VHD结构详解

【概念】

VHD(Microsoft Virtual Hard Disk)是一种虚拟磁盘的实现方式，即通过文件来模拟物理磁盘的方式来存储数据。如同正常的物理磁盘一样，可以分区，格式化等。

VHD最早由Connectix公司定义，之后Connectix公司被Microsoft 收购。

VHD格式用于Microoft Virtual PC 、Microsoft Windows Server 2008 R2和Microsoft Windows 7，包括hypervisor为基础的虚拟化技术- Hyper-V。

VHD也用于xen虚拟化，目前也是xen上虚拟磁盘的默认标准。

【结构概要】

VHD结构有2种实现方式：固定方式和动态方式。

固定方式就是用真实大小的文件模拟同样大小的一个虚拟磁盘。当然，额外的，一定要加上一些对本文件的总的描述(在文件尾部)，否则系统无法知道就是固定磁盘还是动态磁盘。

动态磁盘是按稀疏方式，用文件做容器，来表述一个可能大得多的虚拟磁盘，如果某个区域没写数据，就可能不在文件中真正分配空间。另外，差异磁盘(快照)也基于动态磁盘的方式存储。本文用"$VHD文件"的说法表示VHD文件本身，而用"$虚拟磁盘"的说法表示 VHD文件描述的虚拟磁盘寻址空间。

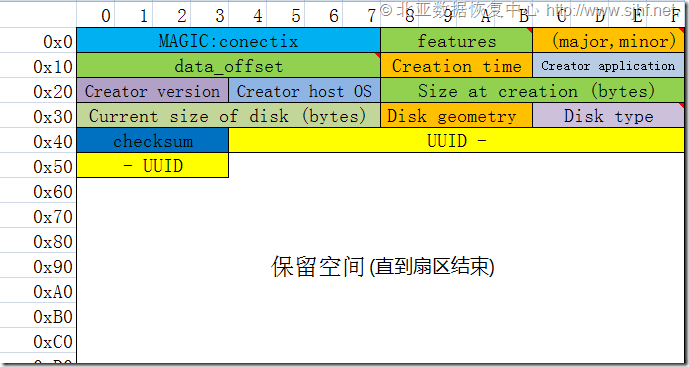
下面数据结构表示，如非特别提示，均为大尾(bigendian)方式。

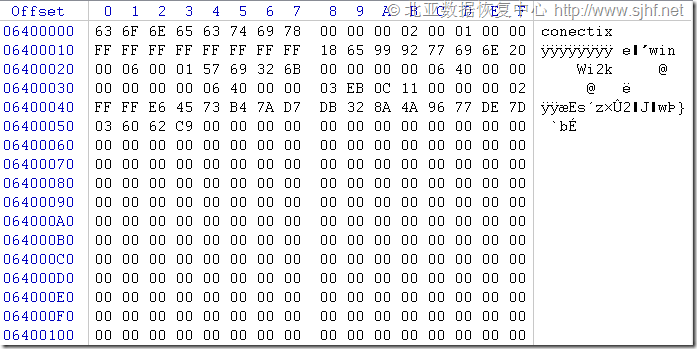
【固定方式的VHD结构】

结构如下图：



文件一开始就是和物理磁盘相同的raw image,即$虚拟磁盘的X扇区就是$VHD文件的X扇区，一对一映射。与物理磁盘不同的是，尾部增加了一个HD footer的结构。HD footer的结构如下图：

 磁盘上的HEX数据表现如下,用WINHEX截取

 xen中的源代码是这样表述的：

struct hd\_ftr {

char cookie[8]; /\* Identifies original creator of the disk \*/

u32 features; /\* Feature Support -- see below \*/

u32 ff\_version; /\* (major,minor) version of disk file \*/

u64 data\_offset; /\* Abs. offset from SOF to next structure \*/

u32 timestamp; /\* Creation time. secs since 1/1/2000GMT \*/

char crtr\_app[4]; /\* Creator application \*/

u32 crtr\_ver; /\* Creator version (major,minor) \*/

u32 crtr\_os; /\* Creator host OS \*/

u64 orig\_size; /\* Size at creation (bytes) \*/

u64 curr\_size; /\* Current size of disk (bytes) \*/

u32 geometry; /\* Disk geometry \*/

u32 type; /\* Disk type \*/

u32 checksum; /\* 1's comp sum of this struct. \*/

vhd\_uuid\_t uuid; /\* Unique disk ID, used for naming parents \*/

char saved; /\* one-bit -- is this disk/VM in a saved state? \*/

char hidden; /\* tapdisk-specific field: is this vdi hidden? \*/

char reserved[426]; /\* padding \*/

};

