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

ciena
Experience. Outcomes.

POLYTECHNIQUE
MONTRÉAL
WORLD-CLASS
ENGINEERING

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

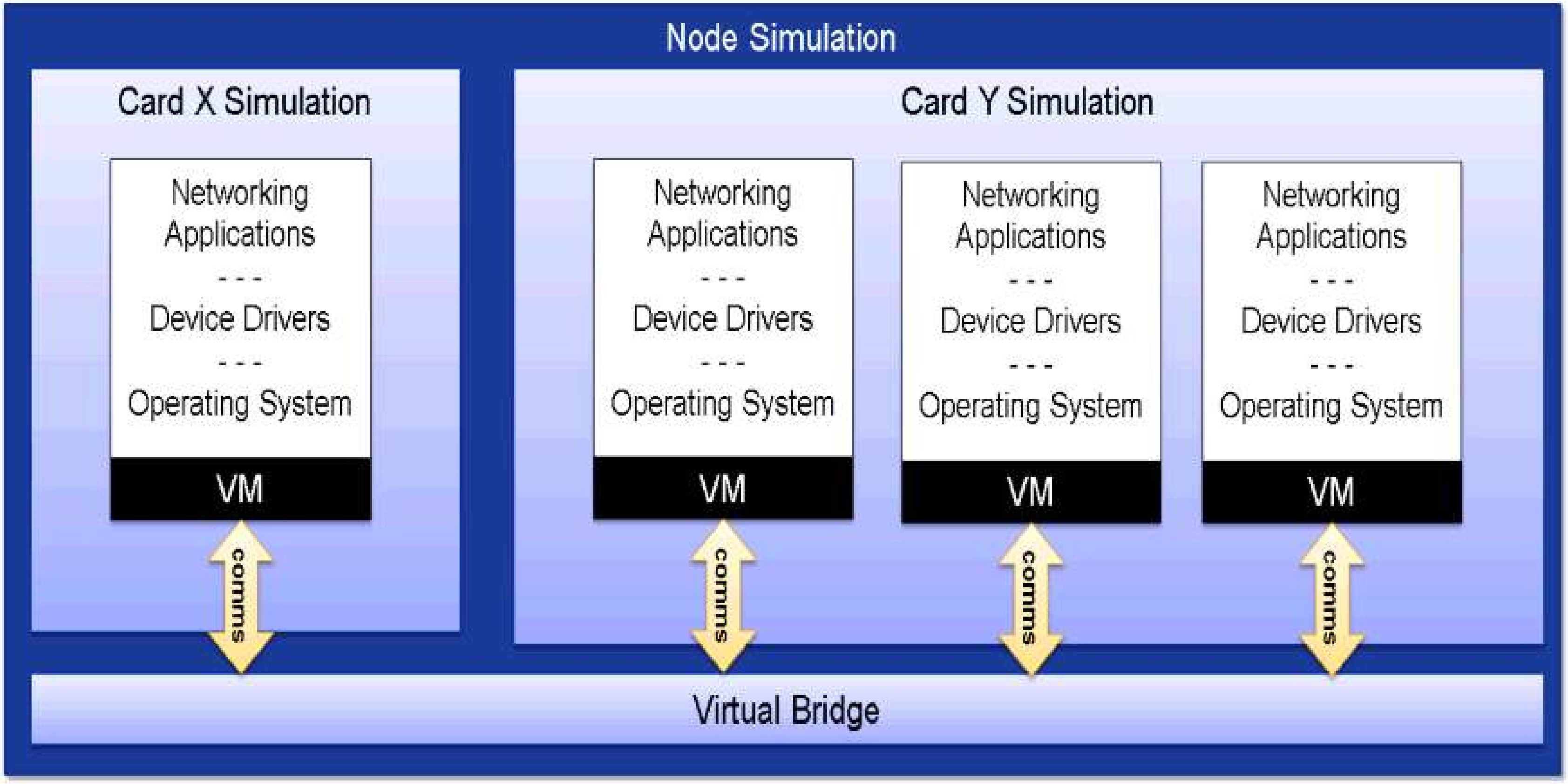
When working on very large complex projects, where do you start to achieve the best performance and scalability?

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

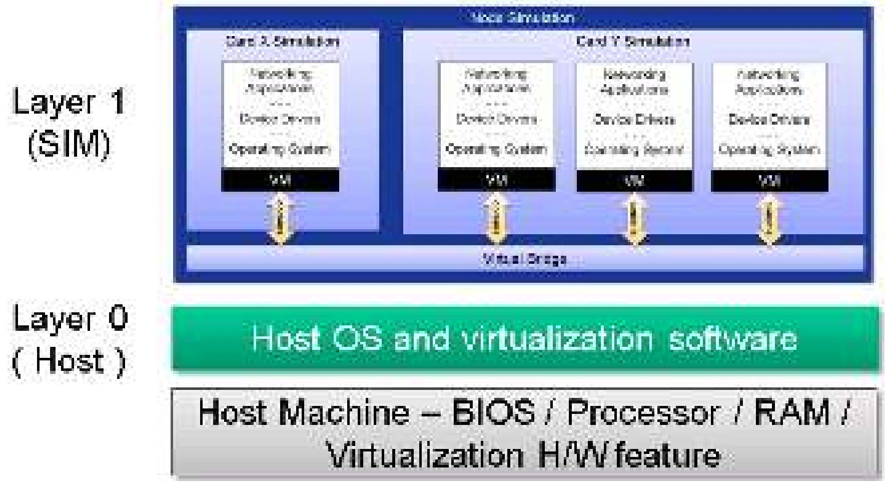
- Each compute system is simulated in each own VM
- Cards and nodes are collectively running distributed applications over heterogeneous operating systems
- A virtual bridge is used to emulate the communication protocols (comms) in between the cards



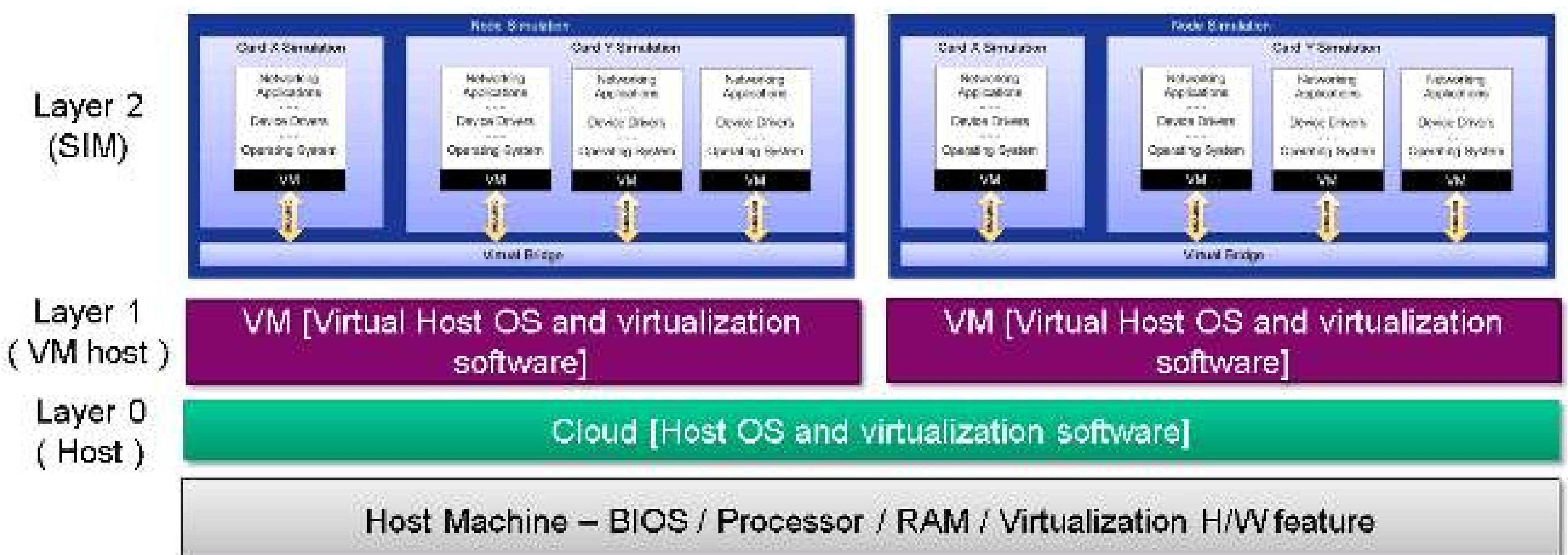
Motivation

Running the simulation environment

Simulation running on a single host



Simulation running in the cloud



Motivation

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

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:

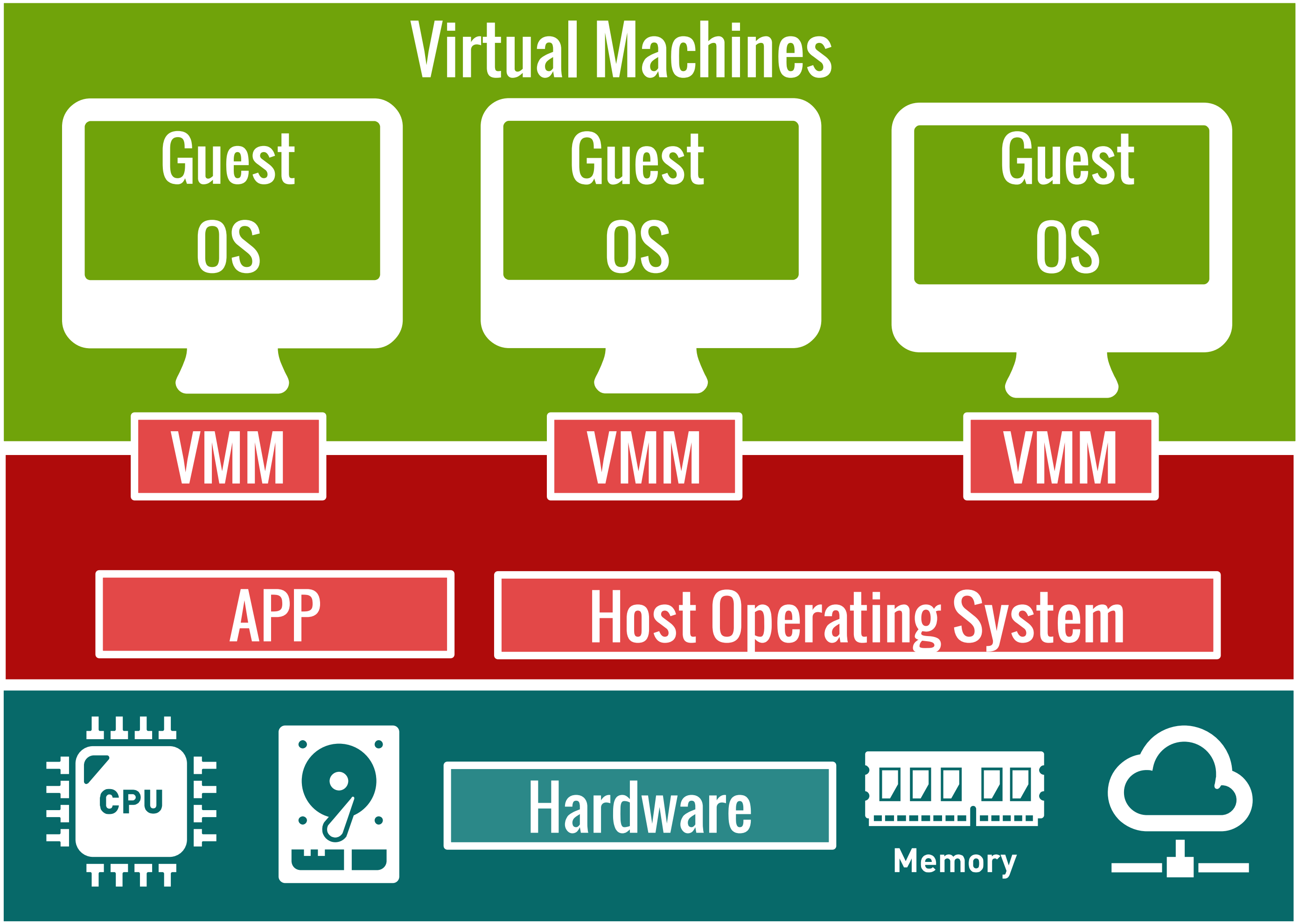
- Layer 0: Host
  - Machine capabilities
  - BIOS configuration
  - OS, Kernel and Library versions
- Layer 1: VM Host
  - OS, Kernel and Library versions
- Layer 2: VM Guest OS
  - OS, Kernel and Library versions
  - Software running on the simulation environment

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

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

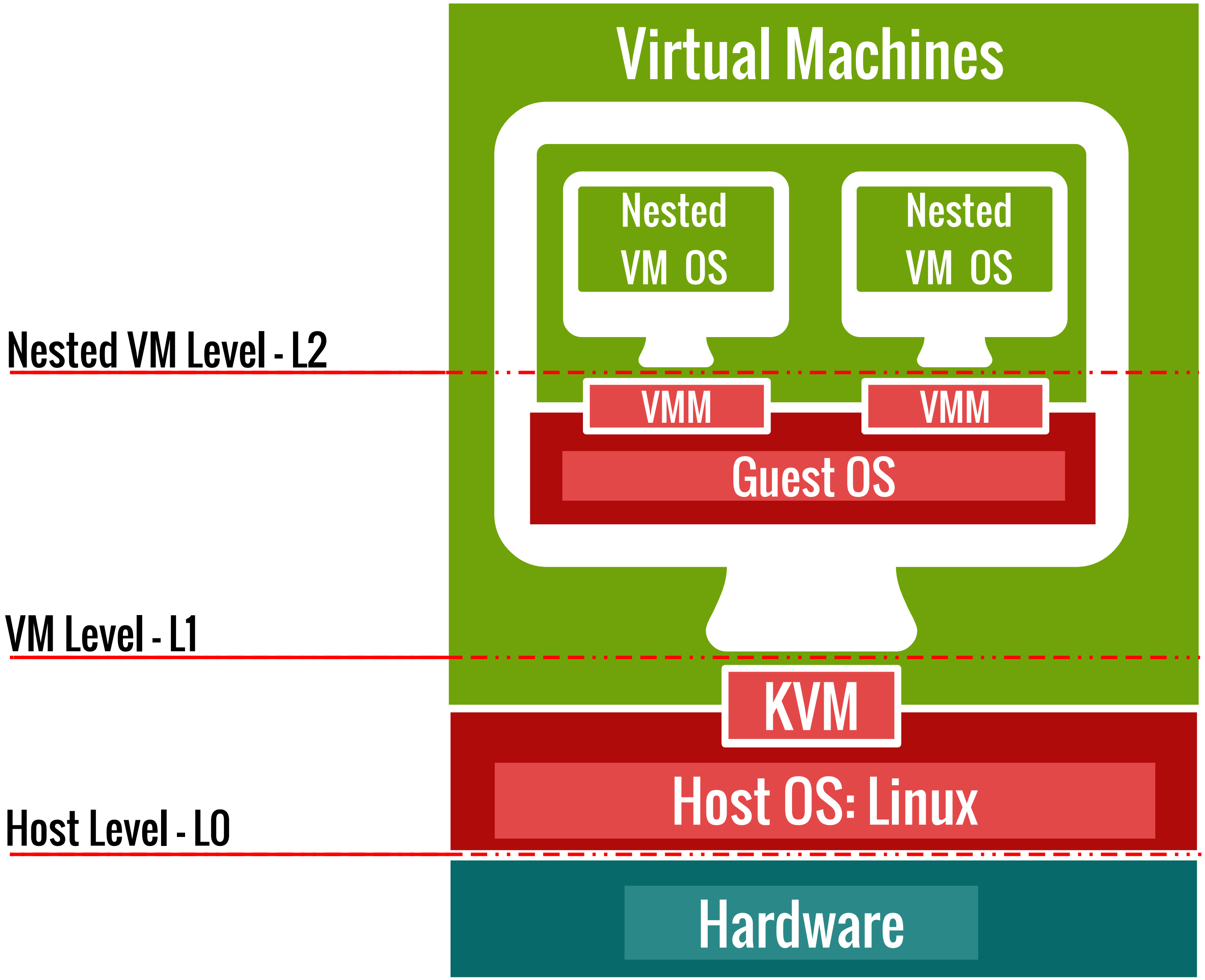
Motivation

Virtual Machine Hierarchy



Motivation

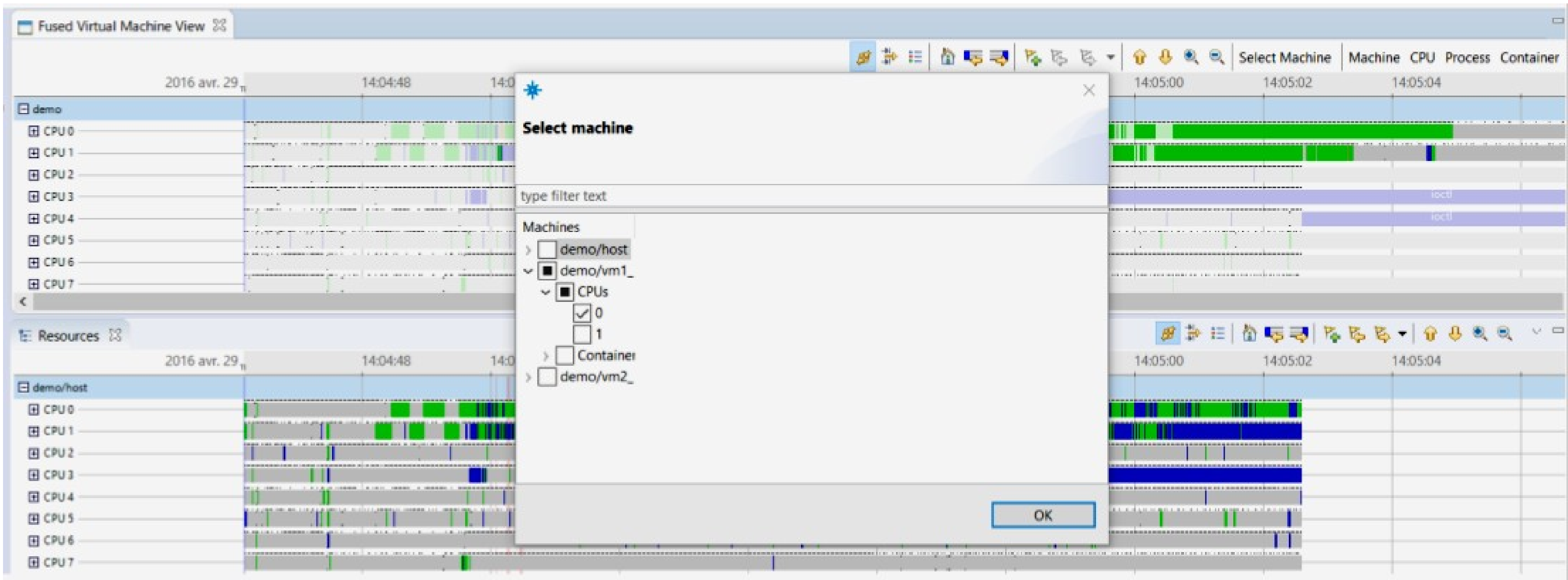
Hierarchical Virtualized Environments - Nested VM



