MICRO 2022

Cronus: Fault-isolated, Secure and High-performance Heterogeneous Computing for Trusted Execution Environment

Jianyu Jiang, Ji Qi, Tianxiang Shen, Xusheng Chen, Shixiong Zhao,

Sen Wang, Gong Zhang, Xiapu Luo, Heming Cui

The University of Hong Kong

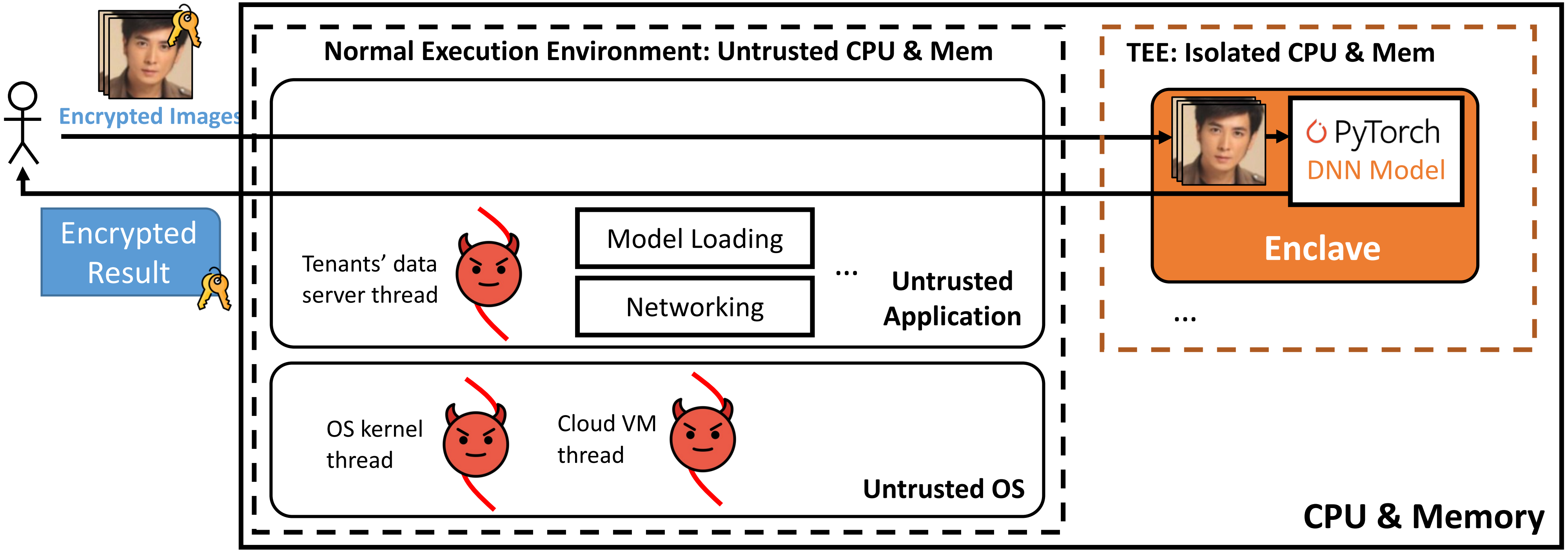
Huawei Technologies Co., Ltd.

The Hong Kong Polytechnic University

Trusted Execution Environment

➢ Trusted Execution Environment (TEE) is prominent for protecting sensitive user data in clouds and demands high computing capacity

- TEE (e.g., Arm TrustZone, Intel SGX, Keystone on RISC-V) provides an isolated execution environment (**enclave**) that cannot be seen or tampered with, and typically runs only data processing logic within the enclave
- Many AI application (DNN training) demands high computing capacity (e.g., 300TFLOPS) for high performance



Motivating example: Using TEE for protecting face recognition

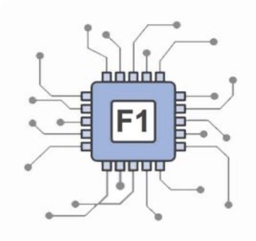
Integrating Accelerators into TEE is Highly Desirable

- More and more user sensitive data are being processed in accelerators for high performance
- Integrating accelerators into TEE is highly desirable for TEE to gain high computing capacity

