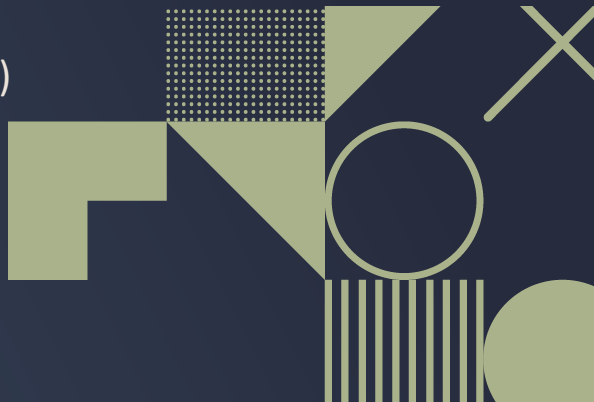
Low Overhead Security Isolation using Lightweight Kernels and TEEs

John R. Lange (ORNL/Pitt), Nicholas Gordon (Pitt), Brian Gaines (SNL)



Post Exascale HPC OS/R Challenges

- Security is becoming increasingly important on large scale HPC systems
  - Edge Integration will introduce co-located workloads from new users
  - Data centric AI/ML workloads will require access to sensitive/protected data
  - Federation of HPC resources will require cross organizational identities

- Existing HPC OS/Rs still rely on traditional security controls
  - Unix account identities
  - Unix file permissions

- **Increased security requirements will require more extensive OS/R security capabilities**

- This work:
  - First step towards leveraging trusted computing hardware features to enable secure compartmentalization of HPC OS/Rs
    - Combine Lightweight Kernels and Trusted Hypervisors

Trusted Computing Capabilities

- Hardware security features are becoming prevalent
  - Intel SGX, ARM TrustZone, AMD SEV
  - Not a HPC viable solution yet, but we're heading in the right direction

- Necessary Features:
  - Isolated Execution
  - Sealed Storage
  - Attestation

TEE Features

- **Isolated Execution** provides isolated HW resources on an untrusted platform
  - Hardware protected confidentiality and integrity for code and data
  - External software cannot access enclave memory
  - Enclaves are permitted to access external memory

- **Sealed storage** allows for the long-term secure storage of protected information

- **Local and remote attestation** allows verification of the authenticity of an enclave
  - Local attestation has limited utility for distributed systems

- **Enclaves are protected from co-located applications and malicious OS/Hypervisors**

Current TEE approaches

- Intel SGX
  - Isolated Execution, Sealed Storage, Local + Remote attestation
  - Enclaves have limited functionality (i.e. no system calls)

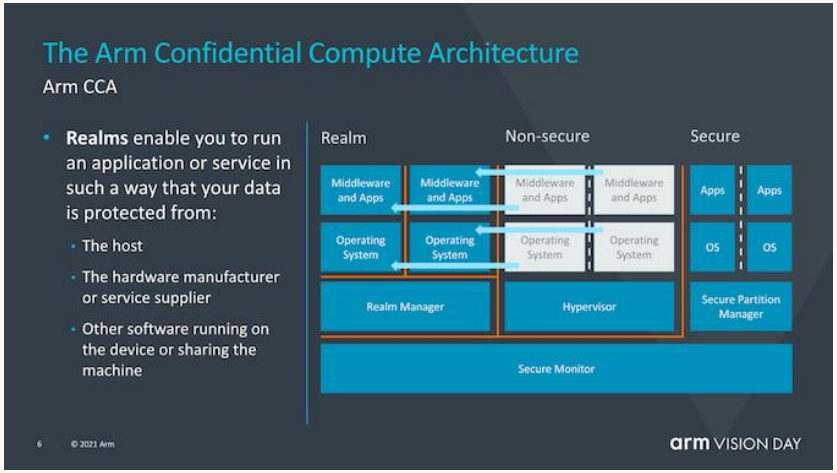
- ARM TrustZone
  - Isolated Execution, Sealed Storage, only Local Attestation on some platforms
  - Can Isolate full OS/Hypervisor stacks
  - Designed for commodity/handset devices

