Regaining Lost Cycles with **HotCalls**: A Fast Interface for SGX Secure Enclaves

Ofir Weisse
Valeria Bertacco
Todd Austin

**SGX Secure Execution**
- Secure enclave is isolated from the system
- Memory encrypted on CPU die (via MEE)
- OS Kernel
- VMM
- SMM
- RAM
- HW
- CPU
- Memory encrypted on EPC metadata
- Encrypted memory is a potential bottleneck

**Cloud Attack Surface**
- To lower costs - computation and storage are moved to third party machines
- This implies trust across the entire software stack
- And cloud provider employees
- The attack surface is large

**SGX in a Nutshell**
- Memory encrypted on CPU die (via MEE)
- User Space
- Enclave
- Physical Memory
- Enclave Page Cache (EPC)
- SMM code (firmware)
- Hardware
- Secure enclave is isolated from the system

**SGX Life-Cycle**
- Application - Untrusted Code
- Enclave – Trusted Code
- Plaintext Shared Memory
- Encrypted Memory
- Can call system API functions (send, read, etc.)
- Can access all memory
- No access to system calls

**Cost of Ecalls/Ocalls**
- Ecalls
  - SDK code
  - EENTER
  - EEXIT
  - Cold cache
  - Warm cache

- Ocalls
  - SDK code
  - EEXIT
  - - ERERESUME
  - Cold cache
  - Warm cache

**Cost of Encrypted Memory**
- Write Latency
  - 102% overhead
  - Write encrypted memory
  - Read encrypted memory
- Read Latency
  - 20% Cache-miss
  - Read plaintext memory

**HotCalls Mechanism**
- Key insight: requesting services does not mandate a context switch
- Requester
- Responder
- No context switch
- Additional thread
- Spinlock
- call_ID
- void *data
- Go | Done

**HotCalls vs. SDK Calls**
- Cumulative Distribution of Call Latencies
- Ecalls
  - - EENTER
  - - EEXIT
  - - SDK Calls
  - - 1,300 cycles
  - - 14,100 cycles
  - - 8,600 cycles
  - - 2.16 μsec
  - - 3.52 μsec

- Ocalls
  - - EEXIT
  - - - ERERESUME
  - - - SDK Calls
  - - - 1,300 cycles
  - - - 8,200 cycles
  - - - 8,600 cycles
  - - - 2.06 μsec
  - - - 3.52 μsec

**HotCalls in Action**
- Normalized Throughput
- HotCalls are order of magnitude faster

Published at ISCA 2017

Making SGX great again