International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Hyper FaaS: A Truly Elastic Server less Computing Framework

Sachin Kumar

Asst.Prof., BCA Dept., IIMT College of Management, Greater Noida

Abstract:
Hyper FaaS is a next-generation serverless computing framework designed to deliver true elasticity, low latency, and efficient resource utilization. It dynamically scales function execution based on real-time workload demands, eliminating over provisioning and cold-start delays. By integrating intelligent scheduling, lightweight virtualization, and adaptive resource management, Hyper FaaS enhances performance and cost efficiency. The framework supports high-throughput, event-driven applications and enables seamless deployment across heterogeneous cloud environments, making it suitable for modern, large-scale distributed systems.

Keywords :- Hyper FaaS, Serverless Computing, True Elasticity, Intelligent Scheduling
Adaptive Resource Management

Introduction & Motivation

More recently, a new paradigm called server less computing (Function-as-a-Service or FaaS) has become prevalent, thanks to container-based virtualization. Server less computing enables a brand new way of building and scaling applications and services by breaking the traditionally monolithic server based application model into finer-grained functions; developers can thus focus on the development of function logics without having to worry about the server or VM management.

While still in its infancy, extant serverless computing infrastructure starts to pose a set of issues. The foremost issue is the significant runtime overhead that hampers its elasticity and scalability. Simple adoption of container engines such as Docker incurs significant container startup and runtime environment setup cost, thus making the server less platforms fail to meet the latency and through put requirement of bursty workloads.

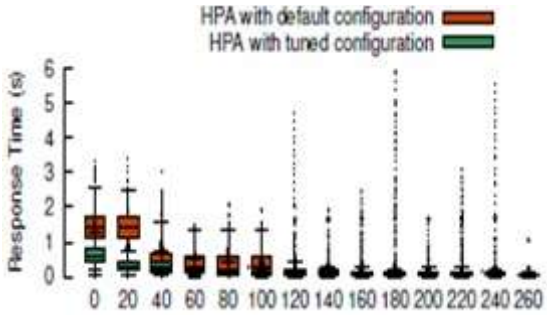

Figure 1: Careful HPA parameter tuning yields effective performance improvement and agile response to a workload burst.

Issues State-of-the-art open-source FaaS platforms heavily rely on container orchestration frameworks such as Kubernetes for container resource scheduling, which is originally designed for managing long-running applications such as Mecached. FaaS-based applications demand elastic auto-scaling when workloads are bursty, whereas Kubernetes' container scheduling and auto-scaling proto coli too heavy weight to realize instant resource provisioning. For example, Kubernetes' horizontal pod auto scaler (HPA) first waits for multiple rounds of statistics collection until the measured metric (e.g., CPU usage of a certain pod) stabilizes before enters into the container scaling-out phase where the number of needed containers is calculated and provisioned.

To demonstrate, werun HPA-managed Open FaaS onaclusterofAWSEC2VMsandissueahighly-concurrent function workload. As shown inFigure1,the workload sees reduced response time after 40 seconds and stabilizes after 120 seconds, which is unacceptably long for a short-lived bursty workload.

By carefully tuning the HPA parameters, we observe dramatically improved elasticity and quickly-converged function response time (the response time stabilizes after 20 seconds). This observation is because recon figured HPA is more sensible to workload burst, and therefore, makes auto-scaling decision in a more timely fashion.

Designing HyperFaaS

Optimization such as container caching (or pre- warming, by launching a pool of empty standby containers ready for serving function requests) cannot fundamentally solve the problem, because bursty workloads can exhaust the container cache pool quickly, resulting in excessive container provisioning afterward. HPA parameter tuning, on the other hand, can marginally mitigate the impact of slow auto-scaling to some extent. However, it pinpoints the root causes of such deficiency and motivates us to rethink the FaaS platform design. To this end, we propose Hyper FaaS—a redesign of the FaaS platform that supports truly elastic resource scaling with high resource efficiency. The goal of Hyper FaaS is two-fold:

1. Maximize the resource utilization and efficiency through hierarchical scheduling and container sharing; and
2. Minimize performance loss due to highly- concurrent bursty function workload sviatransient resource scaling-up.
3. Hierarchical Scheduling Within one container, there are two management modules:
4. In- container scheduler (ICS)that routes requests tofunction workers that serve the function requests.
5. Each ICS performs periodic metrics monitoring and proactively exchanges information with a centralized global scheduler (GS). The ICS keeps the quality-of-service (QoS) of function request serving in check with pre- defined resource (CPU/memory) usage watermarks.

