Wallet

Confidential Serverless Computing

Patrick Sabanic, Masanori Misono, Teofil Bodea, Julian Pritzi,
Michael Hackl, **Dimitrios Stavrakakis**, Pramod Bhatotia

Systems Research Group
https://dse.in.tum.de
Technical University of Munich

USENIX NSDI 2026

Serverless computing

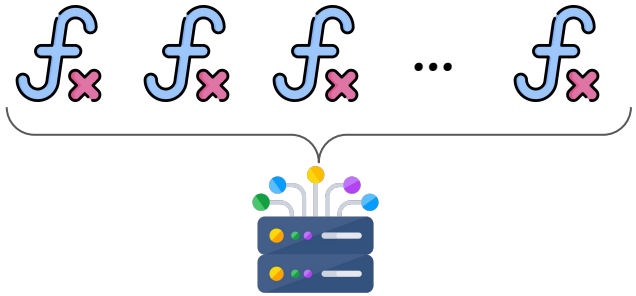
More than **50%** of cloud customers use serverless functions[1]

[1] https://www.datadoghq.com/state-of-containers-and-serverless/

Serverless computing

More than **50%** of cloud customers use serverless functions[1]

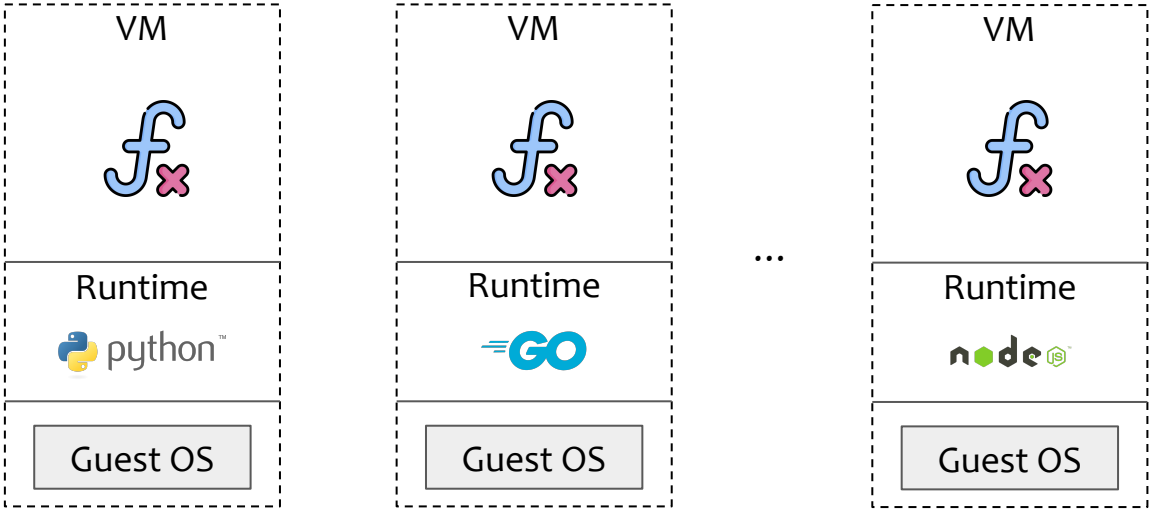
Thousands of functions deployed on shared hardware[2]



[1] https://www.datadoghq.com/state-of-containers-and-serverless/

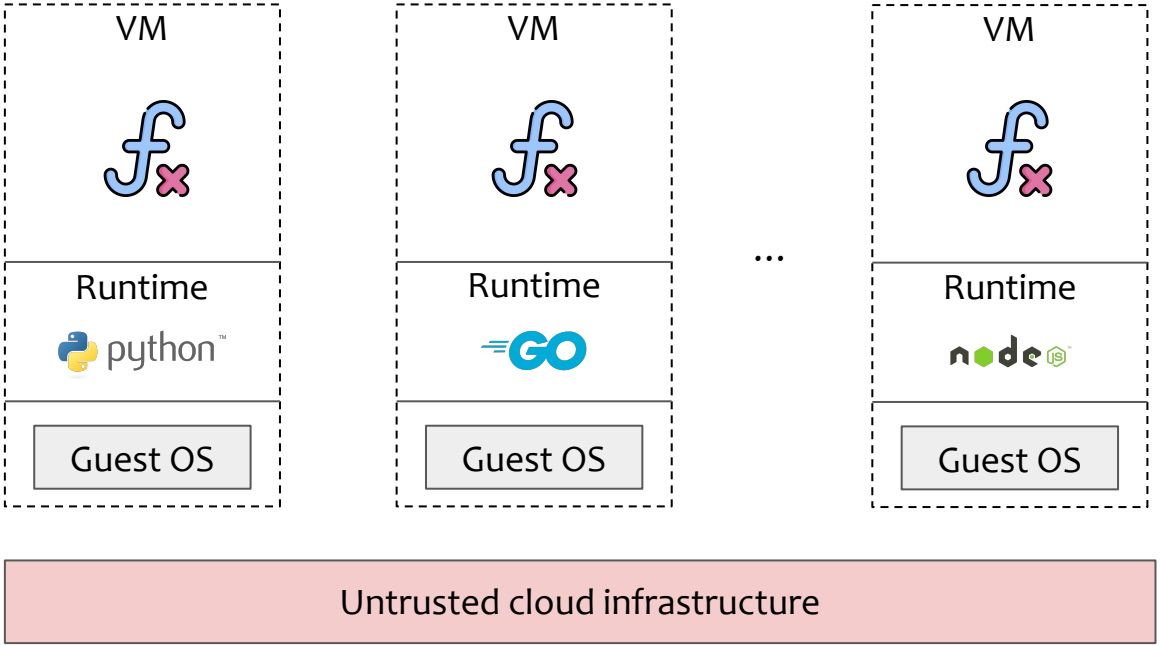
[2] https://www.amazon.science/publications/firecracker-lightweight-virtualization-for-serverless-applications

