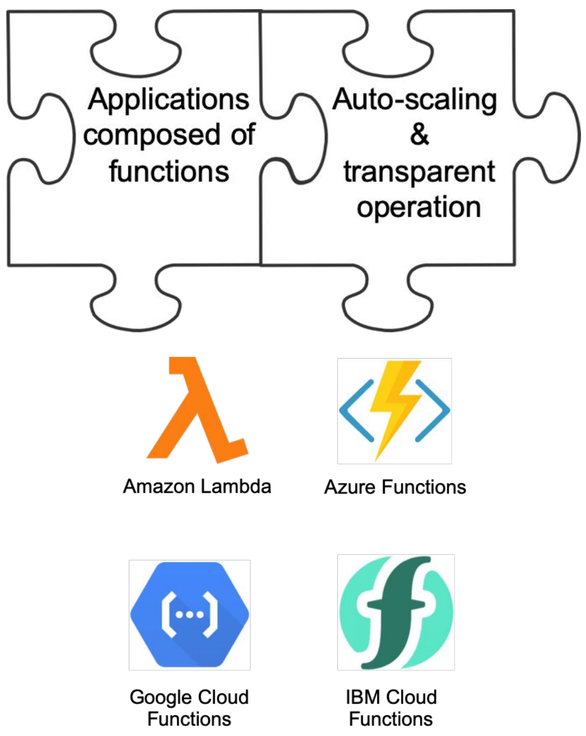
Architectural Implications of Function-as-a-Service Computing

Mohammad Shahrad, Jonathan Balkind, David Wentzlaff
Princeton University

Function-as-a-Service (FaaS) Serverless Computing



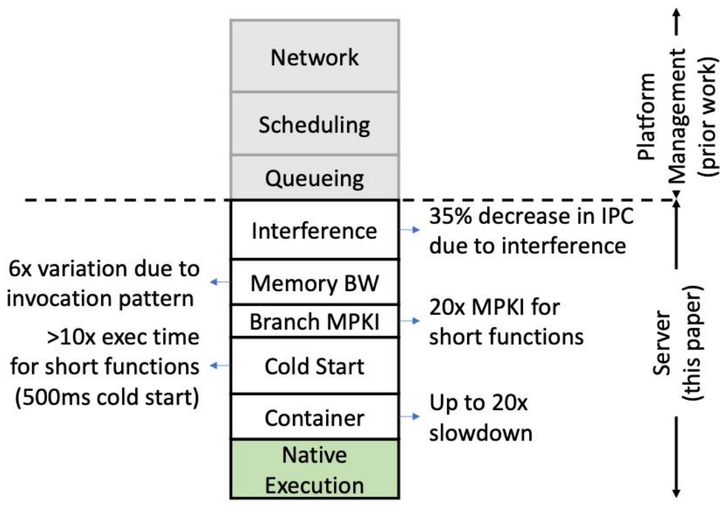
FaaS Differs From Prior Cloud Offerings

- Short function executions
- Developer does no server provisioning
- High concurrency (with inefficient isolation)
- Fine-grained pricing based on execution time, memory, and request counts
- Machine type not guaranteed/unknown

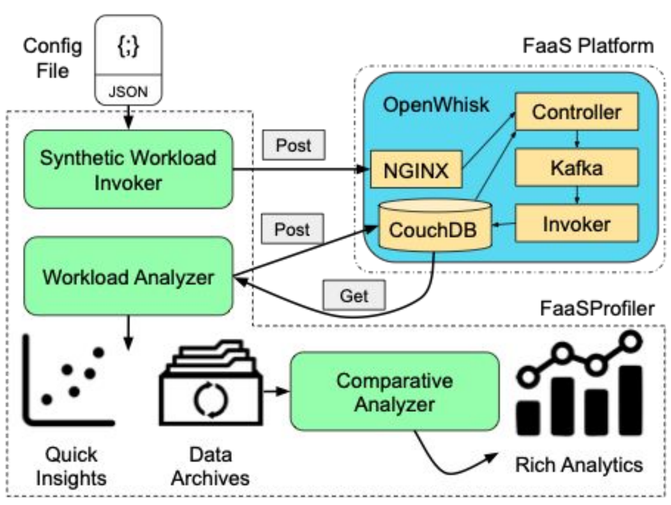
Zooming in on a Serverless Server

Prior work:

- External reverse engineering of serverless offerings
- System-level design
- Building new or mapping old applications



We use Apache OpenWhisk



We built FaasProfiler

- FaasProfiler invokes functions in chosen invocation patterns
- Collects profiling data from standard tools and performance counters
- Results can be compared/analysed using standard Python data science tools

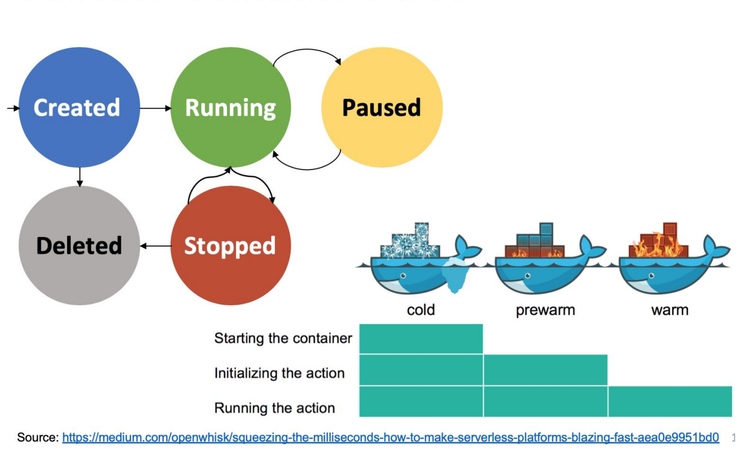
https://github.com/PrincetonUniversity/faas-profiler

Performance Criteria & Breakdown

Test Setup:

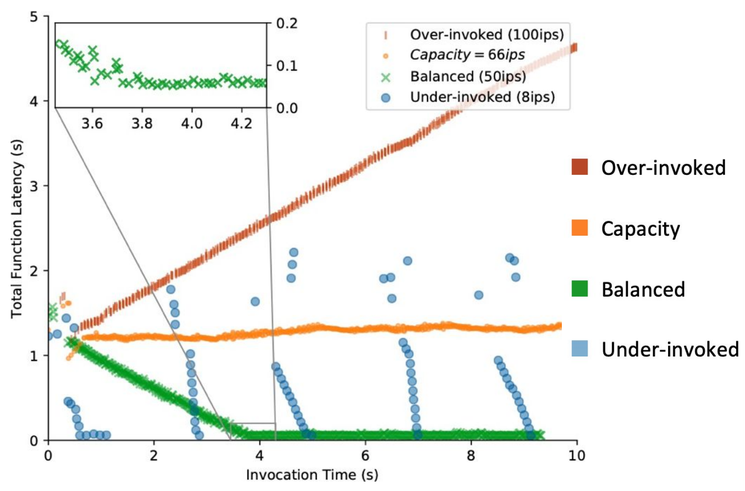
Intel Xeon E5-2620 v4, 8-cores (16-threads), 20MB Last-Level Cache, 16GB 2133MHz DDR4

