Security Level:

KVM虚拟化场景下提升内核的实时性

www.huawei.com

谢祥有 <xiexiangyou@huawei.com>

Oct. 17, 2015

HUAWEI

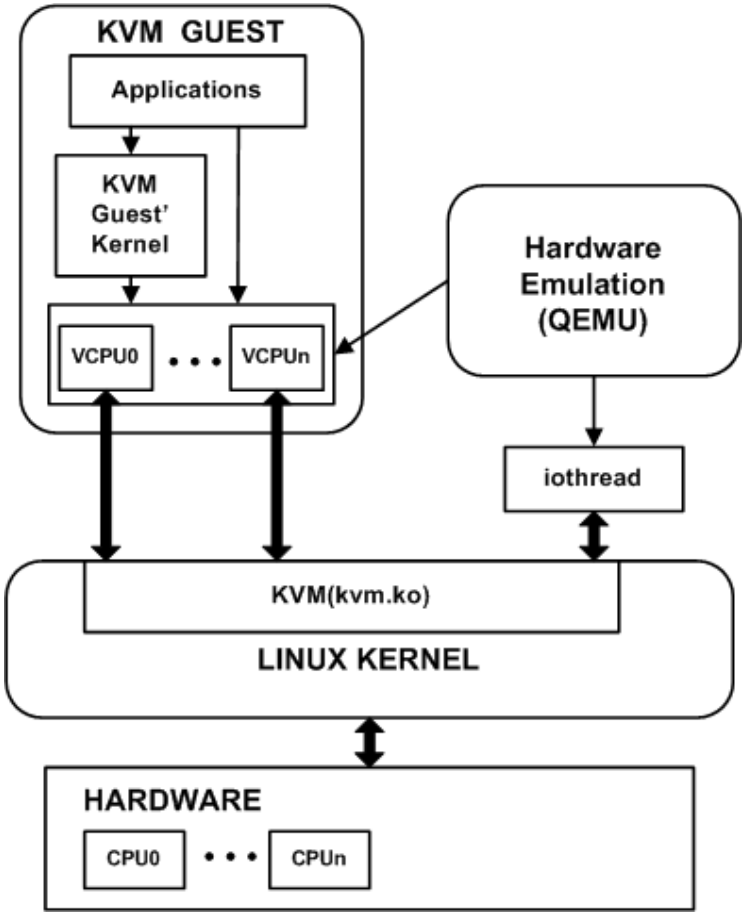
目录

- KVM虚拟化背景
- 计算密集型业务和时延敏感性业务
- 虚拟化实时性能的影响因素---内核噪音
- 设置隔离分区
- 内核噪声隔离
- 设置VCPU和IO线程的调度策略和优先级
- 关于灵活性
- 一种动态隔离的方案

KVM虚拟化背景

- KVM——Kernel-based Virtual Machine，即基于内核的虚拟机

- 虚拟机的VCPU和IO在系统看来都是线程。
- 跟其他进程一样，如果希望提升虚拟机的性能，需要提高VCPU和IO线程的实时性能。
- 而且由于虚拟化导致处理路径更长，所以更容易受到OS的调度、中断等影响。



计算密集型业务和时延敏感性业务

- 虚拟机内对性能要求较高的两类业务：
  - 计算密集型业务：对吞吐量性能有较高的要求。
  - 时延敏感型业务：对中断响应时延有较高的要求。
- 对于计算密集型的业务，需要对CPU尽可能的独占。
- 对于时延敏感型业务，需要对中断快速响应，并且快速调度。