Trust issues in the cloud



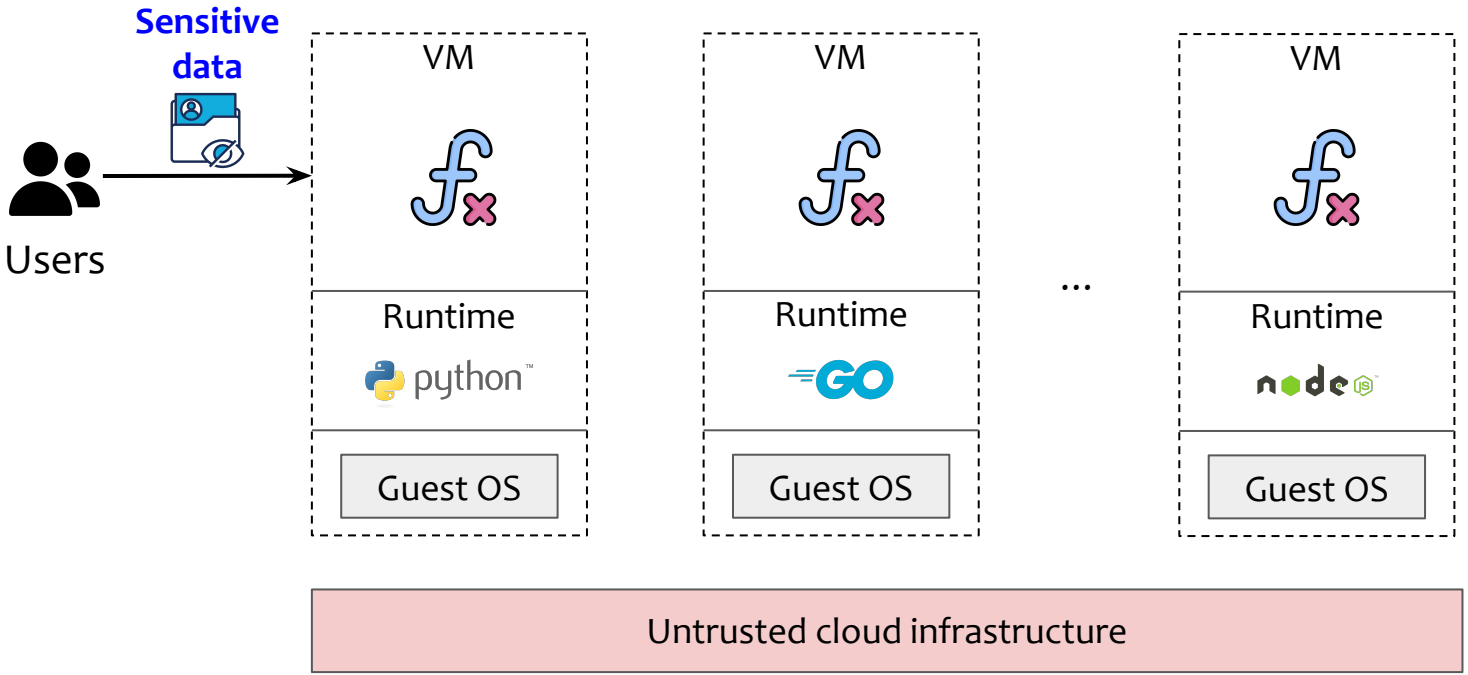
Credits: Flaticon.com

Trust issues in the cloud



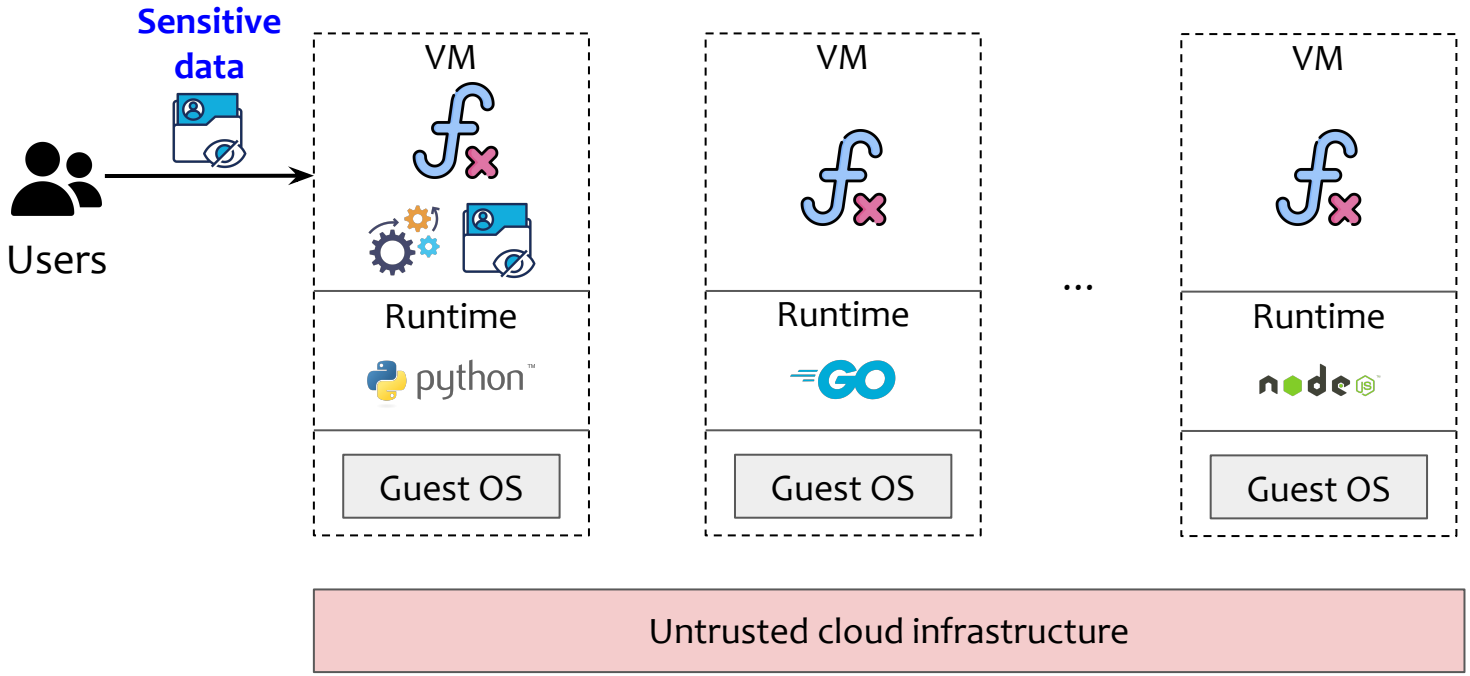
Credits: Flaticon.com

Trust issues in the cloud



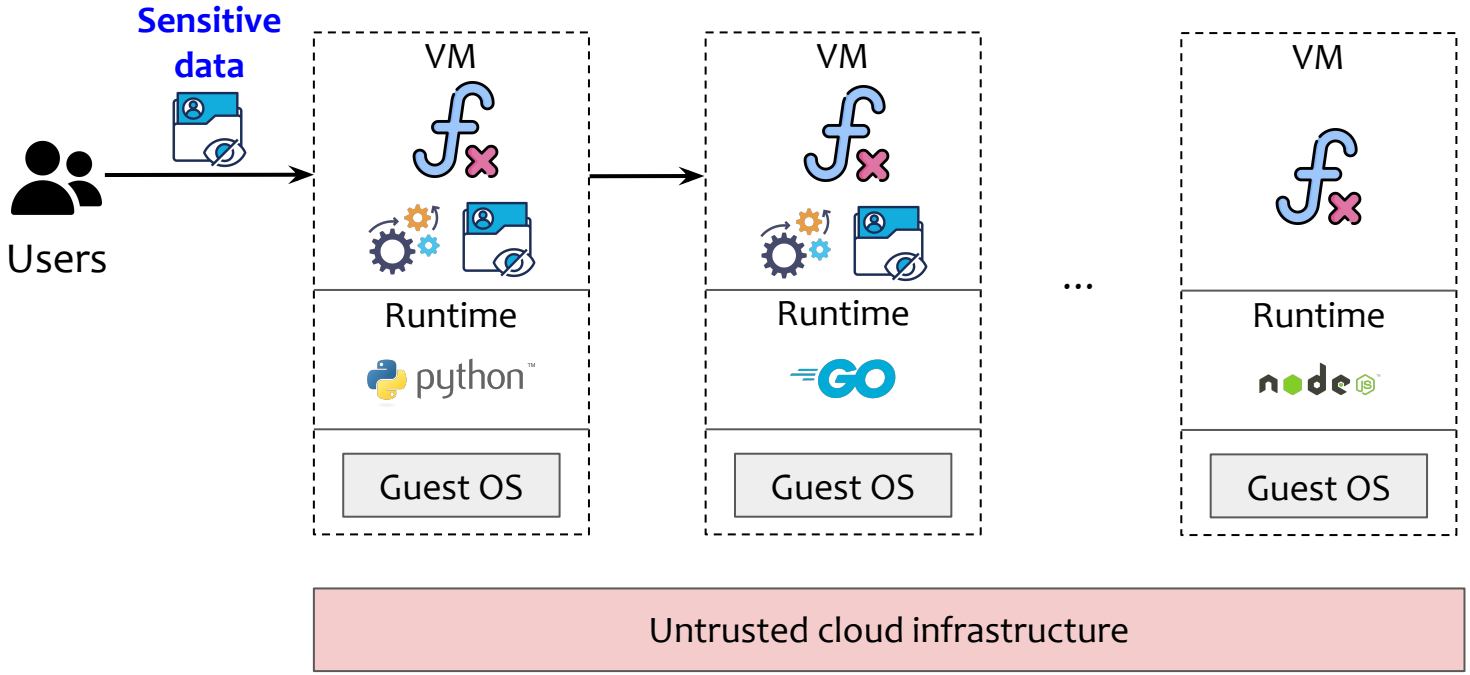
Credits: Flaticon.com

Trust issues in the cloud



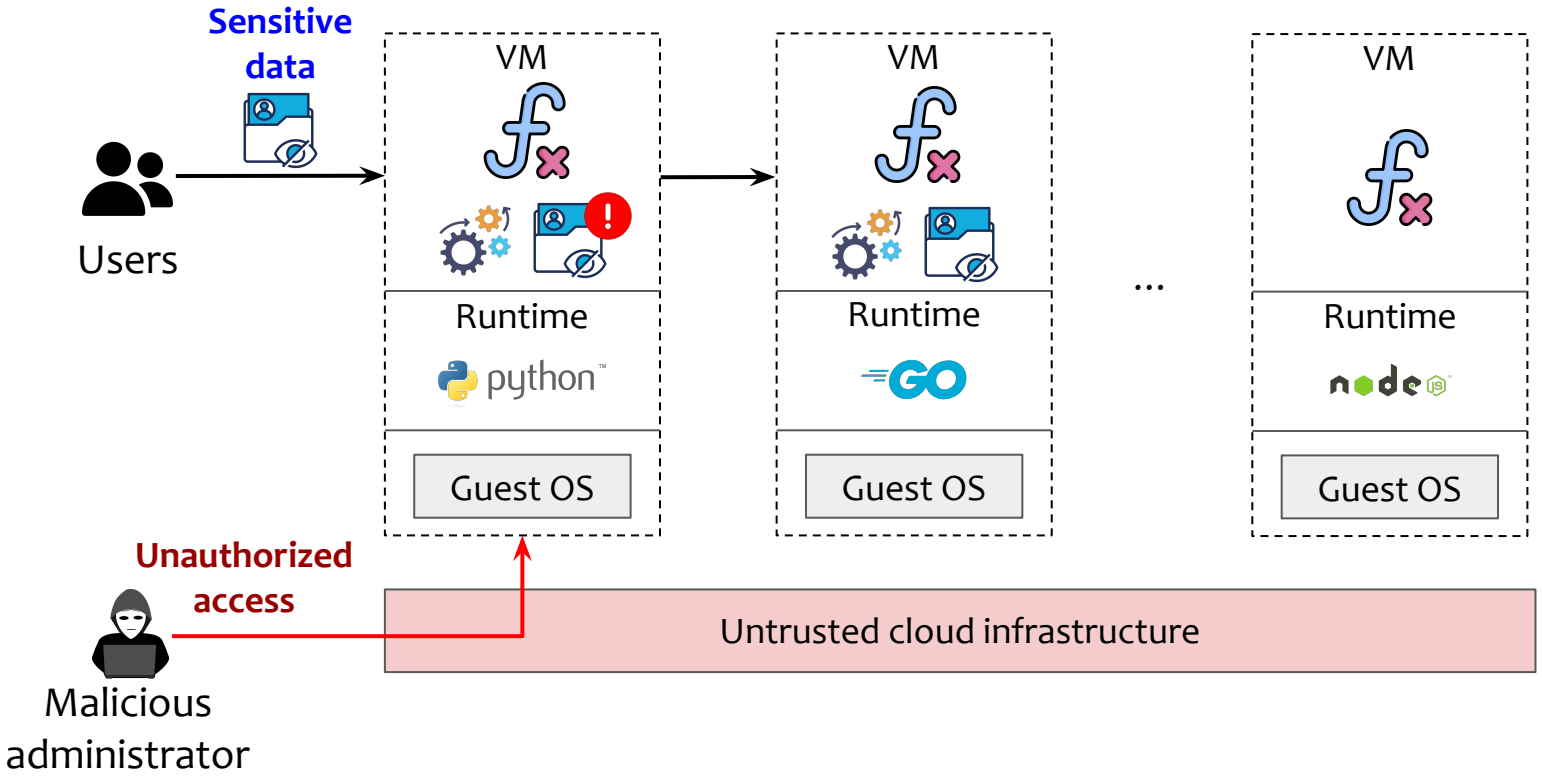
Credits: Flaticon.com

Trust issues in the cloud



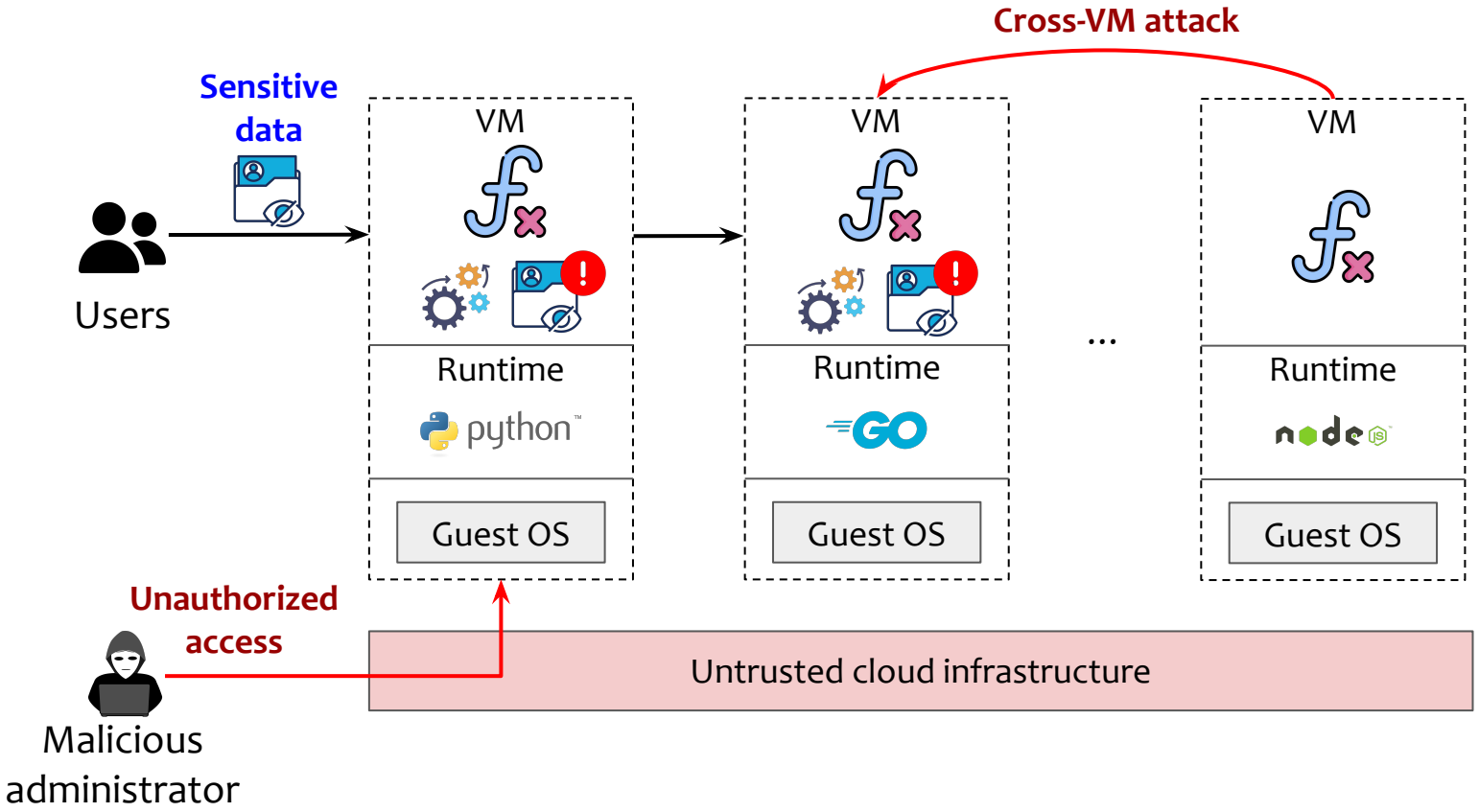
Credits: Flaticon.com

Trust issues in the cloud



Credits: Flaticon.com

Trust issues in the cloud



Credits: Flaticon.com

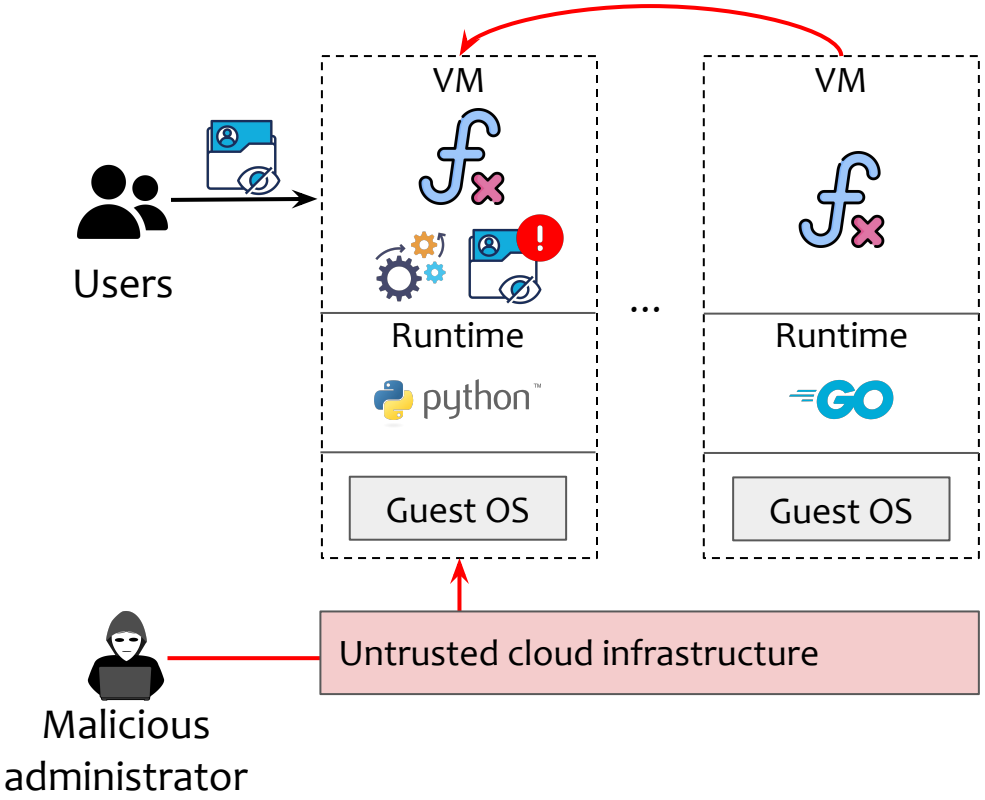
Problem statement

How do we enable **trustworthy serverless computing** in the cloud?

Can we use confidential virtual machines (CVMs)?

CVM properties:

- No trust in hypervisor
- Inter-CVM isolation
- Memory encryption
- Remote attestation

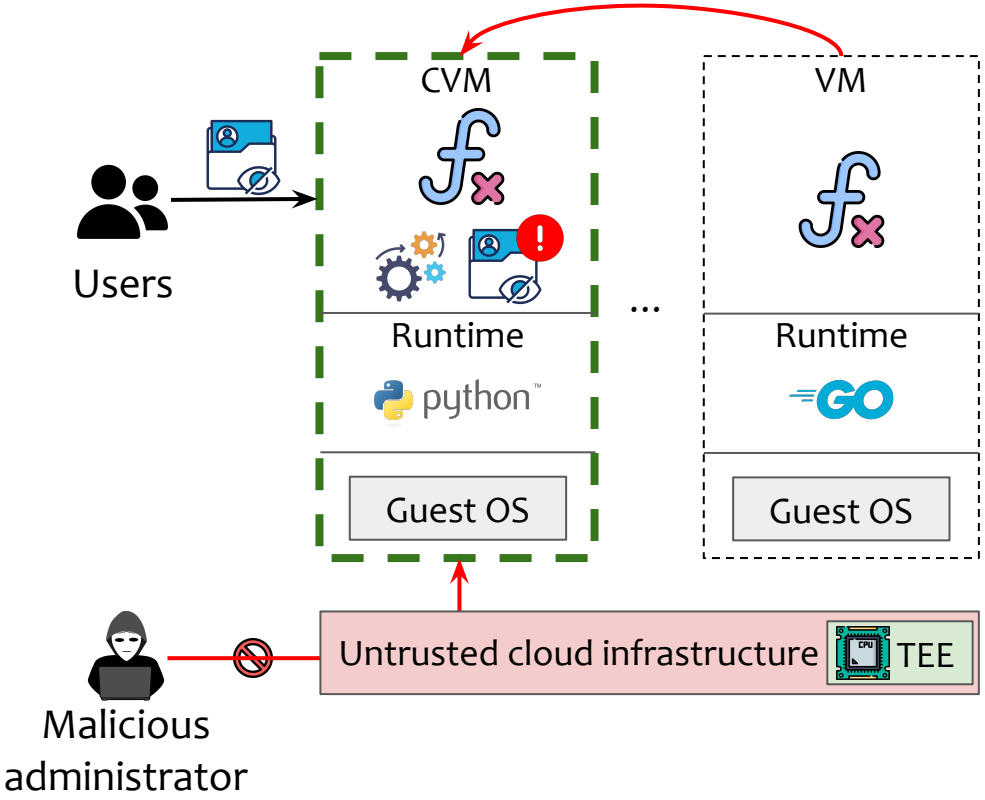


Credits: Flaticon.com

Can we use confidential virtual machines (CVMs)?

CVM properties:

- No trust in hypervisor
- Inter-CVM isolation
- Memory encryption
- Remote attestation

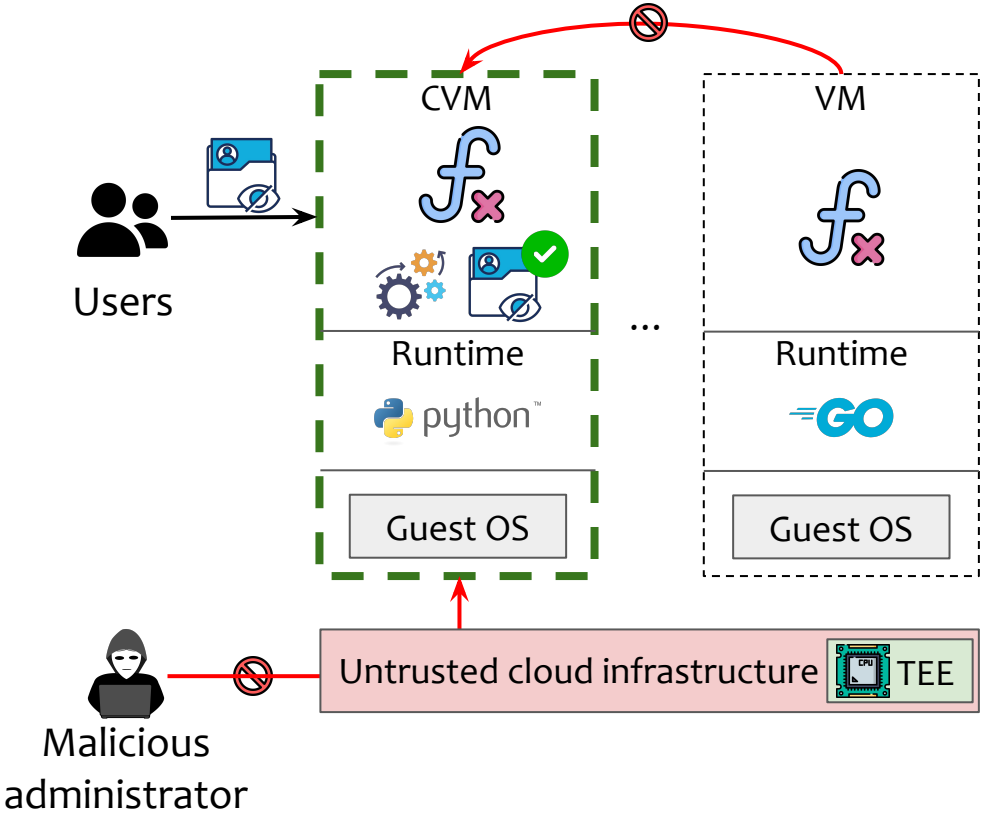


Credits: Flaticon.com

Can we use confidential virtual machines (CVMs)?

CVM properties:

- No trust in hypervisor
- Inter-CVM isolation
- Memory encryption
- Remote attestation

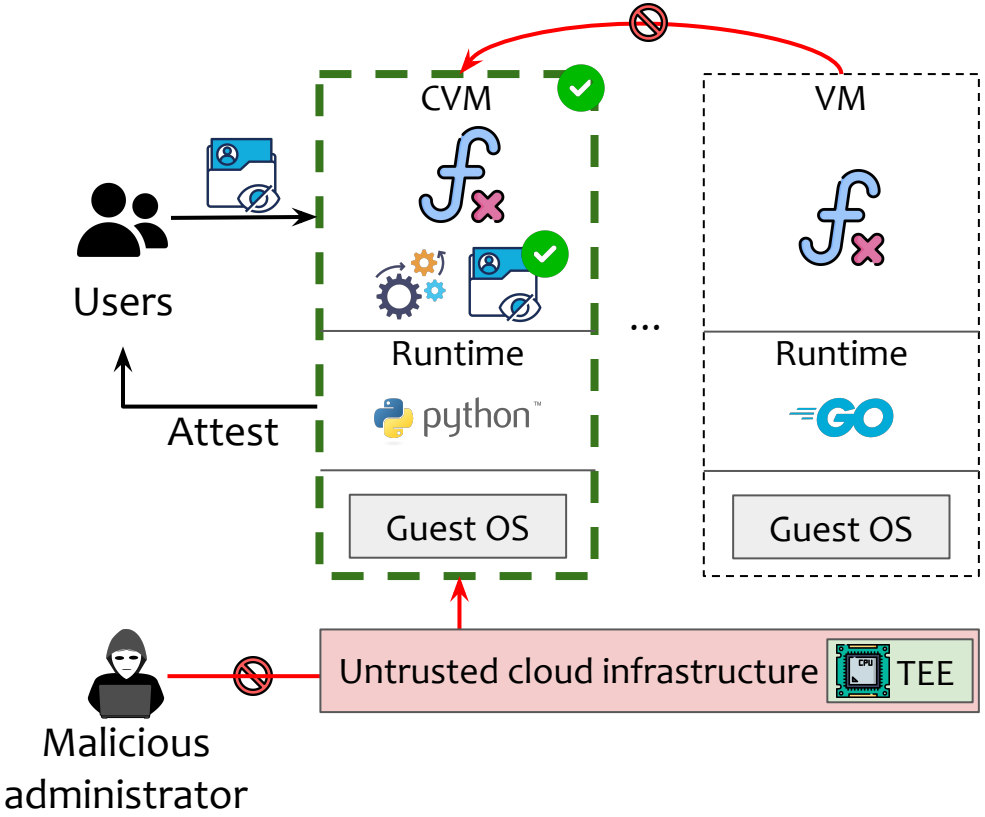


Credits: Flaticon.com

Can we use confidential virtual machines (CVMs)?

CVM properties:

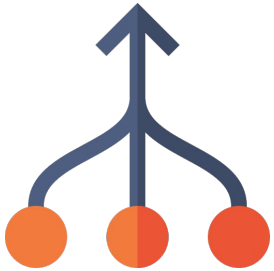
- No trust in hypervisor
- Inter-CVM isolation
- Memory encryption
- Remote attestation



Credits: Flaticon.com

Core CVM Issues for serverless workloads

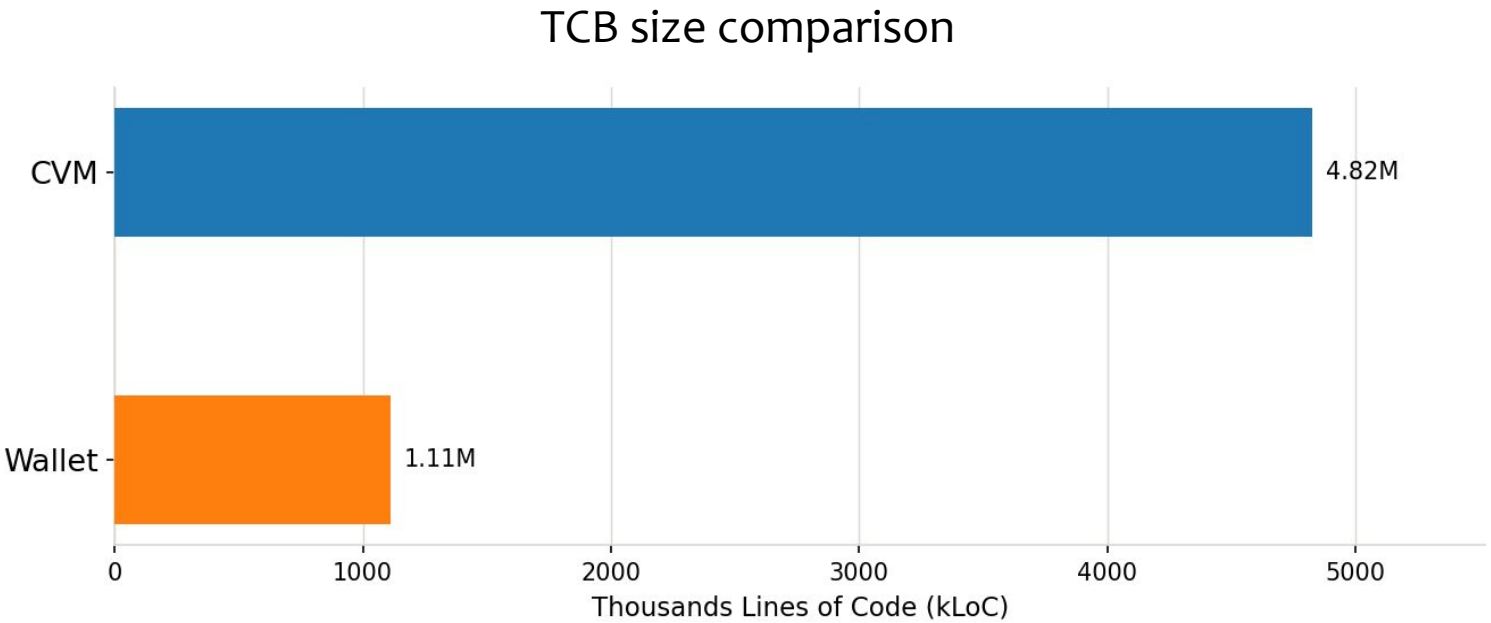
#1: Large Trusted Computing Base (TCB) and high boot times



