EXT4文件系统

===============

EXT4是具有可扩展性和可靠性以增强大型文件系统

（64位）支持的ext3文件系统改进版，

符合增加磁盘容量和先进功能的要求。

邮件列表：linux-ext4@vger.kernel.org

网站：http://ext4.wiki.kernel.org

1.快速使用说明：

===========================

注：更多使用EXT4的扩展信息可以查阅

      ext4的维基网站的网址：

      http://ext4.wiki.kernel.org/index.php/Ext4\_Howto

   - 编译和安装的e2fsprogs的最新版本（这里写作版本1.41.3）从这里：

    http://sourceforge.net/project/showfiles.php?group\_id=2406

或

    ftp://ftp.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/

或获取最新的Git仓库来源：

    git：//git.kernel.org/pub/scm/fs/ext2/e2fsprogs.git

   - 注意非常重要的是要安装在e2fsprogs 1.41.x源码中

/etc/mke2fs.conf 的mke2fs.conf文件。无论您是否

    编辑过在系统上安装的/etc/mke2fs.conf文件，

    你都需要合并你的更改至e2fsprogs1.41.x.

   -使用EXT4文件系统类型创建一个新的文件系统：

    # mke2fs -t ext4 /dev/hda1

    或者配置现有的ext3文件系统以支持扩展：

# tune2fs -O extents /dev/hda1

    如果文件系统用128字节的节点创建时，它可以

    转换为使用256字节以提高效率：

        # tune2fs -I 256 /dev/hda1

    （注：我们目前还没有工具来使的EXT4

    文件系统返回到ext3;所以请不要对生产

    文件系统做这种尝试）。

   - 挂载：

# mount -t ext4 /dev/hda1 /wherever

   - 当与其它文件系统的性能比较，尝试

    多样的工作负担十分重要。;在很多时候

一个工作量参数的微妙变化可以彻底改变

它与其他文件系统比较的排名。在与EXT3比较时，

    注意EXT4默认启​​用写屏障，而EXT3

    没有默认启用写屏障。因此通过'-o barriers=[0|1]'挂载选项

明确指定障碍是否启用对公平的比较是十分有用的。

当调整ext3的最佳参照数字，

    改变数据日志模式是值得尝试的; '-o

data=writeback'对一些工作负担来说更快。（注意

    运行挂载于有数据=写回可能会让陈旧数据

    暴露在最近写文件的情况下像非正常关机，

    这可能在某些情况下是一个安全隐患。）配置

    具有大型日志的文件系统对元数据密集型工作负载

也是很有帮助的。

2.特点

===========

2.1目前可用

\*使用文件系统的能力> 16TB（e2fsprogs支持尚未提供）

\*扩展格式降低了格式元数据开销（内存，IO访问，事物）

\*扩展格式由于magics面对磁盘损坏会更健壮，

\*在树的内部冗余

\*改进的文件分配（多块页头）

\*利用i\_links\_count[1]提升32000子目录限制

\*创建时间，查看时间，修改时间，改动时间的纳秒时间戳

\*磁盘inode上的version域（NFSv4，Lustre）

\*通过uninit\_bg功能减少e2fsck的时间

\*日志校验和稳健性，性能

\*持久性文件预分配（如流媒体，数据库）

\*有能力通过flex\_bg功能打包位图和inode表到更大的虚拟组

\*大文件支持

\*采用通过flex\_bg的大型虚拟块组的inode分配

\*延迟分配

\*大块（最多PAGESIZE）支持

\*在JBD2和EXT4高效的有序模式（避免使用缓冲区头维持有序）

[1]以1K的块大小的文件系统可以看到被强加

目录哈希树具有两层的最大深度的限制。

2.2今后列入的候选特征

\*在线碎片整理（补丁可用，但没有得到很好的测试）

\*通过联结uninit\_bg的懒加载减少mke2fs的时间

（在e2fsprogs中提供这个功能

  但为了安全，一个内核线程懒集合未使用的inode表块

  要在第一次安装的文件系统之后）

还有几个功能正在讨论，他们是否会加入取决于每个

人花了多少功夫在他们身上。像

元数据校验和已经进行了讨论，并计划了一下，但没有补丁，

所以我不确定他们在近期的计划路线上。

最大的性能优化是mballoc，delalloc和flex\_bg

以位图和inode表分组。有些测试结果可在这里看到：

  - http://www.bullopensource.org/ext4/20080818-ffsb/ffsb-write-2.6.27-rc1.html

  - http://www.bullopensource.org/ext4/20080818-ffsb/ffsb-readwrite-2.6.27-rc1.html

