Quer Pagar Quanto? Uma Comparação de Custos de Provedores de Nuvem FaaS

Diogo Bortolini[1], **Rafael R. Obelheiro**[1]

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

{diogo.bortolini,rafael.obelheiro}@udesc.br

Abstract. *Cloud application developers have been increasingly adopting the function-as-a-service (FaaS) service model, where infrastructure provisioning and management are left to providers, freeing developers from such concerns and enabling rapid elasticity. In the FaaS model, billing is based on the number of function invocations and on function execution time, where a function is a self-contained code module, instantiated on-demand. Since billing is based on effective usage of resources, this model points towards cost reduction for customers; at the same time, it reduces cost predictability, especially because costs might be affected by fluctuations in execution time. This work investigates how function performance and its variability influence costs from the perspective of cloud customers, considering different FaaS providers (AWS Lambda, Azure Functions, Google Cloud Functions, and IBM Cloud Functions).*

Resumo. *Cada vez mais, desenvolvedores de aplicações para nuvem vêm adotando o modelo de serviço Function-as-a-service (FaaS), no qual o provisionamento e o gerenciamento da infraestrutura ficam a cargo do provedor, liberando o desenvolvedor de tais preocupações e proporcionando elasticidade rápida. A tarifação no modelo FaaS é baseada no número de invocações e no tempo de execução das funções (módulos estanques de código, instanciados sob demanda). Ao vincular a cobrança à efetiva utilização dos recursos, essa modalidade de tarifação aponta para uma redução dos custos para o cliente; ao mesmo tempo, ela reduz a previsibilidade dos custos, especialmente porque estes podem ser afetados por oscilações no tempo de execução. Este trabalho investiga como o desempenho das funções e sua variabilidade influenciam nos custos sob a ótica do cliente de nuvem, considerando diferentes provedores FaaS (AWS Lambda, Azure Functions, Google Cloud Functions e IBM Cloud Functions).*

1. Introdução

Uma tendência emergente em nuvens computacionais tem sido a adoção do modelo de serviço FaaS (*Function-as-a-Service*), em que aplicações são construídas como conjuntos de funções que executam em instâncias[1] ativadas sob demanda, em resposta a eventos [Roberts 2018]. Essas instâncias são *stateless* e temporárias, podendo permanecer ativas durante a execução de uma única requisição, e gerenciadas inteiramente pelo provedor,

[1]Seguindo [Wang et al. 2018], o termo *instância* será usado para denotar o contêiner ou máquina virtual em que uma função é executada.

que fica responsável por garantir sua escalabilidade e disponibilidade. O cliente é cobrado por invocação, sem pagar por recursos ociosos. Com isso, o modelo FaaS pode propiciar uma economia significativa em relação aos modelos mais tradicionais de infraestrutura e plataforma como serviço (IaaS e PaaS), ao mesmo tempo em que libera o desenvolvedor de preocupações com a gerência da infraestrutura [Adzic and Chatley 2017].

De maneira mais específica, a tarifação em FaaS é baseada no número de invocações e no tempo de execução das funções. Isso faz com que o custo em FaaS seja mais variável do que o custo nos modelos IaaS/PaaS, em que a tarifação depende da capacidade alocada [Eivy 2017]. Além disso, vários estudos [Leitner et al. 2016, Eivy 2017, Jonas et al. 2017] apontam que a visibilidade dos custos é baixa, ou seja, o cliente de nuvem tem dificuldade em antecipar quanto ele irá pagar pelos serviços.

Os custos do modelo FaaS para o cliente de nuvem são discutidos em diversos trabalhos na literatura [Leitner et al. 2016, Adzic and Chatley 2017, Villamizar et al. 2017, Eivy 2017]. Esses trabalhos, porém, apresentam uma visão limitada a um único provedor, e/ou desconsideram como o custo pode ser impactado por variações no desempenho das funções, as quais muitas vezes fogem do controle do cliente mas ocorrem na prática [Jonas et al. 2017, Roberts 2018, Lloyd et al. 2018, Wang et al. 2018].

