KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION

China 2024

NanoVisor: Revolutionizing FaaS Cold Start Performance with Secure, Lightweight Container Runtime

Tianyu Zhou

Contents

- NanoVisor Intro & Role in FaaS
- Cold Start Optimization
- Evaluation
- Summary

NanoVisor Intro & Role in FaaS

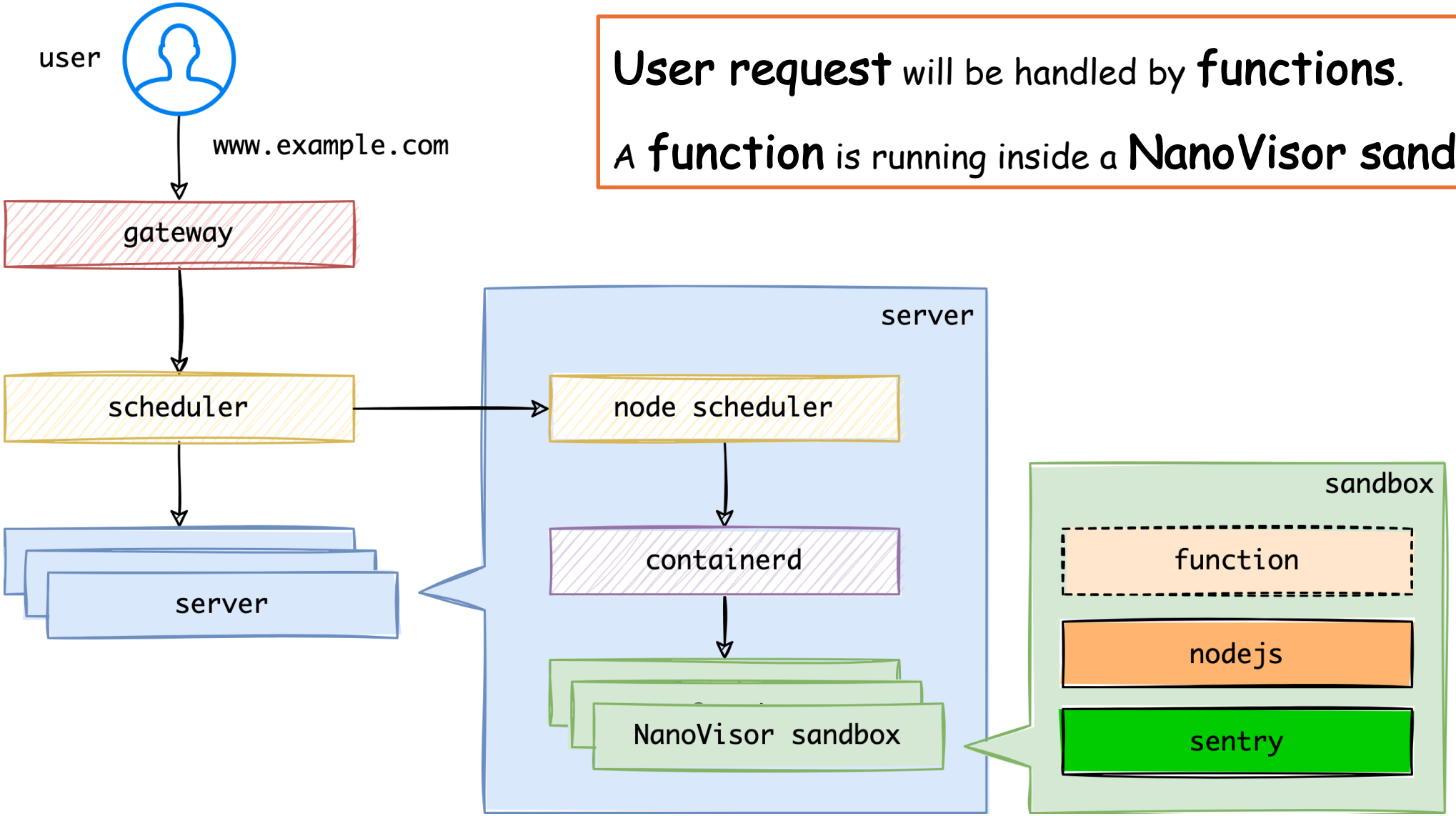
What is NanoVisor

- Lightweight 🪶 & Secure 🔰 Container Runtime
- Based on gVisor[1] 

- Extremely fast in container startup 🚀
- Production ready 😊

[1] https://gvisor.dev/

Role of NanoVisor in FaaS

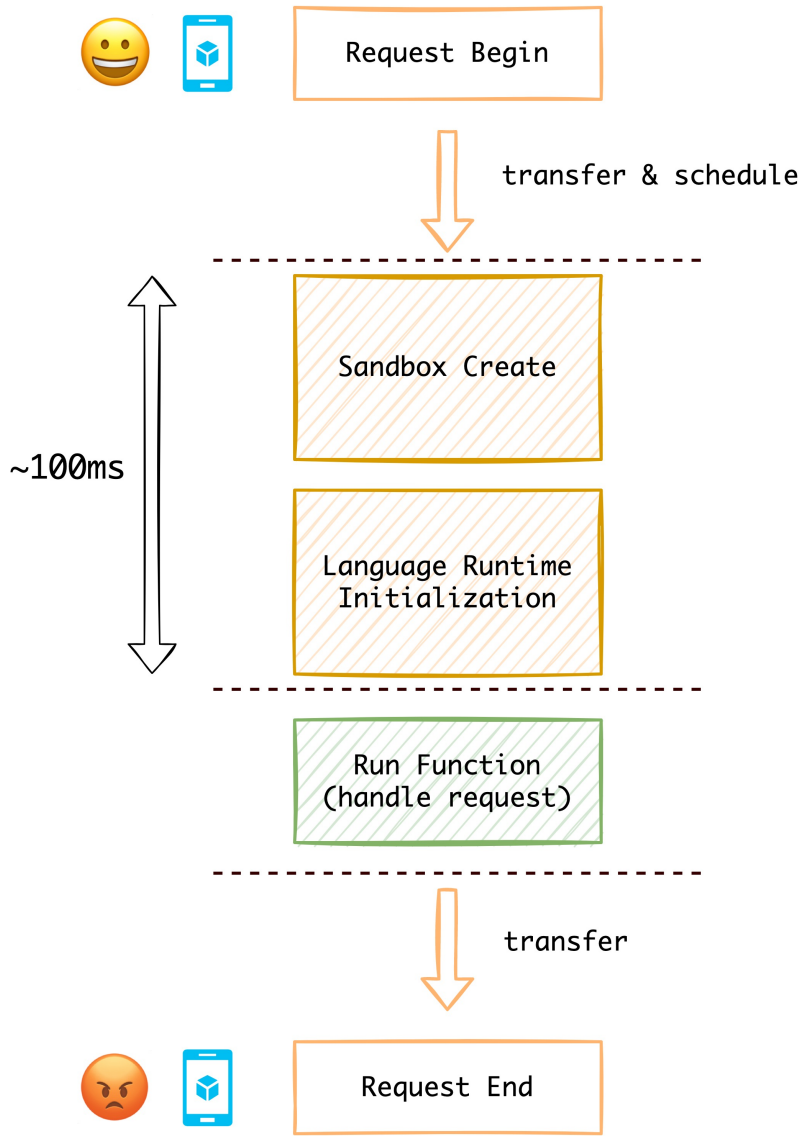


User request will be handled by **functions**.

A **function** is running inside a **NanoVisor sandbox**.