虚拟化实时性能的影响因素---内核噪音

- 内核噪音：

（1）TICK：时钟中断

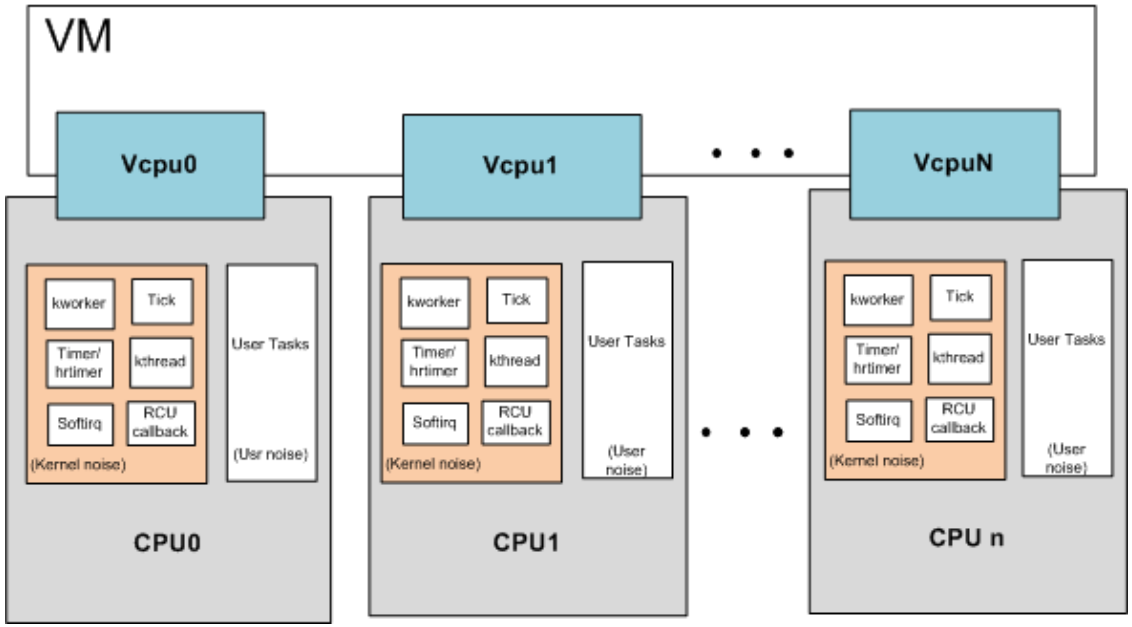
（2）I/O设备中断：上半部/下半部

（3）ksoftirqd：

- TIMER/HRTIMER
- TASKLET
- NET_TX/NET_RX
- BLOCK/BLOCK_IOPOLL
- SCHED
- RCU

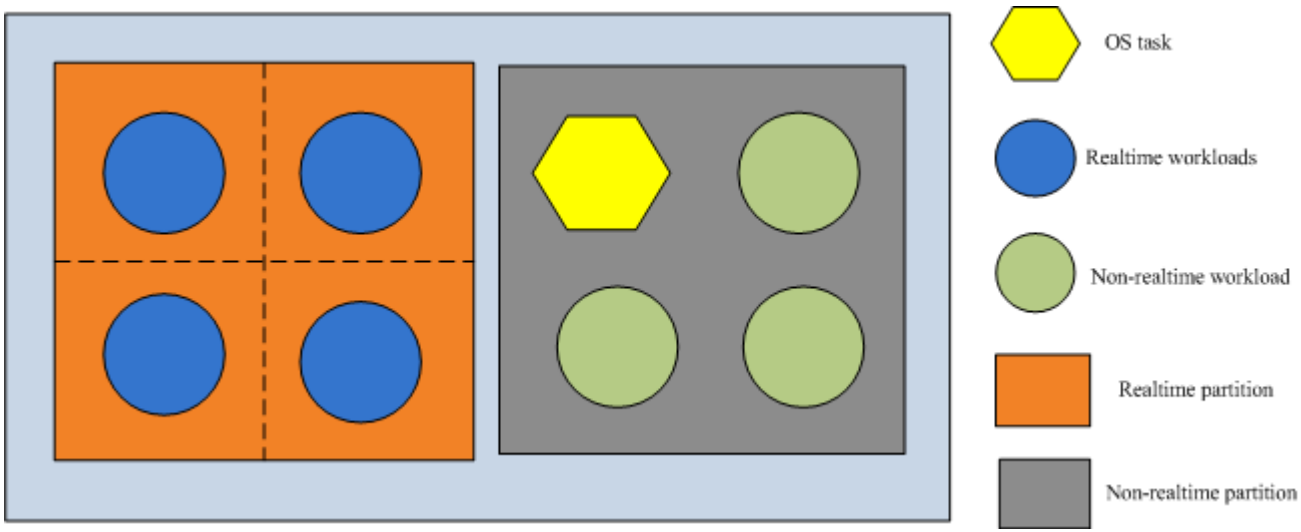
（3）kworker：执行 workqueue请求

（4）其他kthread



设置隔离分区

- **隔离分区：将系统划分成两个区域**
- 实时分区：运行实时任务
- 非实时分区：系统管理、运行非实时任务



- **现有的隔离机制**
- 基于Cgroup的隔离机制：创建不同的Cgroup组，实现进程间隔离。
- 通过系统启动参数“*isolcpus*”设置Isolate CPUs：禁止负载均衡，阻止将用户态进程及系统守护进程调度到隔离CPU上，除非显式绑定到该CPU上。

内核噪声隔离---TICK

- **TICK相当于系统的心跳，提供如下功能：**
- 更新系统状态；
- 触发到期定时器处理；
- 触发调度；
- 延时任务处理
- **TICK 隔离，即配置tickless：**
- 内核支持dyntick-idle和adaptive-tick两种模式。
- dyntick-idle目的是让idle的cpu不接收时钟中断； adaptive-tick目的是让只有一个任务运行的cpu不接收时钟中断。 dyntick-idle默认生效，而adaptive-tick需要通过“nohz_full=”的启动参数配置。
