

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

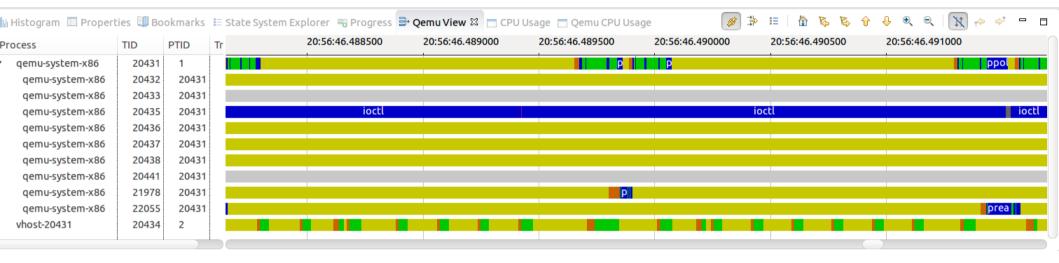
model?

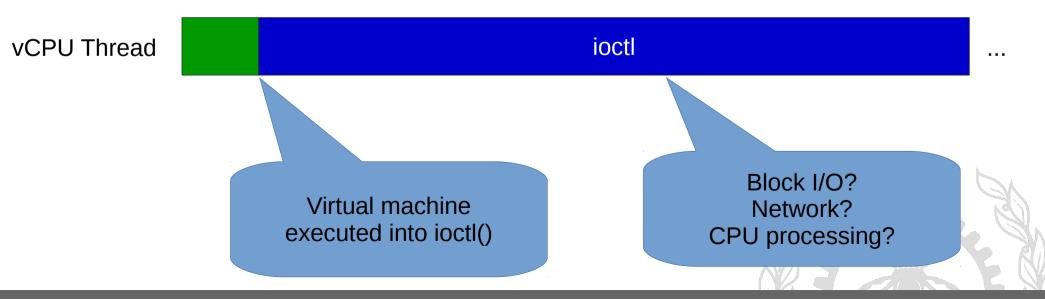
• **Classifying virtual machines** based on different resource consuming patterns [1][3]

