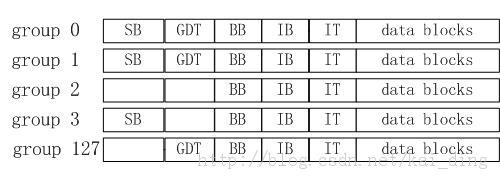
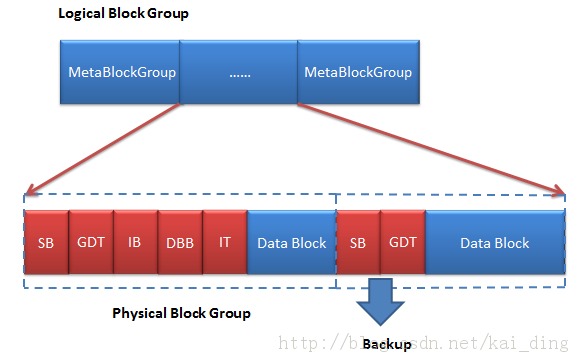
[**ext4文件系统综述**](http://blog.csdn.net/guan506713512/article/details/12001341)

Ext4是Linux文件系统的一次革命。在很多方面，Ext4相对于Ext3的进步要远超过Ext3相对于Ext2的进步。Ext3相对于 Ext2的改进主要在于日志方面，但是Ext4相对于Ext3的的改进是更深层次的，是文件系统数据结构方面的优化。一个高效的、优秀的、可靠的和极具特点的文件系统就此诞生。Ext4相比ext3，有如下几个全新的特性：

* **兼容性**：任何Ext3文件系统都可以轻松的迁移到Ext4文件系统，你只需要在只读模式下运行几条命令即可。这就意味着你完全可以不格式化硬盘、不重装操作系统、不重装软件环境，就能够顺利的升级到Ext4文件系统。这种升级方法不会损害到你硬盘上的数据和资料，因为Ext4仅会在新的数据上使用，而基本不会改动原有数据。
* **更大的文件系统/文件大小：**Ext3支持最大16TB的文件系统，2TB的文件大小。Ext4将支持最大1EB（ ）的文件系统，16TB的文件大小。上述这个特性是由于Ext4采用了48位寻址。有人会问，为什么不采用64位呢？因为就目前的开发进展来看，实现64位寻址存在一些技术限制，但是Ext4已经在考虑这个问题了，在不久的将来，Ext4将实现完全的64位支持。
* **子目录可扩展性：**目前的Ext3中，单个目录下的子目录数目的上限是32000个。而在Ext4中打破了这种限制，可以创建无限多个子目录。
* **Extents：**传统的类UNIX文件系统，比如Ext3，都是使用一个间接数据块映射表来记录每一个数据块的分配情况的。但是这种机制对于超大文件的存储是有缺陷的，特别是当对超大文件进行删除和截断操作时。映射表会对每一个数据块进行记录，而一个超大文件将占有很多的数据块，因此造成映射表将变得无比臃肿，难于维护。Ext4引入了一个新的概念，叫做“Extents”。一个Extents是一个地址连续的数据块的集合。比如一个100MB的文件将被分配给一个单独的Extents，这样就不用像Ext3那样新增25600个数据块的记录（一个数据块是4KB）。而超大型文件会被分解在多个extents里。Extents的实现提高了文件系统的性能，减少了文件碎片。
* **多块分配：**在Ext3中，“将新的数据写入磁盘的哪些空闲块”是由块分配器来控制的。但是Ext3的块分配器存在一定缺陷，那就是它一次只能够分配一个数据块（4KB），这就意味着，如果系统需要向磁盘中写入100MB的数据，那么需要调用块分配器25600次，而且由于块分配器无法获知总的分配块数，所以也无法对分配空间和分配位置进行优化。在Ext4中，使用了“多块分配器”，即一次调用可以分配多个数据块，这种机制提高了系统的性能，而且使得分配器有了充足的优化空间。
* **延迟分配**：延迟分配（Delayedallocation）是一项仅仅少数现代文件系统才具有的优秀特性，比如XFS、ZFS、btrfs(better FS)以及Reiser4。它能够尽可能的积累更多的数据块再分配出去，相对比，传统的文件系统则会尽快的将数据块分配出去，如Ext3，reiser3 等。这项特性会和Extents特性以及多块分配特性相结合，使得磁盘IO性能得到显著提高。
* **更快速的FSCK：**在Ext3中，Fsck本身是个速度很慢的操作，因为它要检查文件系统里的每一个“i节点”。但是，Ext4会维护一个未使用的“i节点”表，在进行 fsck操作时，会跳过表中节点，只检查正在使用中的i节点。这种机制使得fsck的效率提高为原有Ext3文件系统的2到20倍。不过，你要注意到一点，那就是这个未使用的i节点表是由fsck来维护的，而不是由Ext4，因此你必须要首先运行一次fsck来生成，这样，在下次再运行fsck时才可以享受提速。（虽然表是由fsck来维护的，但你还是需要从Ext3升级到Ext4才能够享受这项功能）
* **日志校验:**日志要算是磁盘中最常用的部分了，也是最容易使硬盘出问题的机制之一。如果你不幸使用一个已经崩溃的日志来恢复系统的话，将导致更大规模的系统崩溃。 Ext4提供校验日志数据的功能，可以查看其潜在错误。而且，Ext4还会将Ext3日志机制中的“两阶段提交”动作合并为一个步骤，这种改进将使文件系统的操作性能提升20％。这就是Ext4在日志机制方面对可靠度和性能的双重提升。
* **在线磁盘整理**:这个特性没有包括在内核版本2.6.28之中，但是它很有可能会在下一个版本中引入。虽然Extents、多块分配和延迟分配都有助于减少磁盘碎片，但是磁盘碎片仍然会产生。举例来说：你在一个目录下建立了三个文件（f1，f2，f3），它们被按序写入到连续的一段内存之中。然而几天之后，你想要更新文件f2，也就是位于这段连续内存的中间那一段的那个文件。我要向这个文件中增加一些字符。很明显，在这段连续内存之中已经没有地方放下增加的这些字符，这别无选择，只能将这个f2文件移动到一个能容纳下的新的连续内存之中。这导致了f2文件和f1、f3文件离的非常远，读取也相对缓慢了。看，这就产生了磁盘碎片了。还有，可引导文件应该被放在连续的内存之中，但是磁盘整理机制并不知道哪些文件是可引导文件。为了解决上述问题，Ext4将支持在线磁盘整理，e4defrag工具也被用来支持更智能的磁盘碎片整理功能。
