CONFIDENTIAL COMPUTING SUMMIT 2025

hosted by OPAQUE

TeeMate: Fast and Efficient Confidential Container using Shared Enclave

Jaewon Hur
Georgia Institute of Technology

Confidential Container: Inheriting Both Benefits of Container and Confidential Computing

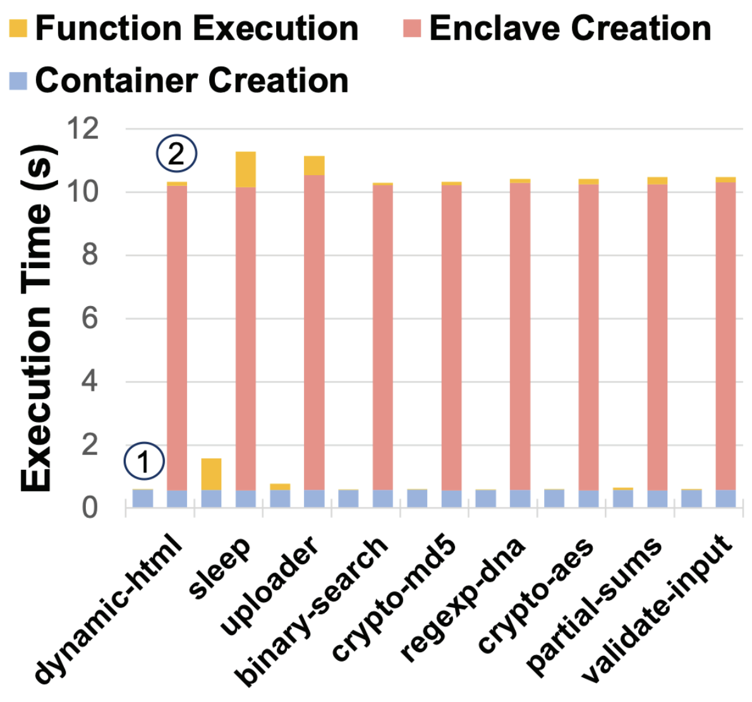
- Benefits of **Containerization**
  - Cloud providers manage system resources (e.g., cgroup) while users focus on their workloads

- Benefits of **Confidential Computing**
  - User's workloads are protected on potentially compromised (or even malicious) cloud environment

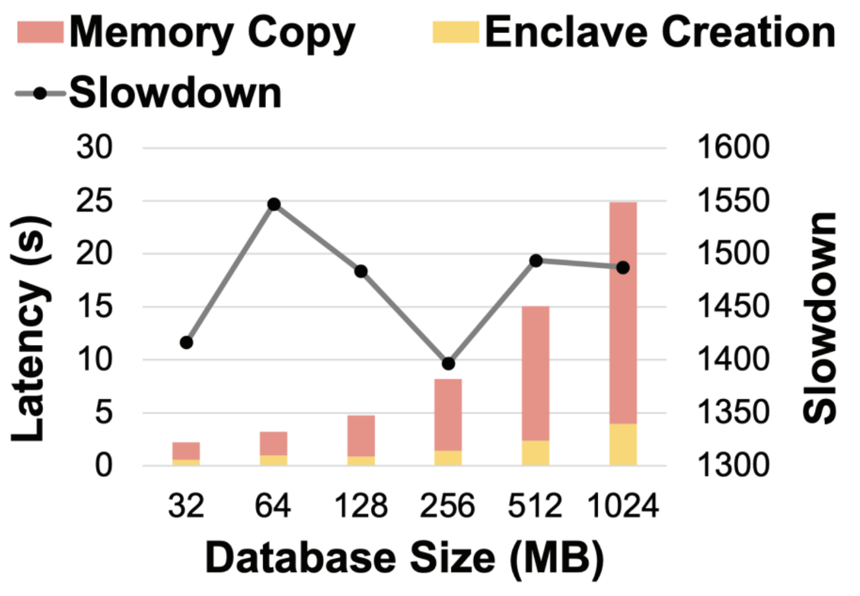
- Benefits of **Confidential Container**
  - Users can easily protect and deploy their workloads while cloud providers still manage the system resources

Motivation: Confidential Container Suffers from Large Performance Overheads

- **Performace overheads** of Confidential Container
  1. **Large bootstrap time** due to the enclave memory measurement
     ➢ Need to create every enclave (or cVM) for every confidential container creation
  2. **No fork-based bootstrap** due to the strict memory management
     ➢ When creating a confidential container through fork, entire parent's memory should be transferred through an encrypted channel



