创新成就龙的梦想

龙芯KVM虚拟化概览

陈华才

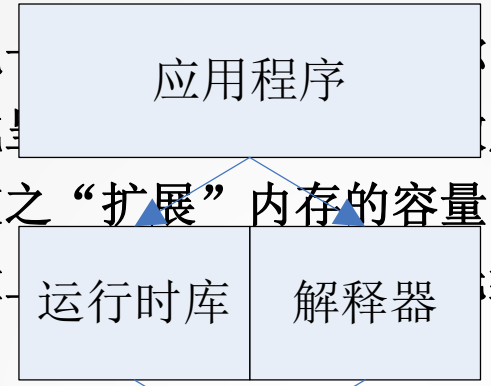
2020-10

目录

CONTENTS

航天龙梦
LEMOTE TECHNOLOGY

- ✓ 最广泛的虚拟化定义
  - ✓ 在计算机领域，凡是把一[illegible]，如硬件层面的设备或软件层面的文件）以另一种形式呈[illegible]虚拟化。比如，用特殊的组织和访问方式操作磁盘，使之“扩展”内存的容量，就叫虚拟内存；用光盘数据的组织方式读写[illegible]盘。



运行时库虚拟化（Wine）　　编程语言虚拟化（JVM）

- ✓ 相对狭义的虚拟化定义
  - ✓ 通常指的是一种软件方[illegible]面上为一个、一类或者所有应用程序虚拟出一个相对隔离的运行环境。

操作系统虚拟化（Chroot、Jail、容器）

  - ✓ 狭义虚拟化分类：全系统[illegible]拟化、编程语言虚拟化和运行时库虚拟化。

全系统虚拟化（虚拟机、模拟器）

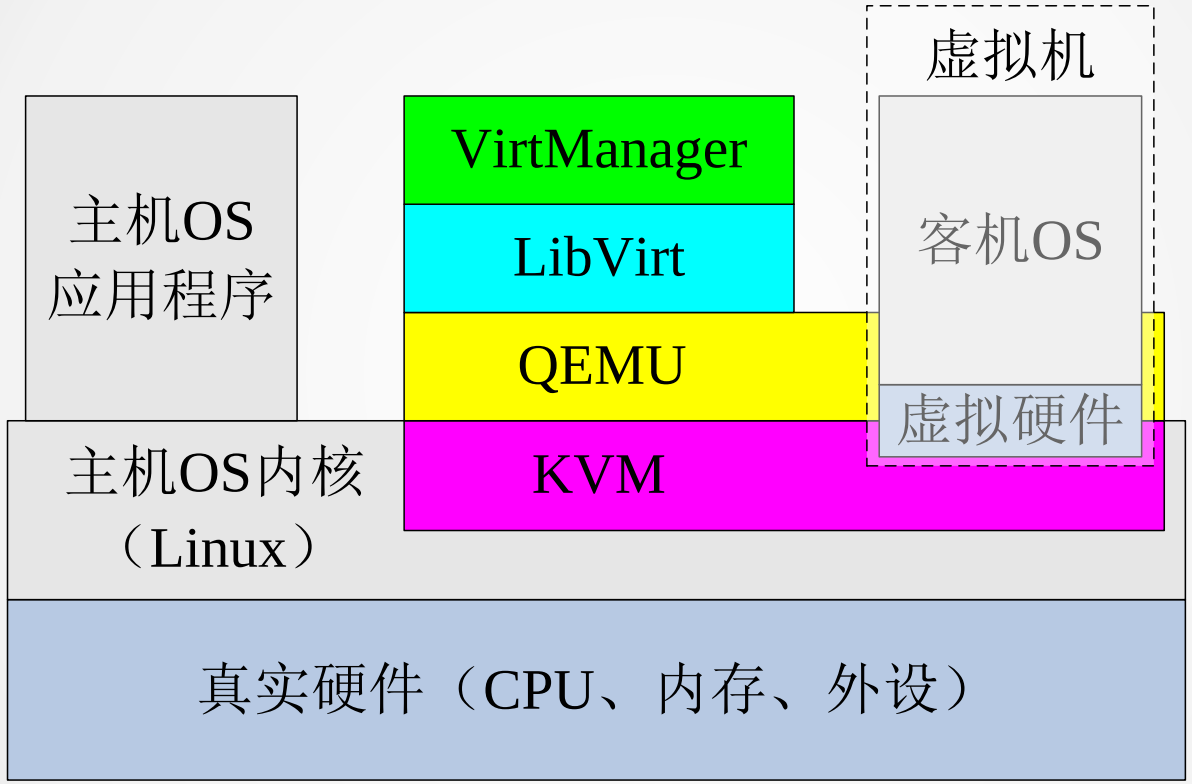
✓全系统虚拟化

- ✓ **虚拟机监控器即Virtual Machine Monitor，常简称VMM，有时也称为Hypervisor，意为管理者。**

那么KVM属于哪种VMM？

怎么理解都行。如果把Linux内核看作一个整体，那么是独立型VMM；如果把KVM模块看作Linux内核的一部分，那么是宿主型VMM。

✓ KVM虚拟化组件全景图



目录

CONTENTS

航天龙梦
LEMOTE TECHNOLOGY

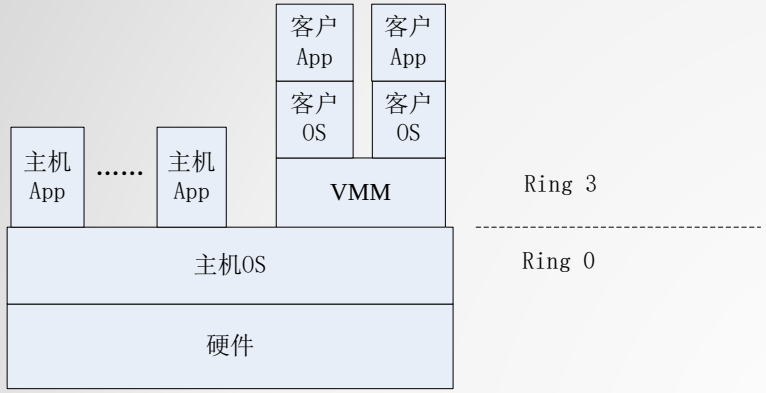
✓ CPU虚拟化的本质——处理指令集与特权级的问题

- ✓ GuestOS的指令执行方式：直接执行（对应HostOS一条真实指令），模拟执行（对应HostOS一段代码即多条指令）
- ✓ 模拟执行的三种方式：自陷并模拟（Trap & Emulate）、静态替换并模拟（Hypercall）、动态替换并模拟（Scan at Loading）
- ✓ 指令分类：特权指令与非特权指令，敏感指令与非敏感指令
- ✓ 特权指令只能在高特权模式（如内核态）执行，在低特权模式（如用户态）下执行会产生自陷异常（Trap）因而可以模拟（Emulate）；非特权指令可以在任意模式直接执行，不需要模拟
- ✓ 敏感指令是指会访问特权资源（如全局配置寄存器），非敏感指令不会
- ✓ 并非所有的敏感指令都是特权指令，敏感非特权指令是虚拟化的一个不利因素（在低特权模式下行为不正确但又不能“自陷并模拟”，只能“替换并模拟”）

