StratoVirt 热插拔技术研发实践与未来展望

中国电信股份有限公司研究院

2023年10月

目录

一 热插拔技术研发实践

二 热插拔技术未来展望

轻量级虚拟化实现性能提升的同时缺失热插拔特性

以RUST-VMM为代表的虚拟机可提供启动速度快、资源消耗低的虚拟机用于Serverless容器业务。但QEMU传统VMM的高级特性支持例如热插拔仍有待社区各方力量提供补齐。

资源动态扩缩

功能需求：虚机资源扩缩、安全容器的资源扩缩

1. cpu热插拔
2. 内存热插拔
3. pci设备热插拔

启动优化加速

性能需求

1. 降低初始化时间，使用通用虚机模板
2. 虚拟设备独立初始化

项目名称	vCPU热插拔	MEM热插拔
QEMU	**支持**	**支持**
FireCracker	-	-
StratoVirt	-	-
Cloud Hypervisor	**仅热插**	**支持**

QEMU和主流RUST-VMM热插拔能力对比

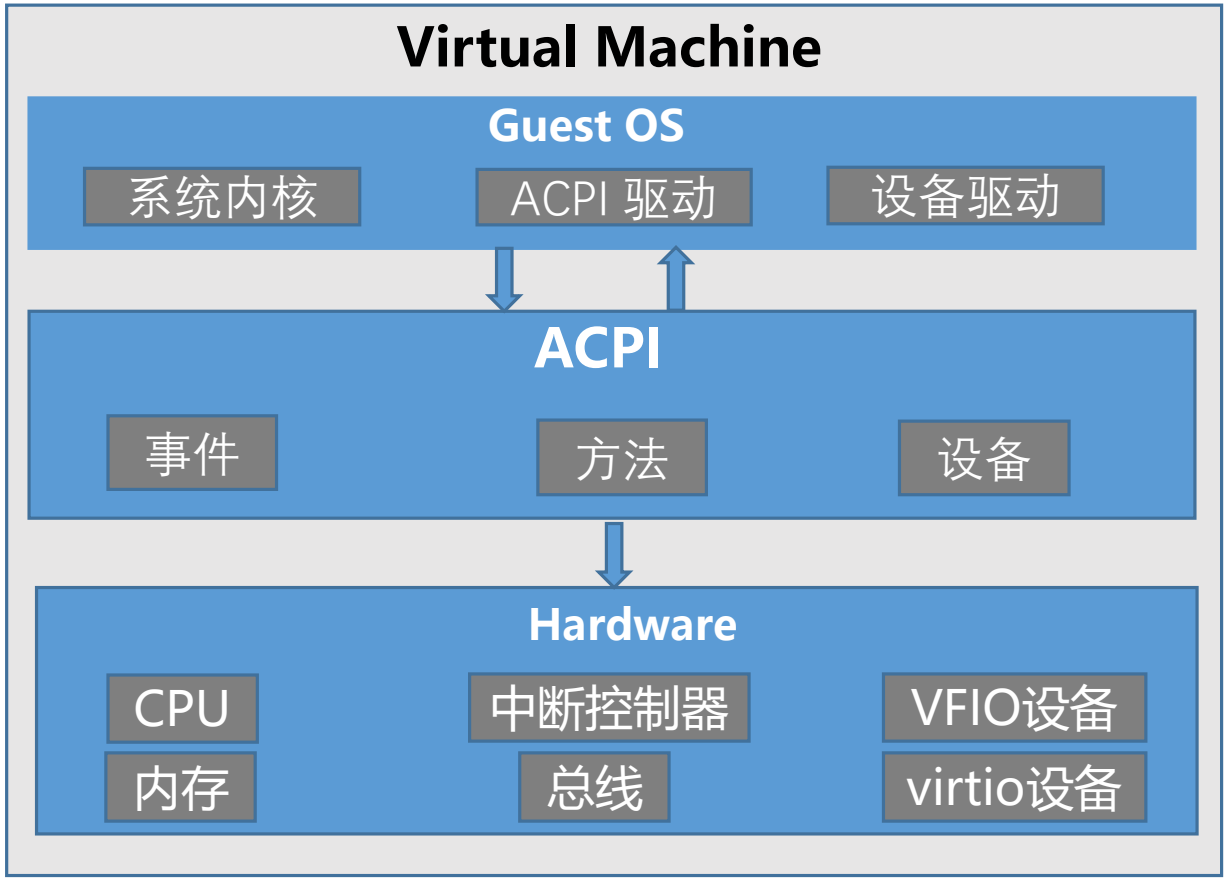
Stratovirt现状

- **内存热插拔**
  - √支持balloon方式调整内存
  - ×不支持新增加内存
- **vCPU热插拔**
  - ×不支持vCPU热插拔
- **PCI热插拔**
  - √standVM支持，基于virtio协议

ACPI 热插拔方案设计思路

ACPI规范是一种标准，可以理解为操作系统和硬件之间的接口层，用于计算机系统中的电源管理、硬件配置和设备控制。生成ACPI规范的事件通知可以对运行中的系统来插入或移除硬件。

- **ACPI事件：**包含固定事件和通用事件，固定事件在ACPI规范中已经预先定义。通用事件没有在ACPI规范中预定义，可以设置自定义事件。
- **ACPI方法：**事件的处理方法，通过方法可以调用其它方法也可调用ACPI中登记的硬件设备。
- **ACPI设备：**登记系统中的硬件设备信息。