NVIDIA DGX-2
with 16 A100

2 PetaFlops

DNN Training
DNN Inference



AWS F1 Instance
with 8 FPGA cards

100X Faster than CPU

Real-time Video Processing
Financial Analytics

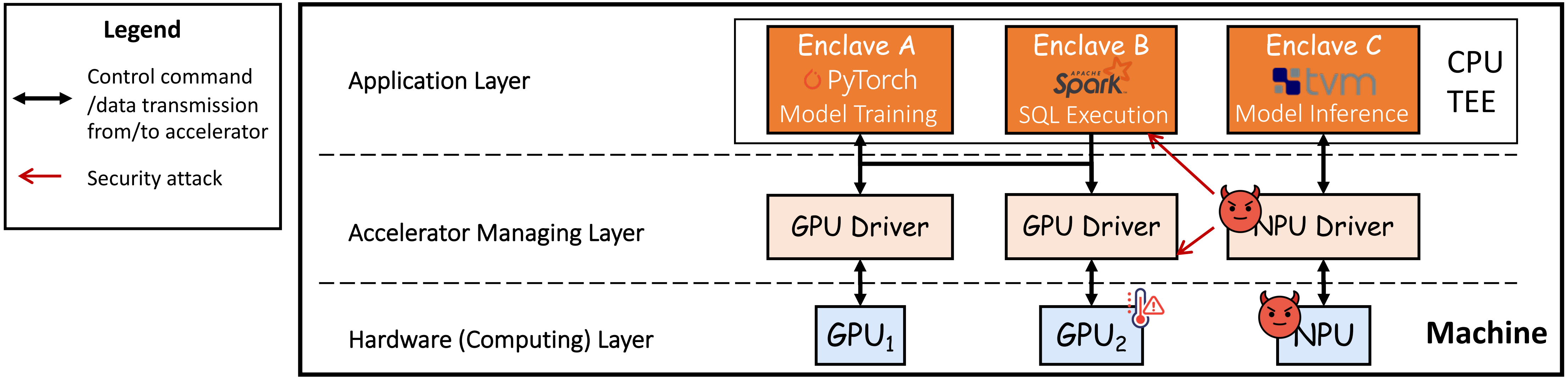
AMD server
with 8 Xilinx Alveo Cards

260 INT8 TOPs

DNN Inference
Video transcoding
Database search & analytics

Requirements for Integrating Accelerators into TEE

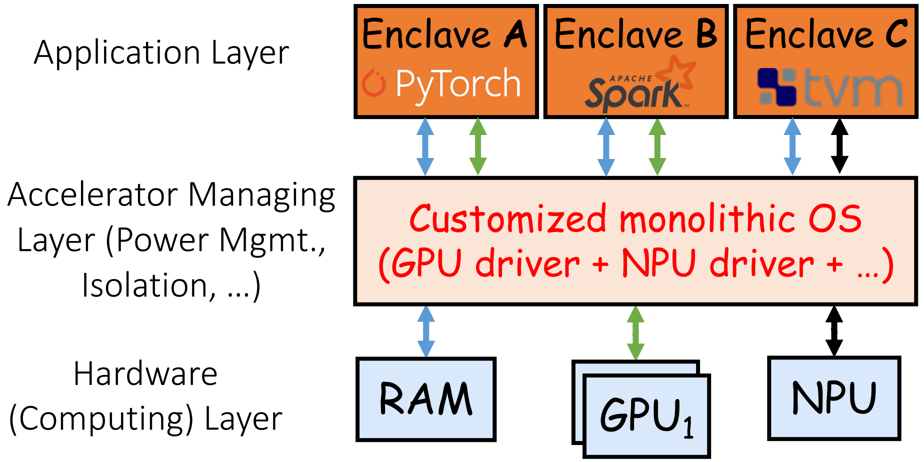
- **Unlike traditional TEE systems with only CPU, our paper takes the first step to identify three special requirements for TEE systems with accelerators**
  - ✓ **Requirement 1: Fault isolation -** an accelerator's failures will not affect other accelerators in the same machine
    - Servers w/ accelerators are 7X more likely to fail due to hardware/driver faults compared with servers w/o accelerators [SC'19]
  - ✓ **Requirement 2: Security isolation -** an accelerator's hardware or its managing software (driver) cannot attack other applications running on other accelerators in the same machine
    - Accelerators/SoC components can contain buggy or malicious code for the adversary to launch attacks [HotOS'21]
  - ✓ **Requirement 3:** A **general** accelerator can be **spatially shared** among enclaves (tenants)
    - Cloud services using GPU without spatial sharing achieve only 10% resource utilization on average [OSDI'20]