- AMD SEV
  - Isolated Execution for Virtual Machines

TEEs for HPC

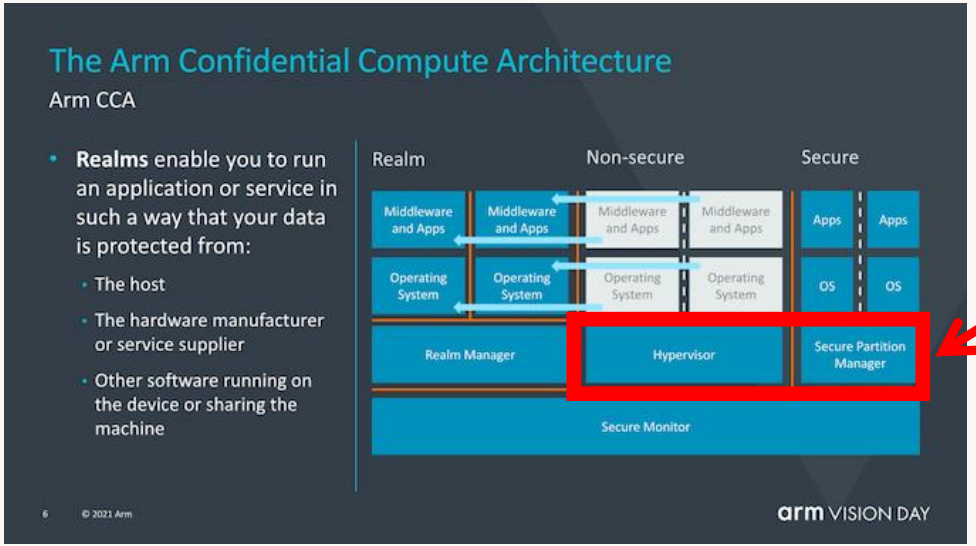
- Ideal solution is probably a combination of SGX and TrustZone
  - Memory isolation and encryption
  - Scalably attestable execution environments
  - Dynamic instantiation of TEE instances
  - Secure I/O capabilities
  - Dynamic resource assignment