基于ACPI Generic Event Deivce(GED)的设备热插拔

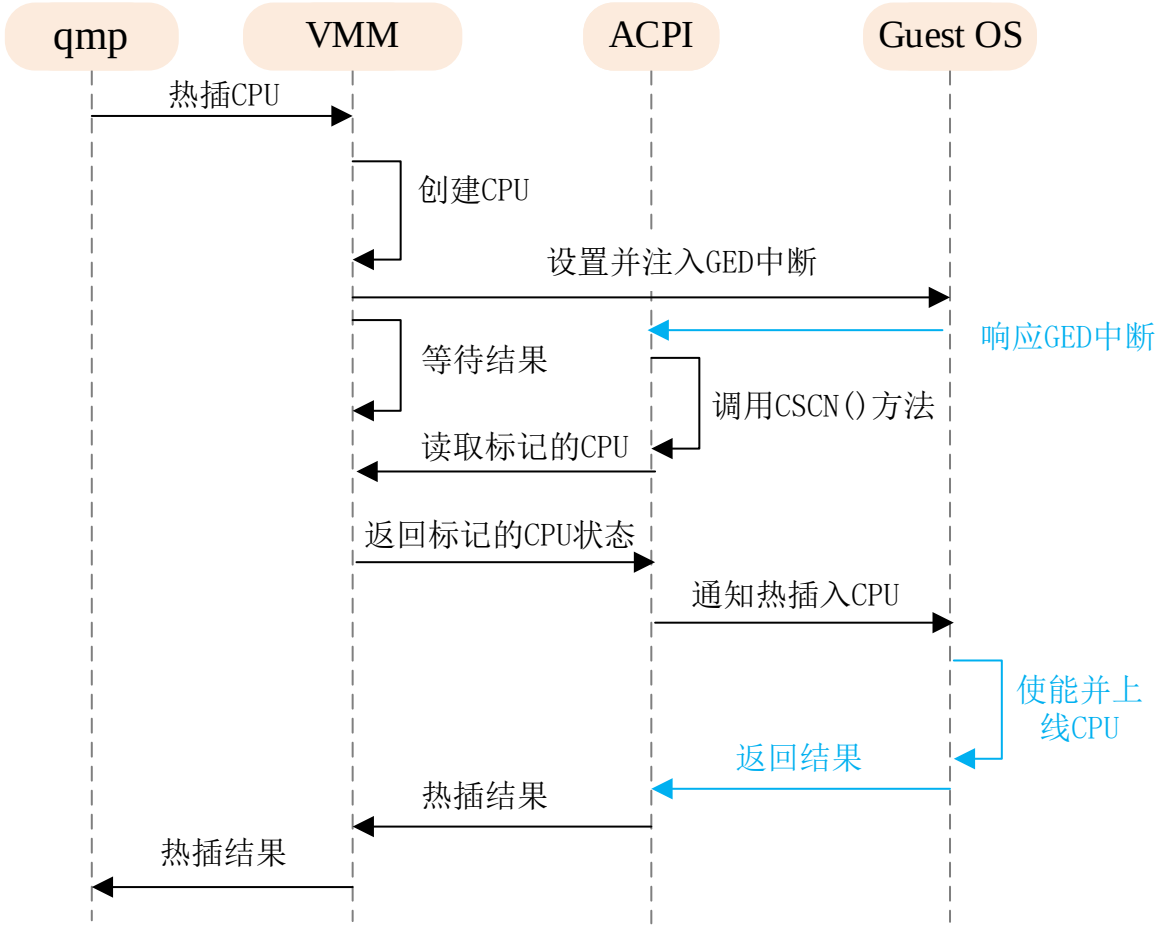
1.GED是ACPI通用事件编程模型，自定义设备热插拔事件

2. 自定义热插拔事件的处理方法，完成硬件的插入拔出

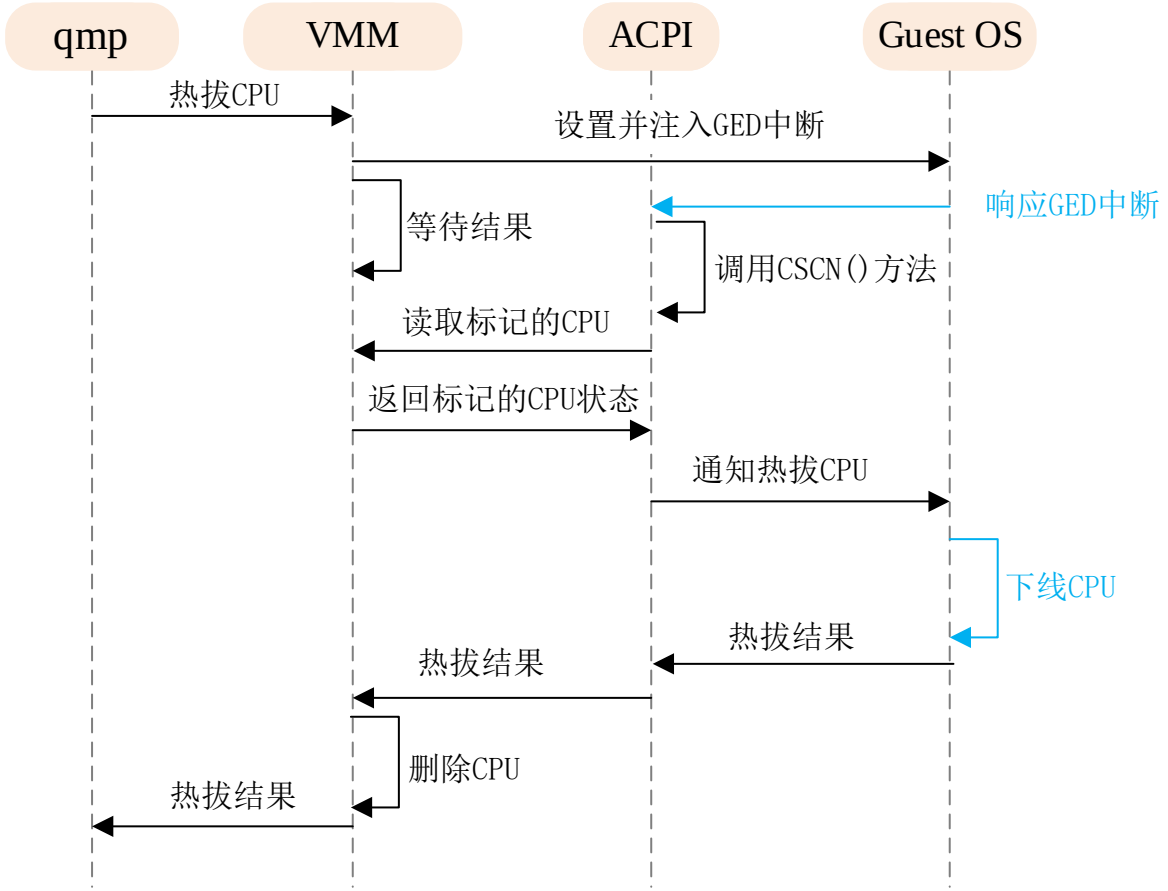
3. VMM负责硬件的创建/删除，向虚拟机注入中断，虚拟机操作系统触发ACPI GED事件

CPU热插拔——基于ACPI方案

CPU热插流程



CPU热拔流程