Latency of creating new function instance in confidential serverless computing



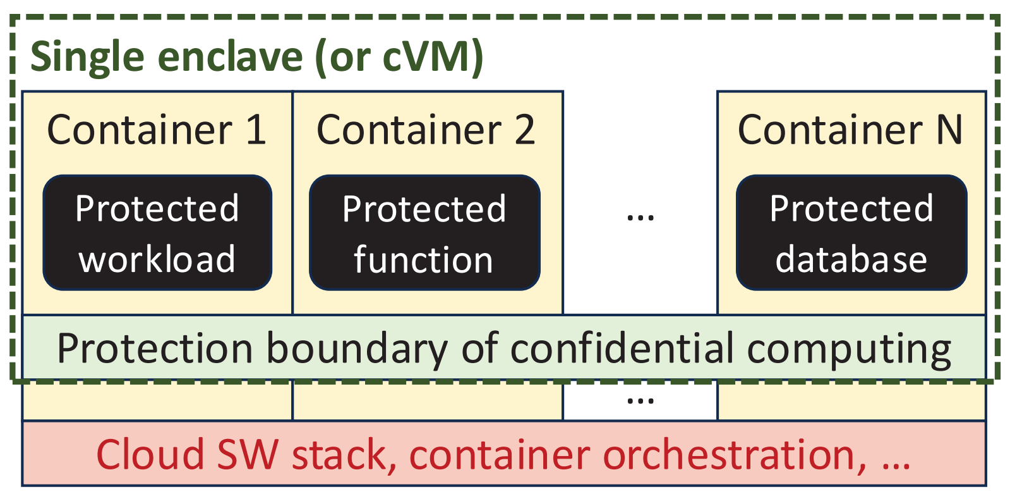
Latency of creating a child process for snapshot in confidental Redis database container

Breaking the Premise: Separating Control and Data Plane of Confidential Container

- **Root cause of the performance overheads**
  - *Strict one-to-one mapping of enclave (or cVM) and container*

- **Our insight**
  - *Actually, they can be separated!*
  - Separating **container (i.e., control plane)** and **enclave or cVM (i.e., data plane)**

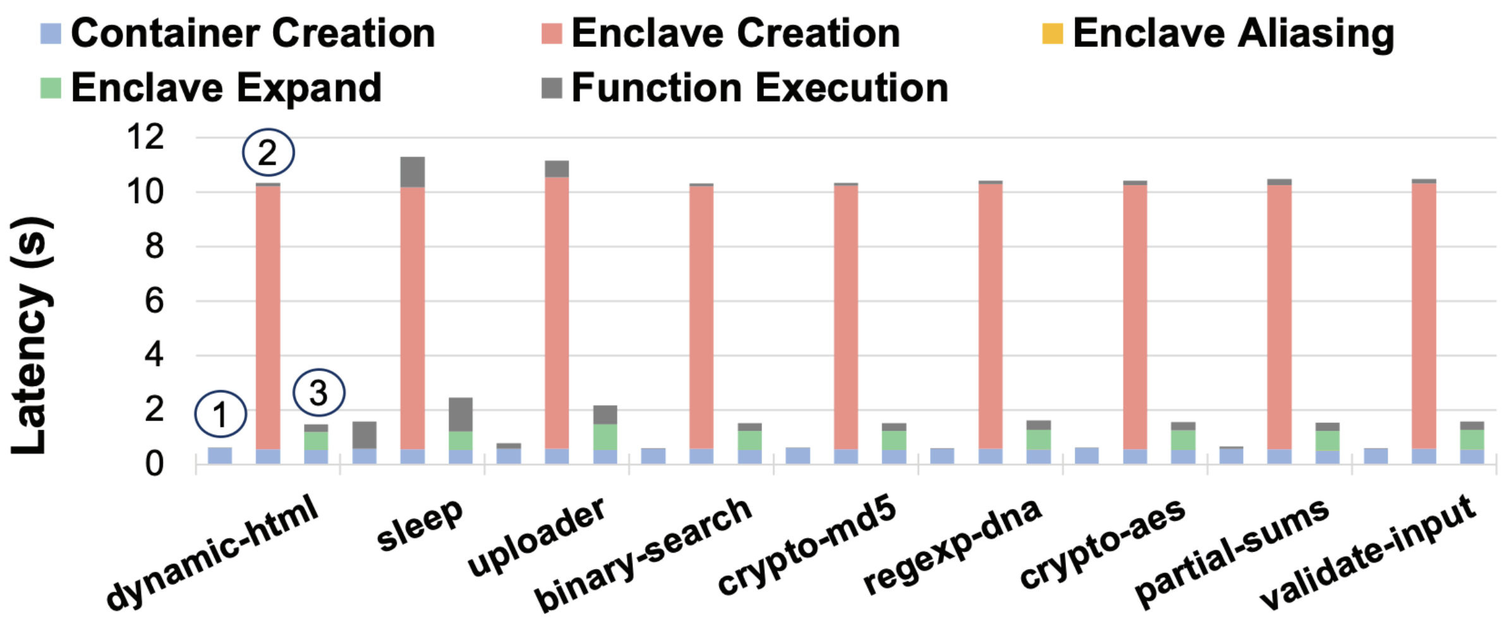
- **By Doing So**
  - We can host ***multiple containers with a single enclave (or cVM)***
  - Cloud providers still manage the resources based on the container
  - User's workloads still protected by confidential computing
  - ➢ **But, no bootstrap overheads & Fork based memory sharing**