Function Container States

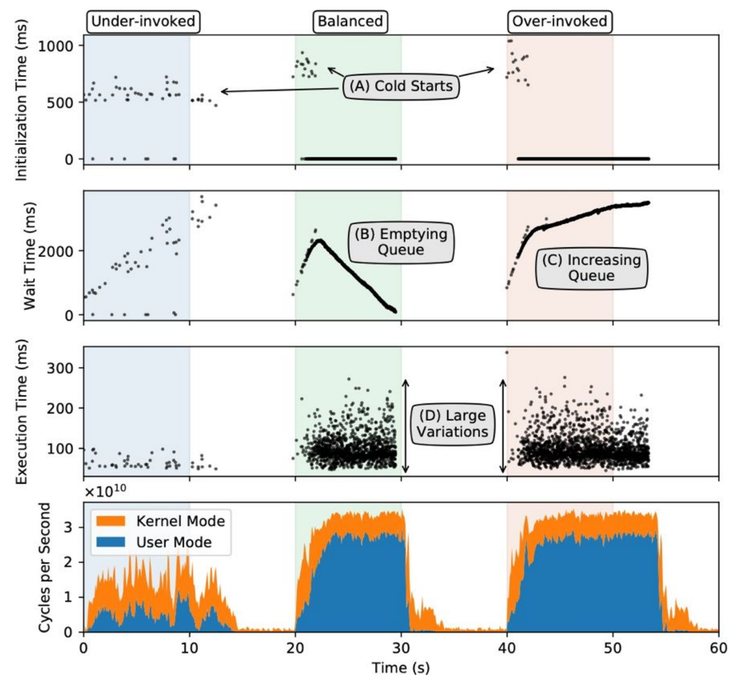


Source: https://medium.com/openwhisk/squeezing-the-milliseconds-how-to-make-serverless-platforms-blazing-fast-aea0e9951bd0

