Computação Serverless e Gerenciamento de Dados

Flávio R. C. Sousa

[1] Mestrado e Doutorado em Ciência da Computação (MDCC)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

flaviosousa@ufc.br

Abstract. *Serverless computing is a technology trend to provide transparent elasticity and prices in milliseconds. However, this computing model requires major technological changes, especially in data management, because in the serverless environment all functions have maximum duration, fixed amount of memory and no storage persistent location. This article aims to discuss serverless computing and data management, highlighting opportunities and challenges of research in this context.*

Resumo. *A computação serverless é uma tendência de tecnologia para fornecer elasticidade transparente e preços em milissegundos. No entanto, esse modelo de computação requer grandes mudanças tecnológicas, especialmente no gerenciamento de dados, pois no ambiente serverless todas as funções têm duração máxima, quantidade fixa de memória e ausência de armazenamento local persistente. Este artigo tem como objetivo discutir sobre a computação serverless e o gerenciamento de dados, destacando oportunidades e desafios de pesquisa neste contexto.*

1. Introdução

A computação *serverless* é uma tendência recente da tecnologia destinada a fornecer elasticidade transparente e preços em milissegundos [Jonas et al. 2019]. É uma evolução em relação à computação em nuvem, na qual o desenvolvedor foca apenas em funções deixando toda a responsabilidade de gestão dos recursos para o provedor. Seu princípio de funcionamento consiste em escalonar recursos de computação para executar funções *stateless* fornecidas por programadores e acionadas mediante eventos. O escalonamento e gestão dos recursos são de responsabilidade do provedor do serviço e o preço é calculado com base na utilização durante o tempo de execução das funções em milissegundos, não havendo cobrança por recursos ociosos. Aplicações *web*, *mobile*, *IoT*, *Big Data* e *chatbots* são exemplos de aplicações que se beneficiam das características do ambiente *serverless*.

Para atingir esse objetivo, os provedores de serviços impõem um modelo computacional onde cada função tem uma duração máxima, uma quantidade fixa de memória e ausência de armazenamento local persistente [Sreekanti et al. 2020]. As funções são executadas em contêineres efêmeros relacionados apenas com a sessão em execução sendo encerradas assim que atingem o tempo máximo, inviabilizando o uso de processos de longa duração. Por exemplo, na *Amazon Lambda* a capacidade de memória RAM das funções é definida pelo usuário e varia entre 128MB e 3GB em incrementos de 64MB, o processador é utilizado utilizado é proporcional a memória limitado a 1.7 núcleos e o espaço em disco para tarefas temporárias é de 500MB.

As plataformas de *Function as a Service* (FaaS) oferecem poder computacional capaz de escalonar para centenas ou até milhares de funções em segundos ou minutos [Schleier-Smith 2019]. Para tanto, estas plataformas implementam a separação entre computação e armazenamento, um princípio de design arquitetural cada vez mais popular no ambiente de computação em nuvem. Por outro lado, as instâncias de execução das funções são *stateless*, isto é, sem persistência de dados, exigindo o uso de serviços de armazenamento externo para a troca de estado em aplicações *stateful* [Klimovic et al. 2018a]. Além disso, existe um *trade-off* entre a latência e o custo que deve ser considerado no uso das plataformas *serverless*.

Embora os provedores *serverless* ofereçam serviços de gerenciamento de dados, existe uma enorme complexidade envolvida na construção de serviços de dados elásticos considerando diferentes cargas de trabalho e a possibilidade de terceiros implementarem o gerenciamento de dados diretamente na parte superior de uma plataforma de computação *serverless*. Desafios tais como armazenamento, processamento de dados e elasticidade são importantes para a computação *serverless* e acredita-se que futuras aplicações centradas em dados irão alavancar o uso de dados nestas plataformas [Jonas et al. 2019]. Este artigo tem como objetivo discutir o gerenciamento de dados no ambiente de computação *serverless*, destacando desafios e oportunidades nesse contexto.

2. Desafios e Oportunidades

A computação *serverless* é um paradigma cada vez mais popular e tem atraído esforços da indústria e da academia [Abadi et al. 2019]. A concepção de serviços de dados baseados em eventos terá impactos significativos na concepção de mecanismos de armazenamento, processamento e análise de dados. A seguir destacamos alguns desafios e oportunidades de pesquisa do gerenciamento de dados no ambiente *serverless*.

