

Low-Latency KVM Hypervisor

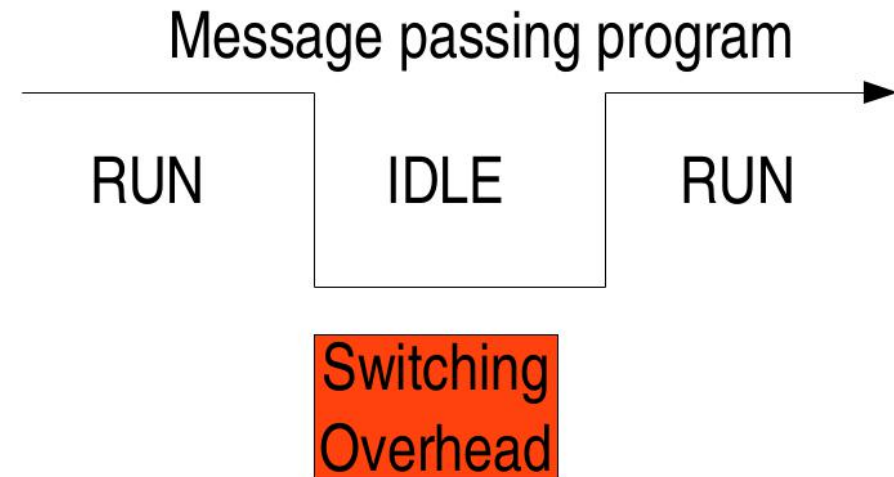
Wanpeng Li

Agenda

- Adaptive halt-polling
 - Background
 - Halt-polling
 - Adaptive polling for guest halt
- VMX Preemption timer

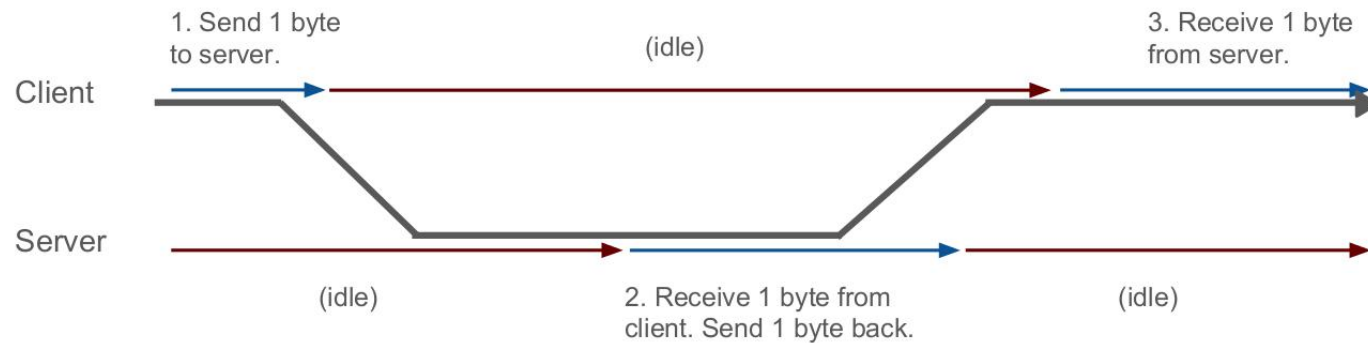
Message passing workloads

- Usually, anything that frequently switches between running and idle
- Event-driven workloads
 - LAMP servers
 - Memcache
 - Redis
 - SAP HANA
- Inter-process communication
 - TCP_RR (benchmark)