- 很多场景下不允许idle CPUs进入dyntick-idle和adaptive-tick模式，比如当CPU处于RCU callbacks pending，所以需要在启动参数里配置“rcu_nocbs=”，此时RCU callback处理将被卸载到“rcuo” kthreads中处理。
- 针对KVM Guest，增加了guest类型的RCU extended quiescent state。在Guest模式下nohz full可以生效。

内核噪声隔离—ksoftirqd

- **ksoftirqd 隔离**

- 设置中断亲和性（/proc/irq/X/smp_affinity或者irqbalance cpu黑名单），可以隔离NET_TX_SOFTIRQ/NET_RX_SOFTIRQ、BLOCK_SOFTIRQ、 TASKLET_SOFTIRQ类型的软中断。

- RCU_SOFTIRQ：配置rcu_nocbs/NOHZ_FULL, 并将rcuo kthreads绑定到非隔离CPU上。

- TIMER_SOFTIRQ: 因为KVM会为每一个VCPU提供模拟Lapic timer，所以难以避免在对应运行的cpu上设置hrtimer。除非系统硬件支持APICv/Post-Interrupt，这样可以将hrtimer设置到其他CPU上，发送IPI给对应的VCPU也不会造成VM-EXIT。

- SCHED_SOFTIRQ：尽量减少在隔离的CPU上运行Kthread，避免调度器为唤醒该线程而发送IPI。同时配置“nohz_full=”也有助于减少SCHED_SOFTIRQ。