结构解释：

cookie：

识别标志，为"conectix"，用于判断VHD是否有效

features：

取值如下。

#define HD\_NO\_FEATURES 0x00000000

#define HD\_TEMPORARY 0x00000001 /\* disk can be deleted on shutdown \*/

#define HD\_RESERVED 0x00000002 /\* NOTE: must always be set \*/

ff\_version：

VHD版本，用处不大，用于结构判断，似乎更多的会用到crtr\_ver。

data\_offset：

描述中指下一个结构的起始绝对字节位置，如果是动态磁盘，这表明了dd\_hdr（稍后会提到）的物理字节 位置。如果是固定磁盘，似乎总是0xFFFFFFFF。

timestamp:

VHD的创建时间，指2000年1月1日00:00:00起始的秒值。和HFS对时间的描述方式一致，也就是说此处数值加上0xB492F400 (即2000/01/01 00:00:00)，即是标准的HFS时间方法对本值的解释。

crtr\_app：

见代码注释

crtr\_ver：

创建版本。根据版本号可实施对应的方法。似乎目前只有当创建版本号为0x00000001时，对于bitmap的操作会有不同(具体细节请参考xen源码)。

crtr\_os：

见代码注释

orig\_size ：

创建时$虚拟磁盘大小，再强调一下，这个大小指虚拟出来的磁盘的可用寻址空间。如果是固定格式的VHD，这个大小等于$VHD文件的大小减去1扇区(尾部 hd\_ftr)。

curr\_size：

或许是用于vhd在线扩容后的最后大小表述，没仔细研究过。同样指$虚拟磁盘的大小，即虚拟出来的磁盘的可用寻址空间，如果没有扩容，和orig\_size相同。

geometry：

VHD的C/H/S结构参数，兼容一些老的应用(其实估计不会用到了)。如下表示

#define GEOM\_GET\_CYLS(\_g) (((\_g) >> 16) & 0xffff)

#define GEOM\_GET\_HEADS(\_g) (((\_g) >> 8) & 0xff)

#define GEOM\_GET\_SPT(\_g) ((\_g) & 0xff)

type：

非常重要的，表示这个VHD的类型。如下表示

#define HD\_TYPE\_NONE 0

#define HD\_TYPE\_FIXED 2 /\* fixed-allocation disk \*/

#define HD\_TYPE\_DYNAMIC 3 /\* dynamic disk \*/

#define HD\_TYPE\_DIFF 4 /\* differencing disk \*/

如果是差异磁盘，在dd\_hdr中会描述父设备的设备号，但结构与动态磁盘相同

checksum：

整个扇区所有字节(当然一开始不包括checksum本身)相加得到32位数，再按位取反。

uuid：

用于VHD识别号，如果是有差异磁盘，这个ID非常重要，决定了VHD间的主从关系。

saved：

动态中使用，见代码注释

hidden：

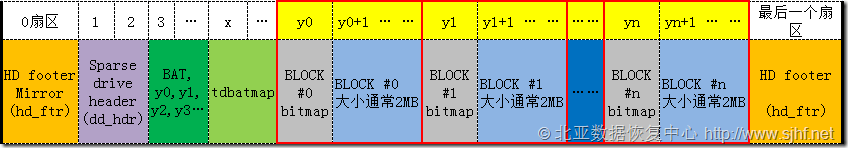
见代码注释

reserved：

保留空间，总是为0

上述结构中最重要的变量为VHD类型、大小、checksum。

【动态方式的VHD结构】

 动态VHD和固定VHD相同的是，尾部也是重要的hd\_ftr格式，不同的是hd\_ftr会表明本VHD是动态方式的。

上图中，位置描述部分黄色区域为数据区，其余区域，均为元数据区(结构管理区)。

0扇区的hd\_ftr mirror是对尾部hd\_ftr的备份。hd\_ftr在固定格式的VHD中已经详细解释，hd\_ftr中data\_offset会描述dd\_hdr的位置，不过，这个位置目前总是1扇区。

