qemu网络虚拟化之数据流向分析一

邮箱：zhunxun@gmail.com

< 2020年5月 >

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索

找找看

谷歌搜索

PostCategories

C语言(2)
IO Virtualization(3)
KVM虚拟化技术(26)
linux 内核源码分析(61)
Linux日常应用(3)
linux时间子系统(3)
qemu(10)
seLinux(1)
windows内核(5)
调试技巧(2)
内存管理(8)
日常技能(3)
容器技术(2)
生活杂谈(1)
网络(5)
文件系统(4)
硬件(4)

PostArchives

2018/4(1)
2018/2(1)
2018/1(3)
2017/12(2)
2017/11(4)
2017/9(3)
2017/8(1)
2017/7(8)
2017/6(6)
2017/5(9)
2017/4(15)
2017/3(5)
2017/2(1)
2016/12(1)
2016/11(11)
2016/10(8)
2016/9(13)

ArticleCategories

时态分析(1)

Recent Comments

1. Re:virtio前端驱动详解
我看了下，Linux-4.18.2中的vp_notify()函数。bool vp_notify(struct virtqueue *vq){ /* we write the queue's selec...
--Linux-inside
2. Re:virtIO之VHOST工作原理简析

插曲：

今天下午欣喜的想写点关于qemu网络部分的功能，但是中途出现了点小插曲，电脑被某人搞得死机了，并且文章也没有保存。结果，，，就只能重新写了！！所以这里强烈建议开发团队提供自动保存的功能！

言归正传，前段时间自己写过关于Linux 内部网桥的实现原理以及数据包从物理网卡到达Linux网桥进行转发，再到Tap设备的流程。从qemu网络虚拟化整体框架来看，这部分只能算是前端，就像是数据到达了交换机，还没有从交换机到达具体客户机。这么比喻也不是很贴切，因为Linux网桥就好比一个交换机了，但是这么说大题上也不伤大雅。今天主要分析下数据是如何经过qemu到达客户机的。

这部分内容准备分成两部分，第一部分介绍主要的数据结构以及数据结构之间的关系，第二部分结合源代码分析其具体实现。

下面看涉及到的主要数据结构：

```
struct NetClientState {
    NetClientInfo *info;
    int link_down;
    QTAILQ_ENTRY(NetClientState) next;
    NetClientState *peer;
    NetQueue *incoming_queue;
    char *model;
    char *name;
    char info_str[256];
    unsigned receive_disabled : 1;
    NetClientDestructor *destructor;
    unsigned int queue_index;
    unsigned rxfilter_notify_enabled:1;
};
```

首先要说的是一个叫做NetClientInfo的结构，它并不对应于具体的实体，而在笔者看来倒像是一个逻辑点，Hub和nic,Hub和Tap都是通过此结构联系的，所以先介绍这个结构，其中的peer指针就指向对等实体的NetClientState结构，incoming_queue就是该点的接收数据的队列，在发送数据的时候要判断对端的incoming_queues是否可用，可用才可以发送数据。其他的字段这里暂时忽略。queue_index好像和网卡的多队列特性相关，暂时不太清楚。next作为一个节点加入到全局的net client链表中。

```
struct NetHub {
    int id;
    QLIST_ENTRY(NetHub) next;
    int num_ports;
    QLIST_HEAD(, NetHubPort) ports;
};
```

NetHub结构，这里就表示一个Hub，实体名字为集线器，其功能就是把收到的数据包从接收端口之外的所有端口转发出去，qemu内部的vlan就是通过这个Hub实现的，Hub id就表示vlan id,num_port是端口的数量，一个Hub下的所有端口通过port连接起来，ports是链表头。

```
typedef struct NetHubPort {
    NetClientState nc;
    QLIST_ENTRY(NetHubPort) next;
    NetHub *hub;
    int id;
} NetHubPort;
```

NetHubPort代表一个Hub中的一个端口，NetClientState表示本端口的逻辑连接点，是一个内嵌结构（非指针），next指向下一个port,Hub 指向所属的Hub对象。

再问一个问题，从设置ioeventfd那个流程来看的话是guest发起一个IO，首先会陷入到kvm中，然后由kvm向qemu发送一个IO到来的event，最后IO才被处理，是这样的吗?

