Bring QEMU to Micro Service World

Zhang Yulei <yuleixzhang@tencent.com>
Xiao Guangrong <xiaoguangrong@tencent.com>
Agenda

- Background
- QEMU Adaption
- Future works
Tencent Cloud can serve globally with large scale of resources.
Tencent cloud has been contributing to 100000+ clients from different industries for digitalization.
Background

- Aim to micro service
- Fast bootup
- Massive deployment
- Short life cycle
- Less memory footprint
- High level of security

Current solutions
- Firecrack, crosvm, RustVMM based hypervisor
Why QEMU adaption?

Good for devops

Good for developer's knowledges
QEMU Adaption

Pro
1. Strong hardware isolation
2. Rich existing features

Con
1. Redundant code path
2. Slow guest start up within seconds
3. More resource used
Our Solution

QEMU Basepoint

- Bypass Guest initialization
- Bypass QEMU initialization
**QEMU basepoint: Bypass Guest init.**

Leverage the existing migration strategy to create the basepoint template which could be used to restore the Guest instead of booting guest kernel every time.

- **Base on shared memory migration with minimum system resource**
- *(Inspirit from kata container)*
QEMU basepoint: Bypass Guest init. (Cont.)

Restore the guest VM from the basepoint image will significantly speed up the guest boot up which eliminates the cost for guest kernel boot up.
QEMU basepoint: Bypass QEMU init.

- Besides skipping the guest kernel boot period, QEMU initial code will introduce appreciable latency.
- Fork the QEMU process after Base VM boots up would be able to further speed up the micro service instance start up
QEMU basepoint: Bypass QEMU init. (Cont.)

- **Base VM**
- **Basepoint mode**
- **QEMU initialization period**
- **Boot up**
- **KVM Module independent**
- **create**
- **template**
- **cpu/device/memory/disk**

- **fork**
- **VM1**
- **VM2**
- **VM3**

- **Micro Service VM1**
- **Micro Service VM2**
- **Micro Service VM3**
## VM startup period

<table>
<thead>
<tr>
<th></th>
<th>Original Qemu with Optimized linux kernel</th>
<th>+ Bypass kernel init</th>
<th>+ Bypass Qemu init</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot up (ms)</td>
<td>500</td>
<td>150</td>
<td>35</td>
</tr>
</tbody>
</table>
Security
Security approach

Enhance the Guest security after restore from the same template

- Use virtio-rng as random number generator for the guest
- Reseed the entropy for random number generator after restore the guest from basepoint template
- However, kernel randomness (e.g. KASLR) still does not work...
  - Each tenant has a dedicated template
Future works
Future works

- Optimize the current QEMU code
- QEMU modulization
- Guest kernel security enhancement
Summary

1. Based on QEMU code to achieve the micro service requirement to fast deploy intensive micro services in a extremely short period.

2. With minimum modifications and easily adapt to the existing framework in public cloud.

3. Our solution, QEMU baspoint, bypass QEMU & Guest init. completely.
Q&A