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

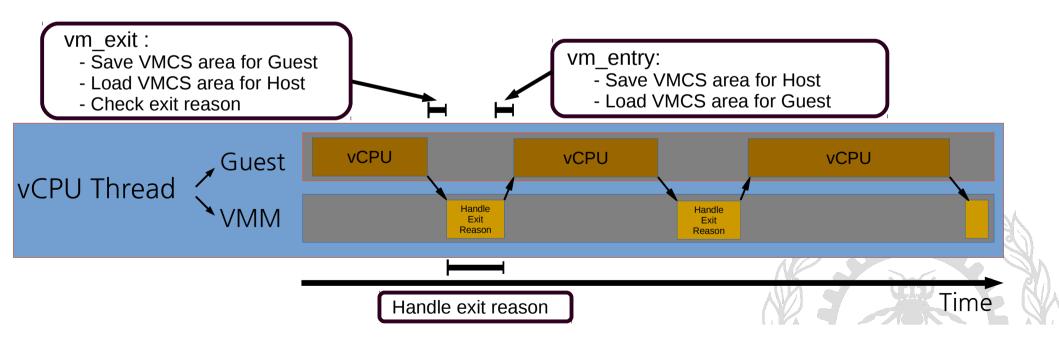
Qemu threads in Control Flow View

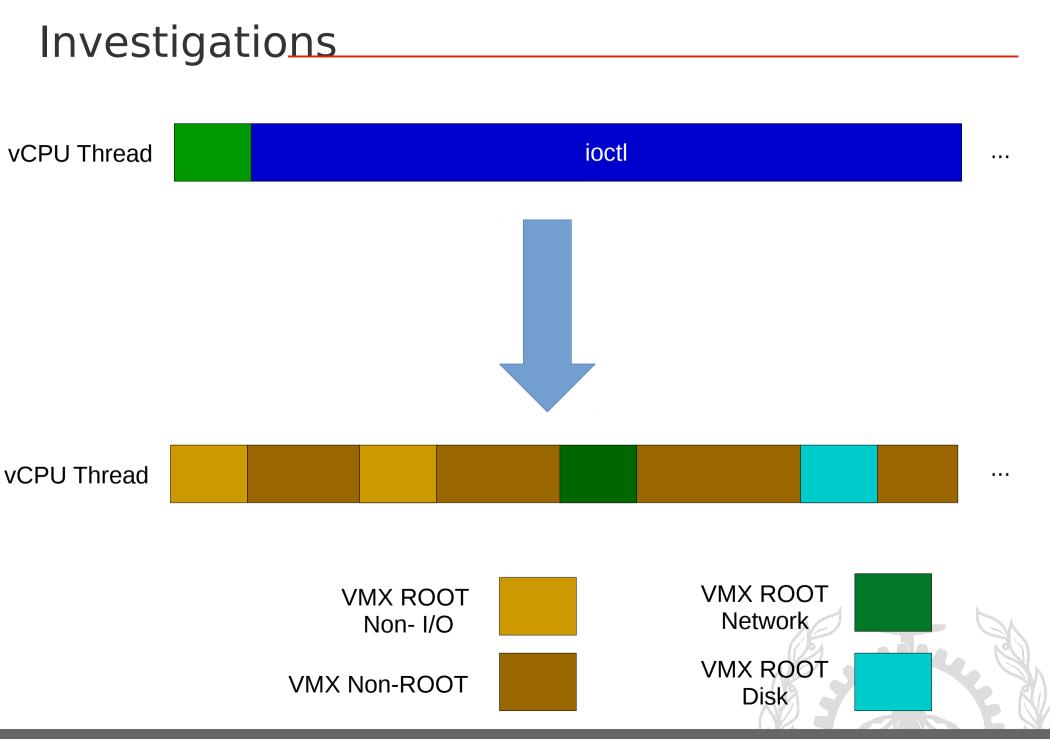




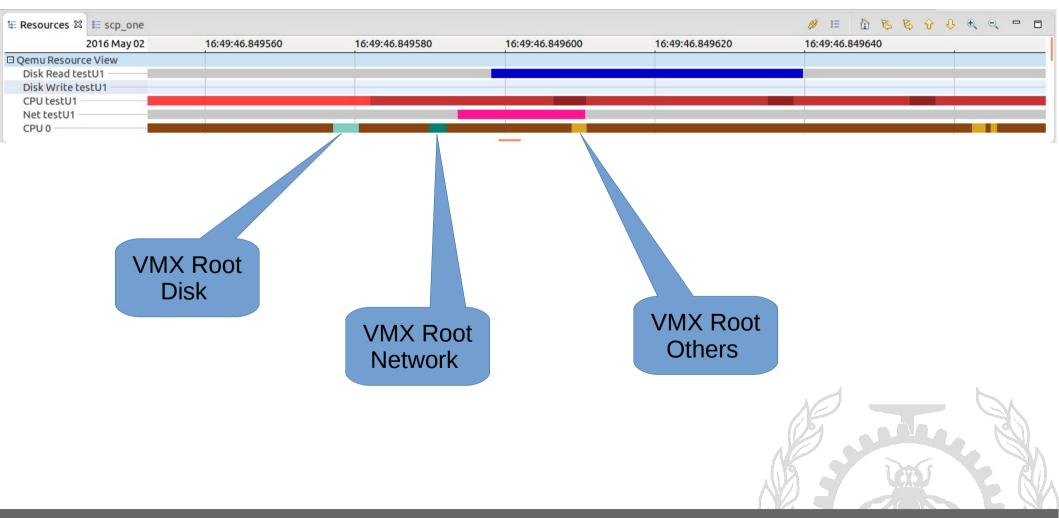
CPU Virtualization with intel-VT-x:

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



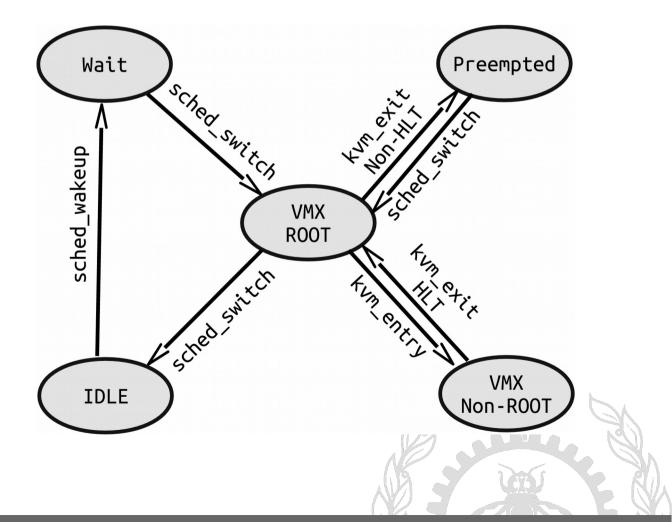


Resource View for Virtual Machine



vCPU State Detection

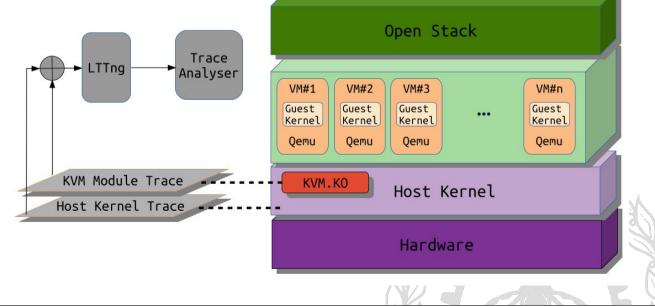
- Virtual CPU State Detection:
 - VMX Root and Non-Root State
 - IDLE state
 - Wait state
 - Preempted State
- VMX Root
 - Disk I/O
 - Network
 - Memory



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



VM Resource view and Thread view

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

문 Resources 없 🗎 scp	one					ø 🗉 🖄 🗞 🗞 🗘 🕀 🗨 🗖 🗖
2016 Ma	y 02	16:49:46.849560	,16:49:46.849580	16:49:46.849600	16:49:46.849620	16:49:46.849640
Disk Read testU1 Disk Write testU1 CPU testU1 Net testU1 CPU 0						
🖉 Tasks 🔚 State System	n Explore	🖶 Qemu View 🛱 🖶 Contro	Flow 🖶 VM View 🔲 Statistic	s 🖷 Progress	🥔 🆆 🖽	
Process	Tri	16:49:46.849560	16:49:46.849580	16:49:46.849600	16:49:46.849620	16:49:46.849640
 qemu-system-x86 				fute ppoll	pread64 write ft	utex
vhost-23842						

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

VM statistics for vCPU

Resources ☎ 🗏 scp_or	ie									#	i≡ 🟠 1	5 6 Û	<mark>⊕</mark> €	2, -
2016 May (2	16:49:42	16:49:43	16:49:44	16:49:45	16:49:46	16:49:47	16:49:48	16:49:49	п	16:49:50	16:49:51		
Disk Read testU1	j													
Disk Write testU1										li	10	1	24	
CPU testU1														
Net testU1														
CPU 0														

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.	16:49:49	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
<u>▶ 12</u>	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.	16:49:49	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

VM statistics for Disk and Network

2097536×512	= 1073938432
1073938432÷1024	= 1048768
1048768÷1024	= 1024.1875
1024.1875÷1024	= 1.0002

(W/)

