com 1 GB de RAM, 16 GB de disco e processador de quatro núcleos de CPU.

Diante desses resultados, conclui-se que o consumo de recursos físicos não foi elevado, considerando a quantidade de ferramentas instaladas e das funcionalidades atribuídas ao ambiente: orquestração, monitoramento e automatização. O baixo nível de consumo demonstra que o FOMA pode ser empregado em dispositivos de poucos recursos físicos. Nessa direção, vale destacar o baixo consumo do Módulo Alocador. Afinal, este é o módulo que deve ser implantado nos nós da Névoa, as máquinas que irão receber e executar os serviços da camada intermediária.

II. Tempo de entrega do serviço IoT através do K3s

A Figura 3 apresenta os resultados dos testes, nos quais foi medido o tempo gasto para escalonar de 2 até 14 réplicas da aplicação com o uso do *framework*. Pode-se notar que a entrega das réplicas ocorre em segundos, com uma variação entre 0,6 e 1,2 segundos. Isso demonstra a agilidade que o K3s pode proporcionar quando precisa-se provisionar várias réplicas de um serviço. Uma tarefa como essa feita manualmente, sem o uso de ferramenta de orquestração, pode levar bastante tempo e esforço do usuário para configurar e executar cada uma das réplicas.

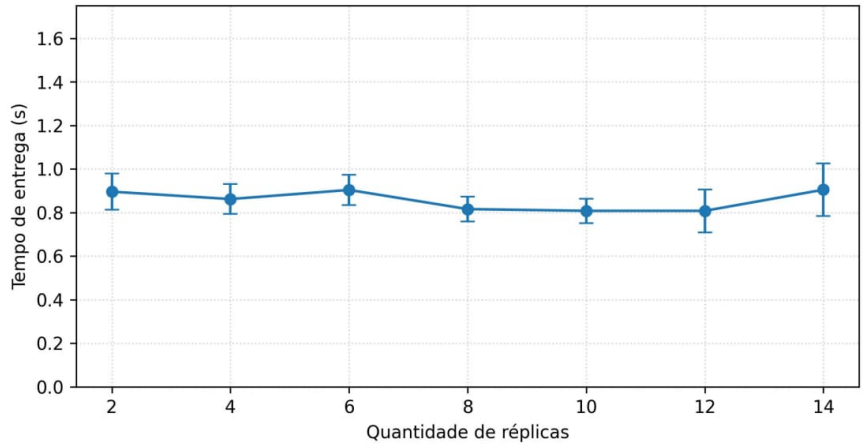

Figura 3. Tempo de entrega do serviço.

III. Tempo da implantação automatizada do FOMA com Ansible

Com a captura dos tempos da execução dos *playbooks* de implantação, foi obtido o tempo médio de 8,62 minutos, com o desvio padrão de 0,76 minutos. Ou seja, o tempo que o FOMA realizou toda a implantação das ferramentas de orquestração e monitoramento foi entre 7,87 e 9,38 minutos. Esse tempo obtido foi relativamente pequeno considerando as diversas tarefas realizadas.

A Fig. 4 ilustra o fluxo que um usuário precisa seguir para instalar e configurar, de forma manual, as ferramentas de (1) automatização, (2) orquestração e (3) monitoramento na infraestrutura. A Figura 5 apresenta o fluxo que o usuário precisa seguir para utilizar a implantação automatizada proposta. Comparando os dois fluxos, verifica-se que o número de tarefas realizadas pelo usuário é menor na implantação automatizada.

Esta comparação de fluxos e o resultado deste experimento nos permite concluir que, de fato, o FOMA oferece uma implantação facilitada, visto que o usuário não precisa

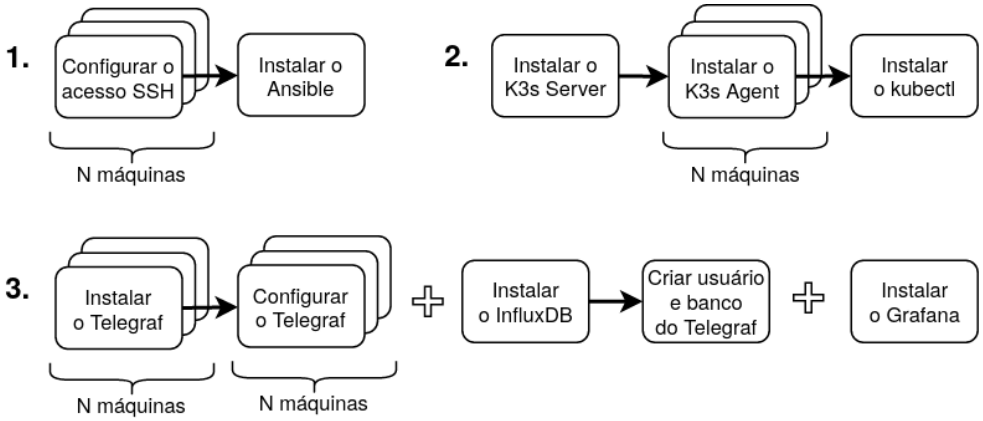

Figura 4. Fluxo de implantação sem a proposta.