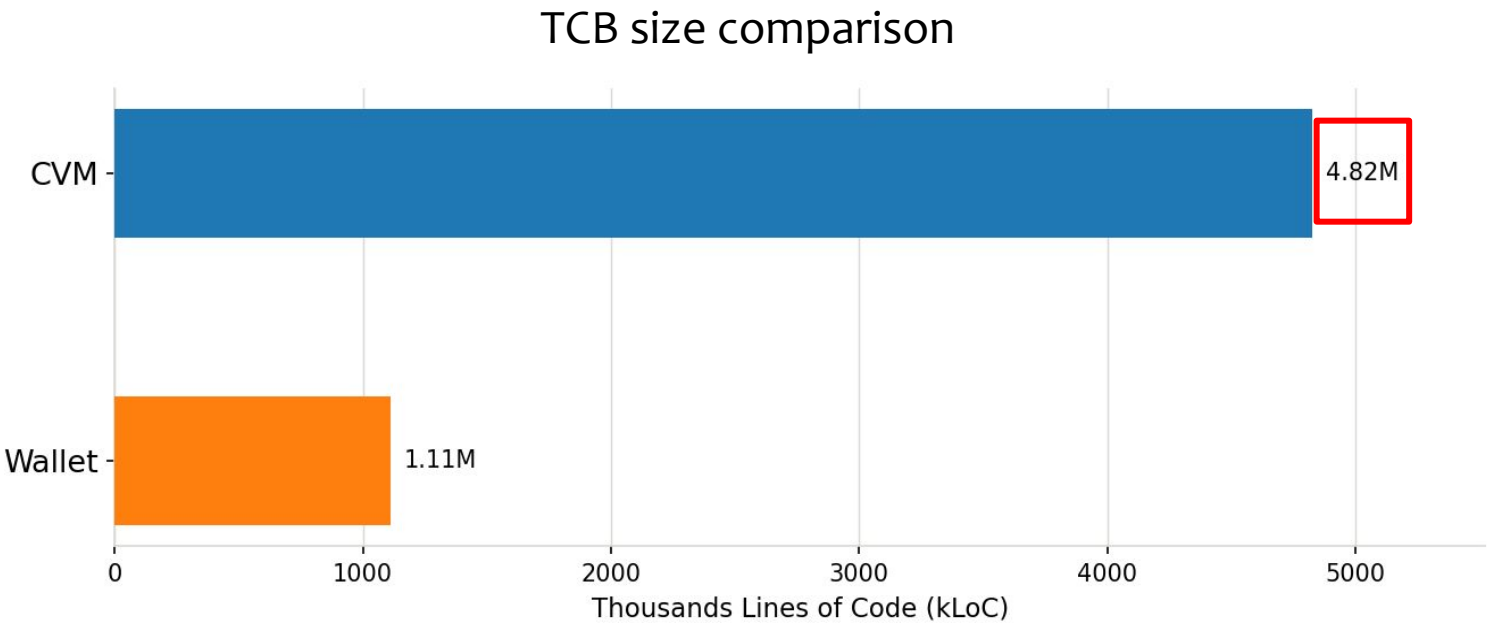
#2: Impractical function consolidation

#3: High inter-function communication costs

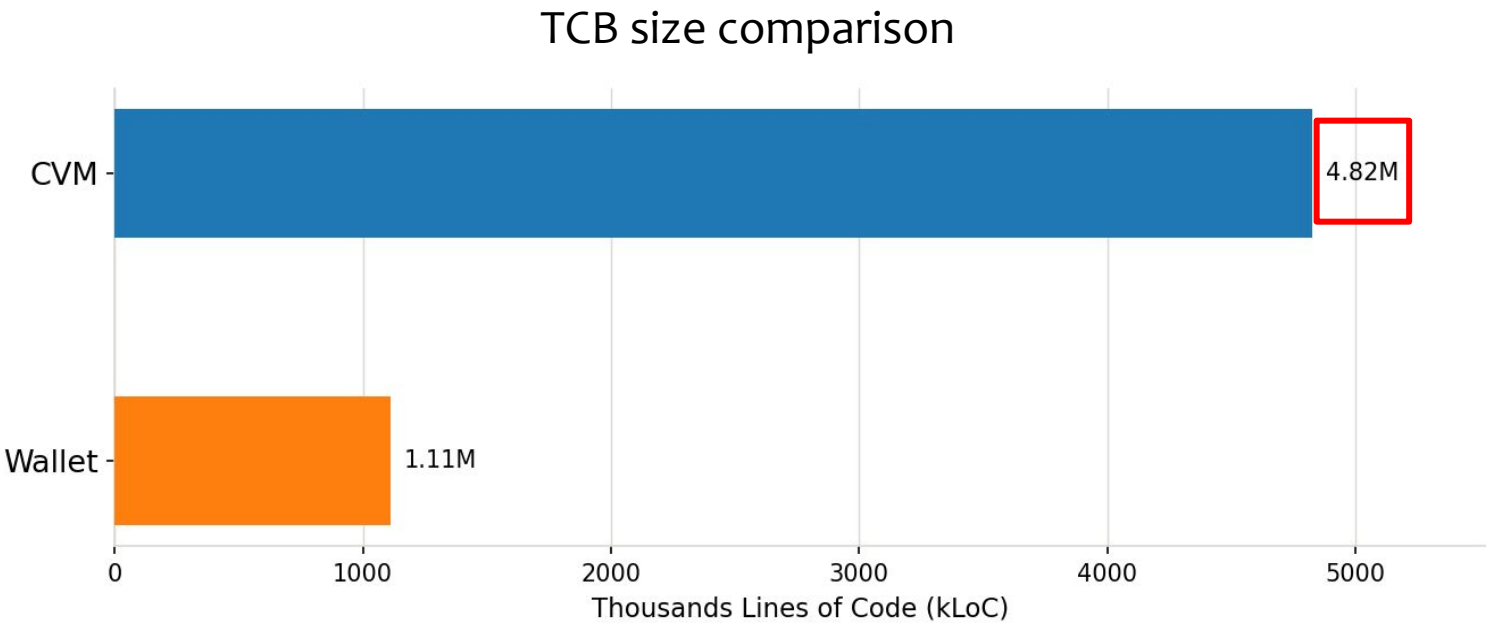
#1: Large TCB and high boot times



#1: Large TCB and high boot times

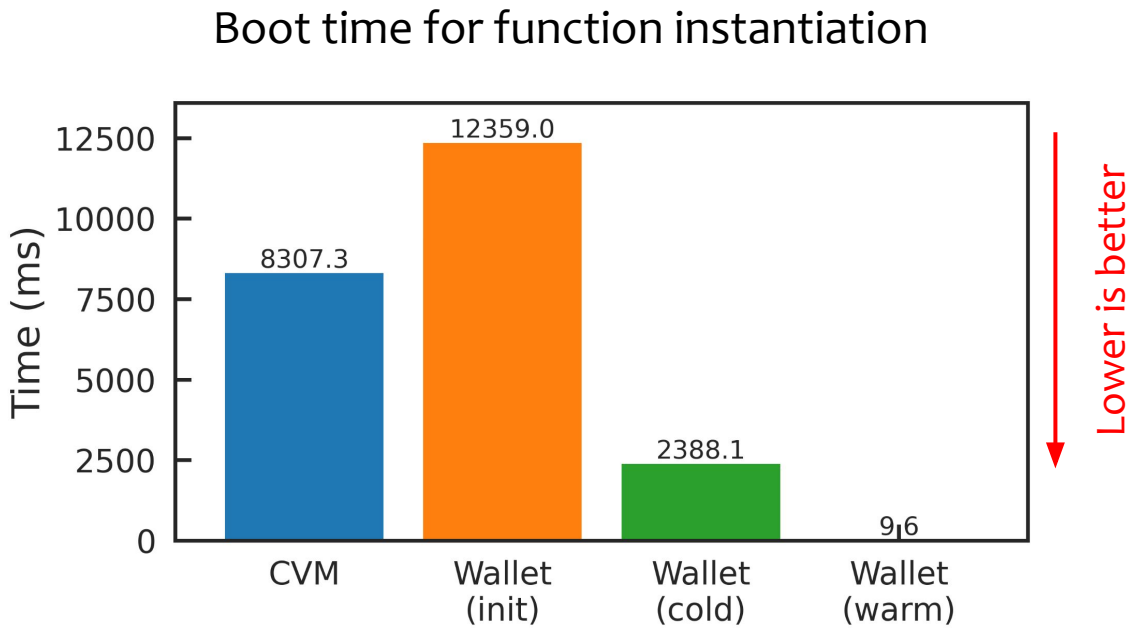


#1: Large TCB and high boot times

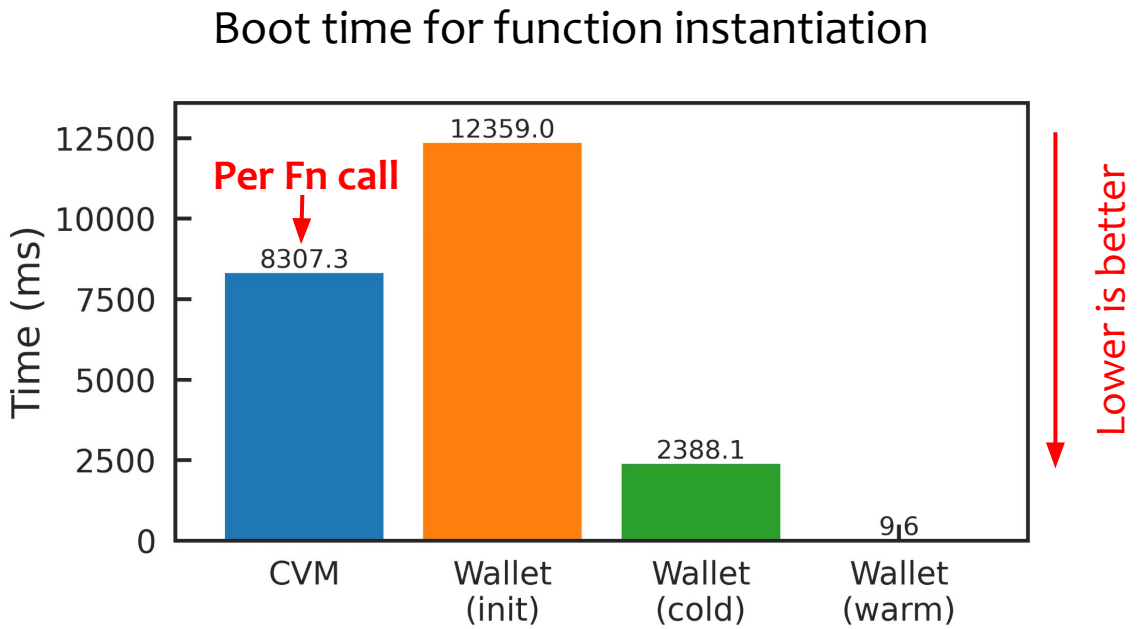


4.3x lower TCB compared to Linux CVM stacks

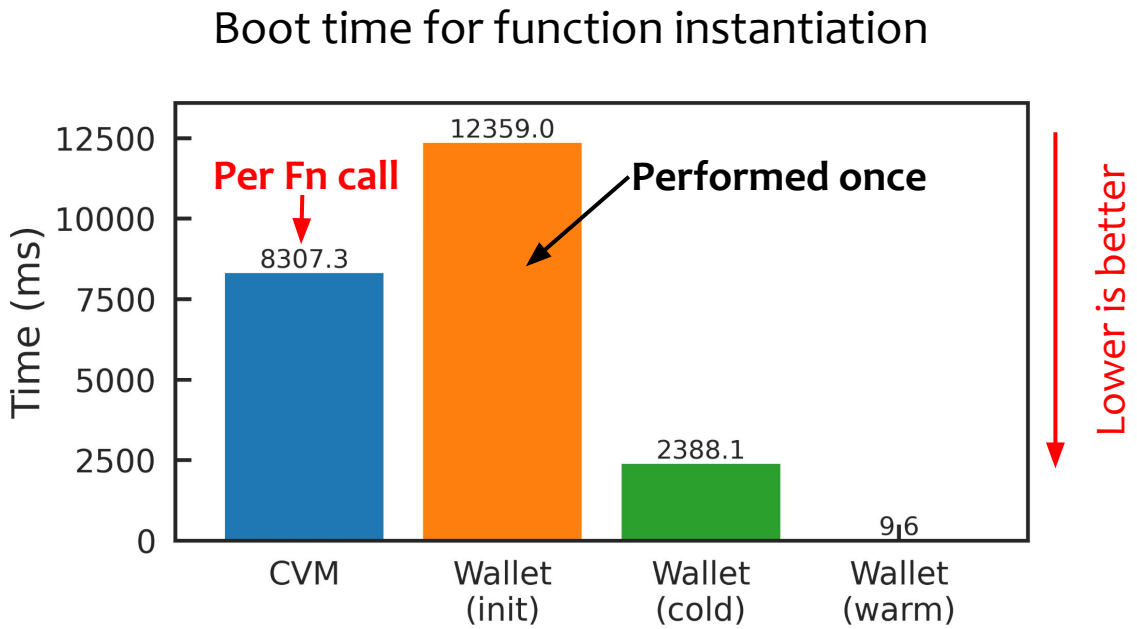
#1: Large TCB and high boot times



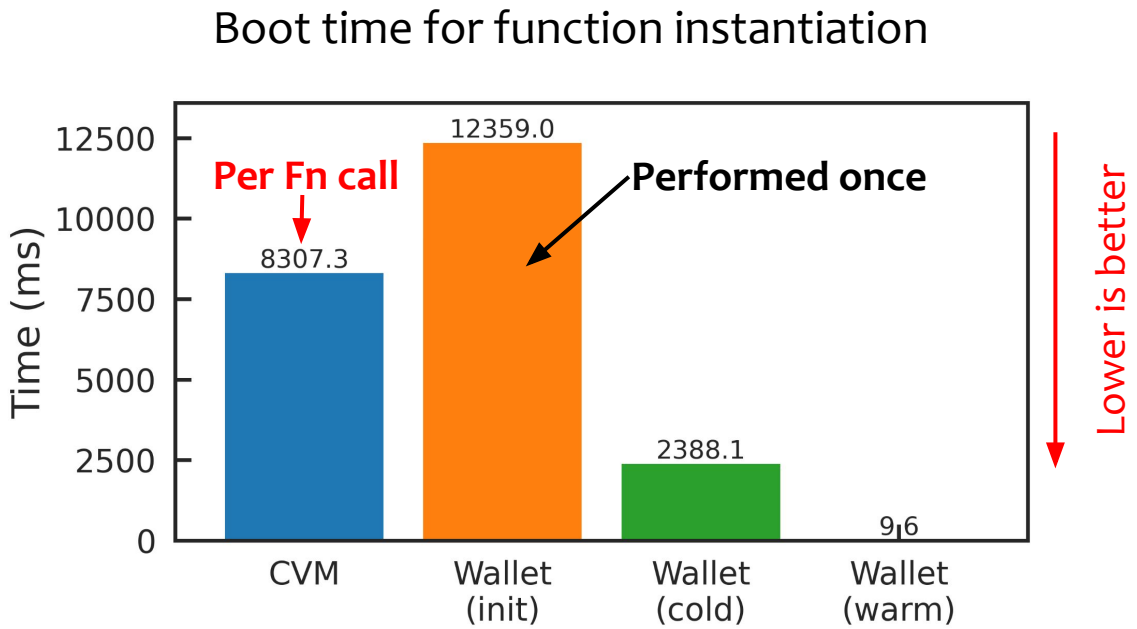
#1: Large TCB and high boot times



#1: Large TCB and high boot times



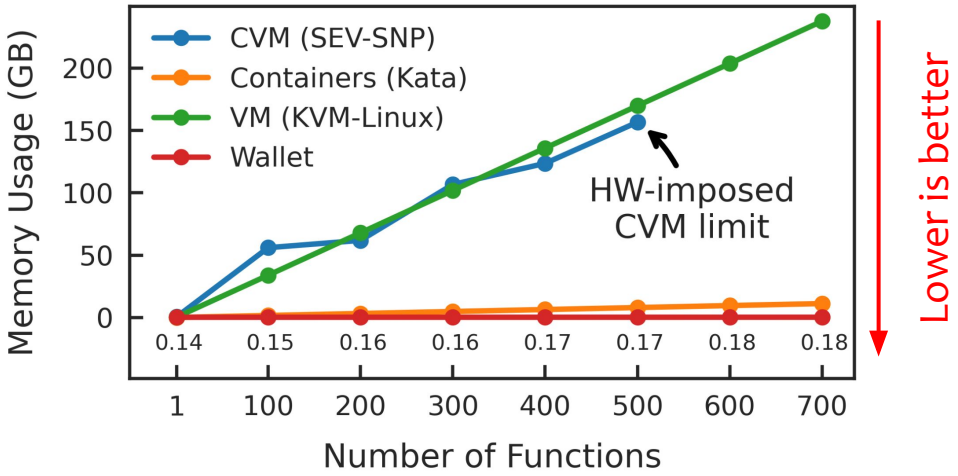
#1: Large TCB and high boot times



Almost negligible boot time

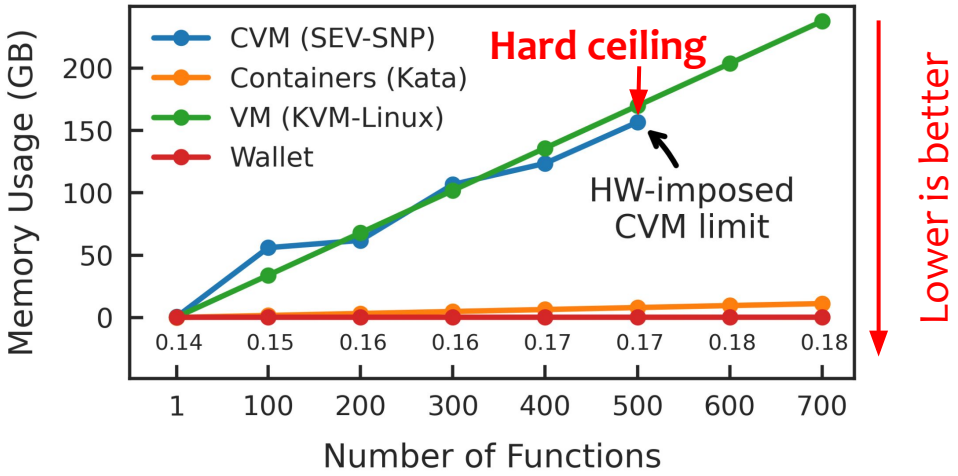
#2: Impractical function consolidation

Memory usage with increasing number of functions (1 Function / VM)



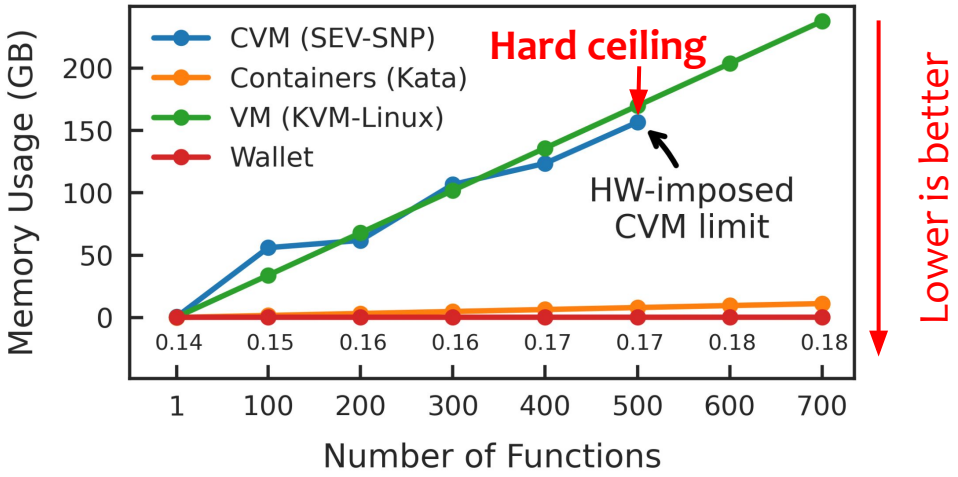
#2: Impractical function consolidation

Memory usage with increasing number of functions (1 Function / VM)



#2: Impractical function consolidation

Memory usage with increasing number of functions (1 Function / VM)

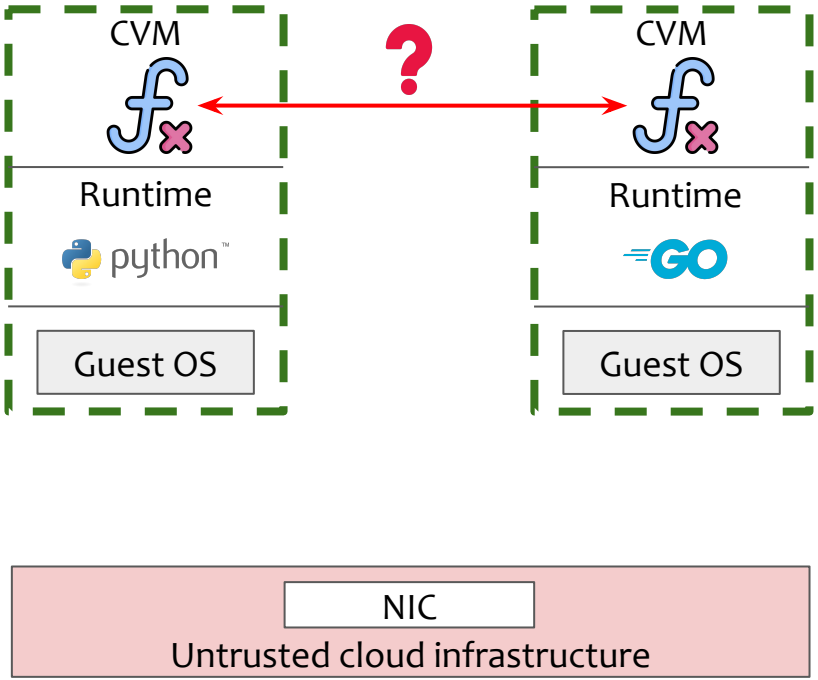


Up to **78%** improvement in memory consumption and **907×** higher deployment density

#3: High inter-function communication costs

CVM networking penalties:

- Disallowed DMA
- Bounce buffer copy overhead
- Crypto operations overhead

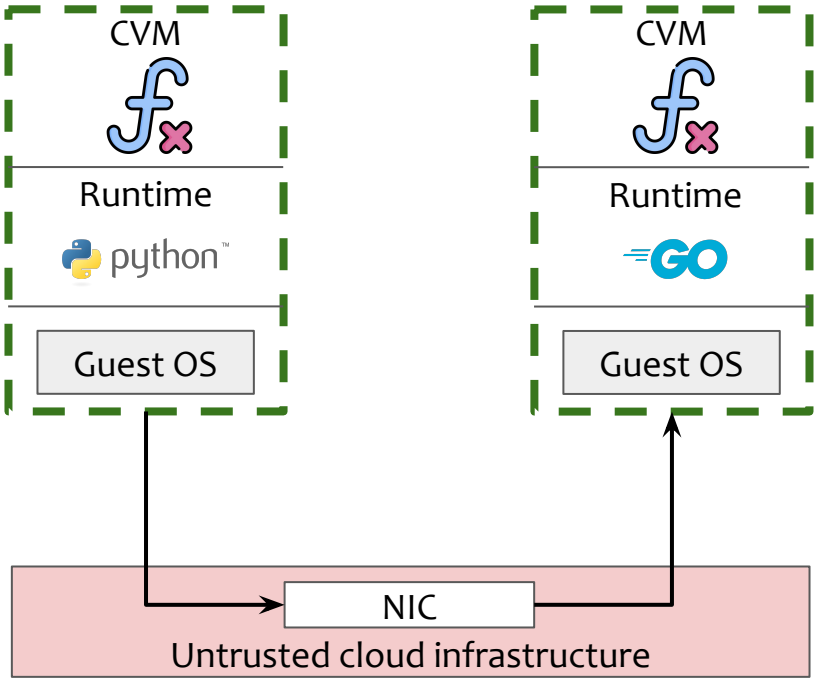


Credits: Flaticon.com

#3: High inter-function communication costs

CVM networking penalties:

- Disallowed DMA
- Bounce buffer copy overhead
- Crypto operations overhead

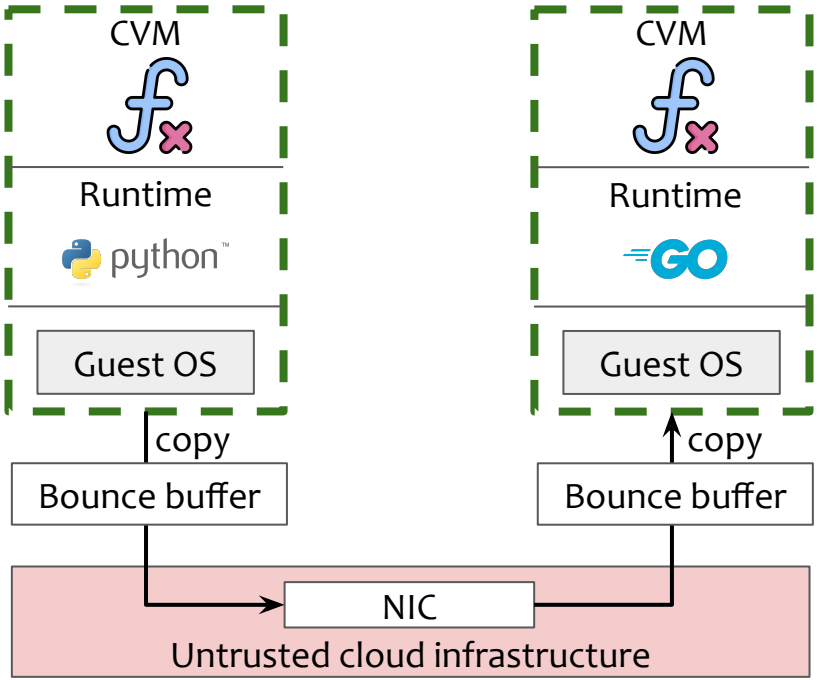


Credits: Flaticon.com

#3: High inter-function communication costs

CVM networking penalties:

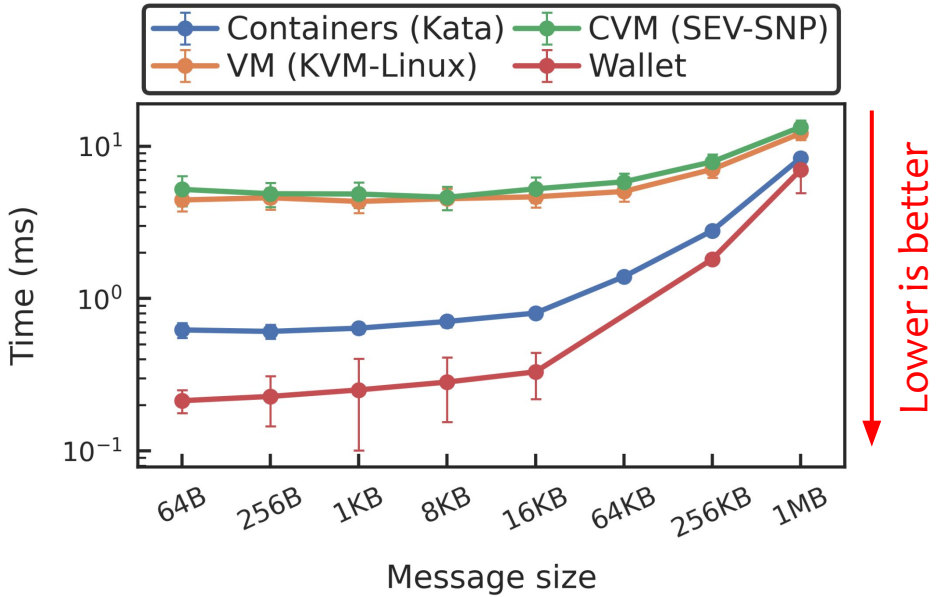
- Disallowed DMA
- Bounce buffer copy overhead
- Crypto operations overhead