Cold Start Optimization

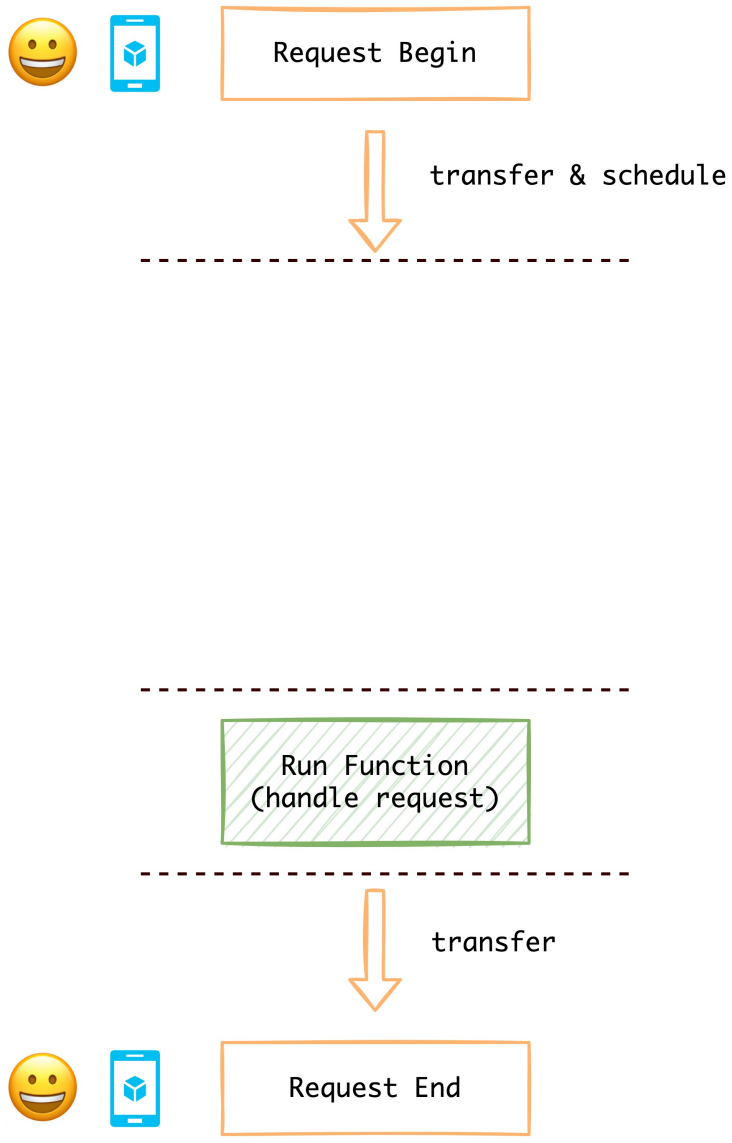
Cold Start —— critical issue in FaaS



On-demand container creation(cold start) does not meet the **latency** requirement in FaaS.

TOO SLOW! 😫

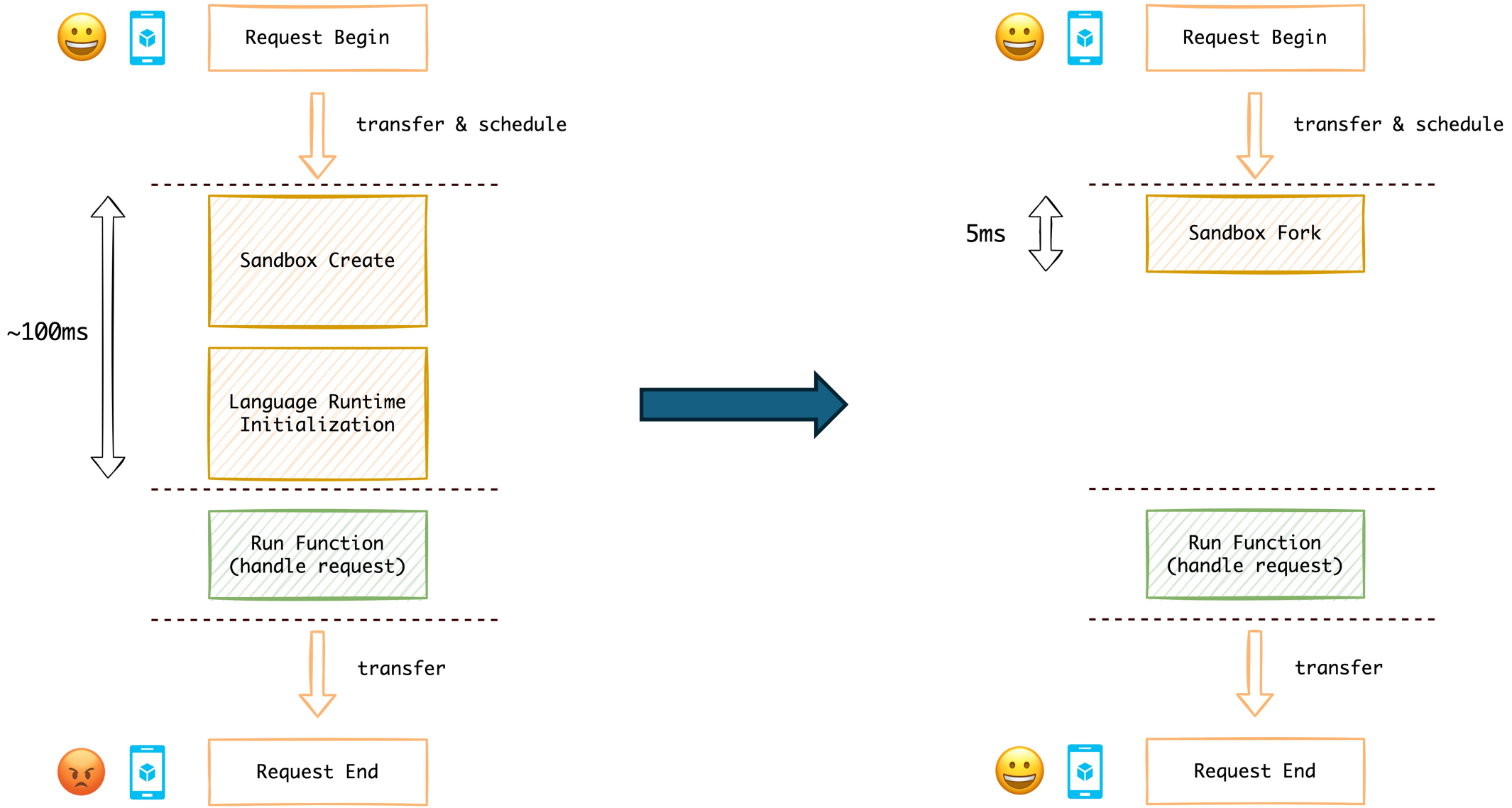
Cold Start —— critical issue in FaaS



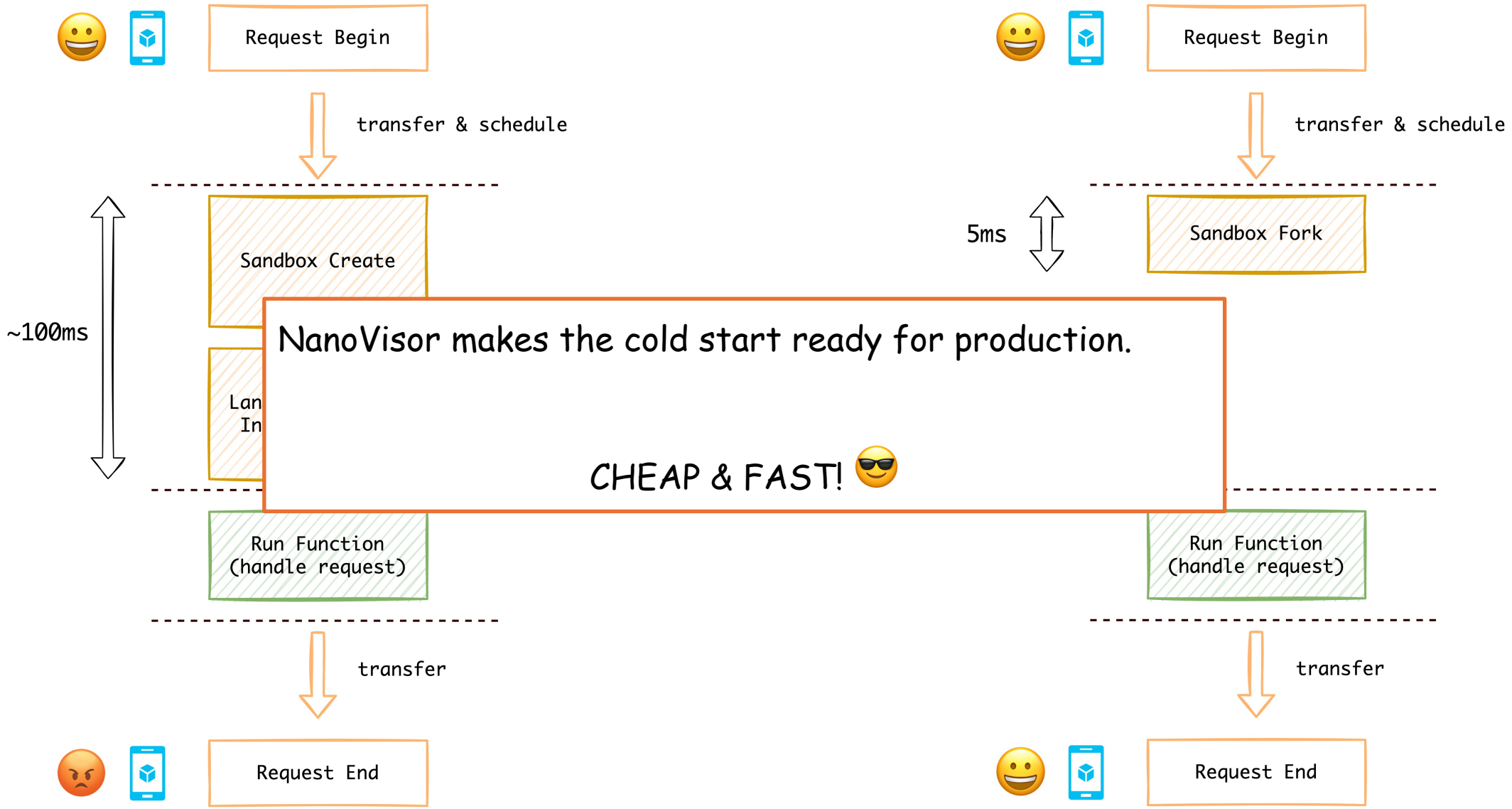
Cached container(warm start) meet the **latency** requirement in FaaS, but wastes much resource.

