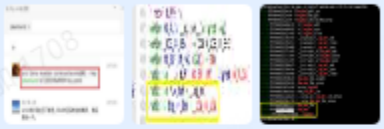
【创新】在主机上直接获取虚拟机内dmesg日志的讨论


```
406: /* record buffer */
407: #define LOG_ALIGN __alignof__(unsigned long)
408: #define __LOG_BUF_LEN (1 << CONFIG_LOG_BUF_SHIFT)
409: #define LOG_BUF_LEN_MAX (u32)(1 << 31)
410: static char __log_buf[__LOG_BUF_LEN] __aligned(LOG_ALIGN);
411: static char *log_buf = __log_buf;
412: static u32 log_buf_len = __LOG_BUF_LEN;
```

```
[root@localhost boot]# grep -n log_buf System.map-5.15.13-0.el9.aarch64
4700:ffff800010241e0 T log_buf_addr_get
4701:ffff800010124200 T log_buf_len_get
4702:ffff800010124220 T log_buf_vmcoreinfo_setup
41766:ffff800010ab3a70 T nf_log_buf_add
41769:ffff800010ab3e20 T nf_log_buf_open
46132:ffff800010c3a4d0 T nf_log_buf_close
63828:ffff80001115a970 r __ksymtab_nf_log_buf_add
63829:ffff80001115a97c r __ksymtab_nf_log_buf_close
63830:ffff80001115a988 r __ksymtab_nf_log_buf_open
73703:ffff800011171e97 r __kstrtabns_nf_log_buf_add
73704:ffff800011171e97 r __kstrtabns_nf_log_buf_close
73705:ffff800011171e97 r __kstrtabns_nf_log_buf_open
86492:ffff800011199a4f r __kstrtab_nf_log_buf_add
86493:ffff800011199a5e r __kstrtab_nf_log_buf_open
86494:ffff800011199a6e r __kstrtab_nf_log_buf_close
88459:ffff80001163dbe0 t log_buf_len_update
88460:ffff80001163dc60 t log_buf_len_setup
88468:ffff80001163e204 T setup_log_buf
90723:ffff8000116d5dd0 d new_log_buf_len
91216:ffff800011743ea2 d __setup_str_log_buf_len_setup
94706:ffff80001176cef0 d __setup_log_buf_len_setup
97794:ffff800011bb614c d log_buf_len
97803:ffff800011bb61e0 d log_buf
112157:ffff80001219f088 b __log_buf
[root@localhost boot]#
```

葛长忠

请教一个问题：通过qemu的pmemsave接口，获取虚拟机的dmesg日志，需要知道log_buf指针指向的物理内存地址，这个物理内存地址怎么可能获取到？



已读

想要guest_va转换到host_va 吗?

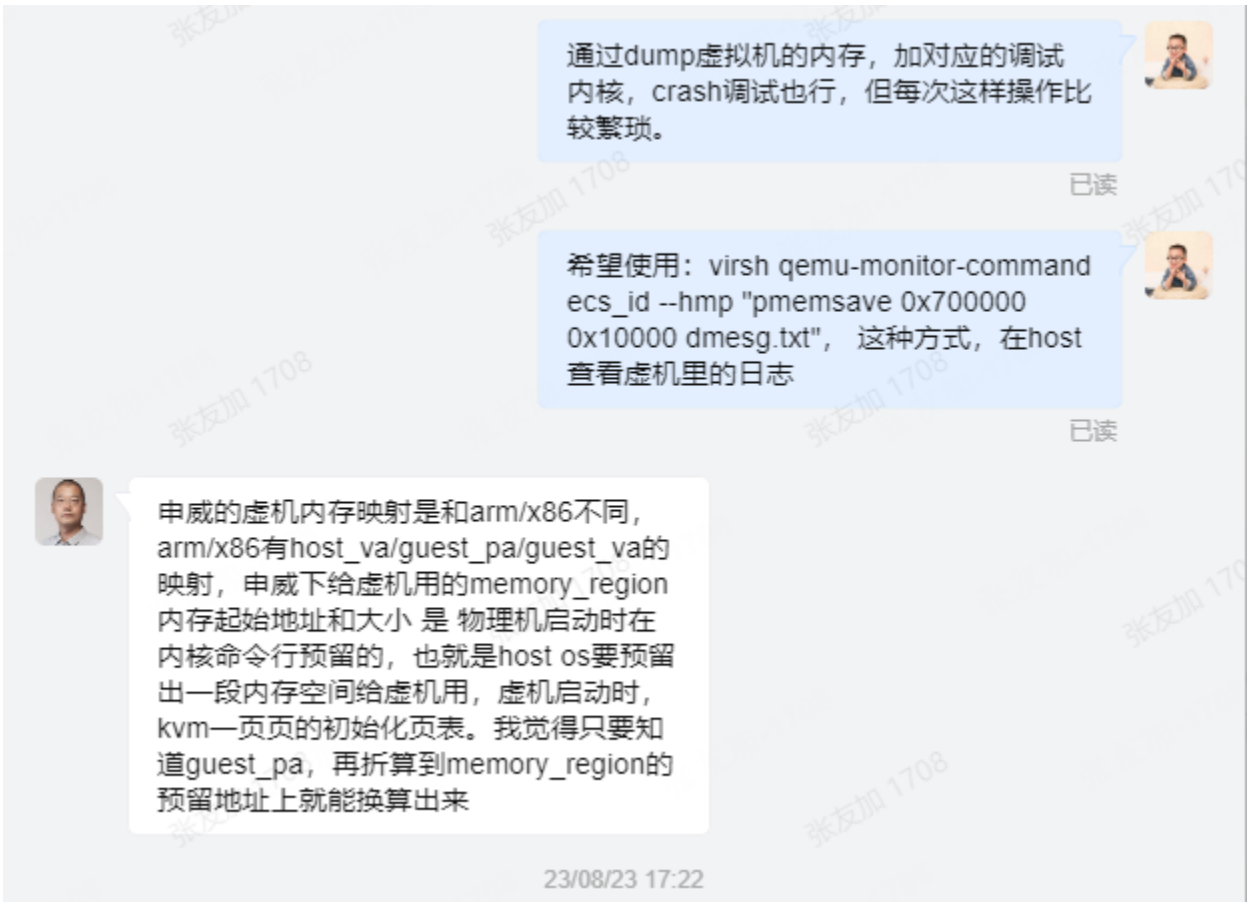
是不是想在host查看虚机里的日志?