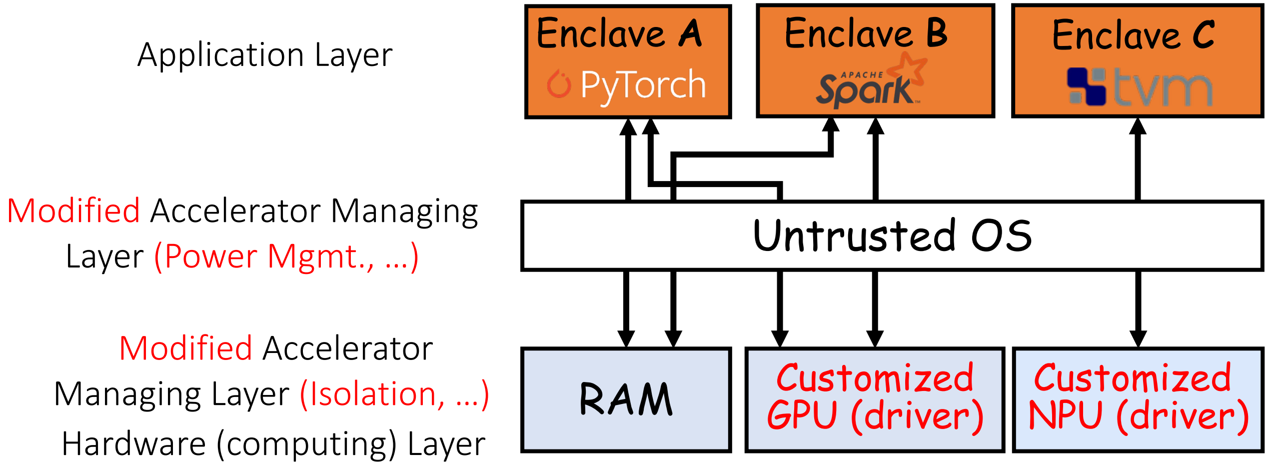
Existing Monolithic Design cannot Meet all Requirements

- Monolithic OS Kernel Approach (SeCloak [MobiSys'18], PROTECTION [NDSS'20], StrongBox [CCS'22]): Integrating all accelerators' drivers into a monolithic trusted (TEE) OS (e.g., Linux)
  - ✓ **Support spatial sharing of general accelerators (R3)**
  - ✗ No fault isolation (violating R1)
  - ✗ No security isolation (violating R2)
- Monolithic Hardware Approach (Gravition [OSDI'18], SGX-FPGA [DAC'21], GuardNN [DAC'22]): Integrating the accelerator's managing logic (driver) into the accelerator
  - ✓ **Enforce fault isolation (R1) and security isolation (R2)**
  - ✗ Cannot support spatial sharing on general (non-modified) accelerators (violating R3)