t Resources ☎ E scp_one 2016 May 02 16:49:	42 16:49:43	16:49:44	16:49:	15 16	:49:46	16:49:47		6:49:48	16:49:49	# 16:49:50		0
Disk Read testU1			10.49.4		.49.40	A STATE OF STATE OF STATE	-		10.49.49	and the second s	16:49:51	
CPU testU1											· · · · · · · · · · · · · · · · · · ·	
Net testU1												
▼ DiskQemu		3651	1		1	16:49	0.41	16:49:52	DiskQer	2011		
 ✓ 23842 		3652				16:49		16:49:52	-	nu/23842		
vmName		3653		EL 14	String			16:49:5:		nu/23842/	vm Name	
✓ read		4055		.01	Sunny	16:49		16:49:5		nu/23842/		
numberOfSubmited		4055			Int	16:49		16:49:4			read/numberOfSubi	mited
transfer		4050			Int	16:49		16:49:4			read/transfer	miceu
STATUS		4057			Int	16:49		16:49:4			read/STATUS	
latency		4050		4708567	Long	16:49		16:49:4			read/latency	
totalTransfer		4068		7536	Int	16:49		16:49:4			read/totalTransfer	
✓ write		5329		1550	inc	16:49		16:49:52		nu/23842/		
numberOfSubmited		5330			Int	16:49		16:49:5	-		write/numberOfSub	mited
transfer		5331			Int	16:49		16:49:5:			write/transfer	meed
STATUS		5332			Int	16:49		16:49:5:			write/STATUS	
latency		5340		18701	Long	16:49		16:49:5			write/latency	
totalTransfer		5341			Int	16:49		16:49:5:			write/totalTransfer	
					1 100							
NetQemu			3922					6:49:41.	16:49:5:			
23870			3923				1	6:49:41.	16:49:5:	NetQe	mu/23870	
STATUS			3924	0		Int	1	6:49:49.	16:49:4	NetQe	mu/23870/STATUS	5
Netif			3925	0		Int	1	6:49:49.	16:49:4	NetQe	mu/23870/Netif	
Netdev			3926	0		Int	1	6:49:49.	16:49:4	NetQe	mu/23870/Netdev	1
tNetdev			3932	26101	62	Long	1	6:49:49.	16:49:4		mu/23870/tNetde	
tNetif			3933	10785	:	Long		6:49:49.	16:49:4		mu/23870/tNetif	

Investigations VM statistics for vCPU

2016 Apr 12	11:52:25	11:52:30	11:52:35	11:52:40	11:52:45	11:52:50	11:52:55
emu Resource View							
Disk Read testU1					30		
Disk Read testU2							
Disk Read testU3							
Disk Read testU5							
Disk Read testU6	· - ·						
Disk Read testU7							
Disk Write testU1	• • • • • •	li.		•			
Disk Write testU2	(•	-				•
Disk Write testU3		-	i l				·
Disk Write testU5	•					•	
Disk Write testU6		ľ.		Í			-
Disk Write testU7		· ·		- · ·	· · ·		· ·
CPU testU1							
PU testU2							
CPU testU3							······································
CPU testU5							
CPU testU6							
CPU testU7			فردفه ويستعدا أنجا أوالا ونفيت				
PU 0							
CPU 1							
CPU 2							

🔄 Tasks 🗄 State System Explorer 🖾 🖶 Qemu View 🖶 Control Flow 🖶 VM View 🗔 Statistics 📆 Progress

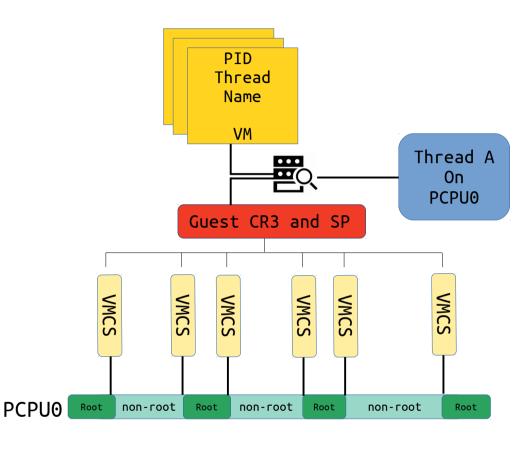
State System / Attribute	Quark Value at timest	amp Type	Start time	End time	Full attribute path
vmName	3185		11:52:21.574 949 968	11:53:10.202 144 279	vmName
 DelayCPU 	3215		11:52:21.574 949 968	11:53:10.202 144 279	DelayCPU
 preempting 	3216		11:52:21.574 949 968	11:53:10.202 144 279	DelayCPU/preempting
▼ 3046	3652 1809577163	Long	11:52:52.693 300 272	11:52:52.957 273 439	DelayCPU/preempting/3046
 Other 	3971		11:52:21.574 949 968	11:53:10.202 144 279	DelayCPU/preempting/3046/Other
- VM	4048		11:52:21.574 949 968	11:53:10.202 144 279	DelayCPU/preempting/3046/VM
▶ 3046	4049 888023378	Long	11:52:51.960 351 008	11:52:58.310 338 418	DelayCPU/preempting/3046/VM/3046
▶ 3144	4584 130523526	Long	11:52:51.874 244 051	11:53:10.202 144 279	DelayCPU/preempting/3046/VM/3144
▶ 2822	4611 233006703	Long	11:52:46.958 630 036	11:52:55.875 261 449	DelayCPU/preempting/3046/VM/2822
▶ 3277	5558 102240683	Long	11:52:51.874 218 518	11:53:10.202 144 279	DelayCPU/preempting/3046/VM/3277
▶ 2933	6096 81162859	Long	11:52:51.874 192 002	11:53:10.202 144 279	DelayCPU/preempting/3046/VM/2933
▶ 2718	6272 288521483	Long	11:52:45.797 038 467	11:52:58.348 951 331	DelayCPU/preempting/3046/VM/2718
▶ 3144	3727 1991714432	Long	11:52:51.677 286 710	11:52:53.665 275 233	DelayCPU/preempting/3144
▶ 2718	4015 1065946539	Long	11:52:51.856 743 328	11:52:53.385 303 802	DelayCPU/preempting/2718
▶ 3277	4036 1733671860	Long	11:52:52.677 276 143	11:52:54.677 292 390	DelayCPU/preempting/3277
▶ 2933	4095 2457673825	Long	11:52:52.307 664 547	11:53:07.834 862 177	DelayCPU/preempting/2933
▶ 2822	4148 564532519	Long	11:52:52.667 846 133	11:52:52.973 921 822	DelayCPU/preempting/2822
usage	3232		11:52:21.574 949 968	11:53:10.202 144 279	DelayCPU/usage
waiting	3234		11:52:21.574 949 968	11:53:10.202 144 279	DelayCPU/waiting