TOO EXPENSIVE! 😓

Make cold start extremely fast



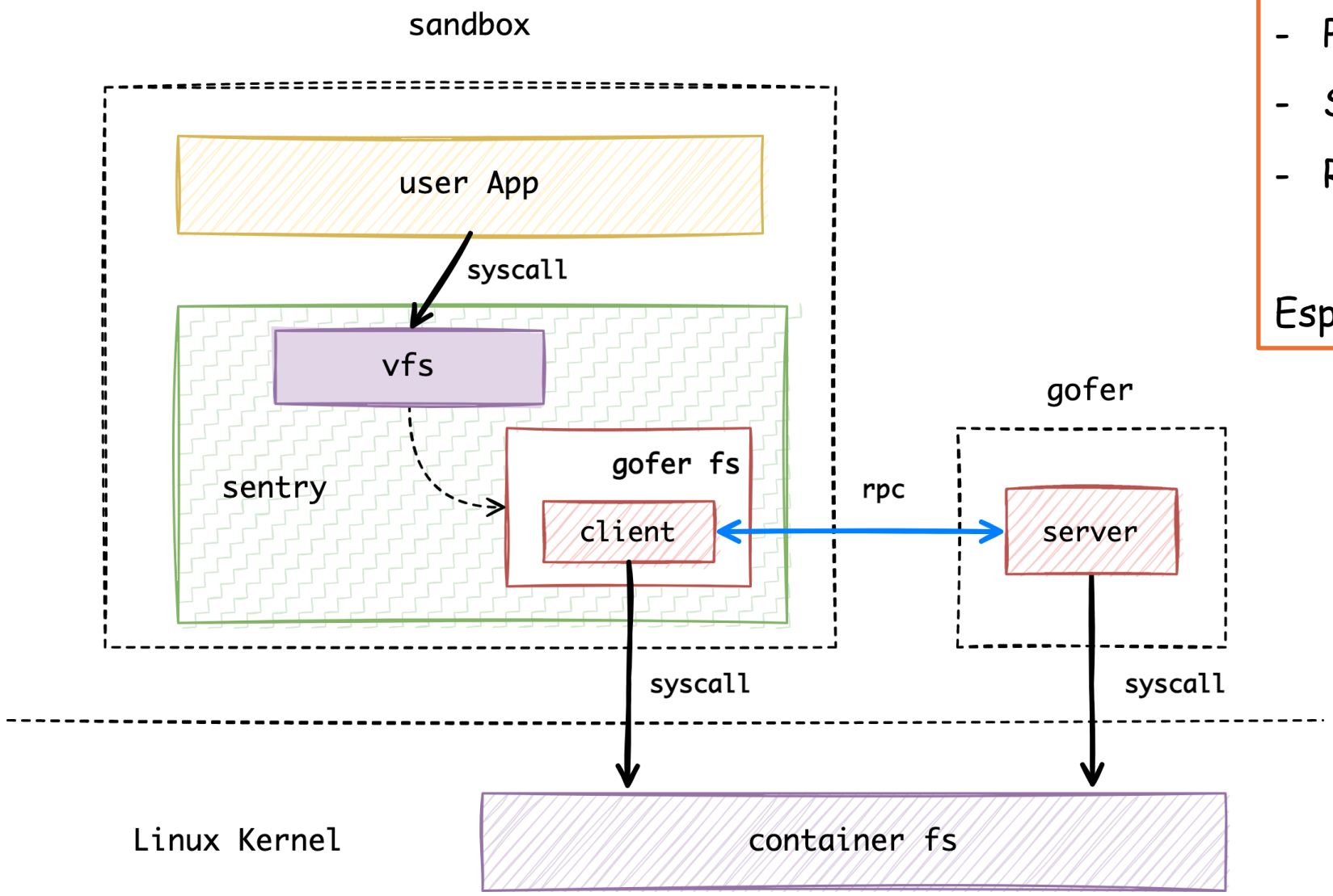
Make cold start extremely fast



How we did it?

- Single Sandbox Process
- Fast inter-component communication
- Sandbox fork

Single Sandbox Process

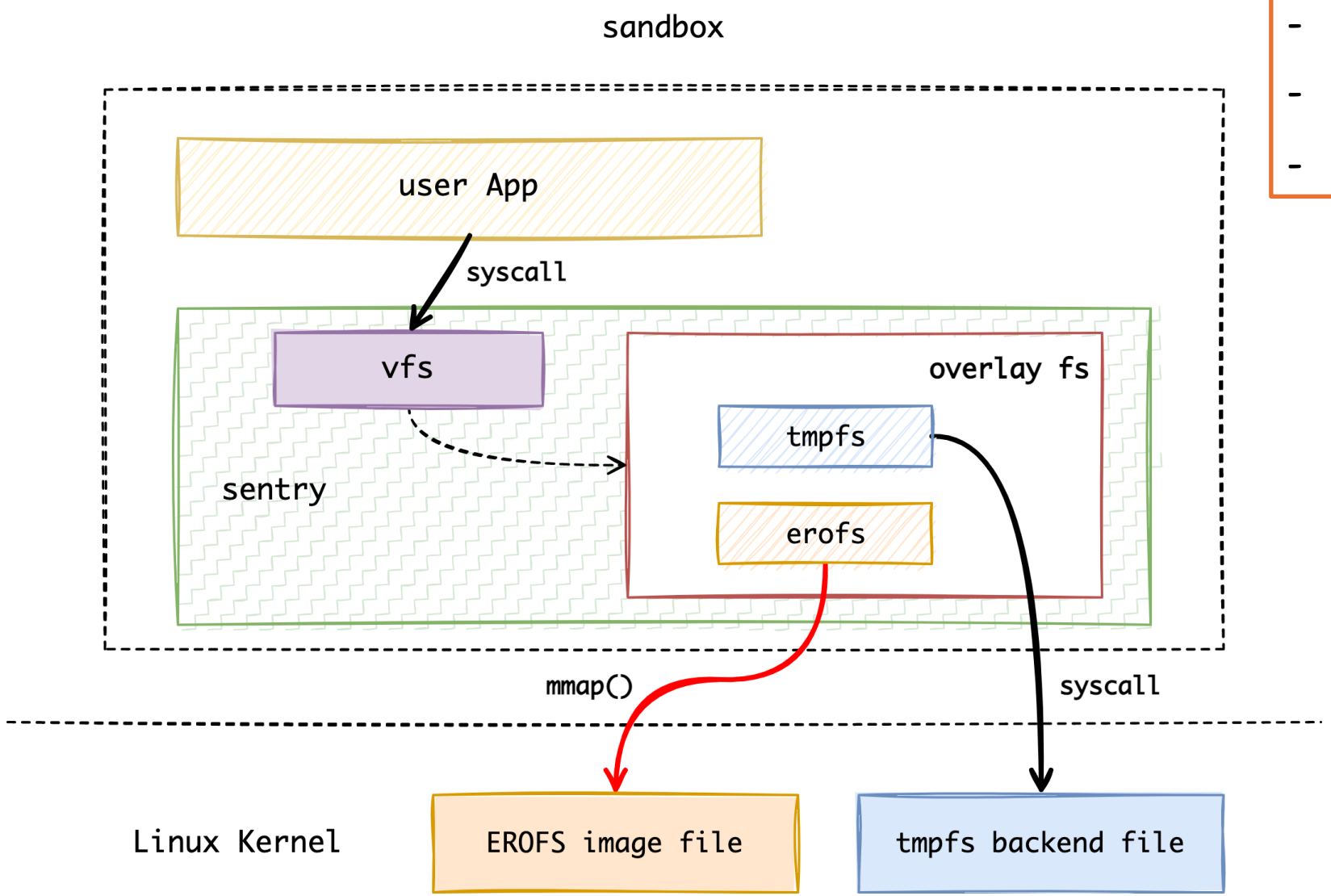


gofer is **weak** at:

- Process complexity
- Setup
- Runtime performance

Especially in FaaS!

