Monitoring and Analyzing Virtual Machines – Resource Overcommitment Detection and Virtual Machine Classification

Hani Nemati

May 5, 2015

Polytechnique Montréal

Laboratoire **DORSAL**

Agenda

Motivation

- Why detecting resource overcommitment is important ?
- Why we need a virtual machine classification model?

Investigations

- Detecting Resource overcommitment for Infrastructure providers
  - Virtual CPU State Detection
    - CPU Overcommitment
    - Memory Overcommitment
- Virtual Machine Dissection
- Virtual Machine Classification

Conclusion and in-progress

References

Motivation

Why detecting resource overcommitment is important?

- Resource Overcommitment lets the administrator allocate more resources to virtual machines than the physical host has available.
- Identifying Over-committed and Under-committed hosts
- Impact of Resource Overcommitment :
  - Cloud User:
    - Latency in response time of programs inside VMs
  - Infrastructure Provider:
    - Increase resource utilization
    - Maximizing profit

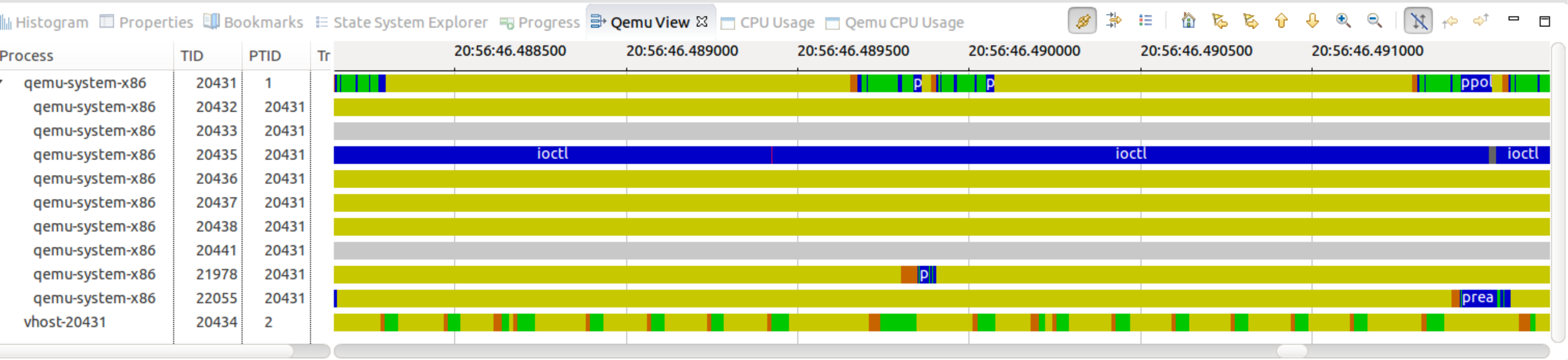
Motivation

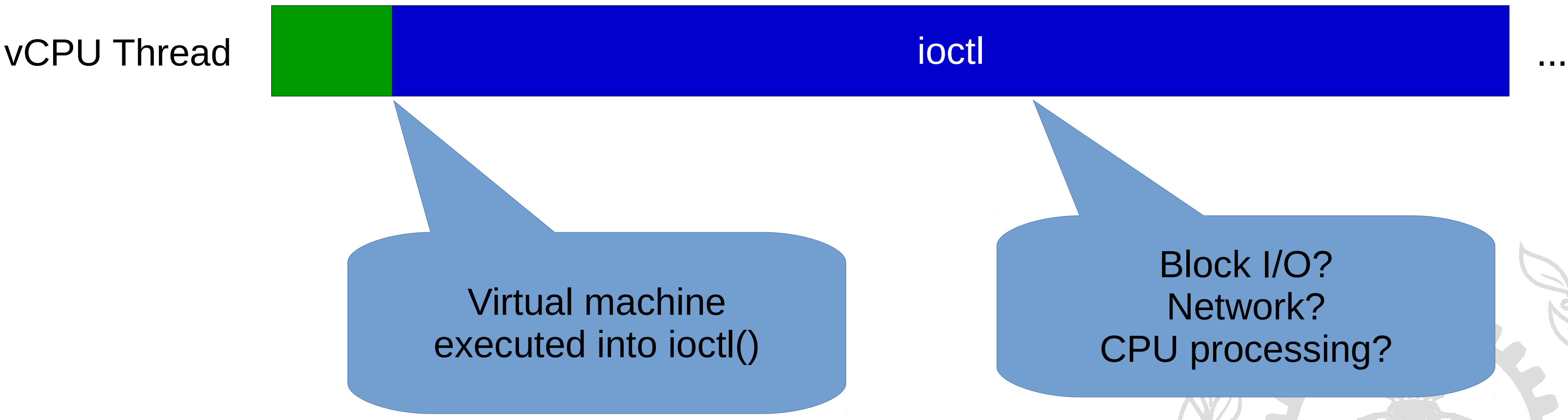
Why we need a virtual machine classification model?

- **Classifying virtual machines** based on different resource consuming patterns [1][3]
- **Categorizing virtual machines into:**
    - CPU-intensive
    - Memory-intensive
    - I/O-intensive
    - Network-intensive
    - Compound
- **Advantages**:
    - Better capacity planning
    - Increase infrastructure revenue by utilizing host resources
    - Better decision for migration of VMs