3.选项

==========

如果安装了EXT4文件系统，以下选项被接受：

（\*）==默认

RO

以只读挂载文件系统。需要注意的是ext4

重现日志（从而写入

分区），即使是“只读”挂载。该

挂载选项“RO，noload”可以用来防止

写入到文件系统。

journal\_checksum

启用日志事务的校验。

这将允许e2fsck的恢复码和

内核检测损坏内核。这是

兼容的变化，将被老版本的内核忽视。

journal\_async\_commit

提交块可写入磁盘，而无需等待

描述块。如果启用的老内核不能

安装设备。这将启用“journal\_checksum”。

journal\_path=path

journal\_dev=devnum

当外部日志设备的主/次号码

已经改变，这些选项允许用户指定

新日志的位置。该日志设备

通过其设备号产生的新主/次编号

或经由设备的路径确定。

NORECOVERY

Noload

不要在挂载时加载日志。注意

如果文件系统没有卸载干净，

跳过日志重播将导致

文件系统冲突可能

导致任何数量的问题。

data=journal

所有的数据都先提交到日志

之后写入主文件系统。启用

这种模式将禁用延迟分配和

O\_DIRECT支持。

data=ordered（\*）

的所有数据都被迫先输出到文件

系统，然后元数据同步到日志

data=writeback

数据顺序不保留，数据可能会被写入

到主文件系统在它的元数据已被

同步到日记之后。

commit=nrsec（\*）

每一个“nrsec'秒都可让Ext4同步所有的

数据和元数据。默认值是5秒。

这意味着，如果你掉了电，你将失去

最新5秒工作（尽管你的

文件系统不会被破坏，感谢

日志记录）。此默认值（或低值）

会影响性能，但它的良好的数据安全性。

如果设置为0将具有在默认的相同效果

（5秒）。

将其设置到非常大的值将提高性能。

barrier=<0|1(\*)>

barrier(\*)

Nobarrier

此启用/禁用了写屏障的JBD代码。

barrier=0禁用，barrier=1启用。

这也需要一个支持障碍的IO栈

，而如果JBD得到写障碍的

错误，它将禁用障碍并产生一个警告。

写障碍强制执行适当的磁盘上的日志

同步顺序，让不稳定的磁盘缓存用起来更安全

，不过伴随一些性能上的损失。如果

您的磁盘以某种方式进行了电池备份，

禁用屏障可以安全地提高性能。

挂载选项“barrier”和“nobarrier”即可

也可用于启用或禁用的障碍，

与其他EXT4挂载选项一致。

inode\_readahead\_blks = n

这个调整参数控制最大

的ext4的inode表预读算法将预先读入

缓冲区高速缓存的inode表块数。默认值是32个块。

nouser\_xattr

禁用扩展用户属性。查看

ATTR（5）手册页和http://acl.bestbits.at/

有关扩展属性的更多信息。

Noacl

该选项禁用POSIX访问控制列表

支持。如果ACL支持在内核配置中启用

（CONFIG\_EXT4\_FS\_POSIX\_ACL），挂载时ACL

在默认情况下启用。请参阅ACL（5）手册

页面和http://acl.bestbits.at/了解更多信息

有关访问控制列表。

bsddf（\*）

使'DF'像BSD。

Minixdf

使'DF'像Minix的。

Debug

额外的调试信息发送到系统日志。

abort

为了调试的目的，模拟调用ext4\_abort（）

的效果。这通常用于同时

重新挂载它已经挂载的文件系统。

errors=remount-ro

在错误时以只读重新挂载文件系统。

errors=continue

错误时继续执行。

errors=panic

如果发生错误，停止机器。

（这些挂载选项覆盖了在超级块中指定的错误行为

，其可以使用tune2fs配置）

data\_err=ignore(\*)

如果错误发生与有序模式在文件数

据缓冲区，就只打印错误信息。

data\_err=abort

如果错误发生与有序模式在文件数

据缓冲区中断日志。

GRPID

Bsdgroups

给对象相同的组ID作为他们的创造者。

nogrpid（\*）

sysvgroups

新对象有其创建者的组ID。

resgid = N

可以使用保留的块组ID。

resuid=n

其可以使用保留块的用户ID。

SB = N

在此位置使用备用超级块。

quota

noquota

grpquota

usrquota

文件系统忽略这些选项。它只被用来

让配额工具来识别卷确认哪里配额该打开。

有关详细信息见文档中quota-tools包

（http://sourceforge.net/projects/linuxquota）。

jqfmt=<quota type>

usrjquota=<file>

grpjquota=<file>

这些选项告诉配额的文件系统的详细信息

，使得配额信息可以在日志重播时

正确地更新。他们替换上面

配额选项。见配额工具

包中文档的详细信息.