就是第一张图中（申威适配），怎么知道dmesg缓冲区的地址是0x700000

已读

是的

> 葛长忠 23/08/23 17:11
> 是不是想在host查看虚机里的日志?




```
open_all_disk_fd  -- VM open all disk file fd
pcie_aer_inject_error [-a] [-c] id <error_status> [<tlp header> [<tlp header prefix>]] -- inject pcie aer error
                        -a for advisory non fatal error
                        -c for correctable error
                       <id> = qdev device id
                       <error_status> = error string or 32bit
                       <tlb header> = 32bit x 4
                       <tlb header prefix> = 32bit x 4
pmemsave addr size file -- save to disk physical memory dump starting at 'addr' of size 'size'
p|print /fmt expr -- print expression value (use $reg for CPU register access)
qemu-io [-d] [device] "[command]" -- run a qemu-io command on a block device
                       -d: [device] is a device ID rather than a drive ID or node name
qom-list path -- list QOM properties
qom-set path property value -- set QOM property
q|quit  -- quit the emulator
```

是的，现在就是想获取log_buf指针指向的物理内存地址（guest_pa），qemu的pmemsave接口会把memory_region中对应的内存dump到文件。

已读

23/08/23 17:30

我没有研究过申威环境下内核的va_2_pa的映射是不是一对一的。我觉得可能是一对一的。

稍等我试验一下哈

23/08/23 17:48

是一对一的

> 葛长忠 23/08/23 17:30
> 我没有研究过申威环境下内核的

我的意思是不是二者的差值是固定的，刚才没找到内核模块的编译环境。。。

23/08/24 09:41

昨天想这样获取log_buf指针指向的地址，不行。

已读

23/08/24 09:55

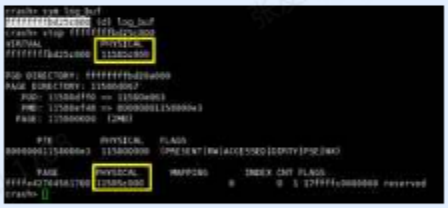
```
86492:ffff800011199a4f r __kstrtab_nf_log_buf_add
86493:ffff800011199a5e r __kstrtab_nf_log_buf_open
86494:ffff800011199a6e r __kstrtab_nf_log_buf_close
88459:ffff80001163dbe0 t log_buf_len_update
88460:ffff80001163dc60 t log_buf_len_setup
88468:ffff80001163e204 T setup_log_buf
90723:ffff8000116d5dd0 d new_log_buf_len
91216:ffff800011743ea2 d __setup_str_log_buf_len_setup
94706:ffff80001176cef0 d __setup_log_buf_len_setup
97794:ffff80011bb614c d log_buf_len
97803:ffff80011bb61e0 d log_buf
112157:ffff80001219f088 b __log_buf
[root@localhost boot]#
[root@localhost boot]# hexedit /proc/kcore

0   7F 45 4C 46  02 01 01 00  00 00 00 00  00 00 00 00  04 00 B7 00  01 00 00 00  00 00 00 00  00 00 00 00  40 00 00 00  .ELF.@...
00000024   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  40 00 38 00  2F 00 00 00  00 00 00 00  04 00 00 00  00 00 00 00  .@.8./.
00000048   88 0A 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  68 1F 00 00  00 00 00 00  00 00 00 00  .h.
0000006C   00 00 00 00  00 00 00 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 0D 10  00 80 00 00  00 00 0C 10  00 80 FF FF  .
00000090   00 00 6C ED  00 00 00 00  00 00 A6 02  00 00 00 00  00 00 A6 02  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  ..l.
000000B4   07 00 00 00  00 00 01 10  00 80 00 00  00 00 00 10  00 80 FF FF  FF FF FF FF  FF FF FF FF  00 00 00 E0  BF 7F 00 00  .
000000D8   00 00 00 E0  BF 7F 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 01 08  00 80 00 00  00 00 00 08  .
000000FC   00 80 FF FF  FF FF FF FF  FF FF FF FF  00 00 00 08  00 00 00 00  00 00 00 08  00 00 00 00  00 00 01 00  00 00 00 00  .