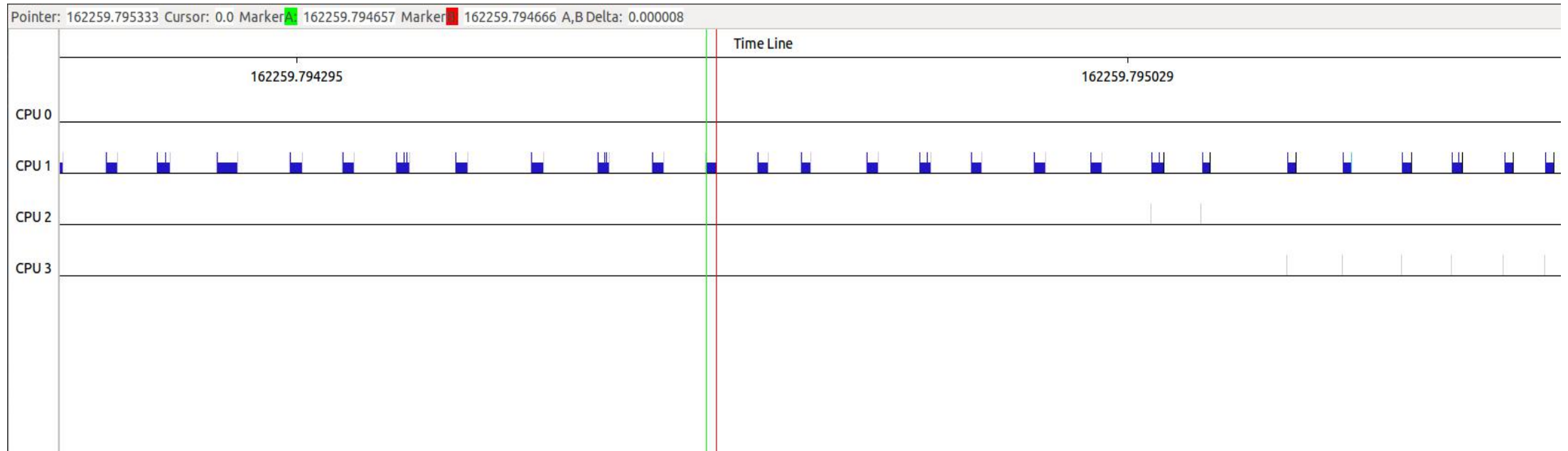
Message passing workloads

- Microbenchmark: Netperf TCP_RR
 - Client and Server ping-pong 1-byte of data over an established TCP connection
 - Performance: Latency of each transaction
- One transaction:



Message passing workloads

- Frequent transitions between running and idle, spends little time processing each message



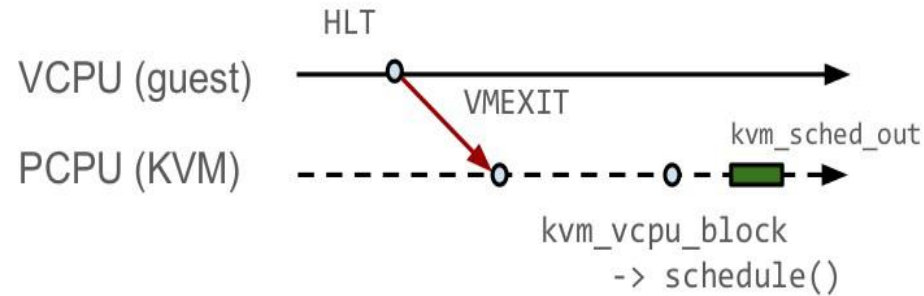
HLT

- HLT

- x86 instruction, CPU stops executing instructions until an interrupt, debug exception etc arrive

- How it works in KVM

- Place vCPU thread on a wait queue
 - Yield the CPU to another task



context switch to another user task, kernel thread, or idle

- The overhead

- around 8500 cycles between later kvm_vcpu_kick and kvm_sched_in

Never schedule!

- defeat the purpose of CPU overcommit in cloud companies
- some cloud management program will monitor pCPU usage and do the load balance, never schedule just make it mess

Halt-Polling

- Step 1: Poll

- For up to halt_poll_ns nanoseconds:
 - If a task is waiting to run on our CPU, go to Step 2
 - Check if a guest interrupt arrived. If so, we are done.
 - Repeat

- Step 2: schedule()

- Schedule out until it's time to come out of HLT.

