S-Cache: Function Caching for Serverless Edge Computing

Posco Fung Po Tso

for Coseners 2023

14th July 2023

Loughborough University

Publications

- Chen, Chen, Lars Nagel, Lin Cui, and Fung Po Tso. "S-Cache: Function Caching for Serverless Edge Computing." In Proceedings of the 6th International Workshop on Edge Systems, Analytics and Networking, pp. 1-6. 2023.
- We're extending this work.

Background – What is Serverless Computing

"Only turn on the containers when they are needed, and turn off when they are not needed"

The Problem

Serverless is great for edge computing because there is fewer resource at the edge.

- but the cold-start latency is the show stopper.

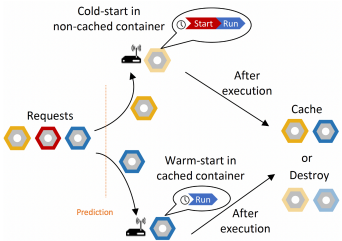
Solution

Function invocation prediction:

- Predicts the arrival of invocation requests
- Gated Recurrent Units[a]

Function Caching:

- Caches a function after being used. When new request arrives, reuse the cached functions, similar to the caching of objects on the memory.
- However, caching functions requires extra computational resources (e.g., memory).
- How to balance the overhead of caching with extra resource usage?



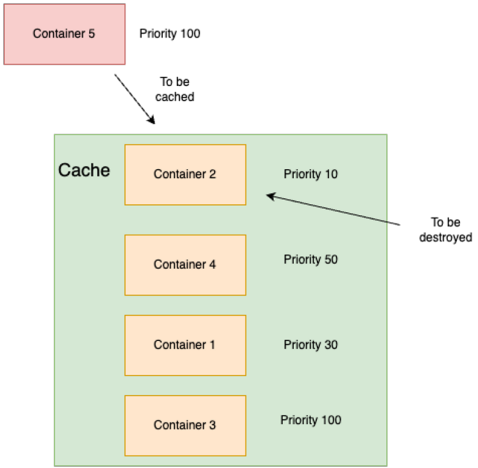
[a] https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsDataset2019.md

Proactive Function Caching

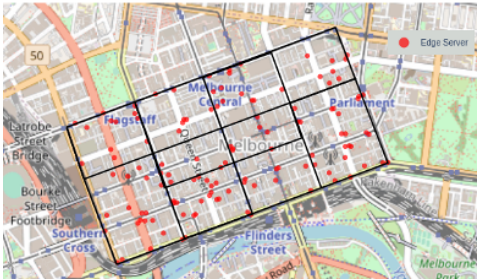
Caching Priority:

$$Priority = Clock + \frac{Freq \times ColdStartTime}{Size}$$

- Clock: A logical clock to reflect the recency of invocation
- Freq: The number of times a function is invoked till the current time interval
- ColdStartTime: The start up time of a container
- Size: Memory usage of a container



Testbed Experiments

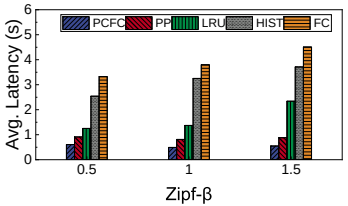


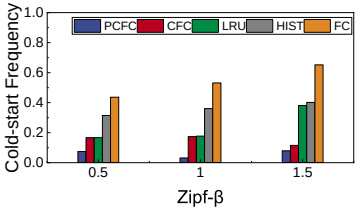
- Used a topology in Melbourne CBD area
- 125 edge servers divided into 10 areas
- Deployed on 10 Raspberry Pi 3Bs
- Implemented on Knative over Kubernetes

- Benchmarks:
  - Histogram (HIST)[a]: It terminates the container after each function execution but pre-warms the container before a potential next invocation.
  - Fixed Caching (**FC**): It is widely used in AWS Lambda. It keeps a container alive for a fixed period of time.

[a] Shahrad, Mohammad, et al. "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider." USENIX (2020).

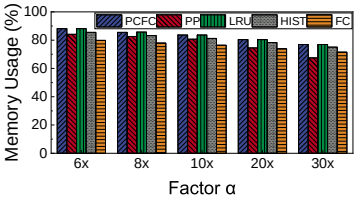
Results - Average Latency & cold-start frequency



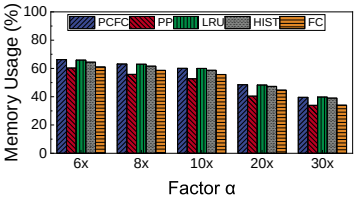


- PCFC achieves about 0.49-0.6s in latency.
- The cold start frequency of CFC ranges from 0.03 to 0.07.
- Reduce the avg. latency by up to 93
- Reduce the cold start frequency by up to 99

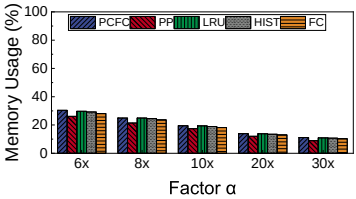
Results - Memory usage



(a) Zipf-β=0.5



(b) Zipf-β=1



(c) Zipf-β=1.5

- PCFC uses extra 4.9% memory resource compared to PP.
- This is a tradeoff by using function caching.

Conclusion

- Proactive Function caching significantly reduces the latency
- Tradeoff between response time and resource usage is key
- Caching needs to consider the nature of containers (e.g., frequency, resource footprint)

That's it – Thank you

Questions?

p.tso@lboro.ac.uk