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 microsecond-scale latencies.

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

OUR WORK
- Redesign worker node isolation mechanism to reduce invocation latency.

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

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

Invocation latency reduction

<table>
<thead>
<tr>
<th>Warm start</th>
<th>50%</th>
<th>99%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.7</td>
<td>1.2</td>
</tr>
<tr>
<td>Cold start</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>2800</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>1500</td>
</tr>
</tbody>
</table>

Invocation latency (μs)
- Process-based isolation
- Language-based isolation