Análise de Desempenho de Funções como Serviço em Nuvem Privada - DATAPREV

Marcelo A. da Cruz Motta[1], **Leonardo Reboucas de Carvalho**[1], **Aleteia P. F. de Araujo**[1]

[1]Departamento de Ciência da Computação
Universidade de Brasília (UNB) – Brasília, DF – Brazil

marcelo.motta@aluno.unb.br, leouesb@gmail.com, aleteia@unb.br

Abstract. *Function as a Service (FaaS) is a cloud service model that allows users to develop, run, and manage application functionality without worrying about the complexity of managing the infrastructure. Given the remarkable growth in its offer by public cloud providers and considering its use in private cloud environments, this paper presents a performance analysis of current open source technologies related to the Function as a Service (Faas) model, using the environment of the private cloud of the Empresa de Tecnologia e Informações da Previdência Social - DATAPREV.*

Resumo. *Função como serviço (FaaS) é um modelo de serviço em nuvem que permite aos usuários desenvolver, executar e gerenciar funcionalidades de aplicativos sem se preocupar com a complexidade da gestão da infraestrutura. Diante do notável crescimento na sua oferta pelos provedores públicos de nuvem e considerando sua utilização em ambientes de nuvem privada, este trabalho apresenta uma análise de desempenho das atuais tecnologias de código aberto relacionadas ao modelo de Function as a Service (Faas), utilizando o ambiente de nuvem privada da Empresa de Tecnologia e Informações da Previdência Social - DATAPREV.*

1. Introdução

Dentre as várias formas de oferta de serviços em nuvem atualmente destaca-se o *Function as a Service* (FaaS) [SPOIALA 2017], também conhecido como *Serverless*, no qual o cliente submete um trecho de código ao provedor, que irá garantir seu processamento independentemente do número de acionamentos. Em 2014 a AWS (Amazon Web Services) lançou o AWS Lambda [Amazon Web Services 2021], que é o modelo de serviço de nuvem denominado Function-as-a-Service [Schleier-Smith et al. 2021]. Desde então, diversos outros provedores de nuvem pública passaram a oferecer em seus catálogos serviços orientados a esse novo modelo, por exemplo o *Google Cloud Plattform* também oferece um serviço de FaaS, o *Google Cloud Function* [Google 2021], a Microsoft o *Azure Functions* [Microsoft 2021], outros provedores como Alibaba, Oracle e IBM também possuem suas opções de FaaS.

Diante do leque de novas possibilidades para esse modelo, o interesse sobre ele cresceu consideravelmente, tem sido tão representativo que existe a perspectiva de que, em pouco tempo, torne-se a forma de adoção de nuvem predominante dentre os modelos disponíveis atualmente [Schleier-Smith et al. 2021]. Isso ocorre porque além de simplificar consideravelmente a execução de software na nuvem, suprimindo a necessidade de

provisionamento, configuração e gestão sobre a infraestrutura necessária para o processamento de softwares escritos nas principais linguagens de programação, com o FaaS ainda é possível racionalizar o investimento em recursos computacionais. Nesse modelo de serviço a cobrança dos provedores é baseada no efetivo tempo de processamento que o software utiliza, e não no tempo de operação como ocorre em serviços de *Infrastructure as a Service* - IaaS tradicionais, nos quais a infraestrutura precisa permanecer alocada continuamente. Essas características, em conjunto com o comportamento auto escalável presente nesse modelo de serviço, tem feito com que diversos projetos migrem suas cargas de trabalho para ambientes orientados a FaaS. Diante desse cenário, diversos trabalhos [Carvalho and Araújo 2019], [García López et al. 2018], [Malawski et al. 2020], têm dedicado esforços no sentido de avaliar o desempenho desse tipo de serviço, seja com relação à utilização de recursos, tempo de processamento, dentre outras métricas. Entretanto, ainda existe na literatura um interesse maior em avaliar as plataformas de nuvem pública, mesmo reconhecendo-se que a implantação desse modelo de serviço em ambientes de nuvem privada possa agregar os mesmos benefícios evidenciados em nuvens públicas.

