VirtFlow: Guest Independent Execution Flow Analysis Across Virtualized Environment

Hani Nemati

Dec 12, 2016

Polytechnique Montréal

Laboratoire **DORSAL**

Agenda

Motivation

- Why analyzing nested VM is important ?

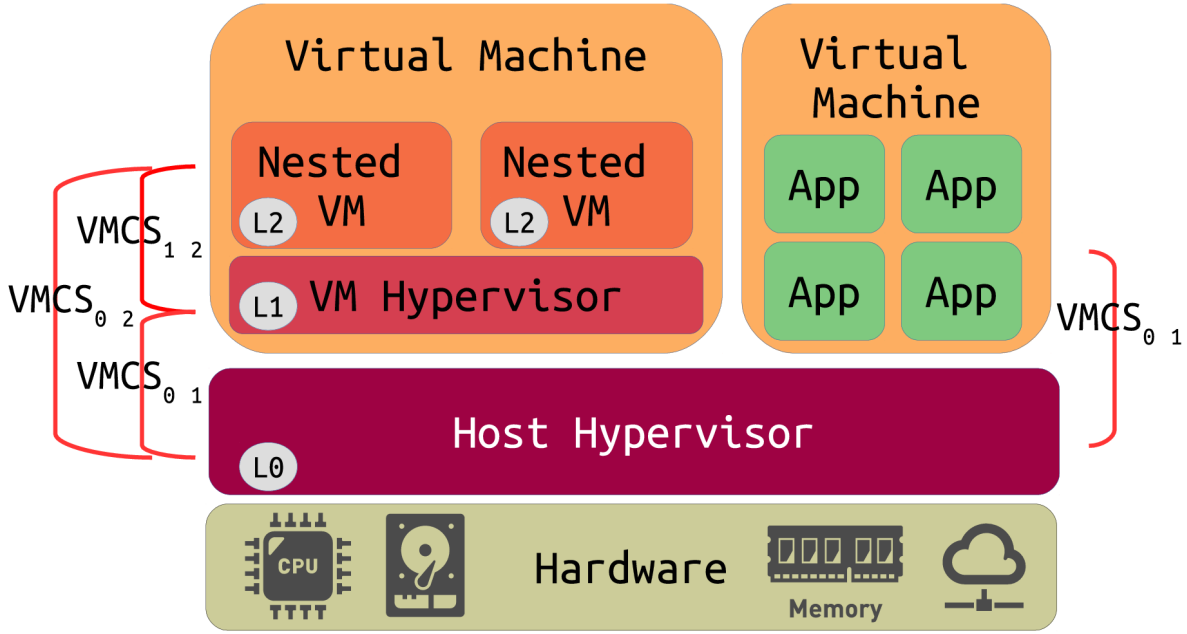
Investigations

- Nested Virtual Machine Process State Transition
- Any-Level VM Detection Algorithm
- Nested VM State Detection (NSD) Algorithm
- Performance Analysis and Evaluation
  - Execution Flow Analysis
  - CPU Cap and CPU Overcommitment Problem
  - Overhead of Virtualization Layer for different Types of Workload
  - Overhead of Virtualization for different Operating System

Conclusion and in-progress

Investigations

What is Nested Virtual Machine ?



- Root (L0) - Code of Host Hypervisor
- Non-Root (L1) - Code of VM Hypervisor
- Non-Root (L2) - Code of Last Level VM

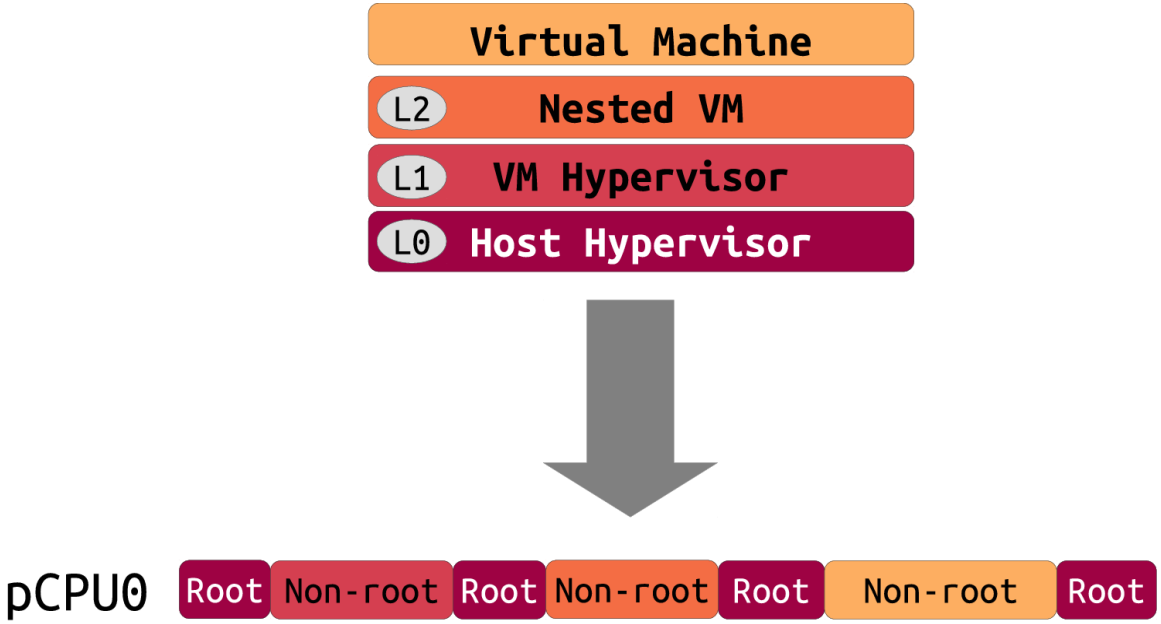
Motivation

Why analyzing Nested VM is important?