2.1. Armazenamento

O armazenamento é um ponto crítico no ambiente *serverless*, visto que sem uma abordagem de armazenamento eficiente e com desempenho satisfatório, é inviável construir aplicações de propósito geral que dependem de persistência [Klimovic et al. 2018a]. A natureza e as limitações das funções, a arquitetura das plataformas *data-shipping*, gargalos de I/O e a inexistência de serviços de armazenamento adequados restringem o alcance da computação *serverless* a um número limitado de casos de uso. Embora existam opções para a persistência de dados, estas não contemplam os requisitos para uso juntamente com *serverless*, tais como a capacidade de armazenamento efêmero com custo e desempenho satisfatórios, provisionamento transparente e acessível às funções *serverless* [Jonas et al. 2019].

Como as funções *serverless* apresentam limitações de memória e disco local, torna-se um desafio executar sistemas tais como o *Tensorflow* ou *Spark* neste ambiente. Os sistemas de armazenamento existentes não foram projetados para atender às demandas de aplições *serverless* em termos de elasticidade, desempenho e custo. Portanto, é crucial implementar um novo armazenamento de dados distribuídos que seja dimensionado automaticamente [Klimovic et al. 2018b]. Um direcionamento é integrar SGBDs com funções *serverless*. Entretanto, a computação *serverless* não possui armazenamento persistente interno, necessitando de armazenamento persistente remoto, o que introduz uma

grande latência [Hellerstein et al. 2019]. Por outro lado, o armazenamentos em memória principal, tais como os sistemas *Redis* e *Memcached*, oferecem alto desempenho mas possuem custo elevado e exigem que os usuários realizem a gestão dos recursos.

Aplicações que fazem uso intensivo de dados são prejudicadas devido ao alto volume de operações de I/O. Isto significa dizer que, em contextos de intensa execução, com várias funções requisitando armazenamento, tais serviços apresentam problemas de latência, criando um cenário de execução inviável para muitas aplicações. Assim, os desenvolvedores precisam escolher entre custos, latência e escalabilidade. Um dos principais desafios na execução de cargas de trabalho de análise em plataformas *serverless* é permitir que tarefas em diferentes estágios de execução comuniquem dados de maneira eficiente entre si por meio de um armazenamento de dados compartilhado. Alguns sistemas como *Crail* ou *Pocket* implementam novos abordagens de armazenamento compartilhado [Klimovic et al. 2018b].

Um direcionamento interessante é construir um serviço de armazenamento que combine diferentes serviços atuais e tendo tendo como base as próprias funções de nuvem. A memória das funções pode ser explorada como armazenamento efêmero de baixa latência em um serviço de *cache* conforme proposto por [Wang et al. 2020]. Para viabilizar soluções onde são utilizadas as próprias funções em instâncias na nuvem, faz-se necessário considerar diferentes aspectos, tais como i) um mecanismo de endereçamento de baixo *overhead* que permita localizar dados armazenados nas instâncias; ii) formas de garantir consistência e tolerância a falhas; e iii) estratégias para lidar com as restrições de rede tendo em vista aumentar taxa de transferência de dados.

2.2. Processamento de Dados e Comunicação

Aspectos de desempenho na execução e a localização aleatória das funções dificultam a concepção de um mecanismo de endereçamento de baixo *overhead*. Como consequência, aplicações se tornam mais sensíveis à latência e com necessidade de comunicação com serviços externos para sincronização. O uso da rede é ainda mais necessário quando se observa que as plataformas *serverless* seguem uma arquitetura *data-shipping* na qual os dados devem ser transportados até as funções para que ocorra o processamento, aumentando a latência, consumo de banda e custos. Funções de nuvem, no entanto, contam com banda de rede bem mais limitada se comparado com máquinas virtuais [Hellerstein et al. 2019].

Os provedores FaaS focam em maximizar o número de funções por máquina virtual (VM), o que contribui para aumentar a contenção de recursos, já que todas as funções dividem a largura de banda disponível [Sreekanti et al. 2020]. Por exemplo, uma função na *AWS Lambda* pode suportar apenas 40 MB/s de largura de banda, em contraste com 1 GB/s no caso de VMs de tamanho médio. Da mesma forma, topologias de comunicação comuns usadas para *datacenter* como *MapReduce* estruturado em árvore ou em anel são complexos de implementar nesses ambientes.

Outro ponto importante é que os SGBDs atuais assumem protocolos orientados à conexão, ou seja, os SGBDs são executados como serviços que aceitam conexões de clientes. Como o ambiente *serverless* é orientado a eventos, torna-se um desafio a construção de serviços de dados. Além disso, SGBDs de alto desempenho utilizam memória compartilhada ao passo que as funções em nuvem são executadas isoladamente e, portanto, não podem compartilhar a memória. Por outro lado, os SGBDs distribuídos, em geral,

utilizam uma arquitetura *shared-nothing* e assim não requerem memória compartilhada. Contudo, estes esperam que os nós dos SGBDs sejam endereçáveis diretamente na rede [Hellerstein et al. 2019]. Dessa forma, deve-se repensar os sistemas atuais para executar no ambiente *serverless*, principalmente em cenários com cargas de trabalho complexas que envolvem comunicação entre as funções [Pu et al. 2019].

