201611071631

整体思路，先修改下表中除快照相关的函数，修改过的用红色字体标注，正在修改的用黄色字体标注

|  |  |
| --- | --- |
| 成员 | 说明 |
| format\_name | 虚拟块设备名称 |
| instance\_size | 虚拟块设备特有状态数据结构体大小，用于确定BlockDriverState结构体中opaque指针指向的内存空间的大小，即BDRVIrowState结构体的大小 |
| **bdrv\_probe** | 指向irow\_probe的函数指针，用于判断虚拟机磁盘镜像的格式 |
| **bdrv\_open** | 指向irow\_open的函数指针，用于打开虚拟机磁盘镜像 |
| **bdrv\_read** | 指向irow\_read的函数指针，用于同步读取虚拟机磁盘镜像 |
| **bdrv\_write** | 指向irow\_write的函数指针，用于同步写入虚拟磁盘镜像 |
| **bdrv\_close** | 指向irow\_close的函数指针，用于关闭虚拟机磁盘镜像 |
| **bdrv\_create** | 指向irow\_create的函数指针，用于创建虚拟机磁盘镜像 |
| **bdrv\_flush** | 指向irow\_flush的函数指针，用于同步将缓存中的数据写入虚拟机磁盘镜像 |
| **bdrv\_aio\_readv** | 指向irow\_aio\_readv的函数指针，用于异步读取虚拟机磁盘镜像 |
| **bdrv\_aio\_writev** | 指向irow\_aio\_writev的函数指针，用于一部写入虚拟机磁盘镜像 |
| **bdrv\_aio\_flush** | 指向irow\_aio\_flush的函数指针，用于异步将缓存中的数据写入虚拟机磁盘镜像 |
| bdrv\_truncate | 指向irow\_truncate的函数指针，用于截断虚拟机磁盘镜像 |
| **bdrv\_getlength** | 指向irow\_getlength的函数指针，用于获取虚拟机磁盘镜像逻辑大小 |
| bdrv\_snapshot\_create | 指向irow\_snapshot\_create的函数指针，用于虚拟机磁盘快照的创建 |
| bdrv\_snapshot\_goto | 指向irow\_snapshot\_goto的函数指针，用于虚拟机磁盘快照的回滚 |
| bdrv\_snapshot\_delete | 指向irow\_snapshot\_delete的函数指针，用于虚拟机磁盘快照的删除 |
| bdrv\_snapshot\_list | 指向irow\_snapshot\_list的函数指针，用于列出所有的虚拟机磁盘快照的信息 |
| **bdrv\_get\_info** | 指向irow\_get\_info的函数指针，用于获取虚拟机磁盘的相关信息 |
| bdrv\_save\_vmstate | 指向irow\_save\_vmstate的函数指针，用于保存在线快照的虚拟机状态 |
| bdrv\_load\_vmstate | 指向irow\_load\_vmstate的函数指针，用于读取在线快照的虚拟状态 |
| create\_options | 指向irow\_create\_options的QEMUOptionParameter结构体指针，用于表明在创建虚拟机磁盘镜像时合法的参数 |

问题：

Map\_file文件需要缓存到内存中么？缓存在内存中的话，大小最大为多少呢？

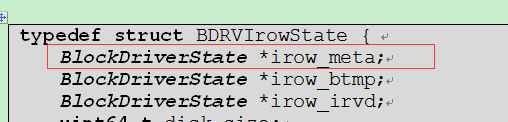
用户看到的虚拟磁盘空间，也就是static int irow\_read(BlockDriverState \*bs, int64\_t sector\_num, uint8\_t \*buf, int nb\_sectors)这个函数里面对应的sector\_num指向的虚拟磁盘空间大小一般是多少，由文件系统预先指定好的吗？和用户指定的虚拟机镜像大小没有实际联系吧？