- Compatibility issues (e.g., Windows 7)
- Security Concern (e.g., McAfee)
- Software Scaling (e.g., SaaS)
- Continuous Integration (CI)

Investigations

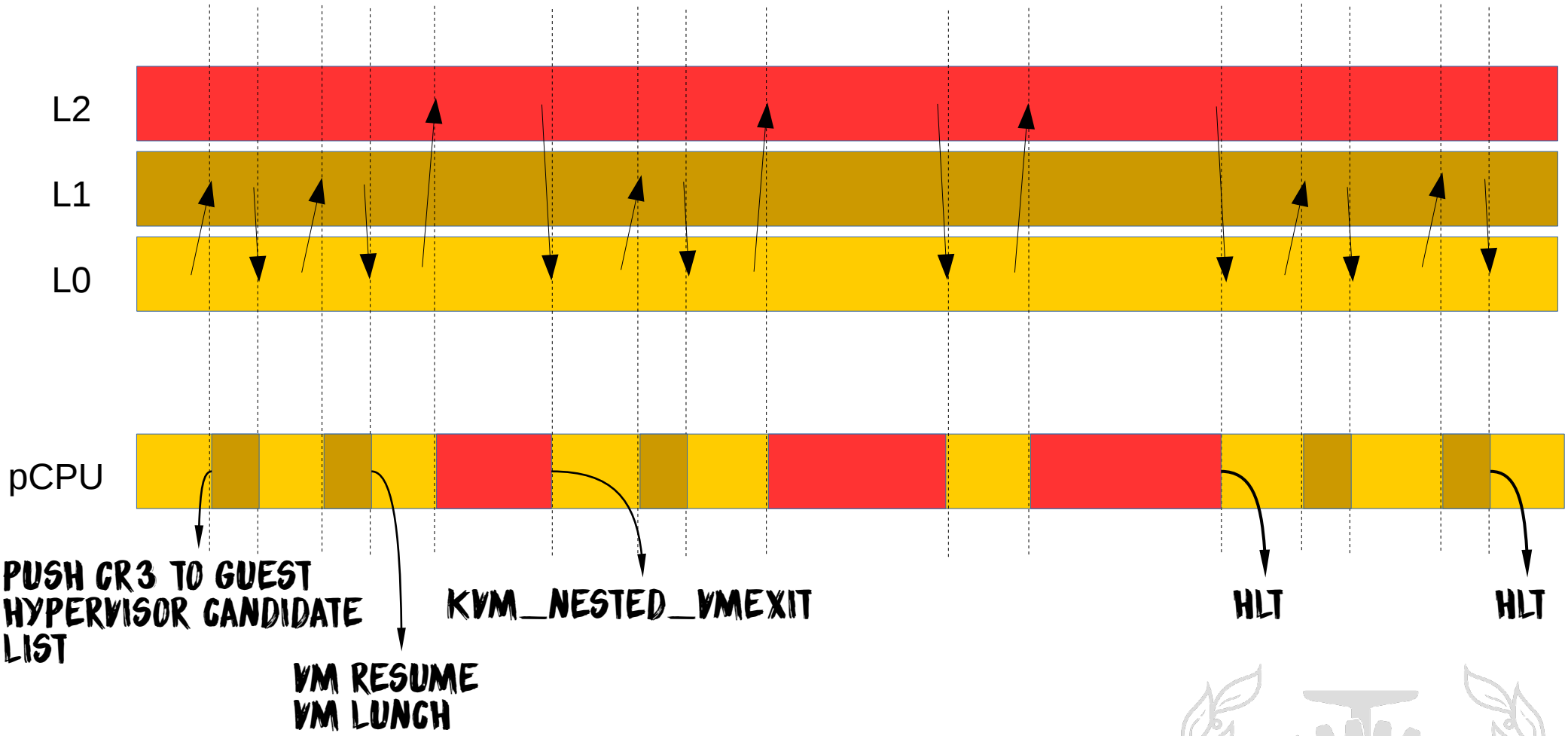
What is our goal in this project?????



To find Execution Flow of Virtual Machine

Investigations

How does it work internally? (Host Tracing)



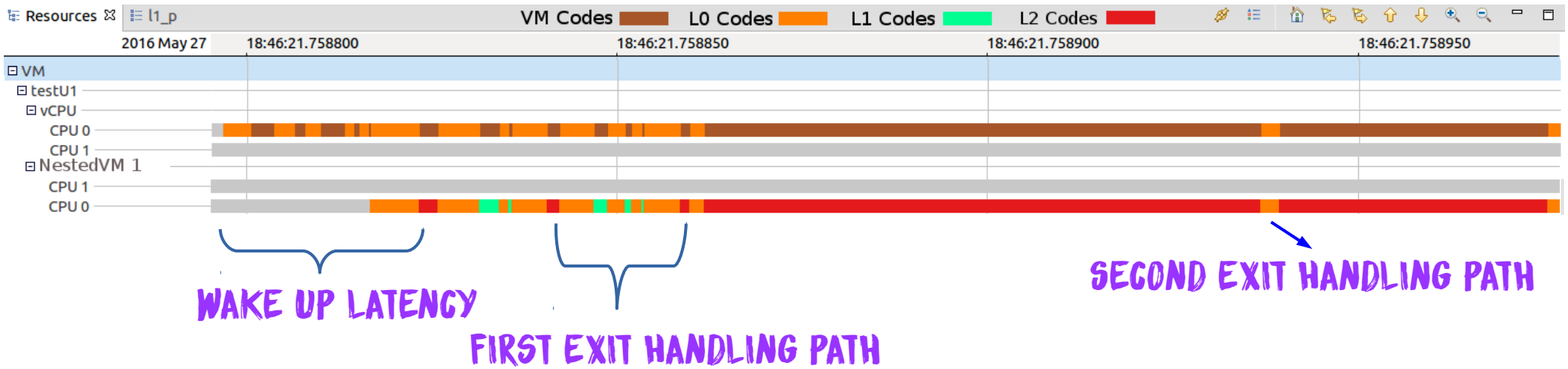
Investigations

What if we only have access to L1 ? (VM (L1) Tracing)

Investigations

Level Detection Result:

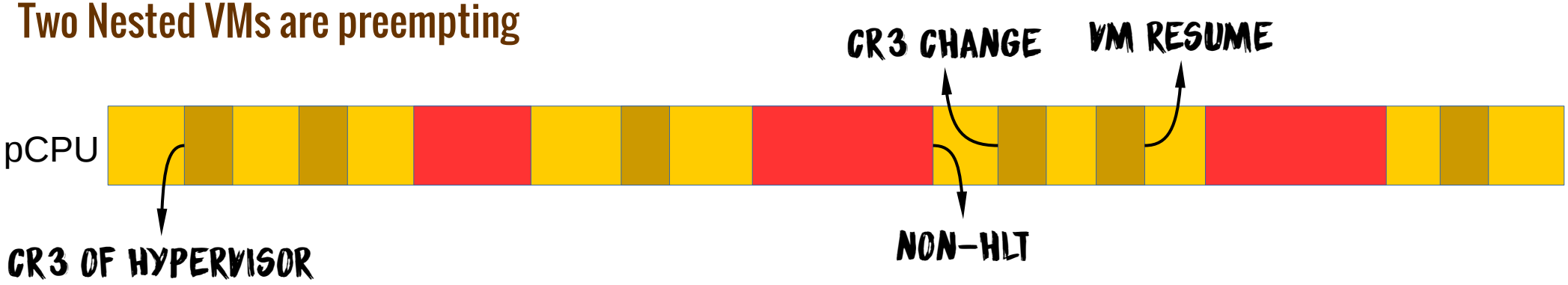
- Calculating first 10000 prime numbers:



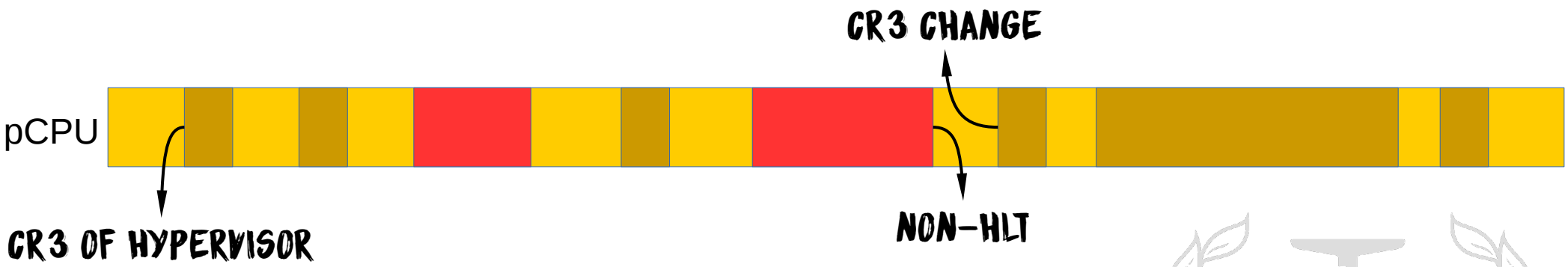
Investigations

Preemption in different levels:

Two Nested VMs are preempting

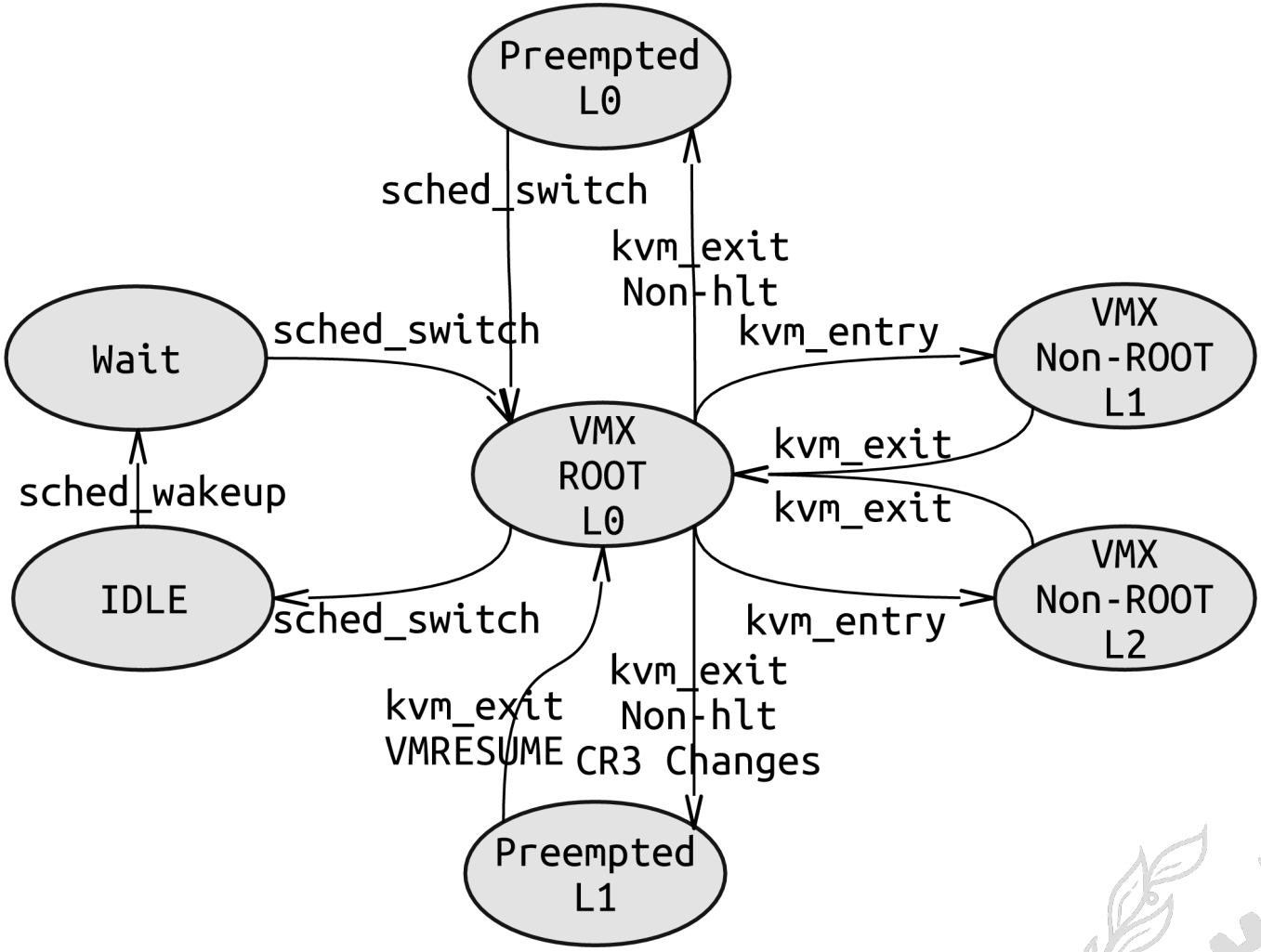


Nested VM is being preempted by a process inside L1



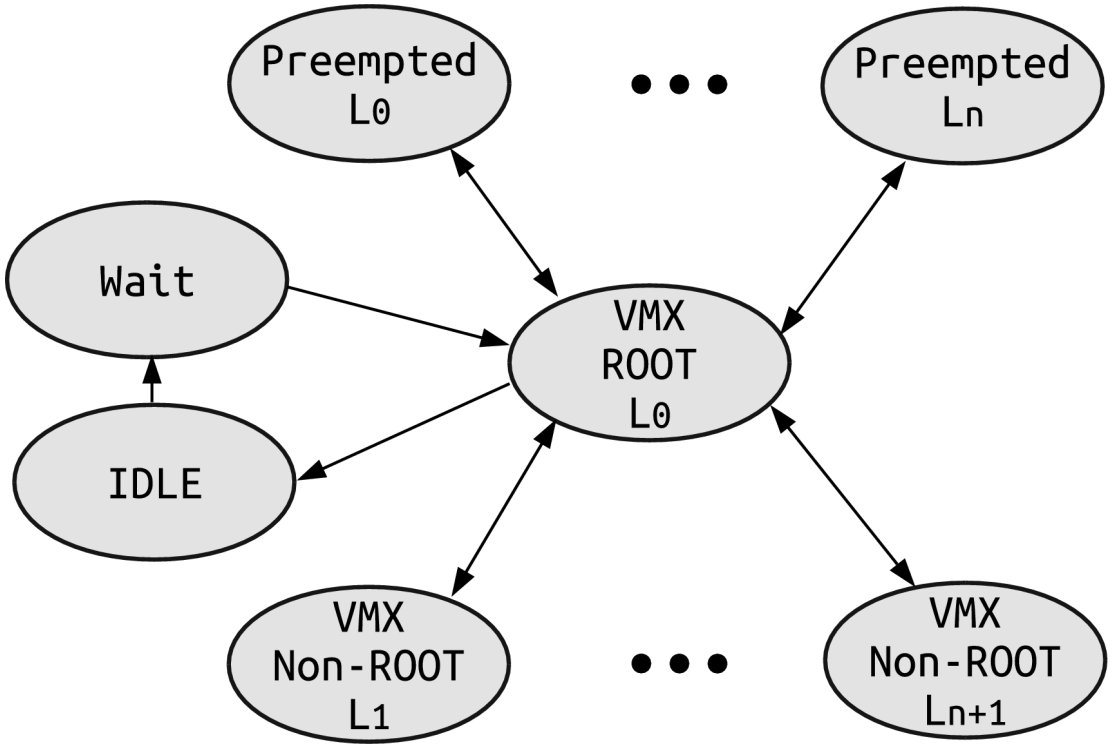
Investigations

Nested Virtual Machine Process State Transition



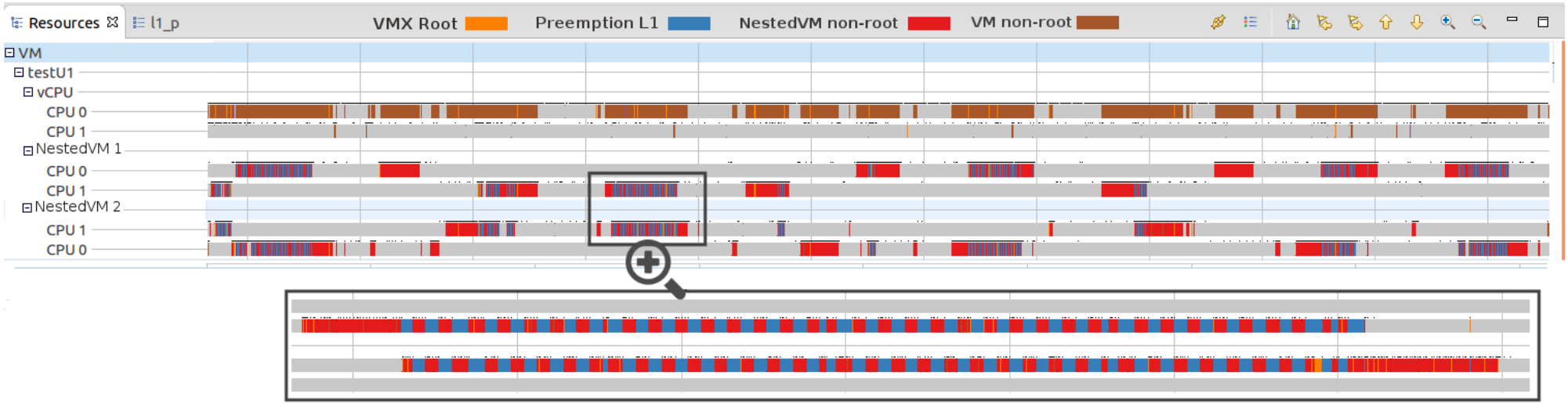
Investigations

N-Level Nested Virtual Machine Process State Transition



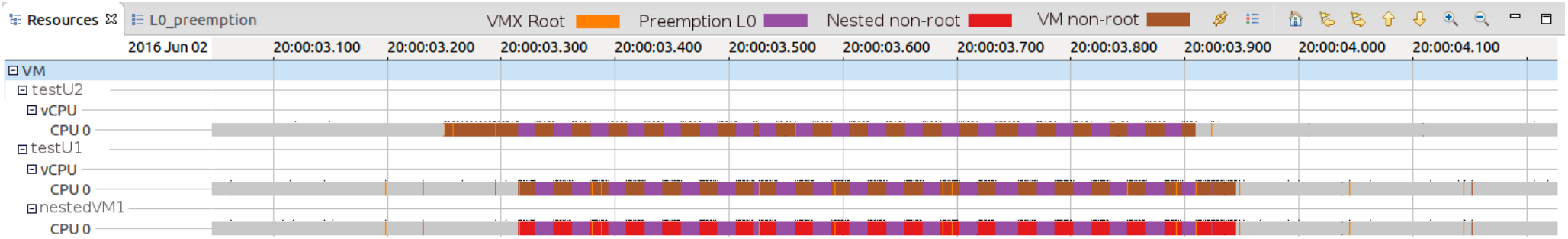
Investigations

Two Nested VMs are preempting each other inside a VM



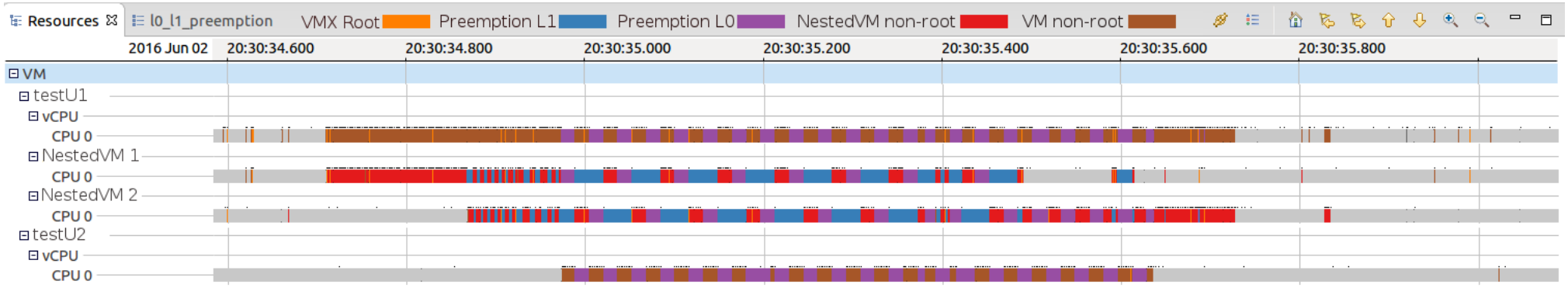