- We're heading in the right direction…



Hardware Trends

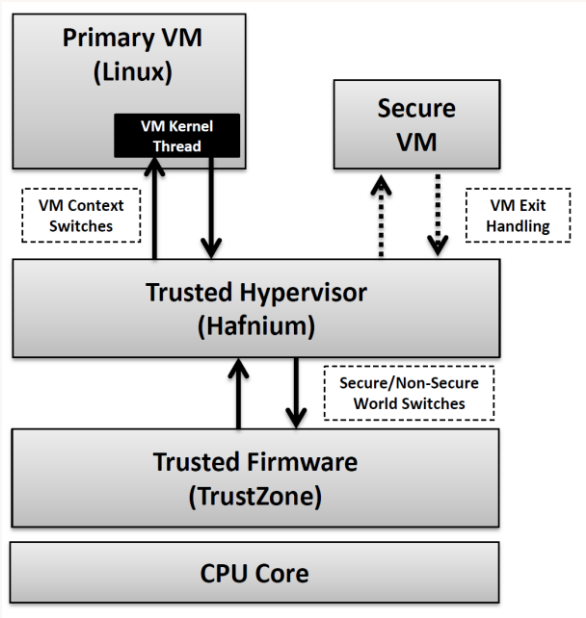
- We’re heading in the right direction…



This work

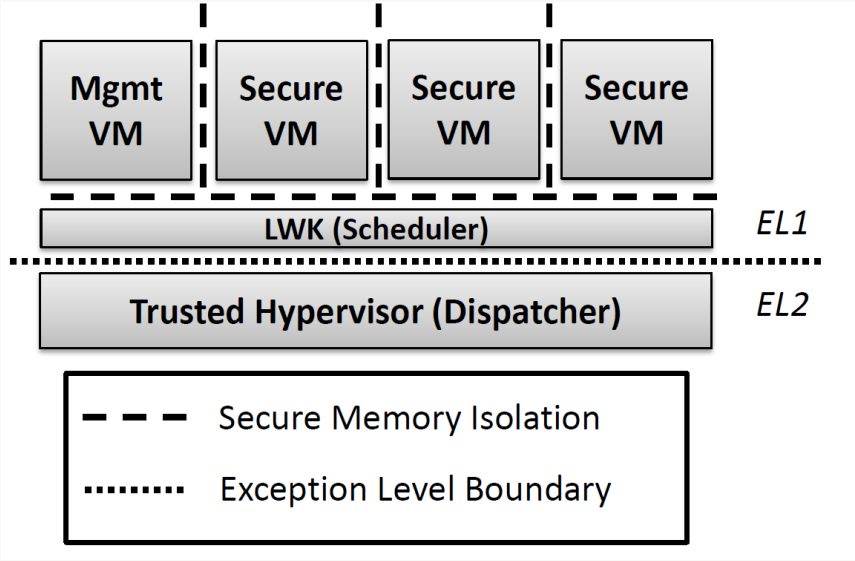
Hafnium Trusted Hypervisor

- Hafnium:
  - **“A reference Secure Partition Manager (SPM) for systems that implement the Armv8.4-A Secure-EL2 extension”**
    - www.trustedfirmware.org
- Type 1 hypervisor running at EL2
  - Statically partitions memory at boot time between pre-configured VMs
  - Acts as a secure dispatcher for VM contexts
    - Relies on Primary VM (Linux) to provide CPU scheduling
- Can leverage TrustZone partitioning



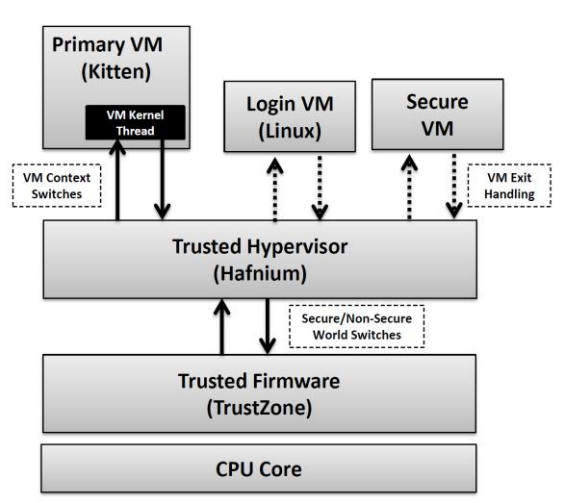
Trusted Hypervisors for HPC

- **Problem:** Every vCPU managed by Linux scheduler
  - Every vCPU is implemented as a kernel thread
  - The primary VM runs on every core
- Our approach:
  - Use an LWK (Kitten) for scheduling
  - Retain Linux for Management
- Kitten runs on every CPU core
  - Linux constrained to a subset of cores
- Pros:
  - Reduced Timer tick rate
  - Overheads from Linux background tasks constrained



Kitten as the Primary VM

- Ported Kitten to ARM64
  - Started at SNL, finished at Pitt
  - Supports Qemu, Raspery Pi, Pine A64
  - Upstreamed to Kitten
    - https://github.com/HobbesOSR/kitten/
- Implemented Hafnium hypercall interface
  - Basic CPU context switching API
  - Hardware timer delivery