00000120   01 00 00 00  07 00 00 00  00 00 01 00  00 00 00 00  00 00 00 00  00 00 FF FF  00 00 00 80  00 00 00 00  00 00 00 39  .9
00000144   00 00 00 00  00 00 00 39  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 01 00  C0 FF 00 00  .9.
00000168   00 00 00 00  C0 FF FF FF  FF FF FF FF  FF FF FF FF  00 00 0F 00  00 00 00 00  00 00 0F 00  00 00 00 00  00 00 01 00  .
0000018C   00 00 00 00  01 00 00 00  07 00 00 00  00 00 02 39  00 00 00 00  00 00 01 39  00 00 FF FF  00 00 01 B9  00 00 00 00  .9.9.
000001B0   00 00 65 37  00 00 00 00  00 00 65 37  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 8B 70  ..e7......e7.p
000001D4   00 00 00 00  00 00 8A 70  00 00 FF FF  00 00 8A F0  00 00 00 00  00 00 10 04  00 00 00 00  00 00 10 04  00 00 00 00  .p.
000001F8   00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 1D 00  C0 FF 00 00  00 00 1C 00  C0 FF FF FF  FF FF FF FF  .
0000021C   FF FF FF FF  00 00 02 00  00 00 00 00  00 00 02 00  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  .
00000240   00 00 9C 74  00 00 00 00  00 00 9B 74  00 00 FF FF  00 00 9B F4  00 00 00 00  00 00 0B 00  00 00 00 00  00 00 0B 00  ...t.t.
00000264   00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 AB 74  00 00 00 00  00 00 AA 74  00 00 FF FF  .t.t....
00000288   00 00 AA F4  00 00 00 00  00 00 0E 02  00 00 00 00  00 00 0E 02  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  .
000002AC   07 00 00 00  00 00 BC 76  00 00 00 00  00 00 BB 76  00 00 FF FF  00 00 BB F6  00 00 00 00  00 00 09 00  00 00 00 00  .v.v.
000002D0   00 00 09 00  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 C7 76  00 00 00 00  00 00 C6 76  .v.v
000002F4   00 00 FF FF  00 00 C6 F6  00 00 00 00  00 00 02 00  00 00 00 00  00 00 02 00  00 00 00 00  00 00 01 00  00 00 00 00  .
00000318   01 00 00 00  07 00 00 00  00 00 F9 76  00 00 00 00  00 00 F8 76  00 00 FF FF  00 00 F8 F6  00 00 00 00  00 00 FD 04  .v.v.

                                                                                New position ? 0xffff80011bb61e0

000003A8   00 00 01 00  00 00 00 00  00 00 01 00  00 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 00 7C  .|
000003CC   00 00 00 00  00 00 FF 7B  00 00 FF FF  00 00 FF FB  00 00 00 00  00 00 01 04  03 00 00 00  00 00 01 04  03 00 00 00  .{.
000003F0   00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  00 00 01 80  7F 00 00 00  00 00 00 80  7F 00 FF FF  00 00 00 00  .
00000414   80 00 00 00  00 00 00 00  04 00 00 00  00 00 00 00  04 00 00 00  00 00 01 00  00 00 00 00  01 00 00 00  07 00 00 00  .
```

```
crash> sym log_buf
ffffffffbd25c800 (d) log_buf
crash> vtop ffffffffbd25c800
VIRTUAL           PHYSICAL
ffffffffbd25c800  11585c800

PGD DIRECTORY: ffffffffbd20a000
PAGE DIRECTORY: 11580d067
   PUD: 11580dff0 => 11580e063
   PMD: 11580ef48 => 80000001158000e3
  PAGE: 115800000  (2MB)

      PTE          PHYSICAL   FLAGS
80000001158000e3  115800000  (PRESENT|RW|ACCESSED|DIRTY|PSE|NX)

      PAGE         PHYSICAL      MAPPING       INDEX CNT FLAGS
ffffe42704561700  11585c000                0        0  1 17ffffc0000800 reserved
crash>
```