dd\_hdr结构用于表述整个动态vhd的概况，分配块大小等的变量。

BAT指Block allocation table,非常重要的，表示$虚拟磁盘地址到$VHD文件地址的块映射表。

tdbatmap指BAT的分配位图，其实就是所有分配块的是否用满的位描述表。在某些版本的vhd中，可能不存在此结构，xen源码中是这样判断是否存在batmap的：

int vhd\_has\_batmap(vhd\_context\_t \*ctx)

{

if (!vhd\_type\_dynamic(ctx))

return 0;

if (!vhd\_creator\_tapdisk(ctx))

return 0;

if (ctx->footer.crtr\_ver <= VHD\_VERSION(0, 1))

return 0;

if (ctx->footer.crtr\_ver >= VHD\_VERSION(1, 2))

return 1;

/\*

\* VHDs of version 1.1 probably have a batmap, but may not

\* if they were updated from version 0.1 via vhd-update.

\*/

if (!vhd\_validate\_batmap\_header(&ctx->batmap))

return 1;

if (vhd\_read\_batmap\_header(ctx, &ctx->batmap))

return 0;

return (!vhd\_validate\_batmap\_header(&ctx->batmap));

}

每个数据块都由块bitmap和块本身组成，块bitmap用于描述块中每个扇区是否占用的位图表，块本身就是数据区。

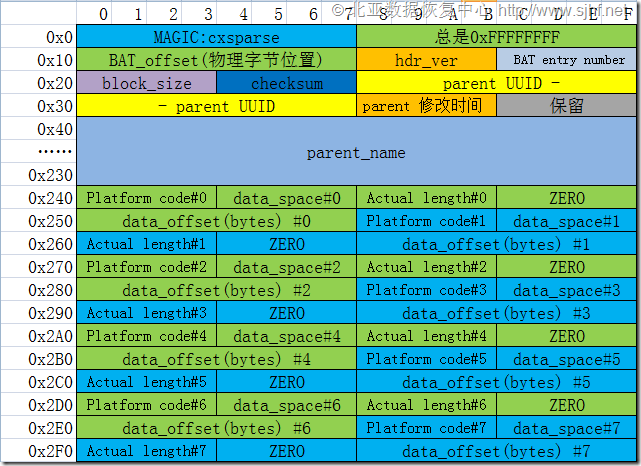
我们依次详细分析上述所有结构：

1、hd\_ftr：

固定格式vhd中已详细分析，在动态磁盘中，不同的是data\_offset总为0x200,类型为3或4。

2、dd\_hdr：

结构如下图：



WINHEX中的磁盘表现如下：

 xen中的源代码是这样表述的：

struct dd\_hdr {

char cookie[8]; /\* Should contain "cxsparse" \*/

u64 data\_offset; /\* Byte offset of next record. (Unused) 0xffs \*/

u64 table\_offset; /\* Absolute offset to the BAT. \*/

u32 hdr\_ver; /\* Version of the dd\_hdr (major,minor) \*/

u32 max\_bat\_size; /\* Maximum number of entries in the BAT \*/

u32 block\_size; /\* Block size in bytes. Must be power of 2. \*/

u32 checksum; /\* Header checksum. 1's comp of all fields. \*/

vhd\_uuid\_t prt\_uuid; /\* ID of the parent disk. \*/

u32 prt\_ts; /\* Modification time of the parent disk \*/

u32 res1; /\* Reserved. \*/

char prt\_name[512]; /\* Parent unicode name. \*/

struct prt\_loc loc[8]; /\* Parent locator entries. \*/

char res2[256]; /\* Reserved. \*/

};

