EFI、UEFI主板BIOS 和 MBR、GPT硬盘分区技术详解

现在EFI主板越来越多，而随着主流硬盘逐渐超过2T，GPT的出现频率也变高了，本帖转载了几个相关名字的介绍以及对比。

**一、EFI** (可扩展固件接口,英文名Extensible Firmware Interface 或EFI)

由英特尔，一个主导个人电脑技术研发的公司推出的一种在未来的类PC的电脑系统中替代BIOS的升级方案。BIOS技术的兴起源于IBM PC/AT机器的流行以及第一台由康柏公司研制生产的“克隆”PC。在PC启动的过程中，**BIOS担负着初始化硬件，检测硬件功能，以及引导操作系统的责任**，在早期，BIOS还提供一套运行时的服务程序给操作系统及应用程序使用。BIOS程序存放于一个掉电后内容不会丢失的只读存储器中，系统加电时处理器的第一条指令的地址会被定位到BIOS的存储器中，便于使初始化程序得到执行。   
　　   
  
**EFI的产生**   
　　　众所周知，英特尔在近二十年来引领以x86系列处理器为基础的PC技术潮流，它的产品如CPU，芯片组等在PC生产线中占据绝对领导的位置。因此，不少人认为这一举动显示了英特尔公司欲染指固件产品市场的野心。事实上，EFI技术源于英特尔安腾处理器(Itanium)平台的推出。安腾处理器是英特尔瞄准服务器高端市场投入近十年研发力量设计产生的与x86系列完全不同的64位新架构。在x86系列处理器进入32位的时代，由于兼容性的原因，新的处理器 (i80386)保留了16位的运行方式(实模式)，此后多次处理器的升级换代都保留了这种运行方式。甚至在含64位扩展技术的至强系列处理器中，处理器加电启动时仍然会切换到16位的实模式下运行。英特尔将这种情况归咎于BIOS技术的发展缓慢。自从PC兼容机厂商通过净室的方式复制出第一套BIOS源程序，BIOS就以16位汇编代码，寄存器参数调用方式，静态链接，以及1MB以下内存固定编址的形式存在了十几年。虽然由于各大BIOS厂商近年来的努力，有许多新元素添加到产品中，如PnP BIOS，ACPI，传统USB设备支持等等，但BIOS的根本性质没有得到任何改变。这迫使英特尔在开发更新的处理器时，都必须考虑加进使效能大大降低的兼容模式。有人曾打了一个比喻：这就像保时捷新一代的全自动档跑车被人生套上去一个蹩脚的挂档器。   
  
　　然而，安腾处理器并没有这样的顾虑，它是一个新生的处理器架构，系统固件和操作系统之间的接口都可以完全重新定义。并且这一次，英特尔将其定义为一个可扩展的，标准化的固件接口规范，不同于传统BIOS的固定的，缺乏文档的，完全基于经验和晦涩约定的一个事实标准。基于EFI的第一套系统产品的出现至今已经有五年的时间，如今，英特尔试图将成功运用在高端服务器上的技术推广到市场占有率更有优势的PC产品线中，并承诺在2006年间会投入全力的技术支持。   
　　   
  
**比较EFI和BIOS**   
　　　一个显著的区别就是EFI是用模块化，C语言风格的参数堆栈传递方式，动态链接的形式构建的系统，较BIOS而言更易于实现，容错和纠错特性更强，缩短了系统研发的时间。它运行于32位或64位模式，乃至未来增强的处理器模式下，突破传统16位代码的寻址能力，达到处理器的最大寻址。它利用加载EFI 驱动的形式，识别及操作硬件，不同于BIOS利用挂载实模式中断的方式增加硬件功能。后者必须将一段类似于驱动的16位代码，放置在固定的 0x000C0000至0x000DFFFF之间存储区中，运行这段代码的初始化部分，它将挂载实模式下约定的中断向量向其他程序提供服务。例如，VGA 图形及文本输出中断(INT 10h)，磁盘存取中断服务(INT 13h)等等。由于这段存储空间有限(128KB)，BIOS对于所需放置的驱动代码大小超过空间大小的情况无能为力。另外，BIOS的硬件服务程序都已 16位代码的形式存在，这就给运行于增强模式的操作系统访问其服务造成了困难。因此BIOS提供的服务在现实中只能提供给操作系统引导程序或MS-DOS 类操作系统使用。而EFI系统下的驱动并不是由可以直接运行在CPU上的代码组成的，而是用EFI Byte Code编写而成的。这是一组专用于EFI驱动的虚拟机器指令，必须在EFI驱动运行环境(Driver Execution Environment，或DXE)下被解释运行。这就保证了充分的向下兼容性，打个比方说，一个带有EFI驱动的扩展设备，既可以将其安装在安腾处理器的系统中，也可以安装于支持EFI的新PC系统中，而它的EFI驱动不需要重新编写。这样就无需对系统升级带来的兼容性因素作任何考虑。另外，由于EFI 驱动开发简单，所有的PC部件提供商都可以参与，情形非常类似于现代操作系统的开发模式，这个开发模式曾使Windows在短短的两三年时间内成为功能强大，性能优越的操作系统。基于EFI的驱动模型可以使EFI系统接触到所有的硬件功能，在操作操作系统运行以前浏览万维网站不再是天方夜谭，甚至实现起来也非常简单。这对基于传统BIOS的系统来说是件不可能的任务，在BIOS中添加几个简单的USB设备支持都曾使很多BIOS设计师痛苦万分，更何况除了添加对无数网络硬件的支持外，还得凭空构建一个16位模式下的TCP/IP协议栈。   
  
