Ext4: The Next Generation of Ext2/3

Theodore Ts'o
IBM
Agenda

- New features in ext3 you might not know about
- What's cool about ext3
- What's not
- Why fork ext3->ext4?
- New ext4 features on deck
- How to get involved
New features in ext3 (you might not know about)

- Directory indexing (hash tree directories)
  - To enable:
    - `tune2fs -O dir_index /dev/XX`
    - `e2fsck -fD /dev/XX`
  - Can make some workloads slower
- Online resizing
  - Need to create filesystem with `-O resize_inode`
  - Requires e2fsprogs 1.36 or greater, default in e2fsprogs 1.39
- Various performance enhancements
What's cool about ext3

- Very large user community
- Very large developer community
  - From a large number of companies:
    - Red Hat, IBM, Bull, Clusterfs, Google, NEC, others
- Emphasis on robustness above all else
  - Simple filesystem format
  - “PC Class hardware sucks”
What's not so cool about ext3?

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

- No development 2.7 tree
  - ... and changes take longer than the 2-3 months between 2.6 releases
- Large userspace community
  - People like davem, rusty, akpm, torvalds get cranky if their source trees get trashed
- Many changes on-deck require format changes
- Allows more experimentation than if the work is done outside of mainline
  - Make sure users understand that ext4 is risky: mount -t ext4dev
- Downsides
  - bug fixes must be applied to two code bases
  - smaller testing community
Features (under development)

- Ability to use > 16TB filesystems (going beyond 32-bit block numbers)
- Replacing indirect blocks with extents
- More efficient block allocation
- Allow greater than 32k subdirectories
- Nanosecond timestamps
- Metadata checksumming
- Uninitialized groups to speed up mkfs/fsck
- Persistent file allocation
- Online defragmentation
Extents

- Indirect block maps are incredibly inefficient
  - One extra block read (and seek) every 1024 blocks
- Really obvious when deleting big CD/DVD image files
Ext2/3 Indirect Block Map

i_data

0  200
1  201
...  ...
...  ...
11  211
12  212
13  1237
14  65530

<table>
<thead>
<tr>
<th>direct block</th>
<th>indirect block</th>
<th>double indirect block</th>
<th>triple indirect block</th>
</tr>
</thead>
</table>

disk blocks
Extents

- 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>
Extent Tree

i_data

index node

leaf node

disk blocks

header

root

0

0

0

...
Extent Related Works

- Multiple block allocation
  - Allocate contiguous blocks together
    - Reduce fragmentation, reduce extent meta-data
    - Stripe aligned allocations

- Delayed allocation
  - Defer block allocation to writeback time
  - Improve chances allocating contiguous blocks, reducing fragmentation
  - Trickier to implement in ordered mode
tiobench sequential write

Throughput (MB/sec)

- 4 threads
- 16 threads
- 64 threads

- ext3 2.4.29
- ext3 2.6.11
- JFS
- XFS
64-bit block numbers

- Most of the hard work done as part of the extents changes
  - Original lustre patches provide 48-bit block numbers. Enough for a $2^{60}$ filesystems
- Other changes
  - Superblock fields
  - Block group descriptors (required doubling their size)
  - Journal inode (new jbd2 layer)
Expanded inode

- Inode size is normally 128 bytes in ext3
- But can be 256, 512, 1024, etc. up to filesystem blocksize
- Extra space used for fast extended attributes
- 256 bytes needed for ext4 features
  - Nanosecond timestamps
  - Inode change version # for Lustre, nfsv4
Persistent file allocation

- Allow applications to preallocate blocks for a file without having to initialize them
  - Contiguous allocation to reduce fragmentation
    - Irrespective of order that blocks are written
    - While avoiding overhead of zeroing blocks
  - Guaranteed space allocation
  - Useful for Streaming audio/video, databases

- Implemented as uninitialized extents
  - MSB of ee_len used to flag “invalid” extents
  - Reads return zero
  - Writes split the extent into valid and invalid extents
Metadata checksuming

- Proof of concept implementation described in the Iron Filesystem paper (from University of Wisconsin)
- Storage trends: reliability and seek times not keeping up with capacity increases
- Add checksums to extents, superblock, block group descriptors, inodes, journal
Getting involved

- Mailing list: linux-ext4@vger.kernel.org
  - Still needs work; anyone want to jump in and help, talk to us
- Weekly conference call; minutes on the wiki
  - Contact us if you'd like dial in
- IRC channel: irc.oftc.net, /join #linuxfs
The Ext4 Development Team

- Alex Thomas
- Andreas Dilger
- Theodore Tso
- Stephen Tweedie
- Mingming Cao
- Dave Kleikamp
- Badari Pulavarathy
- Avantikia Mathur

- Andrew Morton
- Laurent Vivier
- Alexandre Ratchov
- Eric Sandeen
- Takashi Sato
Conclusion

- Ext4 work just beginning
- Extents merged, other patches on deck
- Should be a lot of fun!
- Watch this space....