**GP-linux+RT-KVM项目问题回复**

课题需求综合了现有acrn以及PRT-linux+RT-KVM两种方案的缺点，预期目标基于GP-linux结合RT-KVM开发支持低时延的RT-VM。我们认为基本思路有两种：一是采用对VM的硬件资源的硬划分与隔离，将VM中的大部分资源绑定，在VM中运行RTOS，来支持完成时间确定性的任务。二是可以将VM中的实时任务暴露给hypervisor，可以通过在hypervisor级别实现多级调度算法，来确保任务的确定性时延，同时最大化吞吐率。

方案一，虽然RTOS可以支持确定性任务，但是运行与VM中后，会受到hypervisor的管理，进而从硬件和软件层面对其性能有所影响。

硬件层面：

（1）虚拟化涉及的几个大方面：处理器虚拟化，内存虚拟化，I/O虚拟化，中断虚拟化，其硬件资源可分为可共享和独占式硬件，对于独占式硬件（硬件本身不支持虚拟化）硬划分是否可行，如何进行划分?（举例说明）

Re:

  研究目标硬件范围为基于X86和ARM64服务器处理器的硬件，都提供硬件辅助虚拟化支持，在方案上可不考虑硬件不支持辅助虚拟化的情况。

  在同一节点VM分为两大类，一类是NRT-VM，一类是RT-VM。对于RT-VM可以要求VCPU与PCPU一一对绑定，避免CPU超配，内存也可以要求VM创建时直接分配映射好（内存虚拟化可能的方案是使用二级页表机制本身的隔离能力，并结合内存预占，消除缺页引发的陷入陷出开销）。以避免不必要的VCPU调度或内存动态映射带来的不确定性和时延影响；对于NRT-VM我们希望保留现有GP-LINUX+KVM的现有缺省能力(包括资源共享资源调度能力及吞吐能力)。

  对于RT-VM的IO虚拟化，分为两大类考虑，一类是实时路径IO（目前看主要是业务面实时网络IO，未来可能会存在一些加速设备，如GPU/AI芯片），一类是非实时路径IO（管理面或控制面非实时网络IO、存储IO等），研究目标希望能够确保实时路径IO的确定性和低时延，确保非实时路径IO尽量保持好的吞吐能力，并避免对实时路径IO产生影响。为确保实时路径IO的确定性和低时延，一种选择是要求IO硬件本身支持虚拟化，如网卡要求支持SR-IOV，并支持中断直通（如果实时路径IO设备本身不支持虚拟化，则要求RT-VM独占直通使用此设备，并支持中断直通），一种是支持Poll mode虚拟化方式，避免中断影响。

（2）为了确保确定完成时间，硬件相关中断的处理：I/O中断，时钟中断，IPI等。如何确保不产生副作用？

Re:

  对于RT-VM，可以允许实时路径IO设备直通，中断直通；时钟中断可以考虑CPU的本地高精度定时器直通（如X86的LAPIC直通）,IPI可以考虑HOSTOS PCPU之间去掉IPI影响（可能涉及需要把HOSTOS Per CPU的线程进行隔离），GUEST内VCPU间的IPI希望能够直接在GUEST MODE处理，避免VMEXIT或采用EXITLESS的方案

（3）硬件资源的静态分区后，通过pass-through的方式减少模式切换，是否会造成host本身无法使用该资源，同时hypervisor本身的资源管理应该到什么级别？

Re:

同一物理节点有NRT-VM和RT-VM，对于RT-VM从工程角度来看，RT-VM所申请的资源被预留独占使用是允许的，NRT-VM在余下资源可进行共享调度（在CT领域典型场景下CPU和内存都没有超配）

软件层面：

1.  hypervisor本身对资源进行管理，处理vm对资源的请求，hypervisor在响应vm的请求时，涉及到大量的模式切换，在增强分区减少vmexit的情况下，是否也应该提出一种exitless的方式来完成vm的请求？

Re:

  对于NRT-VM，可以保持通用GP-LINUX+KVM的已有运行机制和能力，允许现有合理的VMEXIT。

  对于RT-VM，创建期间允许有VMEXIT,GUESTOS中RT任务在启动过程中因访问非实时路径IO导致VMEXIT也可以允许，但希望GUESTOS中RT任务开工后是zero VMEXIT，或者产生的vmexit在工程上的影响是可以忽略不计的。

2.  在静态分区的作用下，带来的副作用是资源利用率会下降，是否寻求更好的运行时动态自适应的划分方法？

Re:

  对于NRT-VM，可以保持通用GP-LINUX+KVM支持共享调度的能力。

  对于RT-VM，优先保证实时和确定性，资源利用率不是第一目标。资源分区是相对简单易理解的方式，目的是不希望邻位RT/NRT-VM共享资源带来干扰。对于RT-VM的分区希望是支持根据VM的资源需求动态创建/删除（可要求以物理CORE为单位，内存要求1G大页为单位），一旦创建好，可以要求不允许增减资源。如果采用动态自适应划分，可能因邻位负载的多样性和动态变化，带来不确定性，并导致算法的复杂度，且指标相对静态分区预计会差一些，进而影响在工程上的落地效果。当然如果有简单有效的动态分区划分机制，且效果与静态相近，也可以考虑，但预计难度较大，可以考虑作为长期优化方向。我们希望的是一个中短期能工程落地，并且兼顾长期演进的方向。

  有洞察到redhat和boston大学在合作研究《Outfitting QEMU/KVM with Partitioning Hypervisor Functionality》[https://research.redhat.com/blog/research\_project/hypervisor-with-kvm/](https://research.redhat.com/blog/research_project/hypervisor-with-kvm/" \t "_blank)。推测是基于GP-KVM进行增强研究的，这个研究估计还在进行中，目前还没有看到相关研究论文发布。

3.  给vm预留资源，会导致Host本身资源减少，吞吐量下降，host本身的性能是否需要考虑？

Re:

  RT-VM从HOSTOS预留资源后，能够确保HOSTOS本身及NRT-VM在余下资源范围上吞吐量最优即可（可通过与ACRN和RT-KVM在为RT-VM预留资源后，对NRT-VM+GP-LINUX的吞吐进行对比）

（4）openstack目前实现的rt-kvm在使用中具体存在什么样的问题？

Re:

  RT-KVM在实际工程部署很少用，主流成熟大规模使用的还是GP-KVM（GP-LINUX+KVM），提供NRT-VM能力，更希望在此基础上进行增强提供RT-VM和NRT-VM并存的能力。

  RT-KVM依赖的PRT-LINUX（内核完全全抢占的preempt-rt Linux）本身提供的补丁仍有300+补丁未合入LINUX内核主线，需要额外维护，如使用RT-KVM，维护代价较高。

  从NOKIA的一篇论文《5G QoS: Impact of Security Functions on Latency》看，基于RT-KVM创建的RT-VM仍有VMExit、调度、中断等干扰，会导致延迟激增。

  RT-KVM依赖HOSTOS基于PRT-LINUX,这会导致NRT-VM的吞吐量下降。

  当RT-VM同一个VCPU上有RT和NRT线程并存时，如果NRT线程执行慢IO等操作导致VMEXIT进行后端处理，但HOSTOS并不感知GUESTOS的线程调度信息。存在RT-VM中RT线程不能实时抢占的问题。但如果对RT-KVM进行优化，进而感知GUESTOS调度，则需要具备感知多种不同GUEST OS的差异，并且可能需要GUESTOS修改，在复杂度和通用性上面临挑战。因此希望研究有没有可能在GP-LINUX+KVM作为HOSTOS的情况下，是否有可能在这种场景下让VMEXIT的后端处理交由HOSTOS的其它PCPU进行异步处理，让VMEXIT退出的时间控制在极短时间范围(比如确定的1us内)内能够迅速返回到GUEST模式。

方案二，不做资源的静态划分，仍然统一通过hypervisor提供访问，RTOS本身的进程过线程实际运行的VCPU，将相关信息报告给hypervisor，有hypervisor感知当前运行的所有rt/nrt线程，统一进行资源分配和vcpu调度（intra-aware hypervisor）。同时结合目前已实现的优化技术，支持实时任务，保证整体吞吐率。

Re:

  RT-VM中运行的guest RTOS预计会有多种类型，包括PRT-LINUX，一些传统的RTOS等，如QNX，VXWORKS等。如果由HYPERVISOR来感知不同GUESTOS的线程模型，通用性可能会存在挑战。

  同时RT-VM这些GUEST RTOS可能是由第三方APP厂商自带。因此更希望对于GUEST RTOS最好是不用修改就可以运行在RT-VM获得实时性。

  如果做资源的动态调度，从极致低时延和确定性角度来看，相对静态，在指标上很可能会差一些。且研究难度相比静态预计会有很大差别。我们希望用约1年周期研究完技术可行性，并完成原型。1年后能够工程落地。因此更希望中短期优先考虑简单可靠的方案，本次研究范围希望重点先放在中短期，同时能希望能够支持长期演进。