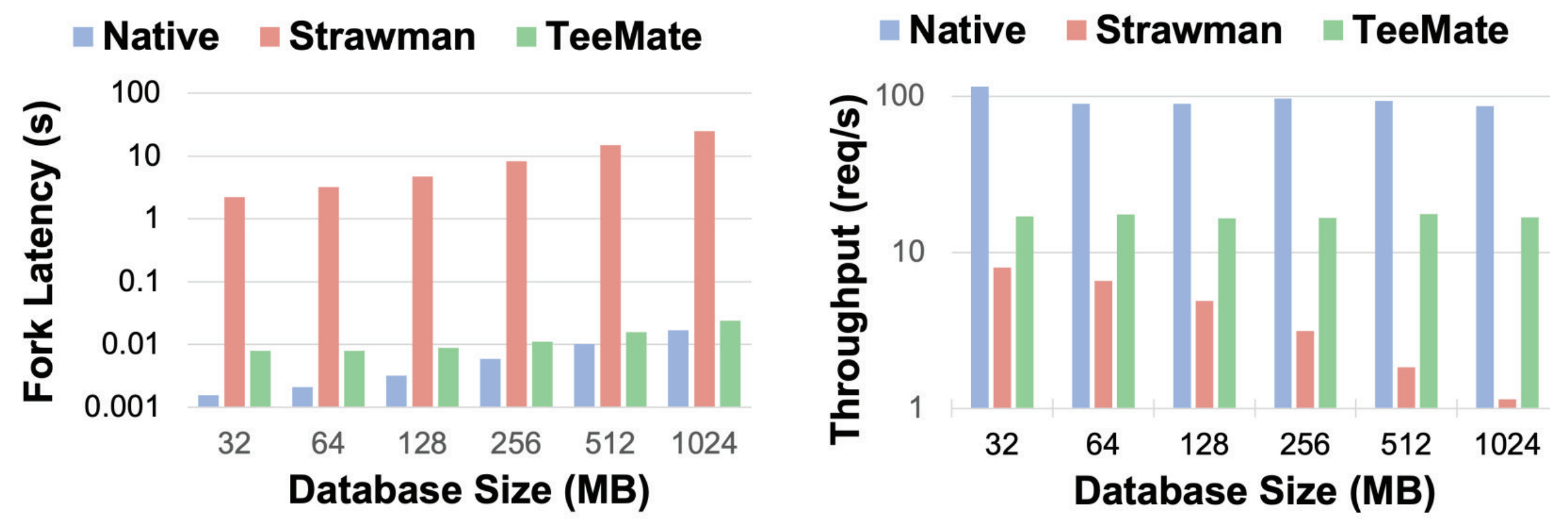
Key approach of TeeMate

TeeMate: Confidential Container with Minimal Performance Overhead

- **TeeMate** reduces bootstrap latency more than 5 times



Latency comparison of serverless applications on 1) native serverless framework (OpenWhisk), 2) strawman, and 3) TeeMate



Latency and throughput comparison of 1) native database (Redis), 2) strawman, and 3) TeeMate

