Serverless Functions Made Confidential and Efficient with Split Containers

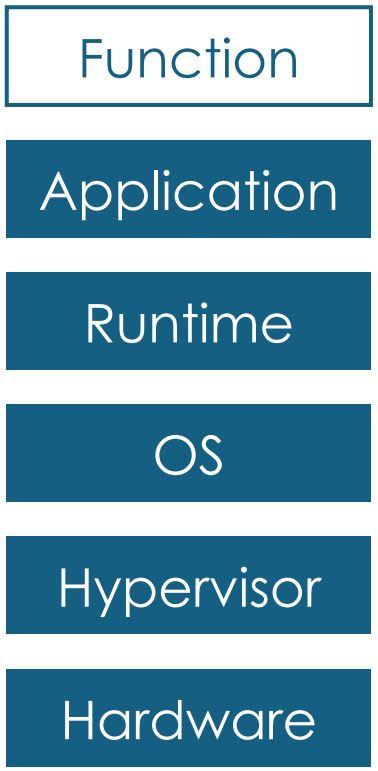
Jiacheng Shi, Jinyu Gu, Yubin Xia, Haibo Chen

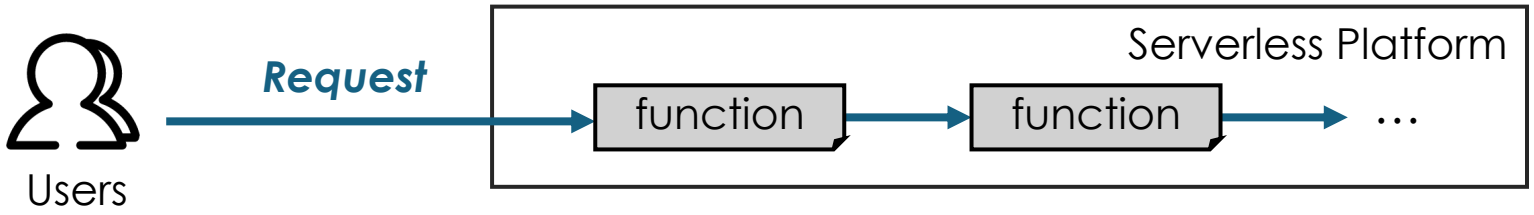
Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University

Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, China

Serverless Requires Security

Function-as-a-Service





Simple DevOps *Auto-scaling* *Pay-as-you-go*

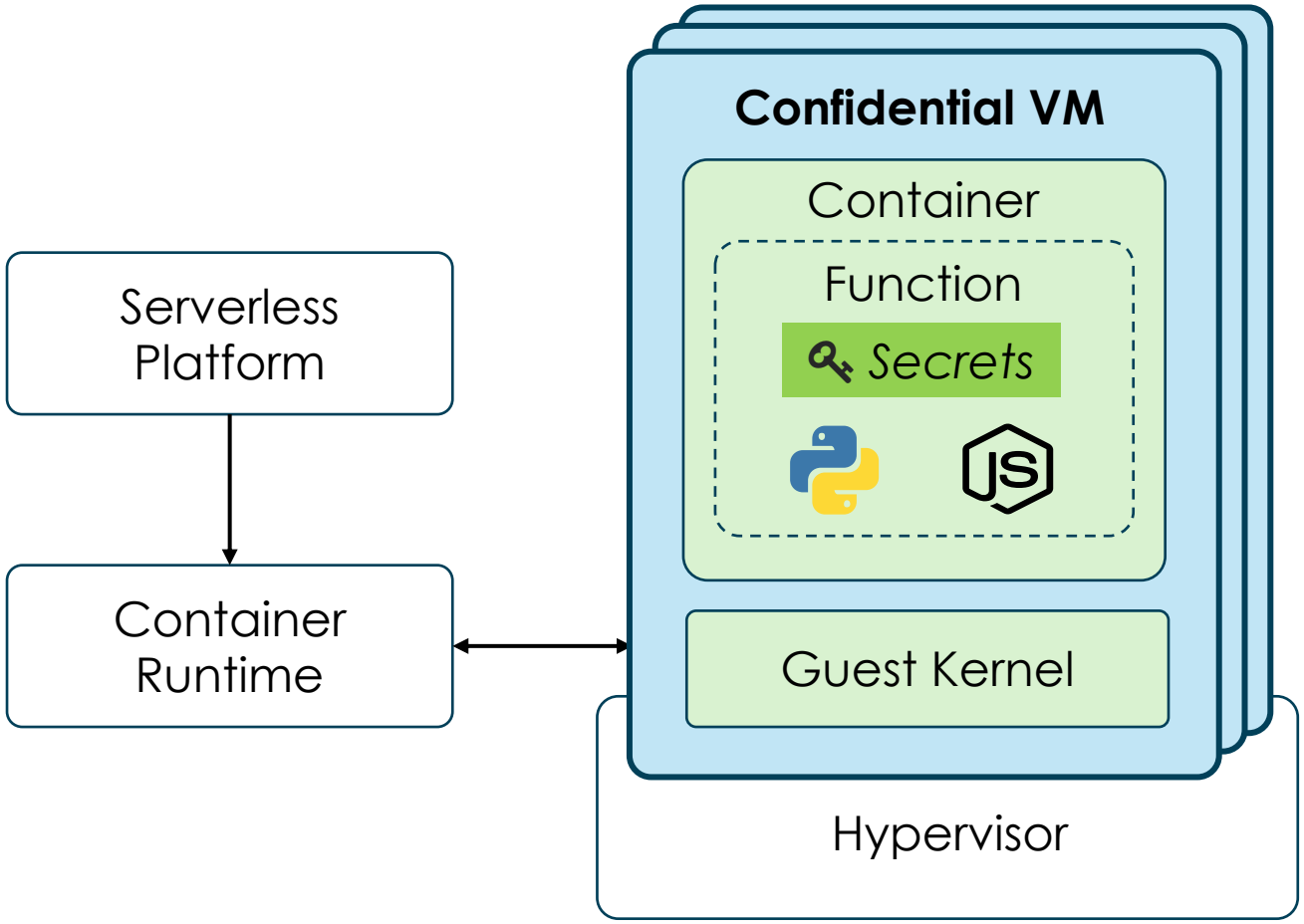
Serverless has been adopted in security-critical scenarios