Kitten as a Secure VM

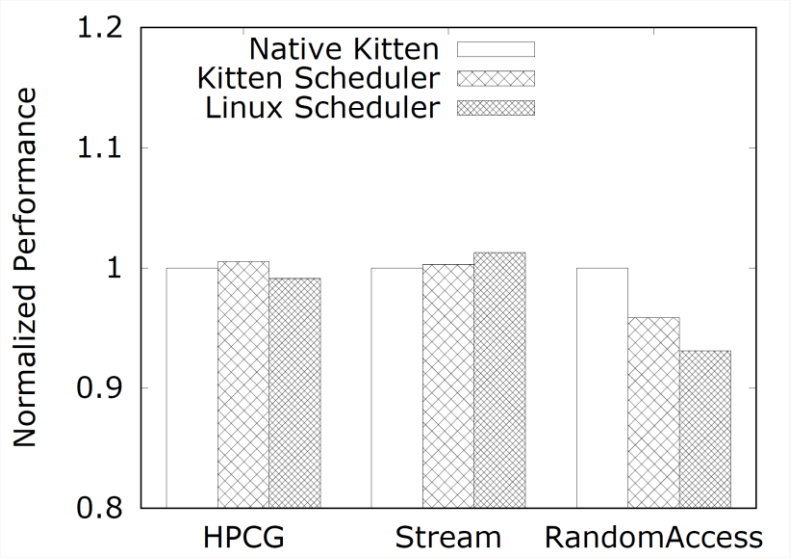
- ARM generally assumes TEEs have very limited functionality
  - Secure Secret Storage, Secure IO for identity verification, etc.
- Hafnium doesn’t provide full hardware virtualization support
  - Disables everything possible to minimize attack surface
  - Some of these things are necessary to run full OS
    - E.g. cycle counters
- Adding Linux support is ongoing, but initially we focused on Kitten
- Running Kitten as a secure VM required modifying both Kitten and Hafnium
  - Hafnium modified to be more permissive
    - Should still be secure, but a full audit is needed
  - Kitten modified to support Hafnium’s para-virtual VM environment
    - Para-virtual interrupt controller and timer

Evaluation

- This is a preliminary prototype with very rough edges, so…
  - Lots of Caveats
- All evaluation was performed on a single Pine A64 LTS SBC
  - 4 Core Allwinner A64 (1.152 GHz)
  - 2GB RAM
  - https://www.pine64.org/devices/single-board-computers/pine-a64-lts/
- Benchmark runs were small due to constrained memory
- Limited number of benchmarks were able to run due to compatibility issues and/or bugs
- No competing workloads and no Linux management VM