Com o objetivo de avaliar a eficiência das principais ferramentas disponíveis para implantar o FaaS em abiente de nuvem privada, este trabalho apresenta uma análise de desempenho entre duas ferramentas, de código aberto, disponíveis para implantação do modelo FaaS em ambiente de nuvem privada. A proposta é, após identificar a ferramenta mais performática, implantá-la na infraestrutura da DATAPREV[1].

2. Ferramentas de FaaS

O projeto Fn [Fn-Project 2021] é uma plataforma sem servidor nativa de contêiner em código aberto que pode ser executada em nuvens públicas e privadas. Ele suporta todas as linguagens de programação, é extensível e de alto desempenho. O OpenFaaS [OpenFaaS 2021] é um *framework* de código aberto que opera sobre o Kubernetes e que se prompõe a tornar mais fácil para os desenvolvedores implantarem funções orientadas a eventos e microsserviços. O Apache OpenWhisk [Apache OpenWhisk 2021] é uma plataforma sem servidor distribuída de código aberto, que executa funções em resposta a eventos, em qualquer escala. Ele gerencia a infraestrutura, os servidores e o dimensionamento usando contêineres Docker. Oferece suporte ao processamento de diversas linguagens de programação, podendo ser agendada dinamicamente e executada em resposta a eventos associados (via *triggers*) de fontes externas (Feeds) ou de solicitações HTTP. O Fission [Fission 2021] é um *framework* de código aberto que opera sobre o Kubernetes e é extensível a qualquer linguagem de programação. No Fission o núcleo é escrito em linguagem Go, e as partes específicas das linguagens são isoladas em algo chamado ambiente mais abaixo. Essa plataforma mantém um *pool* de contêineres "quentes", cada um contendo um pequeno carregador dinâmico.

3. Metodologia

Para realização dos testes, foram implantados ambientes equivalentes de Fn [Fn-Project 2021], OpenWhisk [Apache OpenWhisk 2021], Fission [Fission 2021] e OpenFaaS [OpenFaaS 2021] a fim de que uma ampla análise seja feita sobre a qualidade e eficiência dessas ferramentas em provedores privados de nuvem. Todavia, embora

[1]Empresa de Processamento de Dados da Previdência, empresa pública vinculada ao governo brasileiro

Fn e OpenWhisk tenham sido considerados na análise com alto grau de maturidade, os parâmetros de suas configurações iniciais limitaram consideravelmente o escopo do experimento. O OpenWhisk, por exemplo, define como limite máximo de requisições, dentro de 60s, apenas 100 requisições. Assim como OpenWhisk, o Fn não foi considerado devido a dificuldades tecnicas na sua equalização com as outras plataformas. Assim sendo, foram testadas as ferramentas OpenFaaS e Fission, que foram devidamente equalizadas e não implantam nenhuma limitação inicial.

Conforme arquitetura representada na Figura 1, foram executadas cargas de trabalho idênticas, exponencialmente crescentes, oriundas da ferramenta Jmeter. O propósito foi permitir uma análise comparativa e fatorial das métricas relacionadas a latência dessas plataformas sobre diferentes condições de processamento. Para isso, foram escritos *Scripts* utilizados para as funções *latency* e *matrix* do *benchmark* FaasDom [FASDOM 2021].