* **结点相关特性:** 更大的i结点：Ext3支持自定义i结点大小，但是默认的i结点大小是128字节，Ex4将默认大小提升到256字节。增加的空间用来存储更多的结点信息，这样有利于提升磁盘性能。 i结点预留机制：当新建一个目录时，若干i结点会被预留下来，等新的文件在此目录中创建时，这些预留的i结点就可以立即被使用。文件的建立和删除将变得更加高效。毫微秒级的时间戳：在Ext3中，时间精度是秒。在Ext4中，时间精度提升到了毫微秒。
* **可持续预分配:** 这个特性，已经出现在了Ext3的最后几个内核版本中，并且也可以由glibc在不支持此功能的文件系统中模拟产生，允许应用程序去预分配磁盘空间。应用程序告知文件系统给预留出一定的空间，文件系统会据此预分配必要的数据块，但是这些数据块将会是空的，直到应用程序向里面写入数据为止。这个机制会常常在P2P应用程序中用到，因为P2P应用程序下载文件常常需要几天的时间。这种机制也防止了磁盘碎片的产生，因为文件系统会一次性分配尽可能连续的数据块给应用程序。再者，这种机制对于实时系统非常重要，因为一旦没有这种机制，可能将会导致在一次重要操作的半截，磁盘空间已满。这项特性是通过调用posix\_fallocate()来实现的。
* **FlexibleBlock Groups。**Starting in ext4, there is a new feature calledflexible block groups (flex\_bg). In a flex\_bg, several block groups are tiedtogether as one logical block group; the bitmap spaces and the inode tablespace in the first block group of the flex\_bg are expanded to include thebitmaps and inode tables of all other block groups in the flex\_bg. For example,if the flex\_bg size is 4, then group 0 will contain (in order) the superblock,group descriptors, data block bitmaps for groups 0-3, inode bitmaps for groups0-3, inode tables for groups 0-3, and the remaining space in group 0 is forfile data. The effect of this is to group the block metadata close together forfaster loading, and to enable large files to be continuous on disk. Backupcopies of the superblock and group descriptors are always at the beginning ofblock groups, even if flex\_bg is enabled. The number of block groups that makeup a flex\_bg is given by 2 ^ sb.s\_log\_groups\_per\_flex.以上描述的是ext4的一个全新特性flexible blockgroups，其基本思想是将多个物理块组（block group）组织成一个逻辑块组（logical blockgroup），每个逻辑块组的第一个物理块组用来存储整个逻辑块组的元数据，如超级块，块组描述符，inode位图，inode等。根据其官方文档的描述，增加这个新特性的目的有2，第一是将块组的元数据集中在一起会提高元数据加载速度；第二是有了更大的连续空间，每个逻辑块组除了第一个物理块组外，剩下的物理块组均可以用于存储文件数据。
* **MetaBlock Groups。**Without the option META\_BG, for safety concerns, all block groupdescriptors copies are kept in the first block group. Given the default128MiB(2^27 bytes) block group size and 64-byte group descriptors, ext4 canhave at most 2^27/64 = 2^21 block groups. This limits the entire filesystemsize to 2^21 ∗ 2^27 = 2^48bytes or 256TiB.The solution to this problem is to use the metablock group feature(META\_BG), which is already in