Boosting Dedicated Instance via KVM Tax Cut

KVM FORUM 2019

Wanpeng Li
wanpengli@tencent.com
Agenda

- Exitless Timer
- Exitless IPI
- Per-VM cap to disable exits
- KVM_HINTS_DEDICATED performance hint
- Adaptive tune advance lapic timer
- Adaptive halt-polling in guest/host
Exitless Timer

Motivation
- both arm timer and timer fire incur vmexits
- dedicated instance encounter performance jitter
Exitless Timer

- Injection exitless
  - offload lapic timer to the housekeeping cpus
  - inject expired timer interrupt via posted interrupt
  - fine tuned host via enable nohz_full, disable mwait/pause/hlt vmexits etc
Exitless Timer

- Normal KVM interrupt delivery

- Housekeeping cpus delivery interrupt via posted-interrupt
Exitless Timer

Performance data

![Bar chart showing performance data](chart.png)

- percent of external interrupt vm-exit time
- vanilla
- Exitless Timer
Exitless IPI

- Each writes to ICR register will cause a vmexit in x2apic physical mode, multicast IPIs and “Function Call interrupts” make it worse when scaling to large VMs. Use a hypercall to send IPIs to multiple vCPUs.
Exitless IPI

**Evaluation Environment:**
- Hardware: Xeon Skylake 2.5GHz, 2 sockets, 40 cores, 80 threads
- VM: 80 vCPUs
- Test case: IPI microbenchmark

![Bar chart showing performance comparison between x2apic physical mode and x2apic cluster mode. The x2apic physical mode is significantly slower by 154%, while the x2apic cluster mode is faster by 22%.]
Enable KVM_CAP_X86_DISABLE_EXITS capability on a VM provides userspace with a way to no longer intercept MWAIT/HLT/PAUSE LOOP/read cstate msrs for improved latency in some workloads.

[Bar chart showing percent of bare-metal performance with 19% increase for exposing mwait]

[Bar chart showing hackbench time, with 0.8 vs 0.95 for vanilla and expose mwait]
KVM_HINTS_DEDICATED performance hint

- Allows a guest to enable optimizations when running on dedicated pCPUs
  - choose qspinlock
  - native tlb shootdown
  - disable pv sched yield
  - enable guest halt-polling

![Graph showing performance comparison with and without performance hint. The performance hint shows a 12% improvement in hackbench time.](image-url)
Adaptively tune advance lapic timer

- Hidden hypervisor overhead between lapic timer fires and before vmentry
Adaptively tune advance lapic timer

- Adaptive tune step by step smoothly
  - reduce advance value when it is too early
  - increase advance value when it is too late
Adaptive halt-polling in host

- 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)
Adaptive halt-polling in host

- 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:

![Diagram showing message passing workloads](image)
Adaptive halt-polling in host

- Message passing workloads
  - Frequent transitions between running and idle, spends little time processing each message
Adaptive halt-polling in host

- When a guest vcpu has ceded, the host kernel polls for wakeup conditions before giving up the cpu to the scheduler.
- 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
    - shrink halt_poll_ns aggressively when long halt is detected
Adaptive halt-polling in host

Performance data

![Graph showing performance data]

- 28% improvement in adaptive-polling over vanilla.
Adaptive halt-polling in guest

- `cpuidle_haltpoll` governor and `haltpoll cpuidle` driver
  - **pros**
    - Avoid sending an IPI when performing a wakeup
    - `vmexit` cost can be avoided
  - **cons**
    - Polling is performed even with other runnable tasks in the host
  - But now, it is enabled when hypervisor gives dedicated performance hint
Adaptive halt-polling in guest

Performance data
Reference

- https://lkml.org/lkml/2019/7/5/712
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=b51700632e0e53254733ff706e5bdca22d19dbe5
- https://lkml.org/lkml/2018/2/12/1036
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=3b8a5df6c4dc6df2ab17d099fb157032f80bdca2
- https://www.spinics.net/lists/kvm/msg190684.html
Q/A?