Motivation

VM Analysis features in TraceCompass

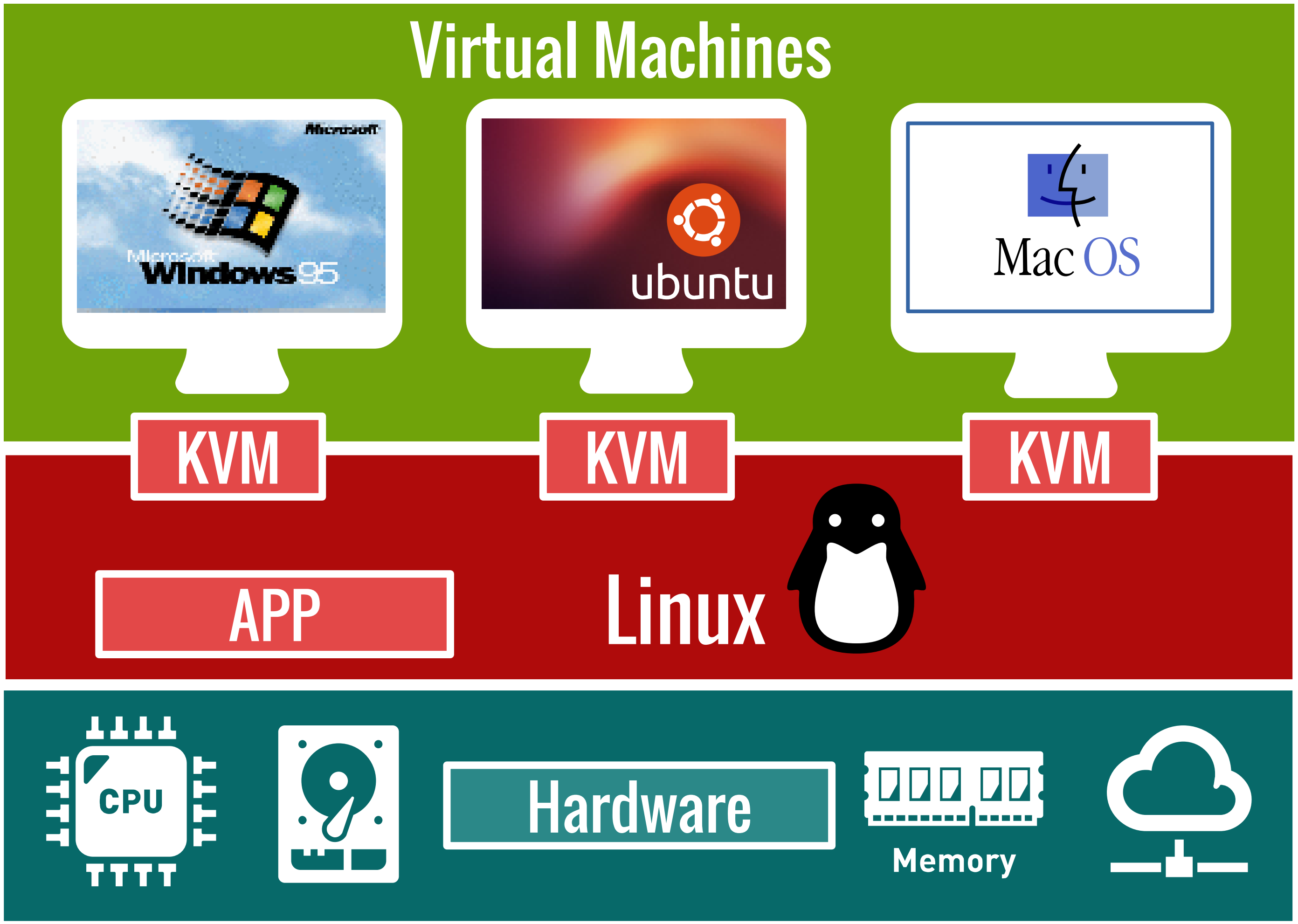
- Fused Virtual Machine Analysis (Trace Host and VMs)
  - Works for VMs and Containers but needs trace synchronization