Facial recognition


Healthcare


Finance

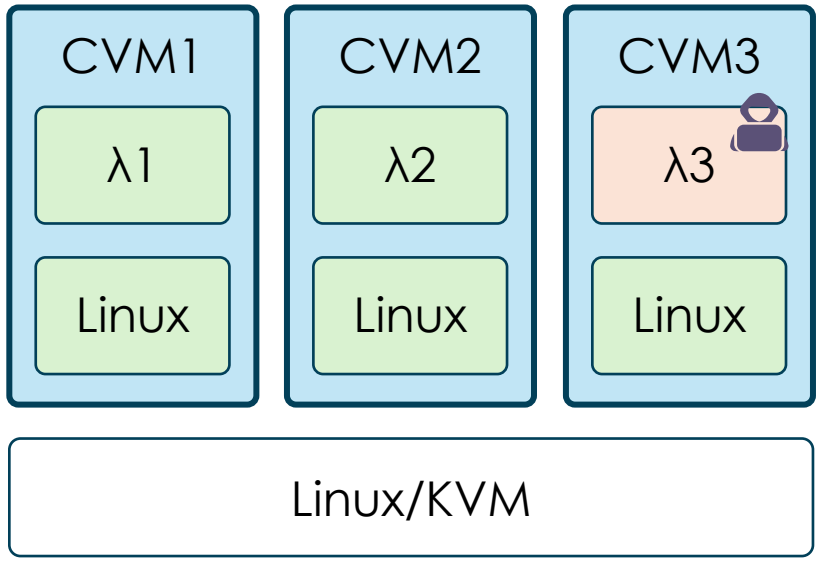
Confidential Containers



- Functions run in containers
- **TCB of containers is huge**
  - Compromised host OS
  - Compromised hypervisor
  - Compromised cloud stack
  - Malicious cloud insiders
- **Run containers in CVMs**
  - CVM protects a whole VM from external attacks