Credits: Flaticon.com

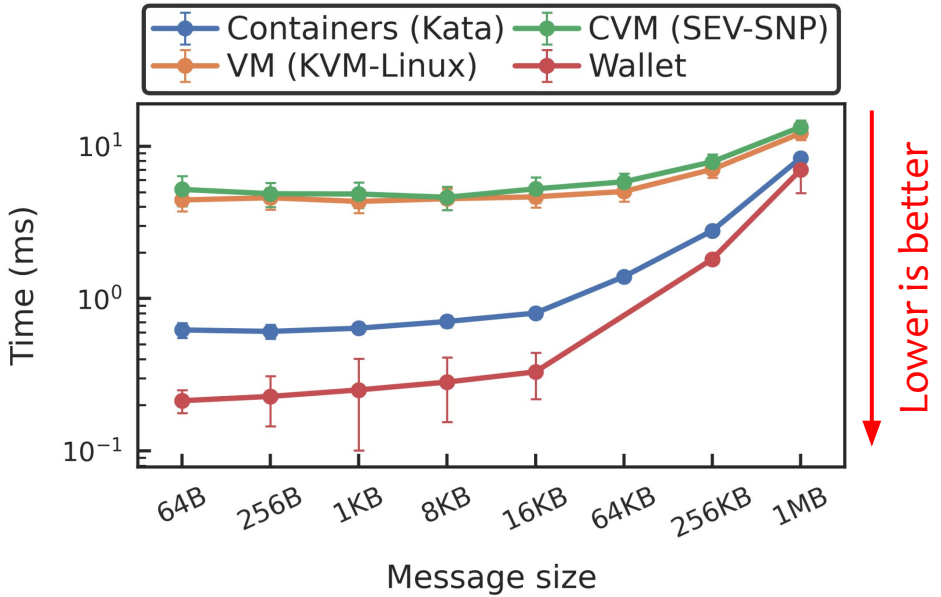
#3: High inter-function communication costs

Latency of data transfer between two functions with increasing message size



#3: High inter-function communication costs

Latency of data transfer between two functions with increasing message size

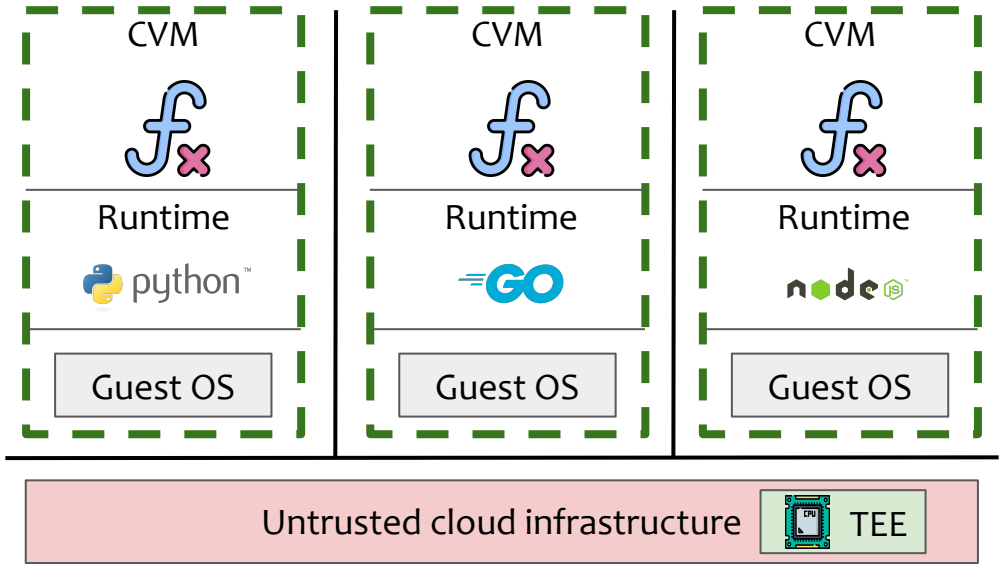


Up to **27x** lower data transfer latency compared to CVMs

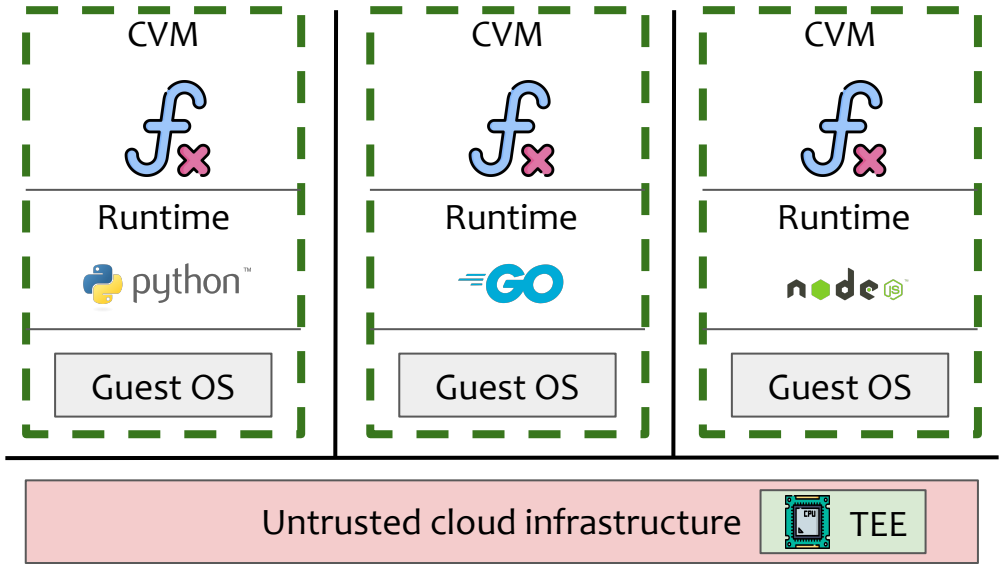
Outline

- ~~Motivation~~
- Key idea
- Design
- Implementation
- Evaluation

Key issue of CVM-based serverless computing

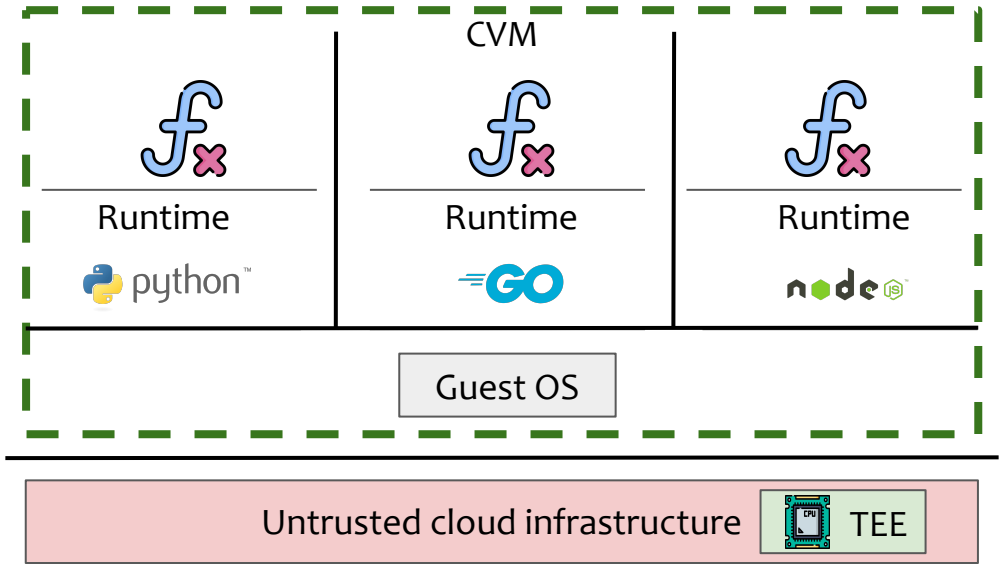


Key issue of CVM-based serverless computing

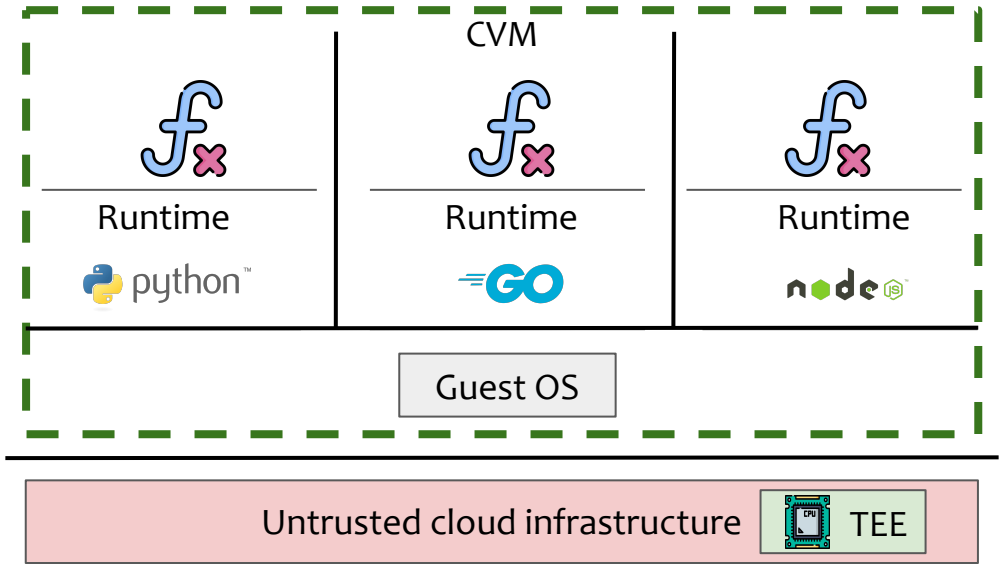


CVM is **ill-suited** as the deployment unit of a single serverless function

Research challenge



Research challenge

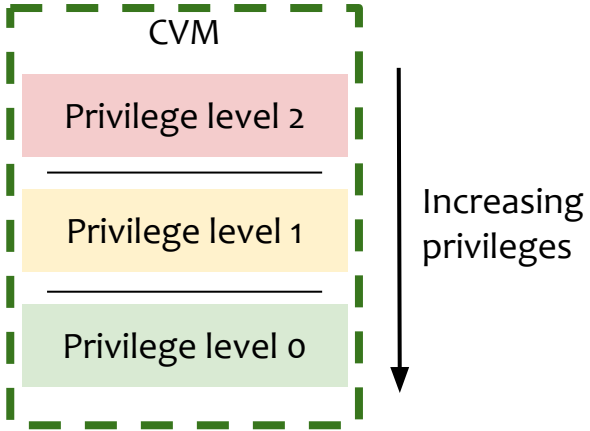


Can we pack **thousands, isolated** serverless functions in a **single CVM**?

Key idea: Nested virtualization

CVM partitioning

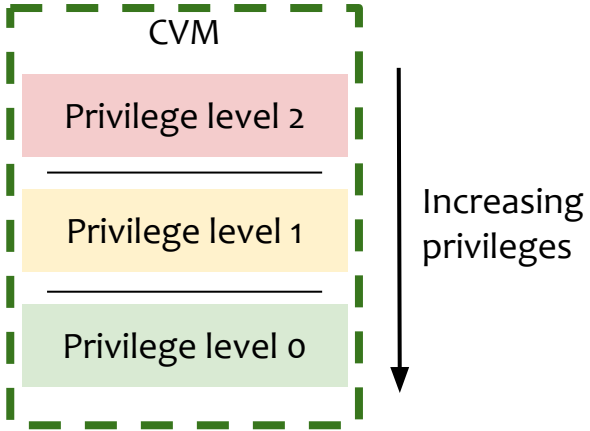
- Separation of CVM in distinct privilege levels
- HW-enforced access control
- Dynamic access level management



Key idea: Nested virtualization

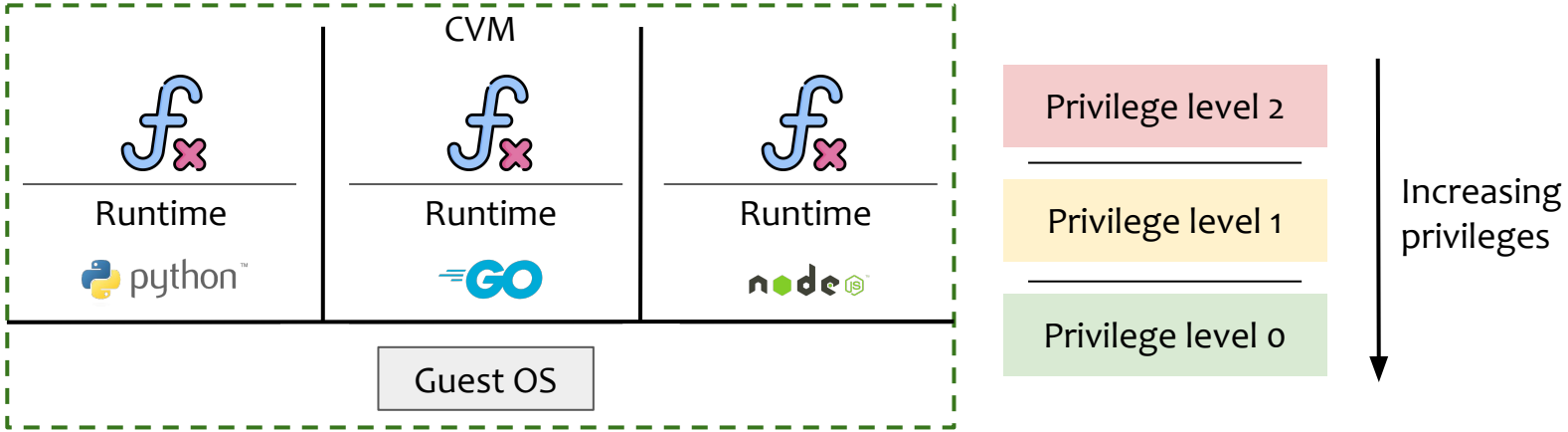
CVM partitioning

- Separation of CVM in distinct privilege levels
- HW-enforced access control
- Dynamic access level management

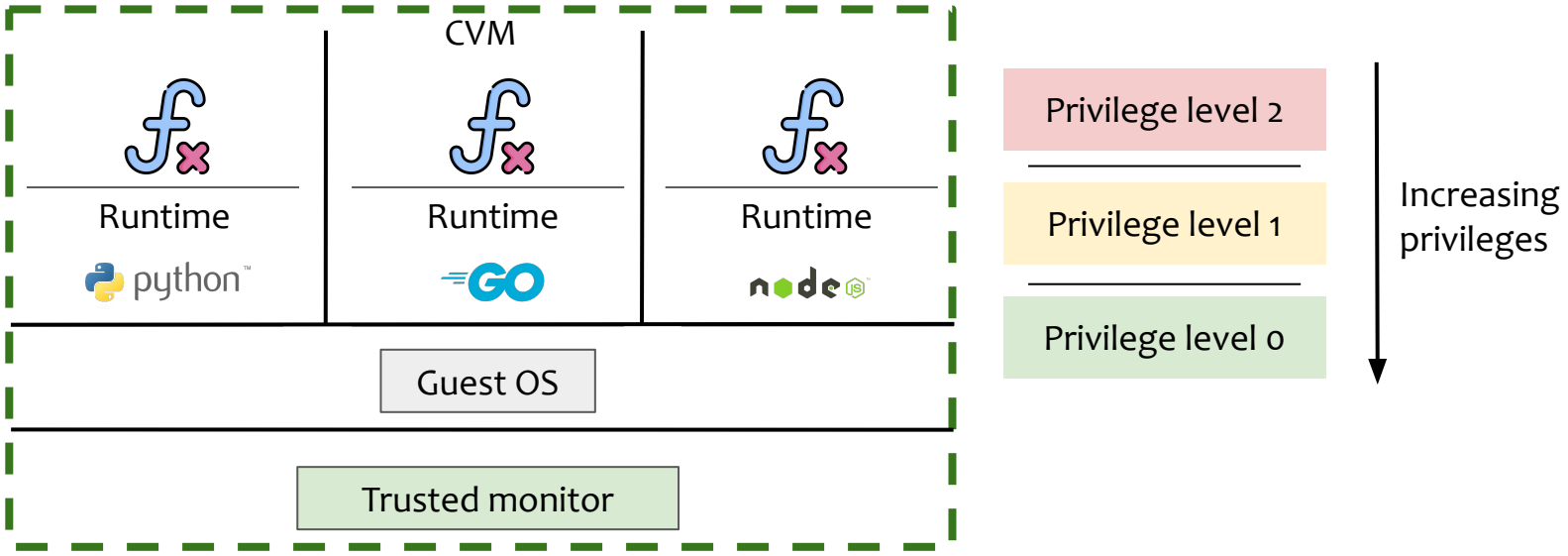


Leverage **CVM partitioning** to split a CVM into smaller isolated compartments

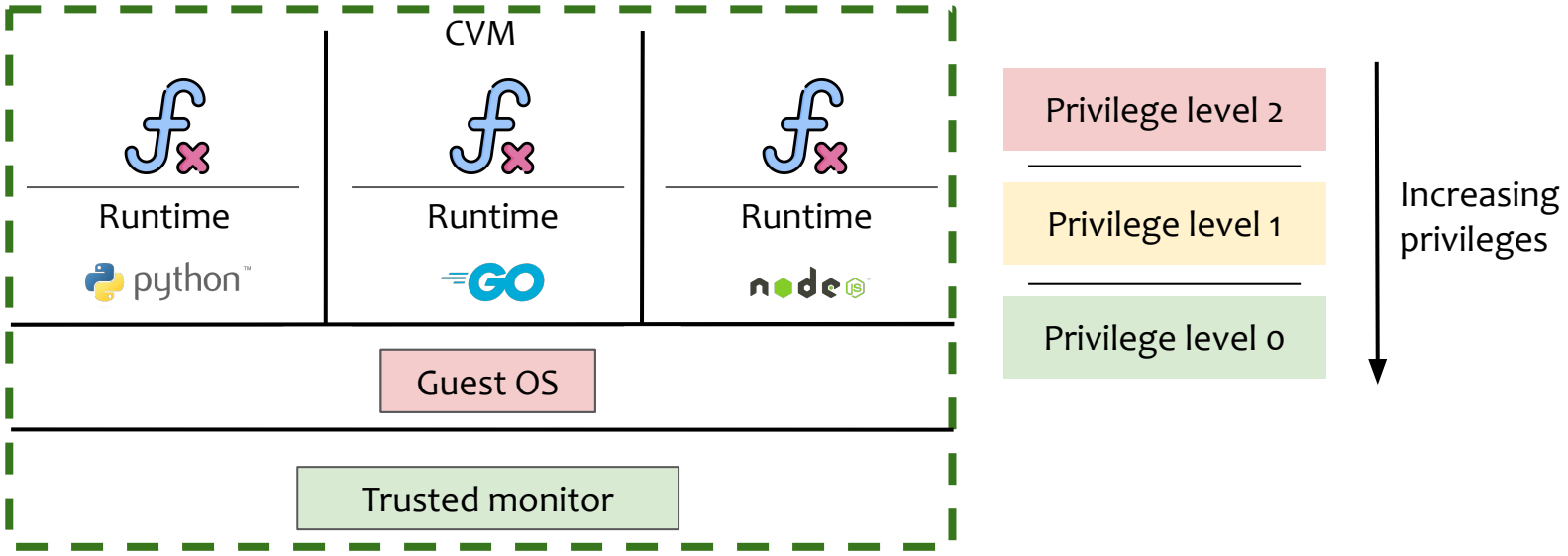
CVM partitioning for serverless computing



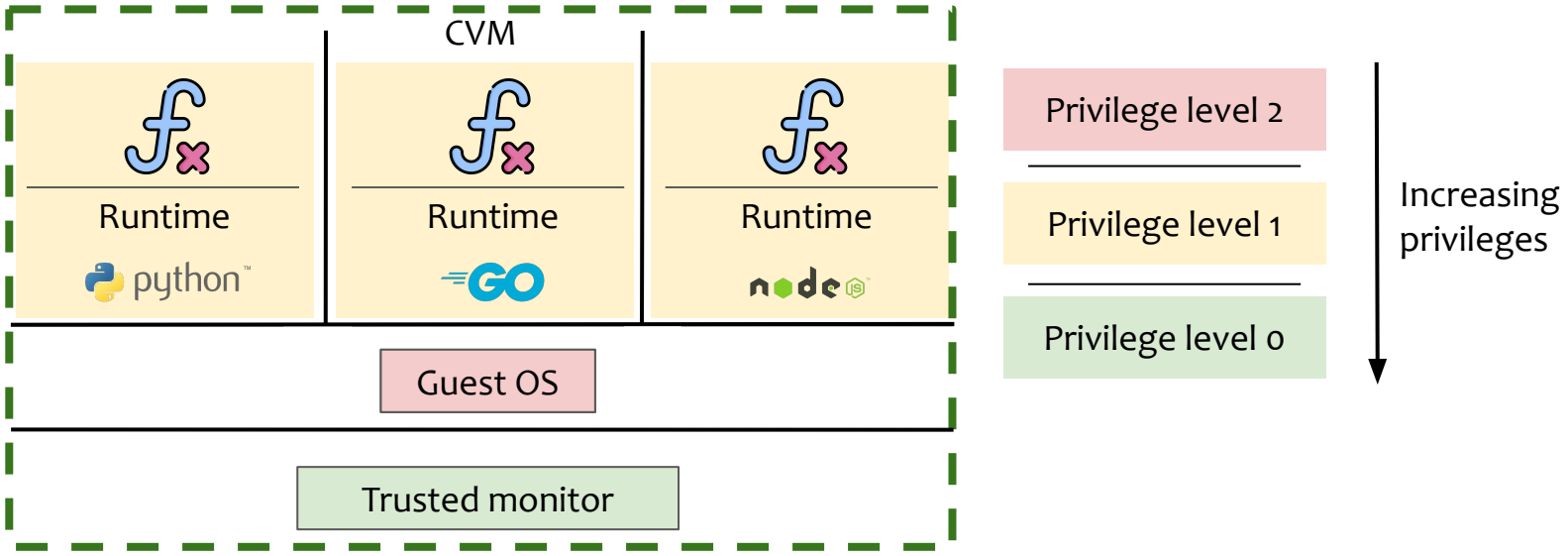
CVM partitioning for serverless computing