(1) Monolithic OS kernel approach

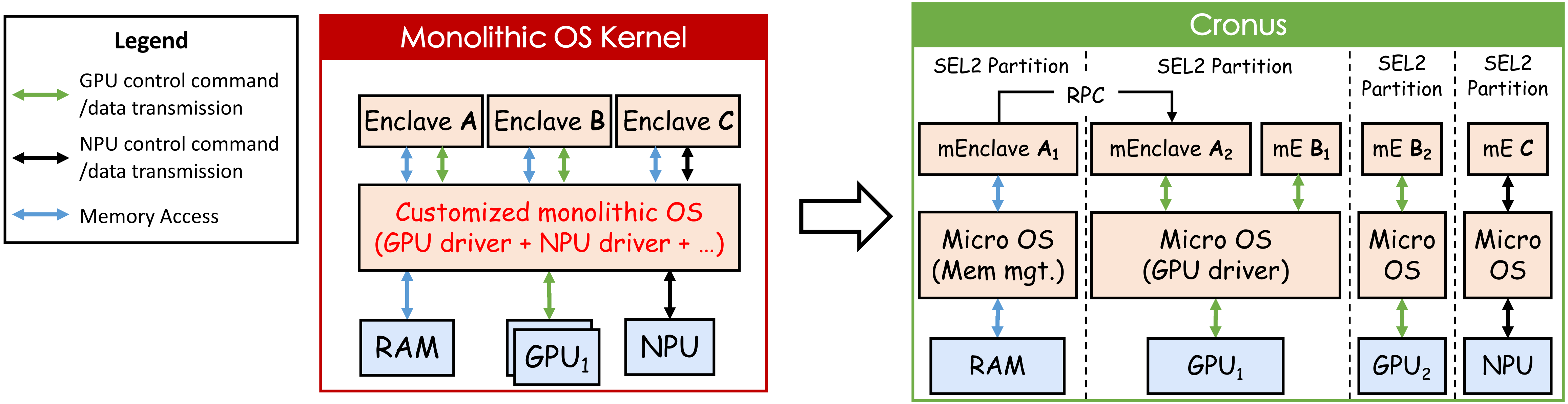


(2) Monolithic hardware approach



Cronus – a Microkernel-inspired OS for Acc. TEE

- A monolithic TEE OS kernel is partitioned into isolated Micro OS (mOS), where each mOS manages only one accelerator; a monolithic enclave is partitioned into mEnclaves running different types of computation
  - ✓ **Security Isolation (R1):** An mOS trusts only its software stack and the managing accelerator
  - ✓ **Fault Isolation (R2):** A failure from mOS or its managing accelerator will not cause failures of other mOSes and accelerators
- Leveraging existing hardware primitives for isolation and spatial sharing **(R3)**
  - Leveraging SecureIO (originally used for protecting sensors) to ensure secure connection between TEEs and accelerators
  - Leveraging hardware isolation techniques (e.g., ARM TrustZone SEL2) to isolate accelerators' software and hardware stack
  - Leveraging accelerators' isolation primitives (e.g., GPU context) to spatially share an accelerator among enclaves with isolation



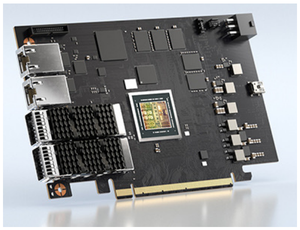
Cronus is a One-fit-all TEE for General Accelerators

- ➢ Cronus is a **one-off developed**, **one-fit-all** TEE for general accelerators to support trusted execution
  - ➢ As long as the machine running Cronus has SecureIO supports (e.g., ARM TrustZone and RISC-V)
- ➢ Cronus does not require time/money-consuming hardware customization for an accelerator
  - ➢ Customizing NVIDIA GPU (H100) with confidentiality supports takes more than two years
  - ➢ Customizing accelerators with trusted execution requires millions of dollars for design and validation [HotOS'18]
- ➢ Accelerator vendors can focus on only improving accelerators' computing performance (throughput/latency)

Nvidia A100

Xilinx U250

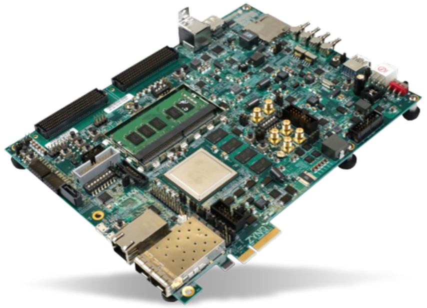
Intel PAC D5005



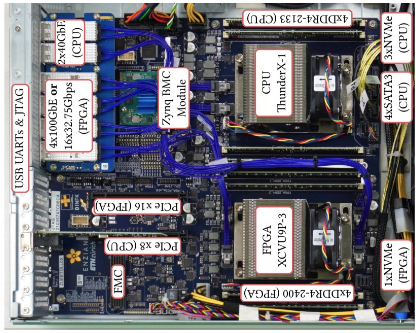
Nvidia BlueField-2 DPU

Cronus Supports Accelerators on SoCs

- A System on a Chip (SoC) integrates more and more accelerator components
  - An SoC can contain more than five FPGA components for accelerating computation or improving I/O performance (Enzian [ASPLOS'22], Coyote [OSDI'20])
  - Different SoC components may be developed by different hardware vendors
- A tenant may leverage its used accelerator of an SoC to attack other tenants using other accelerators on the same SoC
  - Different accelerator components on an SoC can be assigned to different tenants for maximizing utilization
  - A network acceleration component's hardware/software can contain buggy/malicious code suspected of attacks [HotOS'21]
- Cronus is a one-off developed TEE system for general SoCs
- Cronus's security isolation can tackle this attack [future work]
  - As long as the machine running Cronus has SecureIO supporting the accelerators



