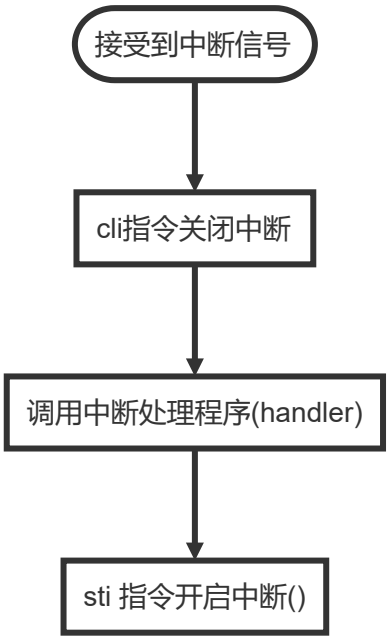
实时操作系统(Linux)抢占模型

- GP-Linux: 任务在用户态可以抢占，但是在内核态不可抢占。
- RT-Linux: 任务在用户态和内核态都可以抢占。

  注： 在RT-Linux下，所有的内核代码段几乎都是可抢占的，处理少数临界区的资源。包括中断处理程序。

RT-OS 的中断处理

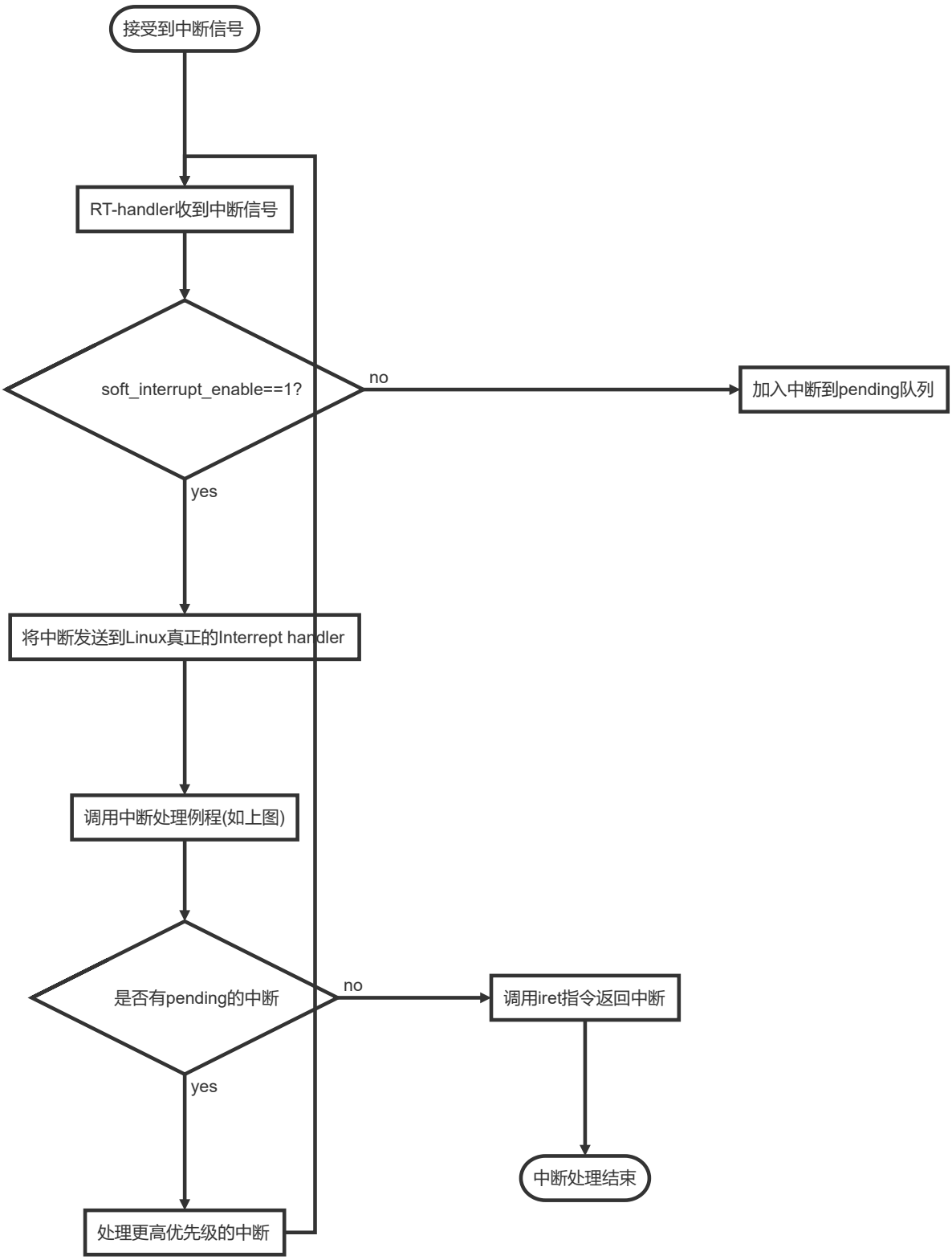
在GP-Linux的情况下，中断处理流程可以看做：



然后在RealTime的情况下，为了保证实时性，不能够真正的关闭中断。为了解决这个问题，RT-Linux的解决方式如下：

- 引入一个全局变量 int soft_interrupt_enable;
- 调用cli指令时，并不会关闭中断，而是会将soft_interrupt_enable清零。
- 调用sti指令时，会生成一个模拟的软中断通知中断的pending队列可以处理中断了。
- 中断返回的操作用一个return_from_interrupt的函数来代替原来的iret指令。

在RT-Linux的情况下，中断流程可以表示为下图：