Motivation

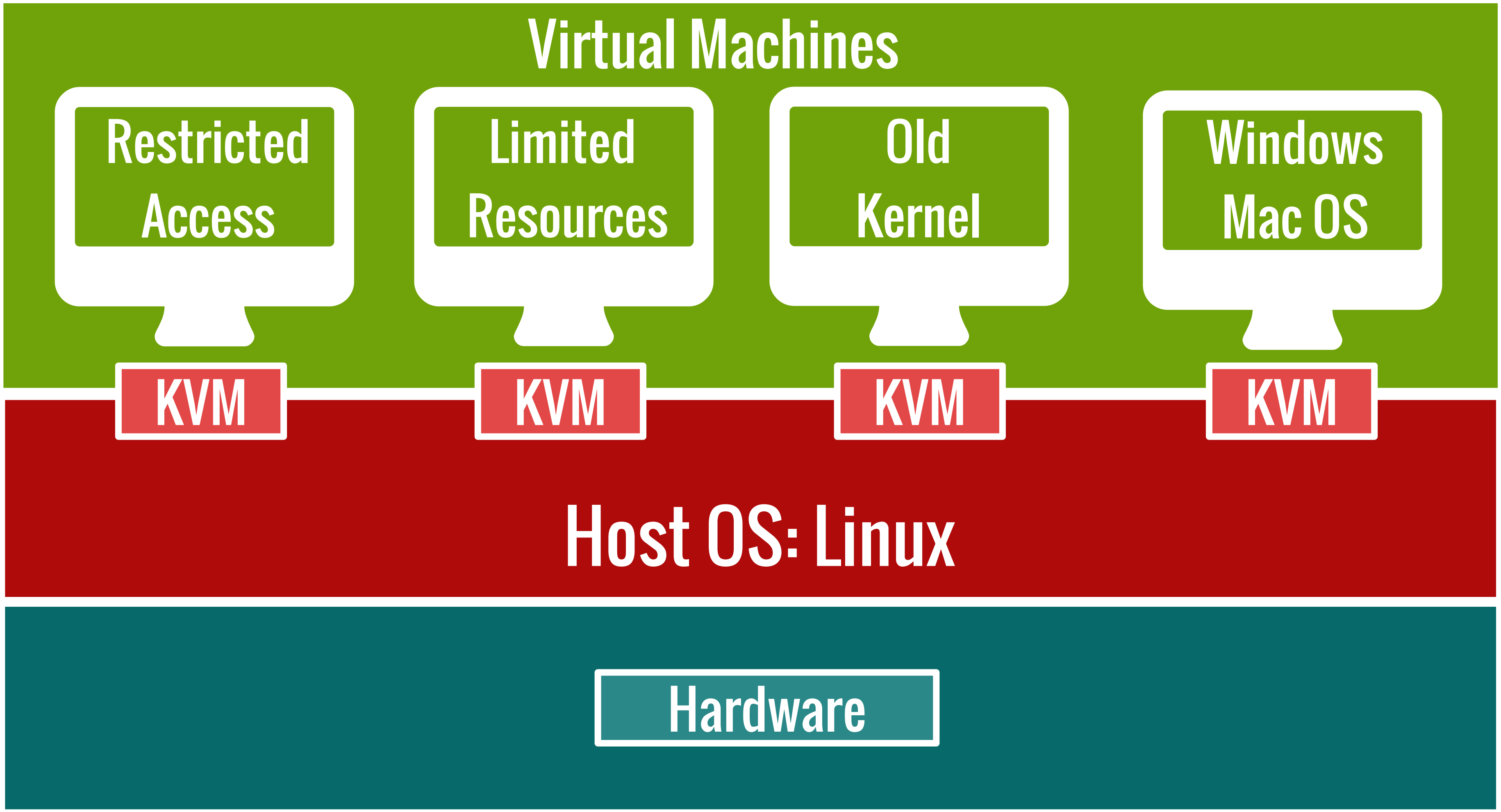
Motivation

Virtual Machine Hierarchy - Arbitrary Guest OS

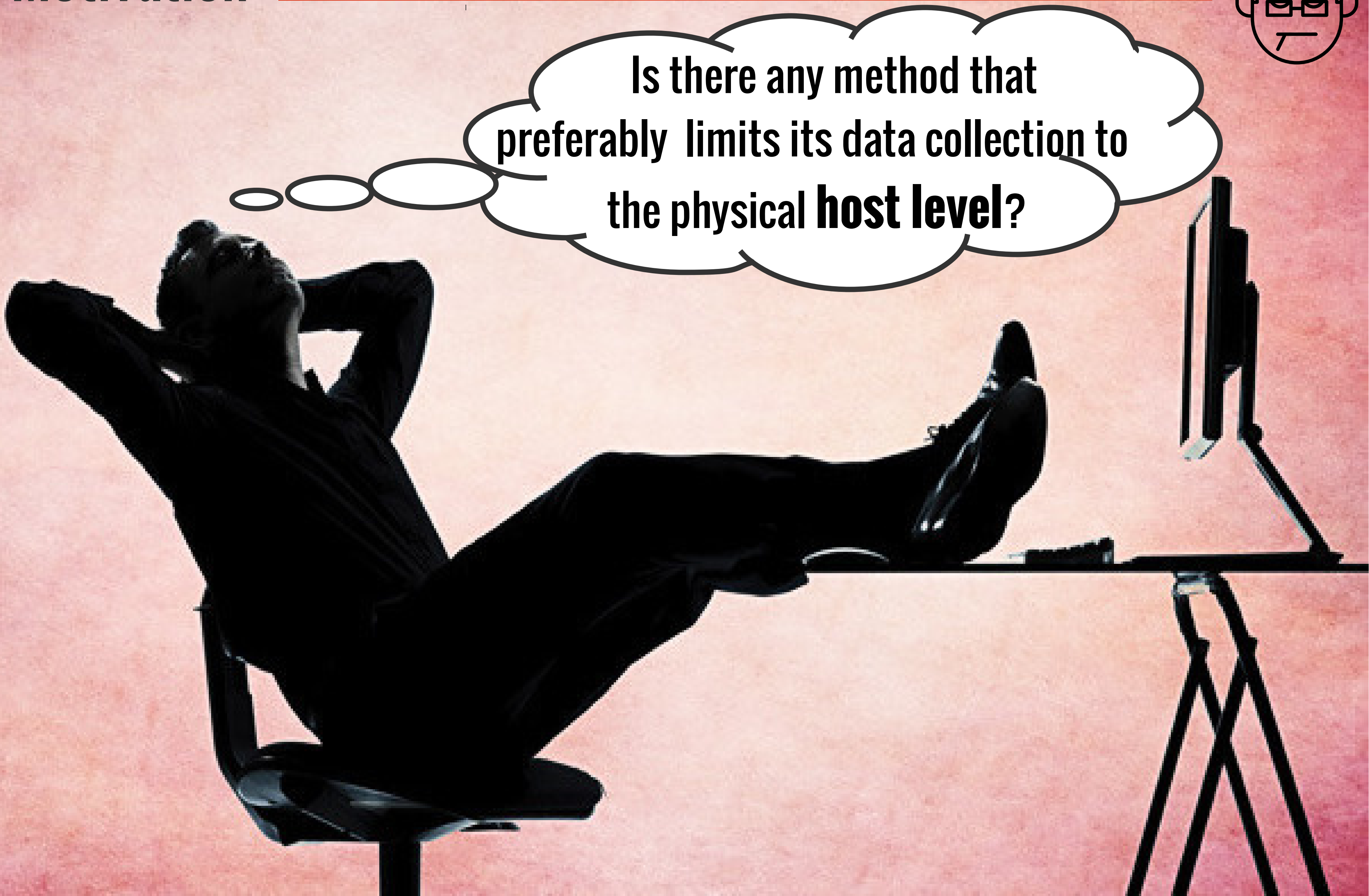


Motivation

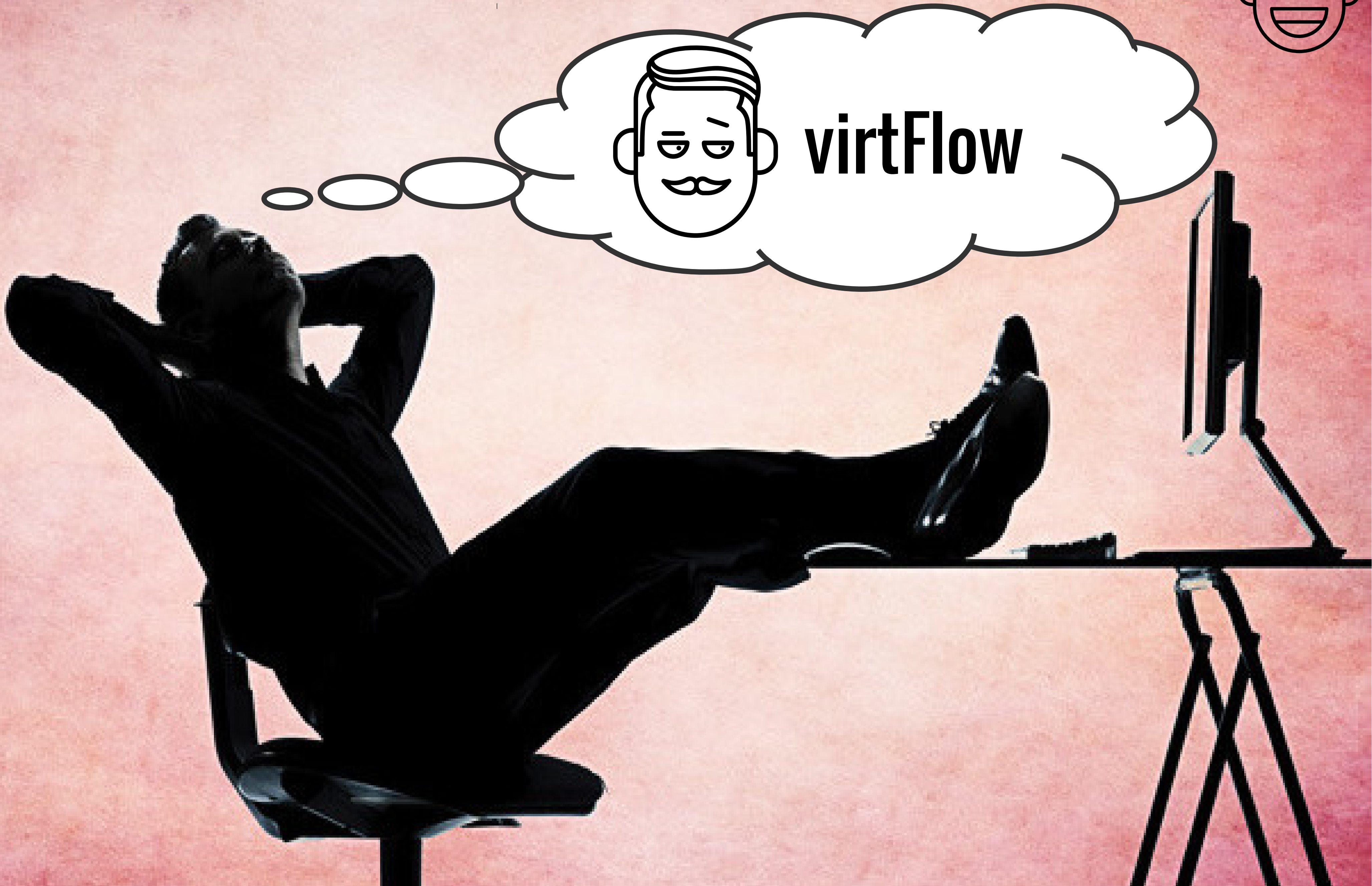
Virtual Machine Hierarchy



Motivation


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

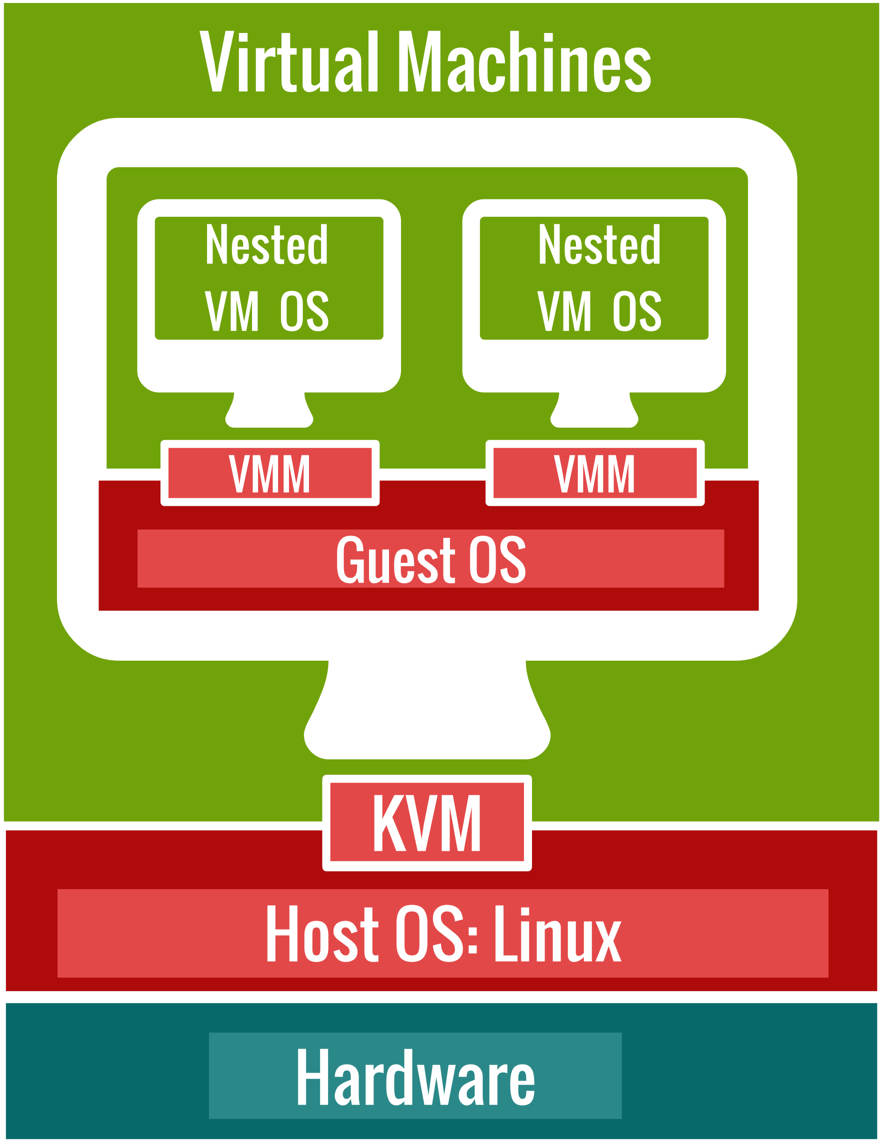
Motivation



Investigation

virtFlow features

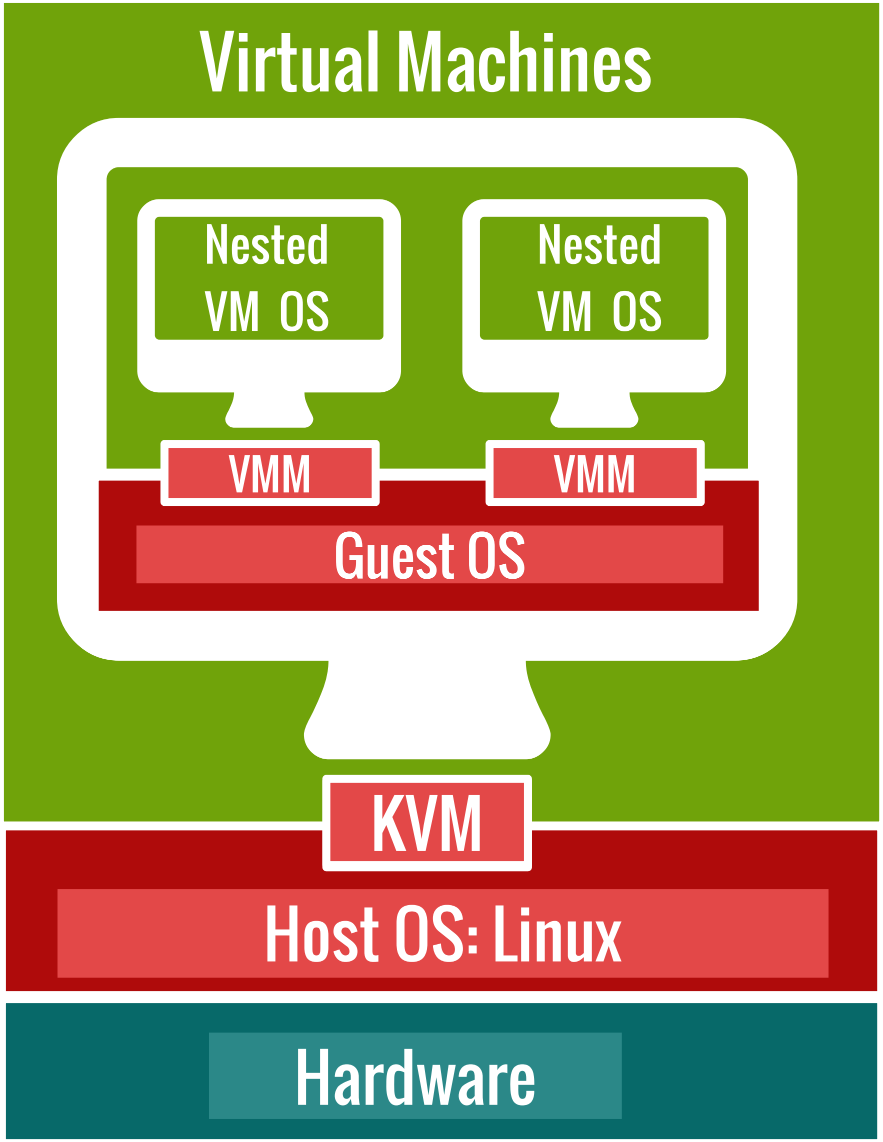
- Hierarchical vCPU view for VM
  - Running States
  - Wait States



Investigation

virtFlow features

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

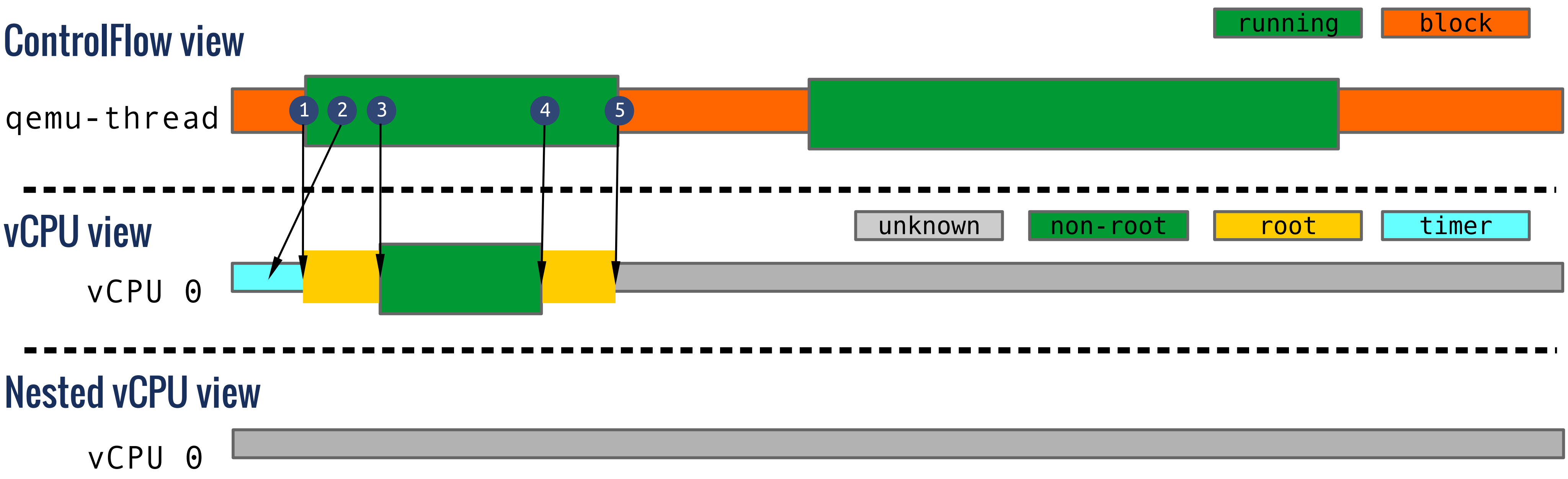


Investigation

