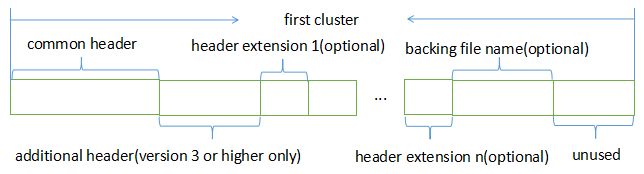
# Qcow2布局

## General

Qcow2 镜像文件由一系列固定大小的簇（cluster，后文以cluster代称）组成，cluster的大小必须介于512B - 2MB之间，默认的cluster大小为64KB。

## Header

位于镜像文件中的第一个cluster中。Header的布局如下：



### Common header

|  |  |  |
| --- | --- | --- |
| Byte range | Field | comments |
| [0, 3] | magic | QCOW magic string ("QFI\xfb"). |
| [4, 7] | version | Version number (valid values are 2 and 3). |
| [8, 15] | Backing\_file\_offset | Offset into the image file at which the backing file name is stored. 0 if the image doesn't have a backing file. |
| [16, 19] | Backing\_file\_size | Length of the backing file name in bytes. Must not be longer than 1023 bytes. Undefined if the image doesn't have a backing file. |
| [20, 23] | Cluster\_bits | Number of bits that are used for addressing an offset within a cluster (1 << cluster\_bits is the cluster size). |
| [24, 31] | size | Virtual disk size in bytes. |
| [32, 35] | Crypt\_method | 0 for no encryption  1 for AES encryption  2 for LUKS encryption |
| [36, 39] | L1\_size | Number of entries in the active L1 table. |
| [40, 47] | L1\_table\_offset | Offset into the image file at which the active L1 table starts. Must be aligned to a cluster boundary. |
| [48, 55] | Refcount\_table\_offset | Offset into the image file at which the refcount table starts. Must be aligned to a cluster boundary. |
| [56, 59] | Refcount\_table\_clusters | Number of clusters that the refcount table occupies. |
| [60, 63] | Nb\_snapthots | Number of snapshots contained in the image. |
| [64, 71] | Snapshot\_offset | Offset into the image file at which the snapshot table starts. Must be aligned to a cluster boundary. |

### Additional header

对于第3版或者更高的版本，header部分还有其它附加数据域。

|  |  |  |
| --- | --- | --- |
| Byte range | Field | comments |
| [72, 79] | Incompatible\_features | Bitmask of incompatible features.  Bit 0: Dirty bit. If this bit is set then refcounts may  be inconsistent, make sure to scan L1/L2  tables to repair refcounts before accessing  the image.  Bit 1: Corrupt bit. If this bit is set then any data  structure may be corrupt and the image must  not be written to (unless for regaining  consistency).  Bits 2-63: Reserved (set to 0). |
| [80, 87] | Compatible\_features | Bitmask of compatible features.  Bit 0: Lazy refcounts bit. If this bit is set then lazy  refcount updates can be used. This means  marking the image file dirty and postponing  refcount metadata updates.  Bits 1-63: Reserved (set to 0). |
| [88, 95] | Autoclear\_features | Bitmask of auto-clear features.  Bit 0: Bitmaps extension bit. This bit indicates  consistency for the bitmaps extension data. It  is an error if this bit is set without the  bitmaps extension present. If the bitmaps  extension is present but this bit is unset, the  bitmaps extension data must be considered  inconsistent.  Bits 1-63: Reserved (set to 0) |
| [96, 99] | Refcount\_order | Describes the width of a reference count block entry (width in bits: refcount\_bits = 1 << refcount\_order). For version 2 images, the order is always assumed to be 4  (i.e. refcount\_bits = 16). This value may not exceed 6 (i.e. refcount\_bits = 64). |
| [100, 103] | Header\_length | Length of the header structure in bytes. |