Xilinx Zynq Ultrascale+



Enzian with many FPGA chips

Cronus Improves Accelerator Utilization with TEE

- Spatial sharing on accelerators significantly improves resource utilization and saves money
  - Public clouds without spatial sharing incur only 10% resource utilization in production (Antman [OSDI'20])
  - Employing spatial sharing of GPU in PaaS services (e.g., DNN training) results in high (e.g., 70%) resource utilization
- PaaS services employ spatial sharing in accelerators but do not have TEE protection
  - AWS's EMR processes sensitive database data using NVIDIA's RAPIDS Accelerator
  - Azure's Cognitive Services processes user sensitive face data using GPU
- Cronus is suitable for securing PaaS services with high resource utilization (low costs)
  - Cronus enables spatial sharing and trusted execution simultaneously on accelerators

aws EMR

Microsoft Cognitive Services

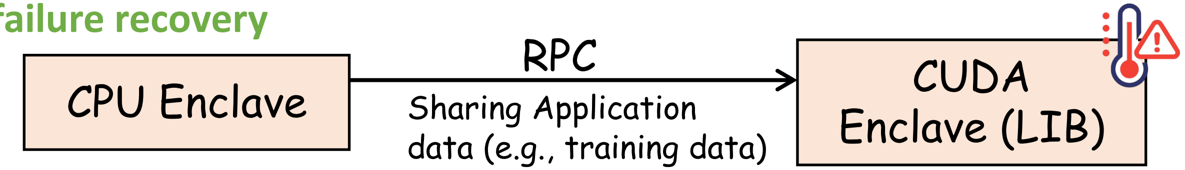
Cronus has Broad Applications

- AI computing and microservices are security-critical and performance-critical
  - AI computing (e.g., DNN training) and microservices (e.g., firewall) process sensitive user data (e.g., face data, social network), and are developed by different parties (e.g., PyTorch by Meta, TensorRT by Google)
  - Microservices demand fast bootup and failover (~seconds)
  - DNN inference on GPT3-2.7B requires 7 TFLOPS for each layer for low inference latency (<1s)
- Cronus's mEnclave/mOS abstraction are lightweight for bootup and failover (<1s)
- Cronus is suitable for running AI computing and microservice [future work]
  - Cronus's fast bootup and failover are favorable for AI computing and microservices
  - Cronus's security isolation can isolate different tenants' AI computing services and microservices within the service suit

PyTorch tvm AWS Lambda Microsoft Azure Automation

Implementation and Evaluation Details

- **Technical Challenges:** Efficient and crash-safe RPCs between mEnclaves managed by different mOSes
  - **Solution: A Streaming RPC (sRPC) protocol that ensures fast communication through secure shared memory, and guarantees safety during failure recovery**



- **Other protocols in Cronus and implementation details:**
  - Remote and local attestation protocols for attesting the integrity of accelerators
  - Automatic partition of a monolithic enclave into multiple mEnclaves
  - Built Cronus on ARM TrustZone and implemented both DNN training and inference (i.e., PyTorch and TVM) on Cronus

arm TRUSTZONE PyTorch tvm

- **Baseline frameworks:**
  - TrustZone (Optee) and HIX-TrustZone [ASPLOS'19]
- **Evaluation setup:**
  - Hardware: 4 NVIDIA 2080Ti GPU and a simulated TVM VTA device
  - Benchmarks: two microbenchmarks (Rodinia and VTA Bench), and two real-world applications (DNN training and inference) on 5 DNN models (LeNet, ResNet, VGG, DenseNet, and Yolo) using the MNIST and ImageNet dataset

4 X  1 X 

VTA FSIM

Evaluation Questions

- What is the **end-to-end performance** of Cronus in microbenchmarks and real-world applications?
- What is the performance gain of **spatial sharing**?
- How fast can Cronus **recover from faults**?
- Can Cronus support multiple accelerators?

