Execution Flow Analysis Across Virtualized Environments for performance understanding and optimisation

Naser Ezzati, Hani Nemati (Polytechnique Montreal - Dorsal Lab)

François Tétreault (Ciena)

October 25, 2018
Agenda

Introduction
- Motivation
- Different Layers of virtualization

New Investigation
- Proposed Approach

Evaluation
- Slow Nested VM
- Nested VM misconfiguration
- Linux Advance Packaging Tool Analysis
- Undesirable parallelism

TraceCompass Update

Demo

Conclusion
Motivation

Emulation and simulation environment are widely used in the industry when developing new products.

There is a rich variety of virtualization technology that is readily available.

- Emulation
- Containers
- Software virtualization (emulation)
- Hardware-assisted virtualization
- Paravirtualization

When working on very large complex projects, where do you start to achieve the best performance and scalability?

For example, you may want to simulate a network configuration with a very large number of Network Elements (NEs). Some NEs may be network nodes with multiple cards and compute systems (Processor Daughter cards and partitions).
Motivation

Sample Node Simulation Configuration

- Each compute system is simulated in each own VM
- Cards and nodes are collectively running distributed applications over heterogeneous operating systems
- A virtual bridge is used to emulate the communication protocols (comms) in between the cards
Motivation

Running the simulation environment

Simulation running on a single host

Simulation running in the cloud
## Motivation

<table>
<thead>
<tr>
<th>Goals</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simulation Performance</strong>&lt;br&gt; - Time to boot and shutdown the sim&lt;br&gt; - Running software performance</td>
<td>Same performance as bare metal (as in the actual product)</td>
</tr>
<tr>
<td><strong>Simulation Scalability</strong>&lt;br&gt; - Number of concurrent sims running on a single host&lt;br&gt; - Number nodes supported for large network simulations</td>
<td>1 to 10 nodes&lt;br&gt; 1,000 to 10,000 nodes</td>
</tr>
<tr>
<td><strong>Software Upgrade Simulation</strong></td>
<td>Get the best performance possible on both hardware and on sim</td>
</tr>
<tr>
<td><strong>Select the most optimal host machine for running the sim</strong>&lt;br&gt; - for non-nested configurations, and&lt;br&gt; - for the cloud</td>
<td>Most favorable “best bang for the buck”:&lt;br&gt; - CPU performance and features&lt;br&gt; - Number of cores&lt;br&gt; - L1/L2 cache sizes&lt;br&gt; - RAM size&lt;br&gt; - File System size and technology&lt;br&gt; - Hardware virtualization features</td>
</tr>
</tbody>
</table>
Challenges

There is a lot of software involved, especially when including a nested configuration with all the software in Layer 0, Layer 1 and Layer 2.

Many angles to consider:

- **Layer 0: Host**
  - Machine capabilities
  - BIOS configuration
  - OS, Kernel and Library versions

- **Layer 1: VM Host**
  - OS, Kernel and Library versions

- **Layer 2: VM Guest OS**
  - OS, Kernel and Library versions
  - Software running on the simulation environment

Layers are segregated from each other, by design and for security.

Traditional tools and techniques don’t apply or are sub-optimal
Motivation

Virtual Machine Hierarchy
Motivation

Hierarchical Virtualized Environments - Nested VM

Virtual Machines

Nested VM Level - L2

VM Level - L1

Host Level - L0

Host OS: Linux

KVM

VMM

Guest OS

Nested VM OS

Nested VM OS

Hardware

KVM

VMM

Guest OS

Nested VM OS

Nested VM OS

Host Level - LO

VM Level - L1

Nested VM Level - L2
Motivation

VM Analysis features in TraceCompass

- Fused Virtual Machine Analysis (Trace Host and VMs)
- Works for VMs and Containers but needs trace synchronization
Motivation

1) Install a tracer on each VM
2) Trace them
3) Sync the traces
Motivation

Virtual Machine Hierarchy - Arbitrary Guest OS
Motivation

Virtual Machine Hierarchy

Virtual Machines

- Restricted Access
- Limited Resources
- Old Kernel
- Windows Mac OS

Host OS: Linux

Hardware
Motivation

Is there any method that preferably limits its data collection to the physical *host level*?
Motivation

virtFlow
Investigation

virtFlow features

- Hierarchical vCPU view for VM
- Running States
- Wait States

Virtual Machines

- Nested VM OS
- Nested VM OS

vCPU view for Tracecompass

- Nested VM vCPU view for Tracecompass

Host OS: Linux

KVM

Hardware
Investigation

virtFlow features

Hierarchical Process view for VM

Running States

Wait States

Virtual Machines

Nested VM OS

Nested VM OS

VMM

Guest OS

KVM

Host OS: Linux

Hardware
**Investigation**

**VM Analysis through Hierarchical Virtualized Environments**

---

**ControlFlow view**

1. sched_switch(in=qemu_thread)
2. inj_virq(vec=timer)
3. vm_entry(vcpu0, cr3#0)
4. vm_exit(reason=12)
5. sched_switch(out=qemu_thread)

---

**Methodology**

**Nested vCPU view**

---

**vCPU view**

---

**Nested vCPU view**

---
Investigation

VM Analysis through Hierarchical Virtualized Environments

Methodology

Nested vCPU view

ControlFlow view

qemu-thread

```
1 sched_switch(in=qemu_thread)
2 inj_virq(vec=disk)
3 vm_entry(vcpu0, cr3#1)
4 vm_exit(reason=24)
5 vm_entry(vcpu0, cr3#2)
6 vm_exit(reason=12)
7 vm_entry(vcpu0, cr3#1)
8 vm_exit(reason=12)
9 sched_switch(out=qemu_thread)
```
Investigation

Nested VM Misconfiguration

- Overhead of Nested VM

- Overhead of Non-Nested VM
Two Nested VMs and One VM are preempting each other

- Slow down for NestedVM 2
  - Preempted by NestedVM 1 and VM testU2
Software Upgrade Scenario

An upgrade strategy often used in the telecommunication industry is best referred to as “rolling upgrade”.