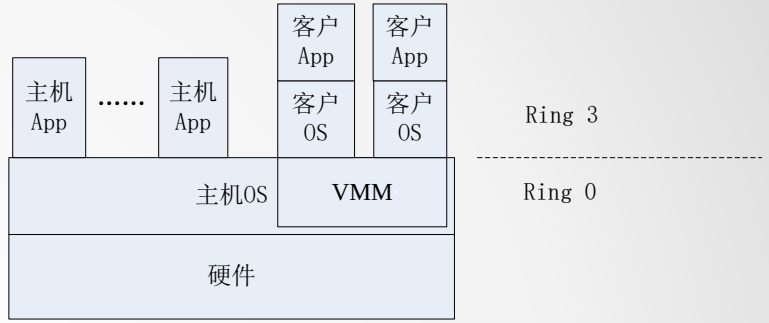
✓ CPU虚拟化的分类

- ✓ 动态翻译模拟所有指令——模拟器
  - ✓ BOCHS，QEMU/TCG
- ✓ 直接执行非敏感指令，动态替换敏感指令——全虚拟化
  - ✓ VMWare，Virtual PC，Virtual Box
- ✓ 直接执行非敏感指令，静态替换敏感指令——半虚拟化
  - ✓ Xen/PV
- ✓ 直接执行非敏感指令，自陷所有敏感指令——硬件辅助虚拟化
  - ✓ Xen/HV，QEMU/KVM
  - ✓ 属于全虚拟化的特例（不需要专门设计的**GuestOS**）

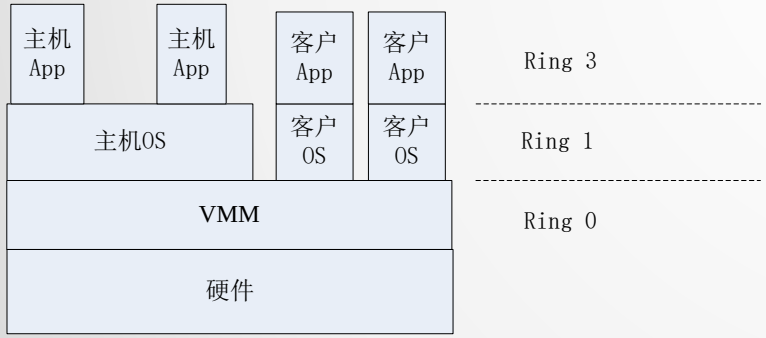
CPU/内存虚拟化



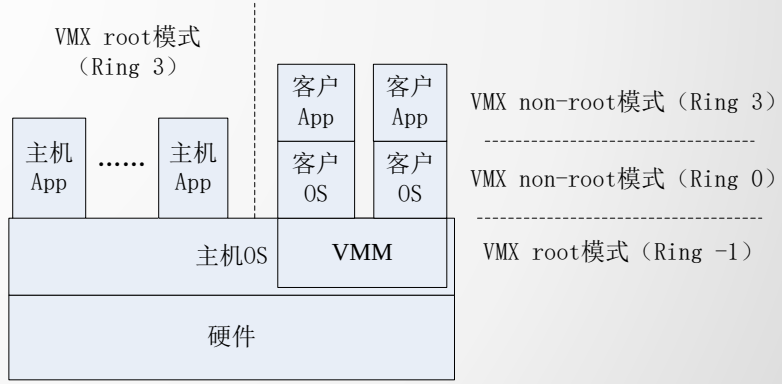
模拟器



全虚拟化



半虚拟化



硬件辅助虚拟化

- X86与MIPS（龙芯）的对比
  - X86特权级：Ring0、Ring1、Ring2、Ring3
  - MIPS特权级：K态，S态，U态
  - X86的硬件虚拟化扩展：VMX，SVM
  - MIPS的硬件虚拟化扩展：VZ
  - 硬件虚拟化扩展的本质：特权资源复制
    - 特权资源包括特权模式、特权指令、特权寄存器、MMU等
    - HostOS运行在根态，GuestOS运行在非根态（客态）
    - X86：根态与非根态各有4个特权级（Ring0~3）
    - MIPS：根态与非根态各有3个特权级（K/S/U）
    - 根态扩展出一些操作非根态特权资源的指令
    - 非根态的所有敏感指令执行时都会发生自陷，因而方便模拟

MIPS/龙芯处理器主要特征

- 处理器核的逻辑结构：主处理器、协处理器0（CP0）、协处理器1（CP1）、协处理器2（CP2）、协处理器3（CP3）
- 龙芯的具体情况：主处理器、系统控制协处理器（CP0）、浮点运算协处理器（FPU，亦即CP1）、多媒体运算协处理器（CP2）

- 通用寄存器：32个，$0～$31
- CP0寄存器：控制系统配置与状态，是主要的特权资源

- 特权级：K态（内核态）、S态（管理态）、U态（用户态），分别等价于X86的Ring0、Ring1/2、Ring3
- Status寄存器的KSU位控制特权级，但Status寄存器中EXL/ERL置位时，不管KSU如何取值，都自动处于K态

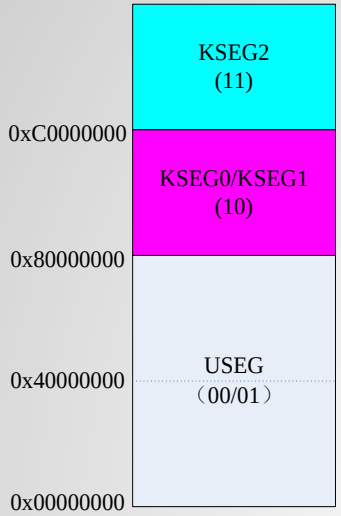
CPU/内存虚拟化-通用寄存器

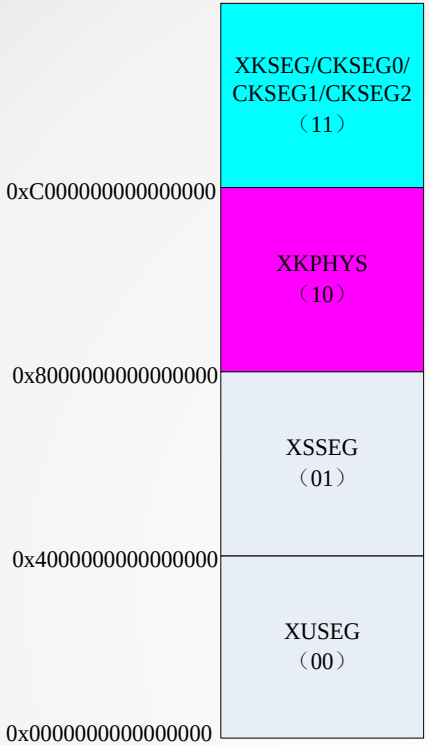
寄存器编号	寄存器名称（O32）	寄存器名称（N32/N64）
$0	zero	zero
$1	at	at
$2～$3	v0～v1	v0～v1
$4～$7	a0～a3	a0～a3
$8～$11	t0～t3	a4～a7
$12～$15	t4～t7	t0～t3
$16～$23	s0～s7	s0～s7
$24～$25	t8～t9	t8～t9
$26～$27	k0～k1	k0～k1
$28	gp	gp
$29	sp	sp
$30	fp/s8	fp/s8
$31	ra	ra

