Luoye's blog - Up Step by Step

路漫漫其修远兮...

7月 12 2014

KVM Run Process之Qemu核心流程

在前文中，介绍了在KVM环境下使用Qemu成功创建并运行了虚拟机，而这一切的背后是什么样的运作机制呢？本文主要介绍在整个创建和运行过程中Qemu与KVM里两者的核心运行流程。

Qemu核心流程

阶段一：参数解析

这里使用qemu版本为qemu-kvm-1.2.0,使用Qemu工具使能虚拟机运行的命令为：

```
1 $sudo /usr/local/kvm/bin/qemu-system-x86_64 -hda vdisk_linux.img -m 1024
```

这时候会启动qemu-system-x86_64应用程序，该程序入口为

```
1 int main(int argc, char **argv, char **envp)   <------file: vl.c,line: 2345
```

在main函数中第一阶段主要对命令传入的参数进行parser，包括如下几个方面：

```
QEMU_OPTION_M                    机器类型及体系架构相关
```

```
QEMU_OPTION_hda/mtdblock/pflash    存储介质相关
QEMU_OPTION_numa                   numa系统相关
QEMU_OPTION_kernel                 内核镜像相关
QEMU_OPTION_initrd                 initramdisk相关
QEMU_OPTION_append                 启动参数相关
QEMU_OPTION_net/netdev             网络相关
QEMU_OPTION_smp                    smp相关
```

阶段二：VM的创建

通过configure_accelerator()->kvm_init() file: kvm-all.c, line: 1281;
首先打开*/dev/kvm*，获得三大描述符之一kvmfd， 其次通过*KVM_GET_API_VERSION*进行版本验证，最后通过*KVM_CREATE_VM*创建了一个VM对象，返回了三大描述符之二: VM描述符/vmfd。

```
1 s->fd = qemu_open("/dev/kvm", O_RDWR);        kvm_init()/line: 1309
2 ret = kvm_ioctl(s, KVM_GET_API_VERSION, 0);   kmv_init()/line: 1316
3 s->vmfd = kvm_ioctl(s, KVM_CREATE_VM, 0);     kvm_init()/line: 1339
```

阶段三：VM的初始化

通过加载命令的参数的解析和相关系统的初始化，找到对应的machine类型进行第三阶段的初始化：

```
1 machine->init(ram_size, boot_devices, kernel_filename, kernel_cmdline, initrd_filename, cpu_model);
2 file: vl.c, line: 3651
```

其中的参数包括从命令行传入解析后得到的，ram的大小，内核镜像文件名，内核启动参数，initramdisk文件名，cpu模式等。
我们使用系统默认类型的machine，则init函数为pc_init_pci(),通过一系列的调用：

```
pc_init_pci()   file: pc_piix.c, line: 294
    --->pc_init1()    file: pc_piix.c, line: 123
        --->pc_cpus_init()  file: pc.c, line: 941
```

在命令行启动时配置的smp参数在这里启作用了，qemu根据配置的cpu个数，进行n次的cpu初始化，相当于n个核的执行体。

```
1 void pc_cpus_init(const char *cpu_model)
```

```
2 {
3     int i;
4     /* init CPUs */
5     for(i = 0; i < smp_cpus; i++) {
6         pc_new_cpu(cpu_model);
7     }
8 }
```

继续cpu的初始化：

```
pc_new_cpu()    file: hw/pc.c, line: 915
    --->cpu_x86_init()    file: target-i386/helper.c, line: 1150
        --->x86_cpu_realize()    file: target-i386/cpu.c, line: 1767
            --->qemu_init_vcpu()    file: cpus.c, line: 1039
                --->qemu_kvm_start_vcpu()    file: cpus.c, line: 1011
```

qemu_kvm_start_vcpu是一个比较重要的函数，我们在这里可以看到VM真正的执行体是什么。

阶段四：VM RUN

```
1  static void qemu_kvm_start_vcpu(CPUArchState *env) <--------file: cpus.c, line: 1011
2  {
3      CPUState *cpu = ENV_GET_CPU(env);
4      ......
5      qemu_cond_init(env->halt_cond);
6      qemu_thread_create(cpu->thread, qemu_kvm_cpu_thread_fn, env, QEMU_THREAD_JOINABLE);
7      ......
8  }
9
10
11 void qemu_thread_create(QemuThread *thread,
12                          void *(*start_routine)(void*),
13                          void *arg, int mode)    <--------file: qemu-thread-posix.c, line: 118
14 {
15     ......
16     err = pthread_attr_init(&attr);
17     ......
18         err = pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
19         ......
20     pthread_sigmask(SIG_SETMASK, &set, &oldset);
21     ......
```

```
22     pthread_create(&thread->thread, &attr, start_routine, arg);
23     ......
24     pthread_attr_destroy(&attr);
25 }
```