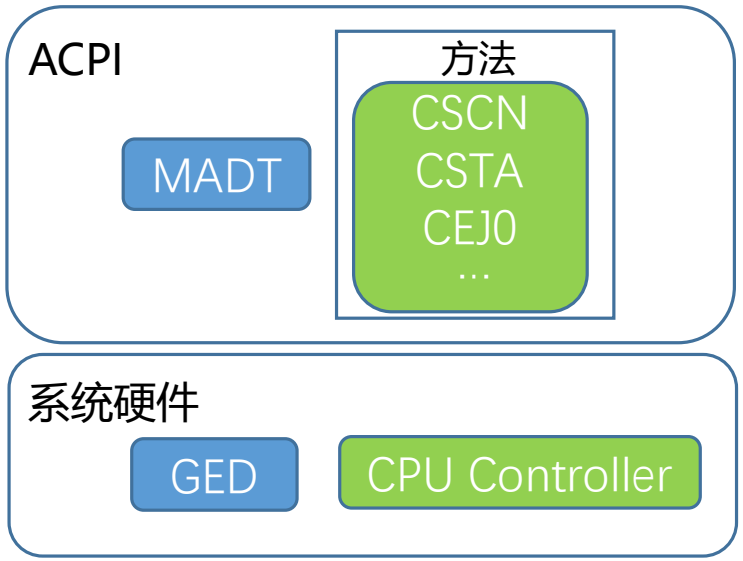


注：蓝色标记部分为操作系统自带功能

CPU热插拔——StratoVirt实现

StratoVirt 基础上修改的部分

- GED: 增加了CPU热插拔的事件
- CPU Controller: 负责可热插拔CPU的生命周期管理，CPU热插拔事件的处理。
- MADT: 增加非启动CPU的描述
- 方法:为CPU设备的_STA、_EJ0、_OST接口实现CSTA、CEJ0、COST方法，为CPU Controller实现CSCN、CTFY方法。
- QMP：增加了CPU热插拔命令