（http://sourceforge.net/projects/linuxquota）。

stripe=n

mballoc将尝试n数量的文件系统的块

用于分配大小和对齐。对于RAID5 / 6

系统中，这应该是文件系统中

磁盘\* RAID块大小的数据的数量。

delalloc（\*）

推迟块的分配，直到在ext4刚输出

有问题的块（多个）。这项

允许EXT4更有效做出更好地分配决策。

Nodelalloc

禁用延迟分配。块被分配

当数据被从用户空间复制到

页缓存，或者通过写系统调用

或者当mmap'ed页以前

未分配并第一次写入。

max\_batch\_time=usec

ext4等待额外同一批去要同步写

的的文件操作所需的最大时间。

由于同步的写操作要强制同步并等

待时间并不长的I/O结束，为了获得吞

吐量的的提升，我们等一段时间看是否

有其他事务要回滚同步写。

使用的算法被设计为自动调节

磁盘的速度，通过测量

它需要完成同步事务的平均

时间。称呼这个时间为

“同步时间”。如果事物运行时间

小于同步时间，EXT4为了同步时间

而休眠以查看是否有其他操作可并入

该事务。提交时间由默认为15000us

的上限max\_batch\_time限制。这种选项

可以由设置max\_batch\_time为0完全关闭。

min\_batch\_time=usec

此参数设置同步时间（如

如上所述）为至少min\_batch\_time。

它默认为零微秒。增加

这个参数可以改善多线的

吞吐量，同步快速磁盘的

工作负荷，以增加延迟为代价。

journal\_ioprio = PRIO

同步操作中由kjournald2提交的

I/O操作的I / O优先级（从0到7，

其中0为最高优先级）。默认为3，

这是比默认的I / O

优先级略高的优先级。

auto\_da\_alloc（\*）

noauto\_da\_alloc

当替换现有文件通过模式

fd = open("foo.new")/write(fd,..)/close(fd)/

rename("foo.new", "foo"),或者更糟糕的，

fd = open("foo", O\_TRUNC)/write(fd,..)/close(fd)时，

许多应用程序崩溃不使用fsync（）。

如果auto\_da\_alloc启用，EXT4将检测

replace-via-rename 与 replace-via-truncate

模式并且强制任何延迟分配的

块被分配，使得在下一

日志同步，默认data=ordered

模式中，新文件的数据块被强制

在rename()操作提交前写入磁盘。

这提供了与ext3大致相同的水平的

保证，并且避免了

“零长度”的问题的发生在系统崩溃

之前，一个延迟分配 块被强迫到磁盘。

noinit\_itable

不要在后台初始化任何未初始化的

inode表块。此功能可能

被安装光盘使用，以便安装

过程可以尽可能快地完成;该

那么inode表初始化进程会

延迟到下一次的文件系统

卸载。

init\_itable = N

懒惰初始化代码将等待n倍

它清零先前块组的节点表的

毫秒数。当文件系统的inode

表被初始化后这参数

最小化对系统性能的影响

。

discard

nodiscard（\*）

控制EXT4是否应该在块被释放时发

出discard/ TRIM命令的底层块设

备时。这对于SSD设备和稀疏/自动

精简配置的LUN非常有用，但它是默认

关闭，直到足够的测试已经完成。

nouid32

禁用32位的UID和GID。这是为

与旧内核的兼容性，只

存储和期待的16位值。

block\_validity

noblock\_validity

该选项允许启用/禁用的内核

设施跟踪文件系统元数据块

的内部数据结构。这使得多

块分配器等程序来快速定位

可能与文件系统元数据重叠

的区域。该选项用于调试

目的并且因为它不利地影响了

性能，默认是关闭的。

dioread\_lock

dioread\_nolock

控制EXT4是否应该使用DIO阅读

锁定。如果指定了dioread\_nolock选项

EXT4将在缓冲写之前分配未初始化的区域

在IO完成后初始化区域。

这种方法允许ext4的代码，以避免

使用inode的互斥体，它提高了高速存储器

的扩展性。然而这对工作

数据日记无效并且dioread\_nolock选项将

忽略内核警告。需要注意的是dioread\_nolock

编码路径只用于基于扩展的文件。

由于包括这一选项的限制

它默认是关闭的（例如dioread\_lock）。

max\_dir\_size\_kb =n

这限制了目录的大小，使得任何

试图扩大目录以至超出规定KB大小

将导致ENOSPC错误。

这在存储器有限制的环境中

非常有用，一个非常大的目录可

导致严重的性能问题，甚至

触发内存不足时结束器。 （例如，

如果仅存在512MB的存储器可用，176mb

目录可能严重限制系统的构成。）

i\_version

启用64位的inode版本支持。此选项

默认关闭。

DAX

使用直接访问（无页面缓存）。见

Documentation/filesystems/dax.txt。注意

此选项与data=journal不兼容。

数据模式

=========

有3种不同的数据模式：

\*写回模式

在data=writeback模式，EXT4没有日志数据。此模式提供

与XFS，JFS，和ReiserFS在默认模式 - 元数据日志

相似的日志记录。崩溃+复苏可能会导致不正确的数据

出现在其中被崩溃前不久的被写过的文件。该模式

通常提供最佳的EXT4性能。

\*有序模式

在data=ordered模式，EXT4只是正式记录元数据，但它在逻辑

上根据称为事务的产生变动的数据块来管理

元数据信息。当它要写新的元数据