结构解释：

cookie：

识别标识，为"cxsparse"，用于是否dd\_hdr的校验。

data\_offset：

未使用，总设置为0xFFFFFFFF。

table\_offset：

很重要,BAT结构在$VHD文件中的绝对字节位置。几乎总是0x600

hdr\_ver：

dd\_hdr的版本

max\_bat\_size：

BAT条目的最大数量，实际上每个bat条目，就相当于一个块。

block\_size：

块大小，几乎总是2MB。

checksum：

同hd\_ftr中 checksum的计算方式相同。计算范围为从dd\_hdr开始的1024字节。

ptr\_uuid：

差异磁盘中非常重要，表示其父vhd的uuid。这样才可以实现快照穿透。

prt\_ts：

父磁盘的修改时间，时间表示方法参考hd\_ftr中timestamp的表示方式。

res1：

保留

ptr\_name：

父磁盘的unicode名称。可以更快地找到父磁盘，但找到后，还需通过uuid校验。

loc：

用来记录在不同平台上的父磁盘的名称，并不很重要。可见dd\_hdr结构图解释，本结构的每个条目会指向一个存储文件名称的$vhd文件的绝对字节位置。

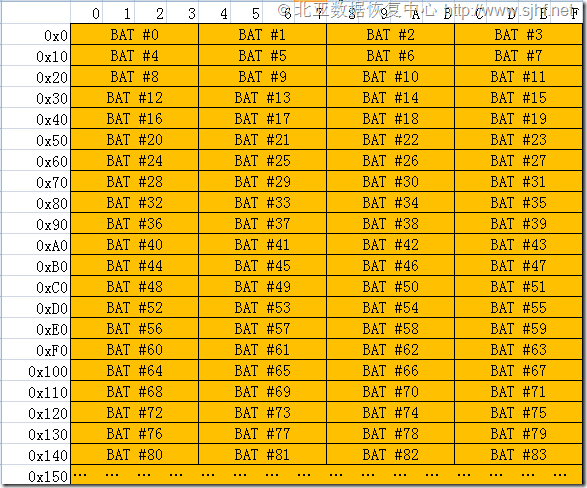
res1：

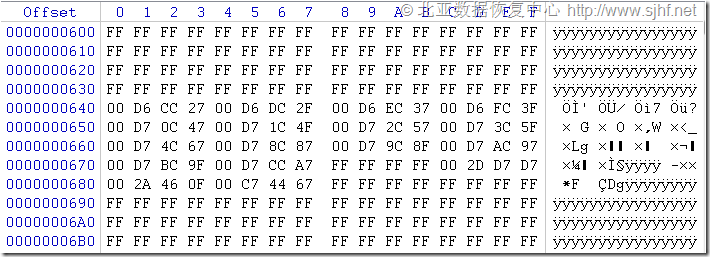
保留。

上述结构最重要的变量为：块大小，BAT位置，BAT数量，父磁盘的uuid（对于差异磁盘）

3、BAT：

结构如下图：

 WINHEX中的磁盘表现如下：



结构解释：

BAT表中，每4个字节表示一个bat entry，从BAT的0字节开始，以4字节为单位，第x个条目(条目从0开始编号)，表示$虚拟磁盘中第x块在$vhd中的扇区位置。如果第x个条目的值为0xFFFFFFFF,表示$虚拟磁盘的第x块为稀疏，返回一整块0。

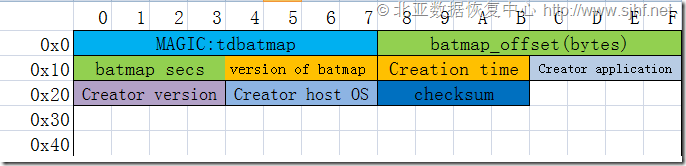
假定块大小为2MB，对应上面winhex的磁盘表现图，从0x600位置起(按左上角offset定位)，前4行全部为0xFFFFFFFF，表明整个$虚拟磁盘的前16个2MB块是全0，并未在$vhd文件中分配空间。位于0x640处的第16个bat entry(从0开始编号的序号)的值为0x00D6CC27，表示$虚拟磁盘的第16块(块大小为2M)的数据流存储于$vhd文件的第0x00D6CC27扇区。

