EXT4

Theodore Ts'o
What's Good About Ext3

- Most widely used filesystem for Linux
- Extremely diverse developer community
  - Good in these days of the economic downturn
  - Distribution expertise needed if they are to feel comfortable supporting the filesystem
What's Not So Good About Ext3

- 16TB filesystem size limitation (32-bit block numbers)
- Second resolution timestamps
- 32,768 limit on subdirectories
- Performance limitations
Is Ext4 really a new filesystem?

- “Ext” in ext2/3/4 stands for “extended”
- The ext4 filesystem driver supports new filesystem features that together makes up what most people consider “ext4”
  - Just as ext3 supports ext2 with some new features, such as “has_journal” and “dir_index”
  - Ext4 (as of 2.6.29) can even support filesystems that do not have the “has_journal” feature flag
- Ext4 fork in 2.6.19 was to make sure that the large ext3 user community would not be affected while ext4 was under development
  - Allowed more experimentation than work was done under ext4.
- Renamed from ext4dev to ext4 in 2.6.28
What's New in Ext4?

- Many new features:
  - Use of extents instead of indirect blocks
  - Delayed Allocation
  - Multiblock Allocation
  - Persistent Allocation
  - Subsecond timestamps
  - NFSv4 version id's for caching
  - Greater than 32000 subdirectories
  - Journal and group descriptor checksums
  - “Huge files” 16TB files on 4k block filesystems
  - ATA TRIM support
  - More intelligent metadata layout

- The most important new feature is Extents
Ext2/Ext3 Indirect Block Map

disk blocks

0
...
200
201
...
213
...
1239
...
65533
...

i_data

0
200
...
...
...
11
211
...
...
...
12
212
...
...
...
13
1237
...
...
...
14
65530
...
...
...

direct block
indirect block
double indirect block
triple indirect block
Extents

- Indirect block maps are incredibly inefficient for large files
  - One extra block read (and seek) every 1024 blocks
  - Really obvious when deleting big CD/DVD image files
- Extents is an efficient way to represent large file
- An extent is a single descriptor for a range of contiguous blocks

<table>
<thead>
<tr>
<th>logical</th>
<th>length</th>
<th>physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>200</td>
</tr>
</tbody>
</table>
On-disk Extents Format

- **12 bytes ext4_extent structure**
  - address 1EB filesystem (48 bit physical block number)
  - max extent 128MB (16 bit extent length)
  - address 16TB file size (32 bit logical block number)

```c
struct ext4_extent {
    __le32  ee_block;       /* first logical block extent covers */
    __le16  ee_len;           /* number of blocks covered by extent */
    __le16  ee_start_hi;    /* high 16 bits of physical block */
    __le32  ee_start;         /* low 32 bits of physical block */
};
```
Ext4 Extent Map

- **header**
  - 0
  - 1000
  - 200
  - 1001
  - 2000
  - 6000
  - ...

- **disk blocks**
  - 200
  - 201
  - ...
  - 1199
  - ...
  - 6000
  - 6001
  - ...
  - 6199
  - ...
  - ...
Extents Tree

- Up to 3 extents could be stored in inode i_data body directly
- Convert to a B-Tree extents tree, for > 4 extents
  - Tree root is stored in inode body (could be in EA or other block)
    - pointing to an index extents block
    - leaf extents block store extents (up to 340 extents)
  - extents look up
    - Leaf/Index extent block is sorted by logical block number
    - Binary search for extent lookup
  - extents insert
    - B-Tree split if leaf block is full
- Last found extent is cached in-memory extents tree
Ext4 Extent Tree

- i_data
- index node
- leaf node
- disk blocks

- header
- extents
- extents index
- node header
Block Allocator changes

• **Needed to best support extents**
  - Extents work best if files are contiguous
  - Delayed allocation and allocating multiple blocks at a time makes this much more likely
  - Responsible for most of ext4's performance improvements

• **Persistent preallocation allows blocks to be assigned to files without initializing first**
  - Most useful for databases and video files
  - Also useful for files that grow gradually via small append operations (i.e., Unix mail files and log files)
  - Can access via posix_fallocate(), but for some applications, direct access to the Linux system call in glibc is needed.
    - When you don't want to fall back to manual initialization
    - When you don't want to change i_size.
Some performance charts....