CPU/内存虚拟化-CP0寄存器

寄存器编号	子寄存器编号	寄存器名称	功能简介
0	0	Index	TLB指定索引寄存器
1	0	Random	TLB随机索引寄存器
2	0	EntryLo0	TLB 表项低位内容中与偶数虚页相关部分
3	0	EntryLo1	TLB 表项低位内容中与奇数虚页相关部分
4	0	Context	包含32位模式页表项指针
5	0	PageMask	VTLB页面大小控制
	1	PageGrain	大物理地址及RI/XI控制
6	0	Wired	控制 VTLB 中固定项数目
7	0	HWREna	RDHWR 指令可访问寄存器使能控制
8	0	BadVAddr	记录最新地址相关异常（TLB重填、TLB无效、TLB修改、非对齐访问等）的出错地址
9	0	Count	处理器时钟计数器
10	0	EntryHi	TLB 表项高位内容
11	0	Compare	计时器中断控制
12	0	Status	处理器状态与控制寄存器
13	0	Cause	存放上一次异常的原因
14	0	EPC	存放上一次发生异常指令的PC
15	0	PRId	处理器ID（标识）
	1	EBase	异常入口基址寄存器
16	0-6	Config0-6	配置寄存器0-6
17	0	LLAddr	存放Load-Link（链接加载）指令访问地址
20	0	XContext	包含64位模式页表项指针
30	0	ErrorEPC	存放上一次发生错误的指令的PC

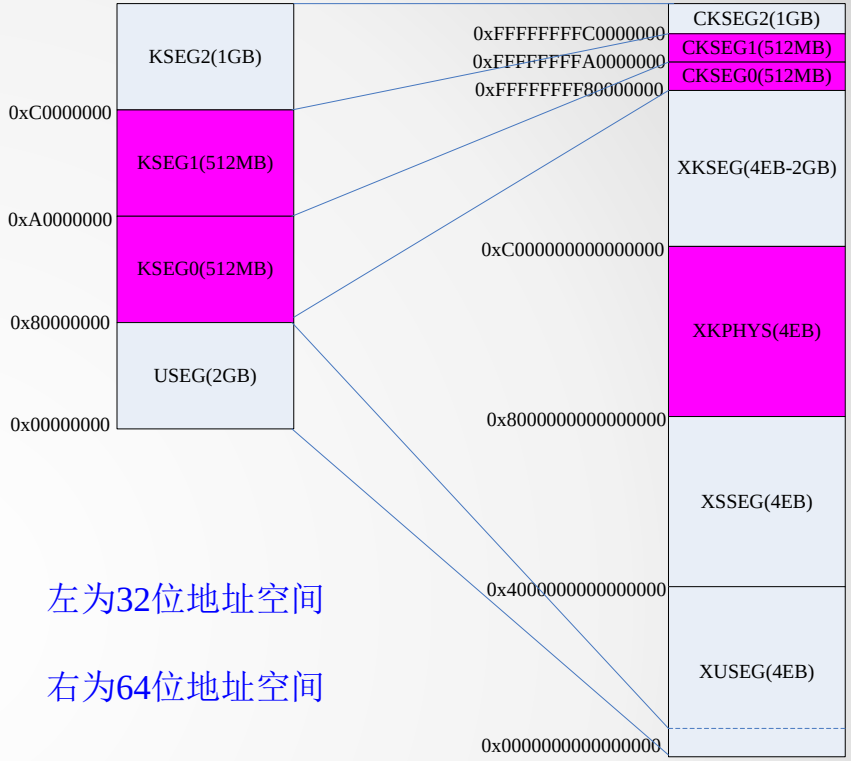
CPU/内存虚拟化-内存地址空间





左为32位地址空间

右为64位地址空间



左为32位地址空间

右为64位地址空间

- ✓ **MIPS32：K态能访问全部地址空间，S态只能访问SSEG和USEG，U态只能访问USEG**
- ✓ **MIPS64：K态能访问全部地址空间，S态只能访问XSSEG和XUSEG，U态只能访问XUSEG**
- ✓ **32位地址空间里面，KSEG2的低半部分也叫SSEG；64位地址空间里面，XKSEG的顶部是兼容32位地址空间的CKSEG0/1/2**
- ✓ **紫红色的区域是不需要页表映射的，虚拟地址去掉一个固定偏移量就是物理地址（线性映射）其他区域是需要页表映射的（分页映射）**

- KVM/MIPS的两种模式
  - TE模式：不要硬件虚拟化扩展，仅支持32位GuestOS，不支持SMP、Host/Guest内核不能通用（半虚拟化）
  - VZ模式：需要硬件虚拟化扩展，支持32/64位GuestOS，可支持SMP、Host/Guest内核能够通用（全虚拟化）
  - TE模式局限的原因
    - 敏感非特权指令（如访存指令）
    - 内存地址空间不够用（切分USEG分别用于Guest应用/Guest内核）
  - VZ模式扩展的内容
    - 特权级复制：根态与客态，各有K/S/U三个特权级
    - 特权资源复制：CP0寄存器上下文（每个CPU核都有一个Root CP0和Guest CP0）
    - 虚拟化扩展指令：hypcall，mfgc0，dmfgc0，mtgc0，dmtgc0，tlbginv，tlbginvf，tlbgp，tlbgr，tlbgwi，tlbgwr等，用于在根态操作Guest CP0
    - 虚拟化扩展寄存器：GuestCtl0/1/2/3，控制运行模式、指令行为等
    - TLB虚拟化：FMT、BAT、双层TLB（Guest：GVA→GPA，Root：GPA→HPA）

- KVM/MIPS/VZ的设计
  - KVM模块初始化：module_init(kvm_mips_init);
  - kvm_mips_init()

```
|--kvm_mips_entry_setup();
\--kvm_init();
    |--kvm_arch_init();
    |    \--kvm_mips_emulation_init(&kvm_mips_callbacks);
    |--kvm_arch_hardware_setup();
    \--misc_register(&kvm_dev);
```

  - 虚拟机是一个容器，其中包含多个可运行实体（虚拟CPU），每个虚拟CPU对应真机上面的Host OS中的一个进程
  - 一个真实的CPU核在同一时刻只能运行一个进程或虚拟CPU，虚拟CPU在真实CPU上轮流运行，本质上就是Host OS中的进程切换

