8th International Workshop on Serverless Computing (WoSC8) 2022

https://www.serverlesscomputing.org/wosc8/papers/p3

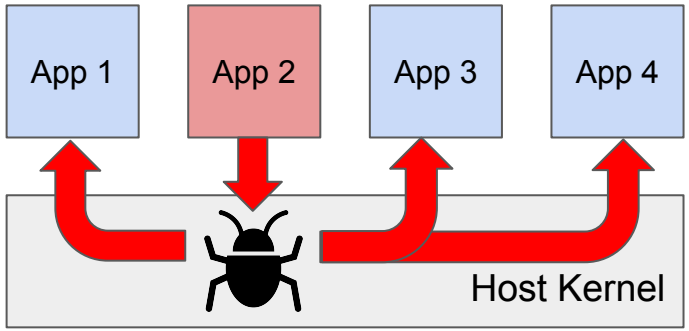
Sentinel: A Fast and Memory-Efficient Serverless Architecture for Lightweight Applications

Joe Hattori, Shinpei Kato
The University of Tokyo

Background: Security Concern in Serverless Computing

From cloud provider's point of view…

- Executes many applications on the same machine
- Vulnerability in host kernel can be fatal without secure execution environment



Goals

- Execute each application **securely**
- Spawn the environment **quickly**

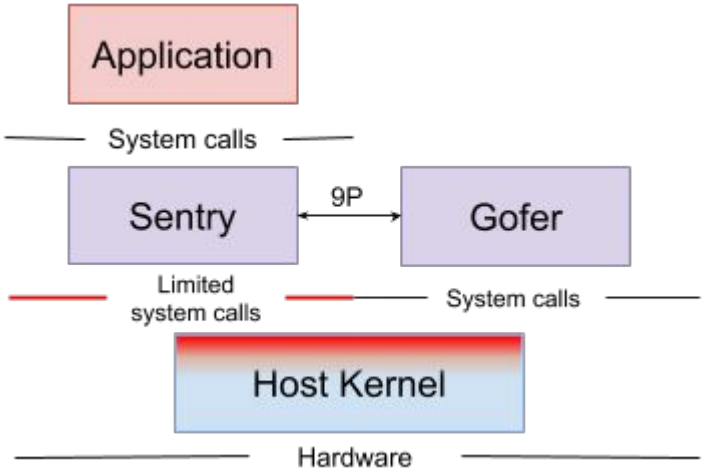
Existing Architecture: gVisor

Container-based architecture

Traps every syscall and emulate in userspace

Main components

- Sentry: Untrusted userspace kernel
- Gofer: A process handling filesystem operations