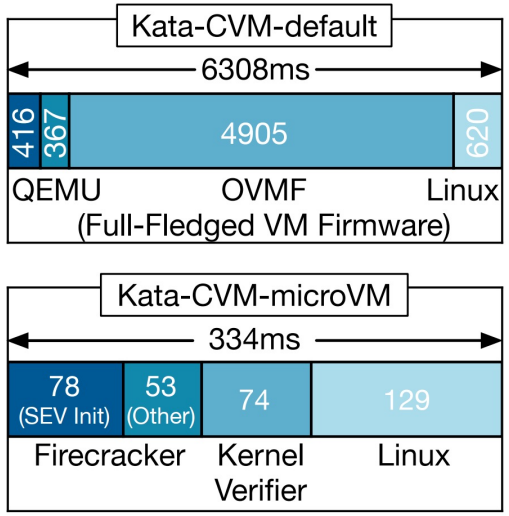
Mismatch: CVM and Serverless

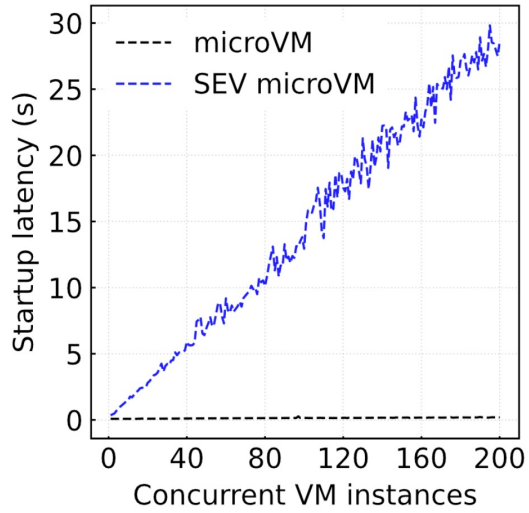
Design choice 1 (Kata-Containers)
Per-Container-CVM



Strong inter-function isolation

Cold start: Large startup latency (CVM init + attest)





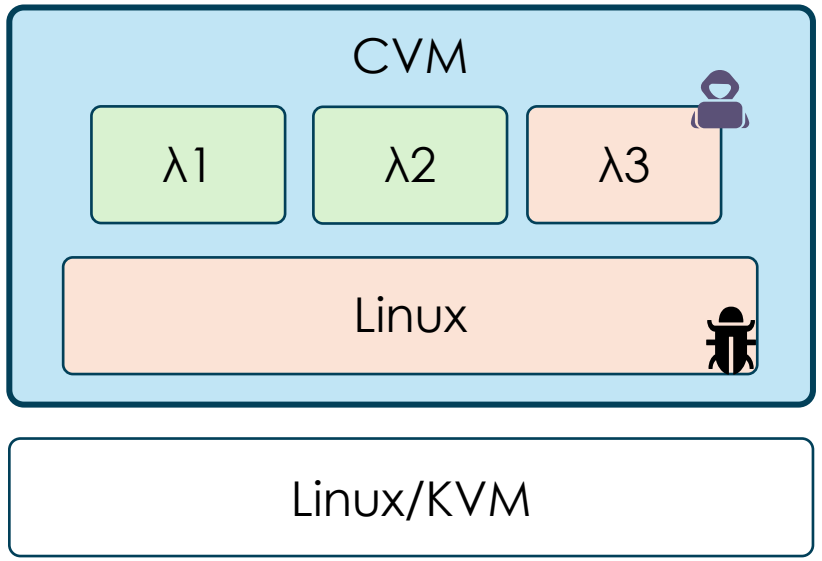
Single microVM: 334ms 200 concurrency: 28.5s

Warm start: Large memory consumption

Caching 500 CVMs: 42.5GB
(No cross-CVM memory-sharing)