　　一些人认为BIOS只不过是由于兼容性问题遗留下来的无足轻重的部分，不值得为它花费太大的升级努力。而反对者认为，当BIOS的出现制约了PC技术的发展时，必须有人对它作必要的改变。   
　　   
**EFI和操作系统**   
　　　EFI在概念上非常类似于一个低阶的操作系统，并且具有操控所有硬件资源的能力。不少人感觉它的不断发展将有可能代替现代的操作系统。事实上，EFI 的缔造者们在第一版规范出台时就将EFI的能力限制于不足以威胁操作系统的统治地位。首先，它只是硬件和预启动软件间的接口规范；其次，EFI环境下不提供中断的访问机制，也就是说每个EFI驱动程序必须用轮询的方式来检查硬件状态，并且需要以解释的方式运行，较操作系统下的驱动效率更低；再则，EFI系统不提供复杂的存储器保护功能，它只具备简单的存储器管理机制，具体来说就是指运行在x86处理器的段保护模式下，以最大寻址能力为限把存储器分为一个平坦的段，所有的程序都有权限存取任何一段位置，并不提供真实的保护服务。当EFI所有组件加载完毕时，系统可以开启一个类似于操作系统Shell的命令解释环境，在这里，用户可以调入执行任何EFI应用程序，这些程序可以是硬件检测及除错软件，引导管理，设置软件，操作系统引导软件等等。理论上来说，对于 EFI应用程序的功能并没有任何限制，任何人都可以编写这类软件，并且效果较以前MS-DOS下的软件更华丽，功能更强大。一旦引导软件将控制权交给操作系统，所有用于引导的服务代码将全部停止工作，部分运行时代服务程序还可以继续工作，以便于操作系统一时无法找到特定设备的驱动程序时，该设备还可以继续被使用。

在实现中，EFI初始化模块和驱动执行环境通常被集成在一个只读存储器中。Pre-EFI初始化程序在系统开机的时候最先得到执行，它负责最初的 CPU，主桥及存储器的初始化工作，紧接着载入EFI驱动执行环境(DXE)。当DXE被载入运行时，系统便具有了枚举并加载其他EFI驱动的能力。在基于PCI架构的系统中，各PCI桥及PCI适配器的EFI驱动会被相继加载及初始化；这时，系统进而枚举并加载各桥接器及适配器后面的各种总线及设备驱动程序，周而复始，直到最后一个设备的驱动程序被成功加载。正因如此，EFI驱动程序可以放置于系统的任何位置，只要能保证它可以按顺序被正确枚举。例如一个具PCI总线接口的ATAPI大容量存储适配器，其EFI驱动程序一般会放置在这个设备的符合PCI规范的扩展只读存储器(PCI Expansion ROM)中，当PCI总线驱动被加载完毕，并开始枚举其子设备时，这个存储适配器旋即被正确识别并加载它的驱动程序。部分EFI驱动程序还可以放置在某个磁盘的EFI专用分区中，只要这些驱动不是用于加载这个磁盘的驱动的必要部件。在EFI规范中，一种突破传统MBR磁盘分区结构限制的GUID磁盘分区系统(GPT)被引入，新结构中，磁盘的分区数不再受限制(在MBR结构下，只能存在4个主分区)，并且分区类型将由GUID来表示。在众多的分区类型中，EFI系统分区可以被EFI系统存取，用于存放部分驱动和应用程序。很多人担心这将会导致新的安全性因素，因为EFI系统比传统的BIOS更易于受到计算机病毒的攻击，当一部分EFI驱动程序被破坏时，系统有可能面临无法引导的情况。实际上，系统引导所依赖的EFI驱动部分通常都不会存放在EFI的 GUID分区中，即使分区中的驱动程序遭到破坏，也可以用简单的方法得到恢复，这与操作系统下的驱动程序的存储习惯是一致的。CSM是在x86平台EFI 系统中的一个特殊的模块，它将为不具备EFI引导能力的操作系统提供类似于传统BIOS的系统服务。   
　　   
**EFI的发展**   
　　　英特尔无疑是推广EFI的积极因素，近年来由于业界对其认识的不断深入，更多的厂商正投入这方面的研究。包括英特尔，AMD在内的一些PC生产厂家联合成立了联合可扩展固件接口论坛，它将在近期推出第一版规范。这个组织将接手规划EFI发展的重任，并将英特尔的EFI框架解释为这个规范的一个具体实现。另外，各大BIOS提供商如Phoenix, AMI等，原先被认为是EFI发展的阻碍力量，现在也不断的推出各自的解决方案。分析人士指出，这是由于BIOS厂商在EFI架构中重新找到了诸如 Pre-EFI启动环境之类的市场位置，然而，随着EFI在PC系统上的成功运用，以及英特尔新一代芯片组的推出，这一部分市场份额将会不出意料的在英特尔的掌控之中。