- Pros:

- Works on short HLTs (< halt_poll_ns ns)
- vCPUs continue to not block the progress of other threads

- Cons:

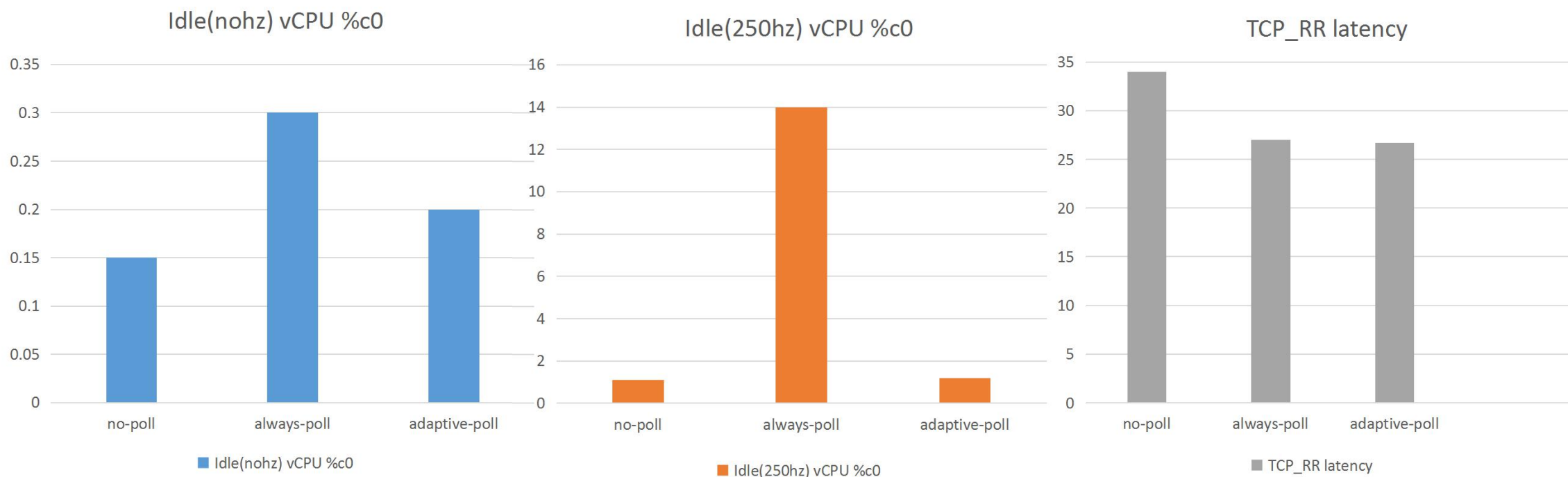
- Increases CPU usage (14% for each idle pCPU(windows guests) if halt_poll_ns = 500,000ns)

Adaptive polling for guest halt

- Step 3: adaptive polling
 - The poll duration can be adaptively shrink/grow according to the history behavior
 - grow halt_poll_ns progressively when short halt is detected (we can get benefit from the polling)
 - shrink halt_poll_ns aggressively when long halt is detected (we can't get benefit from the polling)