Este trabalho visa então suprir esta lacuna, explorando de forma mais abrangente de que modo as características das funções (particularmente as variações de desempenho) influenciam no custo para o cliente. São considerados quatro provedores FaaS: AWS Lambda, Azure Functions, Google Cloud Functions e IBM Cloud Functions. Os resultados obtidos demonstram que o menor custo é encontrado em diferentes provedores de acordo com as características das funções, e que, diferente do que seria esperado, a variabilidade do tempo de execução não induz variações significativas no custo.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 descreve os provedores FaaS considerados e como eles efetuam a tarifação. A Seção 4 compara os custos dos provedores em diferentes cenários, e a Seção 5 conclui o artigo.

2. Trabalhos Relacionados

[Eivy 2017] apresenta uma discussão abrangente sobre os custos envolvidos na adoção do modelo FaaS. Em particular, ele destaca que: *(i)* a tarifação em FaaS é de difícil compreensão; *(ii)* variações no tempo de execução de funções podem representar variação nos custos; e *(iii)* como funções são *stateless*, elas representam apenas parte do custo de uma aplicação, que provavelmente terá custos adicionais com armazenamento, rede e outros serviços do provedor. O enfoque do trabalho é mais conceitual que quantitativo, e custos concretos são discutidos apenas em um cenário restrito, com um único provedor (AWS Lambda) e mantendo constantes o tempo de execução, a memória dimensionada e a taxa de requisições de uma função.

[Adzic and Chatley 2017] também discute o custo associado a FaaS. O trabalho aponta que deixar a escalabilidade e a disponibilidade das aplicações a cargo do provedor tende a gerar uma economia significativa, pois evita que o cliente tenha de gastar com o provisionamento de capacidade de reserva e instâncias de *backup*, recursos que muitas vezes permanecem ociosos. O artigo apresenta como estudos de caso duas aplicações que

migraram para o modelo FaaS: a primeira, que migrou de PaaS, teve redução de custo da ordem de 66%, enquanto a segunda, que migrou de IaaS, teve custos reduzidos em mais de 95%. O trabalho também apresenta uma comparação limitada de custos, considerando um único provedor (AWS Lambda) e mantendo constantes o tempo de execução e a taxa de requisições de uma função.

[Villamizar et al. 2017] faz um estudo experimental comparando o desempenho e o custo de uma mesma aplicação com diferentes arquiteturas: monolítica, microsserviços e baseada em funções. Embora o estudo mostre uma redução de custo na versão FaaS, ele considera um único provedor (AWS Lambda) e uma taxa fixa de requisições.

[Leitner et al. 2016] apresenta um modelo de custo abrangente para aplicações baseadas em microsserviços implementados tanto com IaaS/PaaS quanto com FaaS. O modelo é difícil de parametrizar, e para FaaS considera que o custo por requisição é fixo, ignorando a variabilidade de desempenho.

[Jonas et al. 2017] descreve PyWren, uma plataforma para análise de dados massivos implementada em AWS Lambda. O artigo conclui que liberar o desenvolvedor de gerenciar a infraestrutura oferece ganhos significativos de produtividade, e que o desempenho das funções, embora aceitável, apresenta uma variabilidade perceptível.

Alguns trabalhos fazem estudos de medições comparando diferentes provedores FaaS. [Lloyd et al. 2018] considera AWS Lambda e Azure Functions, ao passo que [Wang et al. 2018] considera esses dois e mais o Google Cloud Functions. A conclusão mais relevante desses estudos para o contexto do presente trabalho é que limitações no isolamento de recursos entre funções/*tenants* e o fato de funções poderem executar em máquinas distintas a cada invocação (dependendo do escalonamento) podem levar a variações significativas no desempenho de uma mesma função. No entanto, nenhum desses trabalhos discute custo diretamente.

O presente trabalho pode ser comparado mais diretamente aos estudos que analisam os custos no modelo FaaS [Leitner et al. 2016, Adzic and Chatley 2017, Eivy 2017, Villamizar et al. 2017], tendo como principal diferença uma exploração mais ampla dos fatores que influenciam os custos, especialmente a variabilidade no tempo de execução das funções. Referências que apresentam medições efetudas em provedores FaaS, como [Jonas et al. 2017], [Lloyd et al. 2018] e [Wang et al. 2018], indicam que essa variabilidade de desempenho é uma realidade concreta, e ajudam a evidenciar a relevância deste trabalho.