- Lies, d*mn lies, and benchmarks...
- **What to insist before you believe benchmarks**
  - Are the benchmarks “fair”?
  - Are the benchmarks “repeatable”?
  - Do the benchmarks fairly represent the workload that you care about?
- **One recent good effort:** [http://btrfs.boxacle.net](http://btrfs.boxacle.net)
  - Done by Steven Pratt (who happens to be a member of IBM's Performance Benchmarking team)
  - Hardware and software configurations are documented in detail; multiple configuration are tested.
Large File Creates, Raid, 1 Threads
Large File Creates, Raid, 16 Threads

[Bar chart showing FFSB Ops per Sec with various file systems and their performance metrics.]
Large File Creates, RAID, 128 threads

FFSB Ops per Sec

- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-fs-transactions-fs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodatasum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocow-nosum
Large File Random Reads, RAID, 1 thread

![Bar chart showing FFSB Dps per Sec with various file systems and configurations.]
Large File Random Reads, RAID, 128 threads

FFSB Ops per Sec

- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-fs-transactions-fs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodatasum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocow-nosum
Large File Random Writes, RAID, 1 thread
Large File Random Writes, RAID, 16 threads
Large File Random Writes, RAID, 128 threads
Large File Sequential Reads, RAID, 1 thread
Large File Sequential Reads, RAID, 16 threads

FFSB Ops per Sec

- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-xfs-transactions-jfs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodatasum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocow-nosum
Large File Sequential Reads, RAID, 128 threads

FFSB Ops per Sec

- ext3
- ext4dev
- xfs
- fs
- btrfs
- btrfs-nodatasum
- btrfs-nocov-nosum
Mail Server Simulation, RAID, 16 threads
Mail Server Simulation, RAID, 128 threads
Large File Creates, Single Disk, 8 threads
Large File Random Reads, Single Disk, 1 thread

FFSB Dps per Sec

- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-ifs-transactions-ifs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodacow
- 2.6.27-rc7-btrfs-transactions-btrfs-nodacowum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocow-nocowum

Ops/sec
Large File Random Reads, Single Disk, 8 threads

FFSB Dps per Sec

Ops/sec

2.6.27-rc7-ext3-transactions-ext3
2.6.27-rc7-ext4dev-transactions-ext4
2.6.27-rc7-xfs-transactions-xfs
2.6.27-rc7-ufs-transactions-ufs
2.6.27-rc7-btrfs-transactions-btrfs
2.6.27-rc7-btrfs-transactions-btrfs-nodacow
2.6.27-rc7-btrfs-transactions-btrfs-nodacsum
2.6.27-rc7-btrfs-transactions-btrfs-nocw-nocsum
Large File Random Reads, Single Disk, 32 threads
Large File Random Writes, Single Disk, 1 thread
Large File Random Writes, Single Disk, 8 threads
Large File Random Writes, Single Disk, 32 threads

FFSB Dps per Sec

Operating Systems:
- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-ufs-transactions-ufs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodacow
- 2.6.27-rc7-btrfs-transactions-btrfs-nodacowum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocw-nocowum
Large File Seq Reads, Single Disk 32 threads

FFSB Dps per Sec

- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-ifs-transactions-ifs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodelacow
- 2.6.27-rc7-btrfs-transactions-btrfs-nodelacowsum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocow-nocowsum
Mail Server Simulation, Single Disk, 1 thread