* - -

Investigatio<u>ns</u>

Virtual Machine Dissection [4]

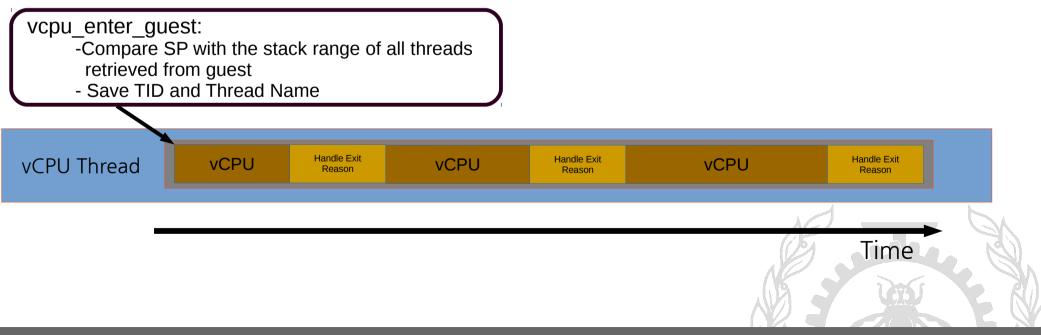
- Process Identifier (PID) and Process Name inside the VMs is not accessible from Host tracing.
- **CR3** points to the page directory of process inside the VM.
- **SP** points to the stack of the thread inside the virtual machine.
- CR3 and SP are unique identifiers of threads, but to be more human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