Investigations

A VM preempts a Nested VM



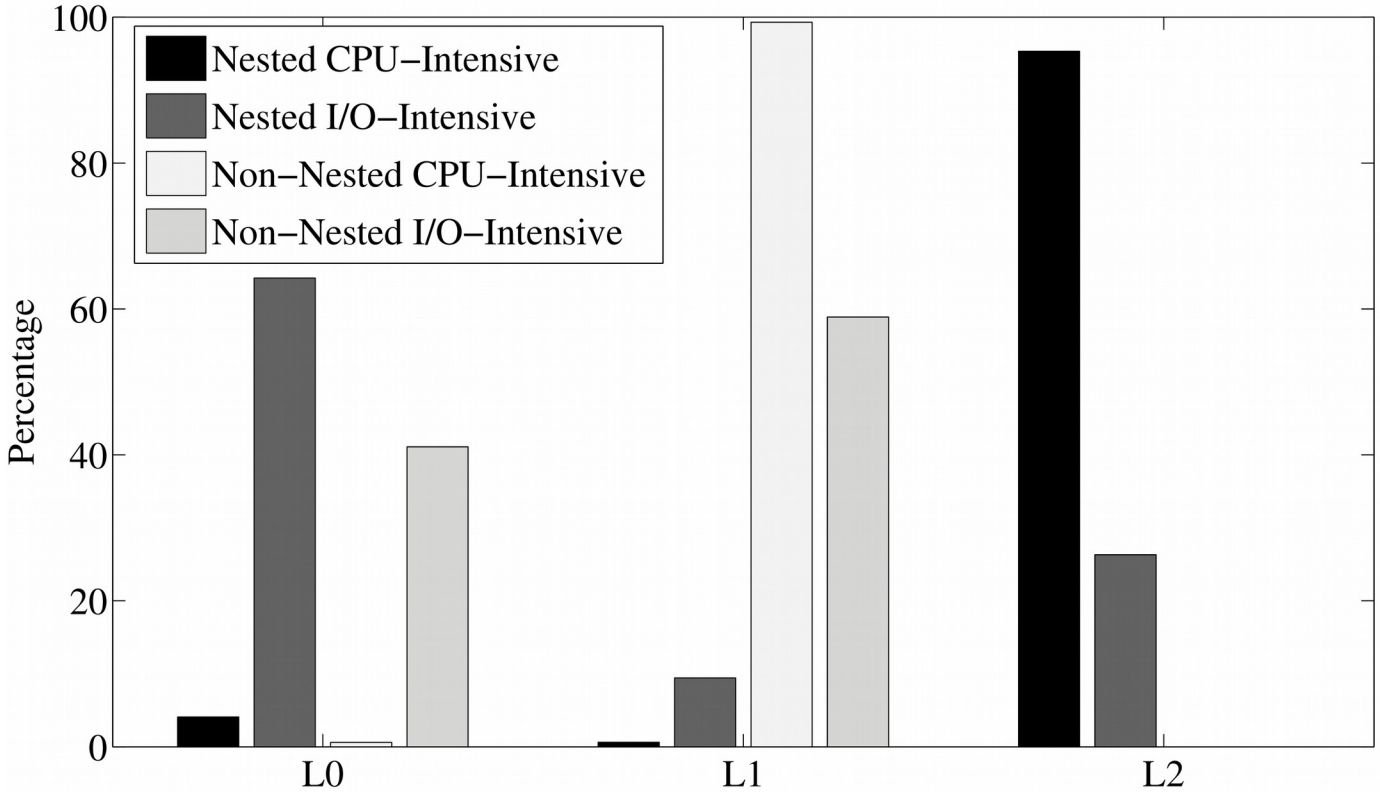
Investigations

Two Nested VMs and One VM are preempting each other



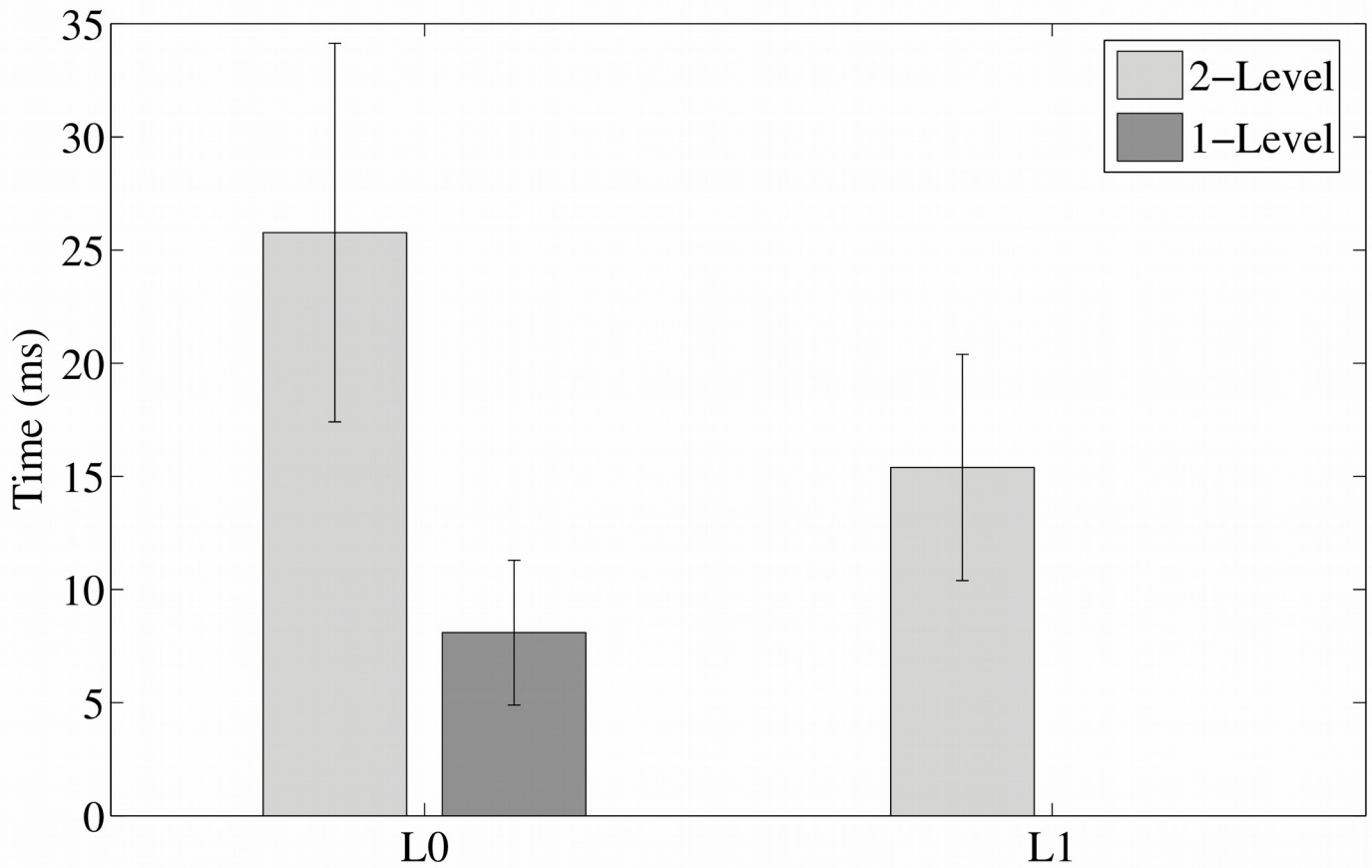
Investigations

Overhead of Virtualization layers for different types of Workload



Investigations

Average wake up latency for Nested VM and Normal VM



Investigations

Execution time for Fibo program on different level of virtualization and different OS

Experiment	T_{L_0} *(ms)*	T_{L_1} *(ms)*	T_{L_2} *(ms)*	U *(%)*	O *(ms)*
Nested-Linux	18.779	4.728	1539.45	98.5	23.507
Nested-Windows	439.864	283.582	1653.62	69.5	723.446
VM-Linux	5.623	1512.18	-	99.6	5.623
VM-Windows	216.362	1569.1	-	88.1	216.362
Host	1508.75	-	-	1	-

Investigations

Demo

Investigations

What do you need to test this project?

- Access to **Host** only
- Run **LTTng** on Host with my new added tracepoint (vcpu_enter_guest)
- Clone **TraceCompass** from my github (virtFlow)
  - https://github.com/Nemati
- Open Resource View of TraceCompass