Mismatch: CVM and Serverless

Design choice 2
Single-CVM-Multiple-Containers



Eliminate CVM cold start overhead
Enable cross-function memory-sharing

Weak inter-function isolation

Large TCB: CVM guest Linux kernel

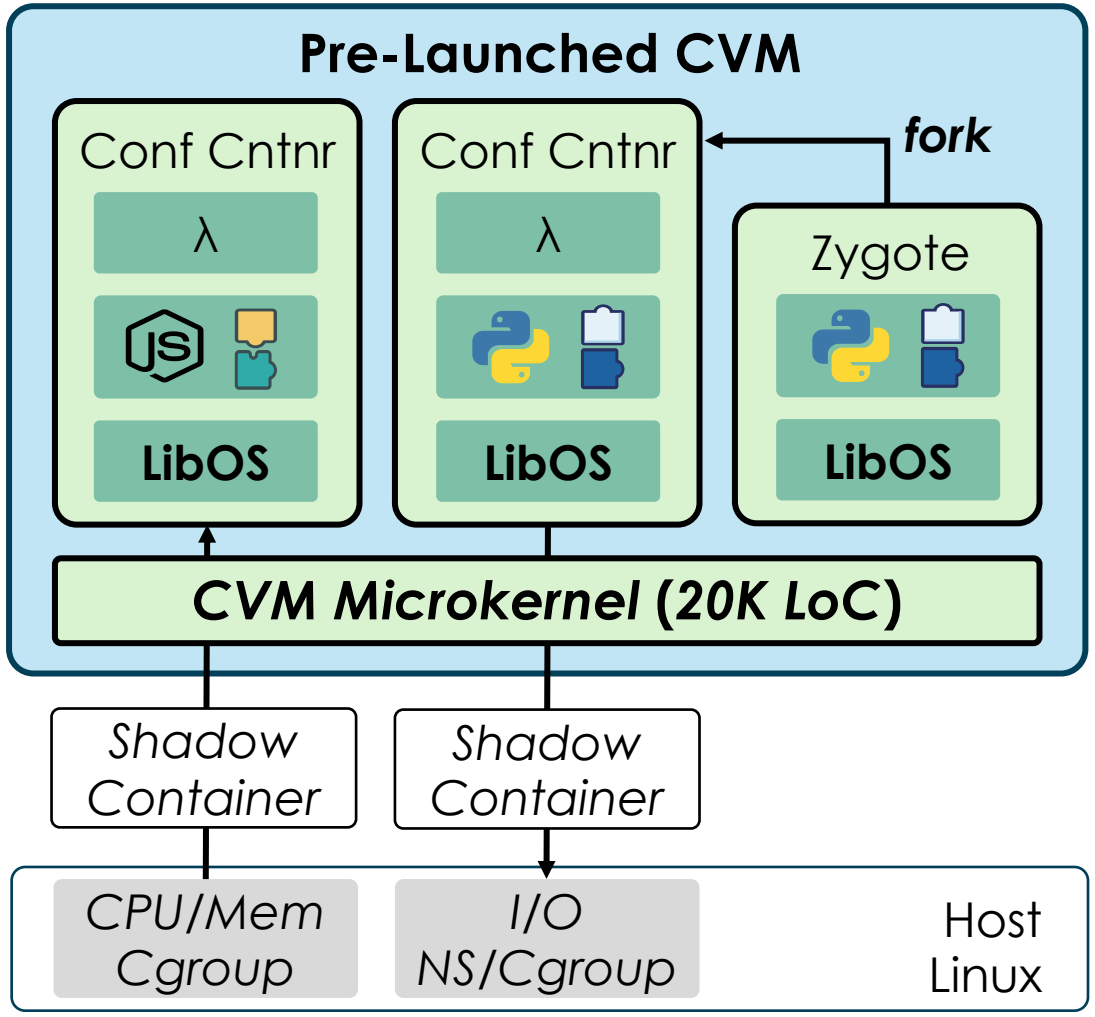
Serverless functions do not require a full-fledged Linux kernel

Single-Process Data-Plane Stateless

System calls used by 150+ functions

Syscall	Number	Syscall	Number
FS	26	Event	4
Network	12	Misc	5
Memory	6	Dummy	15
Sync	3		
Thread	3	Total	**74**