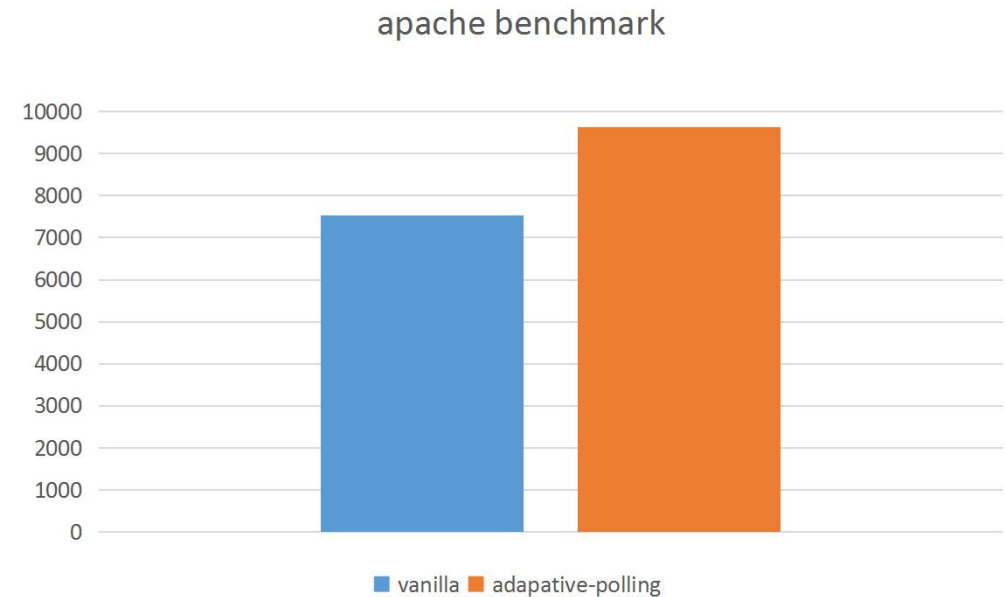
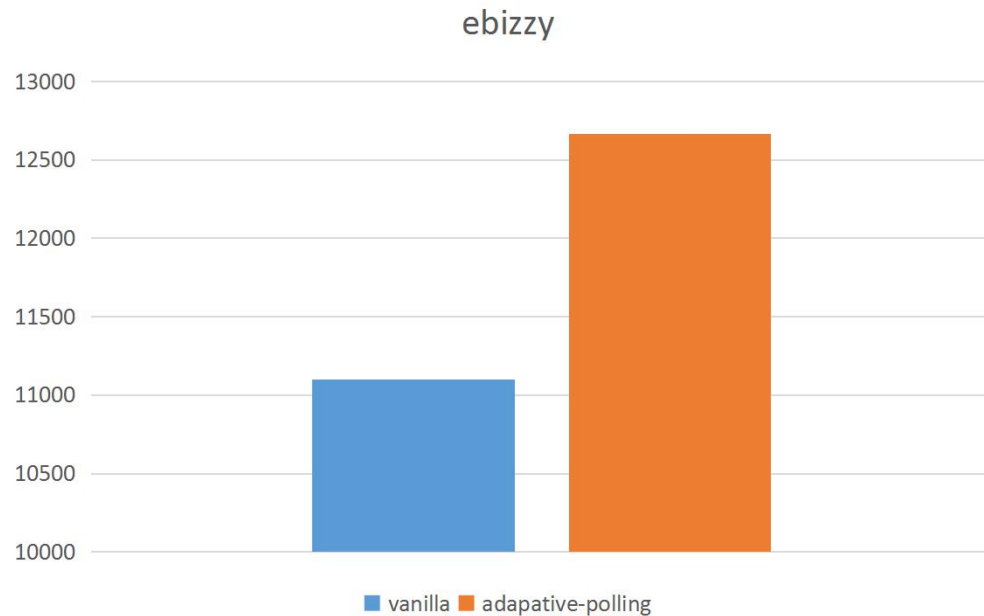
Adaptive polling for guest halt(Cont.)

- Performance data



Adaptive polling for guest halt(Cont.)