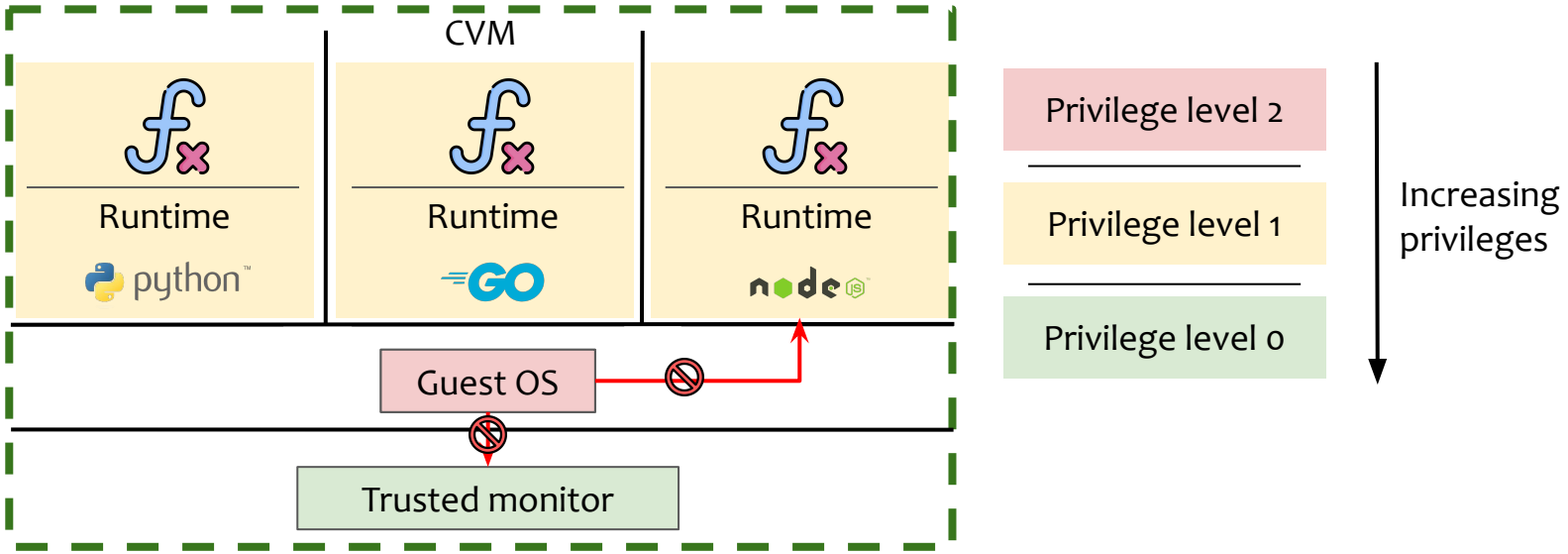
CVM partitioning for serverless computing



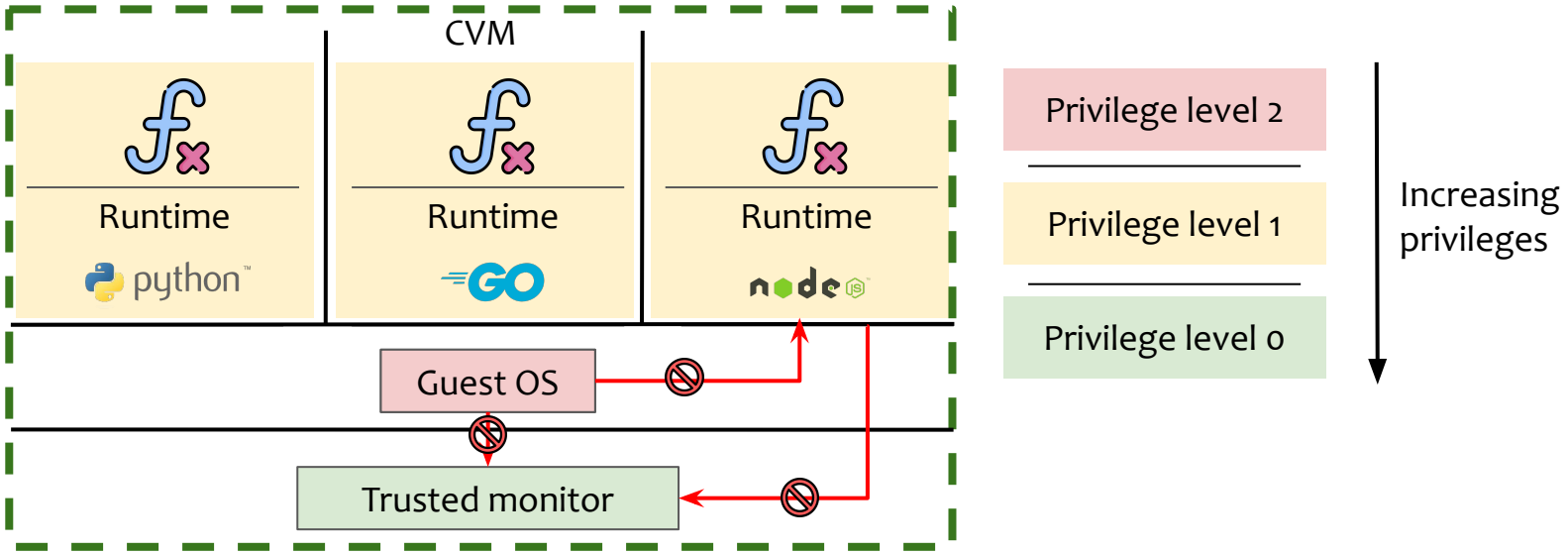
CVM partitioning for serverless computing



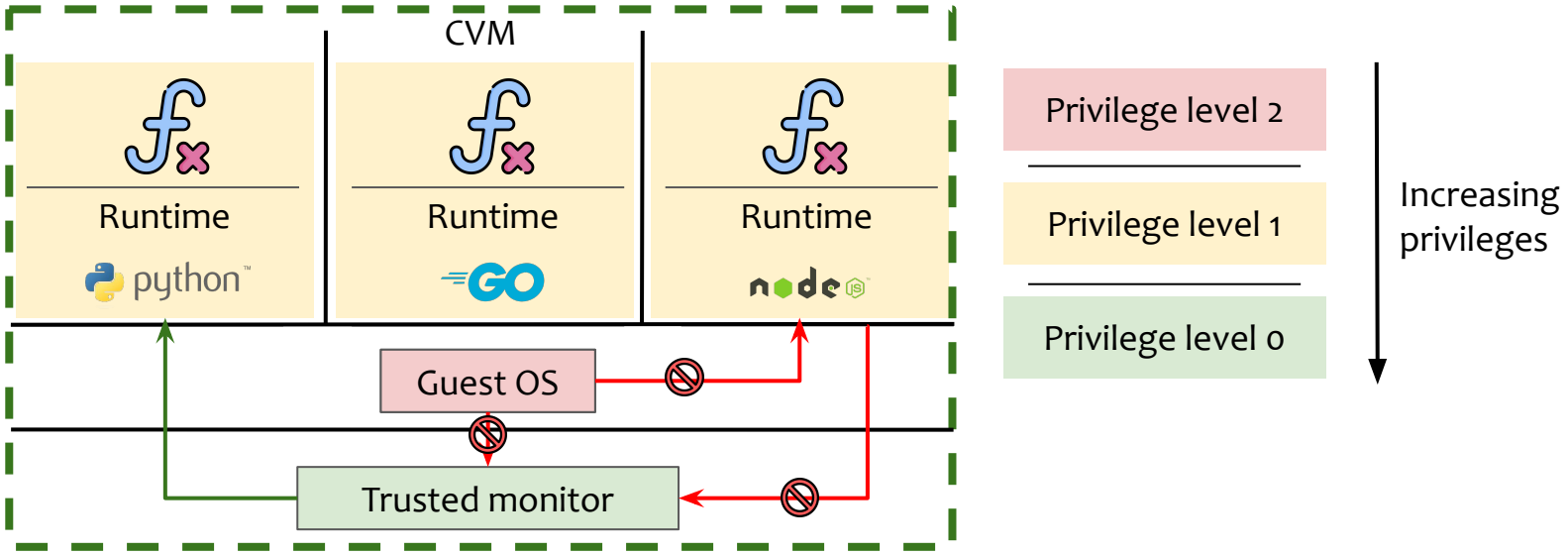
CVM partitioning for serverless computing



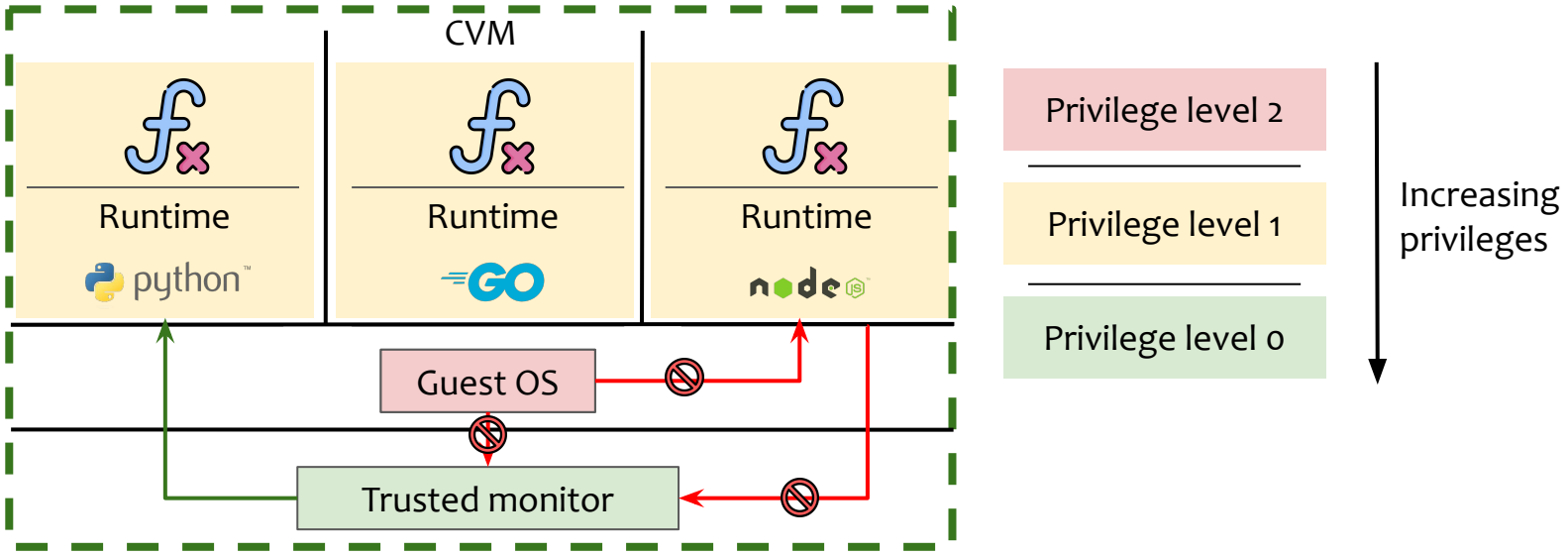
CVM partitioning for serverless computing



CVM partitioning for serverless computing



CVM partitioning for serverless computing

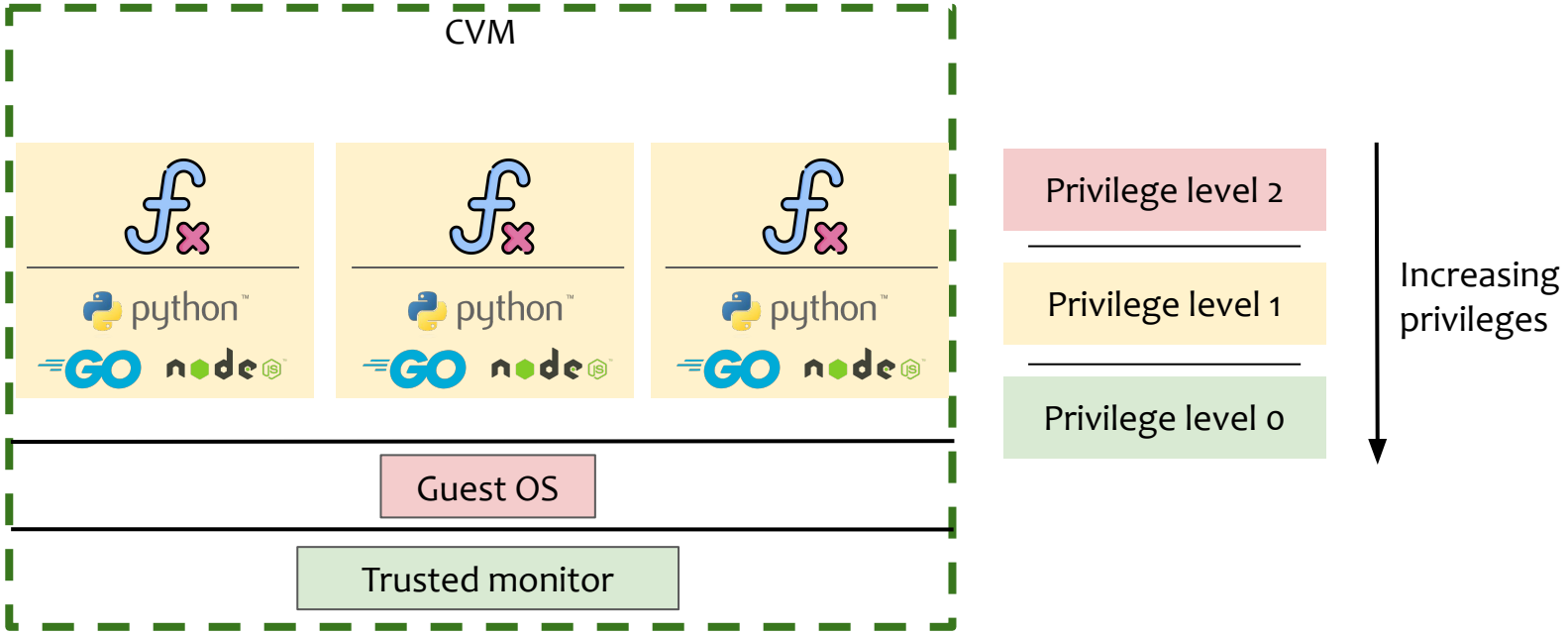


Serverless functions are **isolated** among themselves and **decoupled** from the GuestOS

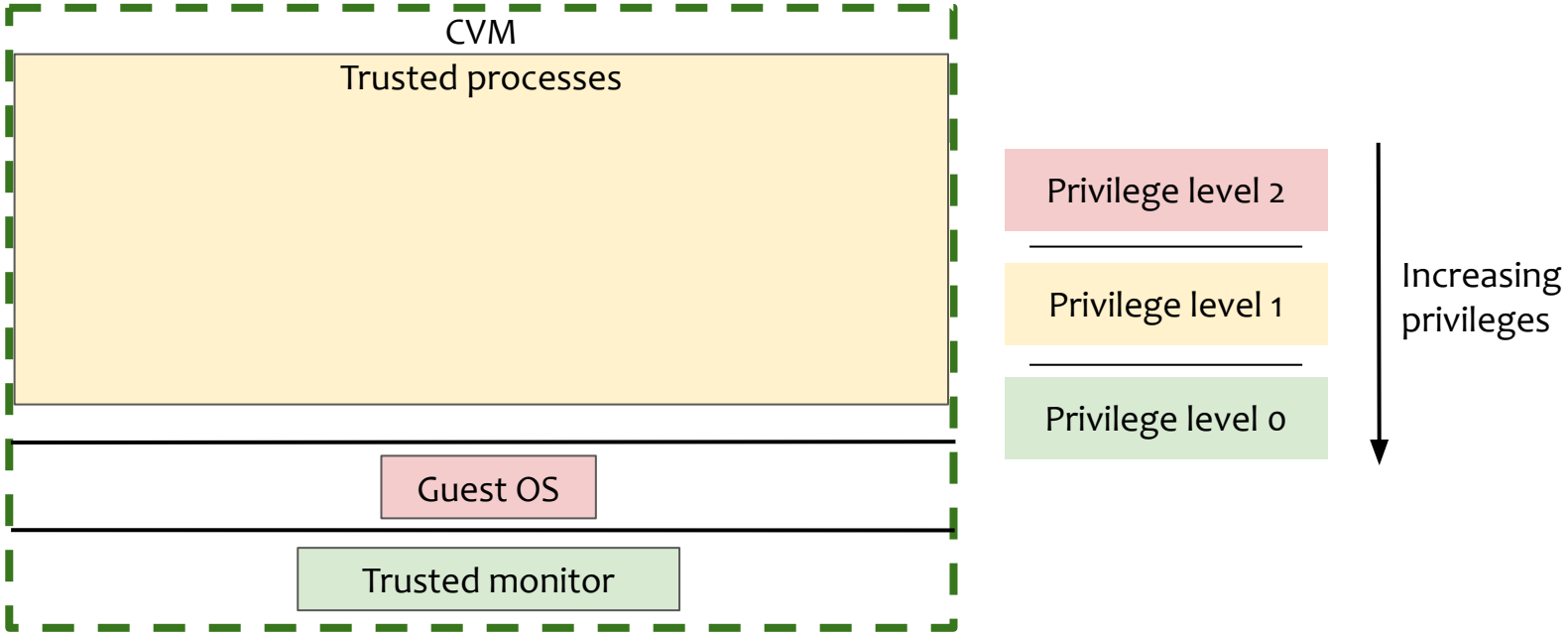
Outline

- ~~Motivation~~
- ~~Key idea~~
- Design
- Implementation
- Evaluation

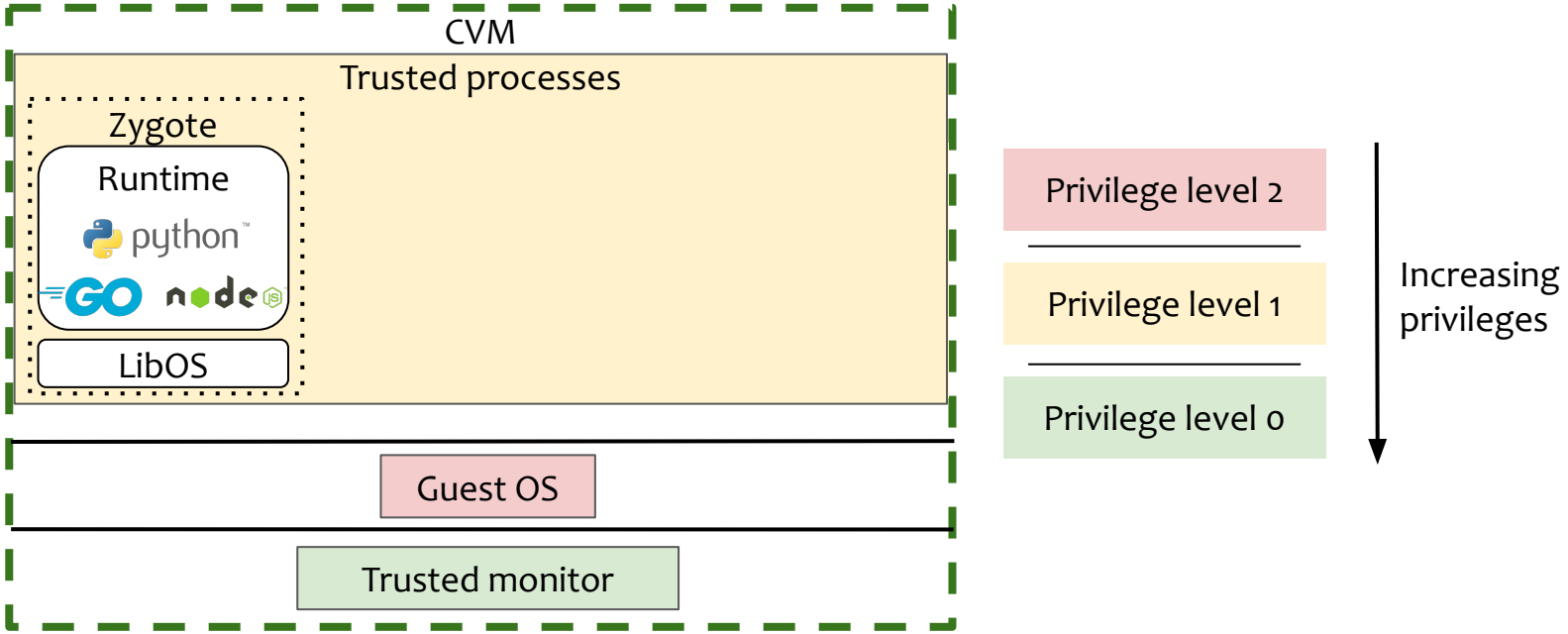
Wallet: Confidential serverless computing