- KVM/MIPS/VZ的设计
  - 创建虚拟机：通过对/dev/kvm执行ioctl()完成
  - kvm_dev_ioctl()

```
\--kvm_dev_ioctl_create_vm();
    \--kvm = kvm_create_vm(type);
        |--kvm = kvm_arch_alloc_vm();
        |--kvm_arch_init_vm(kvm, type);
        |    |--kvm->arch.gpa_mm.pgd = kvm_pgd_alloc();
        |    \--kvm_init_loongson_ipi(kvm);
         \--hardware_enable_all();
             \--kvm_arch_hardware_enable();
                 \--kvm_mips_callbacks->hardware_enable();
                     \--kvm_vz_hardware_enable();
                         \--csr_writel(csr_readl(0xfffffec) | 0x1, 0xfffffec);
```

- KVM/MIPS/VZ的设计
  - 创建虚拟CPU：通过对/dev/kvm执行ioctl()完成
  - kvm_vm_ioctl()

```
\--kvm_vm_ioctl_create_vcpu(kvm, arg);
   |--kvm_arch_vcpu_create(kvm, id);
   |    \--kvm_vcpu_init(vcpu, kvm, id);
   |         \--kvm_arch_vcpu_init();
   |              \--kvm_mips_callbacks->vcpu_init(vcpu);
   |                   \--kvm_vz_vcpu_init(vcpu);
   \--kvm_arch_vcpu_setup(vcpu);
        \--kvm_mips_callbacks->vcpu_setup(vcpu);
             \--kvm_vz_vcpu_setup(vcpu);
```

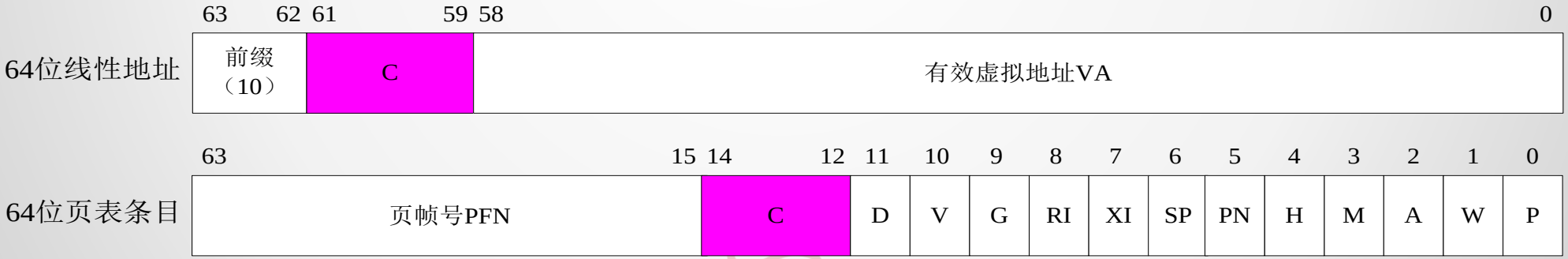
- KVM/MIPS/VZ的设计
  - 运行虚拟CPU：通过对/dev/kvm执行ioctl()完成
  - kvm_vcpu_ioctl()

```
\--kvm_arch_vcpu_ioctl_run(vcpu, vcpu->run);
   |--vcpu_load(vcpu);
   |    \--kvm_arch_vcpu_load(vcpu, cpu);
   |         \--kvm_mips_callbacks->vcpu_load(vcpu, cpu);
   |              \--kvm_vz_vcpu_load(vcpu, cpu);
   \--kvm_mips_callbacks->vcpu_run(run, vcpu);
        \--kvm_mips_callbacks->vcpu_setup(vcpu);
             \--kvm_vz_vcpu_run(vcpu);
```

- Cache属性（CCA）问题
  - 0x2：非缓存（UC，Uncached）
  - 0x3：缓存（CA，Cached）

那么问题来了，GuestOS的Cache属性由谁决定？取决于GuestCCA（GVA→GPA的属性），还是取决于RootCCA（GPA→HPA的属性）？

- MIPS缺省规则：UC>UCA>CA，在GuestCCA跟RootCCA之间取极大值
- 龙芯3号扩展规则：一律由RootCCA决定，有利于龙芯的硬件维护Cache一致性（包括指令/数据一致性、多核一致性、DMA一致性等）



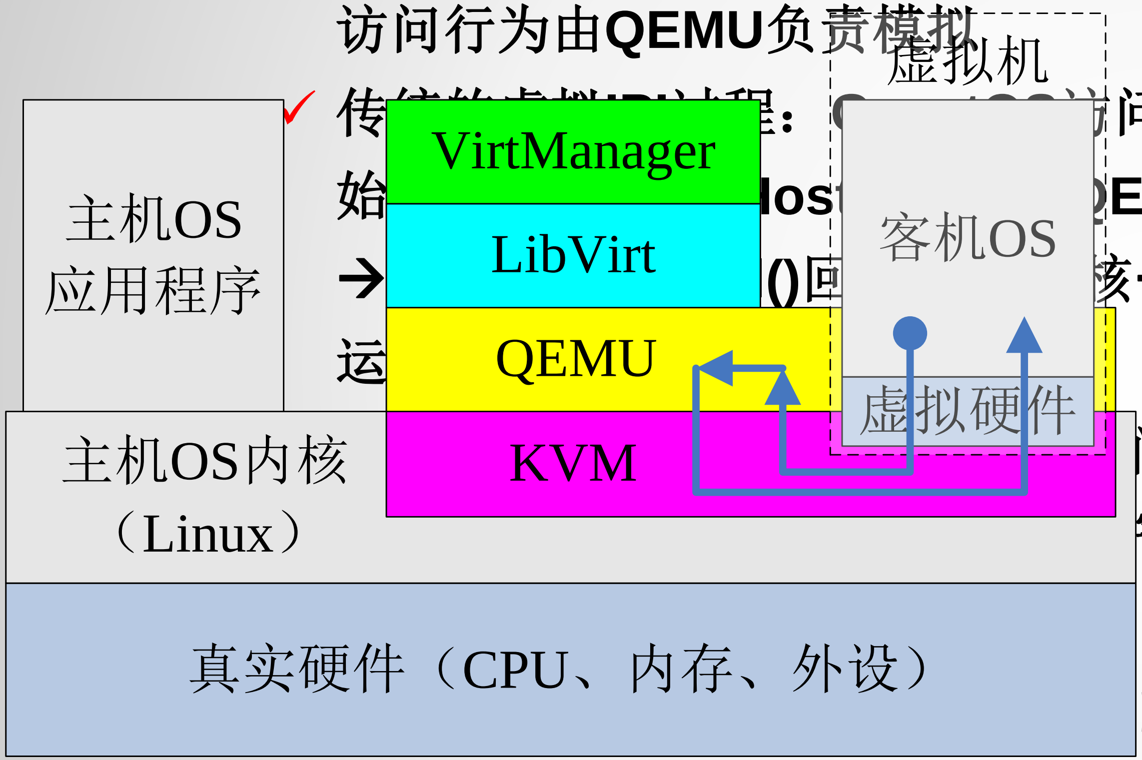
✓核间中断（IPI）虚拟化

- ✓ IPI寄存器是物理地址位于0x3ff01000处的一段MMIO空间
- ✓ 每个核有一组寄存器，可以跨核访问
- ✓ IPI使用频繁，对快速响应的期望较高