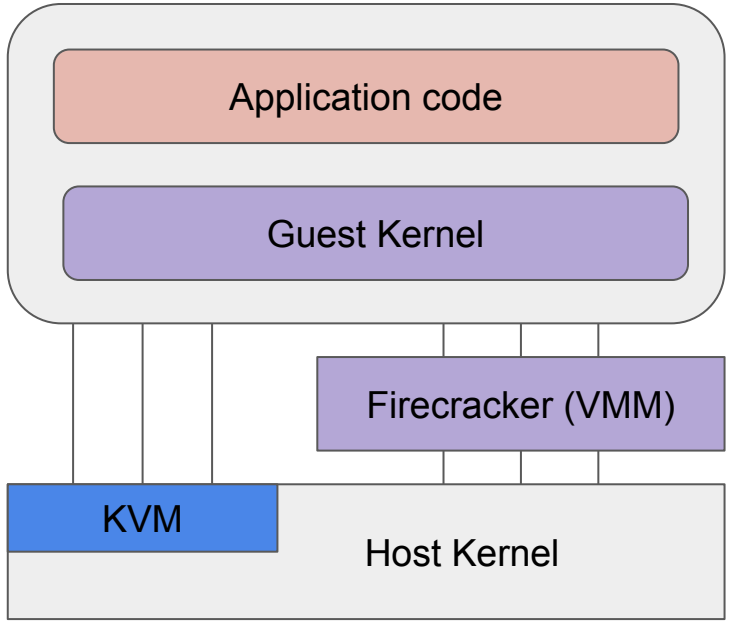


https://gvisor.dev/docs/

Existing Architecture: Firecracker

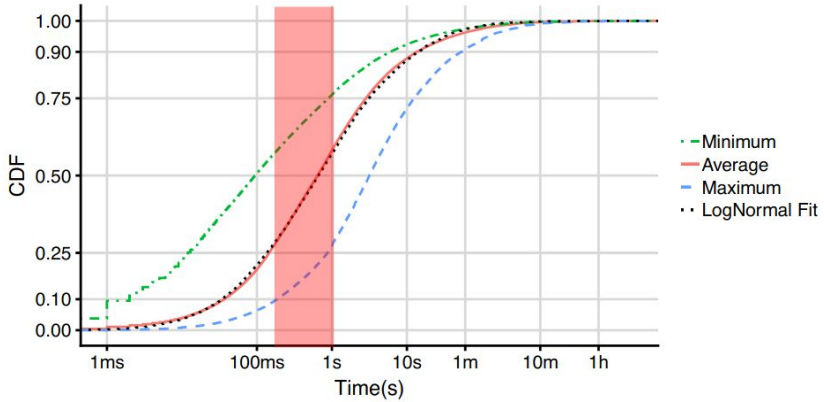
VM-based architecture

Shortens boot time with a lightweight VM specialized in serverless execution



Agache, Alexandru, et al. "Firecracker: Lightweight virtualization for serverless applications." *17th USENIX symposium on networked systems design and implementation (NSDI 20)*. 2020.

Serverless in the Wild



Shahrad, Mohammad, et al. "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider." *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. 2020.

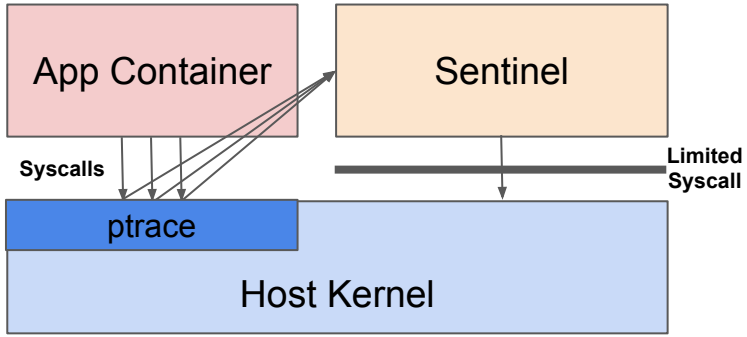
- 50% of applications run within 0.67s
- 20% of applications run within 0.10s

Cold start latency (100~500ms) is too long for most serverless applications.

Proposal: Sentinel [https://github.com/pflab-ut/sentinel]

Lightweight applications' characteristics

- Short execution time (~1s)
- Limited changes to the underlying filesystem
- Only issue simple syscalls
  - Do not use mount(2), bpf(2), etc
- Single-threaded

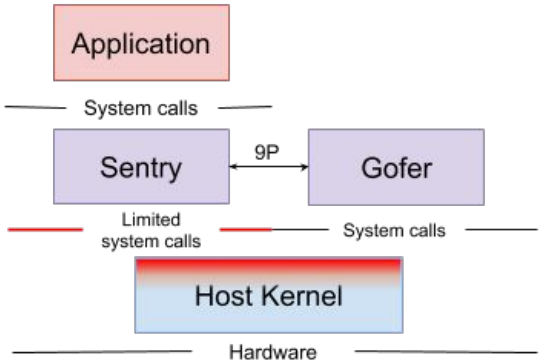


Suited approach to executing lightweight applications

- **Secure** container runtime: shorter spin up time than VM boot time
- **Quick** execution of **bare-minimum syscall virtualization**
  - by ptrace(2) with PTRACE_SYSEMU flag

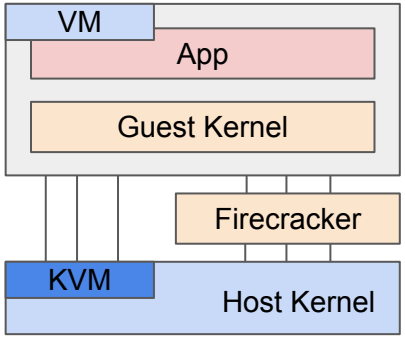
Design Comparison

gVisor (container)



