FUSIONIZE: Improving Serverless Application Performance Using Dynamic Task Inlining

Trever: Needs a new title that shows 35% more

Trever Schirmer, Joel Scheuner, Tobias Pfandzelter and David Bermbach

Abstract—Serverless computing increases developer productivity by removing operational concerns such as managing hardware or software runtimes. Developers, however, still need to partition their application into functions, which can be error-prone and adds complexity: Using a small function size where only the smallest logical unit of an application is inside a function maximizes flexibility and reusability. Yet, having small functions leads to invocation overheads, additional cold starts, and may increase cost due to double billing during synchronous invocations. In this paper we present FUSIONIZE, a framework that removes these concerns from developers by automatically fusing the application code into a multi-function orchestration. Developers only need to write the application code following a lightweight programming model and do not need to worry how the application is turned into functions. Our framework automatically fuses different parts of the application into functions, configures the underlying infrastructure, and manages their interactions. Leveraging monitoring data, the framework optimizes the distribution of application parts to functions, the amount of resources the functions can access, and the interaction between functions to optimize deployment goals such as end-to-end latency and cost. Using two real-world use cases, we show that FUSIONIZE can automatically and iteratively improve the deployment artifacts of the application.

Index Terms—serverless computing, FaaS, function fusion, cloud orchestration

1 INTRODUCTION

WITH the advent of serverless cloud computing, the Function-as-a-Service (FaaS) execution model has become a viable paradigm for large applications [6], [22]. In FaaS, developers compose stateless, event-driven tasks that invoke each other to implement complex workflows [21], [24], [31]. These tasks are deployed as functions on a cloud FaaS platform that abstracts operational concerns such as managing hardware or software runtimes, offering a flexible pay-as-you-go billing model. Today, FaaS platforms are offered by all leading cloud providers, e.g. AWS Lambda[1], Google Cloud Functions[2], and Microsoft Azure Functions[3], and are an area of major research interest [1], [4], [7], [10], [11], [19].

Despite the operational benefits of building applications as compositions of FaaS functions, we observe a gap between the developer-side logical view of complex applications and the performance and cost-efficiency characteristics of commercial cloud FaaS platforms. On the one hand, application developers want to split their applications into isolated, single-purpose tasks that can be independently updated and worked on, may be dynamically recomposed, and that improve code reusability [17]. Notably, in the context of serverless computing, developers also hope to limit the work needed to optimize their software for a particular infrastructure stack. On the other hand, FaaS platforms incentivize large, monolithic functions because of call overheads and invocation costs, and still expose configuration parameters that have complicated effects on the performance of FaaS applications [10], [25].

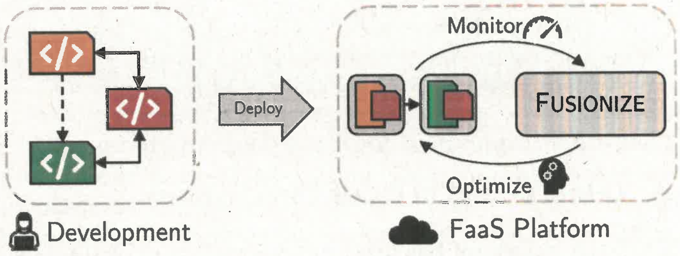

Fig. 1. FUSIONIZE takes existing, unchanged FaaS tasks and optimizes their performance and cost-efficiency by iteratively modifying their deployment configuration and inlining tasks based on monitored performance. The developer-written tasks are deployed inside multiple FaaS functions, where they can be inlined instead of called remotely.

In this paper, we address this gap with FUSIONIZE, a feedback-driven system for the automated configuration of composite task-based applications on cloud FaaS platforms. A high-level overview of FUSIONIZE is shown in Figure 1 We borrow the concept of *inlining* in compilers to expand remote FaaS functions calls with task source code where beneficial, a concept called *function fusion* [4], [32]. Function

- *T. Schirmer, T. Pfandzelter, and D. Bermbach are with the Mobile Cloud Computing research group at TU Berlin & Einstein Center Digital Future, Berlin, Germany.*
  E-mails: {ts, tp, db}@mcc.tu-berlin.de
- *J. Scheuner is with the Internet Computing and Emerging Technologies Lab (ICET-lab) at Chalmers University of Technology and the University of Gothenburg, Gothenburg, Sweden.*
  E-mail: scheuner@chalmers.se

[1] https://aws.amazon.com/lambda
[2] https://cloud.google.com/functions
[3] https://azure.microsoft.com/products/functions/