Investigations

Qemu threads in Control Flow View

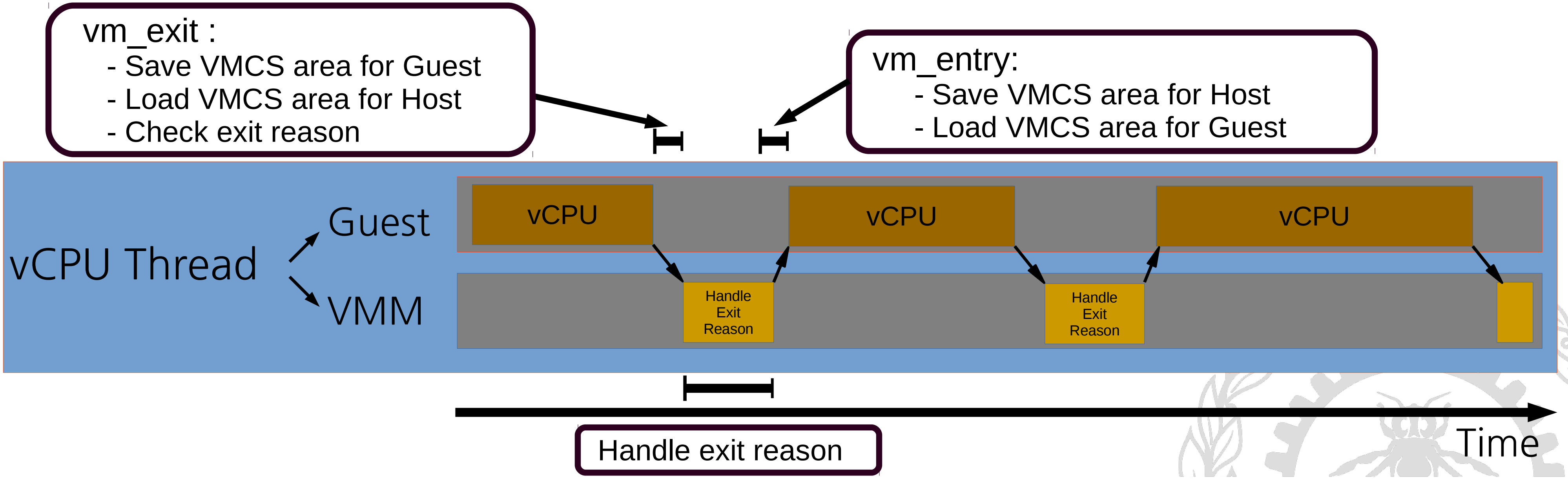




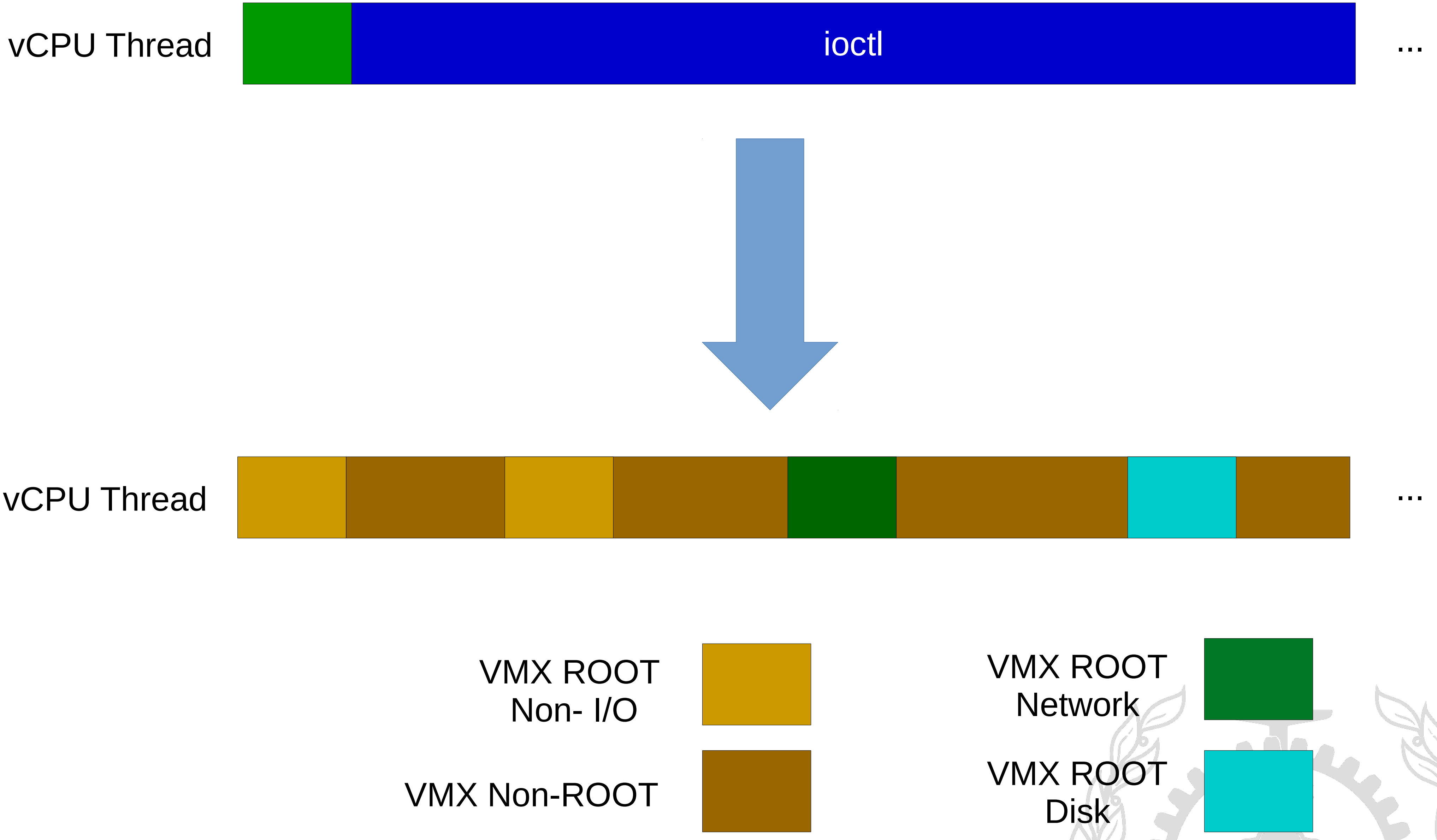
Investigations

CPU Virtualization with intel-VT-x:

- VMX transition [5]
  - Between Guest and Virtual Machine Manager (VMM)
    - vm_entry , vm_exit
    - Virtual Machine Control Structure (VMCS)

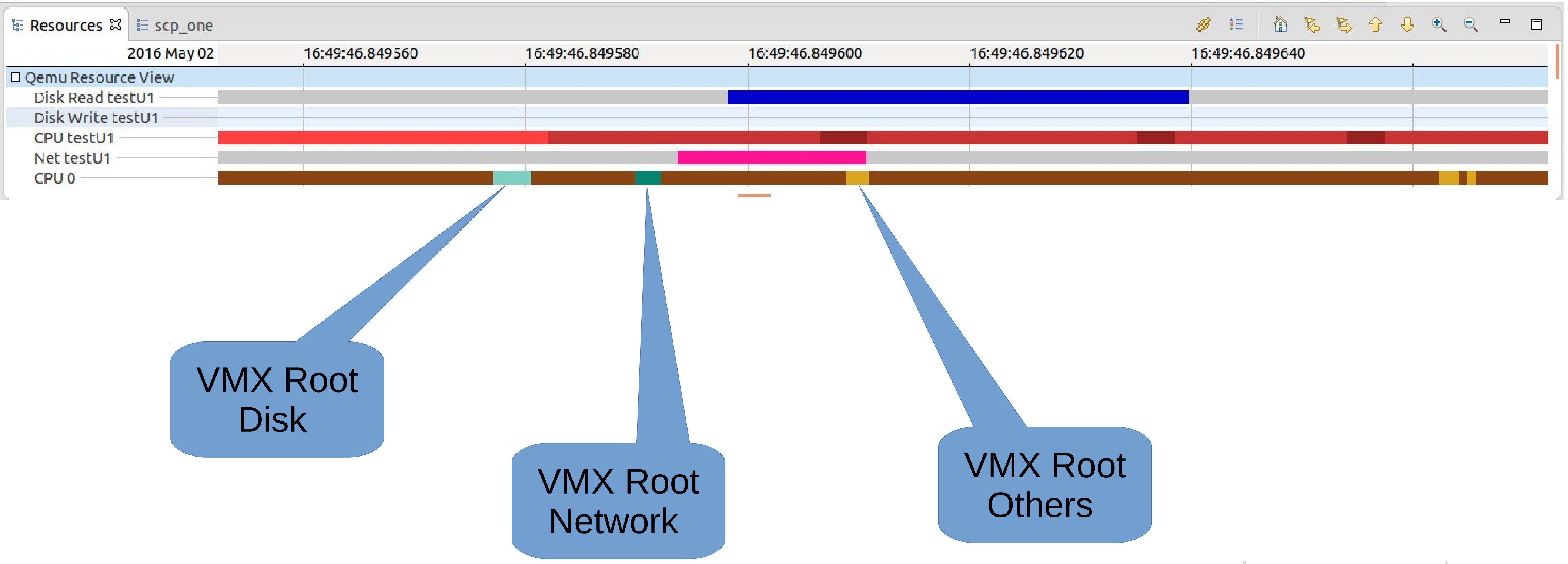


Investigations



Investigations

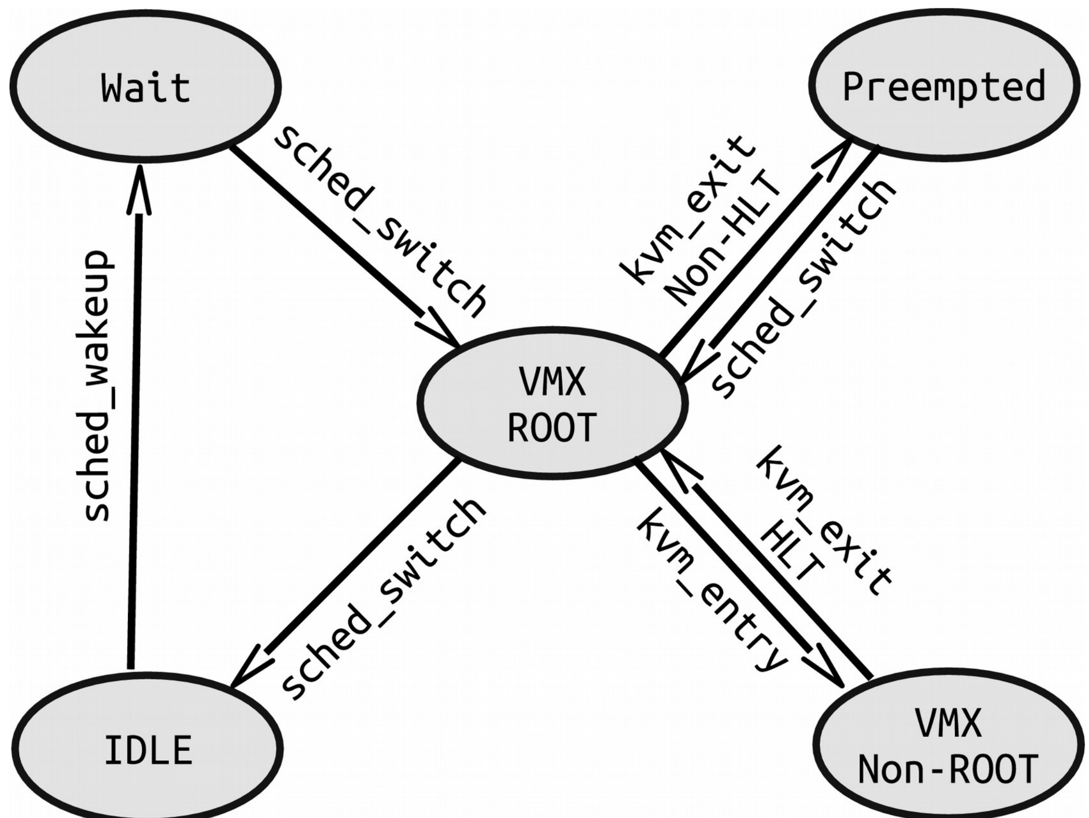
Resource View for Virtual Machine



Investigations

vCPU State Detection

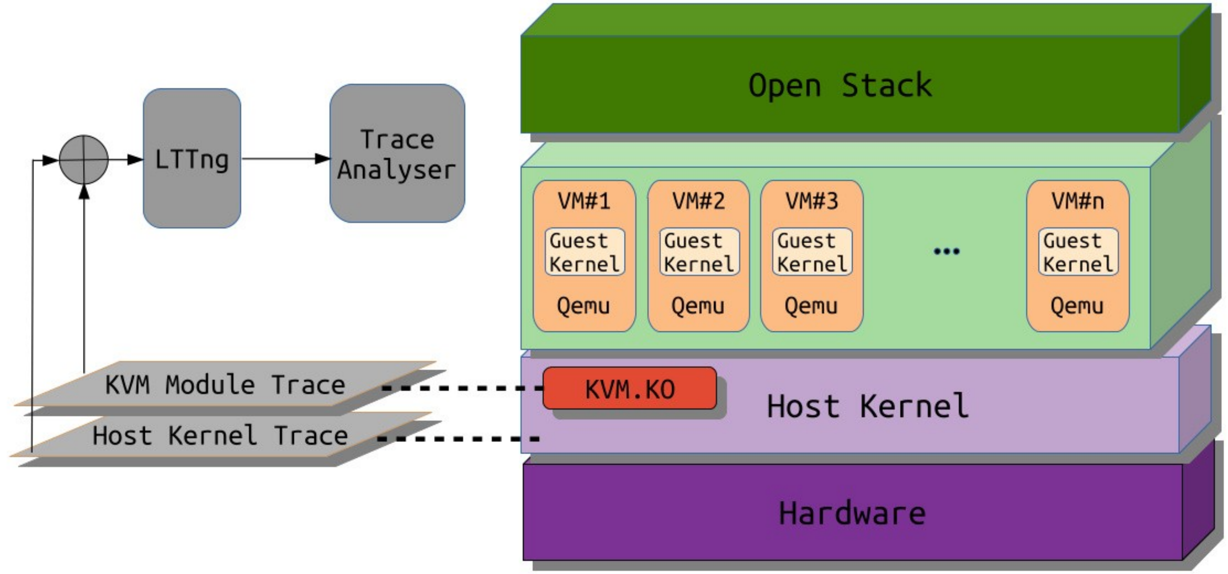
- **Virtual CPU State Detection:**
  - VMX Root and Non-Root State