- A tactic to avoid any system downtime where cards or compute systems are sparing each other.
- A primary (aka master), is active and carrying services, and
- A secondary (aka slave), is ready to take over in case a primary service goes down

Typical sequence of steps for a rolling upgrade:

1. The secondary first upgrades to the new load while the primary remains active on the previous load.

2. Once the secondary has finished to upgrade into the new load, and applications on that card are ready to take over, a switch-over occurs from the primary to the secondary.

3. The secondary then becomes primary, and vice versa.

4. The secondary then upgrades into the new load and synchronizes with the active application in order to be ready to take control.
Motivation

Simulation and bare metal upgrade performance enhancements

- Automation testing
- DevOps - Software load sanity and regression testing
Investigation

virtFlow features
Critical Path Analysis through
Hierarchical Virtualized Environments
Investigation

Distributed Virtualized Environments

Virtual Machine
Guest OS
KVM
Host OS: Linux
Hardware

Virtual Machine
Guest OS
KVM
Host OS: Linux
Hardware

Virtual Machine
Guest OS
KVM
Host OS: Linux
Hardware
Motivation

virtFlow features

Critical Path Analysis through Distributed Virtualized Environments
Containers within Virtualized Environments

- Virtual Machines
- Container
- Container
- Docker Engine
- Guest OS
- KVM
- Host OS: Linux
- Hardware
Containers within Virtualized Environments

Virtual Machines

- Restricted Access
- Limited Resources
- Old Kernel
- Windows Mac OS

Host OS: Linux

Hardware
Containers within Virtualized Environments

Virtual Machines

Host OS: Linux

Hardware

KVM

Restricted Access

Limited Resources

KVM

Containers within Virtualized Environments

Virtual Machines

Host OS: Linux

Hardware

KVM

Restricted Access

Limited Resources

KVM

Exclusive Access
Investigation

Critical Path Analysis

Linux Advance Packaging Tool

What is going on here?

1) apt-get downloads and reads cached packages
2) apt-get installs the packages along with downloaded dependencies
3) The installation of man-pages
Investigation

Critical Path Analysis

Undesirable parallelism

waits for disk

waits for another process
Investigation

Critical Path Analysis

Existing Critical Path Analysis in TraceCompass

Host-based Execution-graph Construction

Preemption State
## Investigation

### Overhead Analysis

**CPA**: Existing Critical Path Analysis in TraceCompass  
**HEC**: Host-based Execution-graph Construction

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Baseline (ms)</th>
<th>CPA (ms)</th>
<th>HEC (ms)</th>
<th>CPA Overhead (%)</th>
<th>HEC Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>File I/O</td>
<td>450.92</td>
<td>480.38</td>
<td>451.08</td>
<td>6.13%</td>
<td>0.03%</td>
</tr>
<tr>
<td>Memory</td>
<td>612.27</td>
<td>615.23</td>
<td>614.66</td>
<td>4.81%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CPU</td>
<td>324.92</td>
<td>337.26</td>
<td>325.91</td>
<td>3.65%</td>
<td>0.30%</td>
</tr>
</tbody>
</table>
Tracecompass Update

- State System Explorer
- Export views to image
- Time event highlighting and filtering
- Resources View Enhancements
  - Resources View shows active threads
  - Resources View shows CPU frequency when available
- CTF trace trimming
- Enabling and disabling XML analysis files
Demo
Investigation

How to try these new features?

- Access to **Host** only

- Run **LTTng** on Host with my new added tracepoint (vcpu_enter_guest) [2]

- Clone **TraceCompass** from github [2] (incubator)
  - Open vCPU block View of TraceCompass (XML view)
  - Open vProcess block View of TraceCompass (XML view)
  - Open Nested VM vCPU Block View of TraceCompass (XML view)
  - Open Nested VM vProcess Block View of TraceCompass (XML view)
  - Use Execution Flow Analysis of TraceCompass

Conclusion

VM Analysis using Host Kernel tracing

- vCPU analysis of VM and nested VM
- vProcess analysis of VM and nested VM
- Wait analysis of VM and nested VM
- Critical path analysis of VM and nested VM

Resource performance analysis:

- **CPU**: Avoiding CPU overcommitment, CPU host configuration, VM thread/process contention, cache configuration
- **Disk**: SSD/HDD for VM, virtio drivers for VM, Cap on disk, contention on disk, Cache configuration
- **Networking**: virtio, virt-host-net, cap on network
- **Memory**: Cache Analysis, Memory overcommitment
Questions?

hani.nemati@polymtl.ca
https://github.com/Nemati
ftetreau@ciena.com
https://www.linkedin.com/in/francoistetreault/