--Linux-inside

3. Re:virtIO之VHOST工作原理简析

你好。设置ioeventfd这个部分和guest里面的virtio前端驱动有关系吗?
设置ioeventfd和virtio前端驱动是如何发生联系起来的？谢谢。

--Linux-inside

4. Re:QEMU IO事件处理框架

良心博主，怎么停跟了，太可惜了。

--黄铁牛

5. Re:linux 逆向映射机制浅析

小哥哥520脱单了么

--黄铁牛

Top Posts

1. 详解操作系统中断(21154)
2. PCI 设备详解一(15806)
3. 进程的挂起、阻塞和睡眠(13713)
4. Linux下桥接模式详解一(13465)
5. virtio后端驱动详解(10538)

推荐排行榜

1. 进程的挂起、阻塞和睡眠(6)
2. 为何要写博客(2)
3. virtIO前后端notify机制详解(2)
4. 详解操作系统中断(2)
5. qemu-kvm内存虚拟化1(2)

```
struct NICInfo {
    MACAddr macaddr;
    char *model;
    char *name;
    char *devaddr;
    NetClientState *netdev;
    int used;         /* is this slot in nd_table[] being used? */
    int instantiated; /* does this NICInfo correspond to an instantiated NIC? */
    int nvectors;
};
```

NICInfo代表一个虚拟网卡，也就是和客户机相关的，这里姑且称之为客户端，通过NICInfo中的netdev指针，NICInfo和Hub ports相连接。qemu中有一个全局的数组来表示NICInfo的空间

NICInfo nd_table[MAX_NICS];.

```
typedef struct NICState {
    NetClientState *ncs;
    NICConf *conf;
    void *opaque;
    bool peer_deleted;
} NICState;
```

这是和虚拟网卡相关的一个结构，ncs被初始化成一个指针指向一个NetClientState数组，conf指向网卡对应的NICConf结构，该结构记录了网卡的硬件信息，其中有个重要的peers,是一个内嵌结构，结合下面的结构展示，peers->ncs是一个指向NetClientState数组的指针，该结构的初始化时在最初根据命令行参数设置的，在网卡初始化的时候前面两个数组正好作为两个逻辑点需要其中的对应项建立关联。**其实NICState 的ncs指向的数组项应该是1，这里的项数是根据NICConf中的queues确定的，而queue在NICConf初始化的时候并没有被设置，笔者猜想这里应该默认被初始化成0了。**

。

```
typedef struct NICConf {
    MACAddr macaddr;
    NICPeers peers;
    int32_t bootindex;
    int32_t queues;
} NICConf;
```

这是网卡的配置结构，各个字段的意义根据名称就显露无疑。macaddr为网卡的物理MAC，peers指向一个NICPeers,该结构记录了对等的NetClientState结构数组，queues记录了网卡的队列个数。