可以看到VM真正的执行体是QEMU进程创建的一系列POSIX线程，而线程执行函数为qemu_kvm_cpu_thread_fn。
kvm_init_vcpu()通过*KVM_CREATE_VCPU*创建了三大描述符之三：vcpu描述符/vcpufd。
并进入了while(1)的循环循环，反复调用kvm_cpu_exec()。

```
1  static void *qemu_kvm_cpu_thread_fn(void *arg)   file: cpus.c, line: 732
2  {
3          ......
4      r = kvm_init_vcpu(env);       <--------file: kvm-all.c, line: 213
5      ......
6      qemu_kvm_init_cpu_signals(env);
7      /* signal CPU creation */
8      env->created = 1;
9      qemu_cond_signal(&qemu_cpu_cond);
10     while (1) {
11         if (cpu_can_run(env)) {
12             r = kvm_cpu_exec(env);      <--------file: kvm-all.c, line: 1550
13             if (r == EXCP_DEBUG) {
14                 cpu_handle_guest_debug(env);
15             }
16         }
17         qemu_kvm_wait_io_event(env);
18     }
19     return NULL;
20 }
```

我们可以看到kvm_cpu_exec()中又是一个do()while(ret == 0)的循环体，该循环体中主要通过*KVM_RUN*启动VM的运行，从此处进入了kvm的内核处理阶段，并等待返回结果，同时根据返回的原因进行相关的处理，最后将处理结果返回。因为整个执行体在上述函数中也是在循环中，所以后续又会进入到该函数的处理中，而整个VM的cpu的处理就是在这个循环中不断的进行。

```
1  int kvm_cpu_exec(CPUArchState *env)
2  {
3      do {
4          run_ret = kvm_vcpu_ioctl(env, KVM_RUN, 0);
```