到磁盘上，相关的数据块先写入。大体，

这种模式的性能比写回速度慢，但​​比日志模式显著快。

\*日志模式

data = journal方式提供了完整数据和元数据日志。所有新数据是

写入日志，然后再到其最终位置。

在冲突发生时，该日志可以重放，使数据和

元数据处于一致的状态。这个模式是最慢的，除非数据

需要被同时读取和写入到磁盘，在那时

优于所有其他模式。启用该模式将禁用延迟

分配和O\_DIRECT支持。

/ proc中的条目

=============

有关安装的ext4文件系统的信息中可以在/proc/fs/ext4

找到。每一个挂载的文件系统根据其设备名称将有一个目录

在/ proc / FS / EXT4（比如，/ proc / FS / EXT4 / HDC或

/ proc / FS / EXT4 / DM-0）。在每个每设备目录中的文件显示

在下表中。

在/ proc / FS / EXT4 / <DEVNAME>的文件

.................................................. ............................

 文件 内容

 mb\_groups 空闲块的多块分配器的好友缓存的细节

.................................................. ............................

/ SYS项

============

有关挂载的ext4文件系统的信息中可以在/ SYS / FS / EXT4

找到。每一个挂载的文件系统将在/ SYS / FS / EXT4下根据

其设备名称有一个目录（例如，/ SYS / FS / EXT4 / HDC或

/ SYS / FS / EXT4 / DM-0）。在每个每设备目录中的文件显示

在下表中。

文件位于/ SYS / FS / EXT4 / <DEVNAME>

（另见文档/ ABI /测试/ sysfs的-FS-EXT4）

.................................................. ............................

 文件 内容

 delayed\_allocation\_blocks 该文件为只读和显示在页面

缓存中脏块的数量，但

                               不具有在它们在文件系统

                               分配的位置。

 inode\_goal 调整控制被inode分配器使用的目标inode的参数（如果不为零），该分配器优于其他的启发式分配算法。

                               这样做只用于调试

                               ，在生产系统上应该为0。

 inode\_readahead\_blks 调整参数控制着最大

                               的ext4的inode表预读算法将预先读入

                               缓冲区高速缓存的inode表块数的

 lifetime\_write\_kbytes 此文件为只读并显示有多少

                              kb的数据自文件创建后

                               被写入。

 max\_writeback\_mb\_bump 写回代码在移动到下个inode

前会试图写出的最大mb。

 mb\_group\_prealloc如果磁条大小没有于ext4超

级块设定多块分配器会收集

分配请求至调节参数。

 mb\_max\_to\_scan 让多块分配器搜索最

佳区域的最大区域数

 mb\_min\_to\_scan让多块分配器搜索最

佳区域的最小区域数。

mb\_order2\_req 控制最小大小使用伙伴

                              高速缓存的请求（作为2的幂）

的调整参数。

 mb\_stats 控制多块分配器是否应

                              收集显示于卸载的统计数据。

1代表收集的统计数据，0代表

                              不收集统计数据

 mb\_stream\_req 拥有比该可调参数少块的文件，

                              会将他们的块分配到

                              一个块组特定的预分配池，所以

                              小文件联系紧密地打包在一起。

                              每一个大的文件将其块分配

                              到自己独特的预分配池。

 session\_write\_kbytes此文件为只读和显示自

                               挂载后被写入文件的kb。

 reserved\_clusters 这是可读写文件，它包含文件系统中

用于避免清零，ENOSPC异常或可能

的数据丢失等特定情况的保留集群的

数目。默认值是2％或4096群集取小，

这是可以改变，但是它不能超过在文件

                              系统中群集的数量。如果没有足够的空间

                              在保留的空间挂载时

                              挂载将\_not\_失败。

.................................................. ............................

