FaaSPR: Latency-oriented Placement and Routing Optimization for Serverless Workflow Processing

Yunshan Jia, Chao Jin, Qing Li, Xuanzhe Liu, *Senior Member, IEEE,* and Xin Jin, *Senior Member, IEEE*

***Abstract*—**

***Index Terms*—Serverless, scheduling, placement and routing.**

I. Introduction

[TODO: Introduction to serverless computing.]

Resource management in serverless computing has been an extensively studied problem [1], with numerous solutions proposed in the literature[TODO: citation]. Most existing approaches operate at the function level; that is, resources are allocated to function instances, which further select requests to process. A key assumption underlying these approaches is that a serverless function exhibits consistent resource demands across all invocations.

However, recent analyses of real-world serverless workloads have revealed significant skew in resource consumption across different requests for the same function[TODO: citation]. Our analysis (§II-B) also confirms that the resource usage of a serverless function can vary by up to XXX times[TODO: variation] across different requests. This high variability leads to significant inefficiencies for conventional, function-level policies. To guarantee reliable performance, these policies must provision resources for a function's peak (i.e., worst-case) demand, resulting in substantial resource wastage for the vast majority of less-demanding requests. Additionally, we find that this input-dependent resource usage is not only skewed but also highly predictable, achieving an accuracy of XXX%[TODO: prediction accuracy].

Concurrently, techniques that allow for dynamic adjustment of a container's resource capacity have been proposed[TODO: citation(k8s, escra, etc.)]. These techniques make it possible to alter the resource allocation of a running container in-place, thereby avoiding the costly overhead of resource reallocation.

Capitalizing on these developments, we propose a *request-level* resource allocation policy that provisions the exact amount of resources for each request. The key idea is to predict the resource demand of each incoming request and dynamically adjust the target container's resource capacity accordingly. Such a fine-grained strategy ensures that requests are processed using only the necessary resources, which can significantly reduce cluster-wide resource wastage.

However, realizing an effective request-level resource management system introduces two challenges. First, to achieve high resource utilization through dynamic resource reallocation, the system must rely on oversubscription, where the sum of maximum possible resources required for colocated containers exceeds the physical capacity of the server. Given the high variability in per-request resource demands, a high oversubscription ratio leads to resource contention and frequent container evictions. This forces the system to adopt a low oversubscription ratio, negating much of the potential efficiency gains from request-level resource allocation.

Second, dynamic resource adjustment mechanisms introduce non-negligible overhead. Specifically, while increasing a container's resource capacity is a relatively fast operation, decreasing it is often much slower. For example, reducing the memory capacity of a container requires reclaiming memory pages, which incurs substantial latency. As resource adjustment occurs on the critical path of request execution, this latency directly increases the end-to-end request handling time.

To address these challenges, we propose FaaSPR, a serverless resource manager that allocates resources at the request level. To mitigate the risk of eviction from oversubscription, FaaSPR adopts a XXX[TODO: placement algorithm name] placement strategy. This strategy classifies requests into distinct tiers based on their predicted resource demands and specializes containers to handle requests from specific tiers. Moreover, this strategy incorporates a XXX[TODO: Other feature of placement algorithm. E.g., coloring.] technique to maximize server utilization by colocating containers with complementary resource requirements.

FaaSPR further tackles the overhead of resource adjustment with a XXX[TODO: scheduling algorithm name] request scheduling algorithm. This algorithm organizes incoming requests into monotonic queues, where requests are sorted by increasing resource demand. By processing the requests in a queue sequentially, a container's resource capacity only needs to be incrementally increased. A costly resource-decreasing operation is only required when the system finishes one queue and switches to another with a lower starting resource requirement. This design reduces the number of resource-decreasing operations, thereby significantly mitigating its impact on overall latency.

Experiments show that [TODO: summary of experiments results.]

[TODO: summary of contributions.]

Yunshan Jia, Chao Jin, Xuanzhe Liu, and Xin Jin are with the School of Computer Science, Peking University, Beijing, China. E-mail: {jiayunshan;chaojin;liuxuanzhe;xinjinpku}@pku.edu.cn.

Qing Li is with the School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China. E-mail: qingli@bupt.edu.cn.

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4500700, in part by the National Natural Science Foundation of China under Grant 62172008 and 62325201, and in part by the National Natural Science Fund for the Excellent Young Scientists Fund Program (Overseas). (Corresponding author: Xin Jin.)

II. Background and Motivation

A. resource management in Serverless

B. Function Resource Demand Analysis

C. Job scheduling in Serverless

add demo experiment here, sota -¿ overhead detail

III. FaaSPR Overview

Instance placement based on heuristic algorithm

job scheduling based on dynamic instance resizing

IV. FaaSPR Deisgn

A. Instance placement based on heuristic algorithm

B. Cold-start-free hybrid placement updating

C. job scheduling(todo: name)

V. Implementation

VI. Evaluation

A. Evaluation Setup

Benchmarks.

Workload generation.

Cluster configuration.

Baseline.

B. Benefits of FaaSPR

Comparison under different benchmark number.

Comparison under different resource distributions.

Performance analysis.

C. Scalability

Larger-scale cluster experiment.

System overhead.

D. Effectiveness of FaaSPR

Technique effectiveness.

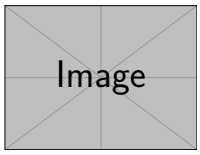
Parameter effectiveness.

VII. Related Work

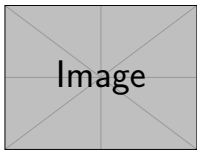
VIII. Conclusion

References

[1] A. Fuerst and P. Sharma, "Faascache: keeping serverless computing alive with greedy-dual caching," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, p. 386–400.



Xuanzhe Liu (Senior Member, IEEE) is a Full Professor in the School of Computer Science at Peking University, Beijing, China. His research interests mainly fall in service-based software engineering and systems. Most of his recent efforts have been published at prestigious conferences including WWW, ICSE, FSE, SOSP, SIGCOMM, NSDI, MobiCom, MobiSys, and in journals including ACM TOSEM/TOIS and IEEE TSE/TMC/TSC. He is a distinguished member of the ACM and the CCF. Web page: http://www.liuxuanzhe.com/.



Xin Jin (Senior Member, IEEE) received the PhD degree from Princeton University, in 2016. He is currently an associate professor (with Tenure) with the School of Computer Science, Peking University. His research interests include computer systems, networking, and cloud computing.