**二、UEFI**

UEFI中图形化的硬件设置界面Extensible Firmware Interface（EFI，可扩展固件接口）是 Intel 为全新类型的 PC 固件的体系结构、接口和服务提出的建议标准。其主要目的是为了提供一组在 OS 加载之前（启动前）在所有平台上一致的、正确指定的启动服务，被看做是有近20多年历史的PC BIOS的继任者。   
  
　　由于电脑教育普及，很多人都知道BIOS就是Basic Input/Output System，翻成中文是“基本输入/输出系统”，是一种所谓的“固件”，负责在开机时做硬件启动和检测等工作，并且担任操作系统控制硬件时的中介角色。   
  
　　然而，那些都是过去DOS 时代的事情，自从Windows NT出现，Linux 开始崭露头角后，这些操作系统已将过去需要通过BIOS完成的硬件控制程序放在操作系统中完成，不再需要调用BIOS功能。一般来说，当今所谓的“电脑高手”，多半是利用BIOS来对硬件性能做些超频调校，除了专业人士外，鲜有人再利用 BIOS 进行底层工作。   
  
　　因为硬件发展迅速，传统式（Legacy）BIOS 成为进步的包袱，现在已发展出最新的EFI（Extensible Firmware Interface）可扩展固件接口，以现在传统 BIOS 的观点来说，未来将是一个“没有特定 BIOS”的电脑时代。   
  
　　UEFI是由EFI1.10为基础发展起来的，它的所有者已不再是Intel，而是一个称作Unified EFI Form（[www.uefi.org](http://www.uefi.org/" \t "_blank)）的国际组织，贡献者有Intel，Microsoft，AMI，等几个大厂，属于open source，目前版本为2.1。与legacy BIOS 相比，最大的几个区别在于：   
　　1. 编码99%都是由C语言完成；   
　　2. 一改之前的中断、硬件端口操作的方法，而采用了Driver/protocal的新方式；   
　　3. 将不支持X86模式，而直接采用Flat mode（也就是不能用DOS了，现在有些 EFI 或 UEFI 能用是因为做了兼容，但实际上这部分不属于UEFI的定义了）；   
　　4. 输出也不再是单纯的二进制code，改为Removable Binary Drivers；   
　　5. OS启动不再是调用Int19，而是直接利用protocol/device Path；   
　　6. 对于第三方的开发，前者基本上做不到，除非参与BIOS的设计，但是还要受到ROM的大小限制，而后者就便利多了。   
  