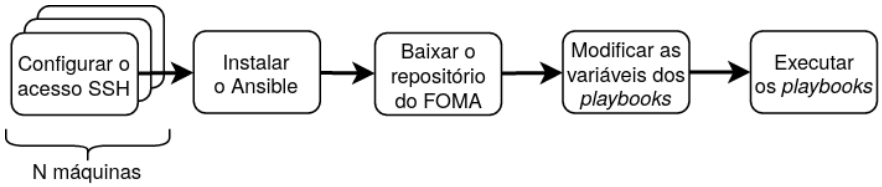

Figura 5. Fluxo de implantação com a proposta.

participar ativamente das tarefas realizadas pelo servidor de implantação, o que poupa bastante tempo e esforço do usuário.

6. Considerações finais

O gerenciamento de serviços e dispositivos de ambientes de névoa é uma tarefa custosa e complexa. Uma forma para solucionar este problema é usar ferramentas que auxiliem no gerenciamento, como orquestradores de contêineres e *softwares* de monitoramento. Entretanto, os nós da névoa possuem uma quantidade limitada de recursos físicos.

Neste contexto, este trabalho apresentou o FOMA, um *framework* com provisionamento e gerenciamento de serviços em contêiner, monitoramento de recursos e automatização de tarefas. No decorrer do trabalho, foi apresentado a arquitetura com seus módulos e a implementação deste *framework* utilizando ferramentas de código aberto voltadas para ambientes de IoT. Por fim, foi realizada a avaliação em ambiente real com três dispositivos, sendo dois destes de baixo custo e de capacidade física reduzida.

A avaliação foi direcionada a testes de desempenho, referentes ao consumo de recursos, ao tempo de entrega de serviços e ao tempo de implantação das ferramentas. No primeiro teste, foi verificado o consumo reduzido do *framework* nas máquinas, destacando que o módulo alocador apresentou acréscimos de consumo inferiores a 7%. Foi evidenciado que este módulo foi designado ao nó da Névoa utilizando o dispositivo de menor capacidade computacional do cenário.

No segundo teste, constatou-se o tempo inferior a 1,2 segundos para entrega de várias réplicas de um serviço por meio do orquestrador. Dessa forma, foi demostrado a eficiência do gerenciamento de serviços do *framework*. No último teste, foi avaliado o tempo da execução dos *playbooks* de implantação. Esse teste resultou na média de

8,6 minutos, um tempo relativamente baixo se for considerada a quantidade de tarefas realizadas durante a implantação da infraestrutura de Névoa.

Por fim, a avaliação nos permitiu identificar o FOMA como uma solução viável e interessante para o gerenciamento em ambientes de Névoa. Isso porque o *framework* entregou funcionalidades de gerenciamento, em troca de um consumo baixo de recursos e um tempo curto para implantação das ferramentas. Isso poderá facilitar o trabalho de usuários que procuram uma forma rápida e simples de implantar e monitorar serviços IoT que utilizam a Névoa.

Para trabalhos futuros, pretende-se adicionar mais nós de Névoa para testes de desempenho e aplicar testes de usabilidade do *framework*, coletando a opinião de usuários acerca das funcionalidades implementadas. Esse trabalho futuro tem o intuito de aprofundar o estudo das ferramentas em ambientes de Névoa e de implementar possíveis melhorias ao FOMA.

Referências

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

Coutinho, A., Carneiro, E. O., and Greve, F. G. P. (2016). Computação em névoa: Conceitos, aplicações e desafios. *Minicursos do XXXIV SBRC*, pages 266–315.

El Khalyly, B., Belangour, A., Erraissi, A., and Banane, M. (2020). Devops and microservices based internet of things meta-model. *International Journal*, 8(9).

Hoque, S., De Brito, M. S., Willner, A., Keil, O., and Magedanz, T. (2017). Towards container orchestration in fog computing infrastructures. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 294–299. IEEE.

Leskinen, A. (2020). Applicability of kubernetes to industrial iot edge computing system; kuberneteksen soveltuvuus teolliseen esineiden internet reunalaskentajärjestelmään. G2 pro gradu, diplomityö, Aalto University, School of Electrical Engineering.

Naha, R. K., Garg, S., Georgakopoulos, D., Jayaraman, P. P., Gao, L., Xiang, Y., and Ranjan, R. (2018). Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE access*, 6:47980–48009.

Santos, B. P., Silva, L., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Wöbker, C., Seitz, A., Mueller, H., and Bruegge, B. (2018). Fogernetes: Deployment and management of fog computing applications. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., and Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330.