VM Analysis through Hierarchical Virtualized Environments

Methodology

Nested vCPU view



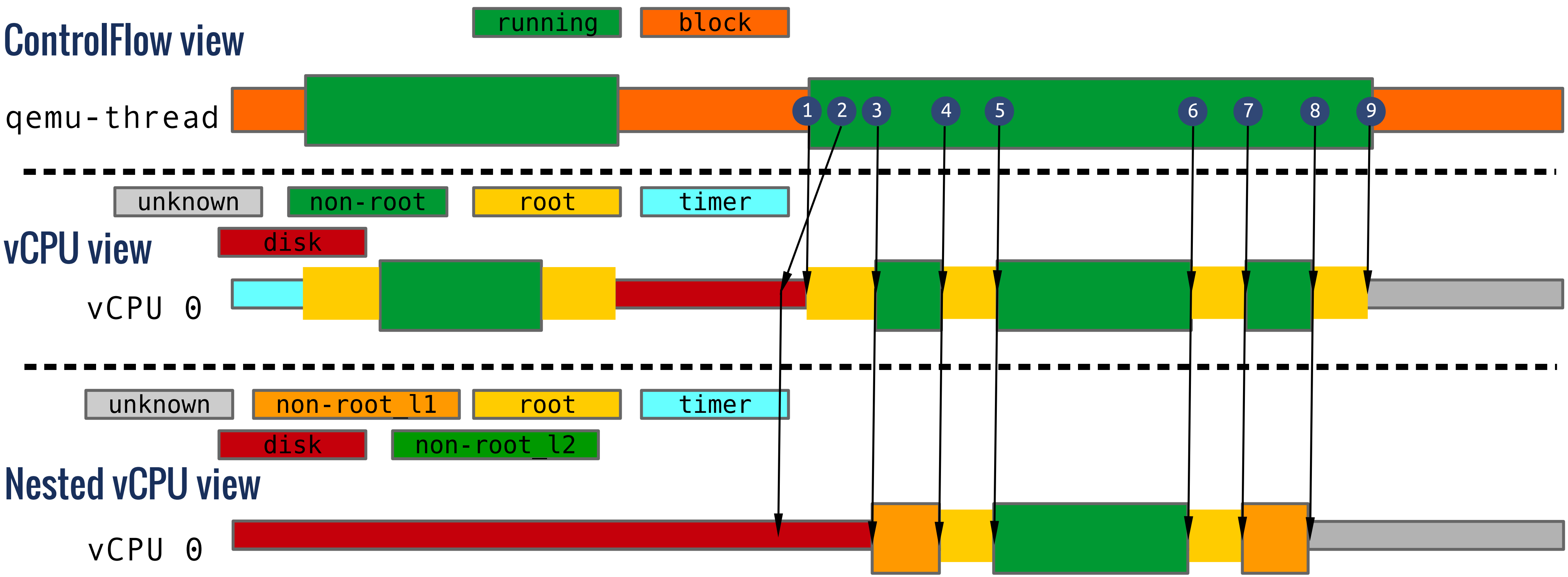
1. sched_switch(in=qemu_thread)
2. inj_virq(vec=timer)
3. vm_entry(vcpu0, cr3#0)
4. vm_exit(reason=12)
5. sched_switch(out=qemu_thread)

Investigation

VM Analysis through Hierarchical Virtualized Environments

Methodology

Nested vCPU view

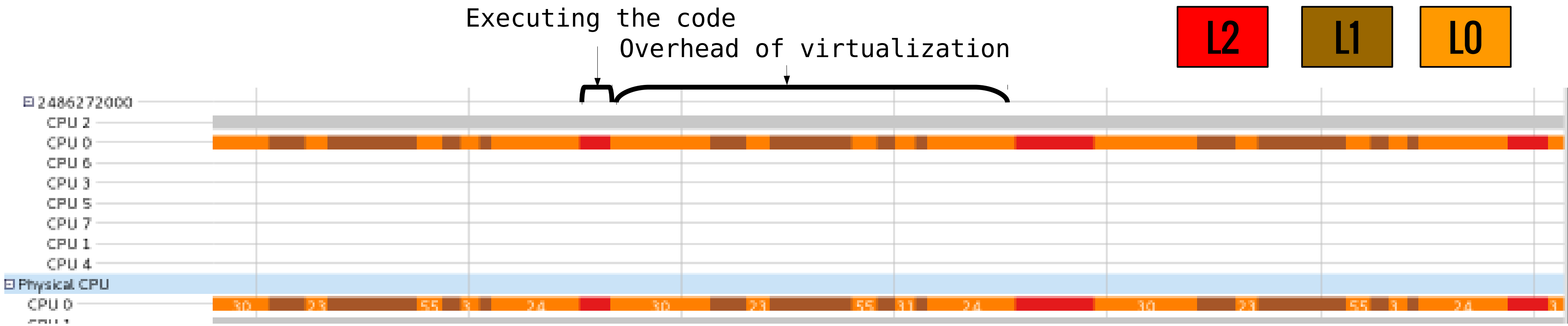


1. sched_switch(in=qemu_thread)
2. inj_virq(vec=disk)
3. vm_entry(vcpu0, cr3#1)
4. vm_exit(reason=24)
5. vm_entry(vcpu0, cr3#2)
6. vm_exit(reason=12)
7. vm_entry(vcpu0, cr3#1)
8. vm_exit(reason=12)
9. sched_switch(out=qemu_thread)