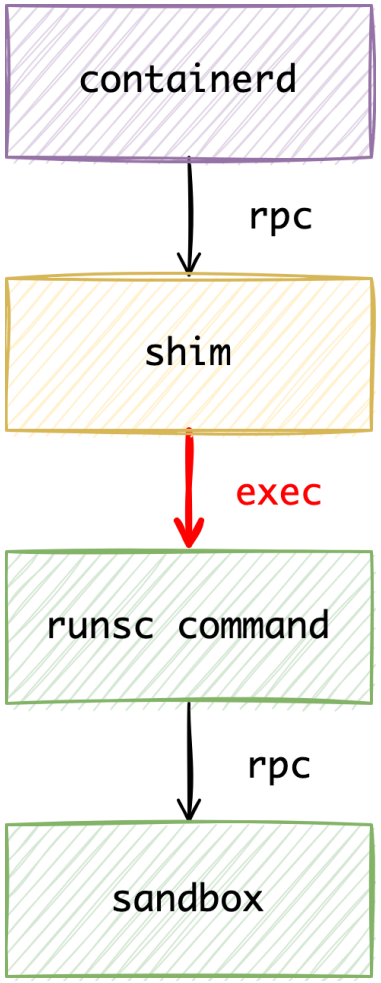
Single Sandbox Process



erofs is **Better** at:

- Single sandbox process
- Less FD (image & tmpfs file)
- Performance (no rpc)

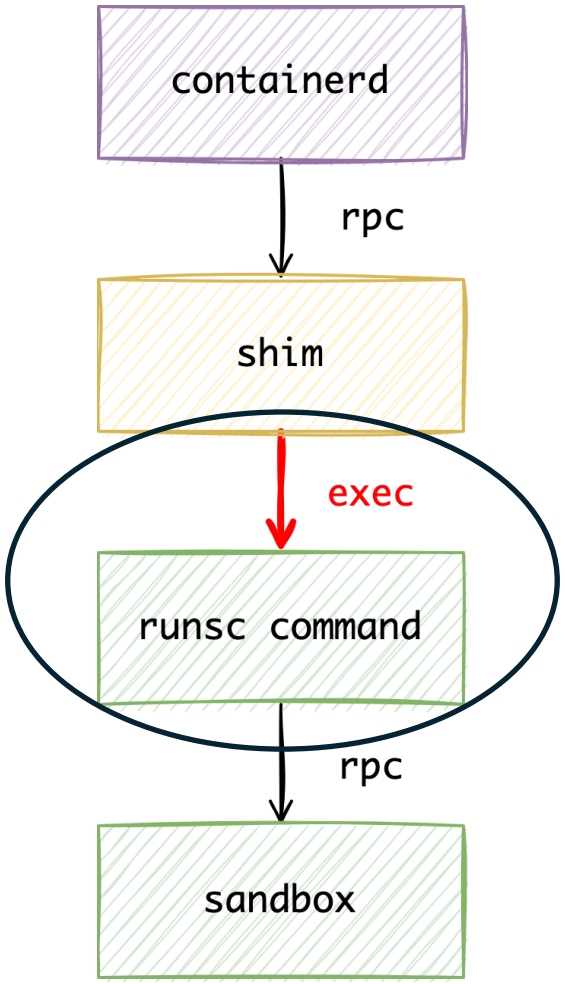
Fast inter-component communication



Command executions during runsc sandbox creation

1. containerd creates a shim process
2. shim executes runsc create to create a sandbox
3. runsc create executes runsc boot to boot a sandbox
4. runsc boot executes self to drop FDs/capabilities
5. shim executes runsc start to start a sandbox

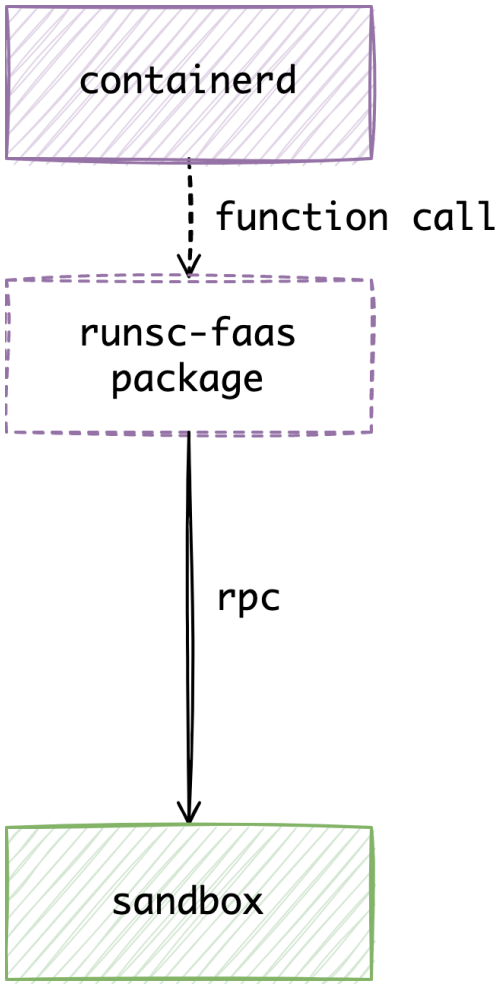
Fast inter-component communication



Command line based interaction is too slow in FaaS.

A simple **`runsc help`** could cost ~30ms.

Fast inter-component communication



- Remove shim
- Provides a Go package to import, which includes helper functions for containerd to call.

No heavy execution at all !

Sandbox fork: motivation

Operations to do in cold start

1. Sandbox process create
2. Sentry(kernel inside sandbox) initialization
3. Language runtime initialization
4. Get request, load and run function
5. Send response

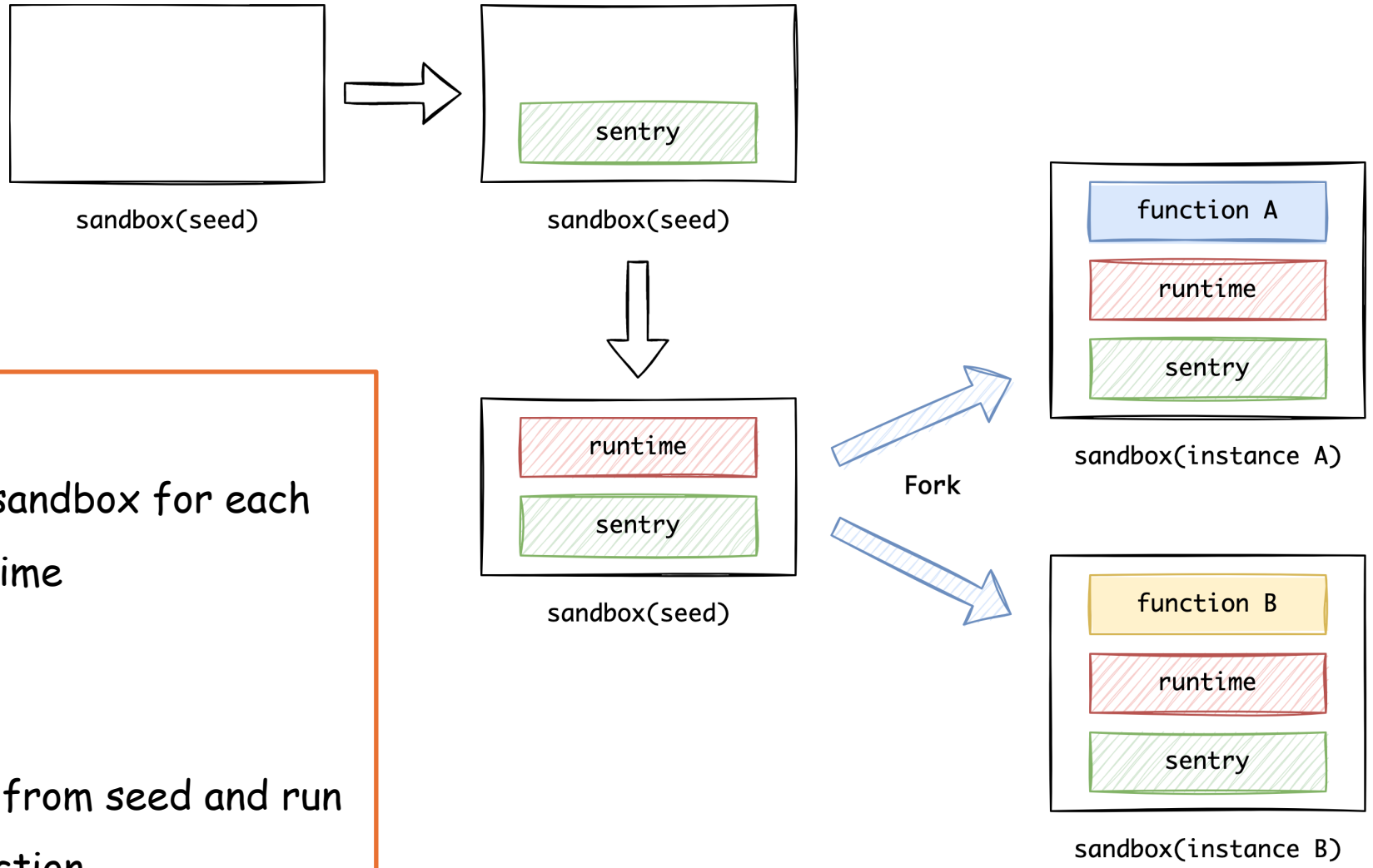
Sandbox fork: motivation

Operations to do in cold start

1. Sandbox process create
2. Sentry(kernel inside sandbox) initialization
3. Language runtime initialization
4. Get request, load and run function
5. Send response