在guest里写个简单内核模块，把log_buf等几个符号的va转化为pa，看下va和pa的对应关系是不是都是一样的差值？然后看下和qemu里分配给虚机的memory_region 的 va首地址是不是有转换关系？

23/08/24 10:07

调试内核 + kdump， 使用crash工具是不是这样查找log_buf的物理地址？



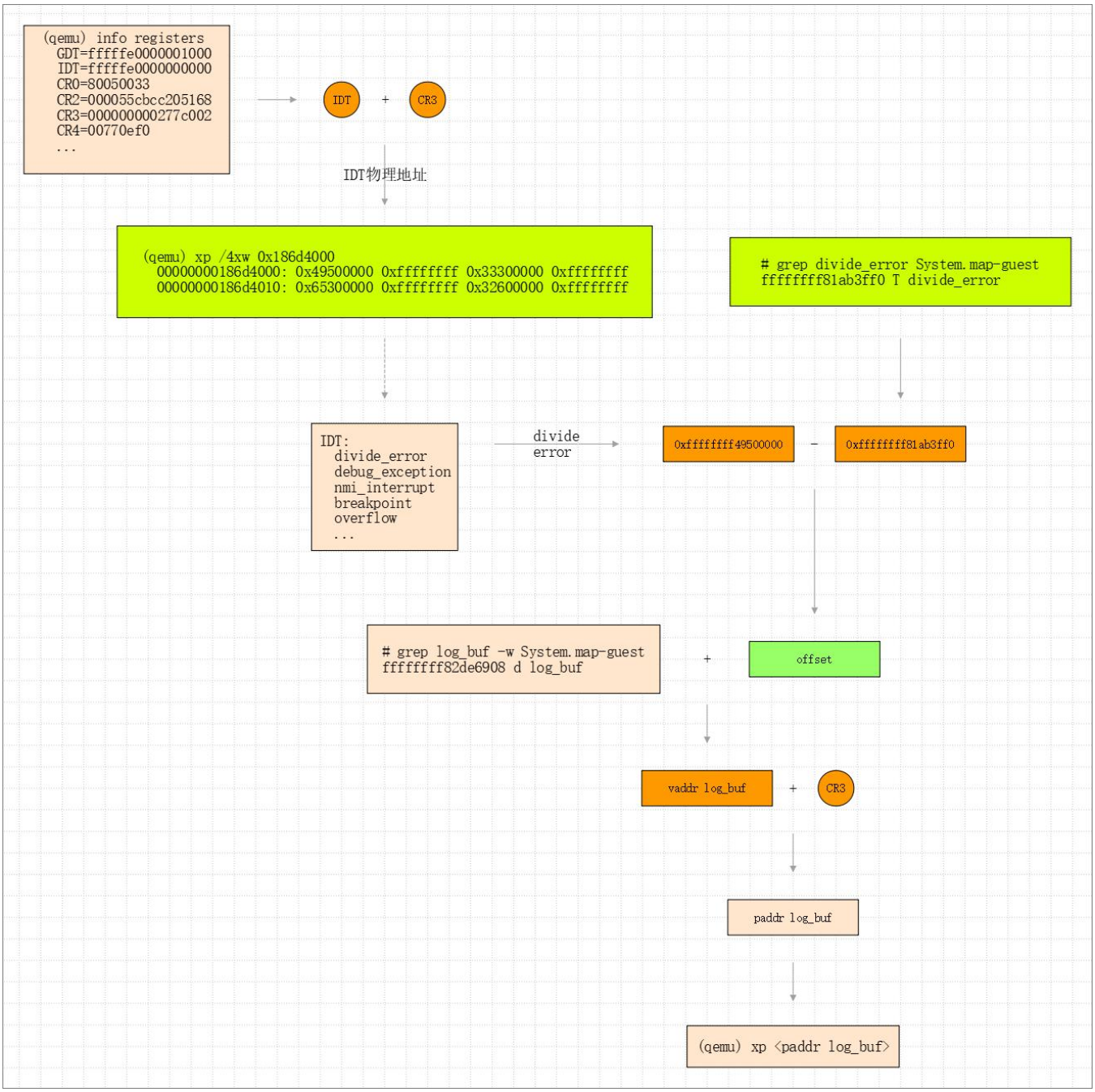
已读

我觉得应该是

23/08/24 10:15

好的


李磊-北京
受众还不少呢，晚上写个软文再
排查虚拟机问题的研发大概都有这个需求。
芦志朋-成都
现在支持 内核随机化地址?
李磊-北京
支持
加油123：现在支持 内核随机化地址?
昨天 19:05
李磊-北京