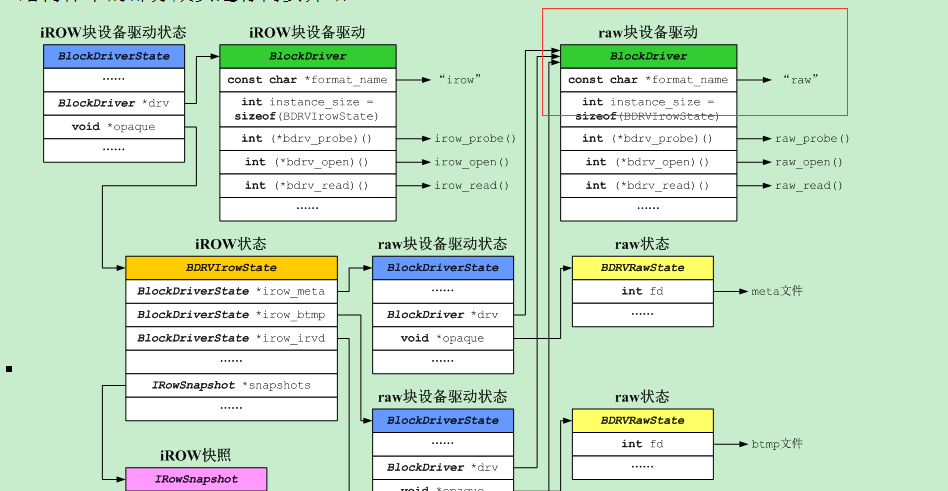
我设计的map数组用来解决写时重定向的映射问题貌似只能解决同步读写事件，异步的貌似不行

研究一下raw和qcow2的磁盘格式，，我设计的磁盘格式主要根据刘寒青的论文来的，所以到时候毕业论文得参考刘寒青的写了

刘寒青将meta, btmp 和ivrd文件指定为raw格式，但是raw不支持稀疏模式，，，，但是raw没有cluster的概念。。。所以不懂刘寒青这些是怎么考虑的,创建ivrd文件时用了如下命令，据说这样是创建稀疏文件的方法

Fallocate() ftruncate()





怎样将irow\_meta等指向了raw格式？

没有做back\_file的处理

为什么要以cluster为基本单位呢？为啥不直接以sector为单位呢

**qrow\_create**里面的back\_file怎么处理？对于参数的判断，如镜像大小，簇大小的判断是否需要做到更严格的判断

**qrow\_open** 参数flags是干嘛的？我的记录重放功能添加到哪里去？目前修改的内容里面没有添加记录重放的功能

//读取元数据信息 部分没有按照irow里面的irow\_open\_meta()的方式打开读取不造会不会有问题

//Map 数组需不需要缓存，当map数组长度很大的时候，是不是全部打开很耗内存，需要缓存部分？？？

**qrow\_close**中有元数据更新和map的写操作，但是刘寒青的论文里说kvm在关闭虚拟机时不会调用虚拟机虚拟块设备驱动的虚拟磁盘关闭函数，而是直接结束了相应的qemu-kvm进程，因此iROW在关闭虚拟机磁盘时，没有对虚拟机磁盘进行任何写操作。如果iROW在其关闭函数中对虚拟机磁盘进行某些写操作（如更新元数据、写入校验和等），那么这些操作在虚拟机关机时并不会被执行，因而有可能影响虚拟机磁盘的完整性。所以iROW将meta文件、btmp文件的更新操作安排在其它函数中。所以这一点要后期处理一下

qrow\_aio\_readv先按照刘寒青的写

qrow\_aio\_writev处理了一下sector\_num

bdrv\_aio\_flush 没有对map文件进行处理

qrow\_read 对于空字符串的处理不知道对不对，就是读到的虚拟扇区的内容为空时直接跳过

调试qemu版本 0.12.5

虚拟机运行注意事项：

qemu-system-x86\_64 -smp 2,cores=2 -m 2G -hda ubuntu.img --enable-kvm -vnc :0

qrow\_irow\_v2.c里面

qrow\_get\_info 肿么办