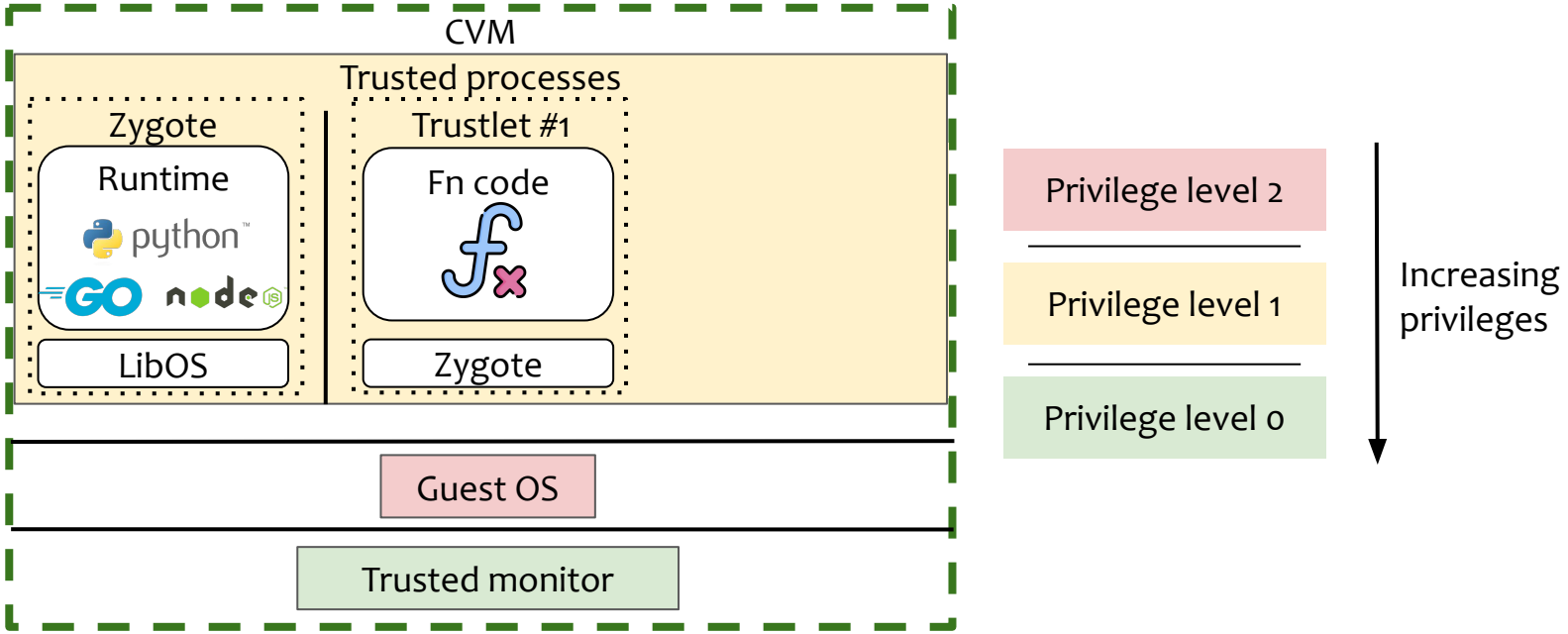
Wallet: Confidential serverless computing



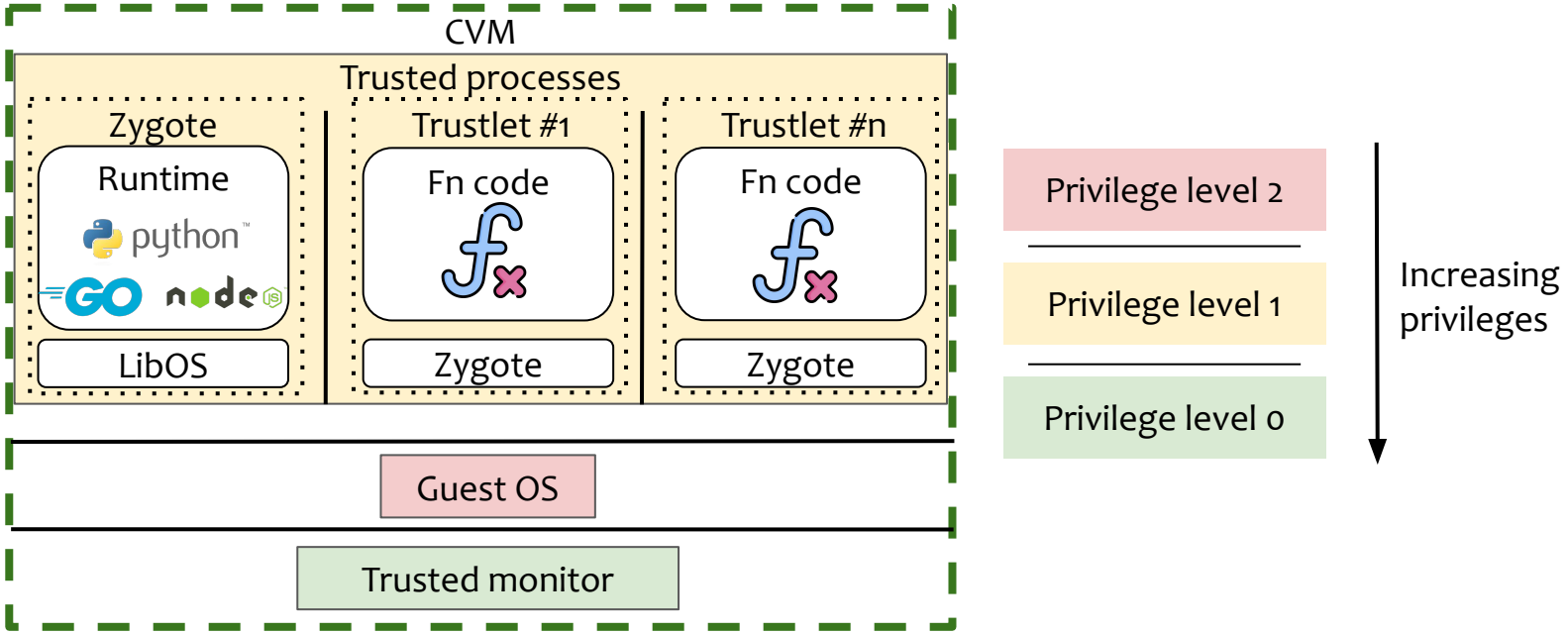
Wallet: Confidential serverless computing



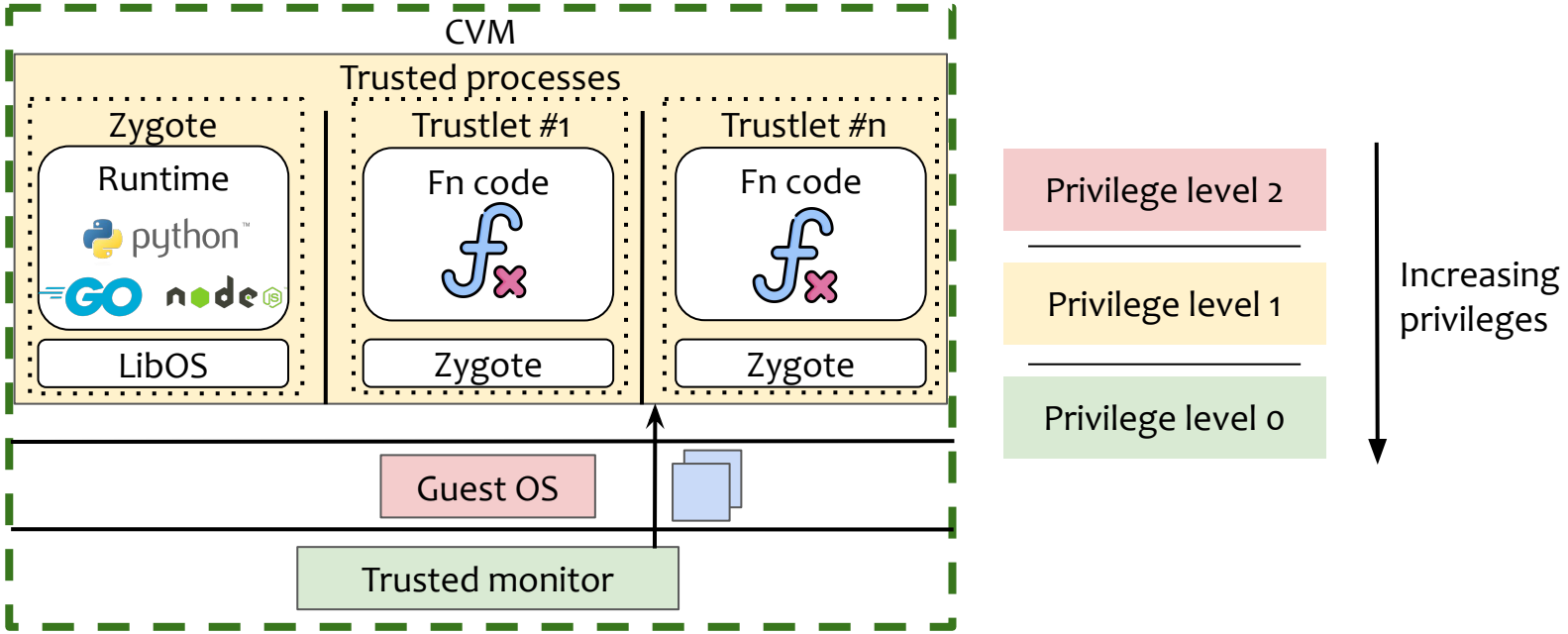
Wallet: Confidential serverless computing



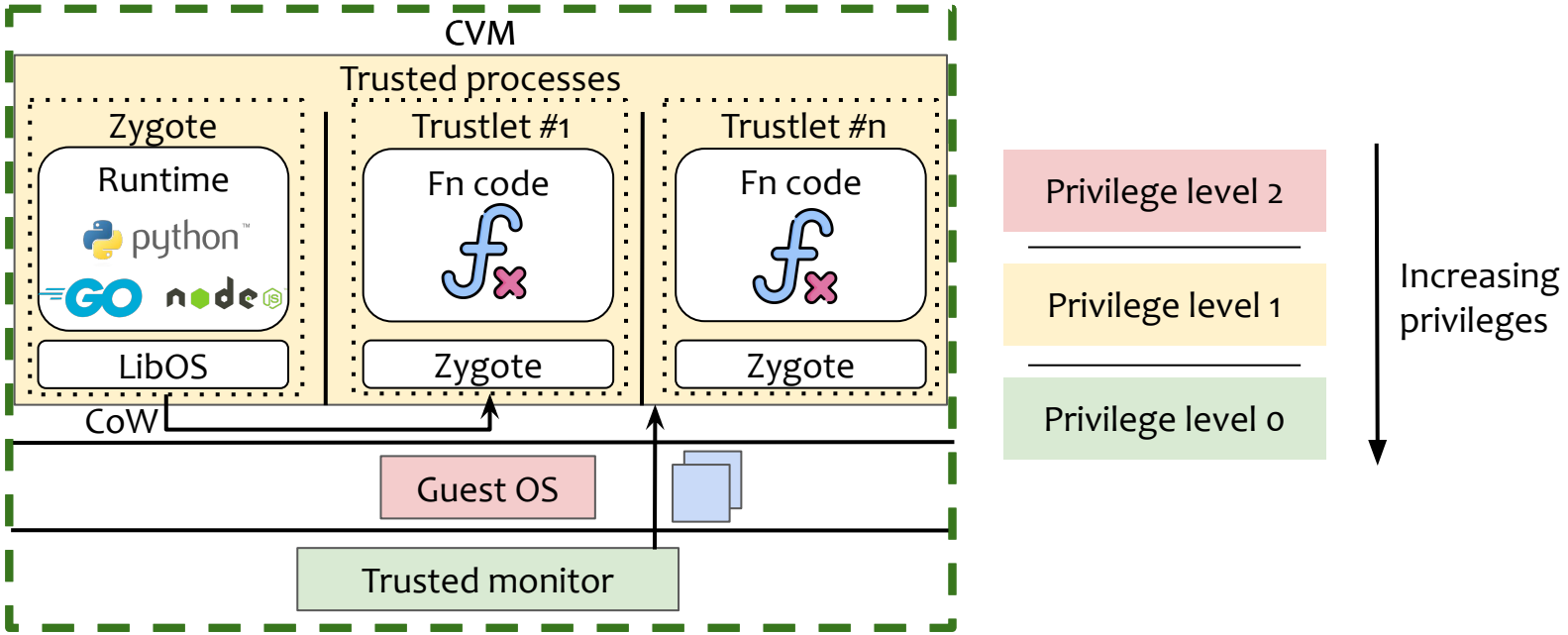
Wallet: Confidential serverless computing



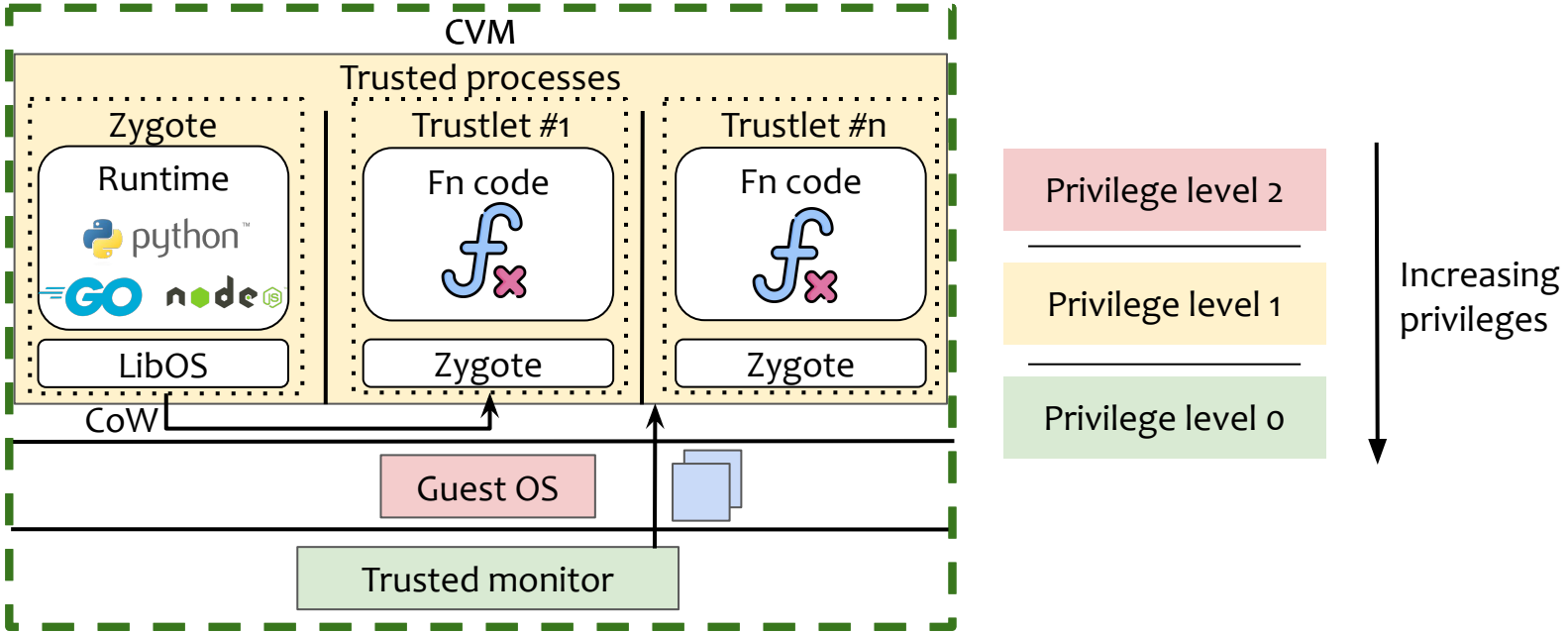
Wallet: Confidential serverless computing



Wallet: Confidential serverless computing



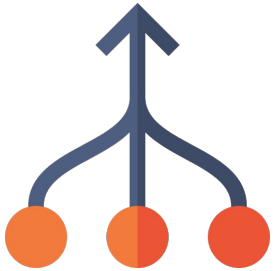
Wallet: Confidential serverless computing



Wallet re-architects the **CVM** based on the **requirements of serverless computing**

Design challenges

#1: Large Trusted Computing Base (TCB) and high boot times

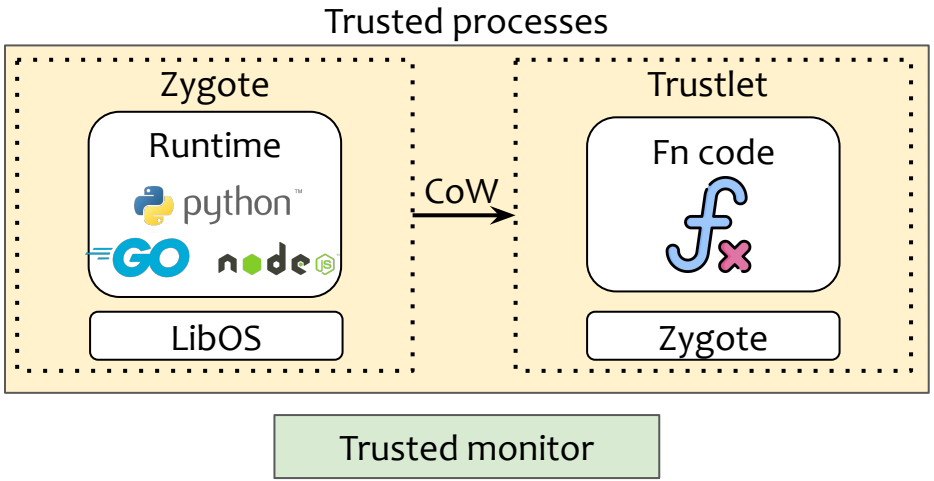


#2: Impractical function consolidation

#3: High inter-function communication costs

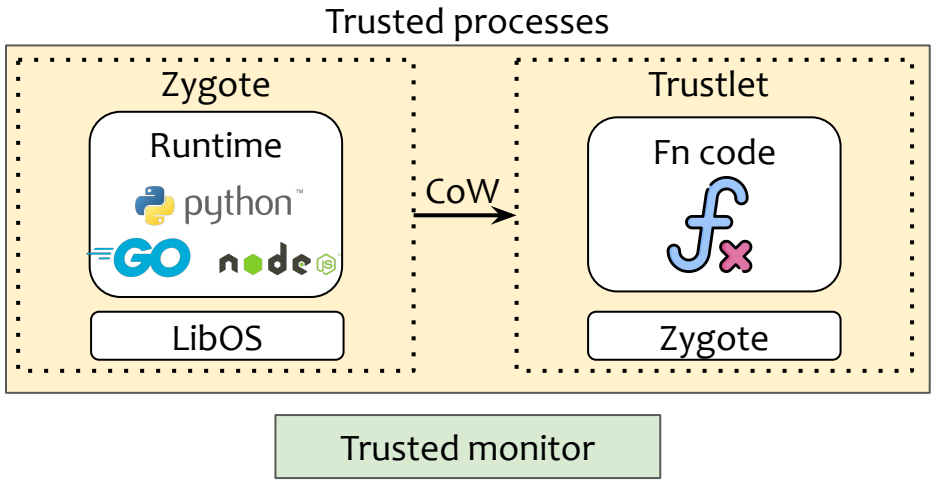
#1 Low TCB and boot time

- Trusted Monitor
  - Minimal software component
- Zygote
  - LibOS-based
  - Initialized once
- Trustlet
  - Derived from a Zygote
  - Copy-on-Write



#1 Low TCB and boot time

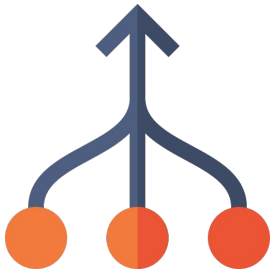
- Trusted Monitor
  - Minimal software component
- Zygote
  - LibOS-based
  - Initialized once
- Trustlet
  - Derived from a Zygote
  - Copy-on-Write



The **minimal** trusted monitor along with **CoW** minimize the TCB and boot time

Design challenges

#1: Large Trusted Computing Base (TCB) and high boot times

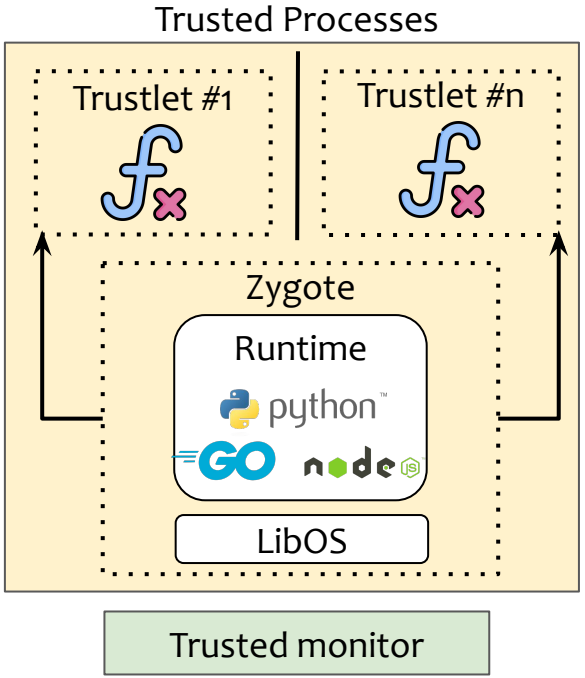


#2: Impractical function consolidation

#3: High inter-function communication costs

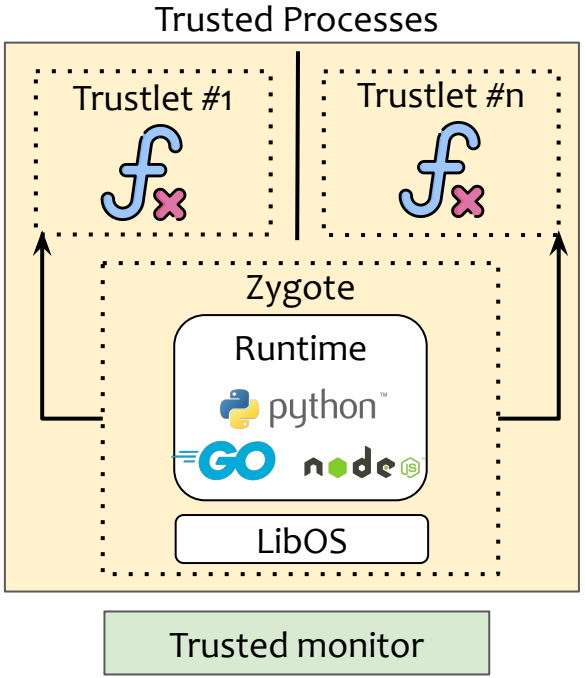
#2 Practical function consolidation

- Trusted monitor
  - Isolation via page tables
  - Memory deduplication
- Zygote
  - Shared between multiple trustlets
  - Static in memory
- Trustlet
  - Contains only the function code



#2 Practical function consolidation

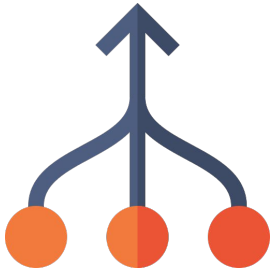
- Trusted monitor
  - Isolation via page tables
  - Memory deduplication
- Zygote
  - Shared between multiple trustlets
  - Static in memory
- Trustlet
  - Contains only the function code



Software-based isolation and **memory deduplication** overcome the CVM density limit

