[Qcow2镜像格式解析](http://blog.chinaunix.net/xmlrpc.php?r=blog/article&uid=26299634&id=4326024)

Qcow2镜像格式是qemu支持的磁盘镜像格式之一。qcow2的表现形式为在一个文件中模拟一个固定大小的块设备。对与qcow2格式，相对于raw格式来说，有几个优点：

1.更小的文件大小，即使是不支持holes的文件系统也可以（这样的话，ls跟du看到的就一样了)；

2.Copy-on-write的支持；

3.快照的支持，可以维护多个快照；

4.基于zlib的压缩；

5.AES加密

qemu-img命令可以用来创建qcow2镜像，或者将qcow2文件转换成raw格式文件，等其它功能：

$> qemu-img create -f qcow2 test.qcow2 4G

Formating 'test.qcow2', fmt=qcow2, size=4194304 kB

$> qemu-img convert test.qcow2 -O raw test.img

qcow2 Header

每一个qcow2文件都是以一个固定格式的数据头开始的，其以大端模式存放，格式如下：

typedef struct QCowHeader {

uint32\_t magic;

uint32\_t version;

uint64\_t backing\_file\_offset;

uint32\_t backing\_file\_size;

uint32\_t cluster\_bits;

uint64\_t size; /\* in bytes \*/

uint32\_t crypt\_method;

uint32\_t l1\_size;

uint64\_t l1\_table\_offset;

uint64\_t refcount\_table\_offset;

uint32\_t refcount\_table\_clusters;

uint32\_t nb\_snapshots;

uint64\_t snapshots\_offset;

} QCowHeader;

•头4个字节包含了字符'Q', 'F', 'I'，并以0xfb结尾；

•之后的4个字节包含了这个文件所用的格式版本，当前存在两种版本的格式，版本1和版本2。在本文，我们讨论的是版本2，即qcow2。版本1将在本文最后做简要介绍；

•backing\_file\_offset字段给出相对于qcow2文件起始位置的偏移，指出一个字符串的位置，该字符串为backing file文件的绝对路径。由于该字符串不是以'\0'结束，所以backing\_file\_size指出字符串的长度。如果当前镜像是一个copy-on-write镜像，则存在backing file文件，否则没有；

•cluster\_bits字段，决定了怎样映射镜像偏移地址到文件偏移地址，其决定了在一个簇中，将拿偏移地址的多少位(低位)来作为索引。L2表占据一个单独的簇，包含若干8字节的项，cluster\_bits最少用3bits作为L2表的索引。L2表的详细介绍，见下一节的2级索引；

•size字段指示镜像以块设备呈现时的大小，单位字节；

•crypt\_method只有两种值，0表示没有加密，1表示采用了AES加密；

•l1\_size字段指示了在L1表中，可用的8字节项的个数，l1\_table\_offset字段给出了L1 table的文件偏移；

•相似的，refcount\_table\_offset字段给出了refcount table的文件偏移，refcount\_table\_clusters字段描述了refcount table大小(单位为clusters);

•nb\_snapshots字段给出了当前镜像中有多少个快照，snapshots\_offset字段给出了QCowSnapshotHeader headers的文件偏移，每个快照都会有这样一个header。

一个典型的镜像文件，其布局如下：

•一个header， 如上描述；

•在下一个簇开始，存放L1 table；

•refcount table，仍然是簇对齐的；

•一个或者多个的refcount blocks；

•Snapshot headers，第一个header要求簇对齐，之后的header要求8字节对齐；

•L2 tables，每一个table占据一个单独的cluster；

•Data clusters。

2级索引

对于qcow2格式，块设备的内容被保存在cluster中。每个cluster包含了若干个sector，每个sector有512个字节。

为了通过给定的镜像地址找到指定的cluster，必须经过1级表和2级表的转换。例如，假设cluster\_bits为12，则地址会被切分成如下三份：

•低12位用来定位一个4Kb的簇内偏移；

•之后的9位为一个512项的数组的偏移，每一项为一个8字节的文件偏移，即L2 table。 这里的9位是这么算出来的， l2\_bits = cluster\_bits - 3，L2 table是一个单独的包含若干8字节项的cluster；

•剩下的43位为另外一个8字节的文件偏移的数组的偏移，即L1 table。

注意，L1 table的最小值，可以通过给定磁盘镜像的大小来计算，公式如下：

l1\_size = round\_up(disk\_size / (cluster\_size \* l2\_size), cluster\_size)

总的来说，为了将磁盘镜像地址映射到镜像文件偏移，需要经历以下几步：

1.通过qcow2 header中的l1\_table\_offset字段获取L1 table的地址；

2.使用高（64 - l2\_bits - cluser\_bits)位的地址来索引L1 table，L1 table是一个数组，数组元素是一个64位的数；

3.通过L1 table中的表项来获取L2 table的地址；

4.通过L2 table中的表项来获取cluster的地址；

5.剩余的cluster\_bits位来索引cluster内的位置。

如果找到的L1 table或L2 table的地址偏移为0，则表示磁盘镜像对应的区域尚未分配。

还需要注意的是，L1或L2 table的地址的高2位被保留下来，拿来置"copied"和"compressed"标记。具体细节见下节"引用计数"。

引用计数

每一个cluster都有一个引用计数，cluster可以被删除，但前提条件是没有任何快照再使用这个cluster。

针对每一个cluster的2个字节的引用计数，存放在cluster sized blocks。通过refcount\_table\_offset字段可以获取到refcount table的位置，refcount\_table\_clusters字段给出refcount table的大小（单位为cluster），refcount table给出了这些refcount blocks在镜像文件中的偏移地址。