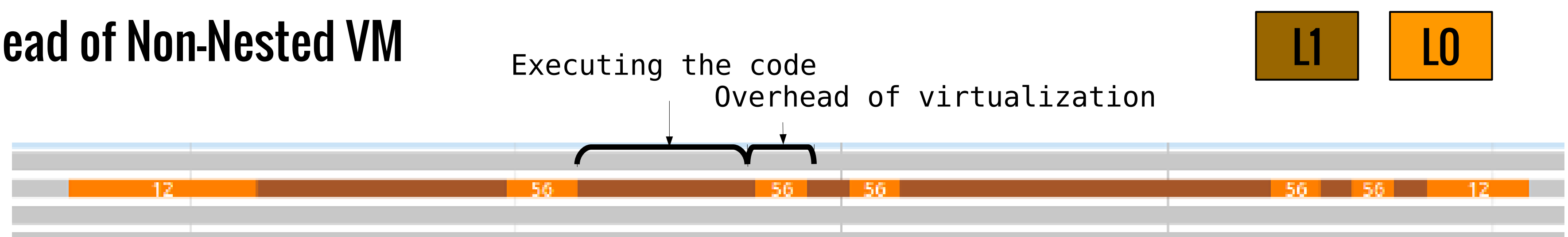
Investigation

Nested VM Misconfiguration

- Overhead of Nested VM



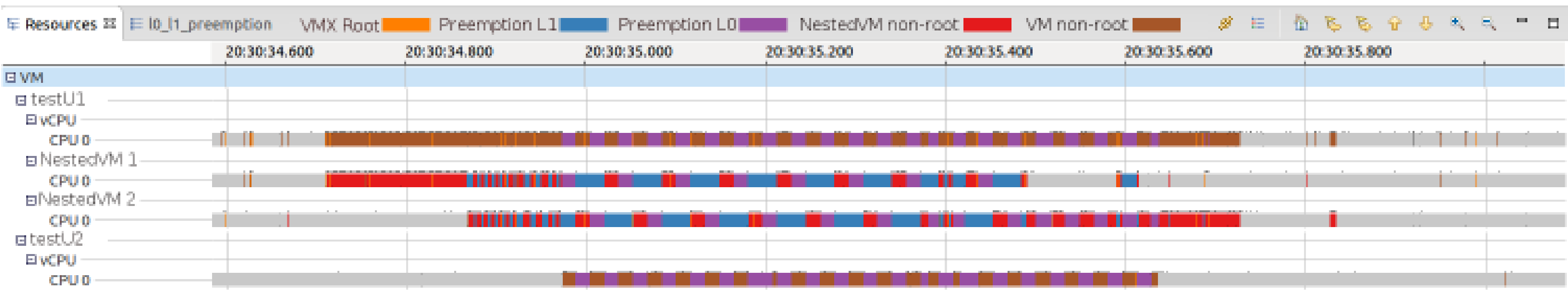
- Overhead of Non-Nested VM



Investigation

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

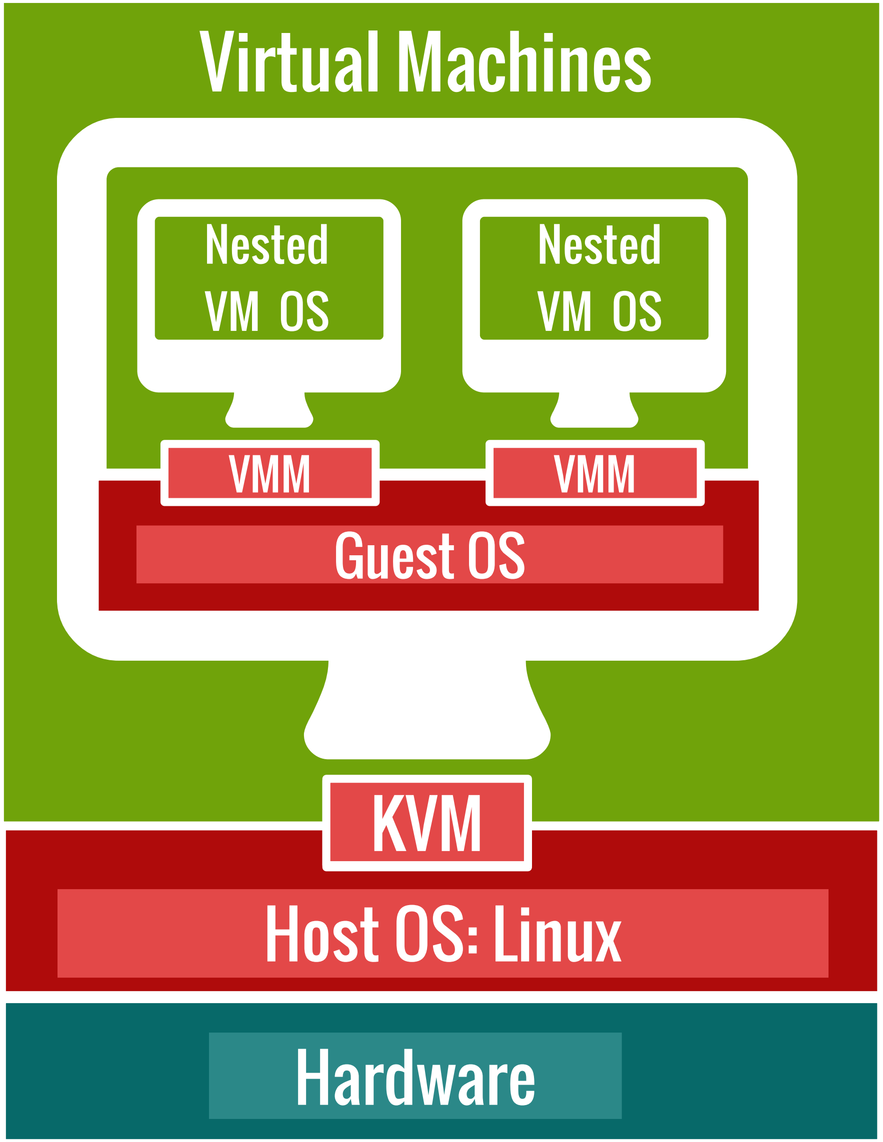
Simulation and bare metal upgrade performance enhancements