End-to-end Performance

- We compared the latency of Rodinia benchmark (Fig. 1) and real-world DNN training (Fig. 2); we also compared the throughput of VTA-bench benchmark (Fig. 3a) and latency of DNN inference (Fig. 3b)

Figure 1

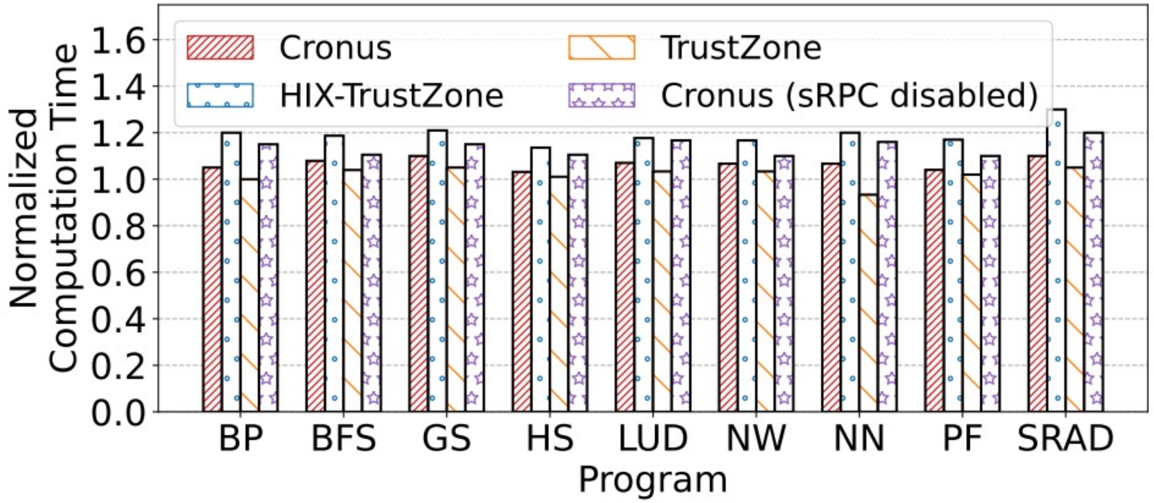


Figure 2

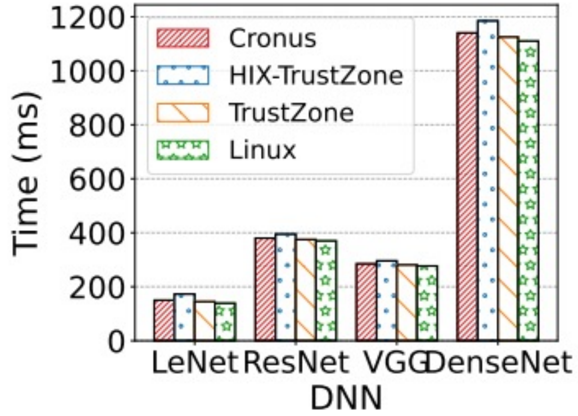
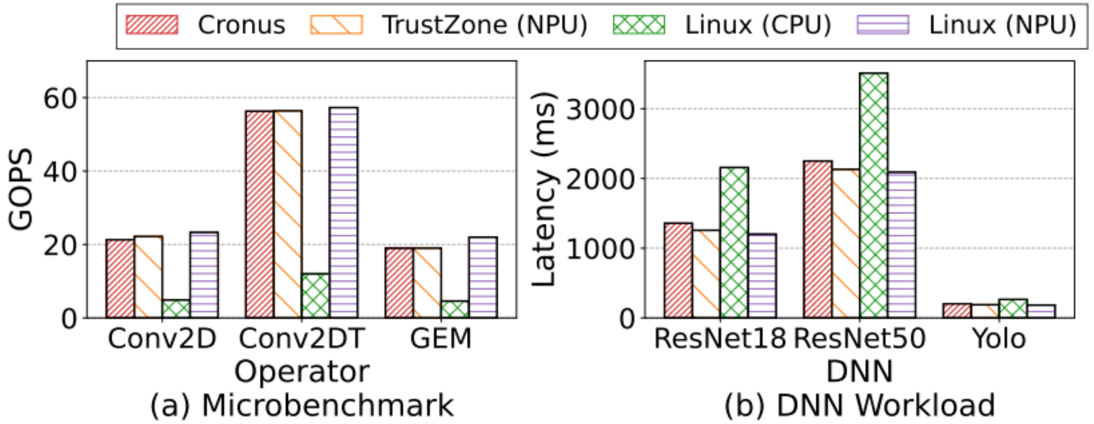