为了获取一个给定的cluster的引用计数，你需要将cluster offset划分成refcount table offset和refcount block offset。一个refcount block是一个单独的cluster，这个cluster里包含了若干个2字节的项，低（cluster\_size -1)位作为block offset，剩余的位作为table offset。

qcow2有一个优化处理，任何一个L1或L2表项指向的cluster的引用计数为1，则L1/L2表项的最高有效位被置上“copied”标记。这表明没有快照在使用这个cluster，所以这个cluster可以马上写入数据，而不需要复制一份给快照使用。

Copy-on-Write特性

一个qcow2镜像可以用来保存其它镜像的变化部分，从而不实际影响到原有磁盘的内容。这就是增量镜像，看着就像一个独立的镜像，其所有数据都是从模板镜像获取的。仅当clusters中的内容跟模板镜像不一样的时候，这些cluster才会被保存到增量镜像中。

写时复制的实现方式比较简单。增量镜像会在qcow2 header中的backing\_file\_offset字段指示一个字符串在qcow2文件内的偏移，该字符串是模板镜像文件的绝对路径，backing\_file\_size字段指明字符串的长度。

当要从增量镜像中读取一个cluster时，qemu会先检查这个cluster在增量镜像中有没有被分配。如果没有，则会去读模板镜像中的对应位置。

快照

快照跟写时复制的概念比较类似。

进一步解释——一个增量镜像也可以被说成是一个“快照”，因为它确实可以作为模板镜像的一个快照。我们可以创建多个增量镜像来实现创建多个“快照”，每一个增量镜像都引用同一个模板镜像。模板镜像必须保持为只读，增量镜像则为可写的。

快照——"实际的快照"——存在于一个镜像里面，这个镜像既当模板，也当增量镜像。每一个快照都是镜像在过去某个瞬间的只读记录。镜像仍然可写，写时复制出来的cluster会被不同的快照引用。

每个快照都对应一个描述信息结构体：

typedef struct QCowSnapshotHeader {

/\* header is 8 byte aligned \*/

uint64\_t l1\_table\_offset;

uint32\_t l1\_size;

uint16\_t id\_str\_size;

uint16\_t name\_size;

uint32\_t date\_sec;

uint32\_t date\_nsec;

uint64\_t vm\_clock\_nsec;

uint32\_t vm\_state\_size;

uint32\_t extra\_data\_size; /\* for extension \*/

/\* extra data follows \*/

/\* id\_str follows \*/

/\* name follows \*/

} QCowSnapshotHeader;

各字段介绍如下：

•快照有名字和ID，都是字符串，id\_str\_size，name\_size给出字符串长度，字符串紧接在QCowSnapshotHeader后面；

•快照至少有原来的L1 table的副本，其通过l1\_table\_offset和l1\_size来定位；

•在快照被创建的时候，qemu会调用gettimeofday()，快照时间被保存在date\_sec和date\_nsec字段中；

•vm\_clock\_nsec给出VM clock当前的状态；

•vm\_state\_size表示作为快照的一部分被保存的虚拟机状态的大小。这个状态被保存在原来L1 table的位置，直接在镜像header的后面；

•extra\_data\_size表示在QCowSnapshotHeader之后的扩展数据的长度，不包括id和name字符串。这一段扩展数据是留给以后用的。

创建一个快照，就会添加一个QCowSnapshotHeader，然后复制一份L1 table，同时会增加所有L2 table和数据clusters的被L1 table引用的引用计数。打完快照之后，如果任何在这个镜像中的L2 table或者data clusters被修改了——也就是，如果一个cluster的引用计数大于1，且"copied"标记被置上了——qemu则会先复制一份这个cluster，然后再写入数据。就这样，所有的快照都不会被修改。

压缩

qcow2镜像格式支持压缩特性，其允许每一个cluster独立的通过zlib进行压缩。

/\*cluster offset表示一个簇在qcow2文件中的偏移，其最高的2位是标记位\*/

从L2 table中获取cluster offset的流程如下：

•如果cluster offset的第二最高有效位是1，则这是一个被压缩的cluster；

•cluster offset中之后的cluster\_bits - 8 位是这个压缩过的cluster的大小，单位是sectors；

•cluster offset剩余的位是压缩的cluster在文件中的实际偏移地址。

加密

qcow2格式，也支持针对cluster的加密。

如果QCowHeader中的crypt\_method字段被置为1，则会采用一个16个字符的密码作为128位AES key。

每一个Cluster中的每一个sector都是通过AES密码块链接模式来单独加密，采用sector的偏移地址（小端模式）来作为128位初始化向量的头64位。

qcow镜像——上一代镜像

qcow2格式相对于qcow格式的不同点有：

1.支持快照的概念，qcow只支持增量镜像；

2.在qcow2中，引入了cluster的引用计数的概念；引用计数也被用来支持快照；

3.在qcow2中，L2 table将一直占一个单独的cluster； 之前，是通过QCowHeader中的l2\_bits来确定的；

4.压缩的cluster的大小，现在单位为sector，之前是字节。