(qemu) info registers
GDT=fffffe0000001000
IDT=fffffe0000000000
CR0=80050033
CR2=000055cbcc205168
CR3=000000000277c002
CR4=00770ef0
...
IDT
+
CR3
IDT物理地址
(qemu) xp /4xw 0x186d4000
00000000186d4000: 0x49500000 0xffffffff 0x33300000 0xffffffff
00000000186d4010: 0x65300000 0xffffffff 0x32600000 0xffffffff
grep divide_error System.map-guest
ffffffff81ab3ff0 T divide_error
IDT:
divide_error
debug_exception
nmi_interrupt
breakpoint
overflow
...
divide
error
0xffffffff49500000
-
0xffffffff81ab3ff0
grep log_buf -w System.map-guest
ffffffff82de6908 d log_buf
+
offset
vaddr log_buf
+
CR3
paddr log_buf
(qemu) xp <paddr log_buf>

李磊-北京

这个是实现原理，但是代码做了部分优化，不完全这样实现的

昨天 19:19

李磊-北京

其实每次执行 qmp命令 info registers 的 CR3 寄存器有可能是不一样的，此时的 CR3的值其实是cpu正好切换到的 进程的页表基地址

李磊-北京

虽然每个进程的CR3不同，但是每个进程都拥有 内核页表的，只是不能访问

李磊-北京

所以还有个依赖就是 ， 内核和用户态 需要 共用 同张页表

李磊-北京

如果设置 内核独立 页表，是没办法的

芦志朋-成都

arm架构不行 两张页表

李磊-北京

是吗？ 不知道这个，大多数的 x86发行版都是同张页表，不然切换页表 花销还是挺大的

芦志朋-成都

每种模式 2个 页表基地址寄存器

卢亮军-武汉

X86用户态和内核态公用CR3寄存器
ARM没有CR3寄存器，但是有个对应的TTBR_EL1/TTBR_EL2, 内核态应该只用TTBR_EL2

李磊-北京

好像又有希望了，只要内核态的页表基地址是固定的就行

> rlu : X86用户态和内核态公用CR3寄存器ARM没有CR3寄存器，但是有个对应的TTBR_EL1/TTBR_EL2, 内核...

卢亮军-武汉

这里的186d4000是啥啊

李磊-北京

idt 虚拟地址 经过转换后的 物理地址，也就是 idt 的物理地址

> rlu : 这里的186d4000是啥啊

卢亮军-武汉

通过CR3指向的页表基地址，代码一级一级查到的么?

> raylee : idt 虚拟地址 经过转换后的 物理地址，也就是 idt 的物理地址

卢亮军-武汉

通过CR3指向的页表基地址，代码一级一级查到的么?

> raylee : idt 虚拟地址 经过转换后的 物理地址，也就是 idt 的物理地址

李磊-北京

是的

> rlu : 通过CR3指向的页表基地址，代码一级一级查到的么?

卢亮军-武汉

有没有在不同内存大小的VM上试下?

卢亮军-武汉

不清楚kernel的线性映射区大小，是否和VM的物理内存大小有关

李磊-北京

最小 512M, 最大4G，都试过

https://github.com/rayylee/kvm-dmesg

LBA工具与自动化测试系统交流群 (292)

YOUPLUS(LBA工具作者): [群公告... 共2条

@raylee 开发并开源代码的创新工具：kvm-dmesg (https://github.com/rayylee/kvm-dmesg), 在主机上直接获取linux虚拟机的dmesg日志，应用场景：(1)虚拟机卡死或者其它异常，无法登录guest内部; (2)使用dump虚拟机内存 + 调试内核 + crash工具调试的方法比较繁琐，另外如果虚拟机使用大规格内存，dump虚拟机内存无法实施，可以用这个工具获取到dmesg日志辅助排查问题。

曾健-天翼云智能网卡

李磊-北京

欢迎大家使用，提bug

工具刚完成开发，还在不断完善中，目前只支持x86架构，以后应该会支持arm架构。

Usage

1. **Using libvirt:**

```
./kvm-dmesg <domain_name> <system.map_path>
```

2. **Using QMP Socket:**

```
./kvm-dmesg <socket_path> <system.map_path>
```

In both commands, replace `<domain_name>` with the name of the virtual machine, `<socket_path>` with the path to the QMP socket, and `<system.map_path>` with the path to the `System.map` file for the guest kernel.

Example

```
$ ./kvm-dmesg /tmp/qmp.sock System.map-5.15.171

[    0.000000] Linux version 5.15.171 (rayylee@ubuntu-dev) (gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0, GNU l
[    0.000000] Command line: console=ttyS0
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000001ffdfffff] usable
[    0.000000] BIOS-e820: [mem 0x00000001ffe0000-0x00000001ffffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000feffc000-0x00000000feffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
[    0.000000] BIOS-e820: [mem 0x000000fd00000000-0x000000ffffffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] SMBIOS 2.8 present.
[    0.000000] DMI: QEMU Standard PC (i440FX + PIIX, 1996), BIOS rel-1.16.3-0-ga6ed6b701f0a-prebuilt.qe
[    0.000000] tsc: Fast TSC calibration using PIT
[    0.000000] tsc: Detected 3194.022 MHz processor
[    0.000569] e820: update [mem 0x00000000-0x00000fff] usable ==> reserved
[    0.000571] e820: remove [mem 0x000a0000-0x000fffff] usable
...
```

倪志广-上海兆芯虚拟化研发

这个是将虚拟机镜像mount到物理机上，然后读取dmesg文件吗?