### Header extension

紧接着header，有一些可选的header扩展项。每一个header扩展项组成结构如下：

|  |  |
| --- | --- |
| Byte range | comments |
| [0, 3] | Extension type.  0x00000000 - End of the header extension area  0xE2792ACA - Backing file format name  0x6803f857 - Feature name table  0x23852875 - Bitmaps extension  0x0537be77 - Full disk encryption header pointer  other - Unknown header extension, can be safely ignored |
| [4, 7] | Length of the header extension data. |
| [8, n] | Header extension data. |
| [n+1, m] | Padding to round up the header extension size to the next multiple of 8. |

**Feature name table**

包含一系列feature name entry，每一个feature name entry组成如下：

|  |  |
| --- | --- |
| Byte range | comments |
| [0] | Type of feature  0: Incompatible feature  1: Compatible feature  2: Autoclear feature |
| [1] | Bit number within the selected feature type’s bitmap (valid values: 0-63). |
| [2, 47] | Feature name. |

**Bitmap extension**

目前只支持dirty tracking bitmap，跟踪从某一时刻以来虚拟磁盘的改变。它和autoclear feature中的bit 0配合工作。Bitmap extension的组织结构如下：

|  |  |  |
| --- | --- | --- |
| Byte range | field | comments |
| [0, 3] | nb\_bitmaps | The number of bitmaps contained in the image. Must be greater than or equal to 1.  Note: Qemu currently only supports up to 65535 bitmaps per image. |
| [4, 7] | reserved | Must be zero. |
| [8, 15] | Bitmap\_directory\_size | Size of the bitmap directory in bytes. It is the cumulative size of all (nb\_bitmaps) bitmap directory entries. |
| [16, 23] | Bitmap\_directory\_offset | Offset into the image file at which the bitmap directory starts. Must be aligned to a cluster boundary. |

**Full disk encryption header pointer**

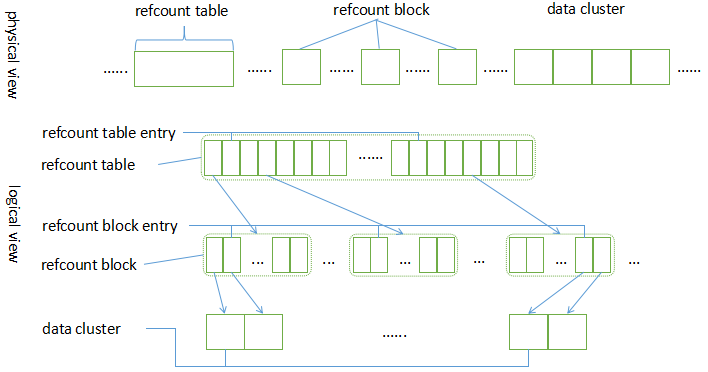
当且仅当header中指定的加密方式(crypt method)需要元数据的情况下该header extension才会存在。目前来说，只有采用LUKS加密方法需要元数据，其它加密方法则不需要。

加密相关，暂不关注。

### Backing file name

如果该镜像文件有backing file，则在header extension之后的区域存放镜像文件名。

## Host cluster management



Qcow2采用引用计数（reference count，refcount）来管理cluster的分配，refcount为0表示cluster处于free状态，refcount为1表示cluster正在被使用，refcount大于等于2表示cluster正在被使用且后续的写必须执行写时复制（copy on write，COW）。

Qcow2采用2级表来管理引用计数，第一级称为refcount table，第二级称为refcount block。

Refcount table存放于镜像文件中一片跨越多个cluster的连续区域（所以镜像文件中只存在唯一一个refcount table），它的大小是可变的，在镜像文件中的位置和大小由header中refcount\_table\_offset和refcount\_table\_clusters确定。Refcount table中包含一系列refcount table entry，每个refcount table entry都指向一个refcount block。

Refcount block在镜像文件中存在多个，每个refcount block在大小上刚好占用一个cluster，它包含一系列refcount block entry，每个refcount block entry中记录相应cluster的引用计数。

### Refcount table entry

每一个Refcount table entry占用8Byte，其组织结构如下：

|  |  |
| --- | --- |
| Bit range | comments |
| [0, 8] | Reserved (set to 0) |
| [9, 63] | The offset into the image file at which the refcount block starts. Must be aligned to a cluster boundary. If this is 0, the corresponding refcount block has not yet been allocated. All refcounts managed by this refcount block are 0. |