3. Provedores FaaS

Atualmente, as plataformas FaaS oferecidas pelos principais provedores de nuvem são AWS Lambda[2], Azure Functions[3], Google Cloud Functions[4], e IBM Cloud Functions[5]. Esta seção apresenta as características mais relevantes de cada plataforma, incluindo o esquema de tarifação. A Tabela 1 resume essas características.

O cliente de um provedor FaaS desenvolve suas funções como módulos de código

[2]`https://aws.amazon.com/lambda/`
[3]`https://azure.microsoft.com/en-us/services/functions/`
[4]`https://cloud.google.com/functions/`
[5]`https://www.ibm.com/cloud/functions`

	AWS	Azure	Google	IBM
Memória	64k MB ($k = 2,3,\ldots,47$)	128k MB ($k = 1,2,\ldots,12$)	128k MB ($k = 1,2,4,8,16$)	128k MB ($k = 1,2,4$)
CPU	proporcional à memória	não informado	proporcional à memória	não informado
Rede (transferência / mês)	1 GB	5 GB	5 GB	não informado
Disco local	512 MB	500 MB	500 MB	não informado
Linguagem	Node.js (JavaScript), Python, Java, C# (.NET Core) e Go	C#, F#, Node.js, Java, Python e PHP	Node.js e Python	JavaScript, Swift, Python, PHP, Java e outras
Tamanho máximo da função	50 MB (compactado) e 250 MB (descompactado)	100 MB (compactado) e 500 MB (descompactado)	100 MB (compactado) e 500 MB (descompactado)	48 MB
Tempo máximo de execução	5 minutos	10 minutos	9 minutos	10 minutos
Granularidade de tarifação	100 ms	1 ms (mín. 100 ms)	100 ms	100 ms
Custo mensal (USD)	\$0,20/1M de execuções, \$0,00001667/GB-s	\$0,20/1M de execuções, \$0,000016/GB-s	\$0,40/1M de execuções, \$0,0000025/GB-s, \$0,0000100/GHz-s	\$0,000017/GB-s
Faixa gratuita (mês)	400.000 GB-s, 1 M de execuções	400.000 GB-s, 1 M de execuções	400.000 GB-s, 2 M de execuções	400.000 GB-s

Tabela 1. Comparação entre as configurações de FaaS e cobrança dos provedores.

que implementam uma determinada lógica de aplicação. Essas funções são empacotadas e carregadas em um serviço de armazenamento na nuvem. Quando uma função é invocada, seu código é carregado e executado em uma ou mais instâncias (tipicamente contêineres), de acordo com o volume de requisições. Depois que a função termina, sua instância torna-se ociosa. Para evitar a latência de ativação de novas instâncias, instâncias ociosas podem ser reaproveitadas para atender a novas requisições, mas não há garantias de que isso ocorra. Para armazenamento temporário, cada instância tem acesso a um espaço no disco local, que será removido quando a instância for desativada; para armazenamento persistente, a opção mais comum é usar um dos vários serviços oferecidos pelo provedor, que são cobrados à parte. Cada função tem um tempo máximo de execução, que varia de 5 min (AWS) a 10 min (Azure e IBM).

Em todas as plataformas consideradas, o cliente precisa especificar a quantidade de memória a ser alocada para uma instância. A capacidade mínima em todas as plataformas é de 128 MB, e a máxima varia entre 512 MB (IBM) e 3008 MB (AWS). A granularidade de alocação também é variável, conforme pode ser visto na Tabela 1. A quantidade de CPU alocada é proporcional à memória alocada para AWS e Google, e não informada para Azure e IBM.

De modo geral, a tarifação dos provedores FaaS tem dois componentes. O primeiro é o número de invocações de funções, considerando todas as funções pertencentes ao mesmo usuário. Essa é uma métrica fácil de compreender, ainda que seja dependente da arquitetura da aplicação: uma aplicação subdividida em um número grande de funções com granularidade fina induz um número maior de invocações do que uma aplicação com um número menor de funções com granularidade mais grossa [Eivy 2017]. Dentre os provedores considerados, o único que não cobra pelo número de invocações é a IBM.