Split Container Architecture



Performance: multiple containers in one CVM

Security: isolated with a ***microkernel***

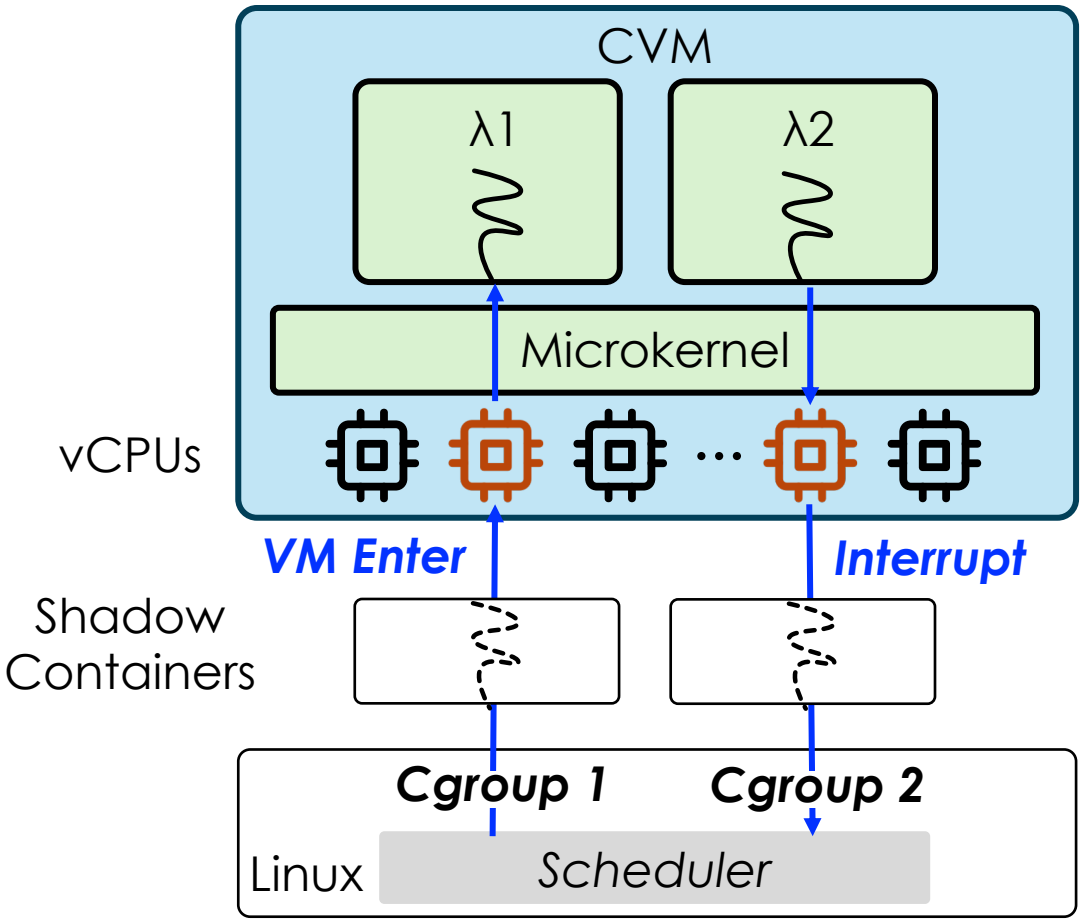
Decouple security from management

- Reuse host cgroups and namespaces
  - Shadow container: ***resource granting*** and ***I/O delegation***
  - Confidential container: secure execution context

- Microkernel + libOS in the CVM

Fast container startup
copy-on-write fork and ***split-attestation***

CPU Resource Granting



One-to-one thread mapping

Shadow Thread – vCPU – Confidential Thread

Cgroup Accounting

Host Scheduler

Allocate a time slice to a shadow thread

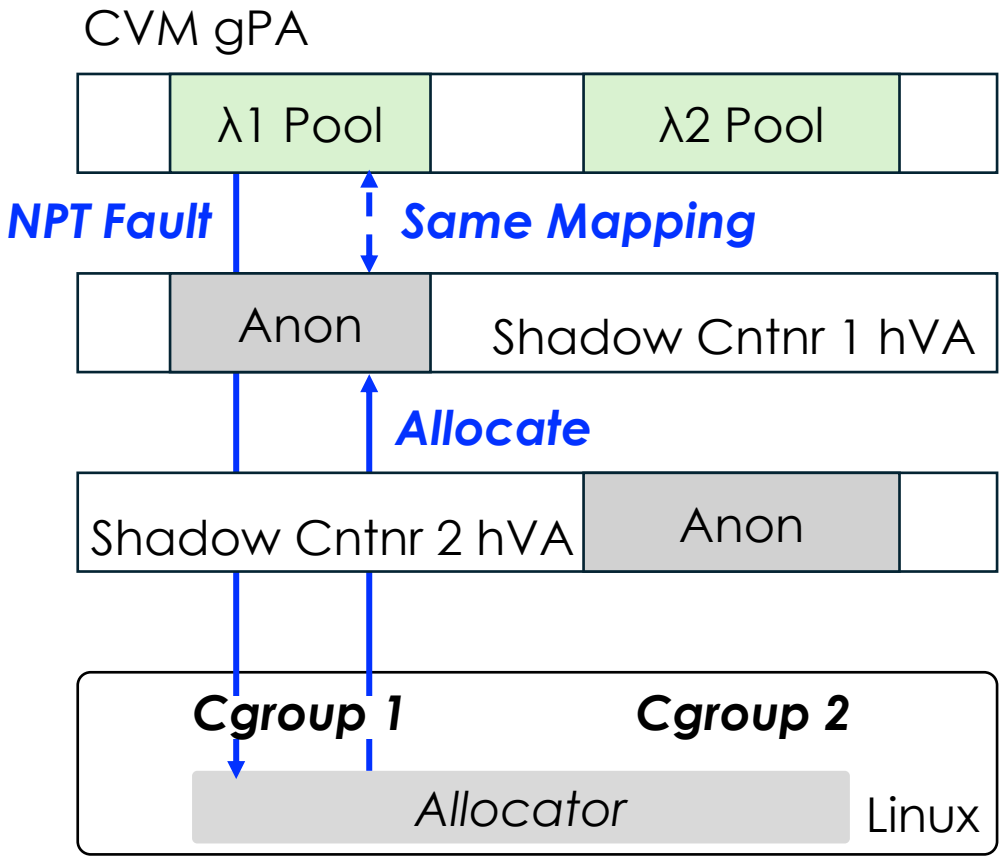
Shadow thread

Activate the confidential thread with VM enter

Confidential thread

Run with the allocated time slice (until interrupted)

Memory Resource Granting



Per-container gPA pool

Same PA mapping: gPA Pool – shadow hVA Pool

Guest Allocator

Conf. cntnr. pgfault → Allocate from gPA pool

gPA not mapped → pvalidate (NPT fault)

