GPT分区数据格式分析（图已补上）

1. 背景与前言

随着技术的不断提高，电子产品的集成度变得越来越高，硬盘是这几年中的一个突出产品，近年来，硬盘容量不断提升，从500G到1TB，目前已经能以很便宜的价格买到3TB的硬盘。
分区就是把一块大的物理硬盘分成一个一个的逻辑盘，这样便于文档归类，减少坏道损失。
传统的分区格式我们称其为MBR分区，传统的MBR分区格式有一个2TB的限制：当个分区大小不能超过2TB。
以前在企业和服务器领域，一个分区达到2T的情况很正常，所以GPT分区其实很早就已经出现了，只是个人用户用不到而已。
现在，很多人自己就能话很少的钱达到组建几TB的磁盘阵列，所以现在个人用户的操作系统也开始使用GPT分区了。
网络上介绍MBR分区的文章很多，本文不做赘述。但对于GPT分区，至少我前段时间查资料发现深入讲解的文章还不算多。
本文就GPT分区格式进行简单的讲解。

2. 术语及缩写

术语/缩写	解释
GPT	GUID Partition Table
MBR	Master Boot Record
LBA	Logic Block Address

3. GPT分区数据格式

3.1 LBA0

LBA0就是存储设备的第0个逻辑存储块。
逻辑存储块，是与物理存储块进行区分的，因为目前的工艺水平导致不论NAND还是机械硬盘都存在坏块的情况，在使用存储设备时，遇到损坏的存储块就会被驱动程序或固件自动跳过，因此坏块对于驱动程序以上的应用程序来说是透明的，他们感受不到坏块，也不关注坏块，他们对存储设备的存储块进行的编号称为逻辑块地址（也可以叫：逻辑块编号）。
GPT分区为了兼容传统的MBR分区，其第一个逻辑块数据格式与MBR分区一致，即：第一个逻辑块就是MBR（主引导记录）。
但为了与传统的MBR分区进行区分，GPT分区的分区类型为EE，在传统的MBR中，EE类型的分区表示保护类型，GPT以此来防止其数据被无意间篡改。
GPT分区的数据格式如下图所示：

图 1 GPT分区数据格式

在GPT分区中，每一个数据读写单元成为LBA（逻辑块地址），一个“逻辑块”相当于传统MBR分区中的一个“扇区”，之所以会有区别，是因为GPT除了要支持传统硬盘，还需要支持以NAND FLASH为材料的SSD硬盘，这些硬盘的一个读写单元是2KB或4KB，所以GPT分区中干脆用LBA来表示一个基础读写块，当GPT分区用在传统硬盘上时，通常，LBA就等于扇区号，有些物理硬盘支持2KB对齐，此时LBA所表示的一个逻辑块就是2KB的空间。
为了方便，我们后面仍然将逻辑块称为扇区。
在图1中可以看出：
第0扇区：和传统MBR分区一样，仍然为主引导记录

第1扇区：我们称之为“主分区头”
第2~33扇区：共计32个扇区，我们称之为“主分区节点”
最后一个扇区（-1扇区）：我们称之为“备份分区头”，它就是“主分区头”的一个Copy
从-2~-33扇区：共计32个扇区，我们称之为“备份分区节点”，它就是“主分区节点”的一个Copy
第34~-34扇区：正常的GPT分区内容，文件系统（如：FAT，NTFS，EXT等）就是构建在这里面。
下面看一个GPT分区的第0个扇区，即MBR实例：

图 2 GPT分区的第0个逻辑块-MBR

图2中只截取了第一个扇区（地址范围：0x0000~0x01FF）的最末部分，图中，反色显示的部分在MBR的数据格式中是用于定义4个主分区的，在图2中，可以明确的看出来：只定义了一个主分区，且其类型为0xEE：
用WinHex工具分析此MBR结果如下：

