Putting the “Micro” Back in Microservices

Sol Boucher[†], Anuj Kalia[†], David G. Andersen[†], Michael Kaminsky[‡]
[†]**Carnegie Mellon University,** [‡]**Intel Labs**

Problem

- **Current serverless platforms exhibit millisecond-scale invocation latencies.**
- **State-of-the-art networks/systems boast *micro*second-scale latencies.**

Goals

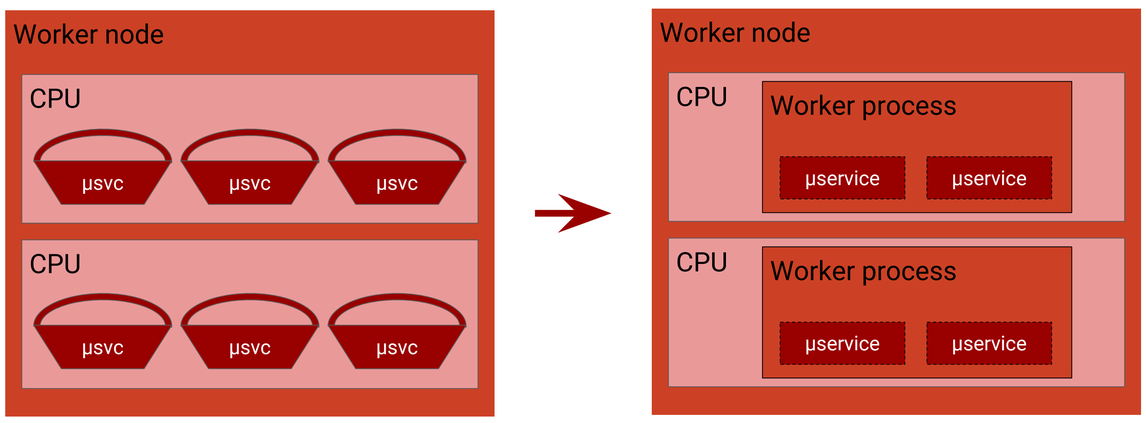
- **Reimagine serverless: shorter, faster jobs.**
- **Enable new use cases.**

Our work

- **Redesign worker node isolation mechanism to reduce invocation latency.**

Shared worker processes

- Eschew per-microservice containers
- Substitute per-core shared worker processes
  - Invoke by *polling* for requests

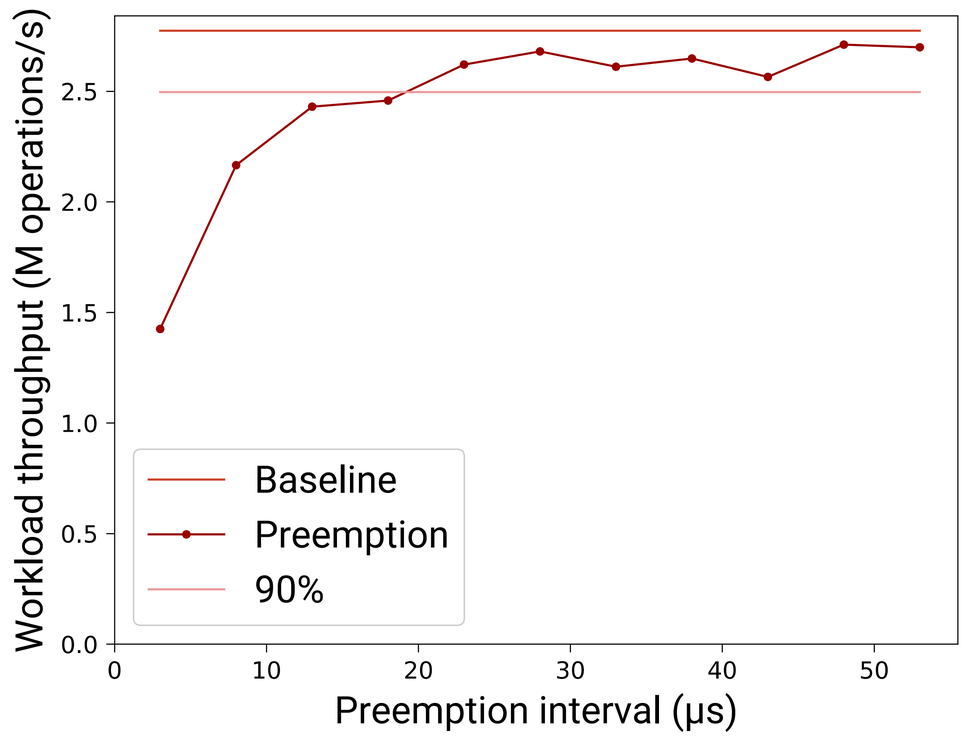


Language-based isolation

- User submits memory-safe Rust code
  - No dereferencing null/dangling pointers
  - All variables initialized
  - Immutable data unchanged
- Provider only permits memory-safe or explicitly trusted dependencies

Fine-grained preemption

1. Regain CPU control from long microservice
   - POSIX signal: 20-µs period!
2. Abort/clean up after microservice’s code
   - Throw Rust exception, catch in worker loop



Invocation latency reduction