接受到中断信号
RT-handler收到中断信号
soft_interrupt_enable==1?
no
加入中断到pending队列
yes
将中断发送到Linux真正的Interrept handler
调用中断处理例程(如上图)
是否有pending的中断
no
调用iret指令返回中断
yes
中断处理结束
处理更高优先级的中断

简单来说实时操作系统采取了一些措施保证中断是可以抢占的。具体流程解释参考：

Real-Time Linux

虚拟机中断相关

虚拟机中断处理流程

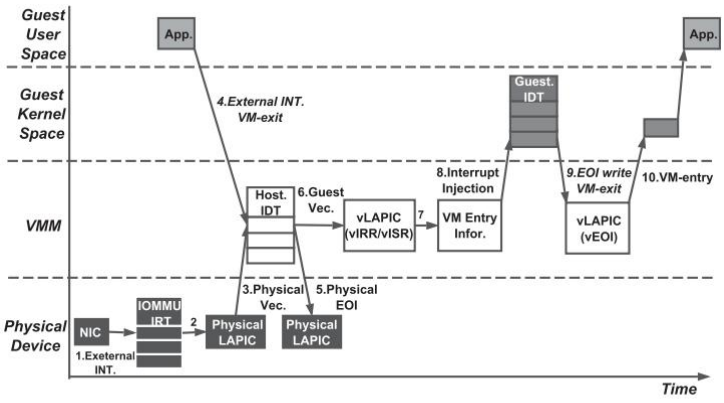


Fig. 3. Interrupt handling in x86 virtualization environment.

KVM磁盘IO性能

- 比较KVM和裸机顺序块写入时IO性能

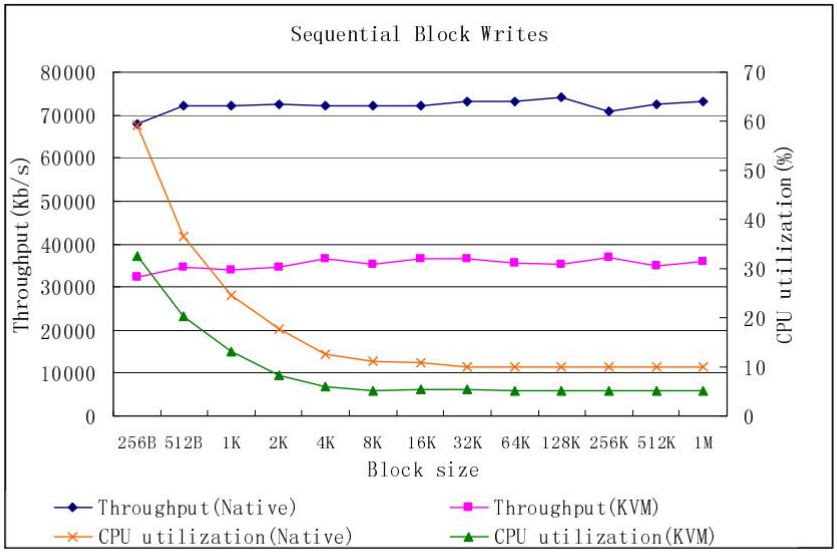


Fig. 1. Comparison of the performance between Native and KVM - Sequential block writes