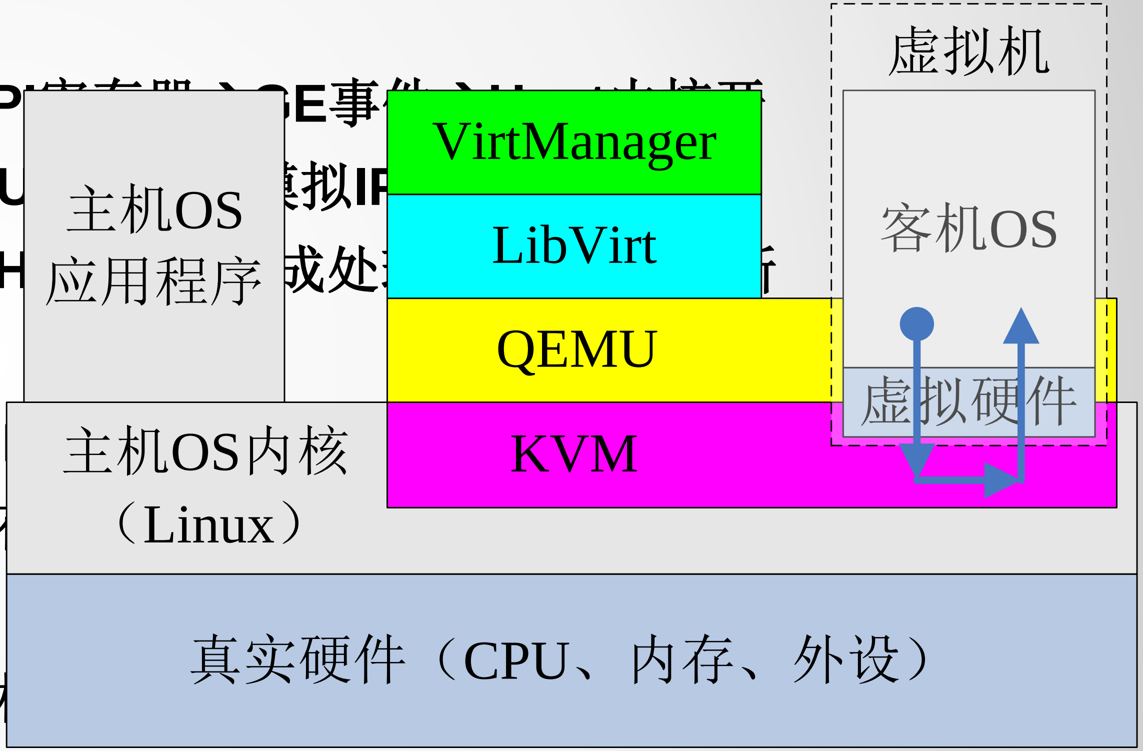
名称	读写权限	功能描述
IPI_Status	只读	32位状态寄存器，某一位置位代表收到某种类型IPI中断
IPI_Enable	读写	32位使能寄存器，控制对应的中断位是否有效
IPI_Set	只写	32位置位寄存器，将某一位写1导致IPI_Status对应为置1
IPI_Clear	只写	32位清零寄存器，将某一位写1导致IPI_Status对应为置0
IPI_MailBox0	读写	64位消息传递缓冲寄存器0，用于核间交换信息
IPI_MailBox1	读写	64位消息传递缓冲寄存器1，用于核间交换信息
IPI_MailBox2	读写	64位消息传递缓冲寄存器2，用于核间交换信息
IPI_MailBox3	读写	64位消息传递缓冲寄存器3，用于核间交换信息

CPU/内存虚拟化

- ✓ 核间中断（IPI）虚拟化
  - ✓ GuestOS访问MMIO会导致GE事件（Guest Exit），而传统的MMIO访问行为由QEMU负责模拟
  - ✓ 传统的虚拟IPI过程：G… 访问IPI寄存器→GE事件→Host内核开始 … Host … QEMU … 模拟IP … →… ()回 … 核→H … 成处 … 运 …