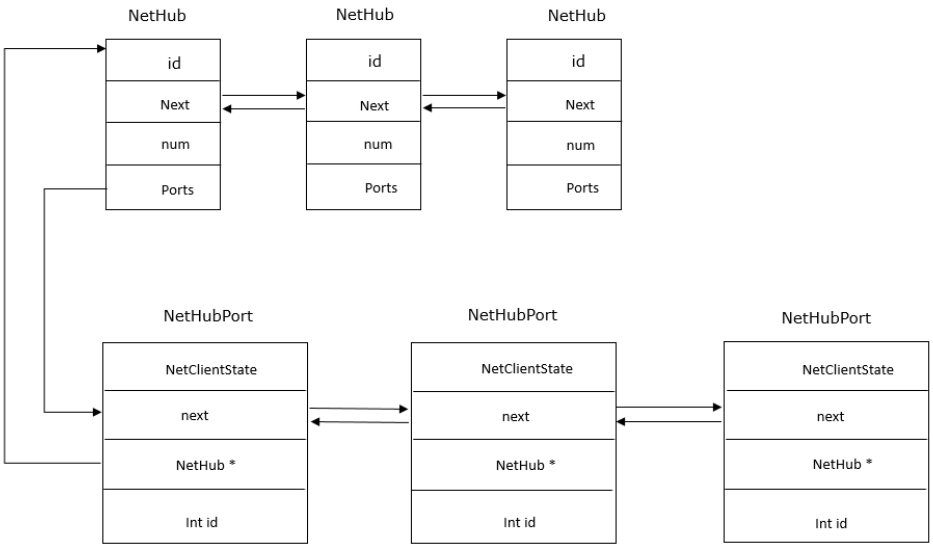
```
typedef struct NICPeers {
    NetClientState *ncs[MAX_QUEUE_NUM];
} NICPeers;
```

```
typedef struct TAPState {
    NetClientState nc;
    int fd;
    char down_script[1024];
    char down_script_arg[128];
    uint8_t buf[NET_BUFSIZE];
    bool read_poll;
    bool write_poll;
    bool using_vnet_hdr;
    bool has_ufo;
    bool enabled;
    VHostNetState *vhost_net;
    unsigned host_vnet_hdr_len;
} TAPState;
```

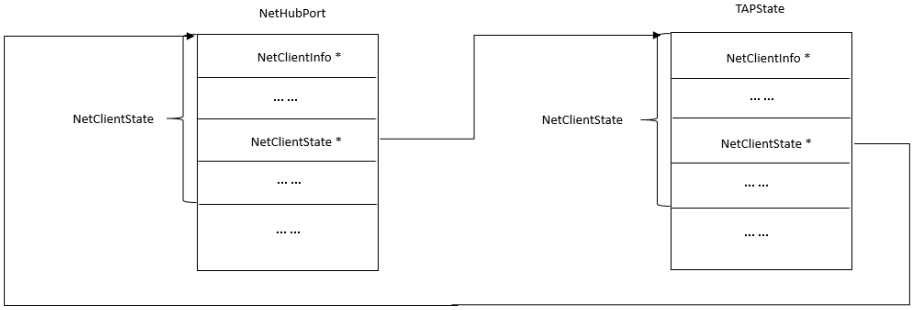
TAPState代表一个tap设备，数据从Linux端口出来发送给tap设备，而用户程序需要从tap设备读写数据（对应于实际意义上的接收和发送），TAPState结构和NetHubPorts结构一样，都内嵌了NetClentStat结构。Tap设备和Hub就是通过这两个NetClentState相关联。

而关于Hub和网卡NIC的关联，虽然有些博文也提到了，但是笔者总感觉说的有些含糊，并且经过笔者自己分析代码，这部分也不是很清楚，所以这部分内容具体就留在下篇结合源代码在说，对这块内容熟悉的朋友也希望多多指点，大家一起学习！

NetHub和NetHubPort关系图：



TAPState和NetHubPort 关系图：



分类：linux 内核源码分析

jack.chen
关注 - 12
粉丝 - 44
+加关注

0 0

« 上一篇： virt-manager 操作 kvm虚拟机中鼠标不同步的问题
» 下一篇： qemu网络虚拟化之数据流向分析二

相关博文：
· Linux下桥接模式详解一
· QEMU网络配置
· 虚拟机体验之 QEMU 篇
· 一分钟看懂Docker的网络模式和跨主机通信
· Wireshark网络分析工具（一）
» 更多推荐...

96秒100亿！哪些“黑科技”支撑全球最大流量洪峰?

最新 IT 新闻:
· 腾讯在列！微软宣布超140家工作室为Xbox Series X开发游戏
· 黑客声称从微软GitHub私人数据库当中盗取500GB数据
· IBM开源用于简化AI模型开发的Elyra工具包
· 中国网民人均安装63个App：腾讯系一家独大
· Lyft颁布新规：强制要求乘客和司机佩戴口罩
» 更多新闻...

以马内利