- 比较KVM和裸机顺序块读入时IO性能

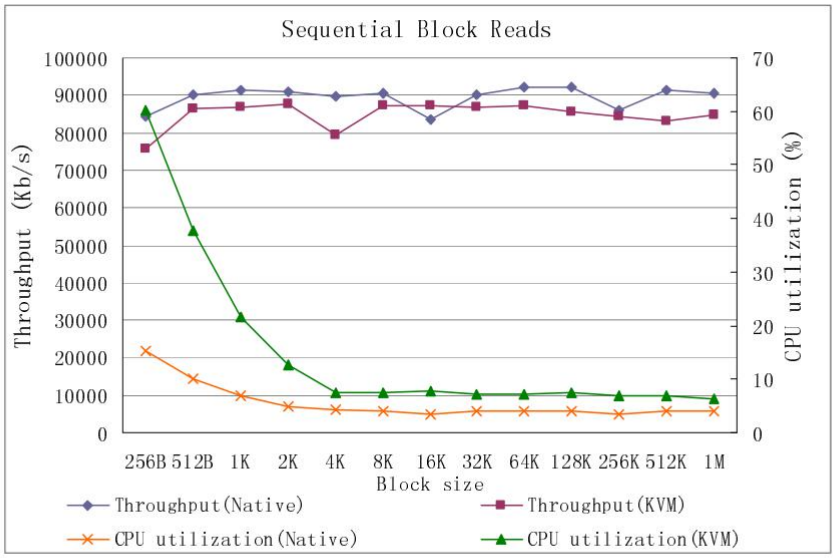


Fig. 2. Comparison of the performance between Native and KVM – Sequential block reads

- Result：KVM在顺序块写入时的吞吐量和CPU利用率只有裸机的一半；而顺序块读入时的吞吐量和CPU利用率与裸机基本相同。因此需要通过提高写磁盘吞吐量和降低读磁盘的IO开销来提高磁盘IO性能。

KVM网络虚拟化开销

- 通过Ping，测量HostA和HostB之间、HostB和GuestB之间以及HostA和GuestB之间的往返时间（RTT）。

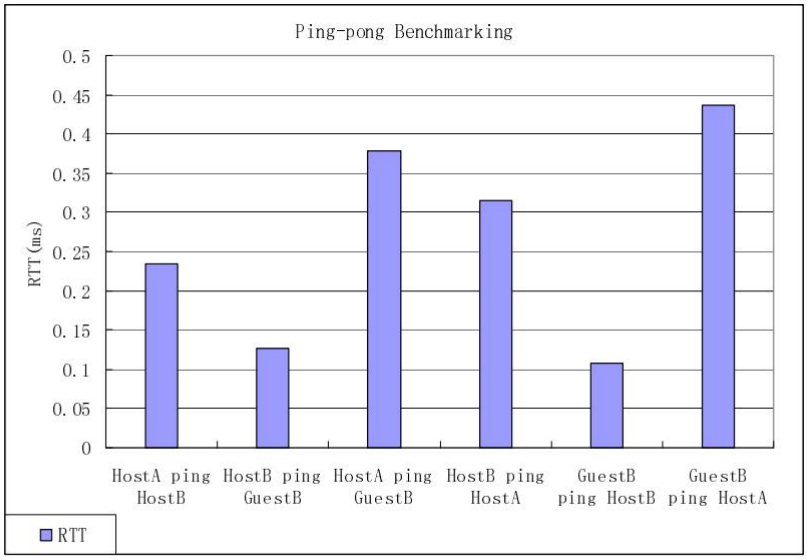


Fig. 3. Ping-pong benchmarking results

- Result：HostB和GuestB之间传输时间（虚拟化开销）占HostA和GuestB之间传输时间的33%。虚拟设备的性能对网络性能有很大影响。

耗时原因

- 多次上下文切换（2次VM-exit，2次VM-entry）
- 长调度延迟（long scheduling latency)

优化方案

- 减少上下文切换开销：

  合并代码段中连续的IO指令批量提交：Guest OS和KVM之间的切换取决于Guest OS的行为，Guest OS中的IO操作在一定程度上是集群化的。可以将连续的I/O指令合并到一个vmcall中，该vmcall将主动退出KVM。将每个指令的信息放入一个队列，包括输入/输出、端口号和值。队列的地址和长度作为vmcall的参数传递给KVM。KVM将从队列中获取每一条指令的信息，并逐一进行仿真。

- 简化Guest OS:

  删除冗余操作：Guest IO的所有IO请求都将在Host OS中重新调度，因此Guest OS中IO调度程序是冗余的，一方面Guest OS不知道物理磁盘信息，另一方面当多个Guest OS同时运行时，Host OS必须重新调度来自所有Guest OS 的IO请求。直接将IO请求交给驱动程序。

- IO直通

  基于硬件辅助的IO直通技术：ATS (Address Translation Services)、 SR-IOV (Single Root IOV)、MR-IOV (Multi-Root IOV)

参考文献：Evaluating and Optimizing I/O Virtualization in Kernel-based Virtual Machine (KVM)