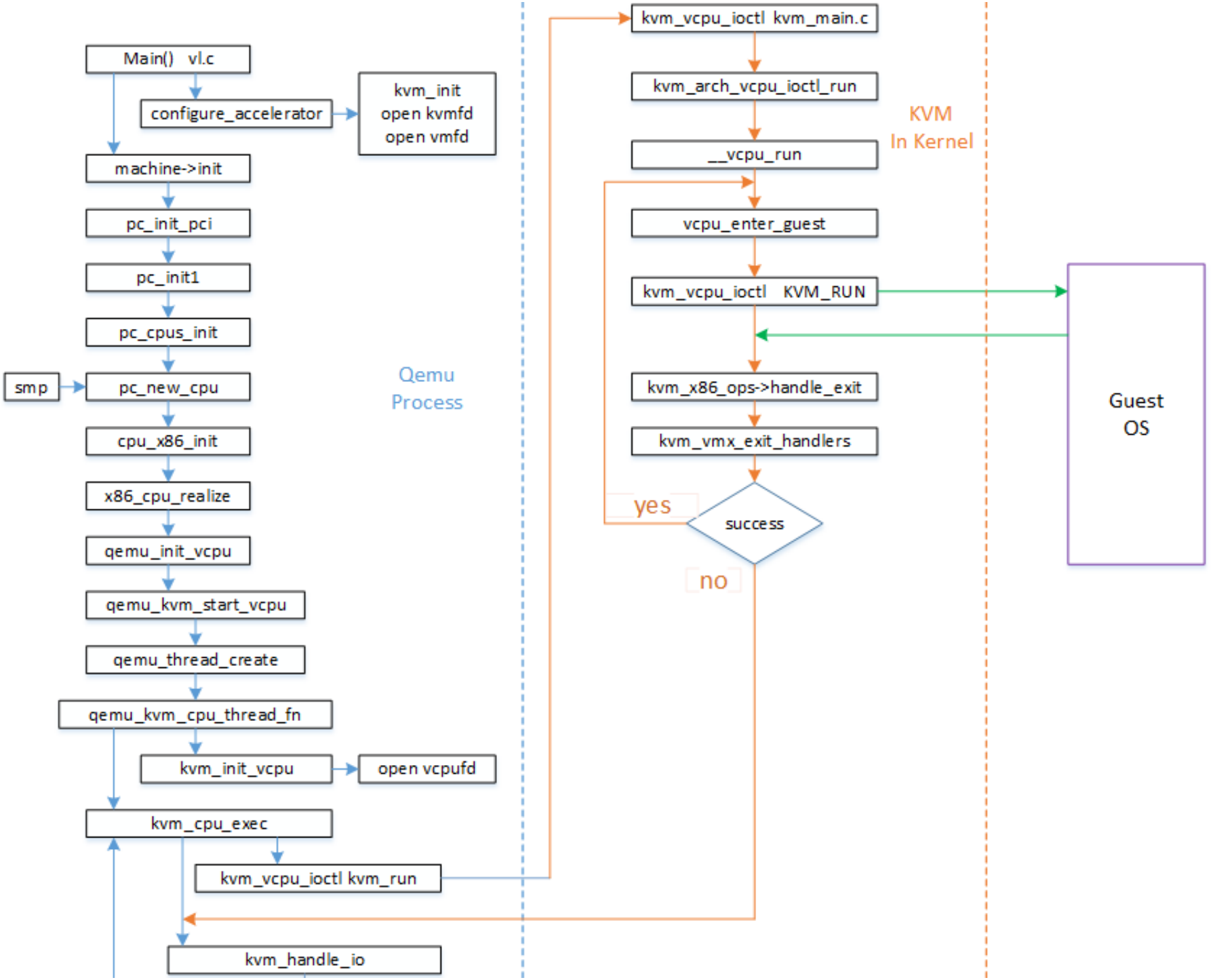
```
5           switch (run->exit_reason) {
6           case KVM_EXIT_IO:
7                   kvm_handle_io();
8                           ......
9           case KVM_EXIT_MMIO:
10                  cpu_physical_memory_rw();
11                          ......
12          case KVM_EXIT_IRQ_WINDOW_OPEN:
13                  ret = EXCP_INTERRUPT;
14              ......
15          case KVM_EXIT_SHUTDOWN:
16                  ret = EXCP_INTERRUPT;
17              ......
18          case KVM_EXIT_UNKNOWN:
19                  ret = -1
20              ......
21          case KVM_EXIT_INTERNAL_ERROR:
22                  ret = kvm_handle_internal_error(env, run);
23              ......
24          default:
25                  ret = kvm_arch_handle_exit(env, run);
26              ......
27          }
28      } while (ret == 0);
29      env->exit_request = 0;
30      return ret;
31  }
```

Conclusion

总结下kvm run在Qemu中的核心流程：

1. 解析参数;
2. 创建三大描述符：kvmfd/vmfd/vcpufd，及相关的初始化，为VM的运行创造必要的条件；
3. 根据cpu的配置数目，启动n个POSIX线程运行VM实体，所以vm的执行环境追根溯源是在Qemu创建的线程环境中开始的。
4. 通过*KVM_RUN*调用KVM提供的API发起KVM的启动，从这里进入到了内核空间运行，等待运行返回；
5. 重复循环进入run阶段。

附图：


Main() vl.c
configure_accelerator
kvm_init
open kvmfd
open vmfd
machine->init
pc_init_pci
pc_init1
pc_cpus_init
smp
pc_new_cpu
cpu_x86_init
x86_cpu_realize
qemu_init_vcpu
qemu_kvm_start_vcpu
qemu_thread_create
qemu_kvm_cpu_thread_fn
kvm_init_vcpu
open vcpufd
kvm_cpu_exec
kvm_vcpu_ioctl kvm_run
kvm_handle_io
Qemu
Process
kvm_vcpu_ioctl kvm_main.c
kvm_arch_vcpu_ioctl_run
__vcpu_run
vcpu_enter_guest
kvm_vcpu_ioctl KVM_RUN
kvm_x86_ops->handle_exit
kvm_vmx_exit_handlers
success
yes
no
KVM
In Kernel
Guest
OS

下一文，将介绍VM在KVM中的核心执行流程。

KVM
Virtualization KVM QEMU
上一页 下一页

分类

- KVM18

标签云

EPTVirtualization KVM QEMUvhostvirtio