　　UEFI将是近3年的趋势，到时候对于PC的利用以及维护都将步入一个新的时代。   
  
  
**UEFI结构**   
　　uEFI概念根据UEFI概念图的结构，我们将把uEFI概念划为两部分：uEFI的实体 （uEFI Image）跟平台初始化框架。   
　　uEFI的实体-uEFI Image（图中蓝框围起部分）   
　　根据uEFI规范定义，uEFI Image包含三种：uEFI Applications, OS Loaders and uEFI Drivers。   
　　uEFI Applications是硬件初始化完，操作系统启动之前的核心应用，比如：启动管理、BIOS设置、uEFI Shell、诊断程式、调度和供应程式、调试应用...等等   
　　OS Loaders是特殊的uEFI Application，主要功能是启动操作系统并退出和关闭uEFI应用。   
　　uEFI Drivers是提供设备间接口协议，每个设备独立运行提供设备版本号和相应的参数以及设备间关联，不再需要基于操作系统的支持。   
　　启动队列   
　　平台初始化框架   
　　uEFI框架主要包含两部分，一是PEI（EFI预初始化），另一部分是驱动执行环境 (DXE)。   
　　PEI主要是用来检测启动模式、加载主存储器初始化模块、检测和加载驱动执行环境核心。   
　　DXE是设备初始化的主要环节，它提供了设备驱动和协议接口环境界面。

**三、MBR** （Master Boot Record，即硬盘的主引导记录）

为了便于理解，一般将MBR分为广义和狭义两种：广义的MBR包含整个扇区（引导程序、分区表及分隔标识），也就是上面所说的主引导记录；而狭义的MBR仅指引导程序而言。  
  
　 　硬盘的0柱面、0磁头、1扇区称为主引导扇区（也叫主引导记录MBR）。它由三个部分组成，主引导程序、硬盘分区表DPT和硬盘有效标志（55AA）。 在总共512字节的主引导扇区里主引导程序（boot loader）占446个字节，第二部分是Partition table区（分区表），即DPT，占64个字节，硬盘中分区有多少以及每一分区的大小都记在其中。第三部分是magic number，占2个字节，固定为55AA。  
  
　　MBR是不属于任何一个操作系统，也不能用操作系统提供的磁盘操作命令来读取它，但可以通过命 令来修改和重写，如在minix3里面，可以用命令：installboot -m /dev/c0d0 /usr/mdec/masterboot来把masterboot这个小程序写到mbr里面，masterboot通常用汇编语言来编写。我们也可以用 ROM-BIOS中提供的INT13H的2号功能来读出该扇区的内容，也可用软件工具Norton8.0中的DISKEDIT.EXE来读取。  
  
　　用INT13H的读磁盘扇区功能的调用参数如下：  
　　入口参数：AH=2 （指定功能号）  
　　AL=要读取的扇区数  
　　DL=磁盘号（0、1-软盘；80、81-硬盘）  
　　DH=磁头号  
　　CL高2位+CH=柱面号  
　　CL低6位=扇区号  
　　CS:BX=存放读取数据的内存缓冲地址  
　　出口参数：CS:BX=读取数据存放地址  
　　错误信息：如果出错CF=1 AH=错误代码  
　　用DEBUG读取位于硬盘0柱面、0磁头、1扇区的操作如下：  
　　A>DEBUG  
　　-A 100  
　　XXXX:XXXX MOV AX,0201 （用功能号2读1个扇区）  
　　XXXX:XXXX MOV BX,1000 （把读出的数据放入缓冲区的地址为CS:1000）  
　　XXXX:XXXX MOV CX,0001 （读0柱面，1扇区）  
　　XXXX:XXXX MOV DX,0080 （指定第一物理盘的0磁头）  
　　XXXX:XXXX INT 13  
　　XXXX:XXXX INT 3  
　　XXXX:XXXX （按回车键）  
　　-G=100 （执行以上程序段）  
　　-D 1000 11FF （显示512字节的MBR内容）  
  
　　在windows操作系统下，例如xp，2003，Vista，windows7，有微软提供的接口直接读写mbr;  
　　FILE \* fd=fopen("\\.\PHYSICALDRIVE0","rb+");  
　　char buffer[512];  
　　fread(buffer,512,1,fd);  
　　//then you can edit buffer[512] as your wish......  
　　fseek(fd,0,SEEK\_SET); //很重要  
　　fwrite(buffer,512,1,fd); //把修改后的MBR写入到你的机器  
　　fclose(fd); //大功告成  
  
　　MBR组成  
　　一个扇区的硬盘主引导记录MBR由如图6-15所示的4个部分组成。  
　　主引导程序（偏移地址0000H--0088H），它负责从活动分区中装载，并运行系统引导程序。  
　　出错信息数据区，偏移地址0089H--00E1H为出错信息，00E2H--01BDH全为0字节。  
　　分区表（DPT,Disk Partition Table）含4个分区项，偏移地址01BEH--01FDH,每个分区表项长16个字节，共64字节为分区项1、分区项2、分区项3、分区项4。  
　　结束标志字，偏移地址01FE--01FF的2个字节值为结束标志55AA,如果该标志错误系统就不能启动。