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

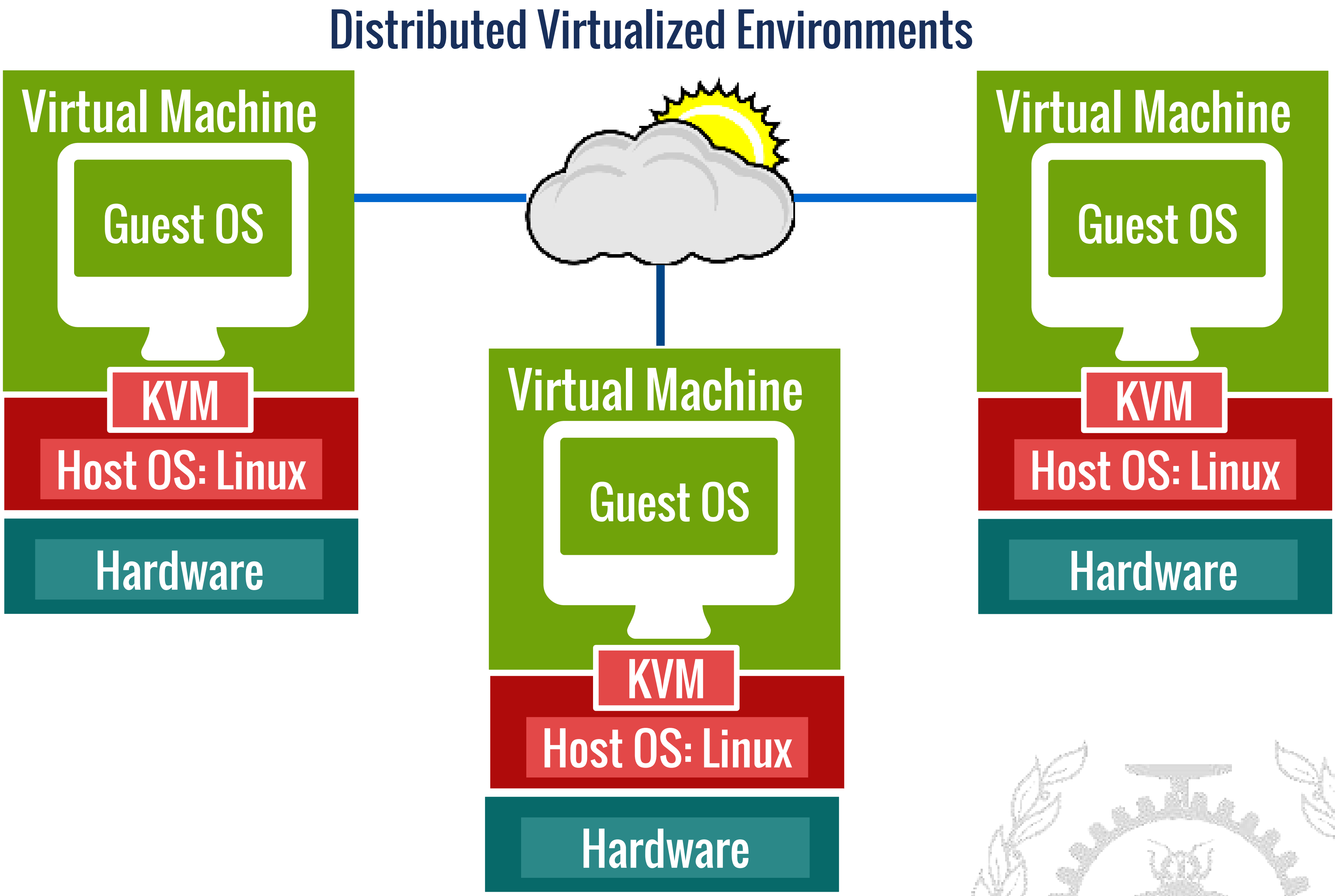
Investigation

virtFlow features

Critical Path Analysis through
Hierarchical Virtualized Environments



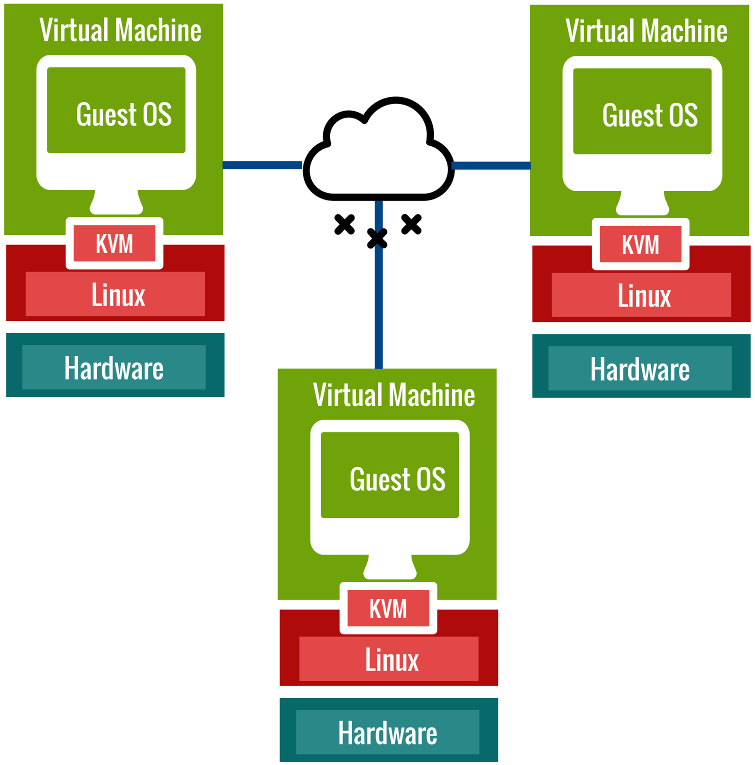
Investigation



Motivation

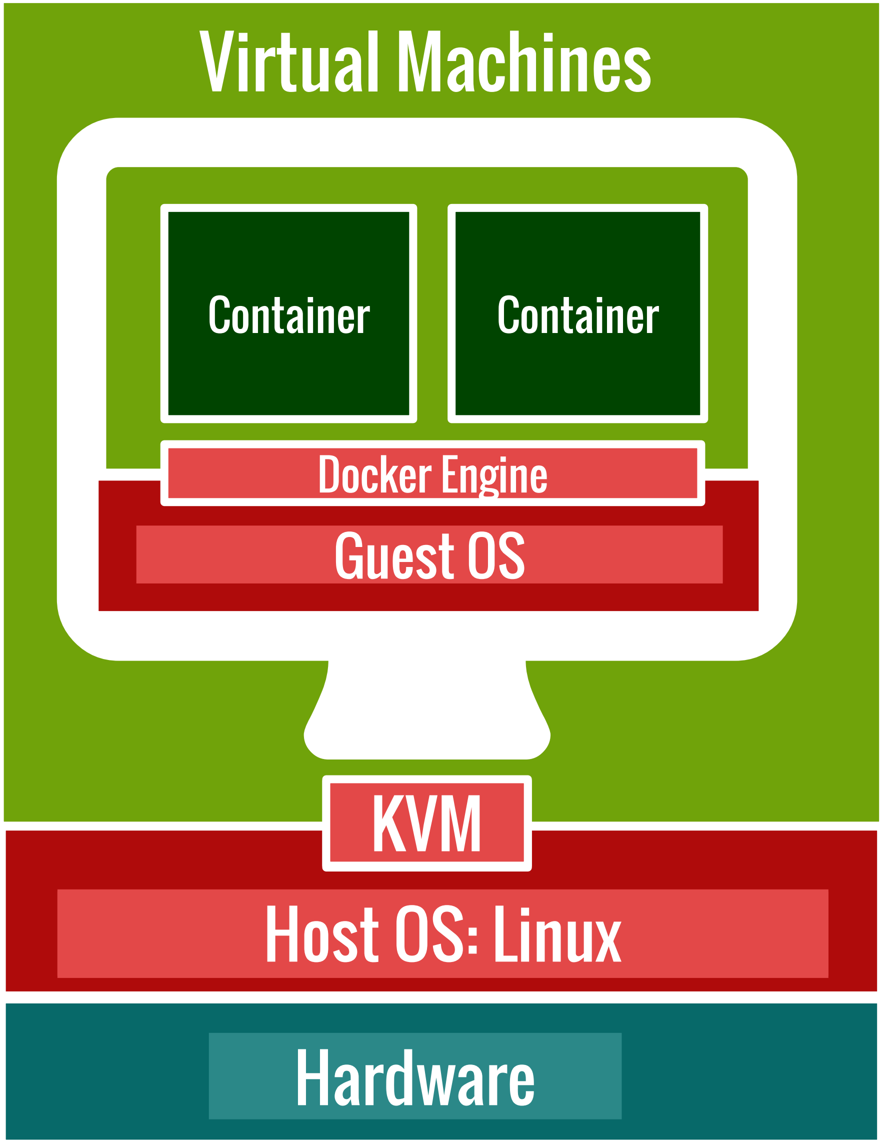
virtFlow features

Critical Path Analysis through Distributed Virtualized Environments



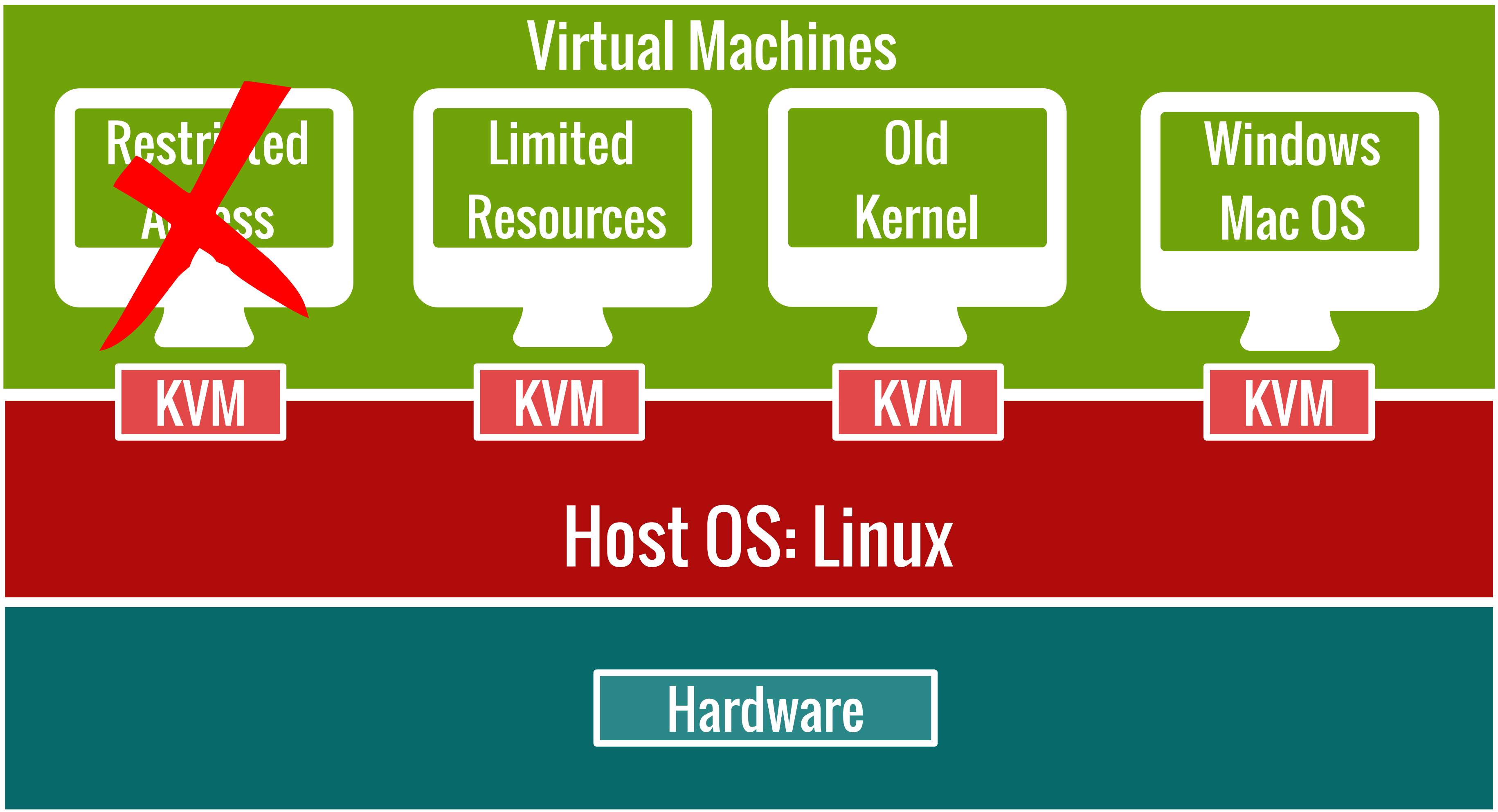
Investigation

Containers within Virtualized Environments



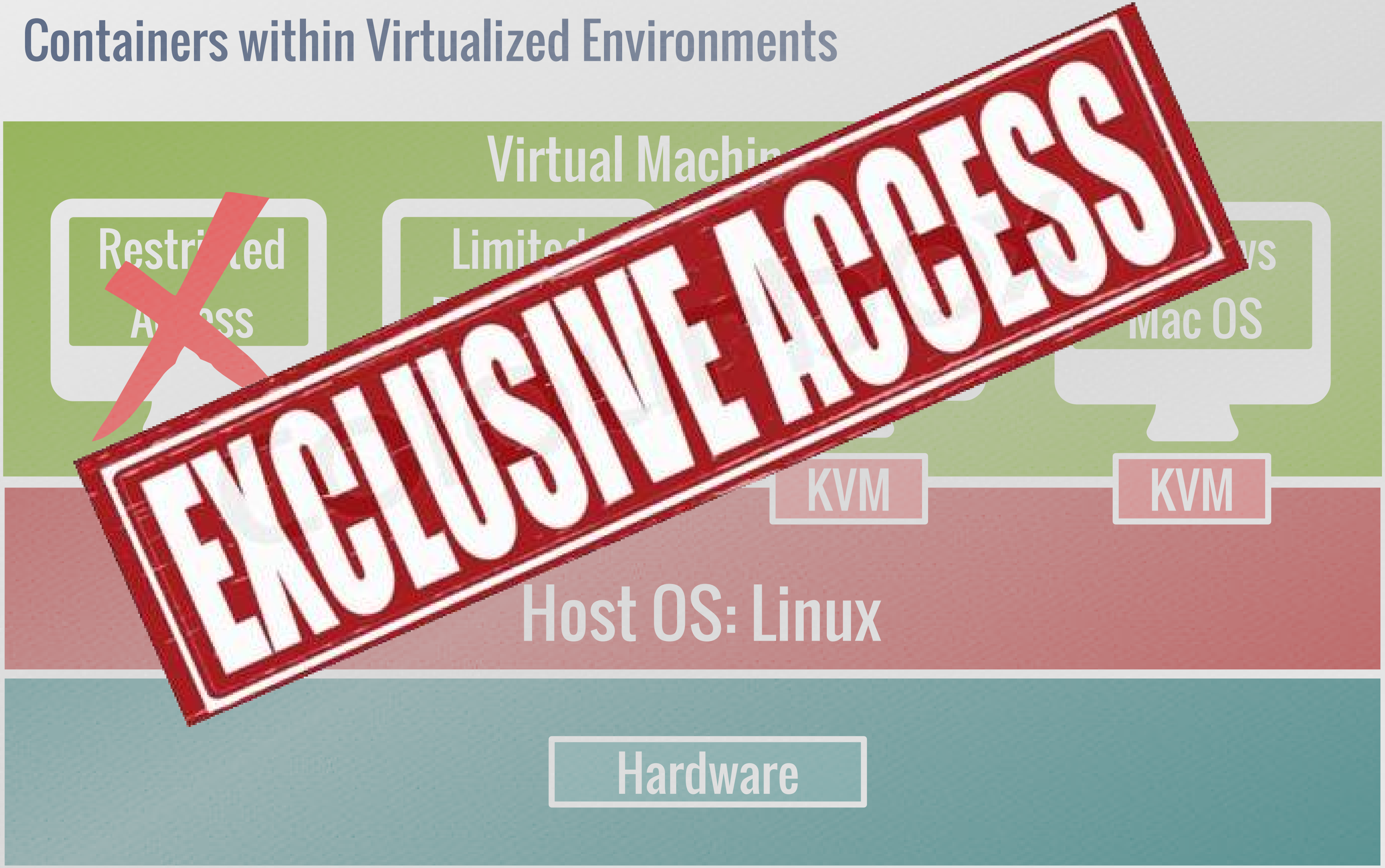
Investigation

Containers within Virtualized Environments



Investigation

Containers within Virtualized Environments

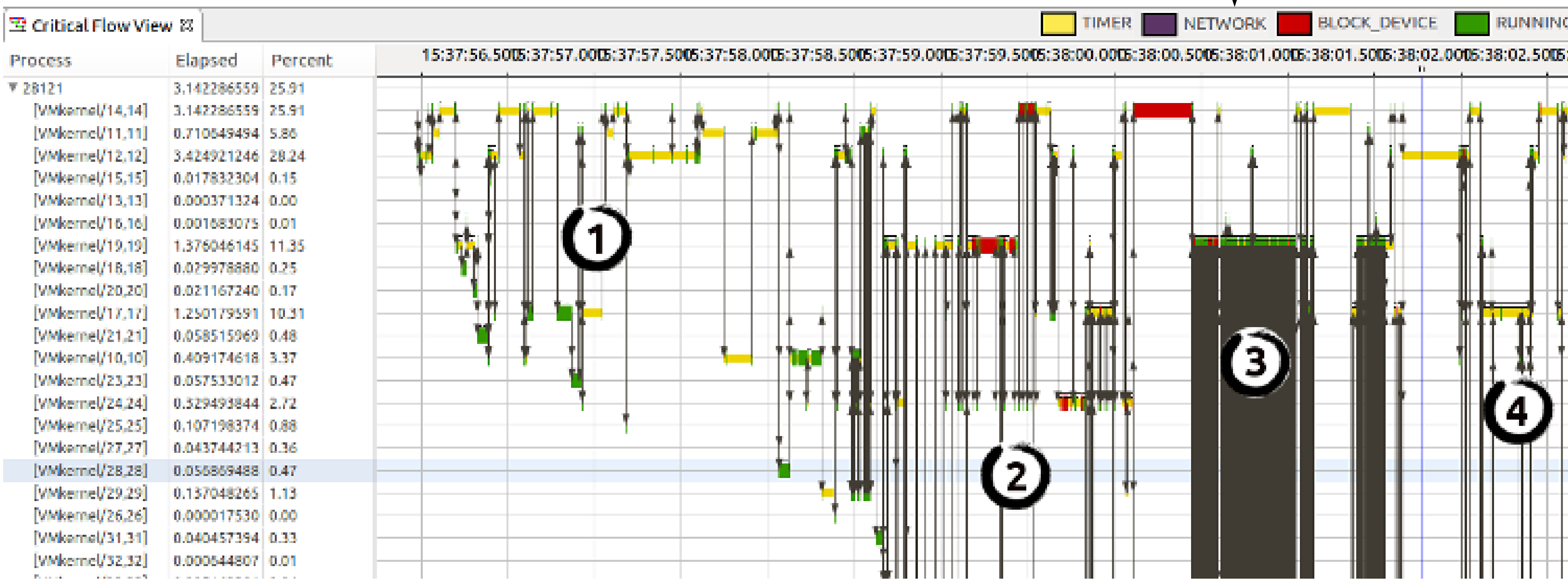


Investigation

Critical Path Analysis

Linux Advance Packaging Tool

What is going on here ?

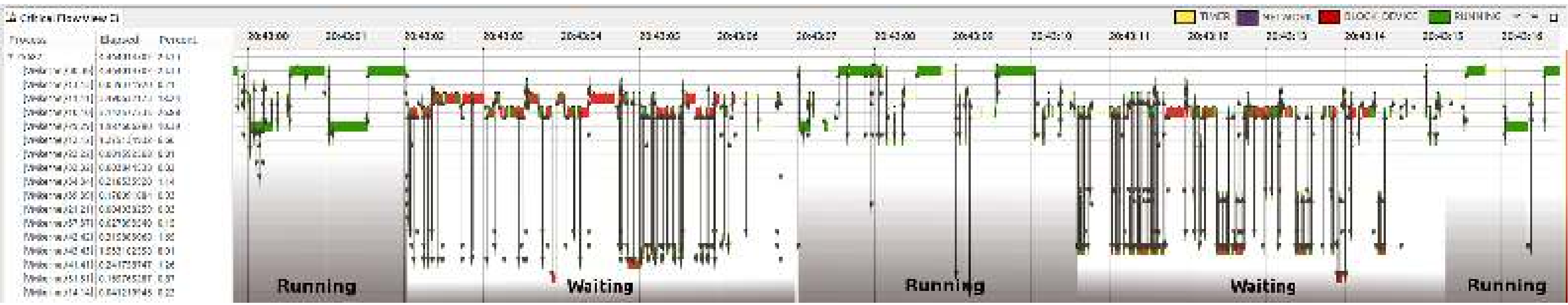



```