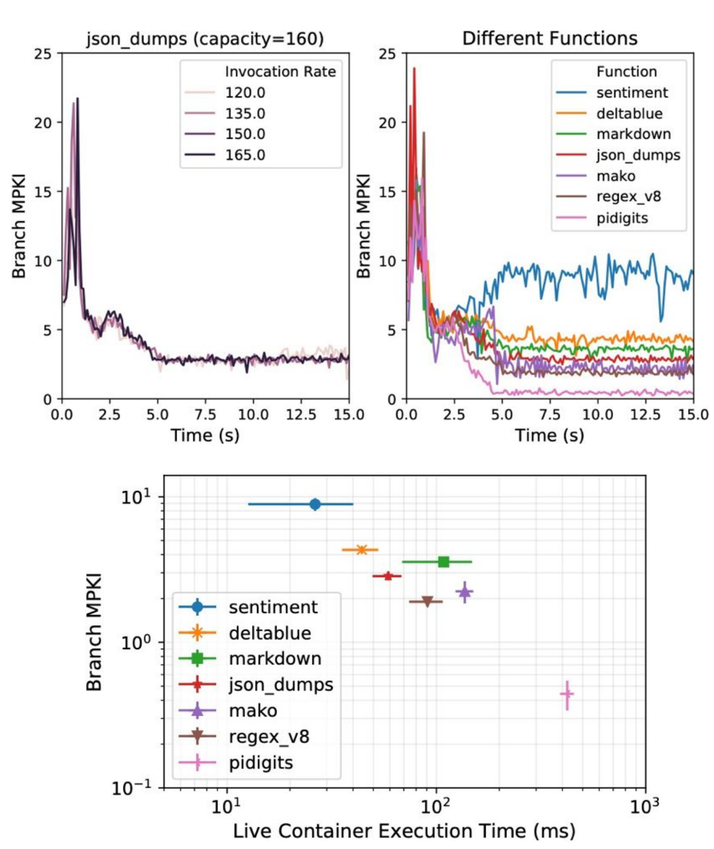
Latency Modes



Performance Breakdown

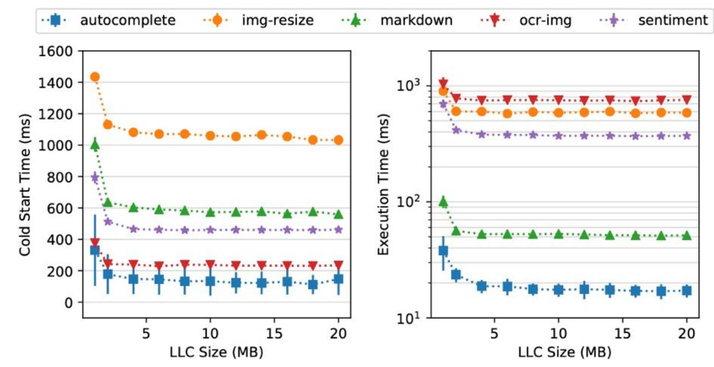


Branch Prediction



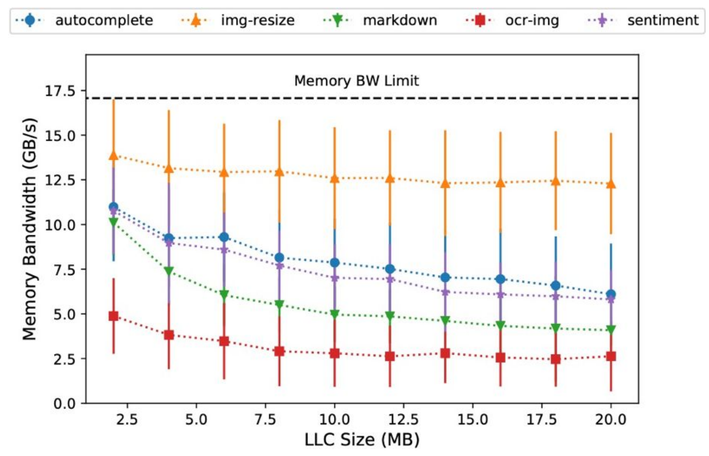
Shortest function has 20x branch MPKI of longest

Last-Level Cache (LLC)

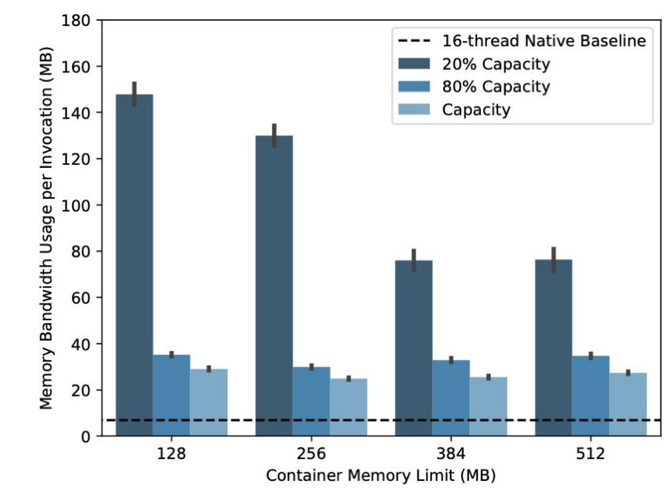


We observed a low LLC requirement for both the platform and our tested applications.

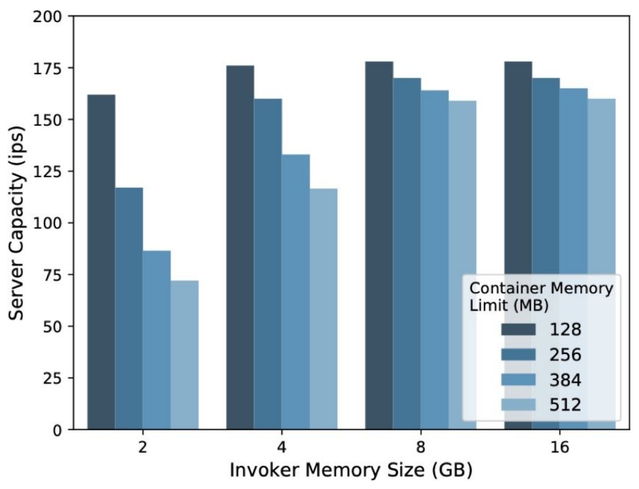
Memory Bandwidth



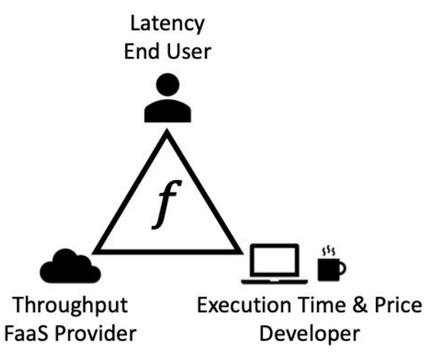
Functions' memory bandwidth demands vary.



Concurrency & Server Capacity



The FaaS Demand Triangle



There is a tension between server capacity, function execution times, and latency.

Paper PDF

MICRO 2019
October 12-16, 2019
Columbus, Ohio, USA

Project Website