### Refcount block entry

每个refcount block entry占用多少空间，是由header中的refcount\_order确定的，1 << refcount\_order就是refcount block entry占用的bit位数目。

|  |  |
| --- | --- |
| Bit range | comments |
| [0, x] | Reference count of the cluster. |

### Refcount retrieve

给定镜像文件偏移为offset，该偏移所在的cluster的引用计数可以通过如下方式获取：

1. 计算每一个refcount block 中包含多少个refcount block entry：

refcount\_block\_entries = (cluster\_size \* 8 / refcount\_bits)

1. 计算offset所在的cluster在refcount block中的第几个refcount block entry：

refcount\_block\_index = (offset / cluster\_size) % refcount\_block\_entries

1. 计算offset所在的cluster在refcount table中的第几个refcount table entry：

refcount\_table\_index = (offset / cluster\_size) / refcount\_block\_entries

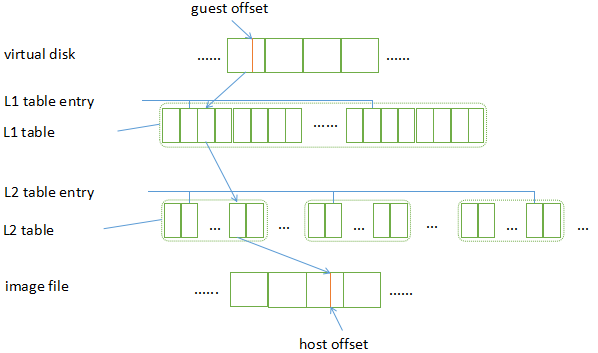
1. 从refcount table中读取相应的refcount block：

refcount\_block = load\_cluster(refcount\_table[refcount\_table\_index]);

1. 从refcount block中读取相应的refcount block entry：

return refcount\_block[refcount\_block\_index];

## Cluster mapping(guest cluster to host cluster)



Qcow2采用2级表结构来管理guest cluster到host cluster的映射，分别是L1 table和L2 table。L1 table存放于镜像文件中一片跨越一个或者多个cluster的连续区域（所以镜像文件中只存在唯一一个L1 table），它的大小是可变的，在镜像文件中的位置和大小由header中l1\_table\_offset和l1\_size（记录L1 table entry数目）确定，它包含一系列L1 table entry。L2 table在镜像文件中则存在多个，每一个L2 table正好是一个cluster的大小，它包含一系列L2 table entry。

### L1 table entry

每一个L1 table entry占用8Byte，其组织结构如下：

|  |  |
| --- | --- |
| Bit range | comments |
| [0, 8] | Reserved (set to 0) |
| [9, 55] | The offset into the image file at which the L2 table starts. Must be aligned to a cluster boundary. If the offset is 0, the L2 table and all clusters described by this L2 table are unallocated. |
| [56, 62] | Reserved (set to 0) |
| [63] | 0 for an L2 table that is unused or requires COW, 1 if its refcount is exactly one. |

*注*：作为一个优化，L1 table entry所指向的cluster，如果其引用计数刚好为1，则L1 table entry的最高位被设置为1(即bit 63被设置为1)，对应到代码中就是QCOW\_OFLAG\_COPIED标识被设置，表明没有任何的snapshot在引用它，可以直接向其写入数据。

### L2 table entry

每一个L2 table entry占用8Byte，其组织结构如下：

|  |  |
| --- | --- |
| Bit range | comments |
| [0, 61] | Cluster descriptor |
| [62] | 0 for standard clusters  1 for compressed clusters |
| [63] | 0 for a cluster that is unused or requires COW, 1 if its refcount is exactly one. |

*注*：作为一个优化，L2 table entry所指向的cluster，如果其引用计数刚好为1，则L2 table entry的最高位被设置为1(即bit 63被设置为1)，对应到代码中就是QCOW\_OFLAG\_COPIED标识被设置，表明没有任何的snapshot在引用它，可以直接向其写入数据。