  - IDLE state
  - Wait state
  - Preempted State
- **VMX Root**
  - Disk I/O
  - Network
  - Memory



Investigations

Our Architecture and tracepoints used for vCPU state detection:

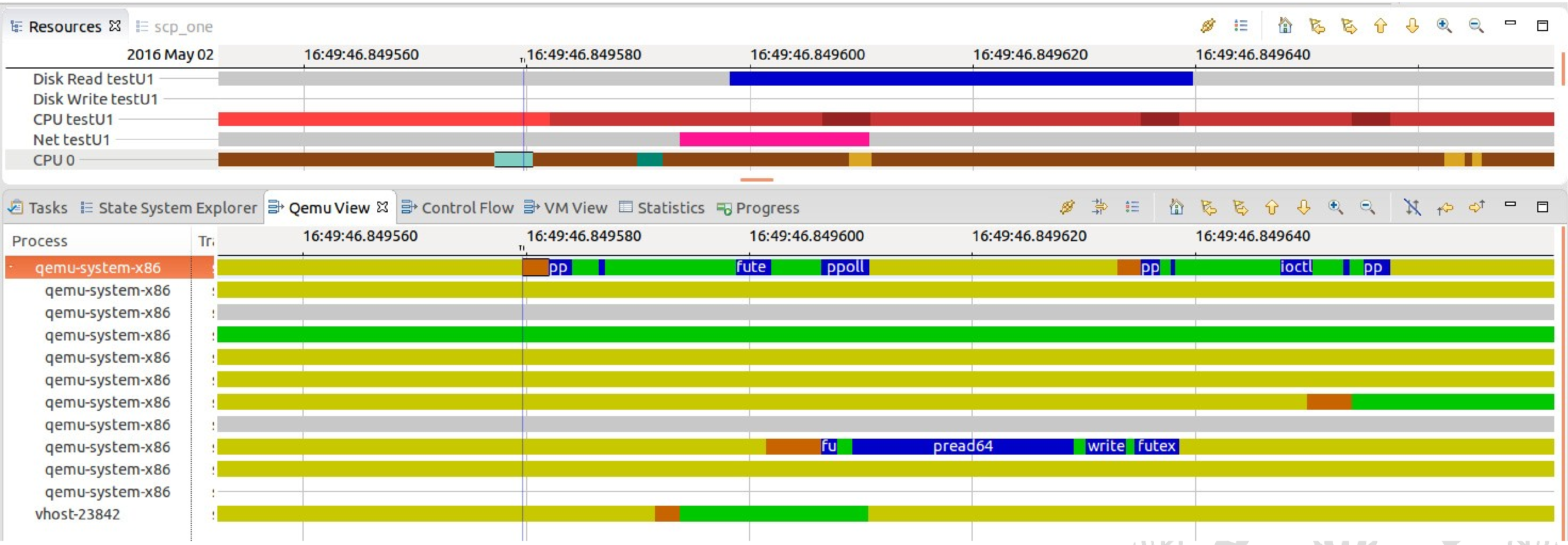
- sched_wakeup: To detect wait state and VMX Root network and Disk
- kvm_exit: To detect VMX Root state
- kvm_entry: To detect VMX Non-root state
- sched_switch: To detect vCPU thread, Preemption state, and IDLE state