Could be reused 🤔

Sandbox fork: overview



Prepare stage

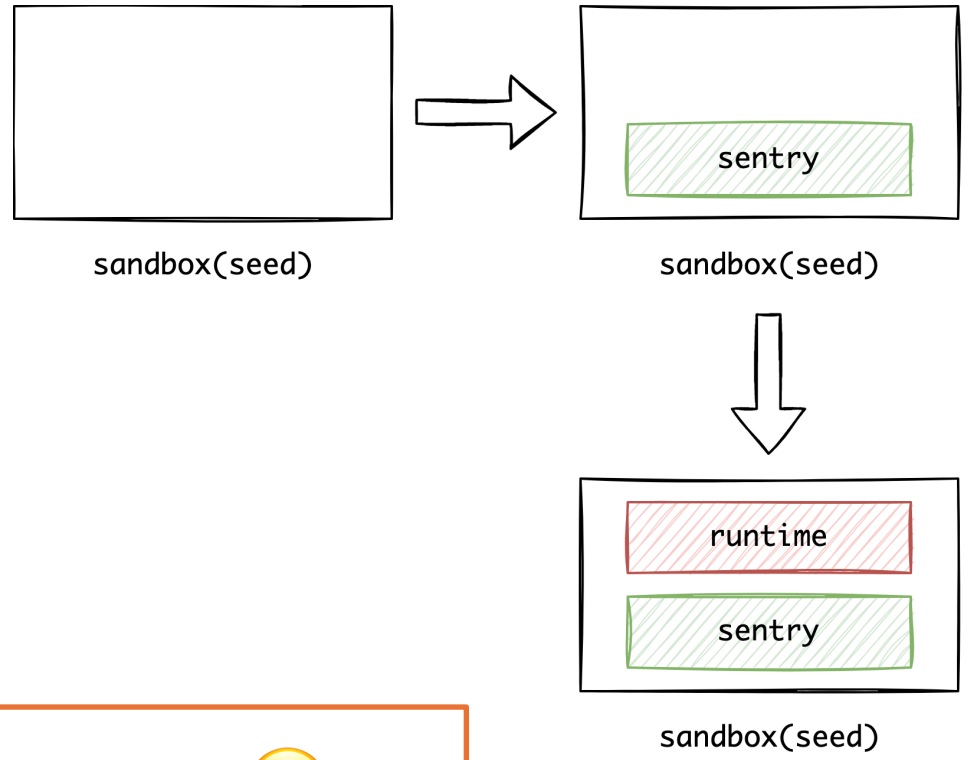
- Launch seed sandbox for each language runtime

Serving stage

- Fork sandbox from seed and run according function

Launch seed

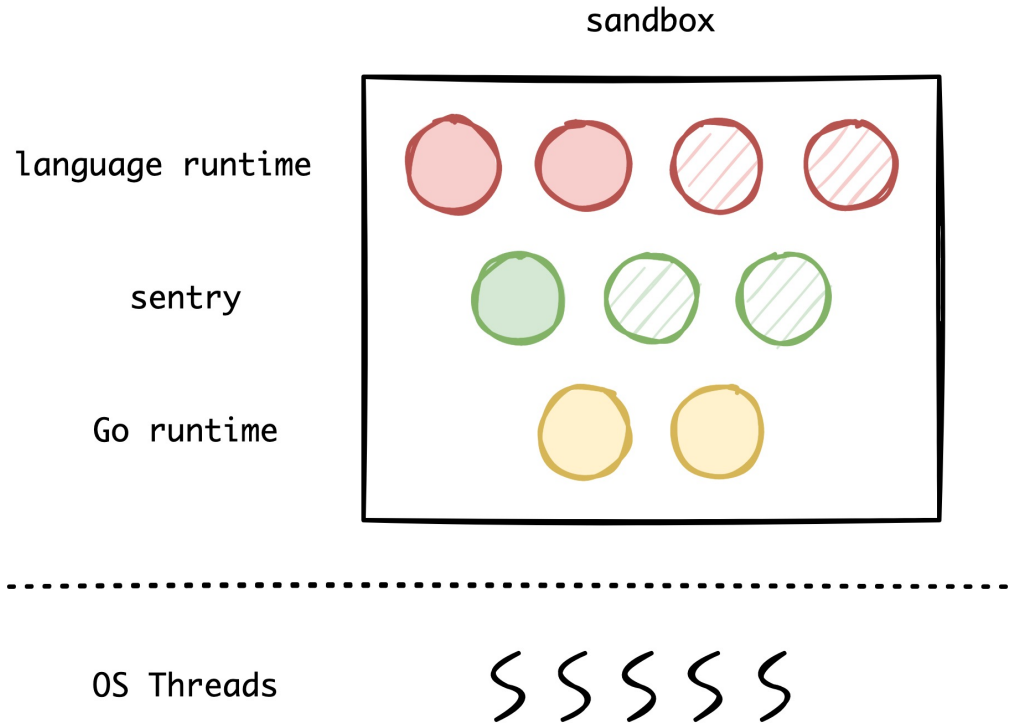
1. Sandbox process create
2. Sentry initialization
3. Language runtime initialization



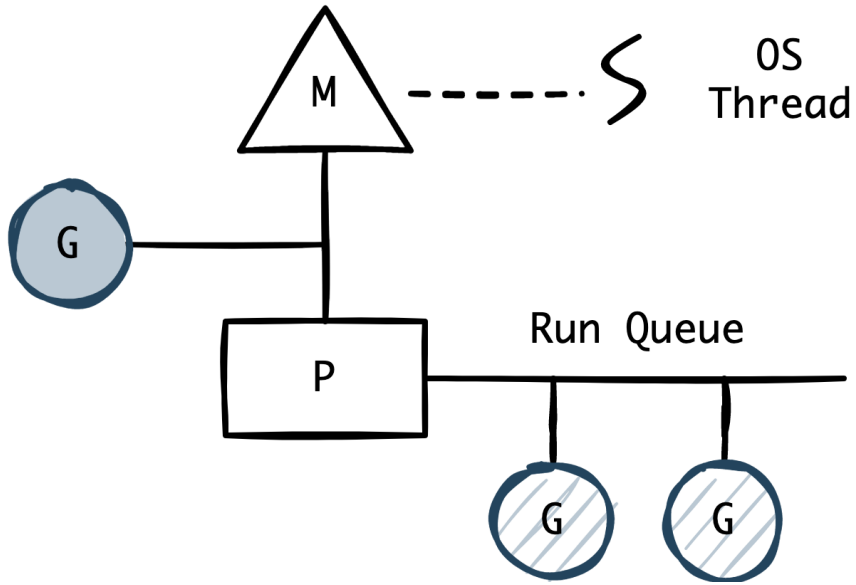
Multithread sandbox is not appropriate for fork 🤔

Launch seed

4. Simplify the sandbox



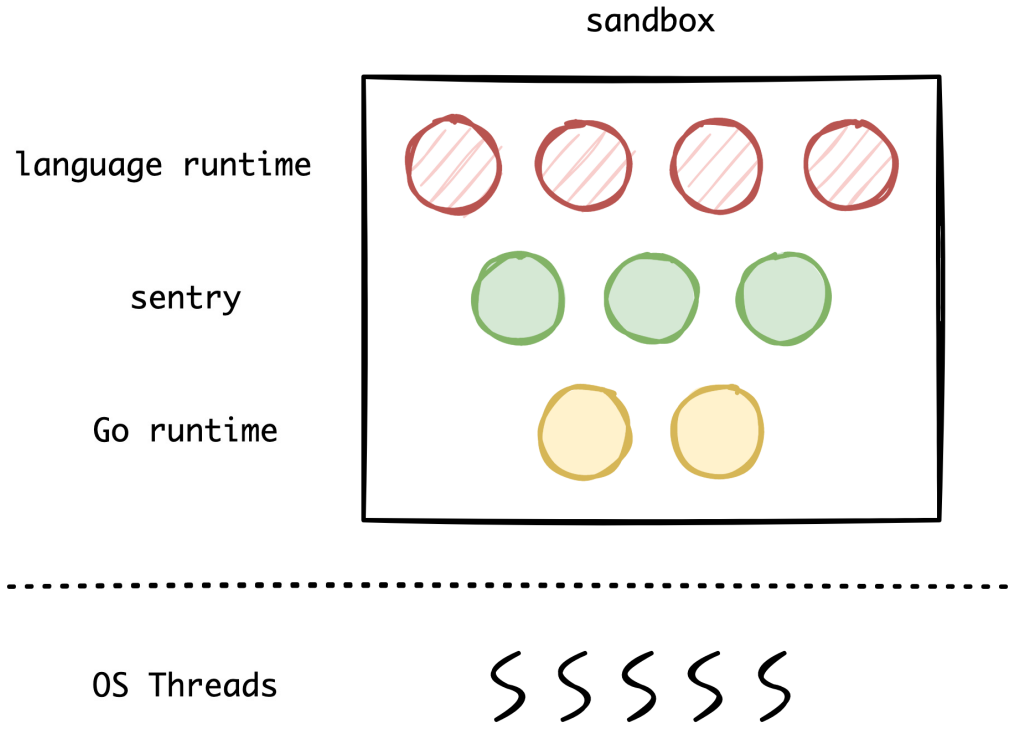
Sandbox process(Go routine view)