Figure 3



- ✓ Compared with **native (unprotected) computation** (i.e., Linux in the figure), Cronus incurs **less than 7% performance overhead** for both microbenchmarks and real-world DNN training/inference; Cronus's performance is close to the monolithic OS kernel approach and is faster than HIX-TrustZone

Spatial Sharing and Failure Recovery

- We ran concurrent enclaves sharing accelerator temporally and spatially in Cronus (Fig. 4a)
- We ran two tasks (A and B) at two different mOSes and made Task A fail deliberately (Fig. 4b)

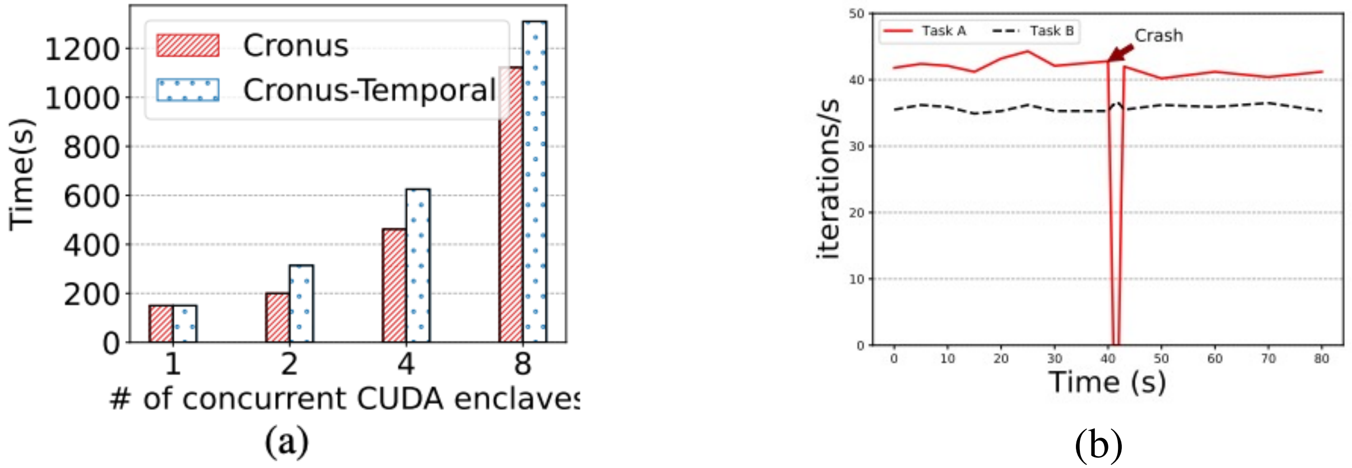


Figure 4

- **Figure 4a:** Compared with Cronus without spatial sharing, **Cronus with spatial sharing decreased up to 43% of the total computation time**.
- **Figure 4b**: Cronus achieves **fault isolation** and **fast failure recovery**: recovering from faults with hundreds of milliseconds.

Conclusion

- In this paper, we design Cronus with a new MicroTEE architecture that enables fault-isolated, high-performance and secure heterogeneous computing
  - The first TEE system that requires only one-off development for enabling trusted execution within general accelerators with security isolation, fault isolation and spatial sharing on accelerators among tenants
- Cronus's future work is broad:
  - Cronus can support distributed workloads on different machines
  - Cronus can be integrated with other TEE hardware (e.g., Keystone [Eurosys'20])
  - Cronus can support diverse accelerators on SoCs and new applications (e.g., microservices)
- Cronus's artifact is available on https://github.com/hku-systems/cronus