问I … 寄 … 内 …



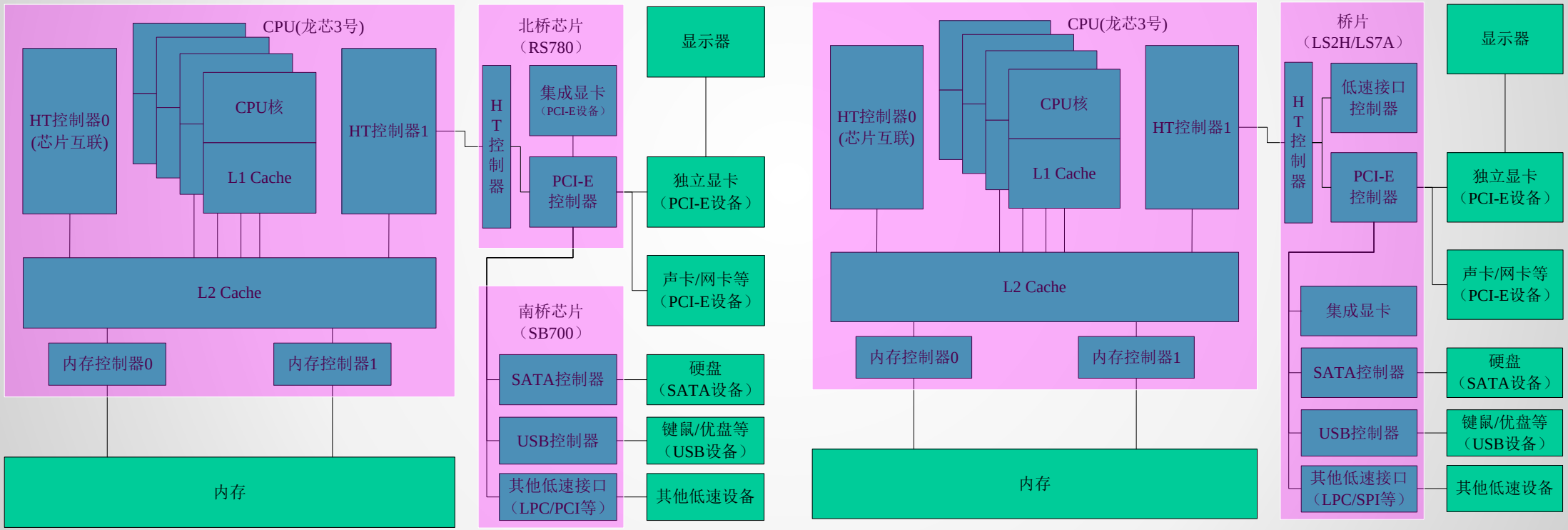
短了MMIO模拟的路径，减少了上下文切换的次数，提高了响应速度

目录

CONTENTS

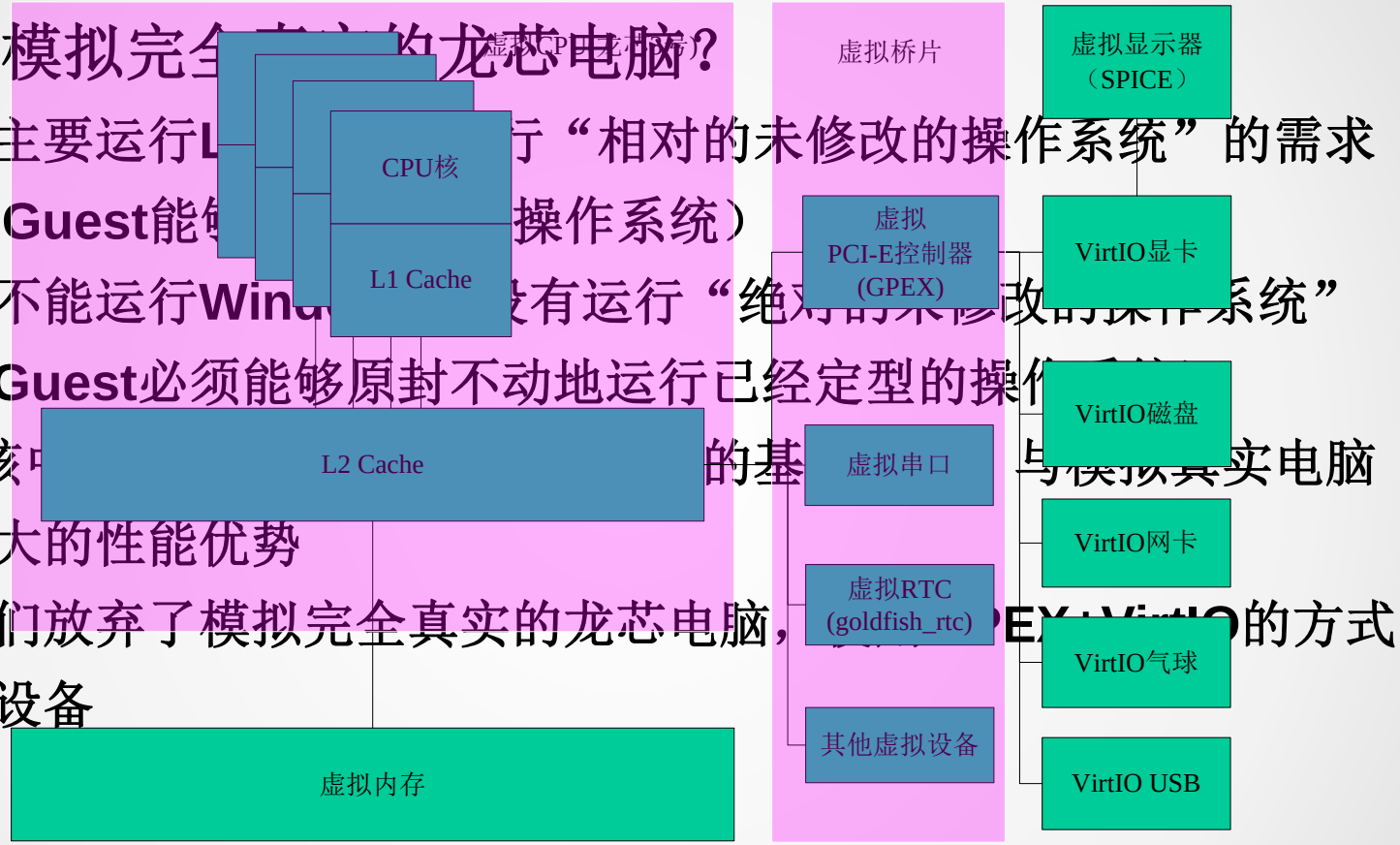
✓ 真实的龙芯电脑

- ✓ CPU采用龙芯3号，外设采用自产LS7A桥片
- ✓ CPU采用龙芯3号，外设采用AMD RS780+SB700桥片

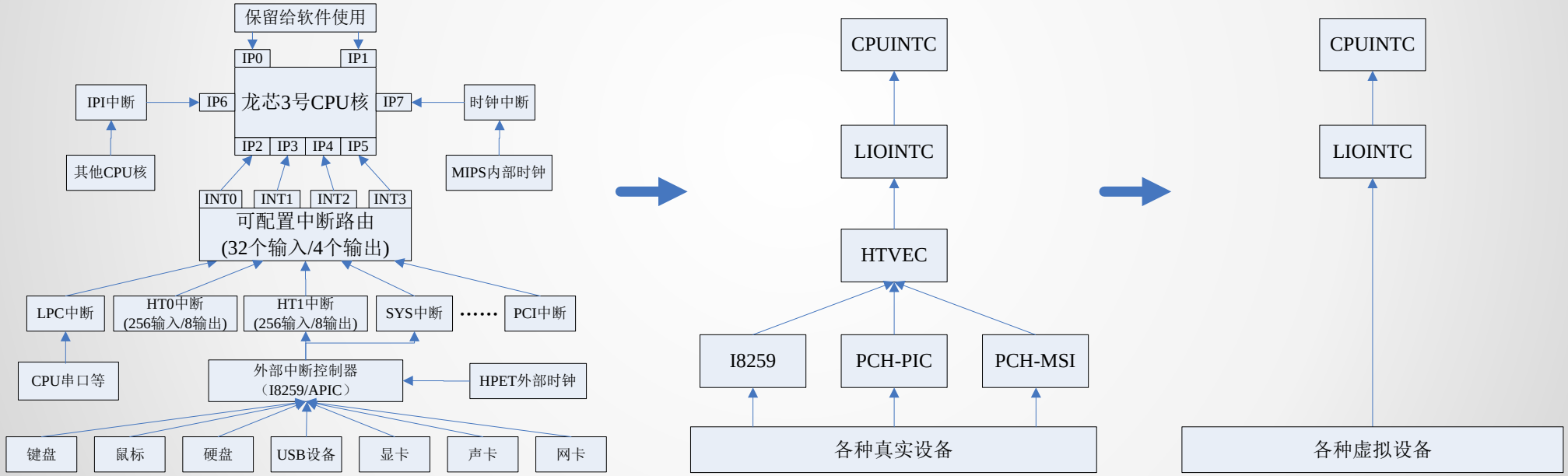


外围设备虚拟化

- ✓ 是否有必要模拟完全真实的龙芯电脑？
  - ✓ 龙芯电脑主要运行Linux，只有运行“相对的未修改的操作系统”的需求（Host和Guest能够运行未修改的操作系统）
  - ✓ 龙芯电脑不能运行Windows，没有运行“绝对的未修改的操作系统”的需求（Guest必须能够原封不动地运行已经定型的操作系统）
  - ✓ Linux内核中的KVM虚拟化技术的基础上，与模拟真实电脑相比有巨大的性能优势
  - ✓ 因此，我们放弃了模拟完全真实的龙芯电脑，采用GPEX+VirtIO的方式模拟外围设备