Go scheduler

Launch seed

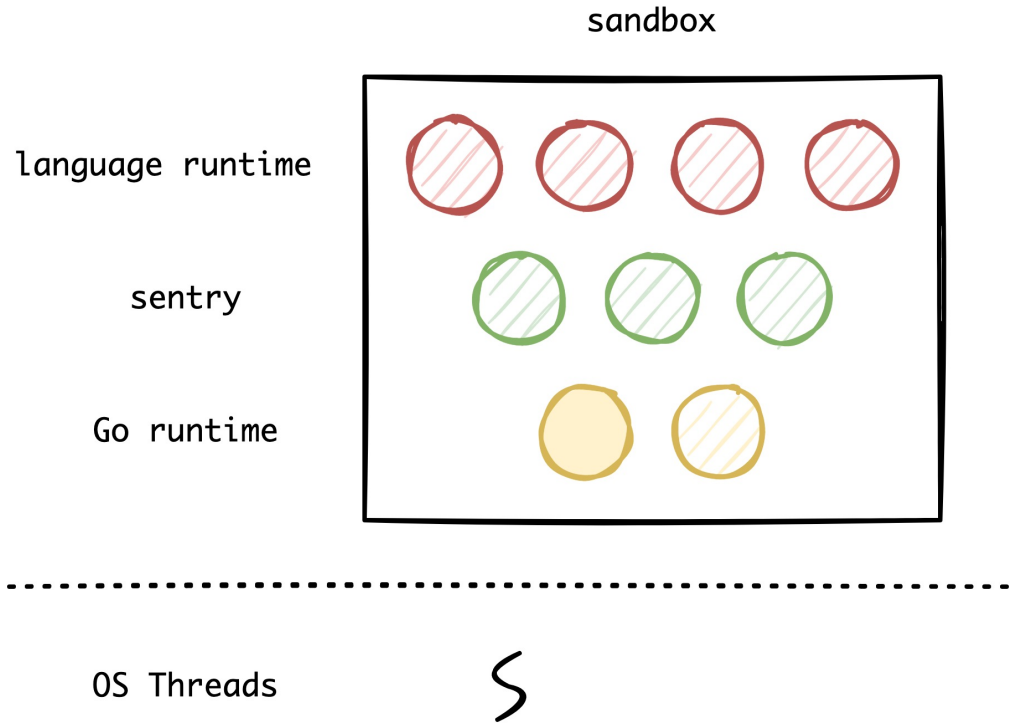
4. Simplify the sandbox



- Pause guest kernel & drop all vCPUs

Launch seed

4. Simplify the sandbox

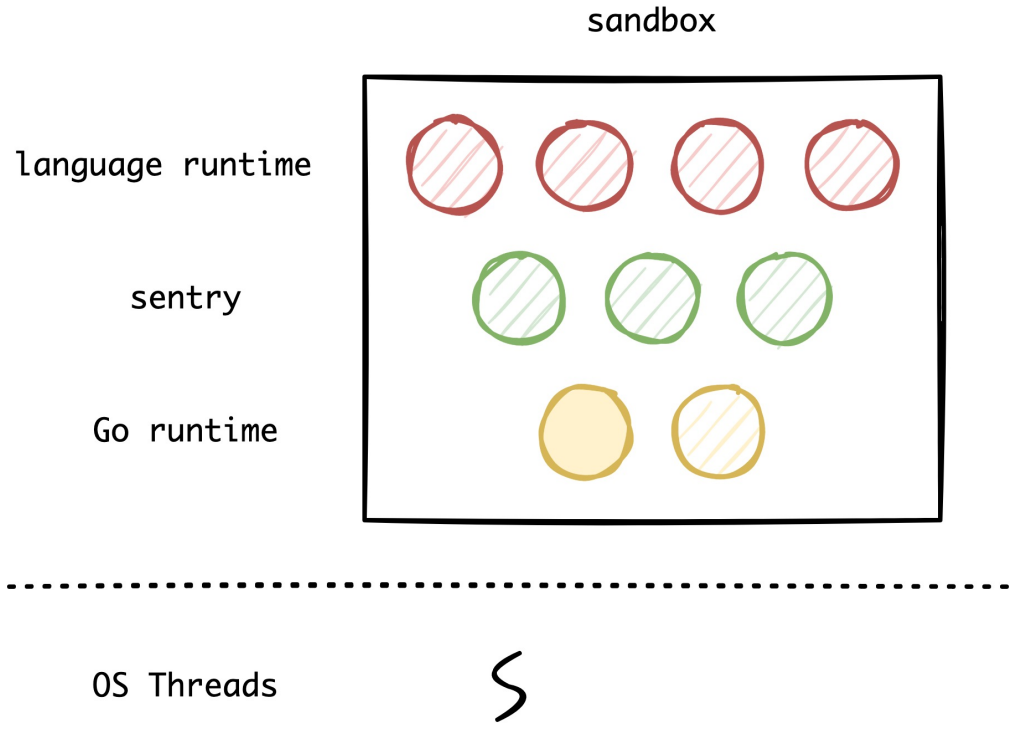


- Pause guest kernel & drop all vCPUs
- Stop the world & reclaim threads

Sandbox process(Go routine view)

Launch seed

4. Simplify the sandbox

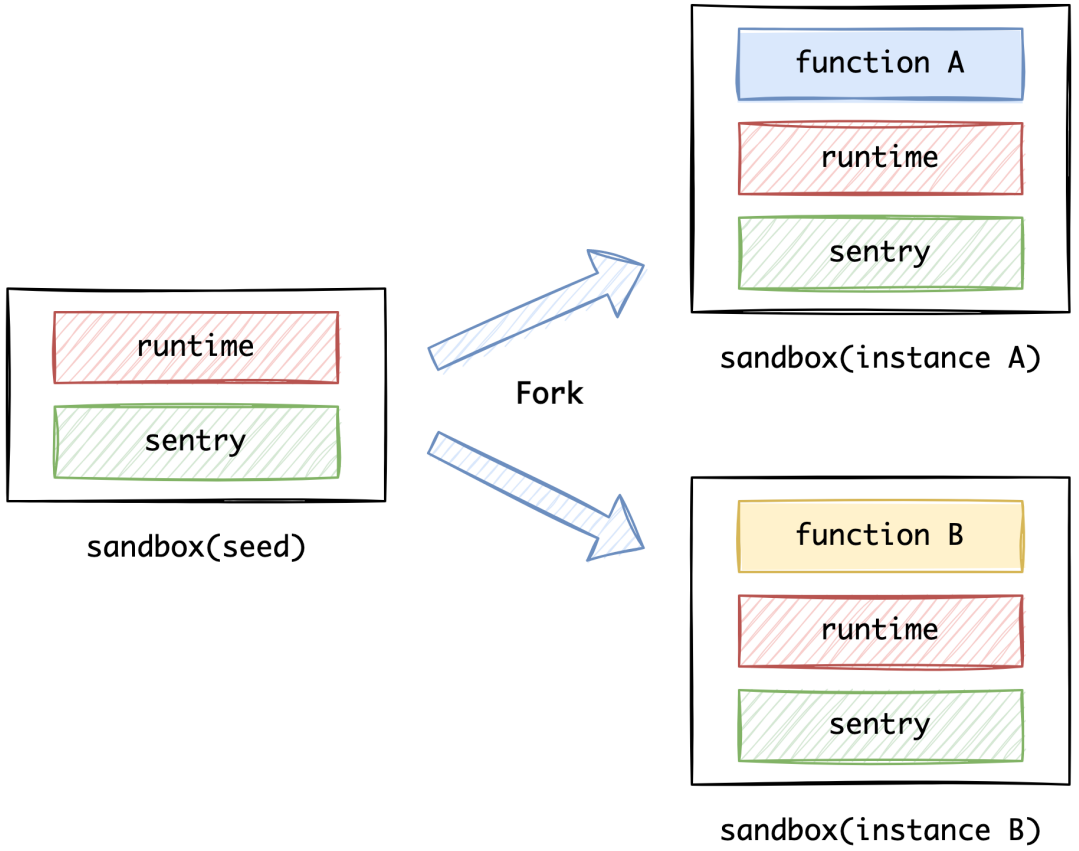


- Pause guest kernel & drop all vCPUs
- Stop the world & reclaim threads
- Invoke accept(), waiting for request