Investigations

VM Resource view and Thread view

- During VMX Root with exit reason 30 (I/O Request)
  - Wake up a thread from qemu thread pool ==> Disk request
  - Wake up a thread from vhost threads ==> Netwok request



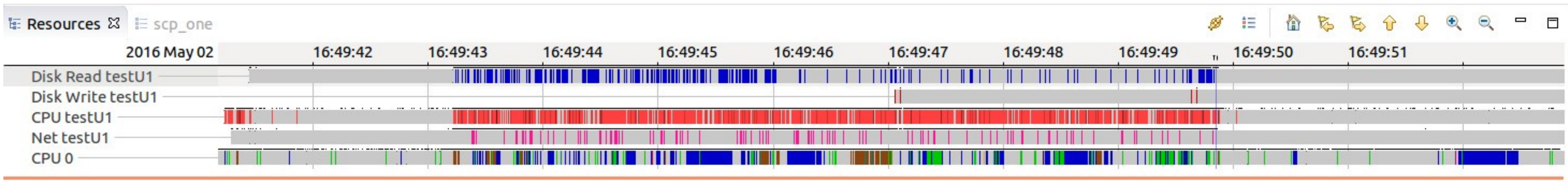
Investigations

For more information about resource usage:

- Disk I/O:
  - Using Qemu trace points
    - qemu:thread_pool_submit, qemu:thread_pool_complete, qemu:bdrv_co_io_em
- Network:
  - Using Host Kernel Trace
    - net_if_rx, net_dev_xmit, sched_switch and sched_wakup for vhost-$(VM-main-thread)
- Memory:
  - Written Module that investigate accessed paged for a period of time

Investigations

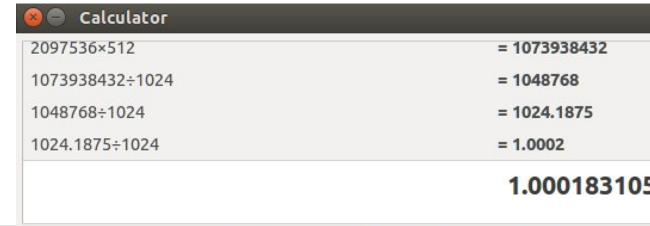
VM statistics for vCPU

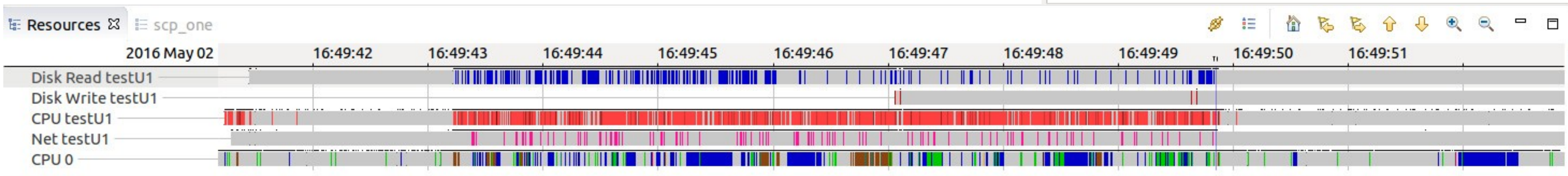


CPUQemu	3648			16:49:41.	16:49:5	CPUQemu
23842	3649			16:49:41.	16:49:5	CPUQemu/23842
vmName	3650	testU1	String	16:49:41.	16:49:5	CPUQemu/23842/vmName
ValueCPU	3809	0	Int	16:49:49.	16:49:4	CPUQemu/23842/ValueCPU
STATUS	3810	0	Int	16:49:49.	16:49:4	CPUQemu/23842/STATUS
vCPU	3833			16:49:41.	16:49:5	CPUQemu/23842/vCPU
2	3834	0	Int	16:49:49.	16:49:4	CPUQemu/23842/vCPU/2
3	3852	0	Int	16:49:49.	16:49:4	CPUQemu/23842/vCPU/3
0	3934	0	Int	16:49:49.	16:49:4	CPUQemu/23842/vCPU/0
1	3949	0	Int	16:49:49.	16:49:4	CPUQemu/23842/vCPU/1
statistics	3840			16:49:41.	16:49:5	CPUQemu/23842/statistics
32	3841	21131	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/32
12	3847	11182	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/12
30	3938	72056	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/30
latency	3939	262512521	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/30/latency
7	3945	3426	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/7
1	3954	80853	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/1
40	3958	92	Long	16:49:49.	16:49:4	CPUQemu/23842/statistics/40
10	3973	450	Long	16:49:41.	16:49:4	CPUQemu/23842/statistics/10