图 3 WinHex分析GPT的第0个扇区

图3中，WinHex分析结果也显示第一个分区的类型是EE。
bootloader(或BIOS)根据这个EE就能知道这是一个GPT分区表。

3.2 LBA1

参考图1，GPT分区的LBA1中存放的内容我们称为“主分区头（Primary GPT Header）”，主分区头的数据格式如下：

表格 1 GPT主分区头数据格式

字节偏移	长度	内容
0	8字节	签名（"EFI PART", 45 46 49 20 50 41 52 54）
8	4字节	修订版本号（在1.0版中，值是 00 00 01 00）
12	4字节	分区头的大小（单位是字节，通常是92字节，即 5C 00 00 00）
16	4字节	分区头（第0－91字节）的CRC32校验，计算前需先将此内容写0
20	4字节	保留，必须是 0
24	8字节	当前LBA（这个分区表头的位置）
32	8字节	备份LBA（另一个分区表头的位置）
40	8字节	第一个可用于分区的LBA（主分区表的最后一个LBA + 1）
48	8字节	最后一个可用于分区的LBA（备份分区表的第一个LBA – 1）
56	16字节	硬盘GUID（在类UNIX系统中也叫UUID）
72	8字节	分区表项的起始LBA（在主分区表中是2）
80	4字节	分区表项的数量
84	4字节	一个分区表项的大小（通常是128）
88	4字节	CRC32 of partition array
92	*	保留，剩余的字节必须是0（对于512B/LBA的硬盘就是420个字节）

下面是一个主分区头的实例：

图 4 GPT分区头实例

在图4的实例中，可以看出，与上面的表1中定义的数据格式是一一对应的。

从图4可以得知，此GPT分区的分区表项是从LBA2开始的：

一共有28个分区：

每个分区表项的大小是128字节：

3.3 LBA2~LBA33

LBA2到LBA33，一共32个逻辑块，是用于存储分区表项的，每一个分区表项就描述了一个分区，分区表项的数据格式如下：

表格 2 GPT分区表项数据格式

起始字节	长度	内容
0	16字节	分区类型GUID
16	16字节	分区GUID
32	8字节	起始LBA（小端序）
40	8字节	末尾LBA
48	8字节	属性标签（如：bit60表示“只读”）
56	72字节	分区名（可以包括36个UTF-16（小端序）字符）

可以看出，GPT分区的分区类型，分区标识，都是用GUID进行区分的（GUID在Linux上通常叫UUID），如果你不了解GUID，请自行请教维基百科。

下面是我截取的两个分区表项的实例：

图 5 GPT分区表项实例

将图5对照表2，即可知道这两个分区的具体信息：

分区1：

- 分区范围：LBA[0x20000] ~ LBA[0x4abdf]，共计87536KB

- 分区名：modem

分区2：

- 分区范围：LBA[0x60000] ~ LBA[0x600FF]，共计128KB

- 分区名：sbl1

分区类型的GUID通常约定如下：

表格 3 分区类型的GUID约定

操作系统	分区类型	GUID[1]
(None)	未使用	00000000-0000-0000-0000-000000000000
MBR分区表	024DEE41-33E7-11D3-9D69-0008C781F39F	
EFI系统分区	C12A7328-F81F-11D2-BA4B-00A0C93EC93B	
BIOS引导分区	21686148-6449-6E6F-744E-656564454649	
Windows	微软保留分区	E3C9E316-0B5C-4DB8-817D-F92DF00215AE
基本数据分区[2]	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7	
逻辑软盘管理工具元数据分区	5808C8AA-7E8F-42E0-85D2-E1E90434CFB3	
逻辑软盘管理工具数据分区	AF9B60A0-1431-4F62-BC68-3311714A69AD	
Windows 恢复环境	DE94BBA4-06D1-4D40-A16A-	