4、tdbatmap

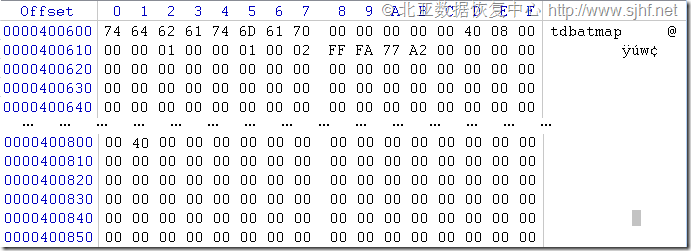
bat entry的bitmap，也可理解为块的分配位图。每一位表示一个block，如果位为1，表示block已经用满。如果为0，表示未使用，或未用满。

tdbatmap由头结构和batmap内容部分组成。

头结构如下图：



在WINHEX中的磁盘表现(连同其后的bitmap区域)如下：



xen中的源代码是这样表述头结构的：

struct dd\_batmap\_hdr {

char cookie[8]; /\* should contain "tdbatmap" \*/

u64 batmap\_offset; /\* byte offset to batmap \*/

u32 batmap\_size; /\* batmap size in sectors \*/

u32 batmap\_version; /\* version of batmap \*/

u32 checksum; /\* batmap checksum -- 1's complement of batmap \*/

};

头结构说明：

cookie：

识别标识，为" tdbatmap "，用于是否dd\_batmap\_hdr的校验。

batmap\_offset：

batmap的起始物理位置，如上面winhex磁盘图中表示的起始位置为0x400800。

batmap\_size：

batmap的大小，以扇区为单位 。其实等于bat entry数量除以8，再对齐扇区大小的扇区数。

batmap\_version：

batmap结构的版本号。

checksum：

参考hd\_ftr中checksum的计算方法。计算范围为包括batmap头和内容区的整个batmap区域

batmap内容区结构说明：

如上面winhex磁盘图中偏移为0x400800起始的数据，每一位表示一个块的分配情况，开始的0x004000....表示整个$虚拟磁盘的第9块(块从0开始编号)是用满了的。而第0、1、2、3等块是未用满或未使用的块。

5、数据块区：

如果块大小为2M，其实每个bat entry所指向的空间大小为512byte+2MB。最前面的512字节是本块内的扇区位图，如果位为1，表示代指的扇区已使用，如果为0，表示代指的扇区为全0，更多的意义用于差异磁盘。举个例子，如果bitmap的开始2个字节为0x80和0x12,表示第0、11、14扇区已分配。

紧跟在块内512字节bitmap后的就是块数据本身，可按bitmap的表述，直接映射到$虚拟磁盘中。

VHD动态磁盘格式总结：

对一个动态vhd磁盘的寻址过程大致为：

1、通过读取hd\_ftr结构，确定是否动态磁盘，以及dd\_hdr的位置。

2、读取dd\_hdr，确定块大小，bat的位置，bat的数量，以及是否差异磁盘(差异磁盘的处理方式后面讲到)

3、定位bat区域，可随机确定任何一个 block的位置，如果bat的值为0xFFFFFFFF,则返回全0。

4、确定block位置后，先读取1扇区的bitmap区域，本block的某个扇区是否已使用，可以通过此bitmap确定。

【差异磁盘的读取方式】

差异磁盘是建立在一个固定或动态磁盘上的快照。其本意是差异磁盘中仅存储自创建备份点以来的所有改动。本质上，差异磁盘就是个动态磁盘。

如何在数据层面合并差异磁盘和其父磁盘呢？

如果要读取某个数据块x，系统会首先读取差异磁盘x块的bat entry。如果其值为 0xFFFFFFFF，表明差异磁盘中未记录数据，接下来就应该读取本差异磁盘的父磁盘的第x块;如果bat entry其值不为0xFFFFFFFF，则可以通过batmap中x块是否用满(或分析块数据区中的bitmap)，来确定是否需要穿透进父磁盘进行补差，如果已用满，则不需再处理父磁盘；如果未用满，再看本块数据区中bitmap哪些位为0，为0的穿透进父磁盘进行补差，为1则直接读取。

【参考资料】

1、http://en.wikipedia.org/wiki/VHD\_(file\_format)

2、http://xen.org/