Sandbox process(Go routine view)

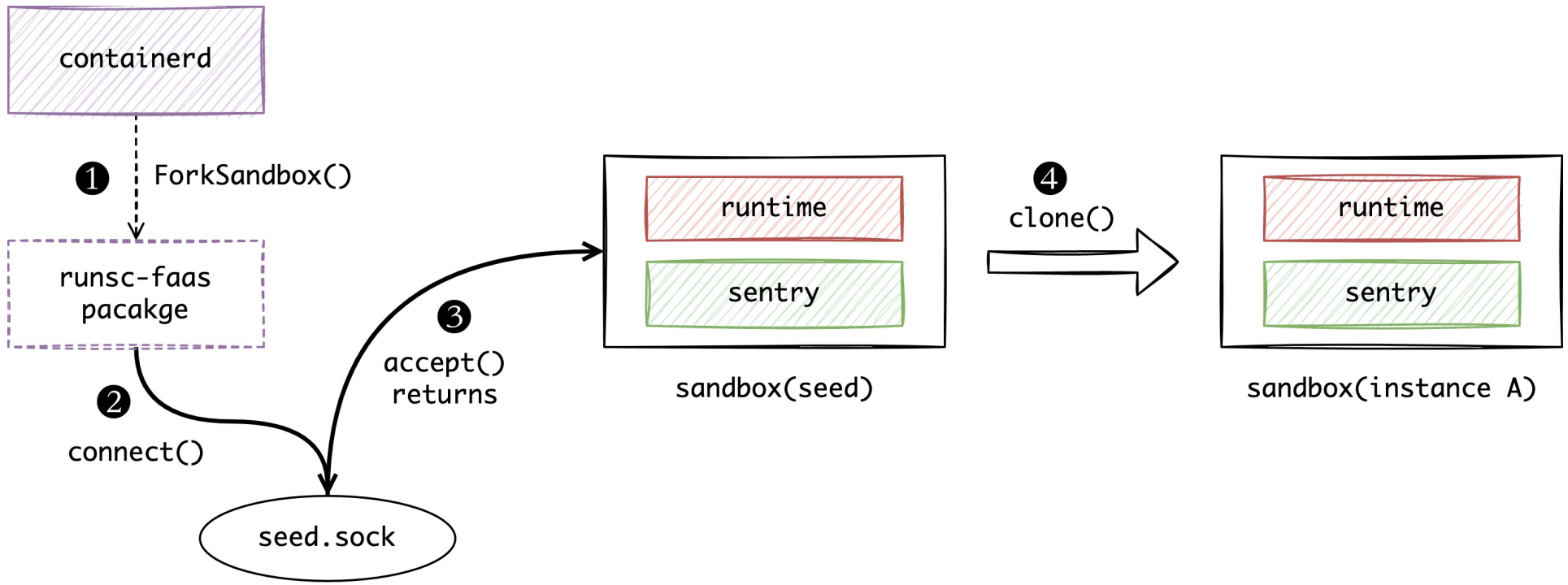
Fork sandbox

1. Connect seed to trigger fork
2. Recover Go runtime
3. Update config and mount code
4. Run function



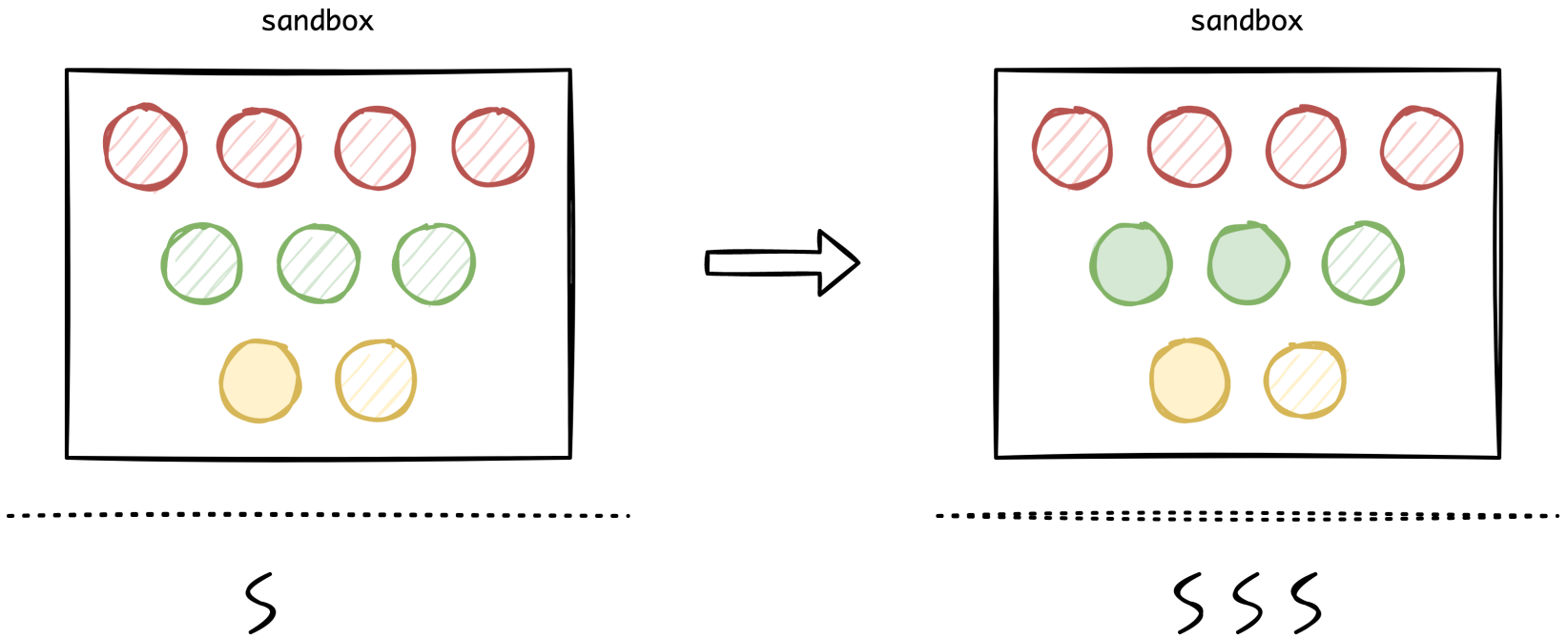
Fork sandbox

1. Connect to seed to trigger fork



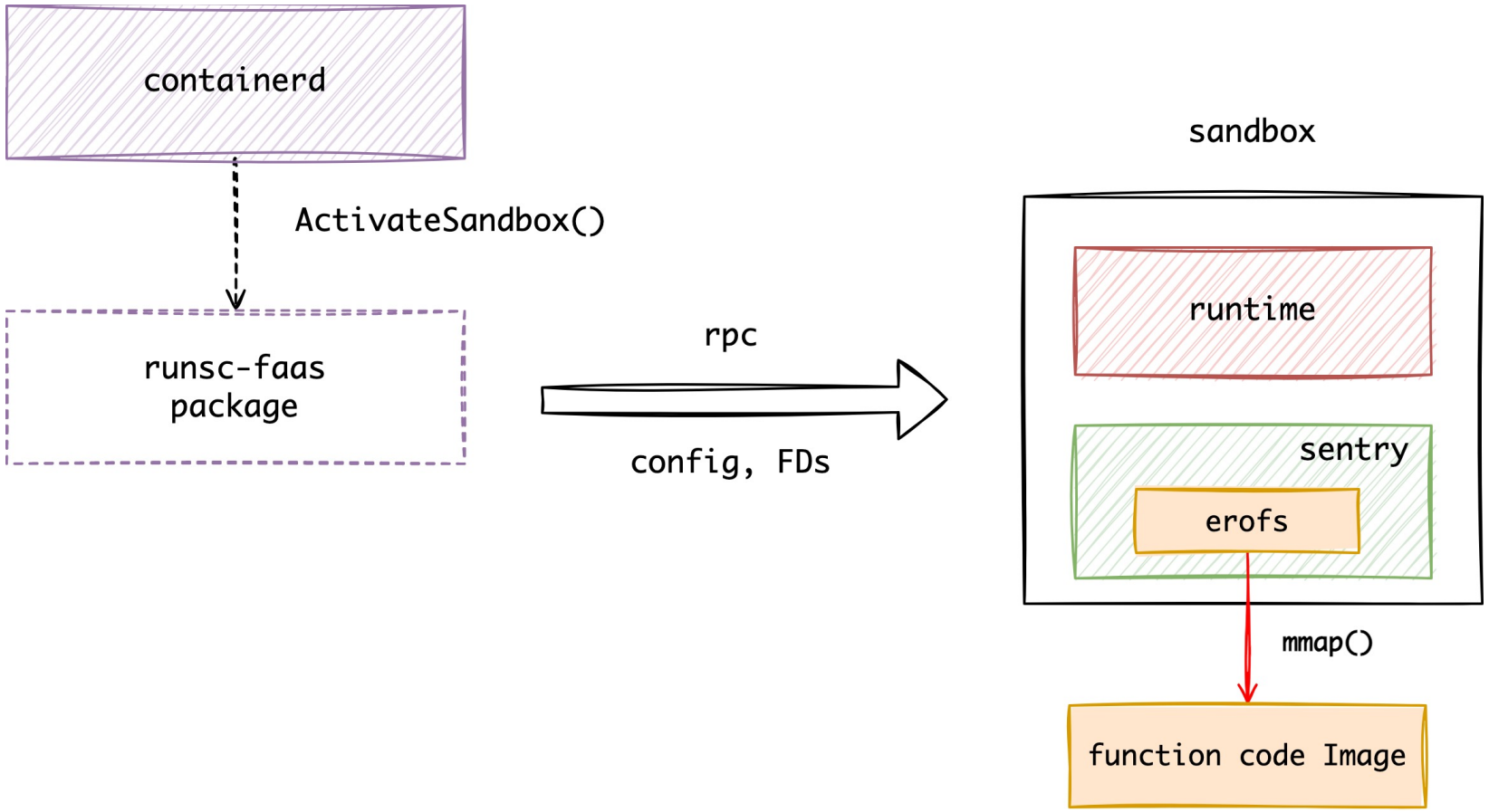
Fork sandbox

2. Recover Go runtime



Fork sandbox

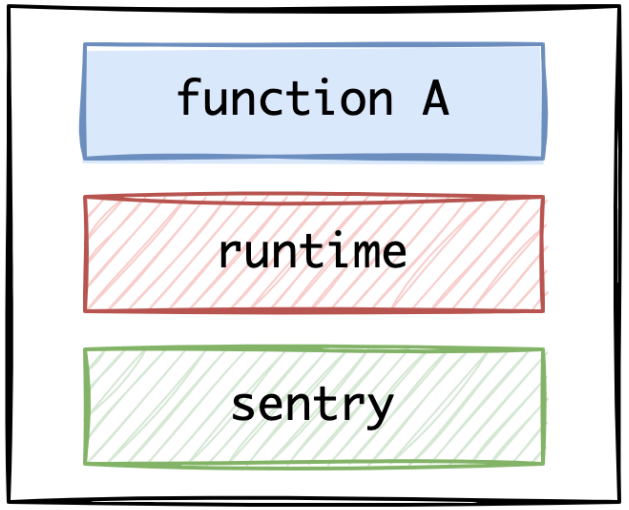
3. Update config and mount code



Fork sandbox

4. Run function

- Resume guest kernel
- Sync with language runtime to run function

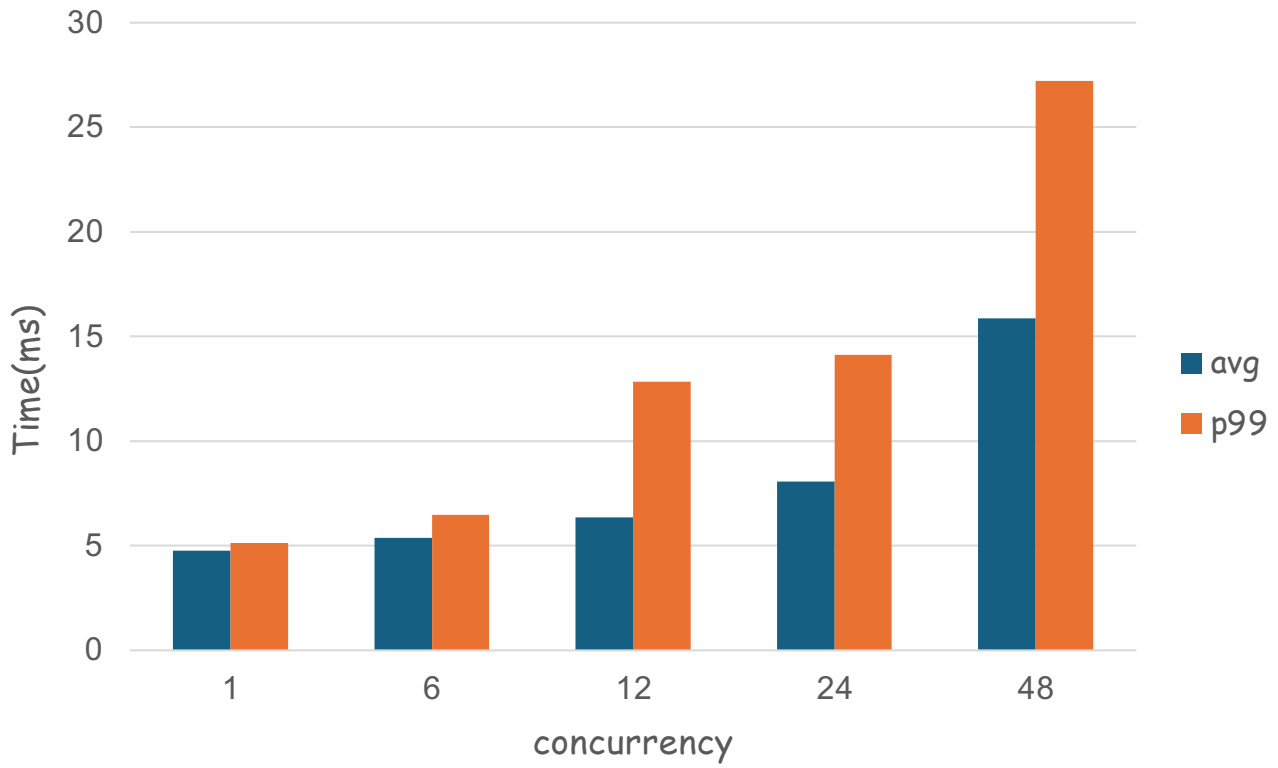


Evaluation

Evaluation

