Serverless Control Planes for Orchestration of Cloud Resources Extended Abstract

Alexander Fuerst

1. Motivation

Cloud computing is comprised of a variety of services and abstractions to reduce the complexity of building and running applications. Function as a Service (FaaS) is a unique abstraction amongst the many: user supplied code entirely managed by the cloud provider. Serverless control planes face unique challenges caused by the highly heterogeneous workloads predominant. Users run machine learning inference, web services, multimedia analysis, and even scientific computing, to the tune of billions of invocations per day. These, in turn, vary widely in resource usage, execution time, and frequency of invocation, which the control plane must handle efficiently. A typical function invocation runs for less than a second and uses only 128 MB of memory, after invocation, these resources can sit idle indefinitely. Handling functions like classical cloud applications can cause significant latency delays and tremendous wasted resources.

The diverse platforms serverless has been proposed to run on vary in scale, prupose, and even hardware design. Edge [5], public cloud, and heterogeneous [3] scenarios have different capabilities and resource requirements. Existing open-source platforms rely on complex and high-overhead systems like Docker or Kubernetes [1, 2] which do not fit on low-powered devices and perform poorly at FaaS scale. Each invocation must run inside an isolation mechanism, which can take several seconds and add orders of magnitude of time to latency. The large tech stacks of these designs wastes precious resources, adds 100s of milliseconds to invocation overhead, and are ill-suited to fast-paced research advancements.

2. Insights

A key insight of this thesis is that existing open-source FaaS platforms are neither performant at scale nor launchpads for advanced research across the many serverless topics. Current server-full mechanisms for hosting software and managing resources poorly support the needs of the emerging FaaS ecosystem. Centralized load balancing or scheduling designs, such as is found in OpenWhisk [1], cause unnecessary overhead or fail to scale with FaaS workloads, often both. A platform meeting these performance goals that can run using limited system resources, is easily extensible, and runnable on devices from the edge-to-cloud continuum is needed.

3. Key Contributions

This thesis addresses performance and resource management challenges by developing novel system designs and tailoring algorithms to handle the new workloads and platforms of FaaS. I detail several algorithms that tackle poor resource allocation and load scheduling at both individual server and cluster levels. The key finding of orchestration at both levels is the reliance on function characteristics in decision-making. The centerpoint of this thesis is a redesigned control plane, called Ilúvatar [4]. Ilúvatar handles requests with less than *2 milliseconds* of platform overhead and reduces latency spikes by 100x found in existing open-source systems. This is primarily accomplished by disaggregating scheduling decisions to avoid contention between distributed services of the control plane. Finally, I leverage the high degree of resource control made possible by this new system to integrate GPU acceleration into the serverless ecosystem. This uses a variety of novel mechanisms to minimize overhead despite limited device resources, boosting performance by several orders of magnitude over baseline solutions. Altogether, the contributions of this thesis serverless improve latency by 75% and cluster resource utilization by up to 20x.

Ilúvatar was written in 20k lines of Rust code, with the goal of making new research easier at both ends of development and experimentation. This thesis shows two examples of this, first implementing alternate container isolation mechanisms that are faster than Docker, and second comparing GPU scheduling policies for complex invocation workloads. Results captured by the built-in workload generation tools are designed to be easily parseable for post-analysis writeup of results.

References

[1] Apache OpenWhisk: Open Source Serverless Cloud Platform, 2020.

[2] OpenFaaS : Server Functions, Made Simple. https://www.openfaas.com, 2020.

[3] Dong Du, Qingyuan Liu, Xueqiang Jiang, Yubin Xia, Binyu Zang, and Haibo Chen. Serverless computing on heterogeneous computers. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 797–813, 2022.

[4] Alexander Fuerst, Abdul Rehman, and Prateek Sharma. Ilúvatar: A Fast Control Plane for Serverless Computing. 2023.

[5] Adam Hall and Umakishore Ramachandran. An execution model for serverless functions at the edge. In *Proceedings of the International Conference on Internet of Things Design and Implementation - IoTDI '19*, pages 225–236, Montreal, Quebec, Canada, 2019. ACM Press.