FFSB Ops per Sec

Ops/sec

0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400

2.6.27-rc7-ext3-transactions-ext3
2.6.27-rc7-ext4dev-transactions-ext4
2.6.27-rc7-xfs-transactions-xfs
2.6.27-rc7-ufs-transactions-ufs
2.6.27-rc7-btrfs-transactions-btrfs
2.6.27-rc7-btrfs-transactions-btrfs-nodacow
2.6.27-rc7-btrfs-transactions-btrfs-nodacowsum
2.6.27-rc7-btrfs-transactions-btrfs-nosum
Mail Server Simulation, Single Disk, 32 threads

FFSB Ops per Sec

- 2.6.27-rc7-ext3-transactions-ext3
- 2.6.27-rc7-ext4dev-transactions-ext4
- 2.6.27-rc7-xfs-transactions-xfs
- 2.6.27-rc7-fs-transactions-fs
- 2.6.27-rc7-btrfs-transactions-btrfs
- 2.6.27-rc7-btrfs-transactions-btrfs-nodacow
- 2.6.27-rc7-btrfs-transactions-btrfs-nodatacow
- 2.6.27-rc7-btrfs-transactions-btrfs-noaccum
- 2.6.27-rc7-btrfs-transactions-btrfs-nocum
e2fsck Performance

- Not something that we had explicitly engineered
- Improvements from
  - Fewer extent tree blocks to read instead of indirect blocks
  - Uninitialized block groups means we don't have to read portions of the inode table

<table>
<thead>
<tr>
<th></th>
<th>e2fsck on ext3</th>
<th>e2fsck on ext4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>MB read</td>
</tr>
<tr>
<td>Pass 1</td>
<td>192.3</td>
<td>1324</td>
</tr>
<tr>
<td>Pass 2</td>
<td>11.81</td>
<td>260</td>
</tr>
<tr>
<td>Pass 3</td>
<td>0.01</td>
<td>1</td>
</tr>
<tr>
<td>Pass 4</td>
<td>0.13</td>
<td>0</td>
</tr>
<tr>
<td>Pass 5</td>
<td>6.56</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>211.1</strong></td>
<td><strong>1588</strong></td>
</tr>
</tbody>
</table>

80 gig filesystem on a laptop drive
Using ext4

- Need e2fsprogs 1.41.8
- Need 2.6.27 kernel or newer. Strongly recommend 2.6.30
- Need a filesystem to mount
  - Can use existing unconverted ext3 (or ext2) filesystem.
  - Can convert an existing ext3 filesystem:
    - Tune2fs -O extents,huge_file,dir_nlink,dir_isize /dev/sdXX
    - Optional: can add uninit_bg and dir_index to the above, but then you must run “e2fsck -pD /dev/sdXX”
  - Can create a fresh ext4 filesystem mke2fs -t ext4 /dev/sdXX
- Shipping in some community distributions
  - Fedora 11
  - Ubuntu 9.04 (but must upgrade to a mainline kernel)
Getting involved

- **Mailing list:** [linux-ext4@vger.kernel.org](mailto:linux-ext4@vger.kernel.org)
- **latest ext4 patch series**
- **Wiki:** [http://ext4.wiki.kernel.org](http://ext4.wiki.kernel.org)
  - Still needs work; anyone want to jump in and help, talk to us
  - Import and improve content from [http://kernelnewbies.org/Ext4](http://kernelnewbies.org/Ext4)
- **Weekly conference call; minutes on the wiki**
  - Contact us if you'd like dial in
- **IRC channel:** irc.oftc.net, /join #ext4
The Ext4 Development Team

- Alex Thomas (Sun)
- Andreas Dilger (Sun)
- Theodore Tso (IBM/Linux Foundation)
- Mingming Cao (IBM)
- Suparna Bhattacharya (IBM)
- Dave Kleikamp (IBM)
- Badari Pulavarathy (IBM)
- Avantikia Mathur (IBM)
- Aneesh Kumar (IBM)
- Eric Sandeen (Red Hat)
- Jan Kara (SuSE)
• This work represents the view of the author(s) and does not necessarily represent the view of IBM or of the Linux Foundation.

• IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.

• Linux is a registered trademark of Linus Torvalds.

• Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

• Other company, product, and service names may be trademarks or service marks of others.