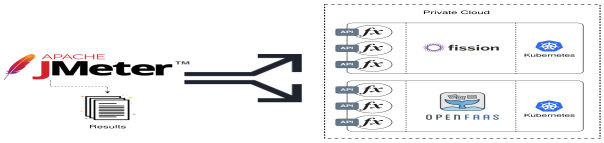
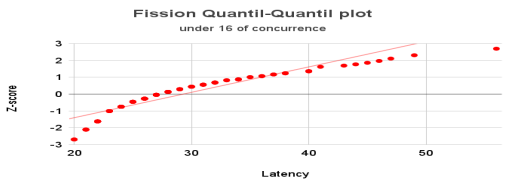


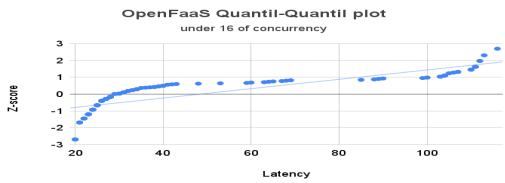
Figura 1. Arquitetura do experimento

4. Resultados

Os resultados mostraram que a plataforma Fission mantém a latência em níveis semelhantes, mesmo com diferentes níveis de concorrência. Todavia, o OpenFaaS mostrou variações conforme o nível de concorrência é alterado. Além disso, o planejamento fatorial mostrou a existência de um fator cujo efeito se sobrepõe aos demais. Para decidir a melhor métrica a ser adotada, dois gráficos quantil-quantil foram produzidos. Ambas as Figuras 2a e 2b se referem as execuções da função *latency* na plataforma Fission e OpenFaaS com concorrência 16 respectivamente.



(a) Fission.

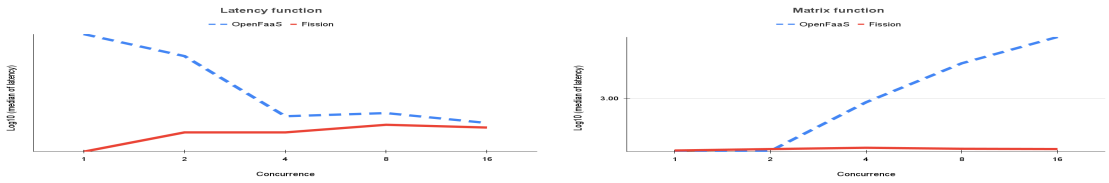


(b) OpenFaaS.

Figura 2. Resultados qqPlot.

Em ambas as figuras é possível perceber que as latências não seguem completamente uma distribuição normal, portanto, optou-se em adotar a mediana como métrica para representar as amostras de cada execução simultânea, uma vez que esta métrica representa melhor os dados com características de dispersão. Em ambos os casos, as medianas foram submetidas a uma transformação logarítmica para proporcionar uma melhor visualização das informações. A Figura 3b mostra a evolução da execução da função *matrix* sob cada nível de competição em ambas as plataformas. Isso possibilitou a

constatação de que sem concorrência, ambas as plataformas apresentam resultados equivalentes para latência. Porém, à medida que o nível de concorrência cresce, o Fission mantém a latência em um nível semelhante ao nível sem concorrência, todavia, o OpenFaaS faz uma escala ascendente. Isso mostra que a plataforma Fission foi capaz de processar a função *matrix* de forma mais eficiente que o OpenFaaS em níveis de concorrência, uma vez que seus tempos de latência eram menores que o OpenFaaS do primeiro nível e a distância entre a latência média do Fission e do OpenFaaS aumenta conforme cresce o nível de concorrência. A Figura 3a mostra a evolução da latência na execução da função *latency* em cada plataforma nos níveis de stress utilizados, demonstrando que ao executar sem concorrência, a plataforma OpenFaaS obteve um resultado de latência muito maior que o valor obtido pela plataforma Fission. Além disso, conforme o nível de concorrência cresce, o OpenFaaS diminui a distância para Fission, porém, ele permanece em um nível mais alto de latência em todos os níveis de concorrência, mostrando que o Fission foi mais eficiente que o OpenFaaS no processamento da função *latency* em todos os níveis de concorrência do experimento.



(a) Função Lantecy. **(b)** Função Matrix.

Figura 3. Resultados das Funções *latency* e *matrix*.