- Performance data

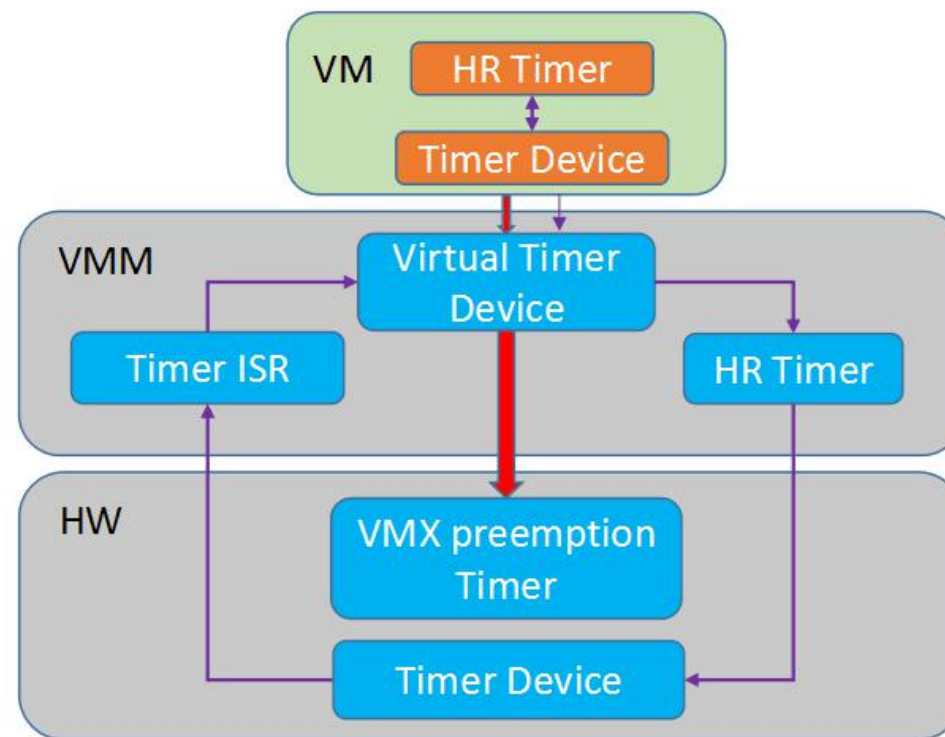


Adaptive polling for guest halt(Cont.)

- As the polling nature, some cloud companies will configure to their preferred balance of cpu usage and performance, and other cloud companies for their NFV scenarios which are more sensitive to latency are vCPU and pCPU 1:1 pin.

VMX Preemption timer

- VMX preemption will count down in VMX non-root mode and VM-exit when it reaches zero
- It reduces the cost of:
 - hrtimer cost
 - timer interrupt ISR
 - vm-exit/entry

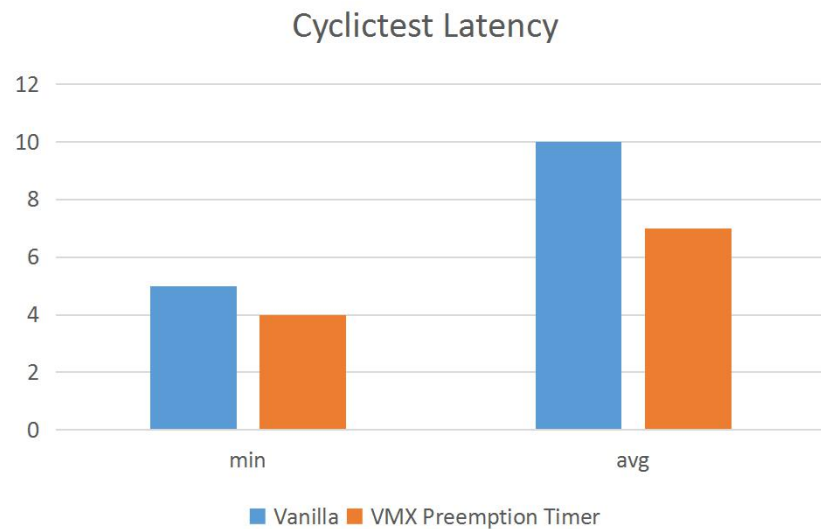


VMX Preemption timer(Cont.)

- The `hrtimer_start/hrtimer_cancel` depends on the current `hrtimer` situation.
- The timer interrupt ISR has far more cost than `VMExit` handling.
- In system w/o PI, a pair of `vm-exit/entry` can be caused by the host timer interrupt.
- Both of LAPIC timer `tsc deadline mode` and `periodic/oneshot mode` can utilize VMX Preemption timer currently.

VMX Preemption timer(Cont.)

- Performance data



Reference

- David matlack, Message Passing Workloads in KVM
- <https://lkml.org/lkml/2015/9/3/615>
- <https://www.spinics.net/lists/kvm/msg134057.html>
- <https://lkml.org/lkml/2016/10/24/219>

Q/A?