

Chinaunix首页 | 论坛 | 问答 | 博客

登录Ⅰ注册

博文 ▼

翱翔在Linux的天空

HumJb & HaHa

首页 | 博文目录 | 关于我



humjb_1983

博客访问: 9500 博文数量: 80 博客积分: 0 博客等级: 民兵 技术积分: 685

用户组: 普诵用户

注册时间: 2014-02-20 08:27

加关注 短消息

论坛 加好友

文章分类

全部博文(80)

硬件相关(5)

虚拟化(13)

其他(1)

Linux其他方面(3)

Linux内核(57)

未分配的博文(1)

文章存档

2014年(80)

我的朋友





321leon

最近访客











ieppeter 刘一痕 SCvsCS

KVM基本原理及架构七-KVM内核模块中重要的数据结构 2014-07-18 19:54:46

CU博客频道6月技术图书有奖试读活动

分类: LINUX

KVM内核模块中重要数据结构

Kvm结构体代表一个具体的虚拟机,当通过KVM CREATE VM指令字创建一个虚拟机后,就会创建 一个新的kvm结构体对象。Kvm结构体中包括了VCPU、内存、APIC、IRQ、MMU、Event事件等相关信 息,该结构体主要在KVM虚拟机内部使用,用于跟踪虚拟机状态。

重要的结构体成员说明如下:

点击(此处)折叠或打开

```
1.
     struct kvm {
     struct kvm_memslots *memslots; //KVM虚拟机分配的内存slot, 用于GPAAHVA的转换, 内存虚拟化使用
5.
            struct kvm_vcpu *vcpus[KVM_MAX_VCPUS]; //KVM虚拟机中包含的VCPU结构体数组, 一个VCPU对应一个
     数组成员。
6.
            struct kvm_io_bus *buses[KVM_NR_BUSES]; //KVM虚拟机中包括的IO总线结构体数组, 一条总线对应一
7.
     个kvm io bus结构体,如ISA总线、PCI总线。
8.
            struct kvm_vm_stat stat; //KVM虚拟机中的运行时状态信息, 比如页表、MMU等状态。
9.
10.
            struct kvm_arch arch; //KVM中跟arch相关的参数。
11.
12.
    }
13.
14.
     7.2 kvm run
     kvm_run结构体记录了KVM的内部运行状态,如VM-Exit发生的原因等。重要字段说明如下:
15.
16.
```

7.2 kvm run

kvm run结构体记录了KVM的内部运行状态,如VM-Exit发生的原因等。重要字段说明如下:

点击(此处)折叠或打开

```
struct kvm run {
        __u8 request_interrupt_window; //向VCPU注入一个中断, 让VCPU做好相关准备工作
2.
          u8 ready_for_interrupt_injection; //响应request_interrupt_window的中断请求,当设置时,说明VCP
    U可以接收中断。
5
         _u8 if_flag; //中断使能标识,如果使用了APIC,则无效
6.
        struct {
                u64 hardware exit reason; //当发生VM-Exit时,该字段保存了由于硬件原因导致VM-Exit的相关
7.
    信息。
8.
           } hw;
        struct {
10.
    #define KVM_EXIT_IO_IN 0
11.
    #define KVM_EXIT_I0_OUT 1
12.
               __u8 direction;
              __u8 size; /* bytes */
13.
              __u16 port;
14.
              __u32 count;
15.
               16.
17.
           } io; //当由于IO操作导致发生VM-Exit时,该结构体保存IO相关信息。
18.
19.
    };
```

7.3 kvm_vcpu

在通过KVM CREATE VCPU指令字为VM创建VCPU后,KVM模块将创建kvm vcpu结构体,其中 包含了VCPU相关的信息,重要字段说明如下:







风铃之音 embedde

订阅

推荐博文

·云计算-Azure-3.负载均衡集...
·读书与写论文的引导书——leo...
·在framework层添加自己的jar...
·tcpdump工具浅析
·python json ajax django四星...
·Solaris文件管理和目录管理...
·Solaris退出系统,改变系统运...
·监控Data Guard实时同步...
·Oracle的告警日志之v\$diag_al...

热词专题

使用AWR生成报表

·Debian设置

·欢迎kkkkkkkybbb在ChinaUnix...

·虚拟机ping不通win7宿主机...

安装oracle

·关于STM32的SPI的问题

点击(此处)折叠或打开

```
struct kvm_run {
        __u8 request_interrupt_window; //向VCPU注入一个中断, 让VCPU做好相关准备工作
 2.
3.
         _u8 ready_for_interrupt_injection; //响应request_interrupt_window的中断请求,当设置时,说明VCP
4.
     U可以接收中断。
         _u8 if_flag; //中断使能标识,如果使用了APIC,则无效
 6.
        struct {
 7.
                _u64 hardware_exit_reason; //当发生VM-Exit时,该字段保存了由于硬件原因导致VM-Exit的相关
     信息。
8.
           } hw:
        struct {
9.
     #define KVM_EXIT_IO_IN 0
10.
     #define KVM_EXIT_I0_OUT 1
11.
               __u8 direction;
12.
               _u8 size; /* bytes */
13.
14.
               __u16 port;
               __u32 count;
15
                16.
           } io; //当由于IO操作导致发生VM-Exit时,该结构体保存IO相关信息。
17.
18.
19.
    };
```

7.4 kvm_x86_ops

kvm_x86_ops结构体只包含了针对具体CPU架构进行虚拟化时的函数指针,在kvm-intel.ko和kvm-amd.ko模块中,提供了不同的接口和实现。在KVM初始化和正常运行过程中,将使用相关接口进行实际的硬件操作。主要包括如下几种类型的操作:

- ü VMM状态初始化
- ü VCPU管理
- ü中断管理
- ü寄存器管理
- ü 时钟管理

Kvm x86 ops结构体在intel架构中初始化为:

点击(此处)折叠或打开

```
static struct kvm_x86_ops vmx_x86_ops = {
 1.
                    .cpu_has_kvm_support = cpu_has_kvm_support,
 2.
                    .disabled_by_bios = vmx_disabled_by_bios,
 3.
                   .hardware_setup = hardware_setup,
 5.
                    .hardware_unsetup = hardware_unsetup,
 6.
                    .check_processor_compatibility = vmx_check_processor_compat,
                   .hardware_enable = hardware_enable,
.hardware_disable = hardware_disable,
 8.
 9.
                   .cpu has accelerated tpr = report flexpriority,
                   .vcpu_create = vmx_create_vcpu,
.vcpu_free = vmx_free_vcpu,
10.
11.
12.
                   .vcpu_reset = vmx_vcpu_reset,
13.
                   .prepare_guest_switch = vmx_save_host_state,
14.
                    .vcpu_load = vmx_vcpu_load,
                    .vcpu_put = vmx_vcpu_put,
15.
                   .set_guest_debug = set_guest_debug,
16.
                   .get_msr = vmx_get_msr,
.set_msr = vmx_set_msr,
.get_segment_base = vmx_get_segment_base,
17.
18.
19.
20.
                   .get_segment = vmx_get_segment,
.set_segment = vmx_set_segment,
21.
                   .get_cpl = vmx_get_cpl,
.get_cs_db_l_bits = vmx_get_cs_db_l_bits,
22.
23.
                   .decache_cr0_guest_bits = vmx_decache_cr0_guest_bits,
.decache_cr4_guest_bits = vmx_decache_cr4_guest_bits,
24.
25.
26.
                   .set_cr0 = vmx_set_cr0,
27.
                   .set_cr3 = vmx_set_cr3,
28.
                   .set_cr4 = vmx_set_cr4
29
                   .set_efer = vmx_set_efer,
                    .get_idt = vmx_get_idt,
30.
                   .set_idt = vmx_set_idt,
.get_gdt = vmx_get_gdt,
31.
32.
33.
                   .set_gdt = vmx_set_gdt,
34.
                   .cache_reg = vmx_cache_reg,
                   .get_rflags = vmx_get_rflags,
.set_rflags = vmx_set_rflags,
35.
36.
37.
                    .fpu_deactivate = vmx_fpu_deactivate,
                    .tlb_flush = vmx_flush_tlb,
38.
                   .run = vmx_vcpu_run,
.handle_exit = vmx_handle_exit,
39.
40.
                   .skip_emulated_instruction = skip_emulated_instruction,
41.
                   .set_interrupt_shadow = vmx_set_interrupt_shadow,
.get_interrupt_shadow = vmx_get_interrupt_shadow,
42.
43.
44.
                    .patch_hypercall = vmx_patch_hypercall,
                   .set_irq = vmx_inject_irq,
.set_nmi = vmx_inject_nmi,
.queue_exception = vmx_queue_exception,
45.
46.
                    .interrupt_allowed = vmx_interrupt_allowed,
```

KVM基本原理及架构七-KVM内核模块中重要的数据结构-humjb 1983-ChinaUnix博客

```
.nmi_allowed = vmx_nmi_allowed,
.get_nmi_mask = vmx_get_nmi_mask,
.set_nmi_mask = vmx_set_nmi_mask,
.enable_nmi_window = enable_nmi_window,
.enable_irq_window = enable_irq_window,
.update_cr8_intercept = update_cr8_intercept,
49
50.
51.
52.
53.
54.
                                .set_tss_addr = vmx_set_tss_addr,
.get_tdp_level = get_ept_level,
.get_mt_mask = vmx_get_mt_mask,
.exit_reasons_str = vmx_exit_reasons_str,
55.
56.
57.
58.
                                 gb_page_enable = vmx_gb_page_enable,
.set_tsc_khz = vmx_set_tsc_khz,
.write_tsc_offset = vmx_write_tsc_offset,
59.
60.
61.
62.
                                 .adjust_tsc_offset = vmx_adjust_tsc_offset,
63.
                                 .compute_tsc_offset = vmx_compute_tsc_offset,
64.
           };
```

阅读(12) | 评论(0) | 转发(0) |

上一篇: KVM基本原理及架构六-KVM API

下一篇: KVM基本原理及架构八-KVM内核模块重要流程分析

0

相关热门文章

KVM基本原理及架构八-KVM内核...linux 常见服务端口C语言 如何在一个整型左边补0...KVM基本原理及架构七-KVM内核...【ROOTFS搭建】busybox的httpd...python无法爬取阿里巴巴的数据...KVM基本原理及架构六-KVM API...xmanager 2.0 for linux配置linux-2.6.28 和linux-2.6.32....KVM基本原理及架构五-IO虚拟化...什么是shelllinux su - username -c 命...完美C++(第5版)(双色)...linux socket的bug??我不得不在这里问一下网站使用...

给主人留下些什么吧!~~

评论热议

请登录后评论。 登录 注册

关于我们 | 关于IT168 | 联系方式 | 广告合作 | 法律声明 | 免费注册

Copyright 2001-2010 ChinaUnix.net All Rights Reserved 北京皓辰网域网络信息技术有限公司. 版权所有

感谢所有关心和支持过ChinaUnix的朋友们 京ICP证041476号 京ICP证060528号