Assim sendo, conforme apresentado na Figura 4, os efeitos para cada fator são obtidos através da soma do cruzamento da tabela fatorial com a média calculada na soma dos quadrados totais de erros da amostragem. A partir dos efeitos foi possível calcular as variações com base na quantidade de repetições do experimento. Uma vez calculada a soma dos quadrados totais, foi possível então determinar as frações de cada efeito e obter uma visão percentual de sua influência nos resultados dos experimentos, onde o fator de função exerceu efeito de 26% sobre os resultados, enquanto os fatores de concorrência e provedores exerceram 12% cada. A composição dos fatores plataforma com função e plataforma com concorrência tiveram um efeito de 11% nos resultados cada uma, enquanto as composições de concorrência com função e plataforma com função tiveram um efeito de 12% nos resultados. O efeito do erro da amostra foi de 0,1%. Portanto, fica claro que o fator de função exerceu efeito preponderante nos resultados deste experimento.

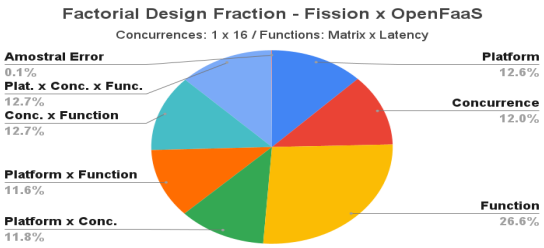


Figura 4. Análise fatorial da latencia.

5. Conclusão

Neste trabalho, foi realizado um experimento no ambiente de nuvem privada da DATAPREV, para avaliar as plataformas FaaS Fission e OpenFaaS sob diferentes condições de competição. Os resultados mostraram que a plataforma Fission apresentou melhor desempenho em relação à latência durante os testes. Por outro lado, o OpenFaaS experimentou degradação da latência linear em um dos testes. Em um planejamento fatorial aplicado aos resultados do experimento, ficou evidente que os resultados obtidos foram influenciados 26% pelo fator de função, enquanto os demais fatores e as relações entre os fatores influenciou cerca de 11% a 12% respectivamente. Em trabalhos futuros, outras funções podem ser incluídas neste experimento, pois este fator se mostrou relevante nos resultados. Além disso, pretende-se agregar outras soluções como OpenWhisk, Fn e KNative, estabelecendo uma configuração customizada em cada plataforma que ofereça igualdade de condições entre as ferramentas para permitir que sejam devidamente comparadas.

Referências

Amazon Web Services (2021). AWS lambda.

Apache OpenWhisk (2021). Open source serverless cloud platform.

Carvalho, L. and Araújo, A. P. F. (2019). Framework node2faas: Automatic nodejs application converter for function as a service. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, pages 271–278. INSTICC, SciTePress.

FASDOM (2021). The faasdom benchmark suite.

Fission (2021). Open source, kubernetes-native serverless framework.

Fn-Project (2021). Open source. container-native. serverless platform.

García López, P., Sánchez-Artigas, M., París, G., Barcelona Pons, D., Ruiz Ollobarren, A., and Arroyo Pinto, D. (2018). Comparison of faas orchestration systems. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 148–153.

Google (2021). Cloud functions. `https://cloud.google.com/functions/`. [Online; accessed 10-August-2021].

Malawski, M., Gajek, A., Zima, A., Balis, B., and Figiela, K. (2020). Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions. *Future Generation Computer Systems*, 110:502–514.

Microsoft (2021). Azure functions.

OpenFaaS (2021). Serverless functions, made simple.

Schleier-Smith, J., Sreekanti, V., Khandelwal, A., Carreira, J., Yadwadkar, N. J., Popa, R. A., Gonzalez, J. E., Stoica, I., and Patterson, D. A. (2021). What serverless computing is and should become: The next phase of cloud computing. *Commun. ACM*, 64(5):76–84.

SPOIALA, C. (2017). Pros and cons of serverless computing.