内核噪声隔离—Kworker

- **Kworker隔离**

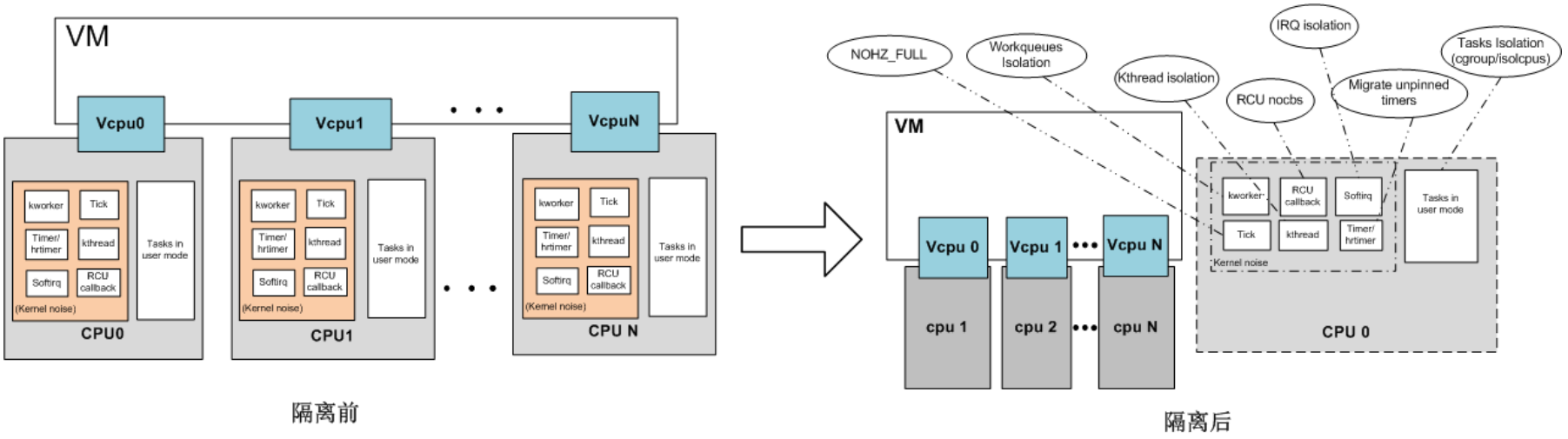
- 使用/sys/devices/virtual/workqueue/*/cpumask 的sysfs文件进行CPU亲和性设置，但前提是对应的workqueue在创建时配置了WQ_SYSFS 属性。

- 可以将所有!WQ_SYSFS的unbound workqueue添加到同一个管理链表中，同时创建一个sysfs文件cpumask，通过设置cpumask，调整unbound类型workqueue的cpu亲和性。

设置VCPU和IO线程的调度策略和优先级

- 可以将VCPU和IO线程分别绑定到对应CPU上，同时设置其调度策略及优先级。
- 设置成SCHED_FIFO线程，但是要求该线程运行的CPU上尽量不要运行其他任务。

隔离效果


VM
Vcpu0
Vcpu1
VcpuN
kworker
Tick
Timer/ hrtimer
kthread
Softirq
RCU callback
(Kernel noise)
Tasks in user mode
CPU0
CPU1
CPU N
隔离前
NOHZ_FULL
Workqueues Isolation
Kthread isolation
RCU nocbs
IRQ isolation
Migrate unpinned timers
Tasks Isolation (cgroup/isolcpus)
Vcpu 0
Vcpu 1
Vcpu N
cpu 1
cpu 2
cpu N
CPU 0
隔离后

关于灵活性

- 灵活性问题:
- Isolcpus需要在启动时就设置好，如果要调整就需要重启系统
- cgroup隔离，建立隔离分区后，如果希望新增的进程同样被隔离，要求新增进程是隔离进程的子进程，继承隔离特性。

一种动态隔离的方案

- 大型的数据中心：成百上千台虚拟机，有些需要实时，有些不需要。如何配置哪些CPU是隔离的CPU，并把该CPU分配给有实时需求的虚拟机？

  是否可以适配业务场景，比如说：实时虚拟机启动时，自动进行隔离调整？

- 所以可以提供一种动态隔离的方案，可以通过系统接口方式设置隔离。而不需要重启系统。

一种动态隔离的方案

- 设置隔离核

  echo 0 > /sys/device/system/cpuN/online

  echo 1 > /sys/device/system/cpuN/isolated

  echo 1 > /sys/device/system/cpuN/online

基本的思想是：通过offline的方式将指定的CPU上的进程（包括内核线程）迁移到其他CPU上，然后设置cpus_allowed和调度域，online之后保证该CPU是完全“纯净”的。

- 涉及内核修改：

（1）新增CPU设备文件： /sys/device/system/cpuN/isolated，如果被置位/清零，则修改cpu_isolated_map。

（2）在CPU热插拔处理通知链中，增加CPUSET中扫描所有task，更新cpus_allowed及sched domains。

Thank you

www.huawei.com