控制操作

======

通过系统调用接口可以由应用程序访问的Ext4

一些特定功能。所有的ext4具体的控制列表

在下表中示出。

表中的ext4特定的控制

.................................................. ............................

Ioctl 描述

 EXT4\_IOC\_GETFLAGS 获取与inode相关联的附加属性。

该控制代码为整数位域，

在ext4.h中用位值描述。这个ioctl是

FS\_IOC\_GETFLAGS的别名。

 EXT4\_IOC\_SETFLAGS设置与索引节点相关联的附加属性。

该控制代码为整数位域，用

在ext4.由位值描述。这个参数是

FS\_IOC\_SETFLAGS的别名。

 EXT4\_IOC\_GETVERSION

 EXT4\_IOC\_GETVERSION\_OLD

获取inode中为每个inode存储的

i\_generation数。该i\_generation数通常只在

创建新的inode时改变，它对于

网络文件系统是特别有用的。该

“\_OLD”版本的该参数是

FS\_IOC\_GETVERSION的别名。

 EXT4\_IOC\_SETVERSION

 EXT4\_IOC\_SETVERSION\_OLD

设置inode中为每个inode存储的

i\_generation数。该“\_OLD”版本的该参数是

FS\_IOC\_SETVERSION的别名。

 EXT4\_IOC\_GROUP\_EXTEND 这个控制代码与挂载选项有相同的目的。

它允许调整文件系统

到最后一个现有块组，

进一步调整大小必须与resize2fs中进行，

或在网上，或脱机。参数点

为的无符号数LOGN数代表了

文件系统的新的块数。

 EXT4\_IOC\_MOVE\_EXT 从orig\_fd将块范围（

控制代码指向的地方）到donor\_fd（

在move\_extent结构中确定被

作为参数传递给这个控制代码）。然后，交换

orig\_fd和donor\_fd之间的inode元数据。

对在线磁盘碎片整理特别有用

，因为分配器具有

机会更好的分配移动的块，

理想情况下分配到一个连续范围。

 EXT4\_IOC\_GROUP\_ADD 添加一个新的组描述到现有或

新的组描述块。新组

描述符是由ext4\_new\_group\_input

结构描述，这是作为参数传递到

此读写控制。这在

与EXT4\_IOC\_GROUP\_EXTEND相结合时尤其有用，

它允许文件系统的在线调整大小

到最后一个现有块组的末尾。

这两个组合的控制中使用用户空间

在线调整大小工具（例如resize2fs）。

 EXT4\_IOC\_MIGRATE 这个控制运行在文件系统本身。

它可以转换（迁移）EXT3间接块映射

inode到以ext4程度映射的inode

通过遍历原始inode的间接块映射

并将连续块的范围

转变为含临时inode的ext4区段。然后，

inode进行交换。当将ext3迁移到ext4文件系统

这个控制参数可能有帮助，

然而建议是创造新的EXT4文件系统

然后从备份中复制数据。注意

文件系统必须支持这个控制参数扩展。

 EXT4\_IOC\_ALLOC\_DA\_BLKS 强制所有延迟分配的块是

保持应用程序预期的ext3

行为分配的。注意，这也将开始

触发数据块的写入，但此

行为可能在将来改变，因为它是

没有必要的并且为了简单起见已经

做这样做了。

 EXT4\_IOC\_RESIZE\_FS 调整文件系统到一个新的大小。

调整大小后的文件系统的块数量经由

64位整型参数被传递。内核中分配

位图和inode表，因此用户空间工具

只是传递新的块数量。

EXT4\_IOC\_SWAP\_BOOT 交换i\_blocks和相关属性

（如i\_blocks，i\_size，i\_flags，...）

跟指定的inode与inode

EXT4\_BOOT\_LOADER\_INO（＃5）。这通常是

用于存储引导加载器中作为文件系统的一个

安全部分，它不能由一个普通用户被偶然的改变。

以前的引导加载程序的数据块

将与给定的inode相关联。

.................................................. ............................

参考

==========

内核源代码：<文件：FS / EXT4 />

<文件：FS / jbd2 />

程序：http://e2fsprogs.sourceforge.net/

有用的链接：http://fedoraproject.org/wiki/ext3-devel

http://www.bullopensource.org/ext4/

http://ext4.wiki.kernel.org/index.php/Main\_Page

http://fedoraproject.org/wiki/Features/Ext4