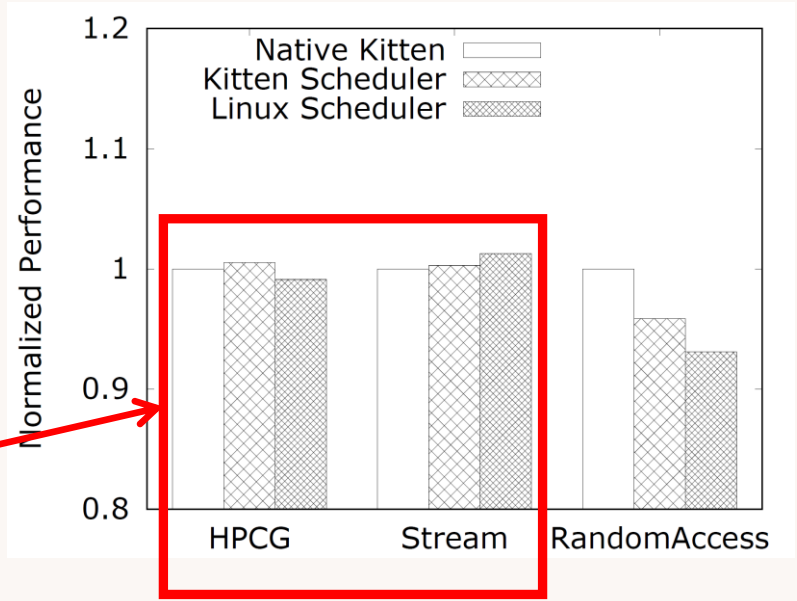
Memory benchmarks + HPCG

- HPCG, Stream and RandomAccess
  - Results reported as normalized



Memory benchmarks + HPCG

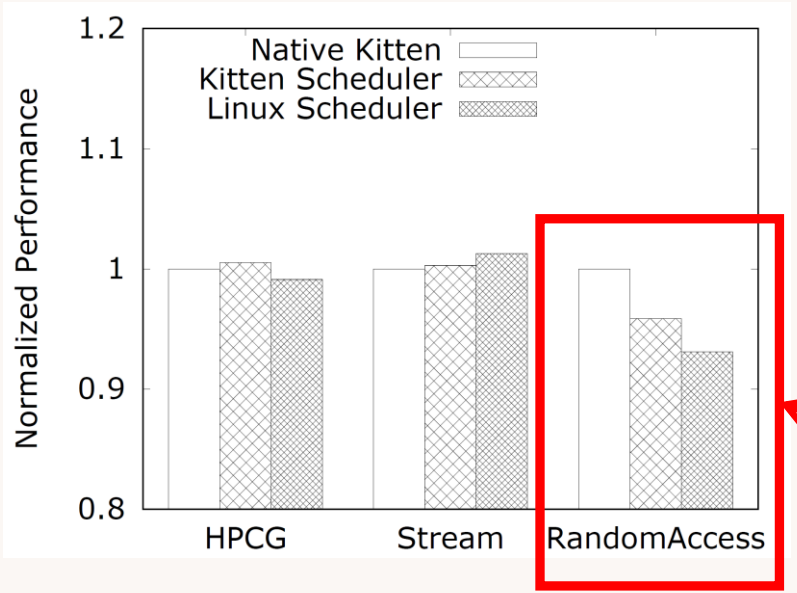
- HPCG, Stream and RandomAccess
  - Results reported as normalized



No significant difference in Stream and HPCG

Memory benchmarks + HPCG

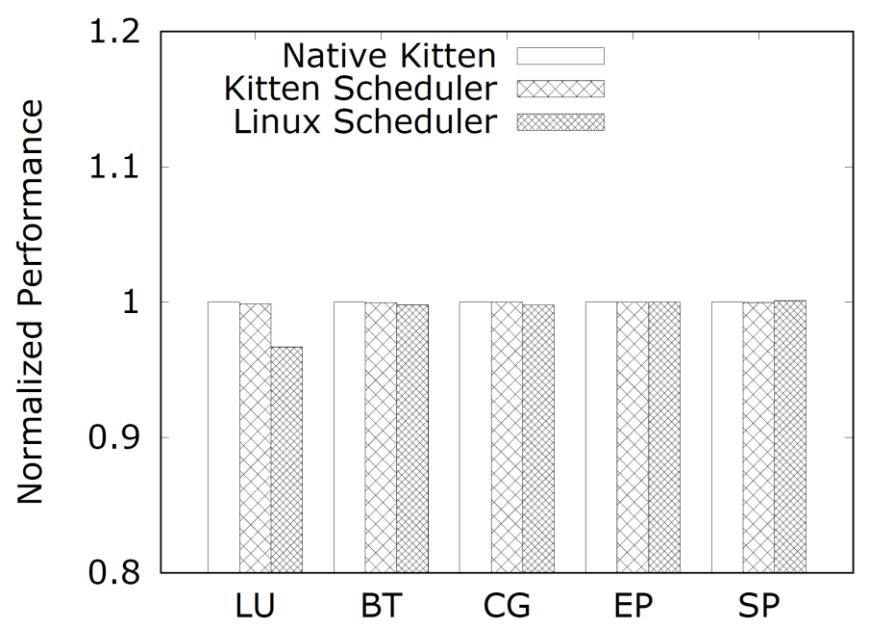
- HPCG, Stream and RandomAccess
  - Results reported as normalized



Virtualization added noticeable overhead and the Linux scheduler impacted performance by ~5%