One More Thing …

VM PT

Intel Processor Trace

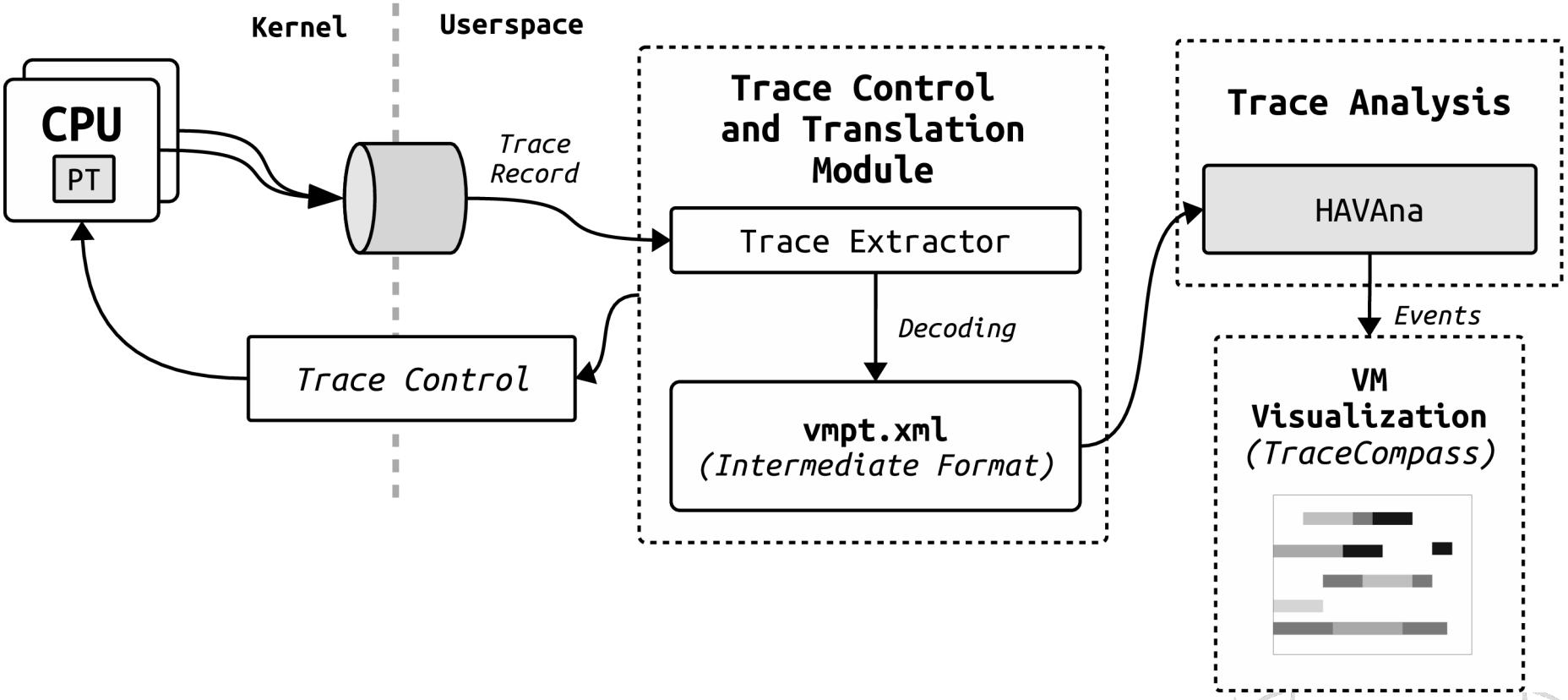
Intel PT

VM PT Packets

- **PIP** (Paging Information Packet) : CR3 change
  - NR bit (VM Entry, VM Exit)
- **VMCS** (Unique base pointer per vCPU): Successful VMPTRLD
- **Timing**
  - TSC : Time Stamp Counter
  - MTC : Mini Time Counter