Ambientes de computação em nuvem têm sido amplamente utilizados para armazenar e processar grandes volumes de dados. No entanto, cargas de trabalho de análise são variáveis, deixando recursos ociosos na maior parte do tempo, implicando em custos desnecessários. Por outro lado, as funções em nuvem tais como *AWS Lambda* ou *Azure Functions* executam tarefas pequenas e de granularidade fina e podem ser uma alternativa neste contexto. Um dos principais desafios na execução de cargas de trabalho de análise em plataformas de computação *serverless* é o compartilhamento eficiente de dados entre tarefas. Ao contrário de aplicativos simples orientados a eventos que consistem em uma única tarefa executada em resposta a um acionador de eventos, as cargas de trabalho de análise geralmente consistem em vários estágios e exigem que os resultados intermediários sejam compartilhados entre os estágios das tarefas. Nas estruturas analíticas tradicionais, tais como *Spark* e *Hadoop*, as tarefas armazenam os dados intermediários no armazenamento local por meio de *buffer* e trocam dados entre tarefas diretamente pela rede, que é um aspecto complexo da implementação no ambiente *serverless*. Trabalhos recentes, tais como [Perron et al. 2020] propõem mecanismo para a execução de consultas construídos diretamente sobre funções em nuvem com estratégias para mitigar a comunicação e o sincronismos entre as funções.

Em [Werner et al. 2018], os autores apresentam uma estratégia de processamento de *big data* em ambiente *serverless* que apresentam melhores resultados em termos desempenho, escala e custos do que as soluções atuais de computação distribuídas. Entretanto, padrões complexos de comunicação e cargas de trabalho variáveis para o contexto *serverless* permanecem uma questão de pesquisa em aberto. Existem algumas iniciativas para manter o estado em um ambiente *serverless*. Por exemplo, o *Azure Durable Functions* é uma extensão do Azure Functions que permite gravar funções com estado. Esta extensão gerencia estado e possui pontos de verificação [Microsoft 2019]. Em [Akhter et al. 2019] é apresentado um modelo de programação de alto nível que permite aos usuários especificar a lógica de negócios na forma de funções com estado. As funções podem ser chamadas por meio de terminais nomeados e podem ser coordenadas por meio de transações distribuídas, garantindo um estado distribuído consistente.

2.3. Elasticidade

A elasticidade é o ponto chave para desenvolver serviços com qualidade, pois permite adicionar ou remover recursos, sem interrupções e em tempo de execução para lidar com a variação da carga. Um SGBD é elástico se ele ajusta automaticamente a quantidade de recursos para a carga de trabalho atual, ou seja, adiciona novos recursos se o sistema não consegue lidar com a carga de trabalho atual ou remove recursos desnecessários. Aplicações que executam cargas de trabalho sensíveis à latência precisam de elasticidade [Sousa et al. 2018].

Vale destacar que mesmo SGBDs elásticos podem se tornar rapidamente gargalos em face da velocidade superior de escalonamento das funções pois estas escalonam

para centenas de funções em segundos [Schleier-Smith 2019]. Um serviço de armazenamento ideal para a computação *serverless* deve escalonar ao nível das funções, responder com baixa latência, apresentar um custo viável e ser tarifado conforme o uso. As soluções atuais apresentam latência e custo médios que dificultam seu uso em cenários de intensa manipulação de dados. Ao mesmo tempo, porém, podem ser opções altamente recomendadas como armazenamento permanente, dadas as garantias de disponibilidade e tolerância a falhas que oferecem.

Desenvolver SGBDs com escalabilidade e elasticidade é uma tarefa complexa, já que a maioria dos SGBDs em nuvem utiliza arquiteturas *shared-nothing*, tornando a disposição dos dados uma questão importante, principalmente considerando que as consultas dos usuários envolvem dados relacionados em diferentes servidores, o que exige o transporte dos dados, diminuindo o desempenho do sistema.

Para obter um bom desempenho, a infraestrutura deve estar apta e disposta a alocar fisicamente as funções e os dados. Em geral, é preferível enviar as funções aos dados, em vez da atual abordagem de extrair dados para as funções. Ao mesmo tempo, a elasticidade exige que as funções e os dados sejam separados logicamente, para permitir que a infraestrutura se adapte a alocação, visto que às vezes os dados precisam ser replicados ou reparticionados para corresponder às necessidades das funções. Em essência, esse é o desafio tradicional da independência de dados, mas em uma escala extrema e variável. Linguagens como SQL associadas com *MapReduce* e *TensorFlow* podem tratar esta questão, melhorando a interação com os dados por meio de uma linguagem de alto nível. Quanto melhor a linguagem declarativa, mais separação lógica é oferecida à infraestrutura [Hellerstein et al. 2019].