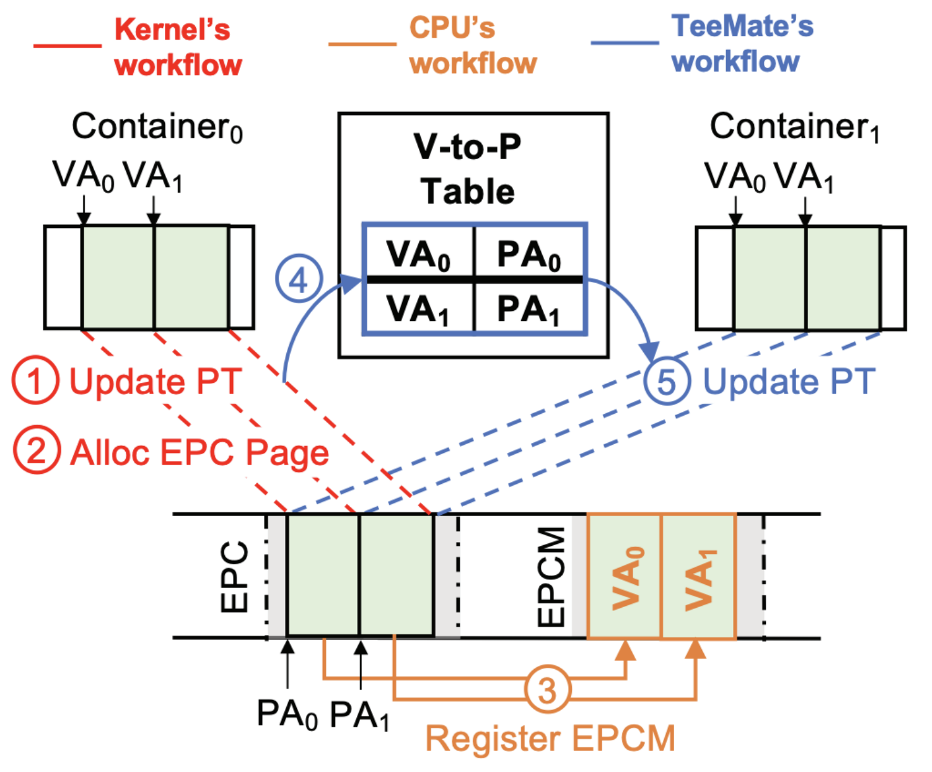
Design of TeeMate

- TeeMate designs **memory abstraction** and **thread abstraction** such that *different containers of the same enclave have their own view of memory address space and CPU thread each*

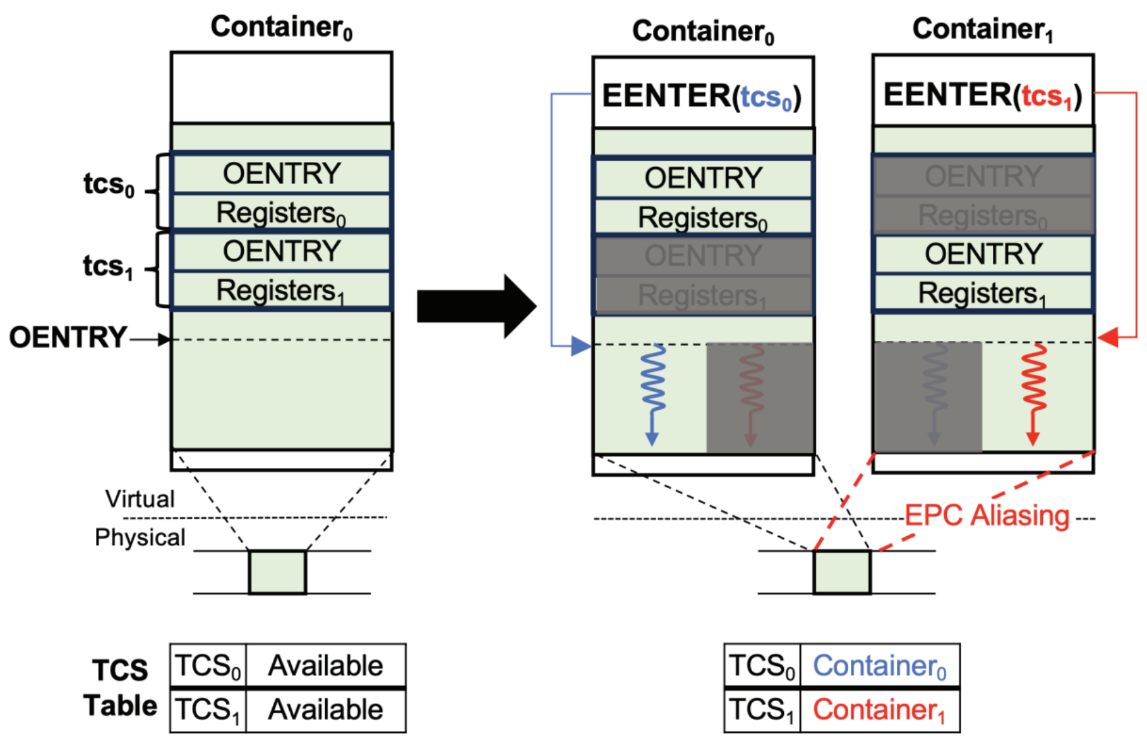
- **Memory abstraction**
  - Map the EPC pages of the same enclave to different address spaces

- **Thread abstraction**
  - Arbitrate threads of the same enclave to different containers

- **Namespace and cgroup**
  - Apply to each container as before



Mapping EPC pages of the same enclave to different address spaces



Assigning threads in an enclave to different containers

Conclusion

- We propose TeeMate, which solves the performance issues of confidential container with groundbreaking ideas.

- TeeMate breaks the premise that an enclave (or cVM) should be dedicated to only a single container.

- TeeMate shows that the container abstraction still works while they are served by a single enclave.

Reference
[1] TeeMate, https://arxiv.org/abs/2411.11423
[2] Confidential container, https://confidentialcontainers.org

Contact
Jaewon Hur, jwhur19@gmail.com
CV: https://jaewonhur.github.io/files/cv.pdf