Intel PT

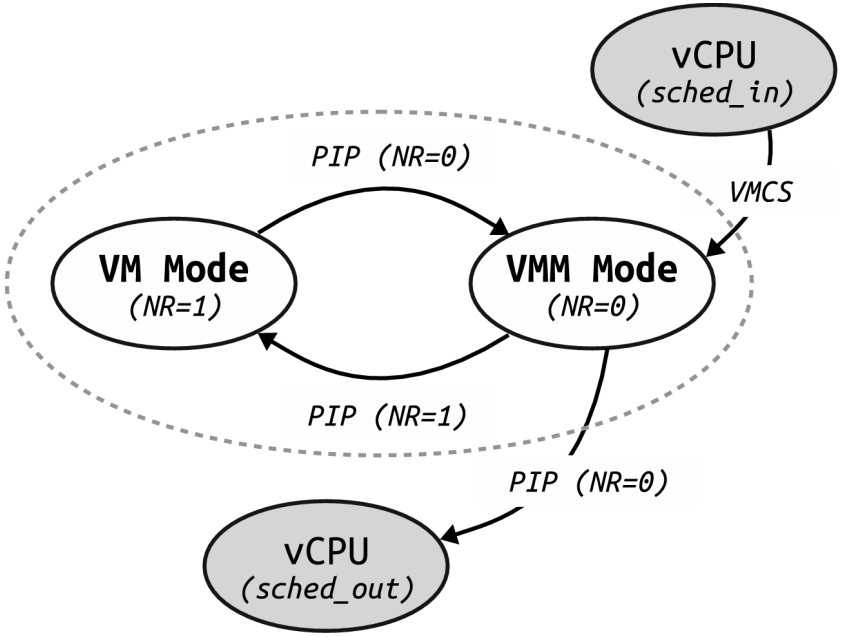
Using Intel PT



HAVAna Algorithm and State Machine

Algorithm 1 HAVAna Algorithm

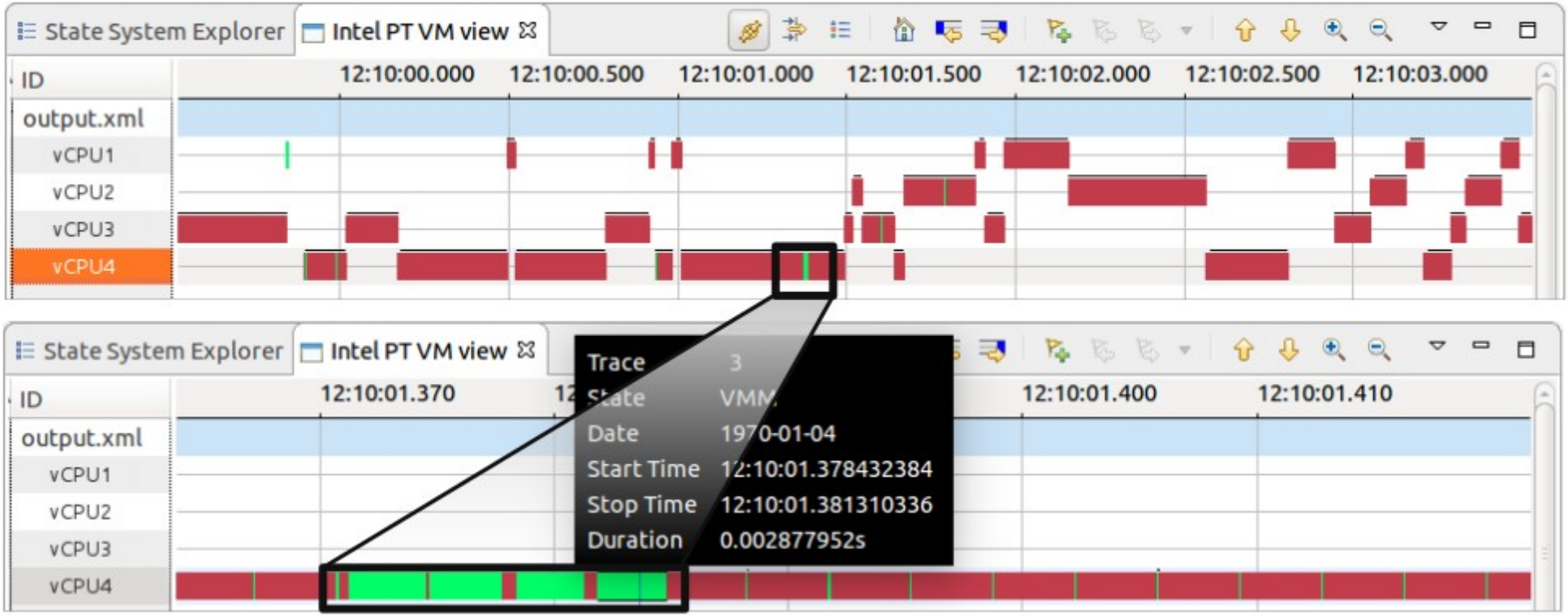
```
procedure HAVAna(Input: Event Packets ($P_e[i]$) from IF Output: Updated SHT)
    $SE[i] = parseXML(P_e[i])$
    if ($SE[i].name == VMCS$) then
        Modify Status attribute of SE[i].base as $VMM$
    else if ($SE[i].name == PIP$) then
        if ($SE[i].NR == 1$) then
            Query Status attribute of current running base
            Modify Status attribute as $VM$
            Modify VMM Status as $IDLE$
            Modify Status attribute of $SE[i].cr3$ as $VM$
        else if (Query Status attribute of $SE[i].base == VMM$) then
            Query Status attribute of current running base
            Modify $base$ Status as $IDLE$
            Modify VM Status as $IDLE$
            Modify VMM Status as $IDLE$
        else
            Query Status attribute of current running base
            Modify $base$ Status as $VMM$
            Modify Status attribute of $SE[i].cr3$ as $VMM$
            Modify VM Status as $IDLE$
        end if
    end if
end procedure
```



Intel PT

VM Analysis

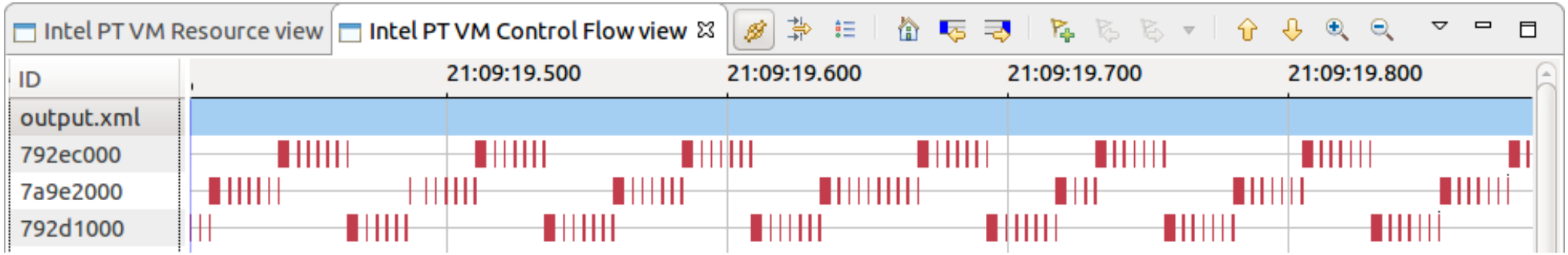
- 4 vCPUs are pinned to one pCPU
- 4 Threaded application to calculate Prime numbers



Intel PT

VM Analysis

- 4 vCPUs are pinned to one pCPU
- MD5 hashing with 3 workers



Conclusion and in-progress

Inferences

- Any-Level VM Detection Algorithm (ADA)
- Nested VM State Detection (NSD) algorithm
- Calculate overhead of virtualization layers for different types of workload
- Analysis wake up latency for VM and Nested VM
- Analysis behavior of KVM for different OS

Going Further

- Wait Analyzing of process inside VM and Nested VM

Outcome of this project

[1] Hani Nemati, Michel R. Dagenais, " Virtual CPU State Detection and Execution Flow Analysis by Host Tracing. " BDCloud-SocialCom-SustainCom 2016: 7-14

[2]Suchakrapani Datt Sharma, Hani Nemati, Genevieve Bastien, Michel Dagenais "Low Overhead Hardware-Assisted Virtual Machine Analysis and Profiling", The Fifth IEEE International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA 2016)

[3] Hani Nemati, Suchakrapani Datt Sharma , Michel R. Dagenais, "Fine-grained Nested Virtual Machine Performance Analysis Through First Level Hypervisor Tracing", submitted to IEEE/ACM CCGrid 2016

[4] Hani Nemati, Michel Dagenais, "virtFlow: Guest Independent Execution Flow Analysis Across Virtualized Environment" , in preparation

Questions?

Hani.nemati@polymtl.ca

https://github.com/Nemati