> YOUPLUS(LBA工具作者)：@raylee 开发并开源代码的创新工具：kvm-dmesg (https://github.com/rayyle...

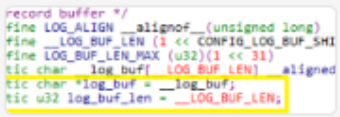
李磊-北京

不是的，是运行状态的vm，如果 crash了，想看 vm dmesg 分析原因，原理是通过 一系列的地址转换，解析vm 的memory

> 倪：这个是将虚拟机镜像mount到物理机上，然后读取dmesg文件吗?

昨天 19:47

kvm-dmesg工具的原理：是通过qemu获取guest的CR3寄存器的值，然后经过页表转换，获取到log_buf指针指向的地址，取出内核环形缓冲区中的内存数据并解析成dmesg日志。



```
record buffer */
fine LOG_ALIGN __alignof__(unsigned long)
fine __LOG_BUF_LEN (1 << CONFIG_LOG_BUF_SHI
fine LOG_BUF_LEN_MAX (u32)(1 << 31)
tic char __log_buf[__LOG_BUF_LEN] __aligned
tic char *log_buf = __log_buf;
tic u32 log_buf_len = __LOG_BUF_LEN;
```

昨天 20:21

倪志广-上海兆芯虚拟化研发

明白啦👍👍👍

qm> dump-guest-memory -p -l /root/vm.dump 【dump-guest-memory当前guest的全部内存为vmcore文件，-l表示以lzo方式压缩，此压缩方式能够被crash识别】

qemu的qmp接口：dump-guest-memory支持内存压缩

昨天 17:47

李磊-北京

-z 参数是支持压缩的，但是有2个问题。1.需要配合带有符号表的vmlinux。2.如果是io error 导致的vm crash，qemu无法落盘，dump-guest-memory会卡主无法使用

