Targeting performance isolation in an experimental testbed

Jack Lange
Assistant Professor
University of Pittsburgh
Workload Interference

• Cross workload interference is now recognized as a problem for consolidated cloud environments

• There are currently two approaches to address it
  – Modify application behaviors to be resilient to interference effects
    • E.g. Google ("The Tail at Scale" in CACM)
  – Use hardware isolation to prevent workloads from interfering
    • E.g. Amazon EC2 - Space sharing of workloads

• Our work focuses on providing isolation at both the hardware and System Software layers
Current systems do support this, but…

Interference still exists inside the system software
  – Inherent feature of commodity systems
Multi-stack Approach

• **Dynamic Resource Partitions**
  – Runtime segmentation of underlying hardware resources
  – Assigned to specific workloads

• **Dynamic Software Isolation**
  – Prevent interference from other workloads
  – Execute on separate system software stacks
  – Remove cross stack dependencies

• **Implementation**
  – Independent system software running on isolated resources
Multi-stack Architecture

• **Allow multiple dynamically created enclaves**
  – Based on runtime isolation requirements
  – Provides flexibility of fully independent OS/Rs
• Isolated Performance and resource management
Isolation Experiments

- **Goal:** Measure isolation capabilities of multi-stack system software
  - Measure application performance at scale with/without co-located workloads

- **Smallish scale evaluation of Co-VMM architecture**
  - **Native performance without background workloads**
  - **Better performance with background workloads as node count increases**
Testbed Requirements

• **Experiments at scale**
  – Evaluation of isolation approaches requires large scale experiments of custom system software
  – We need to run and measure our own OS/runtime directly on hardware at scale

• **Experimental Workloads**
  – Workloads exhibit varying behaviors and sensitivities to interference
  – We need access to a collection of realistic workloads to measure and study their behaviors
Experiments at scale

• Requirements
  – Ability to run custom kernel images
  – Root/sudo access capabilities to insert custom kernel modules and configure system software
  – Hardware support for partitioning
    • SR-IOV capable PCI devices, multi socket/NUMA systems, IOMMUs, etc...
Experimental Workloads

• Requirements
  – Ability to gather low level performance measurements to identify interference problems
    • At both the hardware and software layers
    • Hardware perf counters, Linux tracepoints, etc

• Preferences
  – Ability to instrument and gather performance measurements in a centralized manner
  – Library of public workloads gathered from actual system users
    • Preconfigured real world benchmarks