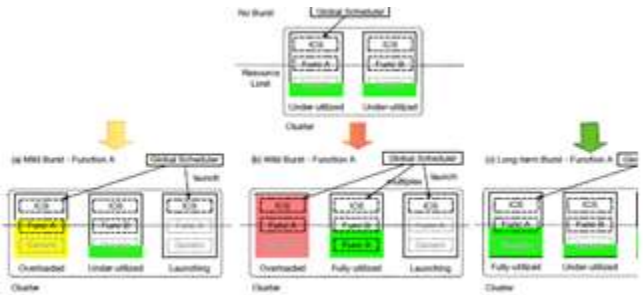

Figure 2: Overview of multi-phase auto-scaling scheme: (a): Transient scaling-up offers a temporary

solution to mitigate the impact of mild burst, while waiting for the new containers to be launched from background; (b): Container multiplexing effectively serves wilder burst; (c): When the scaling-out process is finished, distribute the persisting burst to the new containers.

Whenever the high watermark is reached, ICS notifies GS to take actions. Additional requests that would have overwhelmed that particular container will be bounced back to the GS, which performs rescheduling by looking for a different set of containers with available resources. This hierarchical scheduling scheme can effectively prevent container overwhelming and thus significantly improve the average and tail latencies. Multi-phase Auto-scaling To effectively handle suddenly bursty workloads, we design a multi-phase "burst-buffer" styled scaling mechanism, where we follow a "scaling-up then scaling-out" design choice. A phase-one resource scaling-up is always applied first, whenever a bursty workload comes, with the hope that the burst is ephemeral and short-lived, as shown in Figure2(a); Hyper FaaS triggers the scaling-out as a last resort if the workload spike persists. Our phase two scaling-out policy coordinates with GS to try to locate containers with available resources to sustain the increased workload, (Figure 2(b)); if there are not enough free resources, Hyper FaaS enters into phase-three by asynchronously spinning up (via cold-start) more containers in the background,while phase-one scaling-up is used as a burst buffer to help form a smooth latency decrease curve (instead of a sudden jump suffered by existing platforms), as shown in Figure 2(c). Container Sharing To further complement the above design decisions, Hyper FaaS implements container sharing through a multi-function based resource multiplexing, with the goal of improving resource utilization. For this purpose, we adopt a double-buffering typed approach, where two function workers are co-existing within one container. Our container sharing scheme featuresa carefully-designed and lightweight function switching protocol that works as follows:
when a container is lightly-loaded, one of two workers is loaded with function codes serving requests, while the other worker stays generic without being specialized; workers can switch between generic (unspecialized) and non- generic (specialized with source codes of the specific function). When a burst goes beyond the scaling-up threshold, GS will locate on tainers that are currently under-utilized, and specialize the generic worker for container multiplexing. Implementation in Progress We are in theprogress of implementing a prototype of HyperFaaS. To minimize the over head of ICS, we use request streaming and directly pipe data to the worker without parsing the HTTP headers. The hierarchical scheduling and container sharing substrates are under heavy development. For the evaluation purpose, only CPU and memory resource will be considered in the first version.

Related Work

Extant serverless platforms suffer from containers' cold startup and use container caching for performance improvement. SOCK proposes a serverless-optimized container engine to mitigate the impact of long container latencies. SAND introduces per-workflow container sharing to optimize performance for workflow-based serverless workloads.
The proposed work shares with these systems the goal of reducing the container-level costs, but differs in its focus on resource scheduling.

Conclusion

In conclusion, Hyper FaaS represents a significant advancement in serverless computing by addressing

key limitations of existing platforms. Its ability to provide true elasticity, minimize latency, and optimize resource utilization ensures consistent performance under dynamic workloads. Through intelligent scheduling and adaptive management, Hyper FaaS improves cost efficiency and scalability, making it a robust and practical solution for next-generation, large-scale, event-driven cloud applications.

References:

1. Istemi Ekin Akkus, Ruichuan Chen, Ivica Rimac, Manuel Stein, Klaus Satzke, Andre Beck, Paarijaat Aditya, and Volker Hilt. Sand: towards high-performance serverless computing. In USENIX ATC, 2018.
2. Amazon. Aws lambda. https://aws.amazon.com/lambda/.
3. The Kubernetes Authors. Production-grade container orchestration-kubernetes. https://kubernetes.io/.
4. Docker. Enterprise container platform. https://www.docker.com/.
5. Alex Ellis. Introducing functions as a service. https://goo.gl/c8mnD2.
6. Fission. Serverless functions for kubernetes. https://goo.gl/VJKbGV.