> YOUPLUS(LBA工具作者) : qemu的qmp接口：dump-guest-memory支持内存压缩

昨天 18:08

宋牧春-字节内核

不是你理解的

> arch : kdump-tools做了这个事情

宋牧春-字节内核

不是你理解那样

卷心菜废狗：额，makedumpfile已经做了呀

宋牧春-字节内核

在host上，无法进入guest，需要从 host dump guest

昨天 18:27

卷心菜废狗

那就在host上针对已有的vmcore，再用makedumpfile把dump level设置更高跑一遍就好了呀(

somuch：在host上，无法进入guest，需要从 host dump guest

宋牧春-字节内核

700G的虚拟机，本地用磁盘空间可能没有700G，这就是我遇到的问题

卷心菜废狗：那就在host上针对已有的vmcore，再用makedumpfile把dump level设置更高跑一遍就好了呀(

宋牧春-字节内核

但是 700G 虚拟机内存只用了 20%

在github的readme上加一节，描述实现原理，把这张图放上去吧。

raylee： 

访问者直接读源码，可能比较难理解实现原理，有了这张图就很容易理解了。

李磊-北京

可以

力哥的铁粉

牛掰，支持

IT狂热者

太牛X了，别人都是打工，你这是创作了

11:12

英雄本色

@YOUPLUS(LBA工具作者) 内核地址默认随机化了吧？怎么处理的？

已经支持了，有问题可以单独和作者@raylee 讨论哈。

英雄本色：@YOUPLUS(LBA工具作者) 内核地址默认随机化了吧？怎么处理的？

李磊-北京

支持地址随机化，支持5.10以后的printk无锁ring_buf

英雄本色：@YOUPLUS(LBA工具作者) 内核地址默认随机化了吧？怎么处理的？


BiscuitOS Community (499)
宋牧春-字节内核
readme加一节编译需要的包
宋牧春-字节内核
libvirt-dev
李磊-北京
，目前还是需要依赖这个，后面计划使用dlopen
somuch : libvirt-dev
11:27
以后编译时不依赖libvirt库，只是以libvirt方式使用kvm-dmesg，运行时dlopen调用libvirt库。
g libvirt:
vm-dmesg <domain_name> <system.map_
而以QMP方式使用kvm-dmesg，就只需要C库就行了，不依赖libvirt库。
QMP Socket:
vm-dmesg <socket_path> <system.map_
宋牧春-字节内核
怎么打开 debug？第一次测试，没捞出来 dmesg
@raylee 帮忙看一下
somuch : 怎么打开 debug？第一次测试，没捞出来 dmesg

```
root@centos83-dev /h/g/k/kvm-dmesg (master)# git diff
diff --git a/log.c b/log.c
index 45dcadb..517b4a1 100644
--- a/log.c
+++ b/log.c
@@ -19,7 +19,7 @@

 #include "log.h"

-int loglevel = LOGLEVEL_WARNING;
+int loglevel = LOGLEVEL_DEBUG;

 void log_init(int level)
 {
root@centos83-dev /h/g/k/kvm-dmesg (master)# ./kvm-dmesg ../System.map-4.18.0-348.7.1.el8_5.x86_64 /tmp/qmp.sock
Version 0.1.0

Guest: /tmp/qmp.sock
System.map: ../System.map-4.18.0-348.7.1.el8_5.x86_64
[Debug] (main.c) calc_kaslr_offset:167: kaslr_offset: idtr=fffffe0000000000
[Debug] (main.c) calc_kaslr_offset:168: kaslr_offset: pgd=7fecc000
[Debug] (main.c) calc_kaslr_offset:169: kaslr_offset: idtr(phys)=2e64000
[Debug] (main.c) calc_kaslr_offset:170: kaslr_offset: divide_error(vmcore): ffffffff81a00cf0
[Debug] (main.c) calc_kaslr_offset:174: kaslr_offset: kaslr_offset=0
[Debug] (main.c) calc_kaslr_offset:175: kaslr_offset: phys_base    =0
[Debug] (main.c) dump_variable_length_record_log:290: log_buf: ffffffff82ee0744
[Debug] (main.c) dump_variable_length_record_log:291: log_buf_len: 1048576
[Debug] (main.c) dump_variable_length_record_log:292: log_first_idx: 0
[Debug] (main.c) dump_variable_length_record_log:293: log_next_idx: 29664
[    0.000000] Linux version 4.18.0-348.7.1.el8_5.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 8.5.0 20210514 (Re
1
[    0.000000] Command line: nokaslr console=ttyS0
[    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x200: 'Protection Keys User registers'
[    0.000000] x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
[    0.000000] x86/fpu: xstate_offset[9]:  832, xstate_sizes[9]:    8
[    0.000000] x86/fpu: Enabled xstate features 0x207, context size is 840 bytes, using 'compacted' format.
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000000 7ffdffff] usable
[    0.000000] BIOS-e820: [mem 0x00000000 7ffe0000-0x00000000 7fffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000feffc000-0x00000000feffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] SMBIOS 2.8 present.
[    0.000000] DMI: Red Hat KVM, BIOS 1.13.0-2.module_el8.5.0+746+bbd5d70c 04/01/2014
[    0.000000] Hypervisor detected: KVM
```

需要改代码开启debug，不过开了日志也不太多

李磊-北京

李磊-北京

内核版本是什么呢？需要正确的system.map，还有就是是qmp socket，不是monitor socket。

somuch : 怎么打开 debug？第一次测试，没捞出来dmesg

12:37

宋牧春-字节内核

现在的next分支，qmp 可以工作

raylee : 内核版本是什么呢？需要正确的system.map，还有就是是qmp socket，不是monitor socket。

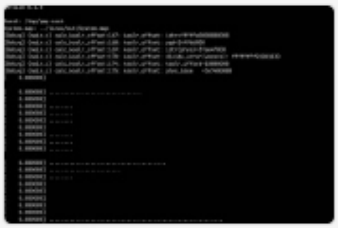
李磊-北京

太新的代码可能也不行，实测最高版本5.15，盲猜可能是内核支持了 实时特性，又修改了printk_buf结构

somuch : 现在的next分支，qmp 可以工作

```
Version 0.1.0

Guest: /tmp/qmp-sock
System.map: ../linux/out/System.map
[Debug] (main.c) calc_kaslr_offset:167: kaslr_offset: idtr=fffffe0000000000
[Debug] (main.c) calc_kaslr_offset:168: kaslr_offset: pgd=244966000
[Debug] (main.c) calc_kaslr_offset:169: kaslr_offset: idtr(phys)=3fae4f000
[Debug] (main.c) calc_kaslr_offset:170: kaslr_offset: divide_error(vmcore): ffffffff92001030
[Debug] (main.c) calc_kaslr_offset:174: kaslr_offset: kaslr_offset=10000000
[Debug] (main.c) calc_kaslr_offset:175: kaslr_offset: phys_base   =3e7400000
[    0.000000]

[    0.000000] ............................
[    0.000000]
[    0.000000] ........
[    0.000000] ........
[    0.000000] ........
[    0.000000]
[    0.000000] ........
[    0.000000] ........
[    0.000000] ........

[    0.000000] ..................................
[    0.000000] ......................
[    0.000000] ........
[    0.000000]
[    0.000000]
[    0.000000]
[    0.000000]
[    0.000000]
[    0.000000] ..................................................
```

