Impact of Microarchitectural State Reuse on Serverless Functions

Truls Asheim, Tanvir Ahmed Khan, Baris Kasicki and Rakesh Kumar

NTNU

UNIVERSITY OF MICHIGAN

Microarcitectural state

- State of in-core performance enhancing structures
  - Branch Target Buffer (BTB)
  - Icache
- Crucial for processor performance
- Need temporal locality to work effectively

Serverless function characteristics

- Short running (often < 1s, many < 100 ms) [ATC’20, 1]
- Possibly infrequent invocations
  - Providers need to interleave the execution of different functions on the same processor core
- This reduces temporal locality

[1] Datalog. 2021. The state of serverless. https://www.datadoghq.com/state-of-serverless-2021/. (2021).

Problem: Interleaved execution thrashes (i.e. overwrites) microarchitectural state [ISCA'22]

Invocation sequence example: AAABBABAB

Cold invocation

Warm invocation

A and B: Two functions executing on the same processor core

Consequence: Performance of serverless functions is adversely affected by microarchitectural state thrashing [ISCA’22]

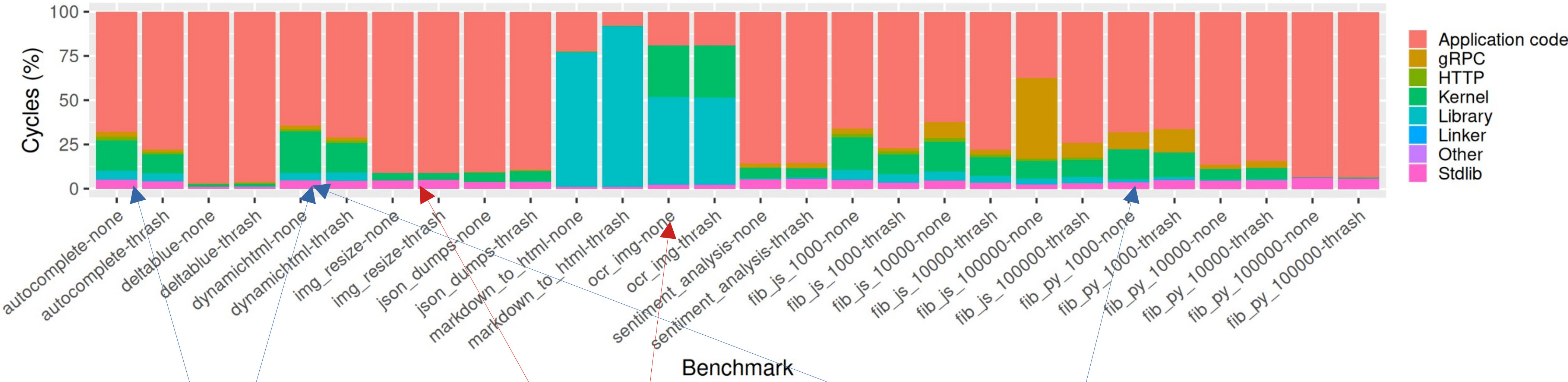
Question 1: Which properties of serverless functions make them vulnerable to performance degradation from microarchitectural state thrashing?

Question 2: What is the performance improvement opportunity of serverless-targeted microarchitectural optimizations?

Experimental setup

- Representative and synthetic functions (NodeJS and Python)
- Two modes: Interleaved and back-to-back
  - Interleaved execution simulated by a executing a microarchitectural state thrashing function after each function invocation

Where is time spent?



Short-running functions are generally more affected by interleaved executions than long-running functions

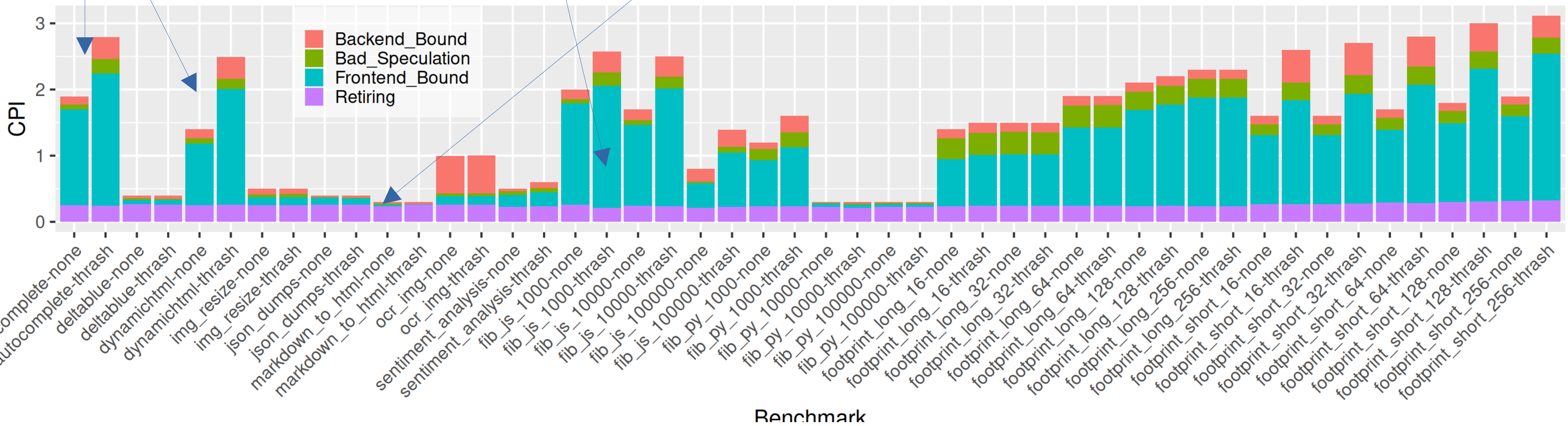
Functions with similar execution time are affected differently by state thrashing