Host Allocator

Find gPA-to-hVA mapping

Allocate pages for the hVA

Map the same pages to the CVM NPT

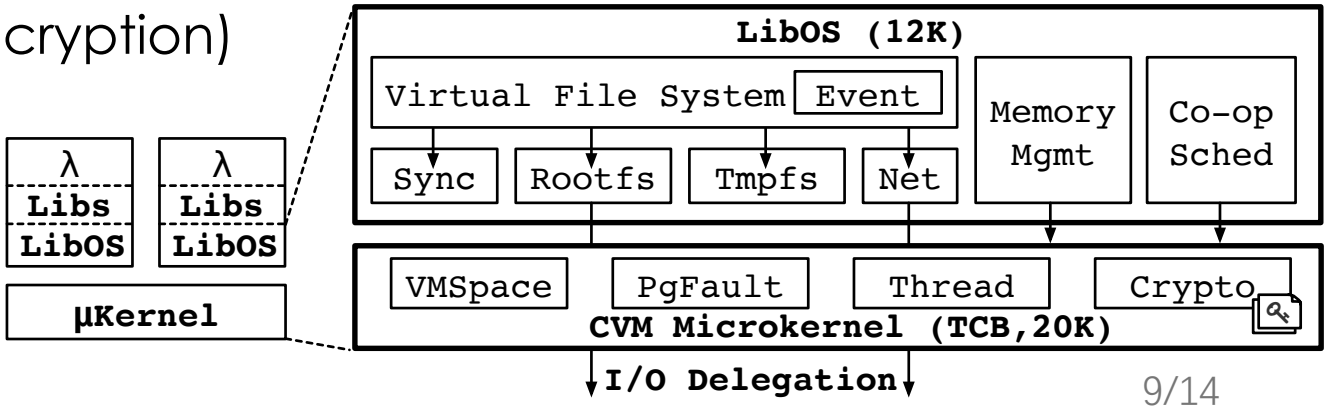
Cgroup Accounting

CVM OS Implementation

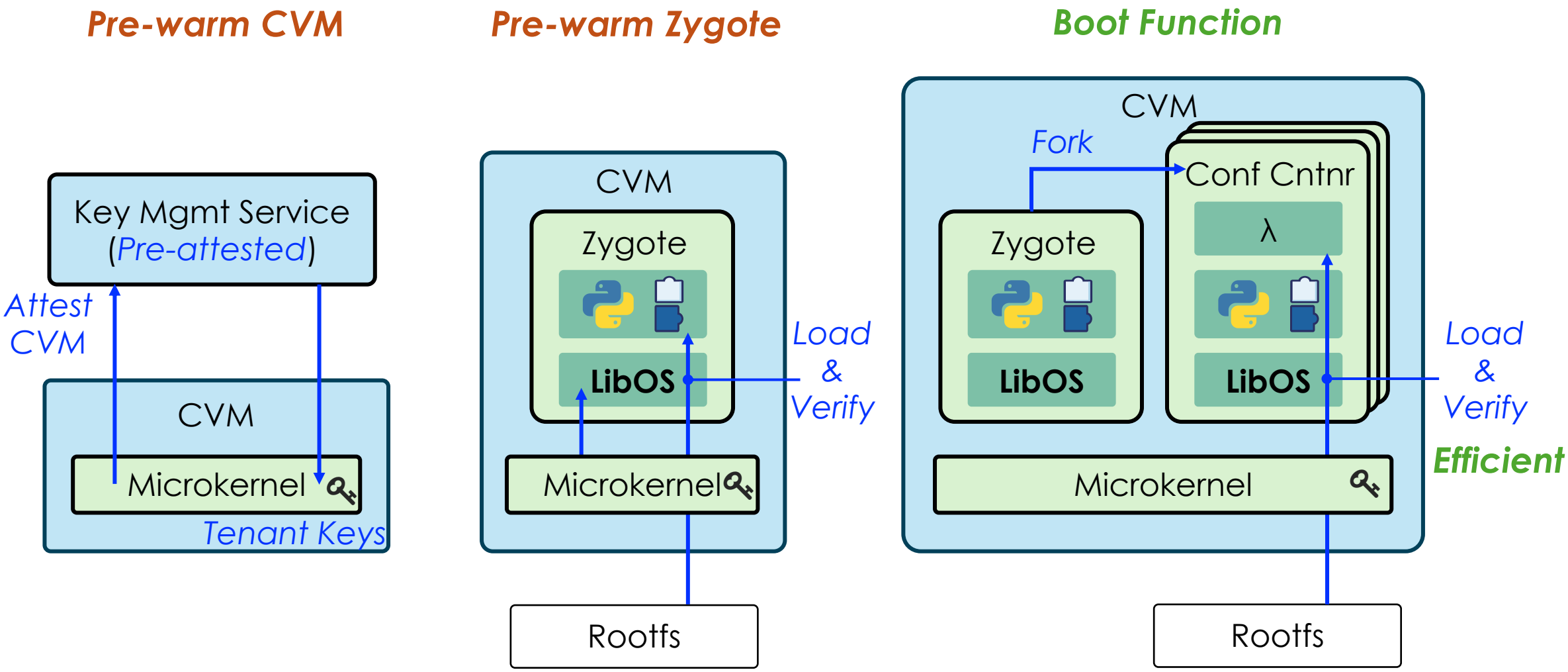
- **Microkernel**: Security-critical components
  - Page table update, crypto library (tenant keys)
- **libOS**: System call functionalities
  - Virtual memory management, VFS, tmpfs, sync. primitives, event, …
- **I/O delegation** with data protection
  - FS: rootfs read (w/ verification)
  - Network: socket syscalls (w/ encryption)