Investigations

VM statistics for Disk and Network

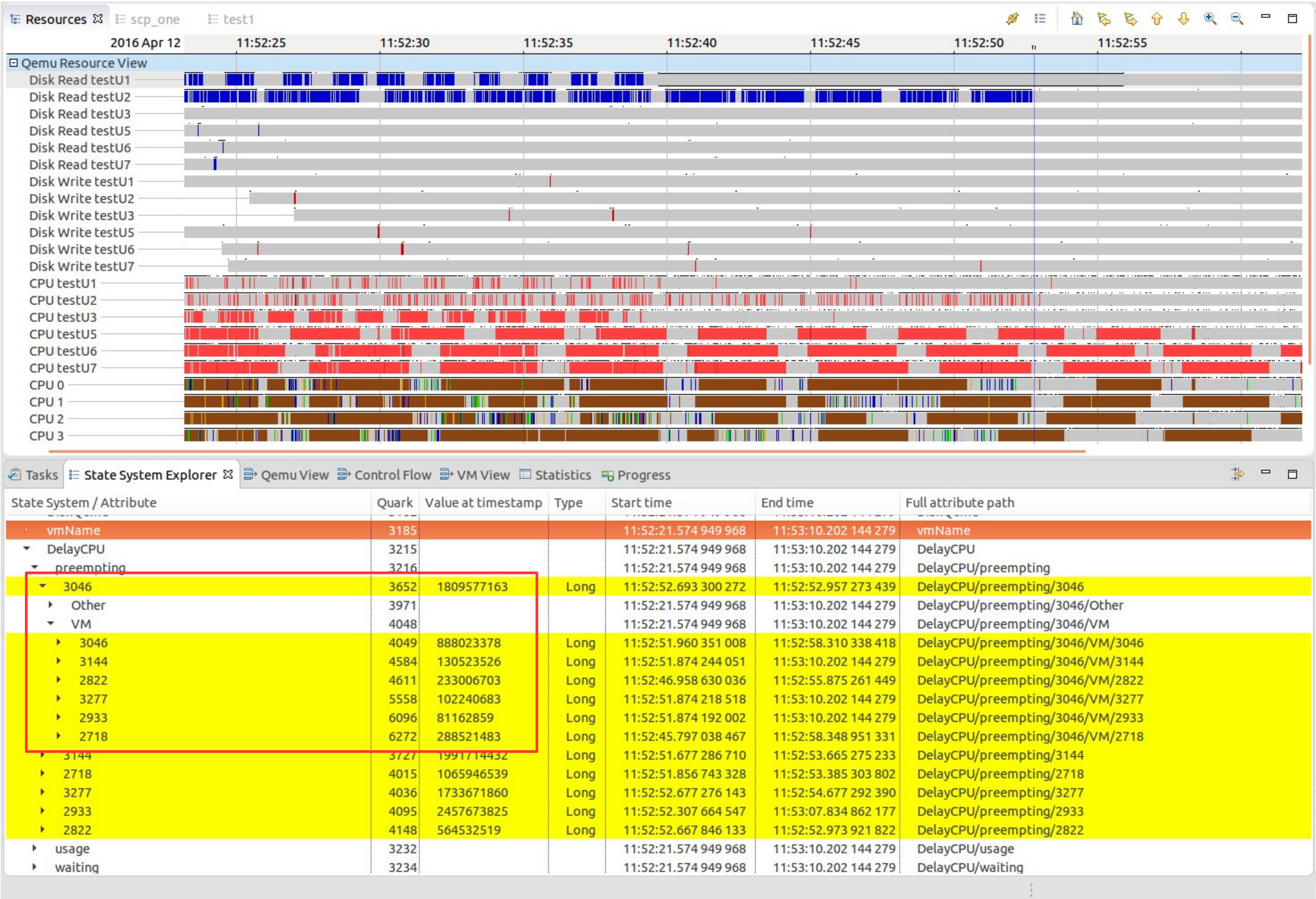




DiskQemu	3651			16:49:41.	16:49:5:	DiskQemu
23842	3652			16:49:41.	16:49:5:	DiskQemu/23842
vmName	3653	testU1	String	16:49:41.	16:49:5:	DiskQemu/23842/vmName
read	4055			16:49:41.	16:49:5:	DiskQemu/23842/read
numberOfSubmited	4056	0	Int	16:49:49.	16:49:4!	DiskQemu/23842/read/numberOfSubmited
transfer	4057	0	Int	16:49:49.	16:49:4!	DiskQemu/23842/read/transfer
STATUS	4058	0	Int	16:49:49.	16:49:4!	DiskQemu/23842/read/STATUS
latency	4067	2654708567	Long	16:49:49.	16:49:4!	DiskQemu/23842/read/latency
totalTransfer	4068	2097536	Int	16:49:49.	16:49:4!	DiskQemu/23842/read/totalTransfer
write	5329			16:49:41.	16:49:5:	DiskQemu/23842/write
numberOfSubmited	5330	0	Int	16:49:49.	16:49:5:	DiskQemu/23842/write/numberOfSubmited
transfer	5331	0	Int	16:49:49.	16:49:5:	DiskQemu/23842/write/transfer
STATUS	5332	0	Int	16:49:49.	16:49:5:	DiskQemu/23842/write/STATUS
latency	5340	61718701	Long	16:49:49.	16:49:5:	DiskQemu/23842/write/latency
totalTransfer	5341	224	Int	16:49:49.	16:49:5:	DiskQemu/23842/write/totalTransfer

NetQemu	3922			16:49:41.	16:49:5:	NetQemu
23870	3923			16:49:41.	16:49:5:	NetQemu/23870
STATUS	3924	0	Int	16:49:49.	16:49:4!	NetQemu/23870/STATUS
Netif	3925	0	Int	16:49:49.	16:49:4!	NetQemu/23870/Netif
Netdev	3926	0	Int	16:49:49.	16:49:4!	NetQemu/23870/Netdev
tNetdev	3932	2610162	Long	16:49:49.	16:49:4!	NetQemu/23870/tNetdev
tNetif	3933	1078579542	Long	16:49:49.	16:49:4!	NetQemu/23870/tNetif