1) apt-get downloads and reads cached packages
2) apt-get installs the packages along with downloaded dependencies
3) The installation of man-pages
```

Investigation

Critical Path Analysis

Undesirable parallelism



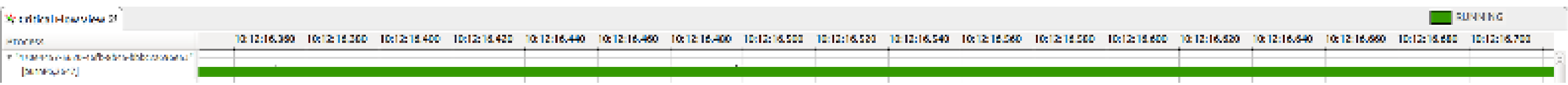
waits for disk

waits for another process

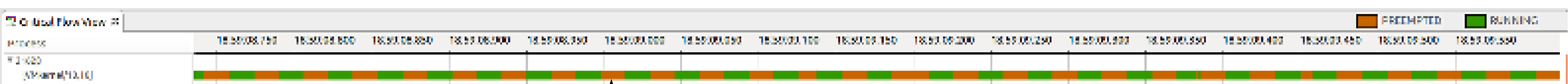
Investigation

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

Benchmark	Baseline	CPA	HEC	Overhead	
				CPA	HEC
File I/O (ms)	450.92	480.38	451.08	6.13%	0.03%
Memory (ms)	612.27	615.23	614.66	4.81%	0.01%
CPU (ms)	324.92	337.26	325.91	3.65%	0.30%

Investigation

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

Investigation

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

[2] https://github.com/nemati

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 overcomittment, 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/