Microkernel (Isolation TCB)
20K Lines-of-Code

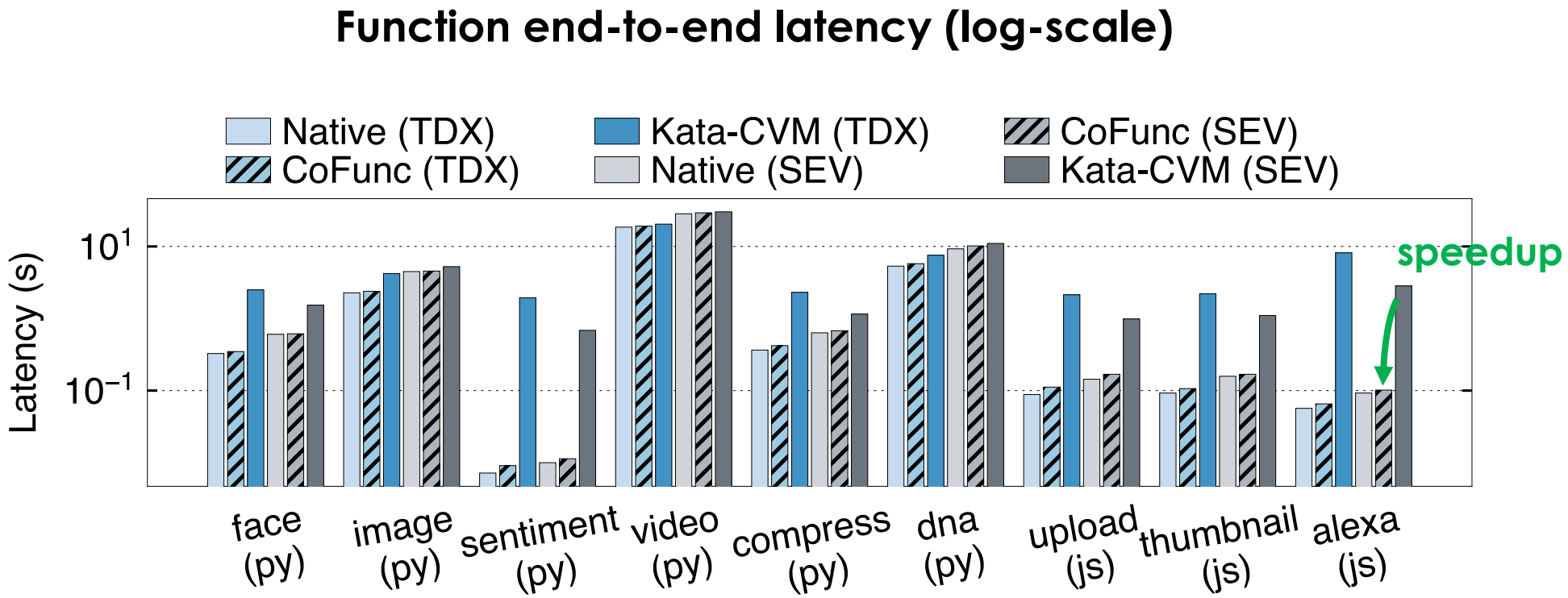
Implemented based on ***ChCore***



Fast Function Boot



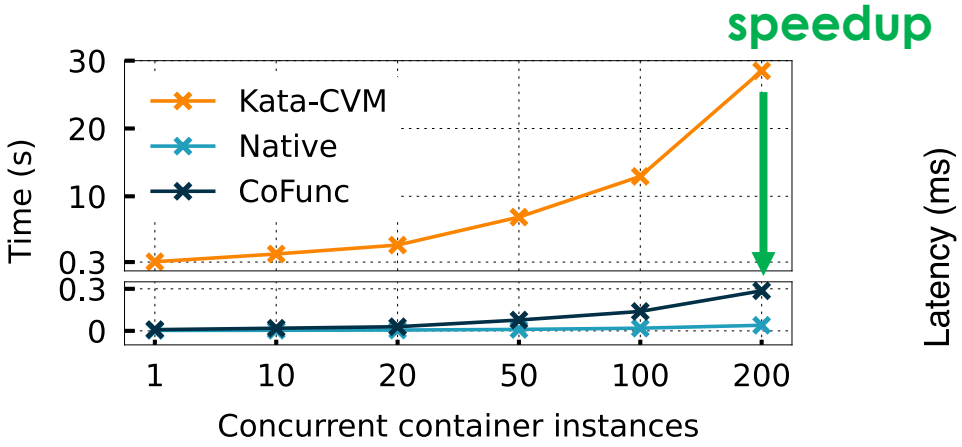
Evaluation: End-to-End Latency



Split-Container (CoFunc**)** v.s. **Per-Container-CVM (**Kata-Containers**)**:
TDX: **1.06×~215× speedup**; SEV (optimized Kata): **1.02×~60× speedup**

Evaluation: Startup Latency

Containerization

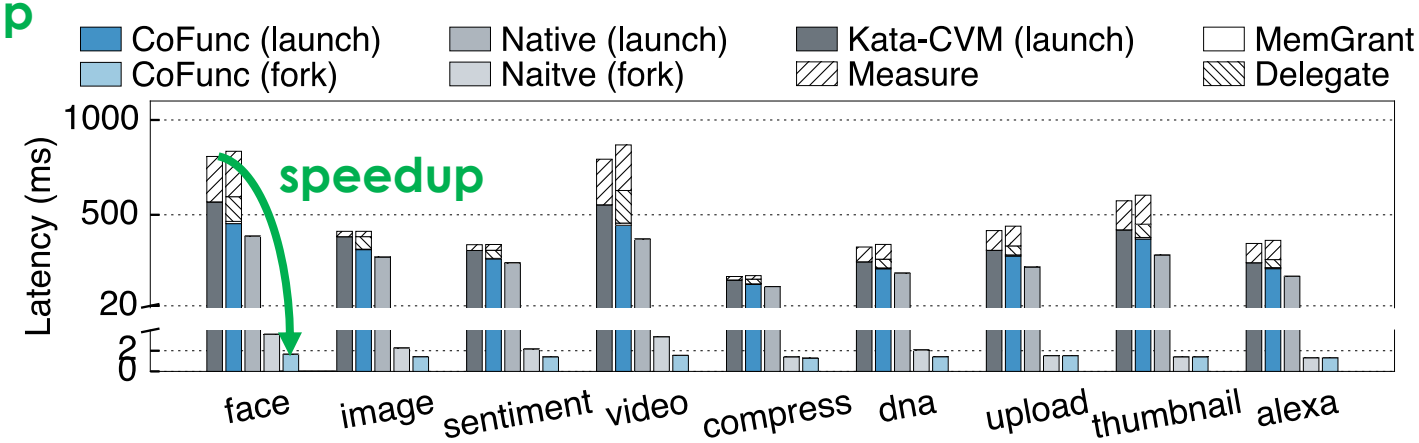


CoFunc v.s. **Kata-Containers**

Single: **120× (TDX), 22.3× (SEV) speedup**

200 concurrent: **100× (SEV) speedup**

Code Loading and Initialization



CoFunc v.s. **Kata-Containers**

148×~499× (TDX), 134×~513× (SEV) speedup

Evaluation

- **Overhead (v.s. native containers)**
  - End-to-end latency (single request): **<14%** on average
  - Startup stage: measurement of function-specific code
  - Execution stage: I/O delegation, encryption, memory granting
- **Reduce memory consumption (v.s. Kata-Containers)**
  - 200 containers w/ Python and libraries: **20×~56× reduction**
- **Function chain optimization (v.s. Kata-Containers)**
  - Communication: network + encryption → shared memory
  - FINRA application: **31× end-to-end latency reduction**

Summary

- **Serverless + CVM**: Enable fast boot of confidential containers while ensuring resource efficiency
- **Split Container**: Securely isolate multiple confidential containers in one CVM with a microkernel
  - Transparently pass through containerization offered by the host kernel into the CVM
  - A new non-Linux CVM OS (microkernel + libOS)