ext3 for all 2.6 releases. With the META\_BGfeature, ext4 filesystems are partitioned into many metablock groups. Eachmetablock group is a cluster of block groups whose group descriptor structurescan be stored in a single disk block. For ext4 filesystems with 4 KB blocksize, a single metablock group partition includes 64 block groups, or 8 GiB ofdisk space. The metablock group feature moves the location of the groupdescriptors from the congested first block group of the whole filesystem intothe first group of each metablock group itself. The backups are in the secondand last group of each metablock group. This increases the 2^21 maximum blockgroups limit to the hard limit 2^32, allowing support for a 512PiB filesystem.The change in the filesystem format replaces the current schemewhere the superblock is followed by a variable-length set of block groupdescriptors. Instead, the superblock and a single block group descriptor blockis placed at the beginning of the first, second, and last block groups in ameta-block group. A meta-block group is a collection of block groups which canbe described by a single block group descriptor block. Since the size of theblock group descriptor structure is 32 bytes, a meta-block group contains 32block groups for filesystems with a 1KB block size, and 128 block groups forfilesystems with a 4KB blocksize. Filesystems can either be created using thisnew block group descriptor layout, or existing filesystems can be resizedon-line, and the field s\_first\_meta\_bg in the superblock will indicate thefirst block group using this new layout.。Ext4系列的文件系统由于其磁盘划分的特点，整个文件系统按块组划分，每个块组最大128MB，每个块组由块组描述符来描述其参数，每个块组描述符大小为64字节，且所有的块组描述符都保存在第一个块组中，因此，这就限制了整个文件系统的大小最大为128MB/64 \* 128MB= 256TB字节，为了解决这一问题，ext4（其实从ext3开始）增加了一个全新特性Meta Block Groups。采用该特性以后，整个文件系统被划分成多个meta block group，其实概念类似上面说的logical blockgroup，都是由多个物理block group组成，每个meta block group的特点是其第一个磁盘块保存整个meta block group的块组描述符，假如文件系统块大小为4KB，则每个meta block group可管理的物理block group是4KB/64 = 64。那么整个meta block group可管理的磁盘空间为128MB \* 64 = 8GB。并且，为了可靠性考虑，每个meta block group的第二个以及最后一个物理block group会保存一份块组描述符的备份。因此，增加了meta block group特性的ext4文件系统的磁盘格式变成如下所示：



        如果考虑到Flexible Block Group和Meta Block Group两种特性的话，那么ext4的磁盘布局很可能是如下图所示：



        如此算是兼顾了两个全新特性的磁盘布局，但这里仍然没有达到Flexible Block Group的最高要求：数据块连续。