O segundo componente de tarifação é o consumo de recursos. De forma geral, esse consumo é mensurado em GB-s, que é o produto da memória alocada para a função, expressa em GB, pelo seu tempo de execução, expresso em segundos. Por exemplo, uma função de 512 MB de memória que execute durante 200 ms tem um consumo de $0{,}5\,\text{GB} \times 0{,}2\,\text{s} = 0{,}1\,\text{GB-s}$. Um ponto importante é que as granularidades de alocação de

memória e de tempo de execução variam entre os provedores. Por exemplo, uma função que necessite de 140 MB de memória irá alocar 192 MB na AWS, mas 256 MB na Azure, Google e IBM. Com relação ao tempo de execução, este é arredondado para cima em múltiplos de 100 ms; portanto, uma função que leve 99 ms para executar será cobrada por 100 ms, e uma função que leve 101 ms será cobrada por 200 ms.

Existem outras diferenças entre os provedores na tarifação por consumo de recursos. Na Azure, o cliente é cobrado pela quantidade máxima de memória efetivamente usada (e não pela memória alocada), e tempos de execução superiores a 100 ms são medidos com granularidade de 1 ms (tempos inferiores são cobrados como 100 ms). No Google, o consumo de recursos tem um componente adicional, o provisionamento de CPU, que é medido em GHz-s e dado pelo produto entre o tempo de execução e a alocação de ciclos de CPU, conforme mostrado na Tabela 2.

memória (MB)	128	256	512	1024	2048
CPU (GHz)	0,2	0,4	0,8	1,4	2,4

Tabela 2. Provisionamento de CPU no Google Cloud Functions

Os provedores proporcionam faixas gratuitas de uso para o primeiro milhão de execuções e 400.000 GB-s a cada mês. Para IBM a gratuidade é de 400.000 GB-s/mês, uma vez que não existe cobrança pelo número de execuções. Para Google, o número de invocações gratuitas sobe para dois milhões/mês. Os provedores Azure e IBM oferecem ainda cobrança por assinatura mensal conforme os recursos contratados. Para Azure essa modalidade de pagamento é denominada Plano de Serviço de Aplicativo, enquanto que para IBM é intitulado plano Bluemix. Essas modalidades estão fora do escopo do trabalho e não serão detalhadas.

Uma análise dos esquemas tarifários dos provedores FaaS reafirma as premissas deste trabalho. Os vários componentes de custo (volume de requisições, consumo de recursos), as faixas de gratuidade e as diferenças entre os provedores dificultam a compreensão e o planejamento dos custos por parte do cliente. Além disso, os custos associados a consumo de recursos são influenciados diretamente pela variabilidade no tempo de execução das funções e pela granularidade de medida desse tempo.

4. Comparação de Custo

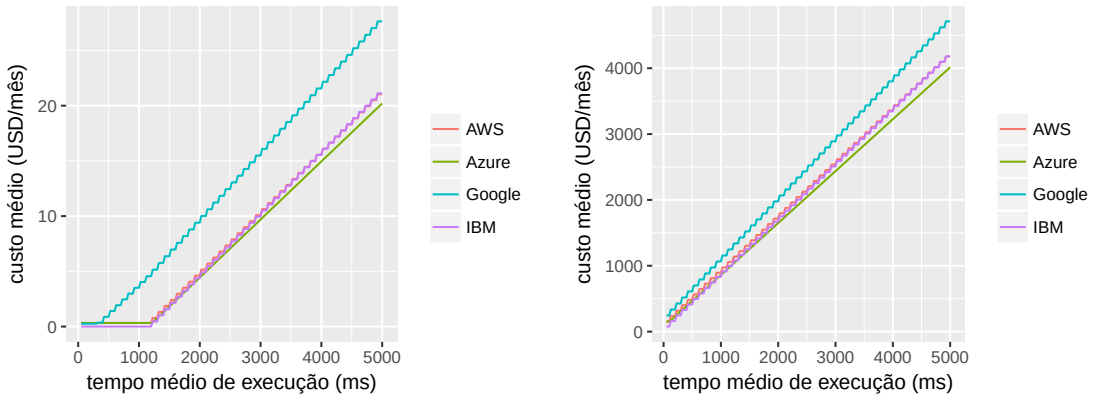
Para melhor entender os custos dos provedores FaaS e permitir sua comparação, foram avaliados os esquemas de tarifação apresentados na Seção 3 em diferentes cenários. A avaliação considerou apenas os custos diretamente relacionados à execução de funções, de acordo com o número de invocações, o tempo de execução e a memória usada (no caso do Google, a alocação de CPU está vinculada à alocação de memória). Sendo assim, ficaram de fora do escopo de análise custos adicionais que podem ocorrer em instâncias com elevado tráfego de rede e/ou utilização de outros recursos e serviços disponibilizados pelo provedor, como armazenamento persistente de dados.