- ✓ 中断控制器虚拟化
  - ✓ 真实的龙芯电脑是多级层次关系
  - ✓ 虚拟的龙芯电脑使用了最简化层级



- ✓ 龙芯电脑已经全面采用FDT（Flatten Device Tree）描述
  - ✓ DTS描述文件位于arch/mips/boot/dts/loongson/
  - ✓ 描述旧（Classic）龙芯CPU：loongson64c-package.dtsi
  - ✓ 描述新（Generic）龙芯CPU：loongson64g-package.dtsi
  - ✓ 描述LS7A桥片：ls7a-pch.dtsi
  - ✓ 描述RS780桥片：rs780e-pch.dtsi
  - ✓ 描述采用旧CPU+LS7A桥片的龙芯电脑：loongson64c_4core_ls7a.dts
  - ✓ 描述采用旧CPU+RS780桥片的龙芯电脑： loongson64c_4core_rs780e.dts
  - ✓ 描述采用新CPU+LS7A桥片的龙芯电脑： loongson64g_4core_ls7a.dts
  - ✓ 描述采用虚拟CPU+VirtIO桥片的龙芯电脑： loongson64v_4core_virtio.dts

外围设备虚拟化

```
/dts-v1/;
/ {
        compatible = "loongson,loongson64v-4core-virtio";
        #address-cells = <2>;
        #size-cells = <2>;

        cpuintc: interrupt-controller {
                #address-cells = <0>;
                #interrupt-cells = <1>;
                interrupt-controller;
                compatible = "mti,cpu-interrupt-controller";
        };

        package0: bus@1fe00000 {
                compatible = "simple-bus";
                #address-cells = <2>;
                #size-cells = <1>;
                ranges = <0 0x1fe00000 0 0x1fe00000 0x100000
                        0 0x3ff00000 0 0x3ff00000 0x100000
                        0xefd 0xfb000000 0xefd 0xfb000000 0x10000000>;

                liointc: interrupt-controller@3ff01400 {
                        compatible = "loongson,liointc-1.0";
                        reg = <0 0x3ff01400 0x64>;

                        interrupt-controller;
                        #interrupt-cells = <2>;

                        interrupt-parent = <&cpuintc>;
                        interrupts = <2>, <3>;
                        interrupt-names = "int0", "int1";

                        loongson,parent_int_map = <0x00000001>, /* int0 */
                                                  <0xfffffffe>, /* int1 */
                                                  <0x00000000>, /* int2 */
                                                  <0x00000000>; /* int3 */


                };

                cpu_uart0: serial@1fe001e0 {
                        compatible = "ns16550a";
                        reg = <0 0x1fe001e0 0x8>;
                        clock-frequency = <33000000>;
                        interrupt-parent = <&liointc>;
                        interrupts = <0 IRQ_TYPE_LEVEL_HIGH>;
                        no-loopback-test;
                };
        };
```

```
        bus@10000000 {
                compatible = "simple-bus";
                #address-cells = <2>;
                #size-cells = <2>;
                ranges = <0 0x10000000 0 0x10000000 0 0x10000000 /* PIO & CONF & APB */
                                0 0x40000000 0 0x40000000 0 0x40000000>; /* PCI MEM */

                rtc0: rtc@10081000 {
                        compatible = "google,goldfish-rtc";
                        reg = <0 0x10081000 0 0x1000>;
                        interrupts = <1 IRQ_TYPE_LEVEL_HIGH>;
                        interrupt-parent = <&liointc>;
                };

                pci@1a000000 {
                        compatible = "pci-host-ecam-generic";
                        device_type = "pci";
                        #address-cells = <3>;
                        #size-cells = <2>;
                        #interrupt-cells = <1>;

                        bus-range = <0x0 0x1f>;
                        reg = <0 0x1a000000 0 0x02000000>;

                        ranges = <0x01000000 0x0 0x00004000 0x0 0x18004000 0x0 0x0000c000>,
                                 <0x02000000 0x0 0x40000000 0x0 0x40000000 0x0 0x40000000>;

                        interrupt-map = <
                                0x0000 0x0 0x0  0x1  &liointc  0x2 IRQ_TYPE_LEVEL_HIGH
                                0x0800 0x0 0x0  0x1  &liointc  0x3 IRQ_TYPE_LEVEL_HIGH
                                0x1000 0x0 0x0  0x1  &liointc  0x4 IRQ_TYPE_LEVEL_HIGH
                                0x1800 0x0 0x0  0x1  &liointc  0x5 IRQ_TYPE_LEVEL_HIGH
                                >;

                        interrupt-map-mask = <0x1800 0x0 0x0  0x7>;
                };

                isa {
                        compatible = "isa";
                        #address-cells = <2>;
                        #size-cells = <1>;
                        ranges = <1 0 0 0x18000000 0x4000>;
                };
        };

        hypervisor {
                compatible = "linux,kvm";
        };
};
```

目录

CONTENTS

上游社区状态

✓ 内核部分：KVM Host

```
0f78355c450835053fed85828c9d6526594c0921 KVM: MIPS: Enable KVM support for Loongson-3
dc6d95b153e78ed70b1b2c04aadffb76bcd2b3ec KVM: MIPS: Add more MMIO load/store instructions emulation
8a5097ee90c25656db23f44520a9dad7983d88fb KVM: MIPS: Add CONFIG6 and DIAG registers emulation
7f2a83f1c2a941ebfee53f504ed5fdbc61cfb333 KVM: MIPS: Add CPUCFG emulation for Loongson-3
f21db3090de2c056728dee76d5fb66343aaf6dd1 KVM: MIPS: Add Loongson-3 Virtual IPI interrupt support
3f51d8fcac7a0925fb1222d932cb3baa776b1e79 KVM: MIPS: Add more types of virtual interrupts
49bb96003fb19b77a0116d9330bdfd61aa41244b KVM: MIPS: Let indexed cacheops cause guest exit on Loongson-3
52c07e1cbb6e16a0fe70646b4bc63f714caa0f3b KVM: MIPS: Use root tlb to control guest's CCA for Loongson-3
3210e2c279fee1076978b49988acdd935a6f7435 KVM: MIPS: Introduce and use cpu_guest_has_ldpte
8bf31295030e35ef1e9e1b25b02041423f8291d3 KVM: MIPS: Use lddir/ldpte instructions to lookup gpa_mm.pgd
bf10efbb81bf0c7ae6ebac3320b5b40c323463f3 KVM: MIPS: Add EVENTFD support which is needed by VHOST
210b4b9112b699df4c33273bf7b02d9debd2f35e KVM: MIPS: Increase KVM_MAX_VCPUS and KVM_USER_MEM_SLOTS to 16
5816c76dea116a458f1932eefe064e35403248eb KVM: MIPS: Fix VPN2_MASK definition for variable cpu_vmbits
fe2b73dba47fb6d6922df1ad44e83b1754d5ed4d KVM: MIPS: Define KVM_ENTRYHI_ASID to cpu_asid_mask(&boot_cpu_data)
```