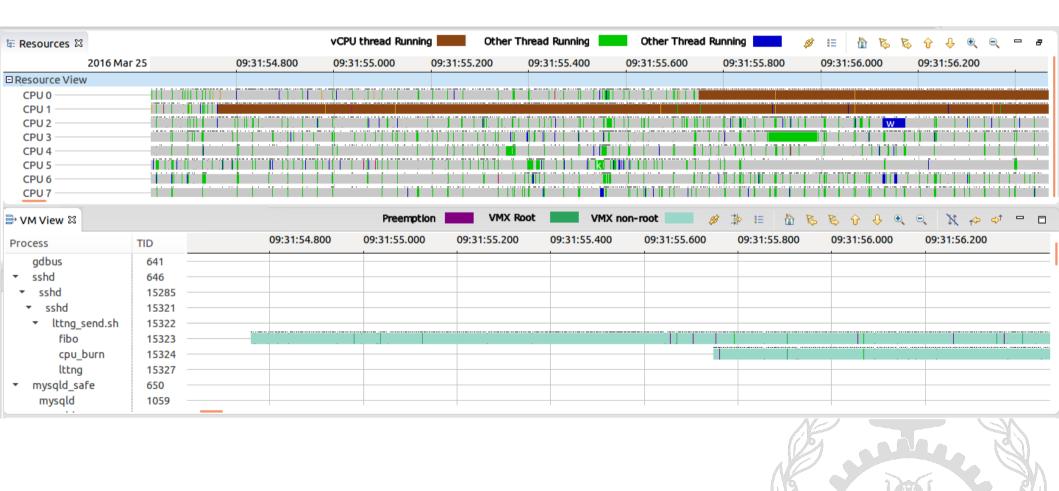
Virtual Machine Dissection

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



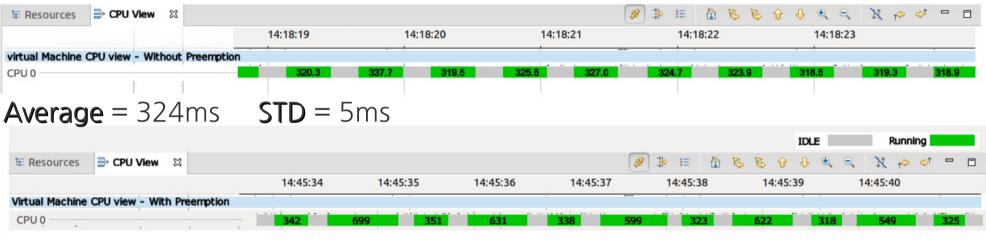
Investigations Virtual Machine Dissection

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

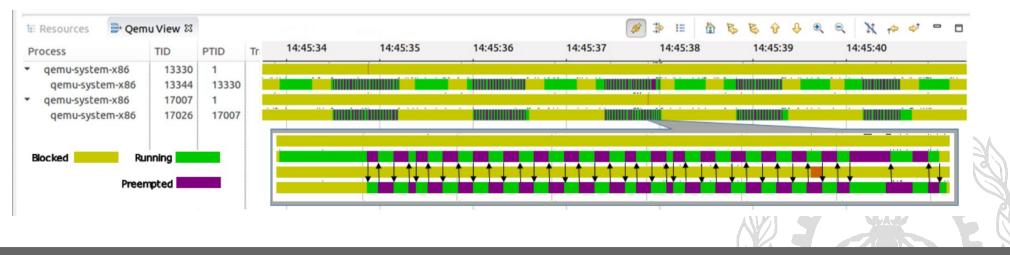


Use Case – Preemptive Virtual Machine

30 runs of Sysbench to find primes < 10000



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



Use Case – Frequent Transition – Memory Overcommitment

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

								Blocked		VMX non-root	VMX root
E Resources	₽ Qemu	រ View ន						🅖 🌼 💷 💧	5 B	· · · · · · · · · · · · · · · · · · ·	X 💠 🗣 🗆 🗆
Process		TID	PTID	Tr	14:45:36	14:45:37	14:45:38		14:45:39		14:45:40
 qemu-system-> gemu-system 		13330 13344	1 13330								

VM name	Execution Time(ms)	Freq EPT Violation	EPT Violation Percentage(% Total Time(ms)				
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			



Investigatio<u>ns</u>

Overload Analysis

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

Benchmark	Baseline	VSB	Multi-level_	Over	Overhead		
Бенсптак	Dusenne	, 5 D	mun-rever—	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		



Demo





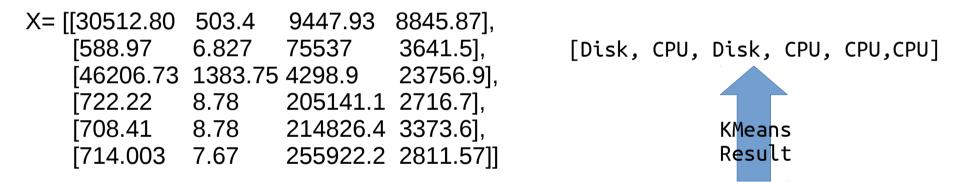
One More Thing ...

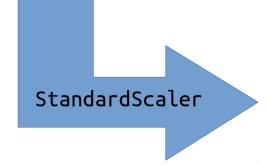


Virtual Machine Classification

- 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





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

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