Inicialmente analisa-se como o custo varia de acordo com o tempo de execução para uma dada taxa de requisições (λ), que se traduz em um número fixo de requisições ao longo de um mês. Essa taxa fixa pode ser interpretada como a taxa média de requisições ao longo de um mês (e não necessariamente como uma taxa constante). A Figura 1

mostra o comportamento do custo para uma função com 128 MB de memória quando o tempo de execução varia entre 50 e 5000 ms (em incrementos de 25 ms) em dois cenários, $\lambda = 1$ req/s (Fig. 1(a)) e $\lambda = 150$ req/s (Fig. 1(b)). Em ambos os cenários, as curvas para AWS, Google e IBM exibem um comportamento serrilhado, devido ao arredondamento do tempo de execução para múltiplos de 100 ms. Azure não exibe esse comportamento devido à granularidade mais fina de medida do tempo de execução (1 ms).

No cenário da Fig. 1(a), a taxa de requisições é baixa (1 req/s), e é possível observar o efeito das faixas de gratuidade: o custo fica próximo a zero até um determinado tempo de execução. A cobrança por recursos alocados começa a ocorrer primeiro no Google, que também torna-se o provedor mais caro, com a diferença aumentando com o tempo de execução; em contrapartida, é o segundo provedor mais barato para tempos de execução de até 300 ms. Até 1200 ms, a IBM apresenta o menor custo entre os provedores, uma vez que não efetua cobrança pelo número de invocações das funções, somente pelos recursos alocados. Entre 1225 e 1600 ms, o menor custo fica oscilando entre IBM e Azure, que passa a ser o provedor mais barato a partir de 1625 ms. O custo da AWS se aproxima ao da Azure enquanto a IBM tem o custo mínimo, e depois passa a acompanhar o custo da IBM.

O segundo cenário (Fig. 1(b)) tem uma taxa de requisições mais alta, de 150 req/s: essa taxa é a mesma usada em [Eivy 2017], e foi adotada para facilitar comparações. Este cenário tem uma tendência semelhante à do anterior: Google é o provedor com o maior custo e IBM é o mais barato para tempos de execução de função baixos, mas torna-se mais custoso que Azure e posteriormente AWS com tempos de execução maiores. A Azure aparece como o provedor mais barato para tempos de execução a partir de 1700 ms.



(a) $\lambda = 1$ req/s (b) $\lambda = 150$ req/s

Figura 1. Custo médio mensal × tempo médio de execução

É importante ressaltar que o comportamento relativo demonstrado nos gráficos da Figura 1 repete-se para outros cenários em termos de taxa de requisições e/ou tamanho de memória alocada, mudando apenas de magnitude. O próprio tempo de execução foi variado até 300 s (tempo limite da AWS, que é o menor dentre os provedores), mas, como o comportamento segue a tendência observada, esses dados foram omitidos da visualização para dar maior clareza.

A segunda análise efetuada verifica como a variabilidade do tempo de execução

afeta os custos. Como não existem dados disponíveis sobre essa variabilidade, avaliou-se o que ocorre quando o tempo de execução segue uma determinada distribuição de probabilidade. Para um número mensal de requisições n, correspondente a uma dada taxa de requisições λ, foram geradas n variáveis aleatórias com a distribuição presumida, cada uma representando um tempo de execução de uma função. O custo para esse conjunto de tempos de execução foi então calculado. Esse processo foi repetido 10 vezes, gerando 10 valores distintos de custo, e a variação do custo foi determinada pela diferença porcentual entre o maior e o menor custo relativa ao custo médio (Eq. 1).

