Real Time & Fast Live Migration Update for NFV

Contributor: Li Liang <liang.z.li@intel.com>
Jiang Yunhong <yunhong.jiang@intel.com>
Speaker: Xiao Guangrong <guangrong.xiao@linux.intel.com>
Agenda

• Real Time Update
  • Hardware features
  • Software enhancement

• Fast Live Migration Update
  • Software enhancement
  • Hardware acceleration
Real Time Update: Hardware Features

- Cache Qos
- APICv & Posted Interrupt
- VMX Preemption Timer
Cache Qos

• Cache Monitor
  • Cache Monitoring Technology (CMT): Monitor L3 Cache Occupancy
  • Memory Bandwidth Monitoring (MBM): Monitor L3 Total & Local External Bandwidth
  • Have integrated to perf tool
Cache Qos (Cont.)

• Cache Allocation
  • Current issue

Without CAT

- Hi Pri App
- Lo Pri App
- Core 0
- Core 1
- Shared LLC, Low priority got more resource

With CAT

- Hi Pri App
- Lo Pri App
- Core 0
- Core 1
- Shared LLC, High priority got more resource
Cache Allocation

- Cache Allocation
  - CAT (Cache Allocation Technology)
  - Specify the amount of cache space into which an application can fill
  - The application is associated to COS (Class Of Server)
Cache Allocation (Cont.)

- Code and Data Prioritization (CDP) Technology
  - It’s an extension of CAT. CDP enables isolation and separate prioritization of code and data fetches to the L3 cache.
Cache Allocation (Cont.)

- Performance data
Real Time Update: Hardware Features

- APICv & Posted Interrupt
  - Inject the interrupt to guest directly
  - Avoid VMExit cost
Real Time Update: Hardware Features

- VMX preemption
  - Latency for tradition vtimer
    - Register access to virtual timer device
    - Linux High Resolution timer system
  - It counts down in VMX non-root mode
- VM-exit when it reaches zero
- Avoid complex host HR timer
- Reduce VMExit and context switch
Real Time Update: Software Enhancement

• Non-threaded VFIO MSI
  • Long path to deliver IRQ for threaded IRQ handler:
    Vcpu thread running -> Hardware IRQ happen -> schedule kernel thread for the VFIO MSI -> schedule to the VCPU thread -> inject IRQ to the guest.
  • With non-threaded IRQ
    Vcpu thread running -> Hardware IRQ happen -> VFIO IRQ handler -> back to vCPU thread and inject to the guest
Non-threaded VFIO MSI

• Performance
Fast Live Migration Update: Software Enhancement

• Skip transmission of guest’s free pages
  • Get free pages information from guest and skip them during live migration

Before skip the free pages

After skip the free pages

Migrated page
Guest used page
Guest Free page
Skip transmission of guest’s free pages

• Implementation details
  • Start dirty page logging before requesting the free page bitmap
  • Traversing the free pages list to construct a free page bitmap
  • Using virtio for communication between guest and hypervisor
  • Process the raw page bitmap contain holes
  • Filter out free pages from migration dirty page bitmap
Skip transmission of guest’s free pages (Cont.)

• Test result
  • Idle guest with 8GiB RAM which just booted (left)
  • Guest with 8GiB RAM, first run an application touches 7GiB of RAM, and then terminate the application (right)

 Idle guest just boots

 Guest has ever run workloads

<table>
<thead>
<tr>
<th>Total time(ms)</th>
<th>Downtime(x10us)</th>
<th>Transferred ram(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>Optimized</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total time(ms)</th>
<th>Downtime(x10us)</th>
<th>Transferred ram(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>Optimized</td>
<td></td>
</tr>
</tbody>
</table>
Skip transmission of guest’s free pages (Cont.)

• Test result
  • DPDK L2 forwarding, line rate 2013Mbps, 64bytes package.
Fast Live Migration Update: Hardware Feature

• QAT (Intel’s Quick Assistant Technology)
  • It’s integrated to the chipset which can provide (de)compression and (de)encryption service
  • Throughput can reach to 24Gpbs (100Gbps with newer product)
  • (De)Compression multiple pages in a single request
  • Can buffer multiple requests
  • Use physical address for (de)compression
QAT

• QAT & QEMU
  • All the jobs are done in migration thread
  • Could send uncompressed page instead of waiting the compression done.
  • Zero page checking is not necessary
  • Pre-reading ‘/proc/self/pagemap’ and cache the entry can accelerate virtual to physical address translation
  • mlock() is required
QAT (Cont.)

• In 10Gbps network environment
  • Workload writes CalgaryCorpus data to the 7GB of guest memory first, and then writes CalgaryCorpus data to 1GB area of guest memory periodically.
  • Shorten the total live migration time about 40%, reduce the VM downtime about 70%, reduce the network traffic about 45% with about 10% extra CPU usage.
QAT (Cont.)

• Worst case in 10Gbps network environment
• Workload writes Random number to the 7GB of guest memory first, and then writes Random number to 1GB of guest memory periodically.
• QAT can do a better job even in the worst case.
Q/A?