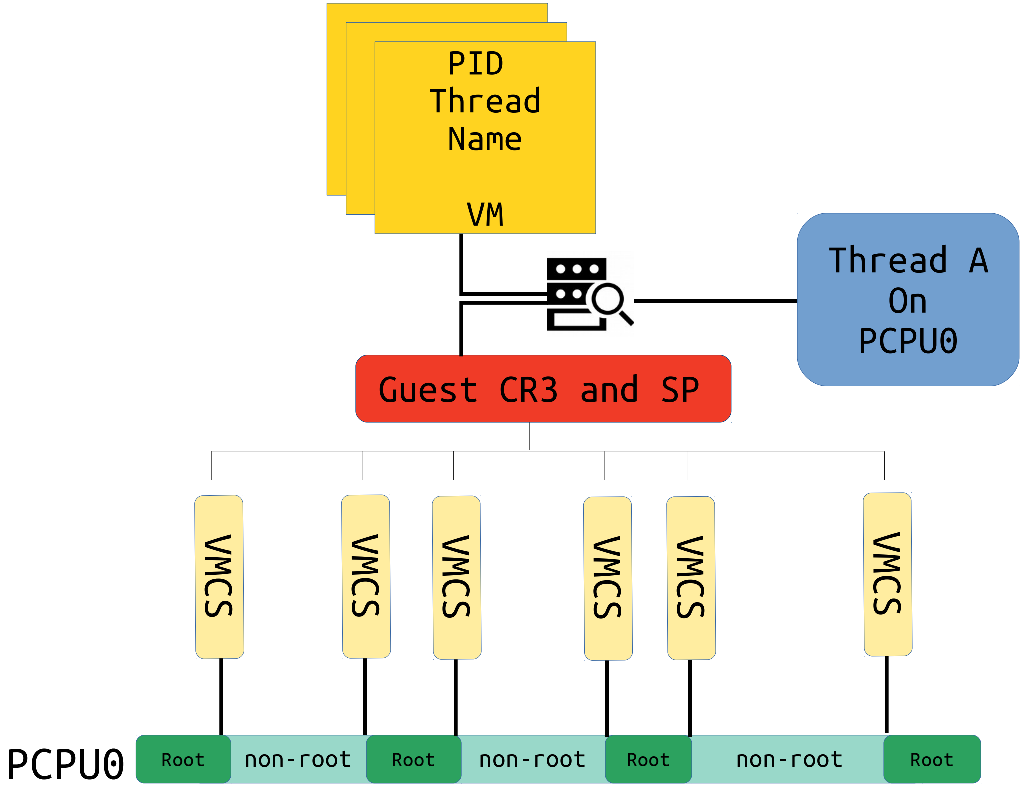
Investigations VM statistics for vCPU



Investigations

Virtual Machine Dissection [4]

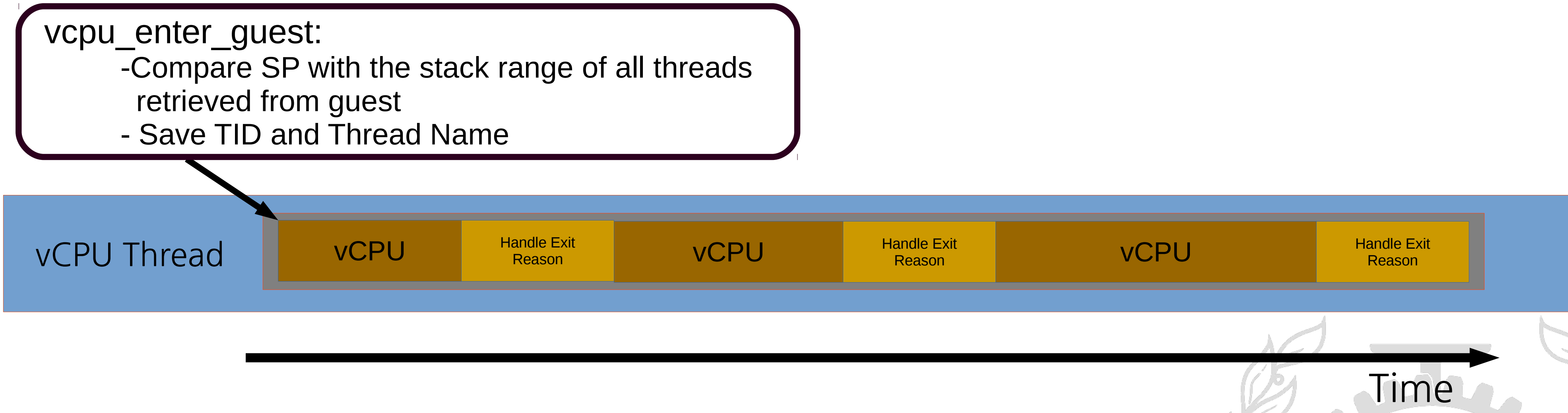
- **Process Identifier (PID)** and **Process Name** inside the VMs is not accessible from Host tracing.
- **CR3** points to the page directory of process inside the VM.
- **SP** points to the stack of the thread inside the virtual machine.
- **CR3** and **SP** are unique identifiers of threads, but to be more human readable we map it with the process info inside the guest.



Investigations

Virtual Machine Dissection

- New trace point:
  - vcpu_enter_guest to obtain unique tuple (CR3, SP, IP)
  - If you want **more** information:
    - lttng_statedump_stack to obtain thread stack range, thread ID, thread name, hostname