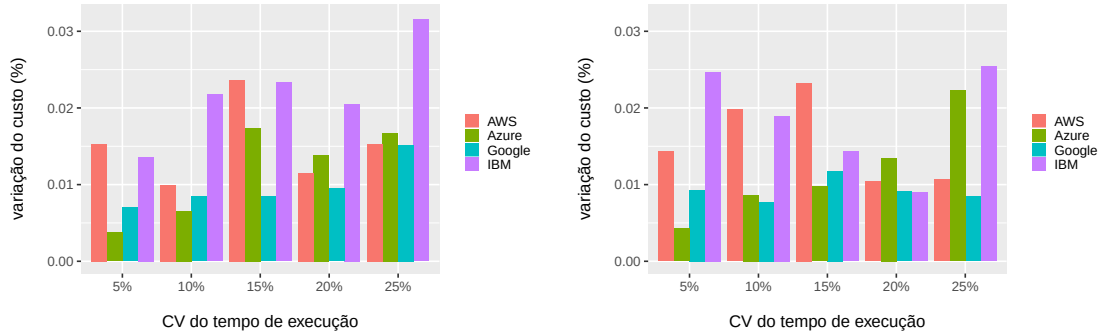
$$\text{variação do custo} = \frac{C_{max} - C_{min}}{C_{med}} \times 100 \tag{1}$$

Foram consideradas duas distribuições de probabilidade, normal e uniforme, e coeficientes de variação (CV) de 5, 10, 15, 20 e 25%. Adotou-se um tempo médio de execução de $\bar{t} = 200$ ms e $\lambda = 10$ req/s, o que corresponde a $n = 26.280.000$ requisições mensais. Para a distribuição normal, o desvio padrão foi estabelecido de acordo com o coeficiente de variação, usando a expressão $s = \text{CV} \cdot \bar{t}$; como o tempo mínimo de tarifação é de 100 ms, tempos próximos a zero (que são impossíveis na prática, mas podem aparecer na geração de números aleatórios) são tarifados como 100 ms, não gerando distorções no cálculo do custo. Para a distribuição uniforme, o intervalo de tempos de execução é dado por $[t_{min}, t_{max}] = \bar{t} \pm \sqrt{3} \cdot \text{CV} \cdot \bar{t}$ (a amplitude do intervalo pode ser derivada da expressão para o desvio padrão de uma distribuição uniforme $U(a,b)$, que é dado por $s = (b-a)/\sqrt{12}$ [Jain 1991, p. 498]). A geração de números aleatórios foi feita usando as funções `rnorm()` e `runif()` do pacote estatístico R [R Core Team 2018].

A Figura 2 apresenta a variação do custo em função da variabilidade do tempo de execução para cada um dos provedores estudados. A Fig. 2(a) mostra o que acontece quando os tempos de execução seguem uma distribuição normal, e a Fig. 2(b) traz a variação do custo quando os tempos de execução seguem uma distribuição uniforme. Observa-se que a variação do custo é bastante pequena: as maiores variações encontradas foram de 0,032% para distribuição normal e 0,025% para uniforme, ambas para IBM com CV de 25%. As diferenças na variação de custo entre os provedores são insignificantes, e a variação de custo não está fortemente correlacionada com o CV: o coeficiente de correlação de Pearson entre CV e variação de custo para os dados foi de $r = 0{,}453$ para a distribuição normal e de $r = 0{,}087$ para a distribuição uniforme. Isso permite concluir que a variabilidade do tempo de execução praticamente não influencia no custo mensal de uma função em um provedor FaaS, ao menos nas condições avaliadas neste artigo (tempo de execução com distribuição normal ou uniforme, e coeficiente de variação de até 25%).

Discussão dos resultados

Os resultados apresentados acima mostram que, dependendo da carga de requisições e das características das funções, o menor custo pode ser oferecido por diferentes provedores. Analisada de forma isolada, essa constatação pode sugerir uma estratégia de migração de provedores como forma de minimizar o custo. No entanto, a dependência de outros serviços do provedor (para armazenamento persistente, por exemplo) e a heterogeneidade de suporte a linguagens de programação podem dificultar essa migração. O



(a) distribuição normal (b) distribuição uniforme

Figura 2. Variação do custo mensal em função da variabilidade do tempo de execução

Serverless Framework[6] é um *framework* que visa permitir a portabilidade entre provedores FaaS, e pode minimizar os problemas de migração.