- ~5ms cold start (concurrency 1)
- <1MB memory overhead per instance

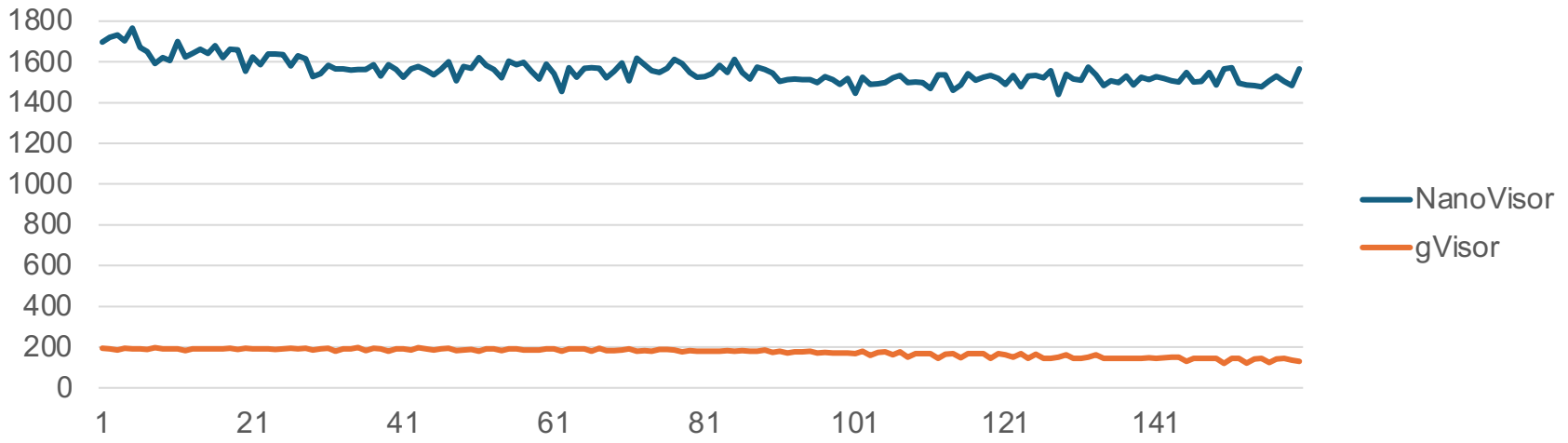
Kernel:	Linux 5.10
CPU:	Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz
Cores:	48 core 96 threads



Evaluation

- ~5ms cold start (concurrency 1)
- <1MB memory overhead per instance
- e2e 1.5K QPS on a single server

Kernel: Linux 5.10
CPU: Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz
Cores: 48 core 96 threads



Summary

Summary

- Fast interaction
- Single Sandbox Process
- Sandbox Fork

NanoVisor based FaaS is available on Alipay Cloudbase[1].

[1] https://cloud.alipay.com/main/product/cloudbase

Thanks!