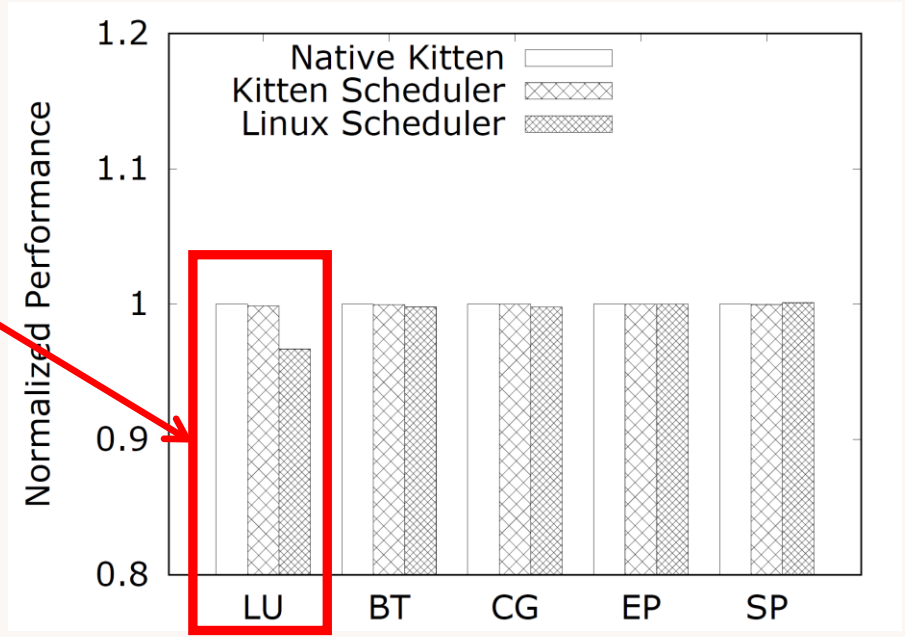
NAS Parallel Benchmarks

- Subset of NPB programs
  - Results reported as normalized



NAS Parallel Benchmarks

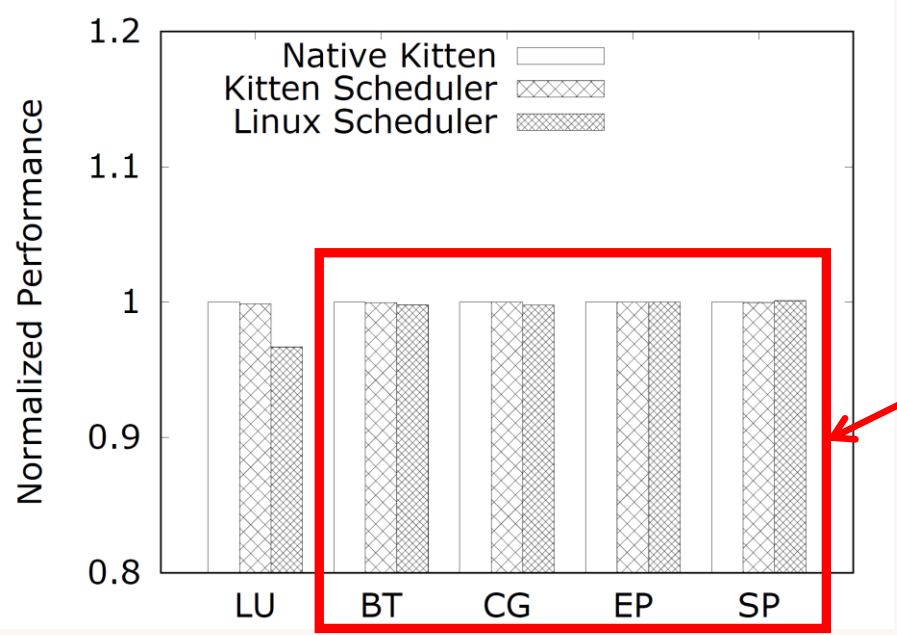
- Subset of NPB programs
  - Results reported as normalized



Linux scheduler degrades LU performance by ~3%

NAS Parallel Benchmarks

- Subset of NPB programs
  - Results reported as normalized



Future Work (Short term)

- Deploy on HPC class resources
  - Looking to support the Astra system (ThunderX2) at Sandia
  - ThunderX2 and A64FX testbed systems at Oak Ridge
- Full audit of Hafnium security features
  - Hafnium is a very restrictive environment
  - What restrictions are necessary vs overly cautionary
- Add support for Linux as a secondary
  - Will require extensive changes to Hafnium
  - Need better support for IO partitioning
  - Need to implement secure IRQ partitioning/routing

Future Work (Long Term)

- Hafnium is not designed for HPC
  - Static hardware partitions
  - Statically pre-configured VMs
  - Limited cross partition communication
- ARM hardware is changing
  - TEE capabilities are expanding in ARMv9
- **Claim**: There will be a need for a node level trusted hypervisor/partition manager designed specifically for HPC environments.
  - An open question is whether it will be hardware or software based

Conclusion

- Secure OS/R compartmentalization will be a key enabling feature post Exascale
  - Can be achieved on current and future hardware
- Trusted computing frameworks are designed for commodity use cases
  - There is a need and an opportunity for trusted computing system software designed specifically for HPC
- We have presented an initial proof of concept of one such approach

Questions?