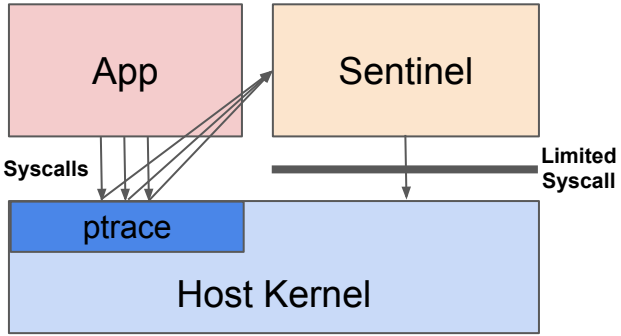
IPC over 9P protocol happens upon every file access

Firecracker (VM)



Large memory consumption by VM creation and execution

Sentinel (container)



RO mount to host filesystem and retain changes in-memory.

Minimized effect on host kernel when compromised.

Virtualize each app with one process.

Evaluation

Targets

- runc: Docker’s default container runtime (**No virtualization**)
  - Sentinel’s goal is to perform as close as possible to runc.
- runsc: gVisor’s container runtime (**container-based**)
- Firecracker: VMM (**VM-based**)
- kata-runtime: Kata Container’s container runtime (**VM-based**)

Environment

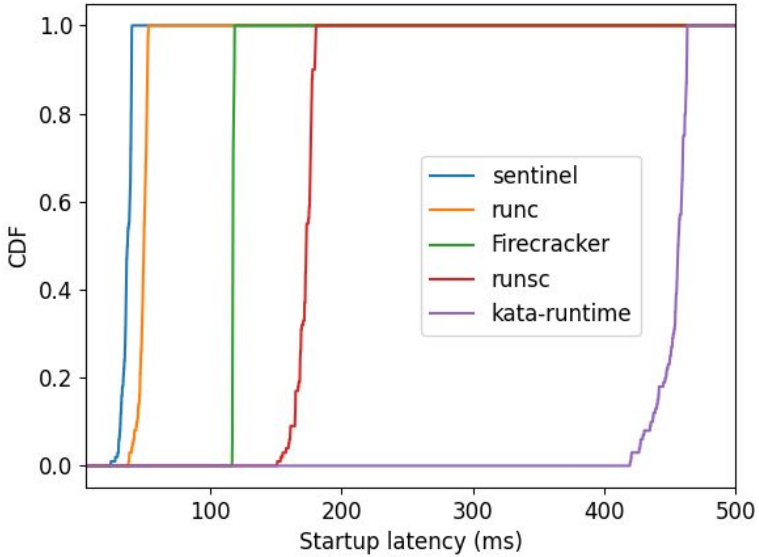
- Ubuntu 20.04 (Linux kernel v5.14)
- 4 core: Intel Core i7 3.0 GHz