**四、GPT** （Globally Unique Identifier Partition Table Format ）

一种由基于 Itanium 计算机中的可扩展固件接口 (EFI) 使用的磁盘分区架构。与主启动记录 (MBR) 分区方法相比，GPT 具有更多的优点，因为它允许每个磁盘有多达 128 个分区，支持高达 18 千兆兆字节的卷大小，允许将主磁盘分区表和备份磁盘分区表用于冗余，还支持唯一的磁盘和分区 ID (GUID)。   
  
　　与支持最大卷为 2 TB (terabytes) 并且每个磁盘最多有 4 个主分区（或 3 个主分区，1 个扩展分区和无限制的逻辑驱动器）的主启动记录 (MBR) 磁盘分区的样式相比，GUID 分区表 (GPT) 磁盘分区样式支持最大卷为 18 EB (exabytes) 并且每磁盘最多有 128 个分区。与 MBR 分区的磁盘不同，至关重要的平台操作数据位于分区，而不是位于非分区或隐藏扇区。另外，GPT 分区磁盘有多余的主要及备份分区表来提高分区数据结构的完整性。   
  
　　在“磁盘管理”中的磁盘属性对话框中的“卷”选项卡上，具有 GPT 分区样式的磁盘显示为 GUID 分区表 (GPT) 磁盘，而具有 MBR 分区样式的磁盘显示为主启动记录 (MBR) 磁盘。如果发生下列意外事件，可以在 GPT 磁盘上执行 MBR 磁盘支持的操作：   
  
  
　　在运行带有 Service Pack 1 (SP1) 的 Windows Server 2003 的基于 x86 的计算机和基于 x64 的计算机上，操作系统必须驻留在 MBR 磁盘上。其他的硬盘可以是 MBR 或 GPT。   
  
　　在基于 Itanium 的计算机上，操作系统加载程序和启动分区必须驻留在 GPT 磁盘上。其他的硬盘可以是 MBR 或 GPT。   
  
　　不能将 GPT 移至运行 Windows NT 4.0、Windows 2000、Windows XP 或 Windows Server 2003 的基于 x86 的计算机上。不过，可以将 GPT 磁盘从运行带有 SP1 的 Windows Server 2003 的基于 x86 的计算机或基于 x64 的计算机移至运行 Windows Server 2003 或 Windows XP 的基于 Itanium 的计算机上，反之亦然。   
  
　　不能使用基于 Itanium 的 Windows 版本，将 GPT 磁盘从基于 Itanium 的计算机移至运行带有 SP1 的 Windows Server 2003 的基于 x86 的计算机或基于 x64 的计算机，然后启动该操作系统。在非基于 Itanium 的计算机上使用的 GPT 磁盘必须仅用于数据存储。   
  
　　在单个动态磁盘组中既可以有 MBR，也可以有 GPT 磁盘。也使用将基本 GPT 和 MBR 磁盘的混合，但它们不是磁盘组的一部分。可以同时使用 MBR 和 GPT 磁盘来创建镜像卷、带区卷、跨区卷和 RAID-5 卷，但是 MBR 的柱面对齐的限制可能会使得创建镜像卷有困难。通常可以将 MBR 的磁盘镜像到 GPT 磁盘上，从而避免柱面对齐的问题。   
  
　　可以将 MBR 磁盘转换为 GPT 磁盘，并且只有在磁盘为空的情况下，才可以将 GPT 磁盘转换为 MBR 磁盘。   
  
　　不支持 EFI 系统分区的镜像。必须使用 bootcfg 命令克隆 EFI 系统分区。   
  
　　不能在可移动媒体，或者在与群集服务使用的共享 SCSI 或 Fibre Channel 总线连接的群集磁盘上使用 GPT 分区样式。   
  
　　可以使用 DiskPart.exe 命令行实用程序或 EFI 固件实用程序 Diskpart.efi 在基本 GPT 磁盘上创建分区。有关 DiskPart.exe 的详细信息，请参阅 DiskPart。有关“磁盘管理”管理单元的详细信息，请参阅磁盘管理概述。有关 Diskpart.efi 的详细信息，请参阅 Intel 网站。   
  
　　在基于 Itanium 的计算机上的系统恢复方案中，请参阅计算机所附带的制造商文档来重新创建或恢复 GPT 磁盘。   
  
　　有关管理 GPT 和 MBR 磁盘的详细信息，请参阅可扩展固件接口 或 Microsoft Windows 资源工具包网站上的“Disk Management”（磁盘管理）。