Design challenges

#1: Large Trusted Computing Base (TCB) and high boot times

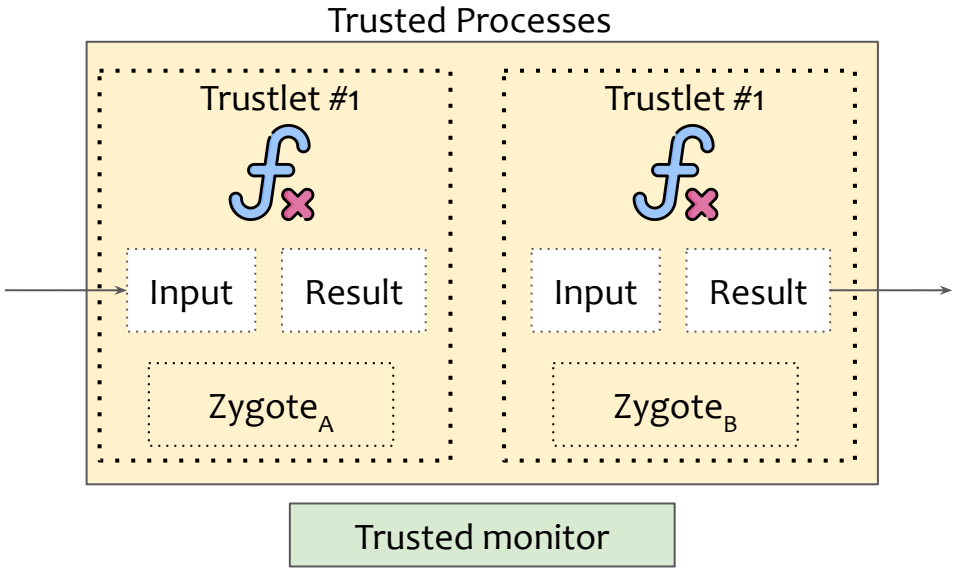


#2: Impractical function consolidation

#3: High inter-function communication costs

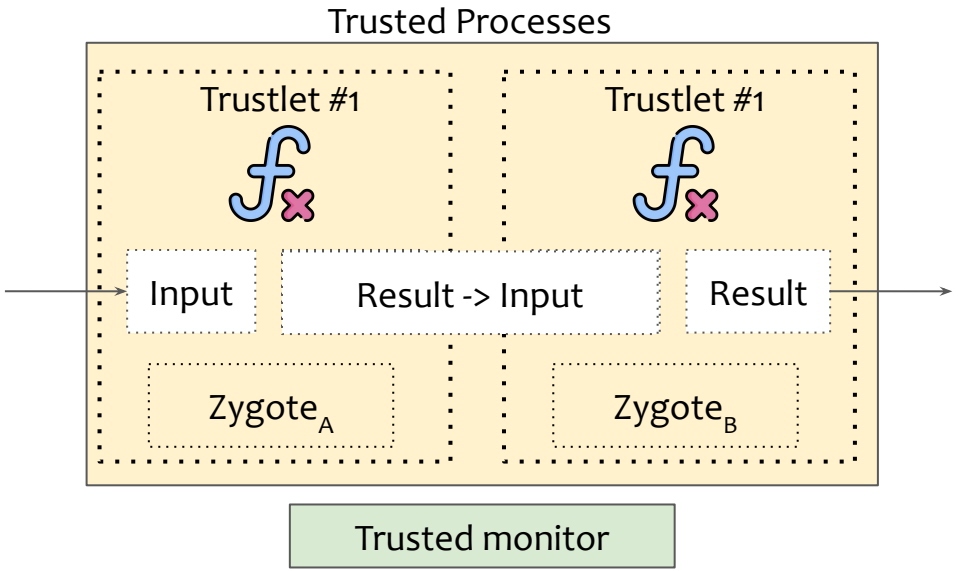
#3 Efficient inter-function communication

- Shared-memory data objects
- Configured by the trusted monitor
- Zero-copy function chaining



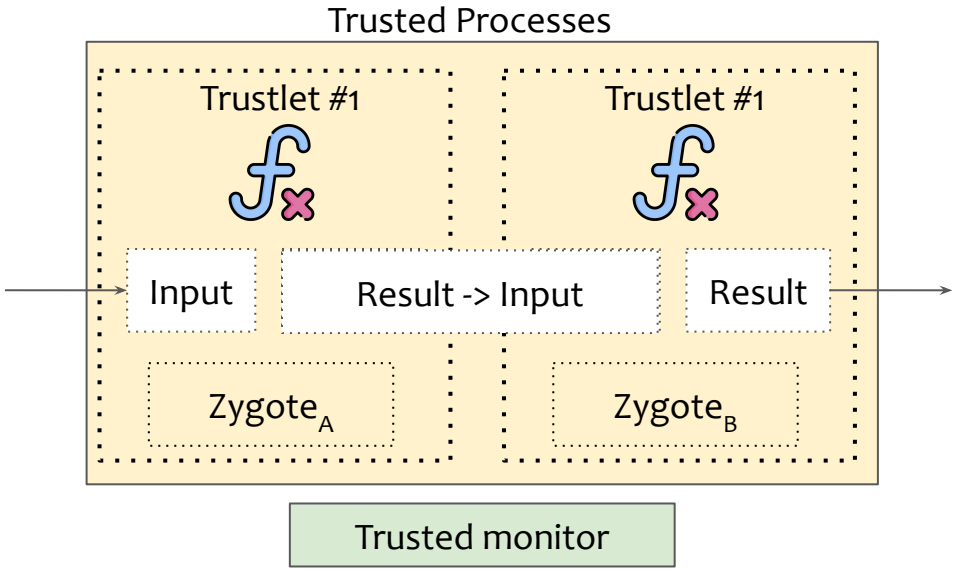
#3 Efficient inter-function communication

- Shared-memory data objects
- Configured by the trusted monitor
- Zero-copy function chaining



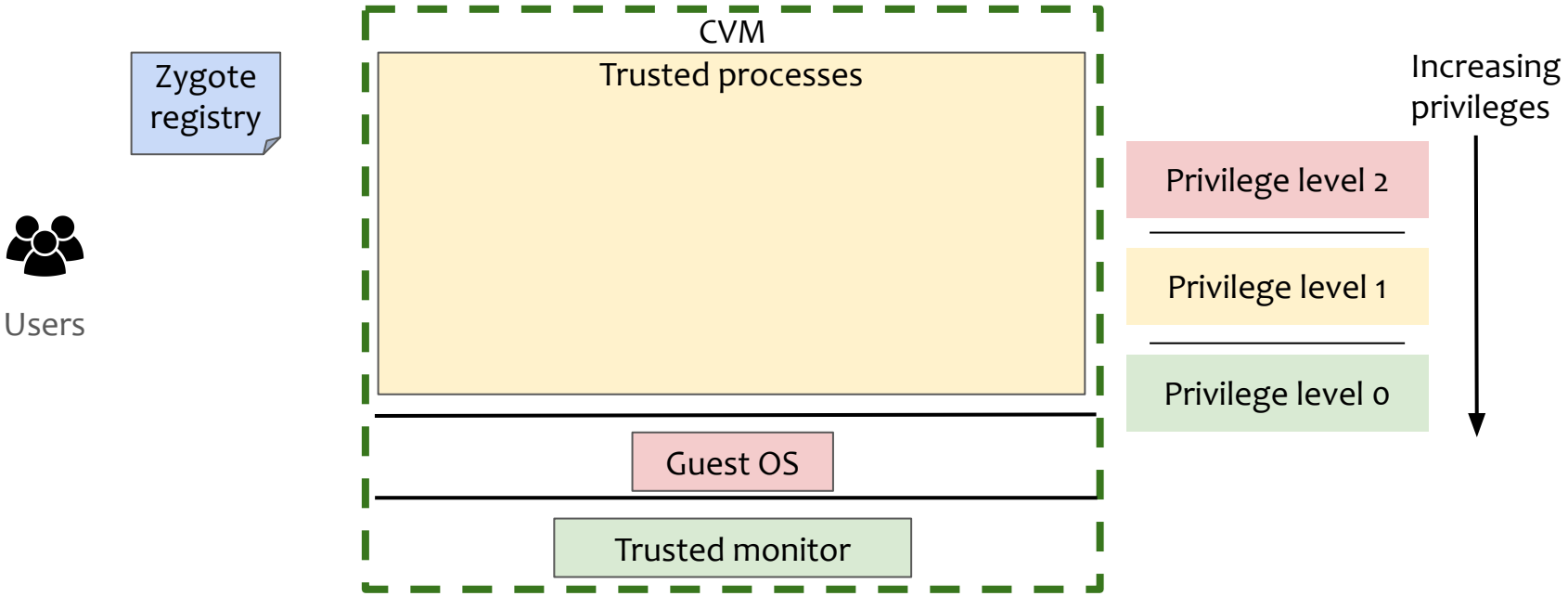
#3 Efficient inter-function communication

- Shared-memory data objects
- Configured by the trusted monitor
- Zero-copy function chaining