Investigations

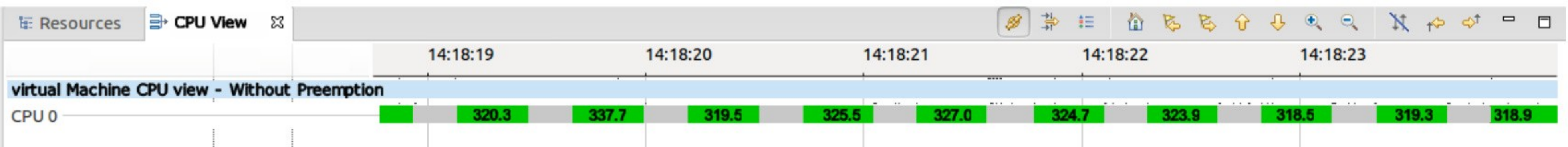
Virtual Machine Dissection

- First, fibo runs then after 1 sec, we run cpu_burn program

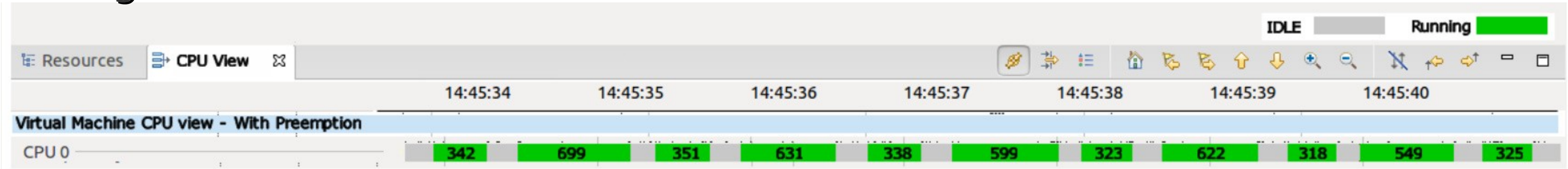
Investigations

Use Case – Preemptive Virtual Machine

30 runs of Sysbench to find primes < 10000



Average = 324ms **STD** = 5ms

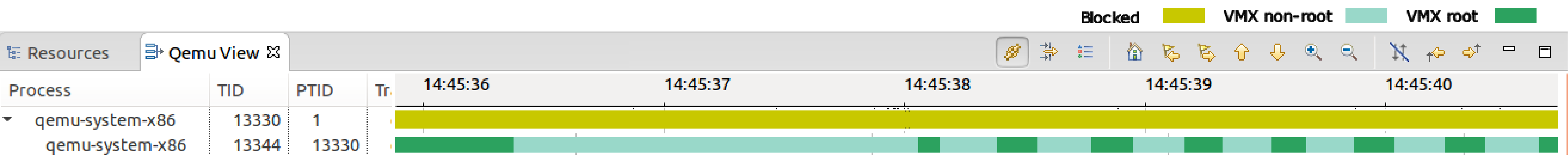


Average = 443ms **STD** = 116ms

Investigations

Use Case – Frequent Transition – Memory Overcommitment

- VM1, VM2, and VM3 are Memory intensive
- VM4, and VM5 are CPU intensive



VM name	*Execution Time(ms)*	*Freq EPT Violation*	*EPT Violation Total Time(ms)*	*Percentage(%)*
VM1	1329.09	3554	237.4	17.8
VM2	1834.5	18801	260.5	14.2
VM3	1332.4	15288	141.2	10.6
VM4	1169.1	0	0	0
VM5	1857.8	30	0.2	0

Investigations

Overload Analysis

- Compare overload of vCPU State Builder (VSB) with multi-level tracing approach by Mohamad Gebai [2]

Benchmark	*Baseline*	*VSB*	*Multi-level*	*Overhead*	
				VSB (%)	Multi-level (%)
File I/O (ms)	233	328	361	28.72	35.29
Memory (ms)	319	331	344	3.67	7.23
CPU (ms)	339	352	361	3.72	6.11

Investigations

Demo

Investigations

One More Thing ...

Investigations

Virtual Machine Classification

- KMeans clustering is a method for cluster analysis in data mining.
- Aim to partition n VMs into K clusters base on resource consuming patterns.

Metrics: disk_transfer, disk_request, CPU_avg, CPU_request

```
X= [[30512.80   503.4     9447.93   8845.87],
    [588.97     6.827     75537     3641.5],
    [46206.73   1383.75   4298.9    23756.9],
    [722.22     8.78      205141.1  2716.7],
    [708.41     8.78      214826.4  3373.6],
    [714.003    7.67      255922.2  2811.57]]
```

StandardScaler

```
Y = [[ 0.94222657  0.36056743 -1.16197163  0.17481621]
     [-0.69031714 -0.61500025 -0.51162465 -0.51364317]
     [ 1.7984347   2.0901037  -1.21264046  2.14731981]
     [-0.68304747 -0.61116339  0.76373987 -0.63598022]
     [-0.6838009  -0.61116339  0.8590478  -0.54908229]
     [-0.68349576 -0.6133441   1.26344907 -0.62343035]]
```

KMeans Result

```
[Disk, CPU, Disk, CPU, CPU,CPU]
```

Conclusion and in-progress

Inferences

- vCPU state builder (VSB) lets the Infrastructure provider to detect Over-committed and under-committed hosts.
- Profiling threads inside the Vms.
- Kmeans could cluster VMs based on resource usage pattern

Going Further

- Using kmeans with more metrics for clustering VMs based on resource usage pattern
- Applying supervise machine learning algorithms like SVM to Cluster VMs

References

[1] X. Zhao, J. Yin, Z. Chen and S. He, "Workload Classification Model for Specializing Virtual Machine Operating System," 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, 2013, pp. 343-350.

[2] Mohamad Gebai, Francis Giraldeau, and Michel R Dagenais. “Fine-grained preemption analysis for latency investigation across virtual machines”. In: Journal of Cloud Computing. December 2014, pp. 1-15. DOI : 10.1186/s13677-014-0023-3

[3] A. Cuzzocrea, E. Mumolo and P. Corona, "Cloud-Based Machine Learning Tools for Enhanced Big Data Applications," Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, Shenzhen, 2015, pp. 908-914.

[4] Jiaqing Du, Nipun Sehrawat, and Willy Zwaenepoel. 2011. Performance profiling of virtual machines. In Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '11). ACM, New York, NY, USA, 3-14

[5] Intel 64 and IA-32 Architectures Software Develope's Manual, Volume 3B, System Programming Guide, Part 2

Questions?

Hani.nemati@polymtl.ca

https://github.com/Nemati

Demo

Back UP

Slide

Questions?

Hani.nemati@polymtl.ca

https://github.com/Nemati