**Standard cluster descriptor[bit: 0 - 61]**

|  |  |
| --- | --- |
| Bit range | comments |
| [0] | If set to 1, the cluster reads as all zeros. The host cluster offset can be used to describe a preallocation, but it won't be used for reading data from this cluster, nor is data read from the backing file if the cluster is unallocated.  With version 2, this is always 0. |
| [1, 8] | Reserved (set to 0) |
| [9, 55] | Host cluster offset. Must be aligned to a cluster boundary. If the offset is 0, the cluster is unallocated. |
| [56, 61] | Reserved (set to 0) |

**Compressed cluster descriptor[bit: 0 - 61]**

(略)。

### Guest offset to host offset mapping

给定虚拟磁盘偏移为offset，则其在镜像文件中的偏移可以通过以下方式获取：

1. 计算每一个L2 table中包含的L2 table entry的数目：

l2\_entries = (cluster\_size / sizeof(uint64\_t))

1. 计算offset所在的cluster在L2 table中的第几个L2 table entry：

l2\_index = (offset / cluster\_size) % l2\_entries

1. 计算offset所在的cluster在L1 table中的第几个L1 table entry：

l1\_index = (offset / cluster\_size) / l2\_entries

1. 从L1 table中读取L2 table：

l2\_table = load\_cluster(l1\_table[l1\_index]);

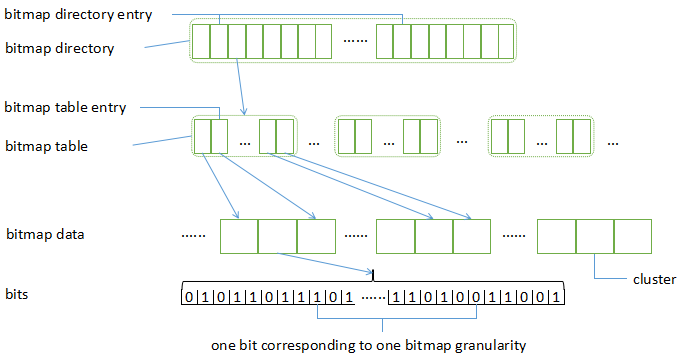
1. 从L2 table中读取相应的L2 table entry：

cluster\_offset = l2\_table[l2\_index];

1. 计算镜像文件中的偏移：

return cluster\_offset + (offset % cluster\_size)

## Bitmaps



Bitmap中的每一个bit都关联到虚拟磁盘中的一块特定大小（bitmap granularity大小，bitmap granularity是bitmap的属性）的区域，对于bit位bit\_nr，其关联的区域为：

[bit\_nr \* bitmap\_granularity .. (bit\_nr + 1) \* bitmap\_granularity - 1]

### Bitmap directory

Bitmap directory存放于镜像文件中一片跨越一个或者多个cluster的连续区域，它在镜像文件中的偏移和大小由bitmap extension中的bitmap\_directory\_offset和bitmap\_directory\_size确定。Bitmap directory中包含一系列bitmap directory entry，每一个bitmap directory entry都用于描述一个特定的bitmap，每一个bitmap directory entry都是变长的，长度取决于bitmap名称和额外数据部分的长度。