CPU Controller Methods

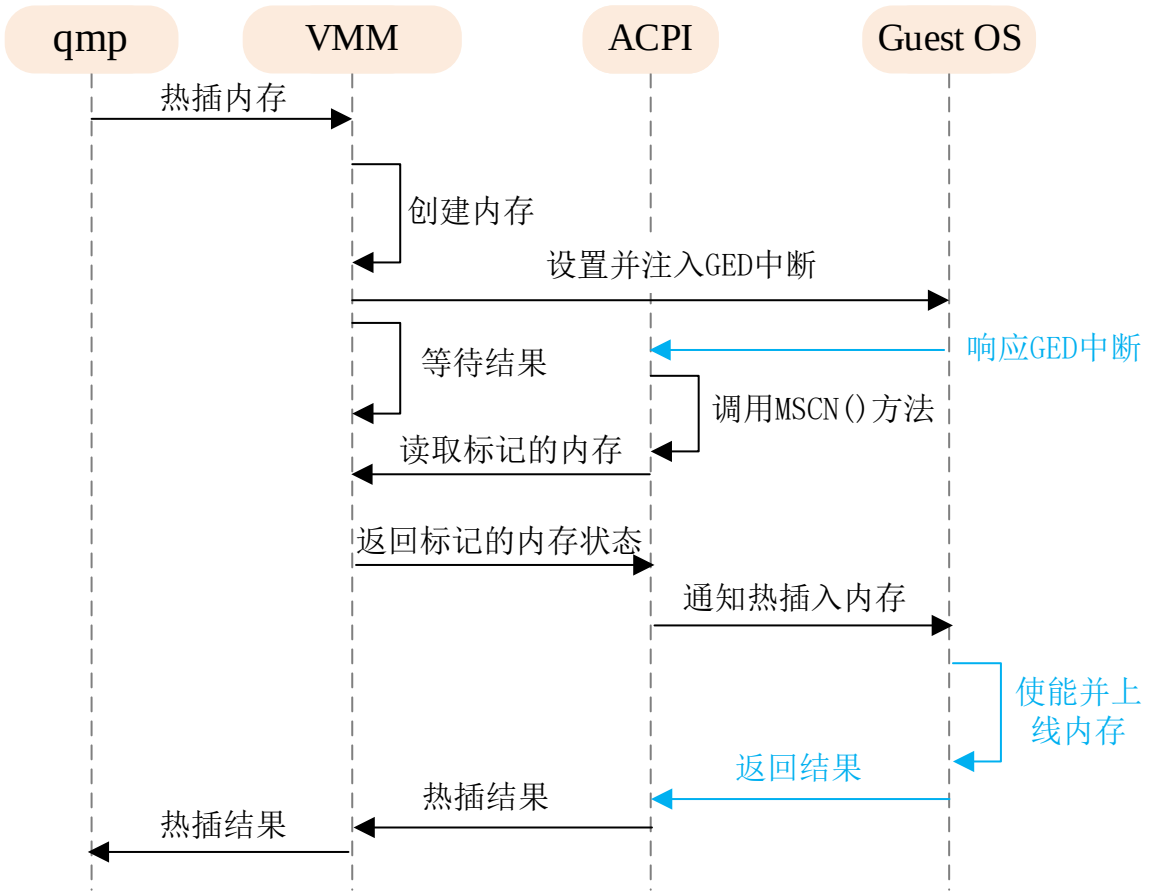
方法	作用	调用时机	方法执行流程
CSCN	流程总控制	ACPI 开始执行CPU热插拔	1.MMIO读操作：获得要插拔的CPU ID 2.MMIO读操作：获得操作类型（热插或热拔） 3.ACPI Notify操作：调用CTFY，提供CPU ID和热插/热拔标记参数 4.MMIO写操作：通知VMM恢复CPU的标记状态
CTFY	发送CPU设备通知	CSCN方法第三步	1.根据参数获得对应ID的CPU设备对象 2.根据参数通知CPU设备对象执行热插或热拔操作

CPU设备的方法接口与方法实现

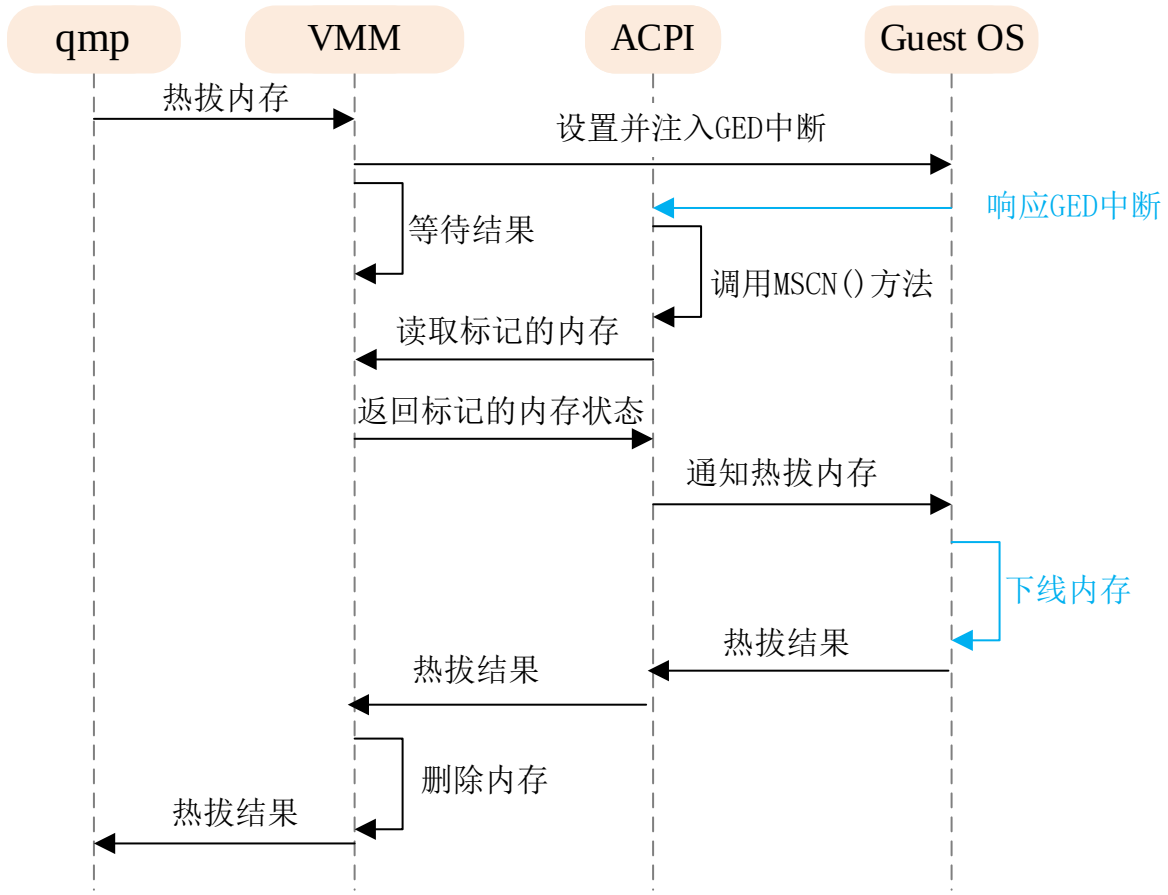
实现方法	接口	作用	调用时机	方法执行流程
CSTA	_STA	查询该CPU设备是否启用	Guest OS在CPU初始化、插入或拔出时调用检查设备状态	1. MMIO写操作：根据ID选中要读取状态的CPU 2. MMIO读操作：获得对应CPU是否启动 3. 如果CPU启动则返回0xf，否则返回0x0
CEJ0	_EJ0	设备拔出通知	Guest OS在CPU拔出完成后调用	1. MMIO写操作：Guest OS拔出已经完成，由VMM完成CPU设备删除
COST	_OST	执行结果通知	Guest OS对CPU热插拔执行结束后	1.MMIO写操作：通知执行结束码 2.（执行失败）MMIO写操作：通知执行错误日志