A organização independente de serviços de computação e armazenamento facilita a implementação da elasticidade. Por outro lado, apresenta desafios de desempenho e consistência para aplicações executadas em plataformas *serverless*. Em [Wu et al. 2020] é apresentada uma solução baseada em cache distribuído que implementa garantias de consistência causal melhorando o desempenho e impedindo anomalias de consistência presentes nas quais a plataformas *serverless*.

Associada a elasticidade, tem-se o desafio de definir a melhor forma de fornecer serviços de banco de dados *serverless* com garantias de QoS e pagamento conforme o uso [Abadi et al. 2019]. As principais plataformas *serverless* ainda não fornecem APIs com SLAs, usando apenas a quantidade de recursos de memória RAM, número de núcleos e o tempo de execução utilizado. Para fornecer certas garantias de QoS, essas plataformas precisam comunicar os requisitos de QoS necessários aos componentes dependentes e devem fornecer suporte para SLAs com preços apropriados.

3. Conclusão

Este artigo apresentou uma visão da computação *serverless* e gerenciamento de dados neste cenário, assim como oportunidades e desafios. Apesar de algumas soluções atenuarem estes desafios, ainda existe muito a fazer, principalmente porque o volume de dados e a velocidade com que os dados são gerados cresce de forma exponencial. Estes desafios geram oportunidades de pesquisa a serem superadas de forma que a computação *serverless* possa ser utilizado pelas empresas, melhorando seus produtos e serviços.

Referências

Abadi, D., Ailamaki, A., and et al. (2019). The seattle report on database research. *SIGMOD Rec.*, 48(4):44–53.

Akhter, A., Fragkoulis, M., and Katsifodimos, A. (2019). Stateful functions as a service in action. *Very Large Data Bases (VLDB) Conference*.

Hellerstein, J. M., Faleiro, J. M., Gonzalez, J., Schleier-Smith, J., Sreekanti, V., Tumanov, A., and Wu, C. (2019). Serverless computing: One step forward, two steps back. In *Conference on Innovative Data Systems Research (CIDR)*.

Jonas, E., Schleier-Smith, J., and et al. (2019). Cloud programming simplified: A berkeley view on serverless computing. Technical Report UCB/EECS-2019-3, EECS Department, University of California, Berkeley.

Klimovic, A., Wang, Y., Stuedi, P., Trivedi, A., Pfefferle, J., and Kozyrakis, C. (2018a). Pocket: Elastic ephemeral storage for serverless analytics. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 427–444.

Klimovic, A., Wang, Y., Stuedi, P., Trivedi, A., Pfefferle, J., and Kozyrakis, C. (2018b). Pocket: Elastic ephemeral storage for serverless analytics. In *USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, pages 427–444.

Microsoft (2019). *Durable Functions patterns and technical concepts* . `https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-concepts`.

Perron, M., Fernandez, R. C., DeWitt, D. J., and Madden, S. (2020). Starling: A scalable query engine on cloud functions. In *Proceedings of the SIGMOD*, pages 131–141. ACM.

Pu, Q., Venkataraman, S., and Stoica, I. (2019). Shuffling, fast and slow: Scalable analytics on serverless infrastructure. In *USENIX Symposium on NSDI*, pages 193–206.

Schleier-Smith, J. (2019). Serverless foundations for elastic database systems. In *Conference on Innovative Data Systems Research (CIDR)*.

Sousa, F. R. C., Moreira, L. O., Filho, J. S. C., and Machado, J. C. (2018). Predictive elastic replication for multi-tenant databases in the cloud. *Concurr. Comput. Pract. Exp.*, 30(16).

Sreekanti, V., Wu, C., Lin, X. C., Schleier-Smith, J., Faleiro, J. M., Gonzalez, J. E., Hellerstein, J. M., and Tumanov, A. (2020). Cloudburst: Stateful functions-as-a-service. *ArXiv*, abs/2001.04592.

Wang, A., Zhang, J., Ma, X., Anwar, A., Rupprecht, L., Skourtis, D., Tarasov, V., Yan, F., and Cheng, Y. (2020). Infinicache: Exploiting ephemeral serverless functions to build a cost-effective memory cache. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*, pages 267–281.

Werner, S., Kuhlenkamp, J., Klems, M., Muller, J., and Tai, S. (2018). Serverless big data processing using matrix multiplication as example. In *BigData*, pages 358–365. IEEE.

Wu, C., Sreekanti, V., and Hellerstein, J. M. (2020). Transactional causal consistency for serverless computing. In *Proceedings of the SIGMOD*, pages 83–97.