SplitServe: Efficiently Splitting Complex Workloads Across FaaS and IaaS

Aman Jain
The Pennsylvania State University
axj182@psu.edu

Ata F. Baarzi
The Pennsylvania State University
azf82@psu.edu

Nader Alfares
The Pennsylvania State University
nna5040@psu.edu

George Kesidis
The Pennsylvania State University
gik2@psu.edu

Bhuvan Urgaonkar
The Pennsylvania State University
buu1@psu.edu

Mahmut Kandemir
The Pennsylvania State University
atk2@psu.edu

Abstract

Amazon Web Services (AWS) Lambdas and other "cloud functions" (CFs) offer much lower startup latencies than virtual machines (VMs) (tens/hundreds of milliseconds vs. a few/several minutes) with lower minimum cost. This makes it appealing to use them for handling unexpected spikes in *simple, stateless* workloads [2, 3, 5]. If the spike persists, additional VMs may be launched and CFs can be decommissioned when the VMs are ready (VMs are cheaper per unit resource procured than CFs). However, it is not immediately clear if using CFs for *complex* workloads - those involving significant state exchange among components - is similarly effective. Current CFs have several restrictions that may limit their efficacy: (i) relatively limited resource capacity, especially main memory (e.g., an AWS Lambda may only have up to 3GB memory), (ii) limited lifetime (e.g., Lambdas are terminated after 15 minutes), and (iii) limited support for sharing of intermediate state (e.g., Lambdas must employ an external storage system such as AWS S3). Contrary to conventional wisdom, we show that **it is possible to exploit the faster startup times of CFs to improve cost and performance of autoscaling even for complex workloads.**

Approach: We design SplitServe [1], implemented as an enhancement of Apache Spark [4], that is capable of simultaneously using AWS VMs and Lambdas for serving the tasks comprising a parallel Spark job. The most salient challenges addressed and design choices made in our efforts are: (i) **State exchange:** Instead of relying on a slower external cloud storage to transfer state, we leverage the resources associated with the procured VMs and employ HDFS for state exchange. We find that this allows both VMs and Lambdas to achieve throughputs close to that of local disks. Since we are using already provisioned disk capacity, we do not pay extra (as we would if we were to use, say, AWS S3). (ii) **Segueing from Lambdas to newly available VMs:** Simply killing ongoing tasks on Lambdas and rerunning them on newly available VMs triggers Spark's high overhead fault tolerance mechanisms. So, a diaphanous scheduling decision, based on the amount of time a Lambda function has been running, is made at per task granularity. Briefly, as the time since a Lambda was launched approaches the common-case startup delay for a VM, new tasks are not sent to the Lambda.

Findings: In our experiments, we find that SplitServe reduces overall job execution time compared to the state of the art with either a **homogeneous or heterogeneous execution environment**, i.e., either all VMs or all Lambdas, or simultaneously involving both VMs and Lambdas to execute a job's tasks. For the heterogeneous case, our experimental evaluation of SplitServe using four different workloads (interactive TCP-DS, K-means clustering, PageRank, and Pi) shows that SplitServe-Spark improves performance up to 55% for workloads with small to modest amount of shuffling, and up to 31% in workloads with large amounts of shuffling, when compared to only VM based autoscaling. Also, with its novel segueing technique, SplitServe can help reduce costs by up to 21% while still providing almost 40% reduction in execution time.

Ongoing Work: We are designing a comprehensive autoscaling system that leverages SplitServe's capabilities. We will carry out an empirical evaluation of the performance/cost improvements such a system can offer over state of the art solutions with diverse workloads that exhibit realistic dynamism and uncertainty.

CCS Concepts

• **Computer systems organization → Cloud computing**; • **Information systems → Cloud based storage**; *MapReduce-based systems.*

ACM Reference Format:
Aman Jain, Ata F. Baarzi, Nader Alfares, George Kesidis, Bhuvan Urgaonkar, and Mahmut Kandemir. 2019. SplitServe: Efficiently Splitting Complex Workloads Across FaaS and IaaS. In *ACM Symposium on Cloud Computing (SoCC '19), November 20–23, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3357223.3366027

References

[1] A. Jain, A.F. Baarzi, N. Alfares, G. Kesidis, B. Urgaonkar, and M. Kandemir. June 2019. SplitServe: Efficient Splitting Complex Workloads across FaaS and IaaS. https://github.com/PSU-Cloud/splitserve-spark/blob/master/Paper/SplitServe.pdf.

[2] J.H. Novak, S.K. Kasera, and R. Stutsman. [n.d.]. Cloud Functions for Fast and Robust Resource Auto-Scaling. https://rstutsman.github.io/papers/feat.pdf.

[3] J. Raj, M. Kandemir, B. Urgaonkar, and G. Kesidis. July 2019. Exploiting Serverless Functions for SLO and Cost Aware Tenant Orchestration in Public Cloud. In *Proc. IEEE Cloud.* Milan.

[4] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proc. USENIX HotCloud.*

[5] Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. 2019. MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving. In *Proc. USENIX ATC.* Renton, WA.

This research was supported by the National Science Foundation grant CNS-171751.

SoCC '19, November 20–23, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6973-2/19/11.
https://doi.org/10.1145/3357223.3366027