内存热插拔——基于ACPI方案

内存热插流程



内存热拔流程

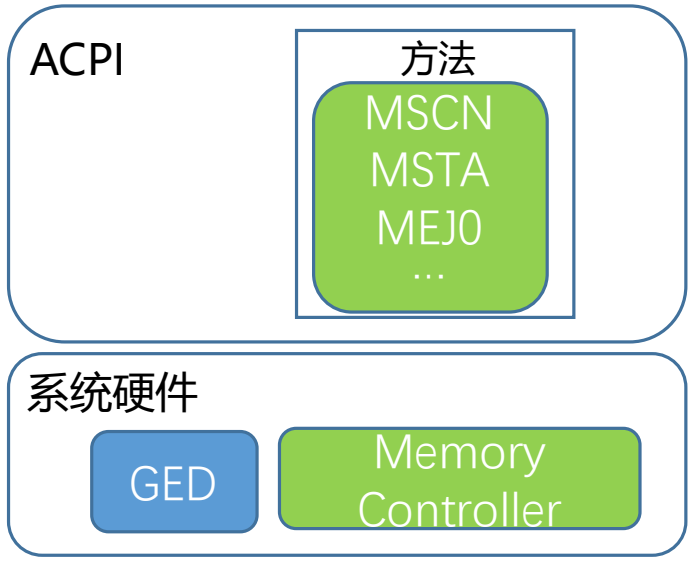


注：蓝色标记部分为操作系统自带功能

内存热插拔——StratoVirt实现

StratoVirt 基础上修改的部分

- GED: 增加了内存热插拔的事件
- Memory Controller: 负责可热插拔内存的生命周期管理，内存热插拔事件的处理。
- 方法:为内存设备的_STA、_EJ0、_OST接口实现MSTA、MEJ0、MOST方法，为Memory Controller实现MSCN、MTFY方法。
- QMP：增加了内存热插拔命令



Memory Controller方法

方法	作用	调用时机	方法执行流程
MSCN	流程总控制	ACPI开始执行内存热插拔	1.MMIO读操作：获得要插拔的内存设备ID 2.MMIO读操作：获得操作类型（热插或热拔） 3.ACPI Notify操作：调用MTFY，提供内存设备ID和热插/热拔标记参数 4.MMIO写操作：通知VMM恢复内存设备的标记状态
MTFY	发送内存设备通知	CSCN方法第三步	1.根据参数获得对应ID的内存设备对象 2.根据参数通知内存设备对象执行热插/热拔操作

内存设备的方法接口与方法

实现方法	方法接口	作用	调用时机	方法执行流程
MCRS	_CRS	查询内存设备的地址描述	Guest OS在内存设备初始化、插入时	1. MMIO写操作：根据ID选中要读取状态的内存设备 2. MMIO读操作：读取内存地址的相关属性 3. 返回内存地址描述
MSTA	_STA	查询该内存设备状态	Guest OS在内存设备初始化、插入或拔出时	1. MMIO写操作：根据ID选中要读取状态的内存设备 2. MMIO读操作：获得对应内存设备是否启动 3. 如果内存设备启动则返回0xf，否则返回0x0
MEJ0	_EJ0	设备拔出通知	Guest OS在内存设备成功拔出后	1. MMIO写操作：Guest OS拔出内存设备已经完成，通知VMM完成内存设备删除
MOST	_OST	执行结果通知	Guest OS对内存设备热插拔执行结束后	1.MMIO写操作：通知执行结束码 2.（执行失败）MMIO写操作：通知执行错误日志