宋牧春-字节内核



宋牧春-字节内核

确实乱码

宋牧春-字节内核

估计格式变了

李磊-北京

是的，我测试最新的master也是这样，看输出，可能索引没问题，但是buf的start addr错了

somuch : 估计格式变了

14:09

宋牧春-字节内核

5.15好使的

14:16

李磊-北京

嗯，最高版本测试过5.15，最低版本测试过3.10

somuch : 5.15好使的

commit: f244b4dc53e520d4570b2610436aba0593ce6f55 Author: Petr Mladek <pmladek@suse.com>

```
diff --git a/kernel/printk/printk_ringbuffer.h b/kernel/printk/printk_ringbuffer.h
index 73cc80e01cef..18cd25e489b8 100644
--- a/kernel/printk/printk_ringbuffer.h
+++ b/kernel/printk/printk_ringbuffer.h
@@ -75,6 +75,7 @@ struct prb_desc_ring {
        struct printk_info      *infos;
        atomic_long_t           head_id;
        atomic_long_t           tail_id;
+       atomic_long_t           last_finalized_id;
 };

 /*
```


```
[root@cclinux2209-4444 boot]# cat System.map-5.15.67-10.cl9.aarch64 | wc -l
115111
```


```
【主机上连接虚拟机的控制台】
【1】虚拟机内核参数：console=ttyS0,115200,8n1 console=tty0
kernel /bzImage root=/dev/sda2 nmi_watchdog=1 console=ttyS0,115200,8n1 console=tty0 quiet

【2】qemu参数给虚拟机配置虚拟串口
-serial unix:/var/run/serial_1007908691446_0.sock,server,nowait

-chardev pty,id=charserial0 \
-device '{"driver":"isa-serial","chardev":"charserial0","id":"serial0","index":0}' \

【3】主机节点上通过virtctl console连接虚拟机控制台
[root@lhcell1 ~]# virtctl console ecs-yj8vezit8uy90q                    【virtctl console -n tenant-220005041600420 ecs-yj8vezit8uy90q】
```

英雄本色

/proc/kallsyms也可以吧，

宋牧春-字节内核

你还想登录客户机器？

英雄本色

，我错了，

这个工具的创新就是：无法登录虚拟机的情况下，获取虚拟机的dmesg日志

英雄本色：/proc/kallsyms也可以吧，

英雄本色

恩恩，

宋牧春-字节内核

这种情况不是 console log 吗

YOUPLUS(LBA工具作者)：这个工具的创新就是：无法登录虚拟机的情况下，获取虚拟机的dmesg日志

1. 使用console log，虚拟机内核是要配置串口参数的，很多linux发行版默认没有配置；2. 虚拟机卡死了，很多情况下通过console log也可能获取不到信息。

somuch：这种情况不是 console log 吗

```
Version 0.1.0

Guest: /tmp/qmp.sock
System.map: /home/rayylee/github/linux/System.map
[    0.000000] Linux version 6.12.0-03657-g43fb83c17ba2 (rayylee@ubuntu-dev) (gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0, GNU ld (GNU B
u Nov 21 18:45:32 CST 2024
[    0.000000] Command line: console=ttyS0
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x000000001ffdffff] usable
[    0.000000] BIOS-e820: [mem 0x000000001ffe0000-0x000000001fffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
[    0.000000] BIOS-e820: [mem 0x000000fd00000000-0x000000ffffffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] APIC: Static calls initialized
[    0.000000] SMBIOS 2.8 present.
[    0.000000] DMI: QEMU Standard PC (i440FX + PIIX, 1996), BIOS rel-1.16.3-0-ga6ed6b701f0a-prebuilt.qemu.org 04/01/2014
[    0.000000] DMI: Memory slots populated: 1/1
[    0.000000] tsc: Fast TSC calibration using PIT
[    0.000000] tsc: Detected 3194.121 MHz processor
[    0.034885] e820: update [mem 0x00000000-0x00000fff] usable ==> reserved
[    0.035419] e820: remove [mem 0x000a0000-0x000fffff] usable
[    0.036002] last_pfn = 0x1ffe0 max_arch_pfn = 0x400000000
```


你那新项目要不要开个推广讲解视频啊?
10:02
可以呀，这个项目不是我的哈，是团队中的一个成员完全实现的，虽然是我最早提出的想法(去年8月我和两个资深OS专家---一个从事OS研发24年，另一个从事OS研发17年，讨论了基本的实现思路，但没有完全想好实现的方案，我把讨论的内容转发到团队群里，然后这位团队成员结合crash工具代码完成了这个很好的实现方案)。
👍👍👍
这个工具短小精悍
嗯，很新颖，很创新。
我觉得是个好东西
多推广推广，多加星🤭
我跟作者说一下，他有空的时候，录个演示讲解视频来推