Evaluation: Sandbox Startup Time

Sentinel is the fastest

- Faster than runc!
  - Performance benefit of Rust
- Lower tail latency

~10x faster startup



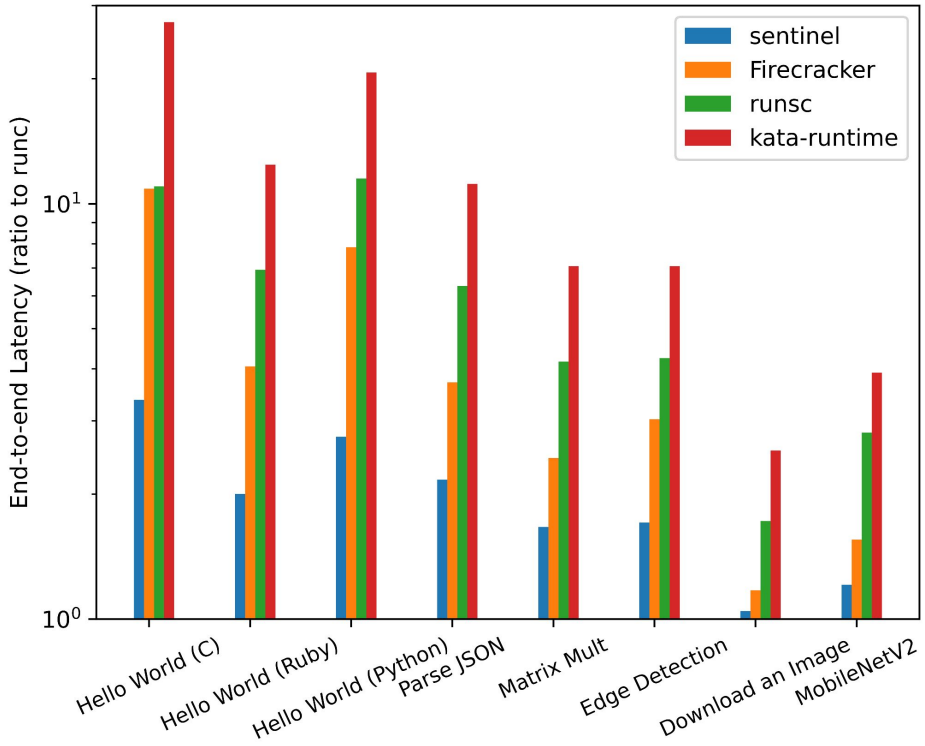
Evaluation: End-to-End Latencies

Benchmark applications:

- Hello World - C, Ruby, Python
- JSON Parser - Ruby
- Matrix multiplication - Python
- Edge detection - OpenCV
- Downloading an image - Python
- MobilenetV2 inference - TensorFlowLite

Sentinel's speed is the closest to runc's

~8.13x shorter execution time



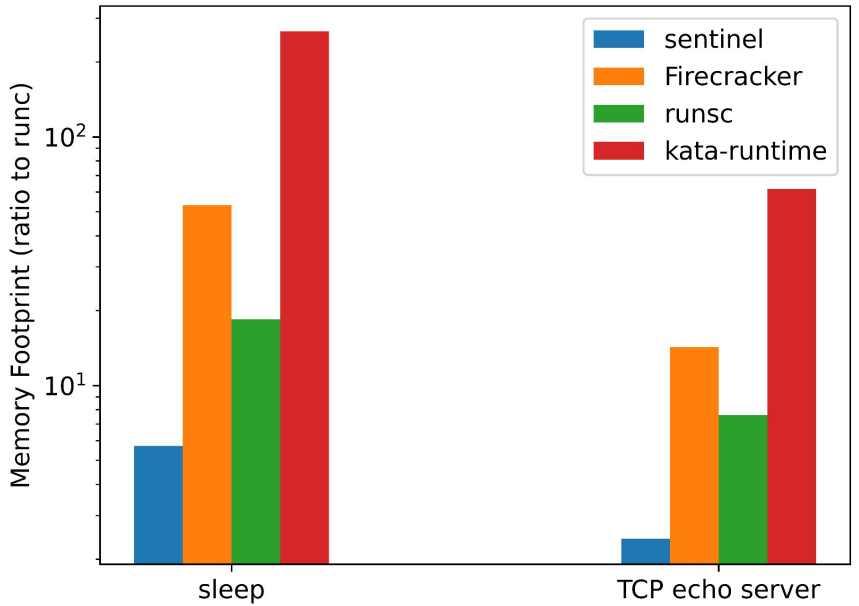
Evaluation: Memory Efficiency

Target applications:

- sleep command
- TCP echo server

Sentinel's performance:

- vs runsc: ~69% lower
- vs Firecracker: ~89% lower
- vs kata-runtime: **~98%** lower



Summary

- Proposed Sentinel, a serverless architecture for lightweight applications
- Demonstrated the performance benefits compared to existing architectures
  - Quicker boot, faster execution, lower memory consumption

https://github.com/pflab-ut/sentinel

Future work

- Support for warm start
- Full OCI (Open Container Initiative) Spec compatibility
- Support environments other than x86 Linux