热插拔演示—虚拟机创建

创建虚机：

启动CPU数量为1，最大CPU数量为3，

启动内存为512MB, 可热插拔内存插槽数量为3，最大内存数量为4096MB。

```
stratovirt-dev \
    -machine q35 \
    -smp 1,maxcpus=3 -m "size=512,slots=3,max_size=4096" \
    -kernel std-vmlinuxz \
    -append "console=ttyS0 root=/dev/vda reboot=k panic=1 movable_node" \
    -drive file=/usr/share/edk2/ovmf/OVMF_CODE.fd,if=pflash,unit=0,readonly=true \
    -device pcie-root-port,port=0x0,addr=0x1.0x0,bus=pcie.0,id=pcie.1 \
    -drive file=openEuler-22.03-LTS-stratovirt-x86_64.img,id=rootfs,readonly=false \
    -device virtio-blk-pci,drive=rootfs,bus=pcie.1,addr=0x0.0x0,id=blk-0 \
    -qmp unix:stdvm.sock,server,nowait \
    -serial stdio
```

```
[root@StratoVirt ~]# lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         45 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  1
  On-line CPU(s) list:   0
Vendor ID:               GenuineIntel
```

虚机启动后，在线CPU数量为1

```
[root@StratoVirt ~]# ls /sys/devices/system/memory/
auto_online_blocks  memory0  memory2  power  uevent
block_size_bytes    memory1  memory3  probe
[root@StratoVirt ~]# free -h
               total        used        free      shared  buff/cache   available
Mem:           466Mi        26Mi       395Mi       1.0Mi        44Mi       426Mi
Swap:             0B          0B          0B
```

虚机启动后，内存块数量4(每块大小128MB)

热插拔演示—CPU热插拔

CPU热插

```
[root@localhost img]# ncat -U stdvm.sock
{"QMP":{"version":{"qemu":{"micro":1,"minor":0,"major":5},"package":"StratoVirt-2.2.0"},
{"execute": "device_add","arguments": { "id": "cpu-1", "driver": "generic_x86_cpu"}}
{"return":{}}
```

热插cpu-1, 返回插入成功

```
[root@StratoVirt ~]# CPU1 has been hot-added
SMP alternatives: switching to SMP code
x86: Booting SMP configuration:
smpboot: Booting Node 0 Processor 1 APIC 0x1
kvm-clock: cpu 1, msr 27401041, secondary cpu clock
kvm-guest: KVM setup async PF for cpu 1
kvm-guest: stealtime: cpu 1, msr 3d2b4080
Will online and init hotplugged CPU: 1
```

虚拟机 cpu1自动上线

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         45 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  2
  On-line CPU(s) list:   0,1
```

虚拟机 在线CPU数量为2

CPU热拔

```
{"execute": "device_del", "arguments": { "id": "cpu-1"}}
{"return":{}}
```

热拔cpu-1, 返回热拔成功

```
[root@StratoVirt ~]# kvm-guest: Unregister pv shared memory for cpu 1
smpboot: CPU 1 is now offline
```

虚拟机cpu1自动下线

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         45 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  1
  On-line CPU(s) list:   0
```

客户机 在线CPU数量为1

热插拔演示—内存热插拔

内存热插

```
[root@localhost img]# ncat -U stdvm.sock
{"QMP":{"version":{"qemu":{"micro":1,"minor":0,"major":5},"package":"StratoVirt-2.2.0"},"capabilities
{"execute": "device_add", "arguments": { "id": "dimm-0", "driver": "pc_dimm", "mem_size": "128"}}
{"return":{}}
```

热插dimm-0,128M内存, 返回插入成功

```
[root@StratoVirt ~]# Fallback order for Node 0: 0
Built 1 zonelists, mobility grouping on.  Total pages: 151228
Policy zone: DMA32
```

虚拟机 dimm-0内存自动上线

```
[root@StratoVirt ~]# ls /sys/devices/system/memory/
auto_online_blocks  memory0  memory2  memory32  probe
block_size_bytes    memory1  memory3  power     uevent
[root@StratoVirt ~]# free -h
              total        used        free      shared  buff/cache   available
Mem:          594Mi        28Mi       520Mi       1.0Mi        45Mi       551Mi
Swap:            0B          0B          0B
```

虚拟机 新增一个内存块，内存增加128MB

内存热拔

```
{"execute": "device_del", "arguments": { "id": "dimm-0"}}
{"return":{}}
```

热拔dimm-0,128MB内存,返回拔出成功

```
[root@StratoVirt ~]# Offlined Pages 32768
Fallback order for Node 0: 0
Built 1 zonelists, mobility grouping on.  Total pages: 118456
Policy zone: DMA32
```

虚拟机 dimm-0内存自动下线

```
[root@StratoVirt ~]# ls /sys/devices/system/memory/
auto_online_blocks  memory0  memory2  power  uevent
block_size_bytes    memory1  memory3  probe
[root@StratoVirt ~]# free -h
              total        used        free      shared  buff/cache   available
Mem:          466Mi        26Mi       395Mi       1.0Mi        45Mi       426Mi
Swap:            0B          0B          0B
```