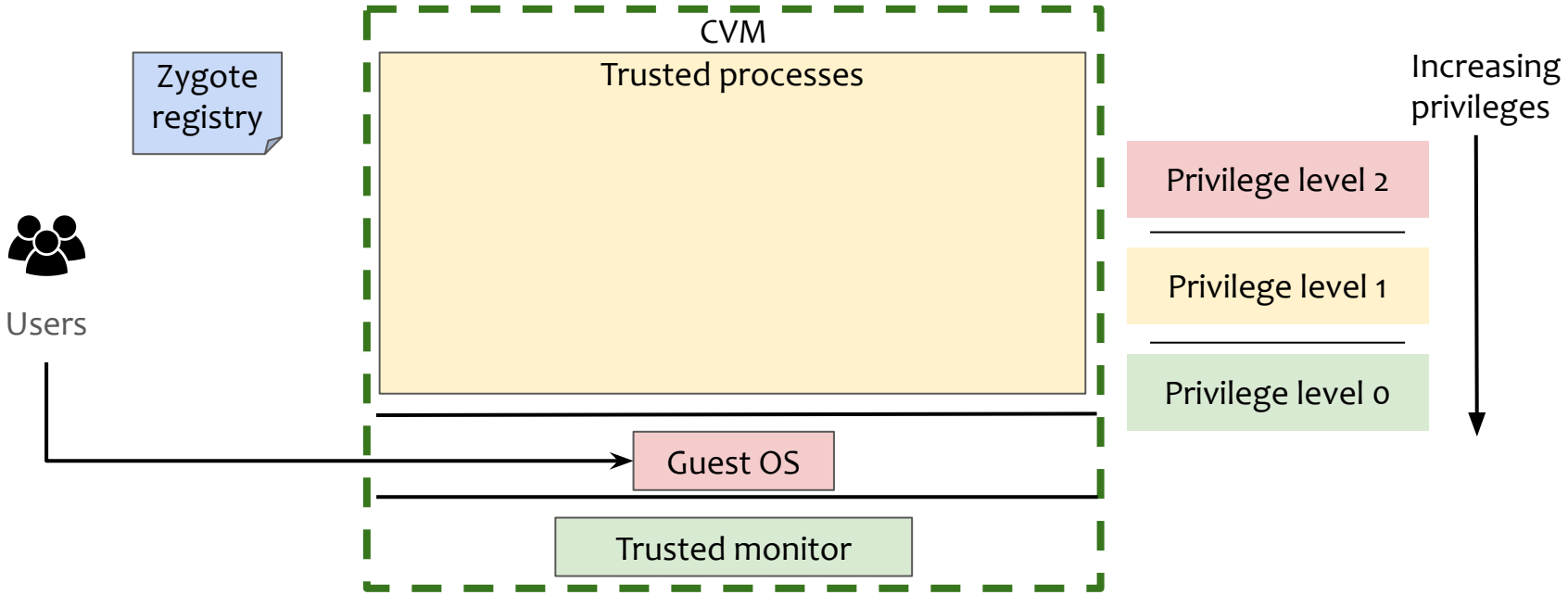


Shared-memory data objects optimize chaining of co-located functions

Deployment workflow - Trustlet

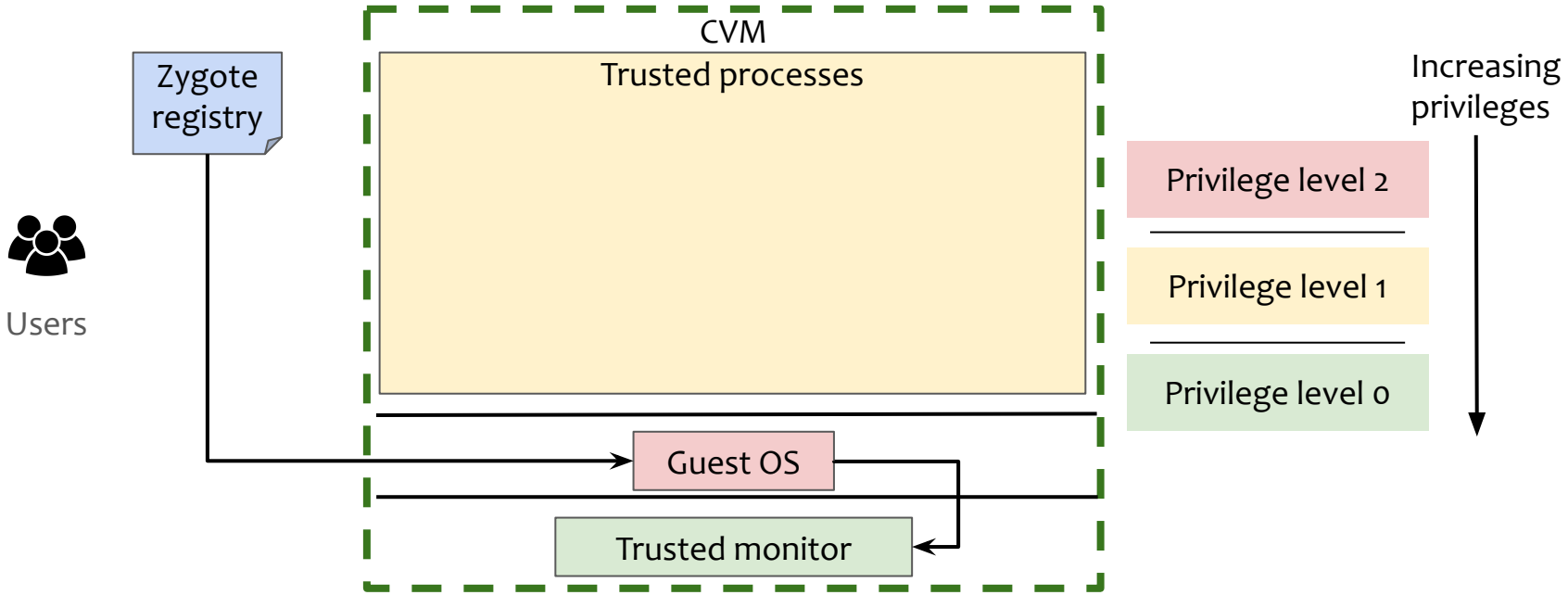


Deployment workflow - Trustlet



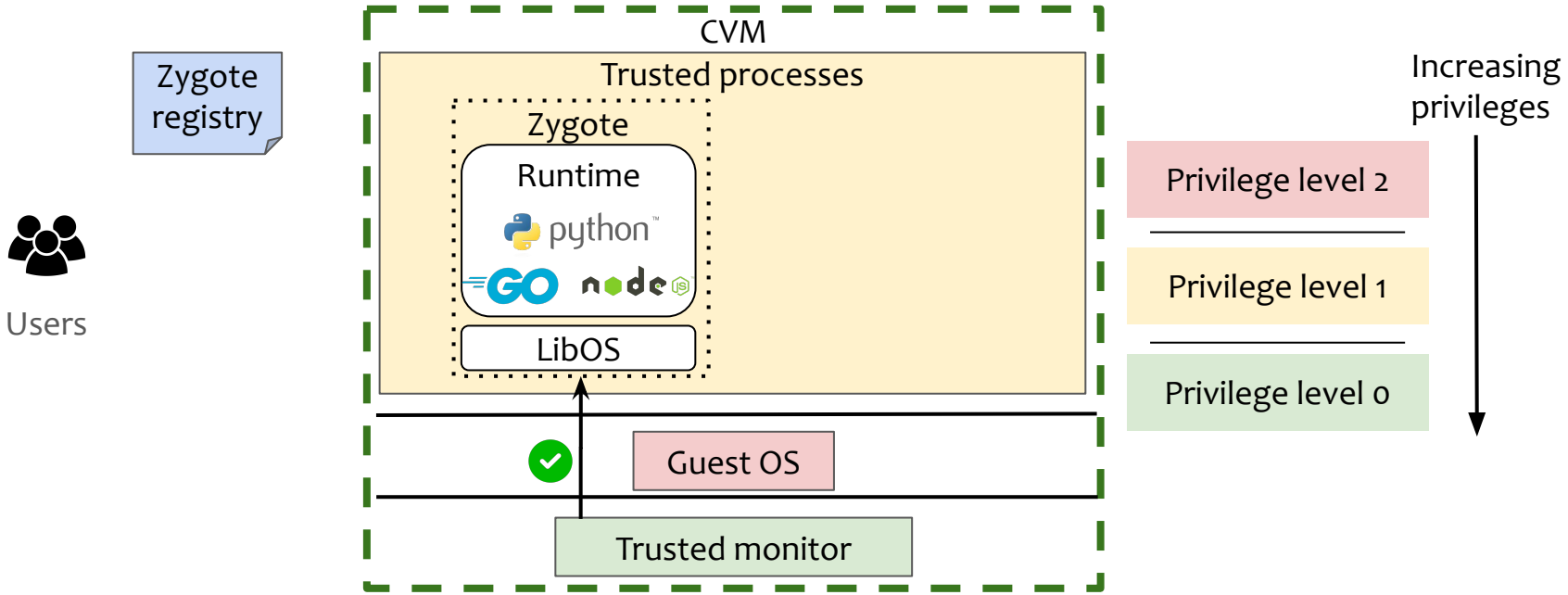
Deployment workflow - Trustlet

1. Fetch zygote from registry and register with monitor (cold start)



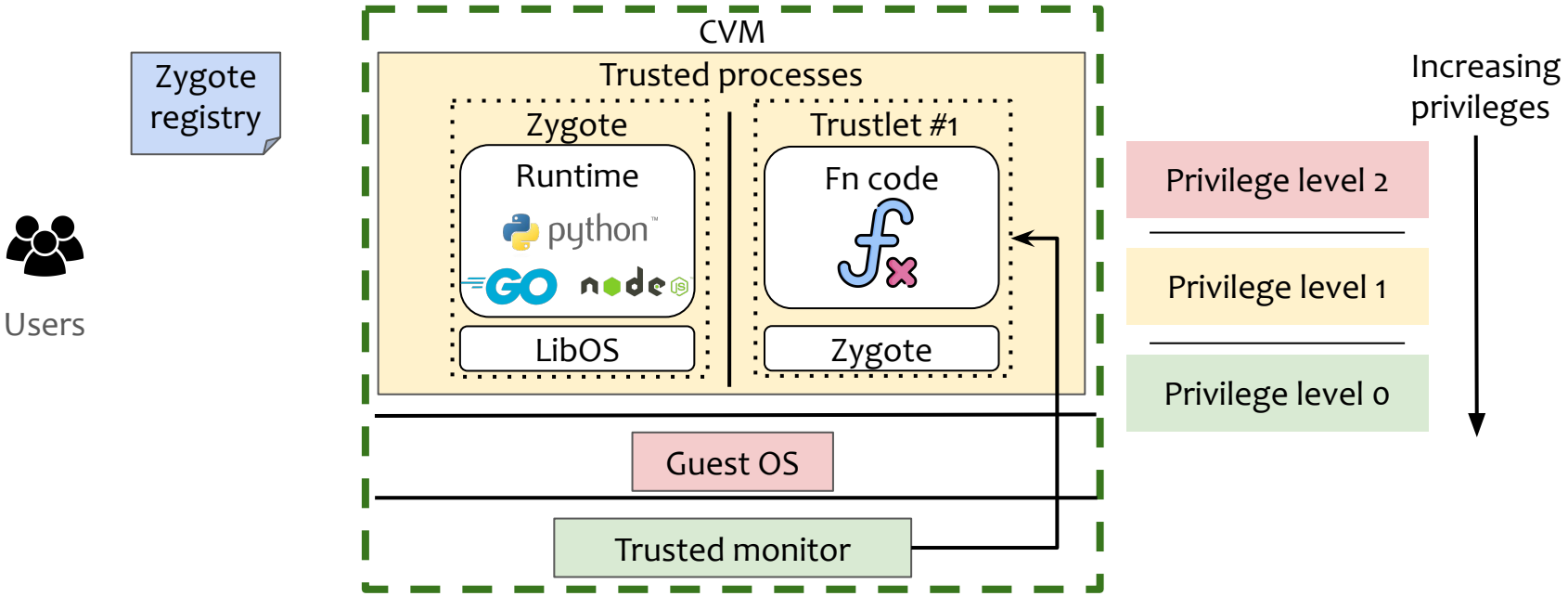
Deployment workflow - Trustlet

1. Fetch zygote from registry and register with monitor (cold start)



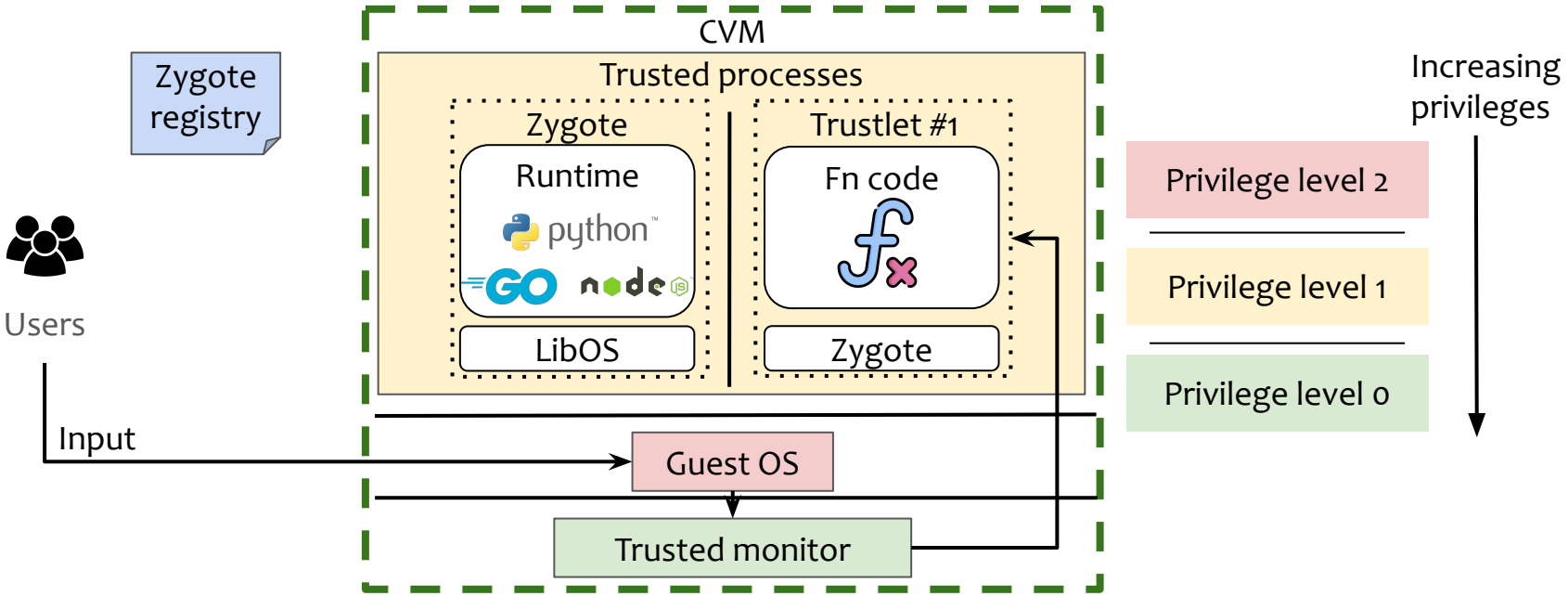
Deployment workflow - Trustlet

2. Create trustlet with function data (lukewarm start)



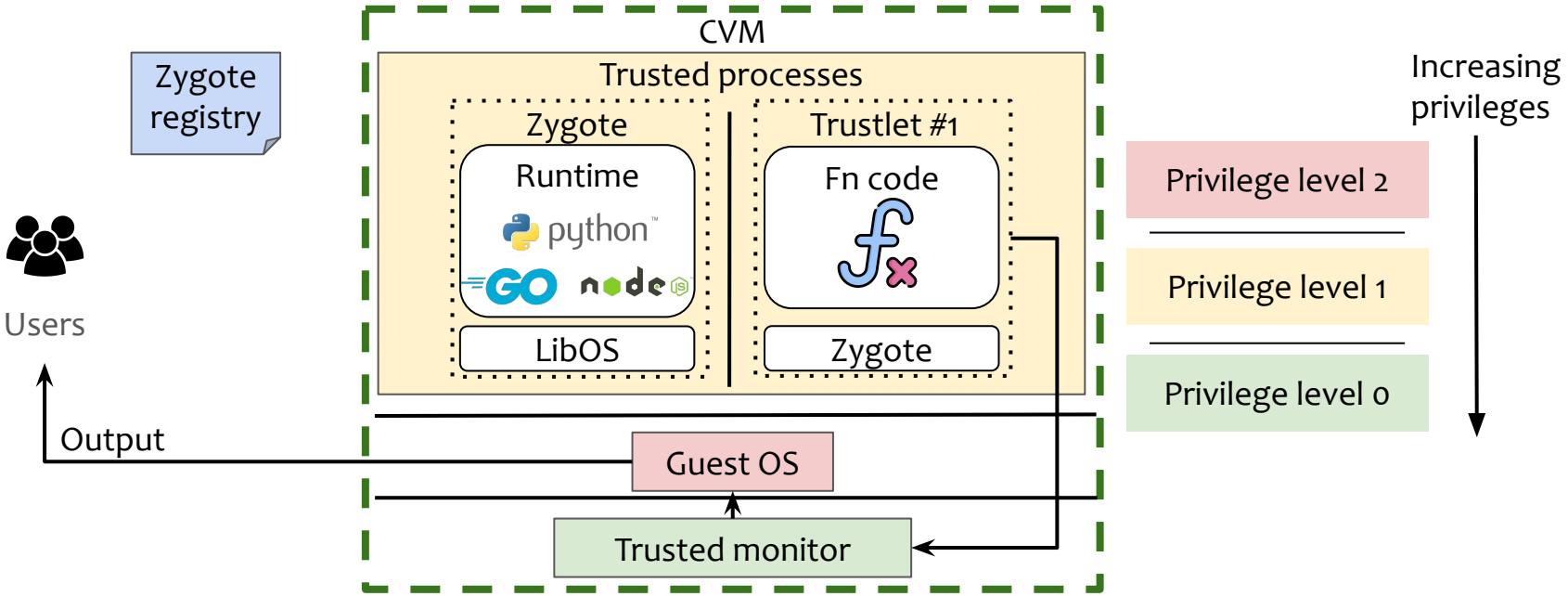
Deployment workflow - Trustlet

3. Invoke trustlet with user input (warm start)



Deployment workflow - Trustlet

3. Invoke trustlet with user input (warm start)

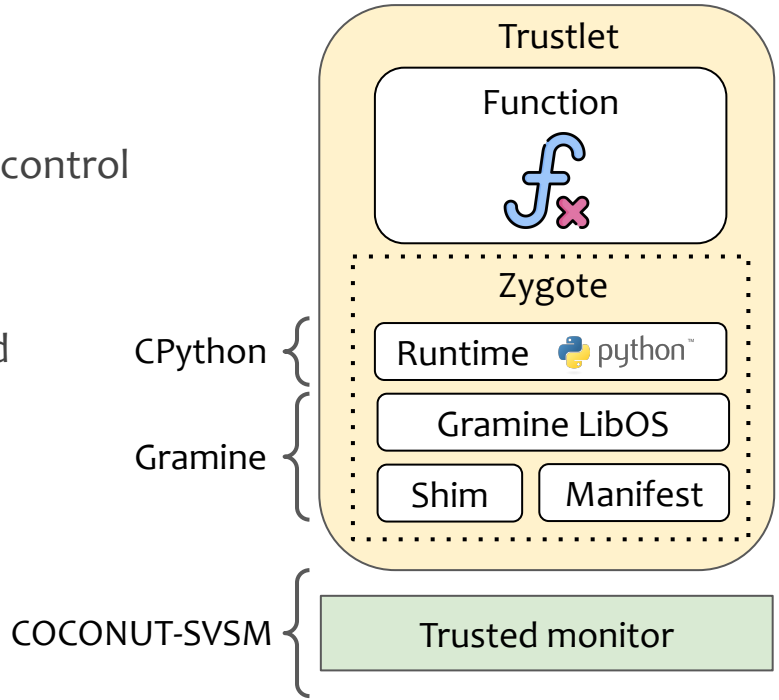


Outline

- ~~Motivation~~
- ~~Key idea~~
- ~~Design~~
- Implementation
- Evaluation

Implementation

- Monitor extends **COCONUT-SVSM**
  - New monitor and communication API calls
  - Interface for privilege-level-based process control
- **Gramine** as our chosen libOS
  - Bundle a static filesystem
  - Manifest specifies how application is called
- **CPython**-based runtime
  - Provides runtime via static filesystem



[1] Coconut-svsm: https://github.com/coconut-svsm/svsm
[2] Gramine: https://gramineproject.io/

Outline

- ~~Motivation~~
- ~~Key idea~~
- ~~Design~~
- ~~Implementation~~
- Evaluation

Evaluation

- Experimental setup:
  - AMD EPYC 7713P 64 cores
  - 1 TB DDR4 memory
  - Host & guest kernel: 6.8 & 6.5 with VMPL enhancements

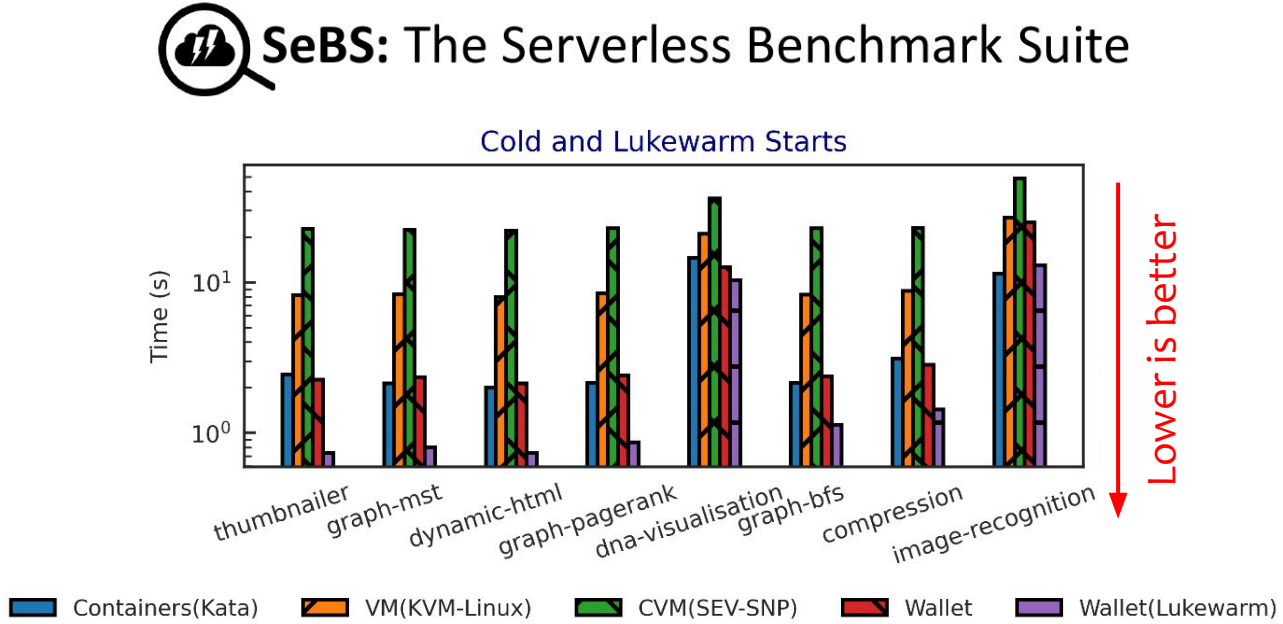
- Benchmarking variants:
  - **Container:** Kata Container runtime using Qemu
  - **VM:** Function in a standard VM with Linux kernel
  - **CVM:** Function in a AMD SEV-SNP VM with Linux kernel
  - **Wallet:** Execution of functions in Wallet framework (single CVM)

Evaluation

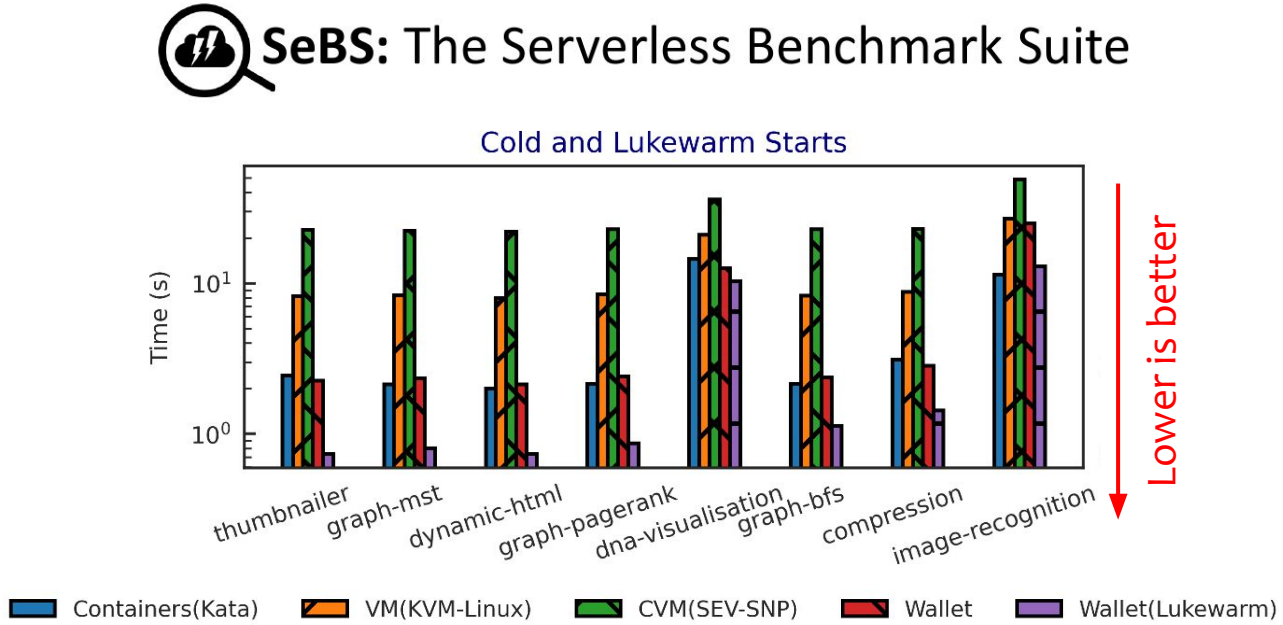
- Questions
  - What is Wallet's **end-to-end performance** in real-world serverless functions?
  - How does Wallet's communication mechanism perform with **function chaining**?
  - How does the Wallet's **Copy-on-Write** affect memory usage?

- For more results please refer to the paper

Performance



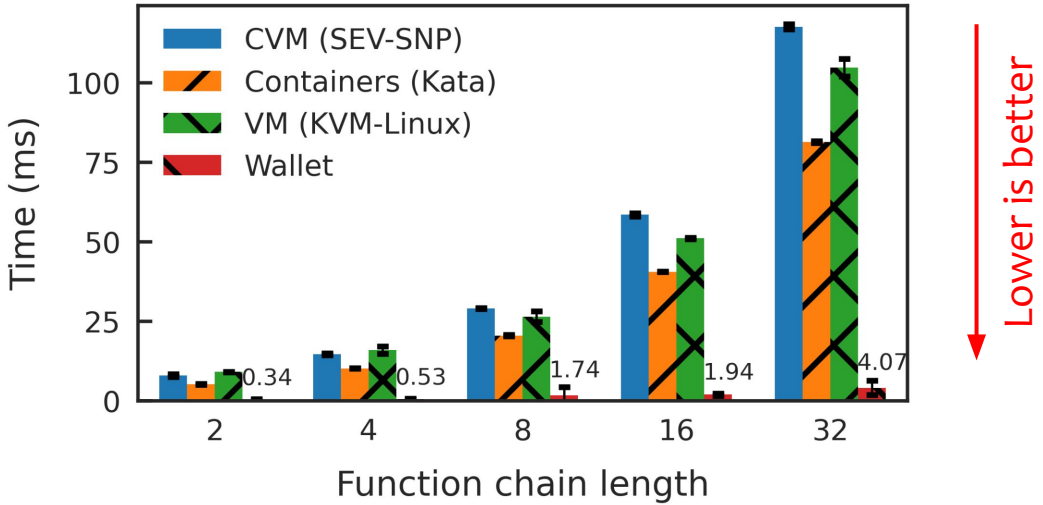
Performance



Wallet achieves lower end-to-end latencies compared to CVM deployments **(85%/93%)**

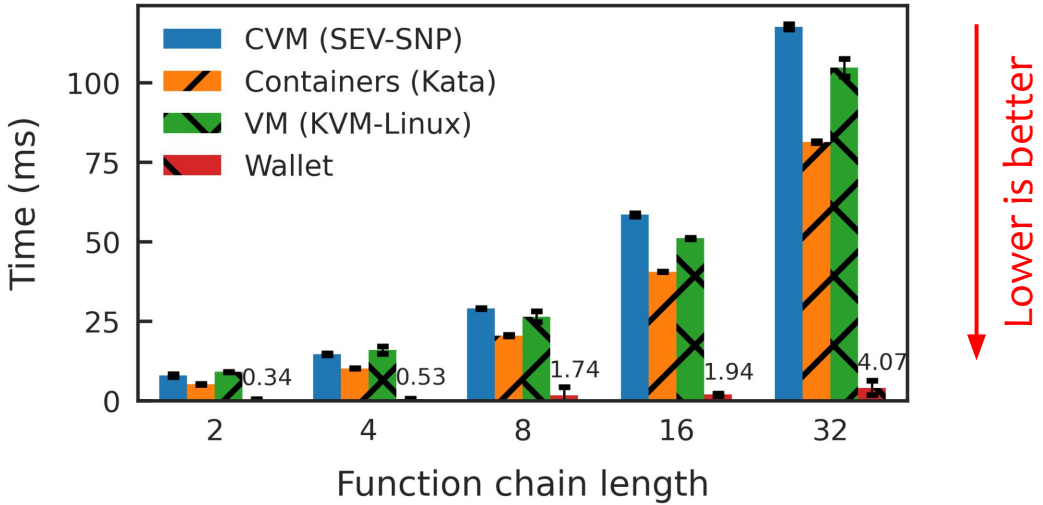
Communication channel performance

Latency of functions with fixed message size (16 kB) and increasing chain length



Communication channel performance

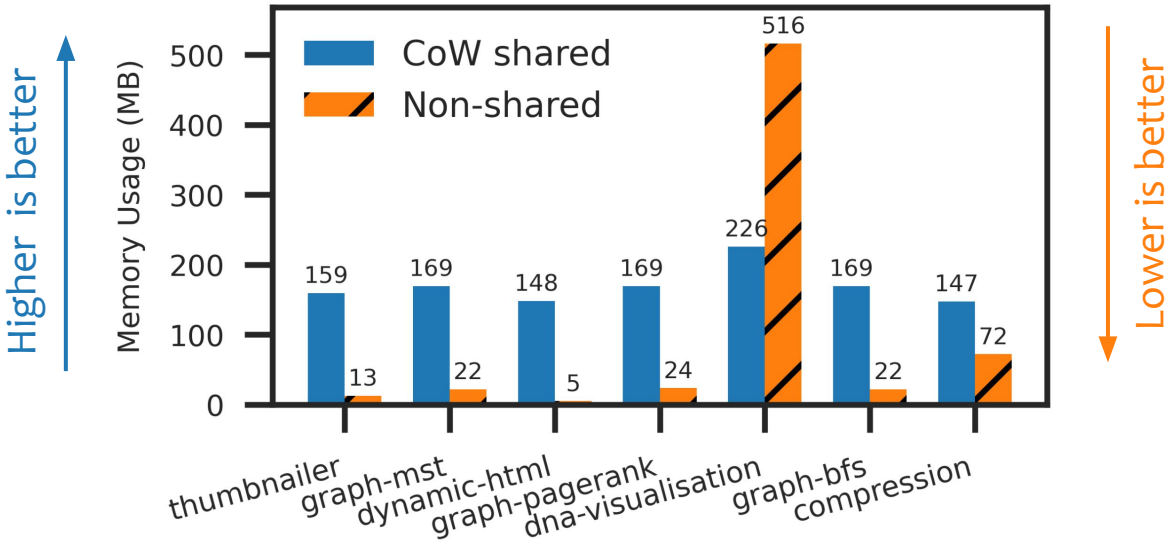
Latency of functions with fixed message size (16 kB) and increasing chain length



Wallet achieves low communication latency even with long chains (up to **30x**)

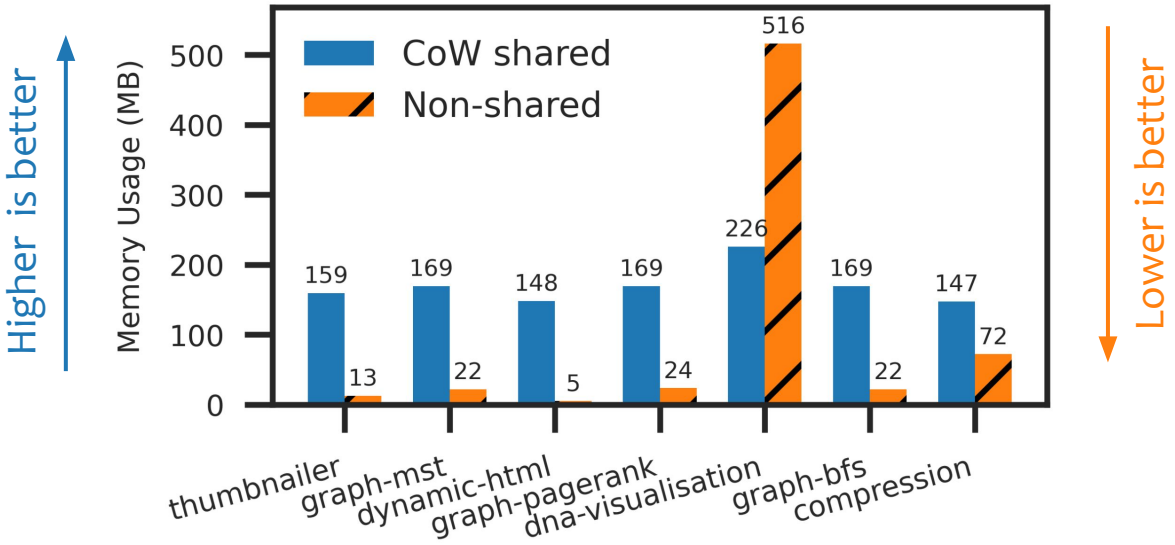
Copy-on-Write performance

Split between memory pages after trustlet execution



Copy-on-Write performance

Split between memory pages after trustlet execution



Wallet lowers memory requirements through its CoW mechanism (**78%** shared)

Summary

How do we enable **trustworthy serverless computing** in the cloud?

Wallet: Confidential Serverless Computing

- Small TCB and fast boot: Minimal trusted monitor, libOS-based functions
- Efficient function chaining: Direct shared-memory channels within a CVM
- Efficient resource utilization: Templated functions with CoW, privilege-level-based isolation

https://github.com/TUM-DSE/Wallet-VMPL