Windows恢复环境	BFD50179D6AC	
IBM GPFS分区	37AFFC90-EF7D-4e96-91C3-2D7AE055B174	
HP-UX	数据分区	75894C1E-3AEB-11D3-B7C1-7B03A0000000
服务分区	E2A1E728-32E3-11D6-A682-7B03A0000000	
Linux	数据分区[2]	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
RAID分区	A19D880F-05FC-4D3B-A006-743F0F84911E	
交换分区	0657FD6D-A4AB-43C4-84E5-0933C84B4F4F	
逻辑卷管理器(LVM)分区	E6D6D379-F507-44C2-A23C-238F2A3DF928	
保留	8DA63339-0007-60C0-C436-083AC8230908	
FreeBSD	启动分区	83BD6B9D-7F41-11DC-BE0B-001560B84F0F
数据分区	516E7CB4-6ECF-11D6-8FF8-00022D09712B	
交换分区	516E7CB5-6ECF-11D6-8FF8-00022D09712B	
UFS分区	516E7CB6-6ECF-11D6-8FF8-00022D09712B	
en:Vinum-volume-manager分区	516E7CB8-6ECF-11D6-8FF8-00022D09712B	
ZFS分区	516E7CBA-6ECF-11D6-8FF8-00022D09712B	
Mac OS X	HFS(HFS+)分区	48465300-0000-11AA-AA11-00306543ECAC
苹果公司UFS	55465300-0000-11AA-AA11-00306543ECAC	
ZFS[3]	6A898CC3-1DD2-11B2-99A6-080020736631	
苹果RAID分区	52414944-0000-11AA-AA11-00306543ECAC	
苹果RAID分区，下线	52414944-5F4F-11AA-AA11-00306543ECAC	
苹果启动分区	426F6F74-0000-11AA-AA11-00306543ECAC	
Apple Label	4C616265-6C00-11AA-AA11-00306543ECAC	
Apple TV 恢复分区	5265636F-7665-11AA-AA11-00306543ECAC	
Solaris	启动分区	6A82CB45-1DD2-11B2-99A6-080020736631
根分区	6A85CF4D-1DD2-11B2-99A6-080020736631	
交换分区	6A87C46F-1DD2-11B2-99A6-080020736631	
备份分区	6A8B642B-1DD2-11B2-99A6-080020736631	
/usr分区[3]	6A898CC3-1DD2-11B2-99A6-080020736631	
/var 分区	6A8EF2E9-1DD2-11B2-99A6-080020736631	
/home 分区	6A90BA39-1DD2-11B2-99A6-080020736631	

备用扇区	6A9283A5-1DD2-11B2-99A6-080020736631	
保留分区	6A945A3B-1DD2-11B2-99A6-080020736631	
6A9630D1-1DD2-11B2-99A6-080020736631		
6A980767-1DD2-11B2-99A6-080020736631		
6A96237F-1DD2-11B2-99A6-080020736631		
6A8D2AC7-1DD2-11B2-99A6-080020736631		
NetBSD[4]	交换分区	49F48D32-B10E-11DC-B99B-0019D1879648
FFS分区	49F48D5A-B10E-11DC-B99B-0019D1879648	
LFS分区	49F48D82-B10E-11DC-B99B-0019D1879648	
RAID分区	49F48DAA-B10E-11DC-B99B-0019D1879648	
concatenated分区	2DB519C4-B10F-11DC-B99B-0019D1879648	
加密分区	2DB519EC-B10F-11DC-B99B-0019D1879648	

[1] 本表中的GUID使用小端序表示。例如，EFI系统分区的GUID在这里写成C12A7328-F81F-11D2-BA4B-00A0C93EC93B但实际上它对应的16字节的序列是 28 732A C1 1F F8 D2 11 BA 4B 00 A0 C9 3E C9 3B ——只有前3部分的字节序被交换了。

[2] Linux和Windows的数据分区使用相同的GUID。

[3] Solaris系统中/usr分区的GUID在Mac OS X上被用作普通的ZFS分区。

[4] 具体定义见src/sys/sys/disklabel_gpt.h。NetBSD的GUID在单独定义之前曾经使用过FreeBSD的GUID。

实际上，GUID的约定主要是为了BIOS方便识别分区类型，在嵌入式系统中，这主要由bootloader来约定。

GPT分区表项中，有一个分区属性的区域，此区域主要是指明此分区的属性，如只读，隐藏，启动等，通常的约定如下：

表格 4 GPT分区属性标签定义

Bit	Content
0	System partition (disk partitioning utilities must preserve the partition as is)
2	Legacy BIOS bootable (equivalent to active flag (typically bit 7 set) at offset +0h in partition entries of the MBR partition table)[6]
60	Read-only
62	Hidden
63	Do not automount (i.e., do not assign drive letter)

分区属性标签是位结构的，即：一位表示一个开关

3.4 备份分区头与备份分区表项

备份分区头实际上与主分区头的内容完全一样，只是它存储在最后一个逻辑块（LBA-1）
备份分区表项的内容与分区表项的内容也是完全一致的，只是它存放的位置是LBA-33到LBA-2的区域。
备份分区头与备份分区表项存在的主要意义就是数据恢复。