- ✓ 在Linux-5.9时进入上游

✓ QEMU部分

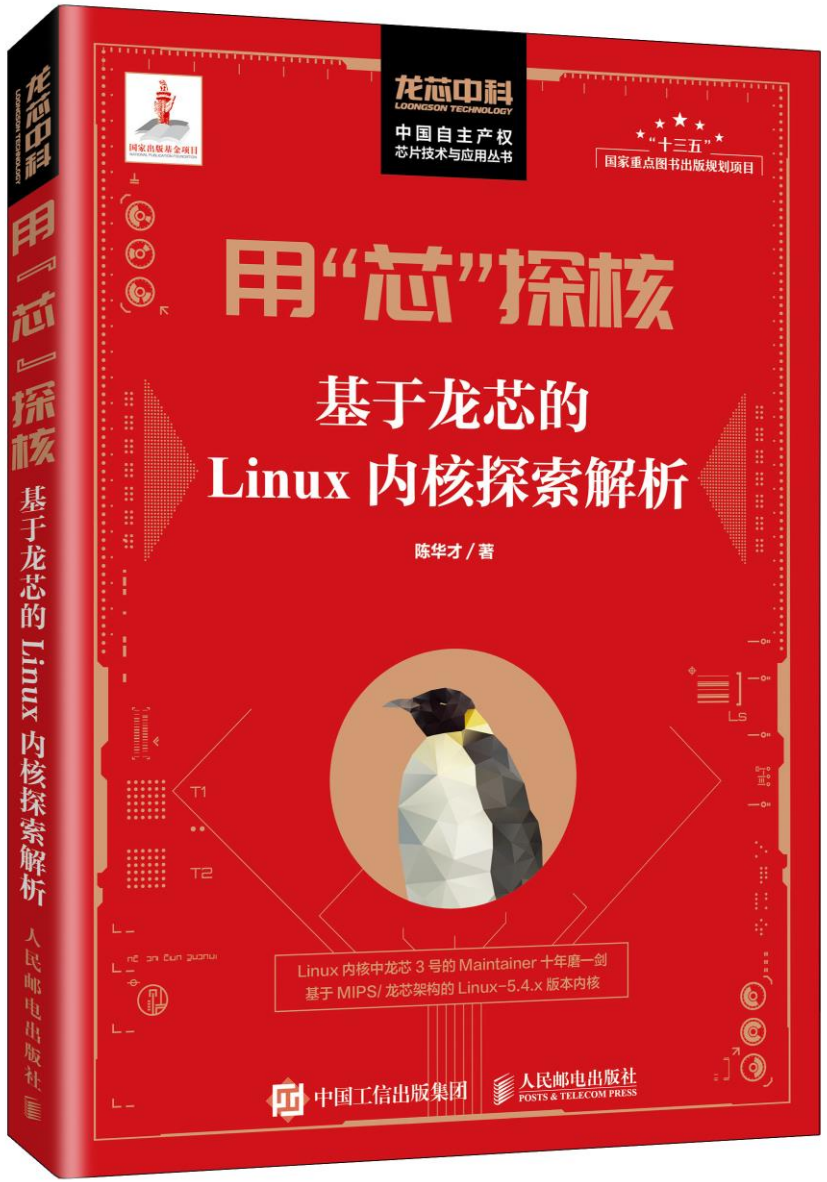
```
326747fa0f77c20a545348596af0c05898605a7d hw/mips: Add Loongson-3 machine support (with KVM)
c012e0b1f9121f3d8d2315af4dfd63084ea23faa hw/intc: Add Loongson LIOINTC support
719d109b7fe153b0ec4dfeba2f303f864555d819 hw/mips: Implement the kvm_type() hook in MachineClass
af868995e1b7641577300d1342ede452ef0c5565 target/mips: Add Loongson-3 CPU definition
7e0896b0e134f5023107d8b4454c020ba51f2d42 target/mips: Add more CP0 register for save/restore
c3173a35bc2a759dbfac4e76e9a7695b1d44e97a hw/mips: Add CPU IRQ3 delivery for KVM
56b92eeeac8074501858e15b7658ec6099456f04 hw/mips/mips_int: De-duplicate KVM interrupt delivery
0009b4f32ea103b44ea3d61f515b5ff2d5dfc031 hw/mips/mips_int: Simplify cpu_mips_irq_init_cpu()
```

- ✓ 下一步计划：QEMU/TCG支持（KVM只能同架构模拟，TCG可以跨架构模拟）

上游社区状态

- ✓ 鸣谢
  - ✓ 李雪峰 <lixuefeng@loongson.cn>
  - ✓ 龙芯中科内核组组长，龙芯Linux内核代码的源头
  - ✓ 李星 <lixing@loongson.cn>
  - ✓ 毛碧波 <maobibo@loongson.cn>
  - ✓ 龙芯中科QEMU/KVM私有方案的主要开发者（私有方案与上游方案设计上有差异）
  - ✓ 吴瑞阳 <wuruiyang@loongson.cn>
  - ✓ 黄沛 <huangpei@loongson.cn>
  - ✓ 龙芯中科工程师，为社区解答诸多有关龙芯硬件虚拟化细节的疑问
  - ✓ 杨嘉勋 <jiaxun.yang@flygoat.com>
  - ✓ QEMU/TCG龙芯部分的主要开发者，QEMU/KVM上游方案的共同设计者与开发者

我和《用芯探核》



- ✓ 我是谁？
- ✓ 陈华才，航天龙梦副总工，内核开发者，Linux内核中MIPS/LOONGSON64和MIPS/KVM的Maintainer

- ✓ 《用芯探核》是什么书？
- ✓ 立足龙芯而包罗万象
- ✓ 第一本基于龙芯处理器的内核书籍
- ✓ 第二本基于Linux-5.x版本的内核书籍
- ✓ 创造性地使用“树形视图”和“链式视图”来解析源代码的

关于航天龙梦

中科梦兰 LEMOTE TECHNOLOGY → 龙芯梦兰 LEMOTE TECHNOLOGY → 航天龙梦 LEMOTE TECHNOLOGY

航天龙梦官方微信公众号

服务热线：400-606-6065

官方网站：www.lemote.com

地　　址：江苏省常熟市梦兰工业园区梦兰路1号
南京市江宁区九龙湖国际企业总部园A1栋7F-8F

龙芯产业化的排头兵，信息技术应用创新的脊梁

谢谢观看

THANKS