虚拟机 内存减少128MB

后续推进计划

代码开源

1. 根据openEuler社区wiki代码规范自查
2. 撰写测试用例
3. 整理代码并提交PR

现网应用

1. 结合kata-container组成安全容器在现网验证业务承载
2. 在现有私有云进行嵌套虚拟化的验证

技术探索

1. 电信NFV网元场景使用轻量级虚拟化
2. NFV容器编排层加速技术结合安全容器实践

目录

一 热插拔技术研发实践

二 热插拔技术未来展望

ACPI功能补全· ARM指令集下的热插拔功能实现

Arm社区近年来也非常关注vCPU，MEM的热插拔功能，其中内存热插拔实现较早（2016-11），而CPU热插拔的实现面临较多挑战，近年正在攻克中

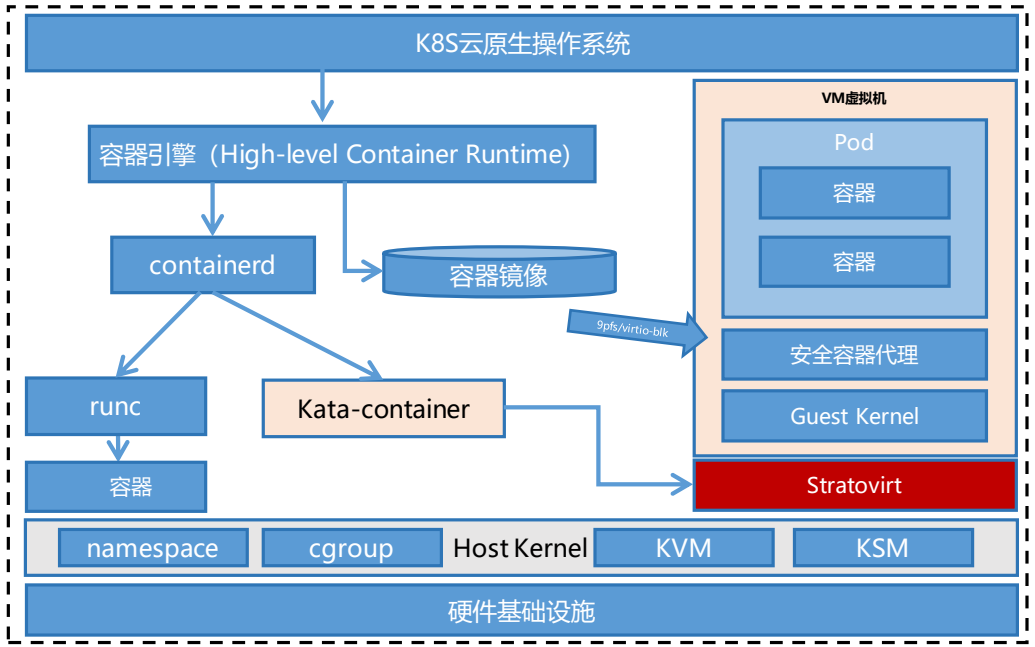
- **CPU热插拔：**必须依赖ACPI协议，目前生态仍未完善
- **内存热插拔：**基于ACPI协议的内存热插拔已有开源rust-vmm实现

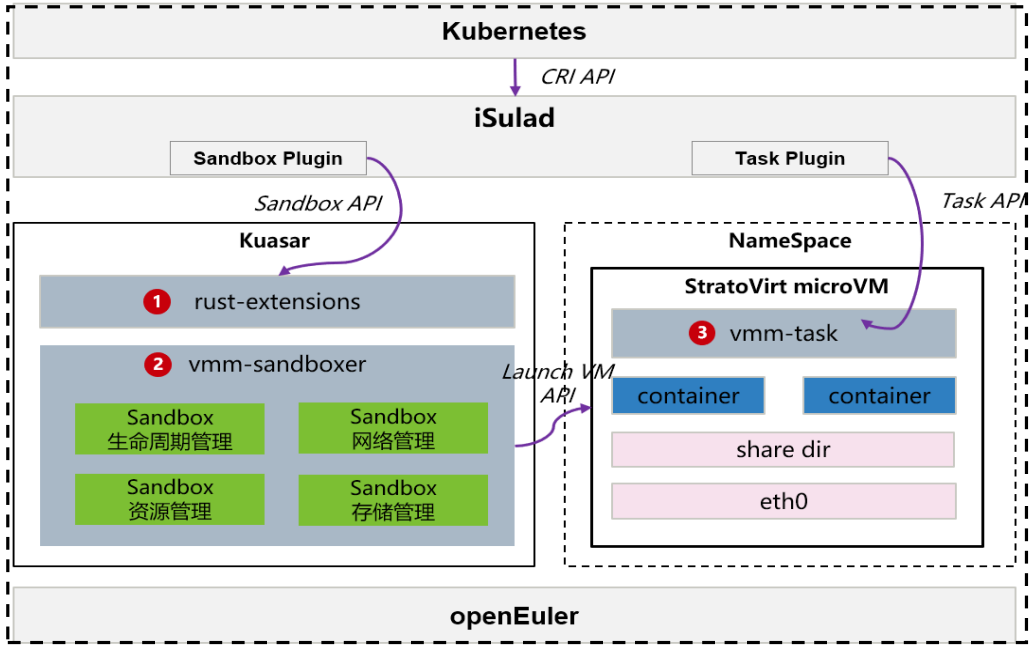
2022-07 Cloud-Hypervisor commit： 在aarch64环境基于ACPI协议实现了vCPU的热插拔实验性功能
2023-02 Linux内核RFC ACPI/arm64：添加了对虚拟 CPU 热插拔的支持