Top-Down bottleneck analysis

Short running functions (< 1 ms): Lower performance and more vulnerable to interleaving

Short running NodeJS functions are heavily front-end bound

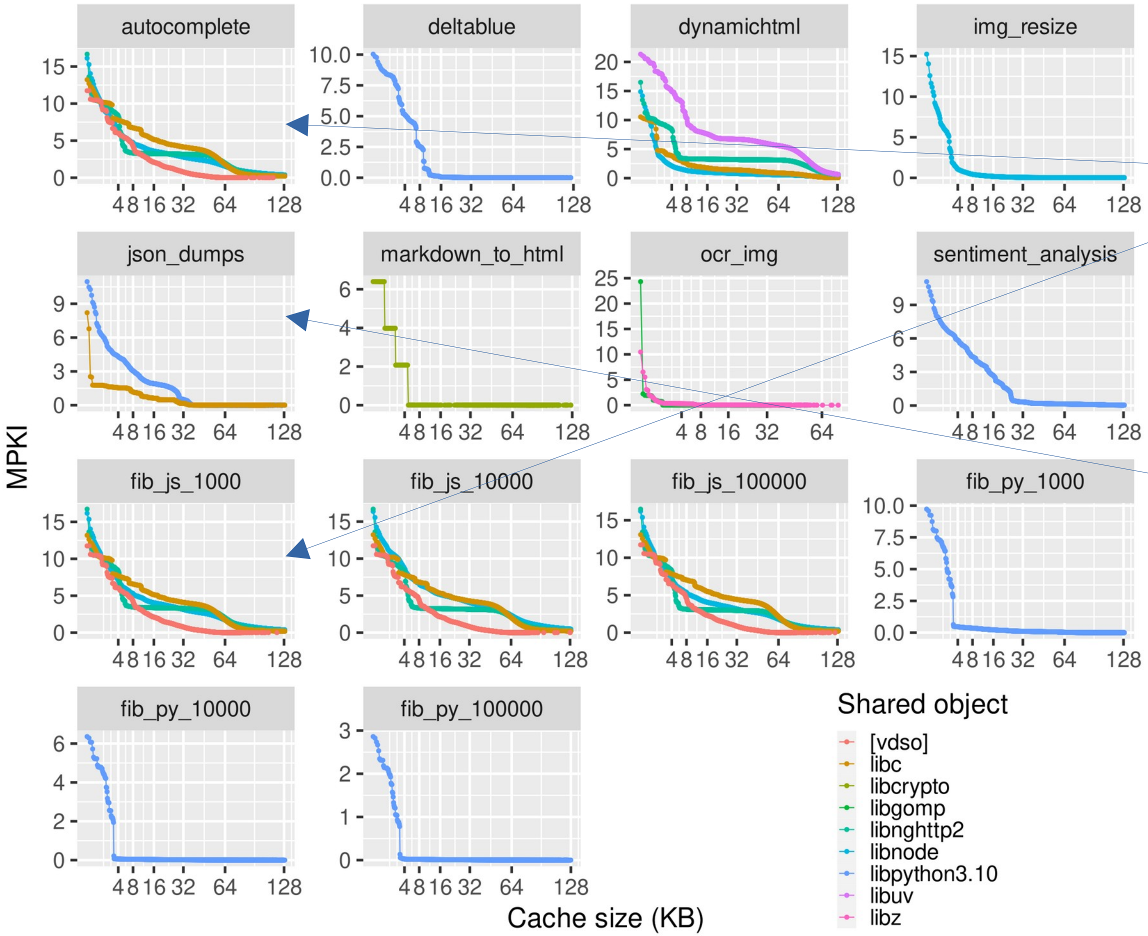
Longer-running functions (> 10 ms): Better performance and no vulnerability to interleaving



Hypothesis

- Sensitivity to state thrashing depends on
  - Function execution time
  - Function implementation language
- Heavily front-end bound
  - Prior work suggests this is observed in functions with a large code footprint [ASPLOS’18, HPCA’17]

Code footprint



NodeJS: Bigger code footprint

Python: Smaller code footprint

Confirms hypothesis about the impact of function code footprint

Conclusions

- Microarchitectural structures warm up quickly
- Only certain functions benefit from warm microarchitectural states
  - Functions with short runtimes (< 1ms) and
  - Functions with large code footprints
- Such functions are quite uncommon

Questions?