Uma característica que chama a atenção de desenvolvedores para o modelo FaaS são as faixas de gratuidade oferecidas pelos provedores. O presente estudo mostra que essas faixas podem manter o custo próximo a zero até determinados limiares de uso. Em contrapartida, como o custo é diretamente proporcional à utilização das funções, é preciso atentar para a possibilidade de receber altas cargas de requisições, que levarão a despesas significativas. Regular o acesso a funções, seja por *rate limiting* ou mesmo evitando que elas possam ser invocadas diretamente por usuários finais (ou por ações desses usuários), pode ser uma forma de manter o custo sob controle.

Uma outra questão analisada é de que forma os custos podem ser afetados pela variação no tempo de execução das funções. Os resultados apresentados acima evidenciam que essa variabilidade levam a uma variação inexpressiva nos custos para o cliente, ao menos nas condições consideradas em nosso estudo.

Embora a análise apresentada nesta seção tenha permitido quantificar o efeito de diferentes fatores (como taxa de requisições e tempo de execução) sobre os custos em provedores FaaS, ela foi baseada em valores hipotéticos para esses fatores. Estudos de medições em provedores FaaS e com seus clientes permitiriam identificar com mais clareza quais são os fatores mais críticos e embasar melhor a definição de níveis apropriados para esses fatores, e são sugeridos como trabalhos futuros.

5. Conclusão

A adoção do modelo de serviços FaaS por parte de desenvolvedores de aplicações para nuvens computacionais tem sido motivada por dois fatores preponderantes, economia de custos e redução das preocupações com gerenciamento e provisionamento da infraestrutura. Este trabalho apresentou um estudo sobre como os custos são influenciados por características como o tempo de execução e a taxa de invocação das funções, e comparou os custos de quatro provedores. Observou-se que os variados esquemas de tarifação adotados pelos provedores fazem com que a escolha do provedor de menor custo seja condicionada pelas características das funções. Constatou-se ainda que a variabilidade do

[6] `https://serverless.com/`

tempo de execução produz variações muito pequenas no custo para um mesmo volume de requisições, ao menos considerando as distribuições normal e uniforme.

Como continuidade deste trabalho, pretende-se realizar estudos de medições em provedores FaaS para melhor quantificar a variabilidade de desempenho das funções, tanto em um mesmo provedor quanto entre provedores, e incorporar à análise os planos mensais de assinatura oferecidos por provedores como Azure e IBM.

Agradecimentos

Os autores agradecem o apoio da Universidade do Estado de Santa Catarina (UDESC) e da Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC).

Referências

Adzic, G. and Chatley, R. (2017). Serverless computing: Economic and architectural impact. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, pages 884–889, New York, NY, USA. ACM.

Eivy, A. (2017). Be wary of the economics of "serverless" cloud computing. *IEEE Cloud Computing*, 4(2):6–12.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Wiley.

Jonas, E., Pu, Q., Venkataraman, S., Stoica, I., and Recht, B. (2017). Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, pages 445–451, New York, NY, USA. ACM.

Leitner, P., Cito, J., and Stöckli, E. (2016). Modelling and managing deployment costs of microservice-based cloud applications. In *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pages 165–174.

Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L., and Pallickara, S. (2018). Serverless computing: An investigation of factors influencing microservice performance. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pages 159–169. IEEE.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roberts, M. (2018). Serverless architectures. Disponível em `https://martinfowler.com/articles/serverless.html`.

Villamizar, M., Garcés, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M., Casallas, R., Gil, S., Valencia, C., Zambrano, A., and Lang, M. (2017). Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures. *Service Oriented Computing and Applications*, 11(2):233–247.

Wang, L., Li, M., Zhang, Y., Ristenpart, T., and Swift, M. (2018). Peeking behind the curtains of serverless platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 133–146, Boston, MA. USENIX Association.