Light Weight Virtualization with QEMU/KVM

Anthony Xu <anthony.xu@intel.com>
Chao Peng <chao.p.peng@intel.com>
Haozhong Zhang <haozhong.zhang@intel.com>
Outline

Background
Optimization for VM Launch Time
Memory Footprint
Integration with Intel Clear Container
Upstream
Status & TODO
## Container vs. Traditional VM

<table>
<thead>
<tr>
<th></th>
<th>Container</th>
<th>Traditional VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Time</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Resource Efficiency</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
## Light Weight VM

<table>
<thead>
<tr>
<th></th>
<th>Container</th>
<th>Traditional VM</th>
<th>Light Weight VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Time</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Resource Efficiency</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Density</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Deployment</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Security</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
QEMU Lite

• Target for container-like usage scenarios

• Optimized for the launch time
  • 1276 ms ➔ 335 ms

• Smaller memory footprint
  • More optimization on the way

• Integrated with Intel Clear Container
  • Docker-like engine
Outline

Background
Optimization for VM Launch Time
Memory Footprint
Integration with Intel Clear Container
Upstream
Status & TODO
Baseline

- Host
  - Intel Xeon E5-2698 v3 @ 2.30 GHz / [Memory]

- Guest (QEMU configurations)
  - Q35 / 6 VCPUs / 1G RAM / virtio-blk / -kernel / no network

Total: 1276 ms
Tweak QEMU & Kernel Configurations

- Disable QEMU features at compile time
- Tune guest kernel boot parameters
  - rcupdate.rcu_expedited=1
  - pci=lastbus=0
  - ...
QEMU Optimizations

• Cache KVM_GET_SUPPORTED_CPUID
  • 5922 duplicated calls during QEMU initialization
  • QEMU commit 494e95e “target-i386: kvm: cache KVM_GET_SUPPORTED_CPUID data”
  • QEMU: 272 ms ➔ 90 ms

• Parallelize VCPU initialization
  • QEMU: 90 ms ➔ 54 ms
New Machine Type pc-lite

- Minimum Devices
  - APIC
  - PCI Host
  - Virtio Console
  - NVDIMM
  - ...

- SeaBIOS: 43 ms ➔ 29 ms
- Guest kernel: 151 ms ➔ 142 ms
And more ...

• Use vNVDIMM device as guest disk drive
  • DAX
    • Guest kernel: 142 ms ➔ 124 ms
    • Guest userspace: 240 ms ➔ 195 ms

• Remove guest BIOS completely
  • Patch guest ACPI in QEMU
  • Load guest ELF kernel in QEMU
  • SeaBIOS: 29 ms ➔ 0 ms
  • OptionROM: 468 ms ➔ 0 ms
  • Guest kernel: 124 ms ➔ 72 ms
Final Result

Total: reduced by 74% (1276 ms → 335 ms)
<table>
<thead>
<tr>
<th>Feature</th>
<th>QEMU</th>
<th>SeaBIOS</th>
<th>OptionROM (linuxboot)</th>
<th>Linux Kernel</th>
<th>Userspace (systemd)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove BIOS</td>
<td>68</td>
<td>72</td>
<td>195</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vNVDIMM</td>
<td>54</td>
<td>29</td>
<td>468</td>
<td>124</td>
<td>195</td>
</tr>
<tr>
<td>Q35 Opt</td>
<td>54</td>
<td>29</td>
<td>468</td>
<td>142</td>
<td>240</td>
</tr>
<tr>
<td>Parallelize VCPU Init</td>
<td>54</td>
<td>43</td>
<td>468</td>
<td>155</td>
<td>240</td>
</tr>
<tr>
<td>Cache CPUID</td>
<td>90</td>
<td>43</td>
<td>468</td>
<td>155</td>
<td>240</td>
</tr>
<tr>
<td>Kernel Parameter</td>
<td>232</td>
<td>43</td>
<td>468</td>
<td>155</td>
<td>240</td>
</tr>
<tr>
<td>QEMU Config</td>
<td>232</td>
<td>43</td>
<td>468</td>
<td>211</td>
<td>271</td>
</tr>
<tr>
<td>Baseline</td>
<td>283</td>
<td>43</td>
<td>468</td>
<td>211</td>
<td>271</td>
</tr>
</tbody>
</table>
Outline

Background
Optimization for VM Launch Time
Memory Footprint
Integration with Intel Clear Container
Upstream
Status
Memory Consumption

- 1 x VM: 6 vCPU, 1G RAM
- Read /dev/$pid/status
Potential Optimizations

- Still ongoing
- Remove unnecessary devices
  - Reduce memory consumption by QEMU itself
- COW share ROM
  - e.g. ROM for guest ELF kernel
  - Reduce duplicated host/guest copies if (almost) no modifications
- Lazy creation of dirty memory bitmap
  - Defer to its first usage (migration)
- KSM
- VMFork/Clone
- Tools to fine-grained profile/trace the memory consumption
Outline

Background

Optimization for VM Launch Time

Memory Footprint

Integration with Intel Clear Container

Upstream

Status
Integration with Intel Clear Container

- QEMU Lite in Intel Clear Container v2.0
  - Replace kvmtool used in Intel Clear Container v1.0

- Benefit from Docker/rkt compatible interface
  - Support docker images
  - Better deployment
Outline

Background
Optimization for VM Launch Time
Memory Footprint
Integration with Intel Clear Container
Upstream
Status
PC-lite or Q35

• [RFC 0/9] Introduce light weight PC platform pc-lite

• All optimizations currently go into a new machine type pc-lite.
  • Less pollutions to other machine types
  • Easier to optimize
  • Extra cost for maintenance

• Optimize Q35?
  • Make the work more widely useful
  • Add ability to disable (more) features and devices
No Firmware or Light-weight Firmware

- Completely remove guest firmware currently.
  - Significant speedup
  - Specific to Linux

- Light-weight firmware
  - e.g. qboot
  - Support more guest operating systems
  - More optimizations maybe needed

- Keep no firmware as an option
  - For usage scenarios requiring extreme launch time
Outline

Background
Optimization for VM Launch Time
Memory Footprint
Integration with Intel Clear Container
Upstream
Status
Status

- [RFC 0/9] Introduce light weight PC platform pc-lite
- Github repo
  - Patches: https://github.com/chao-p/qemu/tree/pc-lite-v1
  - Tools/Guides: https://github.com/chao-p/qemu-lite-tools
- Integrated in Intel Clear Container v2
- Ongoing
  - More optimizations for launch time
  - Optimizations for memory footprint
  - Upstream