通过适配arm转译acpi协议的方式即可移植现有x86架构的acpi热插拔cpu、内存功能。

未来演进展望· Stratovirt作为一种沙盒运行时

使用MicroVM作为容器底层运行时目前有两种技术方案，Kata-container，Kuasar





Kata-container方案

containerd/isulad+kata-container+Stratovirt

√ Kata 2.1沙盒适配Stratovirt启动参数

Todo：Kata适配Stratovirt qmp vCPU、vMEM扩缩参数

Kuasar方案

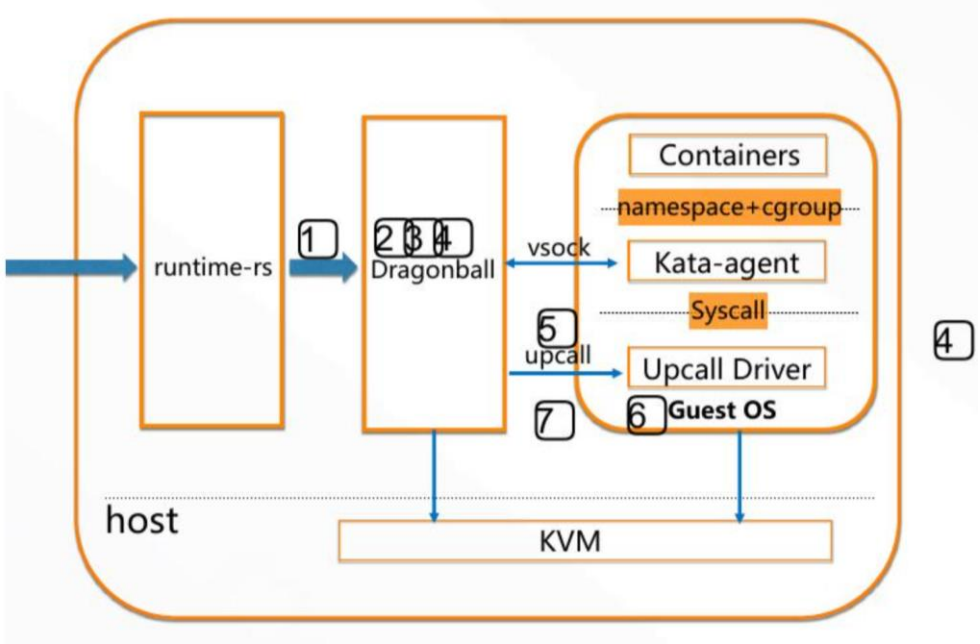
isulad+kuasar+Sandbox API+Stratovirt

√ 使用最新沙盒API，支持MicroVM WASM混部

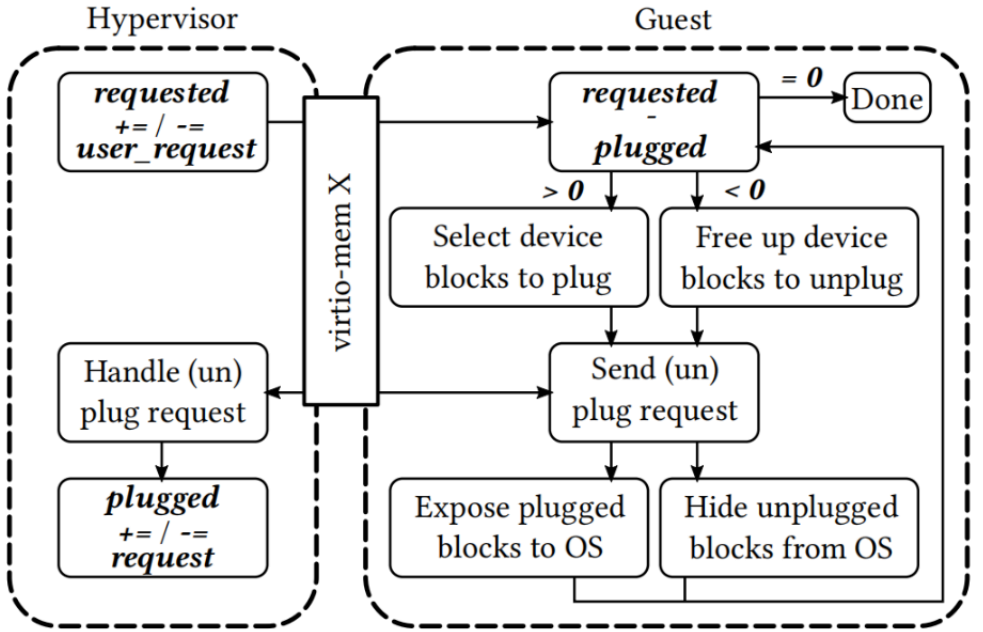
Todo：Kuasar目前有待测试云原生功能和接口

热插拔·跳出ACPI的框架之外

ACPI协议受制于x86原生特性并且具有较大冗余，RUST-VMM业界也积极尝试跳过ACPI来实现热插拔的功能，提供更好的灵活性、颗粒度和更低的损耗



- **Upcall的cpu热插拔方案**：通过对vmm和Guest kernel的定制化，降低ACPI协议链路的损耗同时增强os可控性。目前为dragonball实验特性，已开源dbs-upcall等crates。



- **Virtio-mem方案**：打破内存最小块限制，提高热拔成功率，解决了NUMA无法感知等问题。Virtio-mem是新型通用架构内存热插拔方案，在Servelss时代或更具优势。目前在Cloud Hypervisor已支持该方案。