### Bitmap directory entry

每一个bitmap directory entry的大小都是可变的（因为名称和额外数据部分的长度是变长的），其组织结构如下：

|  |  |  |
| --- | --- | --- |
| Byte range | field | comments |
| [0, 7] | Bitmap\_table\_offset | Offset into the image file at which the bitmap table for the bitmap starts. Must be aligned to a cluster boundary. |
| [8, 11] | Bitmap\_table\_size | Number of entries in the bitmap table of the bitmap. |
| [12, 15] | flags | Bit 0: in\_use  The bitmap was not saved correctly and may be  inconsistent.  1: auto  The bitmap must reflect all changes of the virtual disk  by any application that would write to this qcow2 file  (including writes, snapshot switching, etc.). The type  of this bitmap must be 'dirty tracking bitmap'.  2: extra\_data\_compatible  This flags is meaningful when the extra data is  unknown to the software (currently any extra data is  unknown to Qemu). If it is set, the bitmap may be  used as expected, extra data must be left as is. If it is  not set, the bitmap must not be used, but both it and  its extra data be left as is.  Bits 3 - 31 are reserved and must be 0. |
| [16] | type | This field describes the sort of the bitmap.  Values:  1: Dirty tracking bitmap  0, 2 - 255 are reserved. |
| [17] | Granularity\_bits | How many bytes of the image accounts for one bit of the bitmap.  granularity = 1 << granularity\_bits.  Note: Qemu currently supports only values 9 - 31. |
| [18, 19] | Name\_size | Size of the bitmap name. Must be non-zero.  Note: Qemu currently doesn't support values greater than 1023. |
| [20, 23] | Extra\_data\_size | Size of type-specific extra data.  For now, as no extra data is defined, extra\_data\_size is reserved and should be zero. If it is non-zero the behavior is defined by extra\_data\_compatible flag. |
| variable | Extra\_data | Extra data for the bitmap, occupying extra\_data\_size bytes. |
| variable | name | The name of the bitmap (not null terminated), occupying name\_size bytes. |
| variable | pad | Padding to round up the bitmap directory entry size to the next multiple of 8. All bytes of the padding must be zero. |

### Bitmap table

每一个bitmap table是变长的，它的起始偏移和大小由bitmap directory entry中的bitmap\_table\_offset和bitmap\_table\_size确定，它一定存放于跨越镜像文件中一个或者多个cluster的连续区域。它由一系列bitmap table entry组成。

### Bitmap table entry

每一个bitmap table entry占用8Byte，它的组织结构如下：

|  |  |
| --- | --- |
| Bit range | comments |
| [0] | Reserved and must be zero if bits 9 - 55 are non-zero.  If bits 9 - 55 are zero:  0: Cluster should be read as all zeros.  1: Cluster should be read as all ones. |
| [1, 8] | Reserved and must be zero. |
| [9, 55] | The host cluster offset. Must be aligned to a cluster boundary. If the offset is 0, the cluster is unallocated; in that case, bit 0 determines how this cluster should be treated during reads. |
| [56, 61] | Reserved and must be zero. |

### Bitmap data

Bitmap data存放于独立的cluster中，由bitmap table entry[bit: 9 - 55]确定其起始位置。在bitmap data中每一个bit位记录一个bitmap granularity相关的信息。给定虚拟磁盘中的一个偏移offset（以bytes为单位），其在镜像文件中的偏移（以bit为单位）可以通过如下方式获取：

1. 计算offset对应的是第几个granularity：

granularity\_index = offset / granularity

1. 计算该granularity在bitmap table中的bitmap table entry：

Bitmap\_table\_entry\_index = (granularity\_index / 8) / cluster\_size

1. 计算该granularity在bitmap data中的偏移：

Bitmap\_data\_offset = (granularity\_index / 8) % cluster\_size

1. 计算该granularity所在的bitmap data的起始地址：

Bitmap\_data\_addr = bitmap\_table[bitmap\_table\_entry\_index]

1. 计算该granularity在镜像文件中的（byte）偏移：

Image\_offset\_bytes = Bitmap\_data\_addr + Bitmap\_data\_offset

1. 计算该granularity在镜像文件中所在的bit偏移：

Image\_offset\_bits = (image\_offset\_bytes) \* 8 + (offset / granularity) % 8

### Dirty tracking bitmap

镜像文件使用过程中，dirty\_tracking\_bitmap可能被“enable”也可能被“disable”。如果处于“enable”状态，则所有的写操作都必须在bitmap中反映，如果某个bit位被设置，则表示相应的区间的数据被写过，否则表示未被写过。

无需在每次写操作之后都实时同步RAM中和镜像文件中的bitmap，这个时候，bitmap directory entry中的in\_use标识应当被设置。

Dirty\_tracking\_bitmap是否被“enable”，可以通过auto标识确定，如果auto标识被设置，则认为dirty\_tracking\_bitmap被“enable”，否则dirty\_tracking\_bitmap被“disable”。