Speeding up file system checks in ext4

Theodore Ts'o

THE LINUX FOUNDATION

Why File System Checks Are Necessary

- **Software is not perfect**
  - Bugs in kernel (File system, VM, Device Driver code)
- **Hardware is not perfect**
  - Disk errors
  - Memory errors
    - File system checksums don't protect against corruption in memory

Why we need fast file system checkers

- **MTBU – “maximum time belly up”**
- **File system checks become less useful when it takes significantly longer than restoring from backups**
  - Assuming, off course, that backups are available!

Original e2fsck optimizations

- **The original version of e2fsck was based on fsck.minix, written by Linus Torvalds**
- **E2fsprogs was developed to create a faster fsck**
  - Based on ideas from “A Faster fsck for BSD Unix” by Bina & Emrath, Winter 1989 Usenix Technical Conference
  - Speeded up e2fsck by factor of 6-8 times
- **Key Ideas**
  - Cache as much information as possible in pass 1 and pass 2
    - In the normal case, each file system meta data block should only be read once
  - Read directory blocks in pass 2 in sorted order to avoid seek penalties
  - Read inodes with indirect blocks in sorted order in pass 1 to avoid seek penalties

Summary of e2fsck's operation

- **Pass 0 – basic superblock**
- **Pass 1 – inode table and indirect blocks/extents**
  - Iterate over all inodes and all indirect/extent tree blocks
  - Cache location of directory blocks and inode type info
  - 70-90% of total e2fsck time
- **Pass 2 – directory structures**
  - Read all directory blocks
  - Store all parent directory information for pass 3
  - 10-25% of total e2fsck time
- **Pass 3 – directory connectivity**
  - Make sure all directories are reachable from the root
- **Pass 4 – inode reference counts**
- **Pass 5 – block and inode allocation bitmaps**

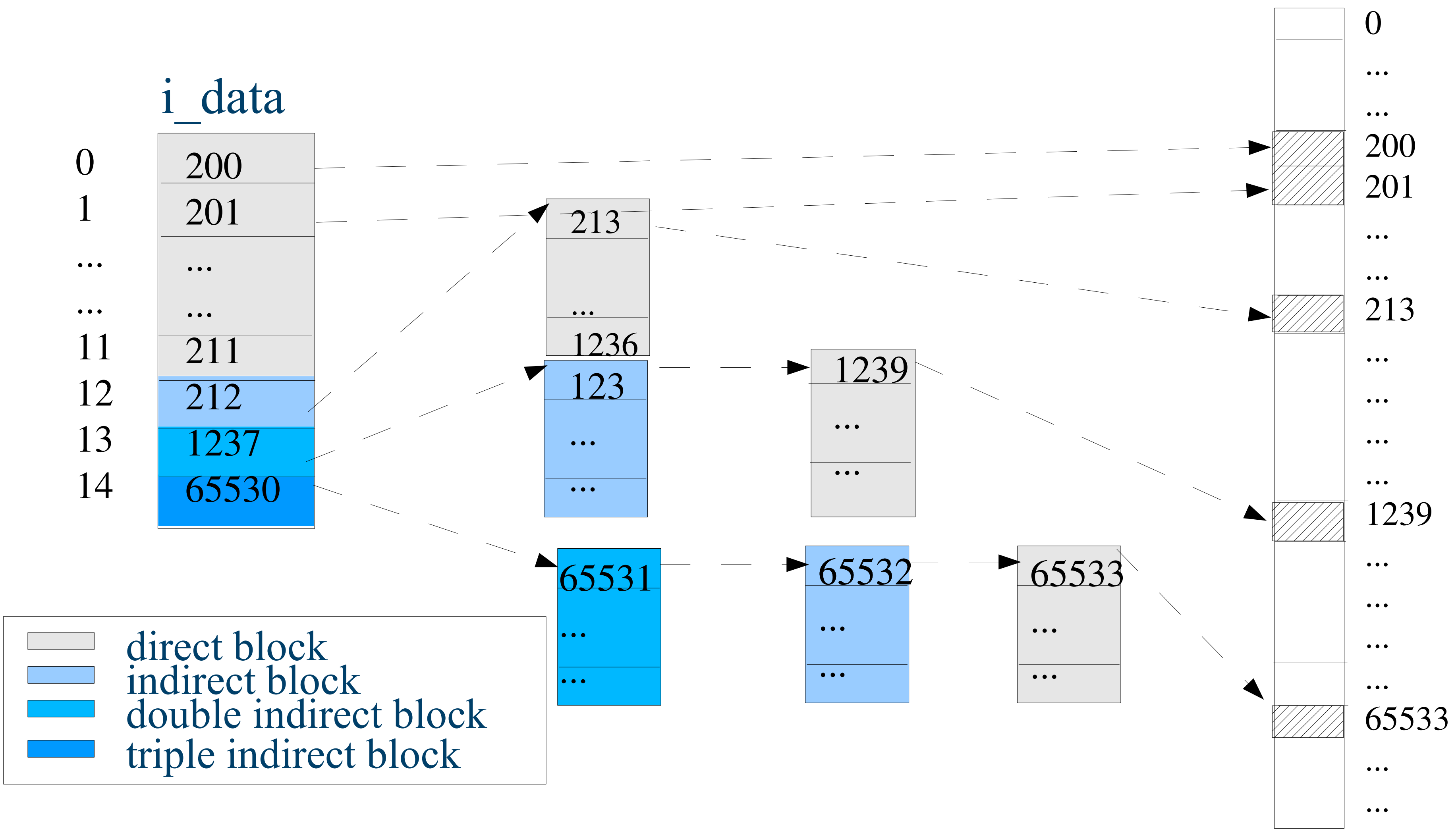
Optional e2fsck passes

- **Pass 1b/1c/1d – multiply claimed blocks handling**
  - In the case that one or more blocks are claimed by more than one inode
  - Pass 1b – record all of the inodes that reference each multiply claimed block
  - Pass 1c – scan directory blocks so we can report these inodes using full pathnames (and not just an inode number)
  - Pass 1d – for each inode, prompt whether the multiply claimed blocks should be cloned, or the inode deleted
- **Pass 3a – directory optimization**
  - E2fsck will invoke pass 3A under two conditions
    - If a directory is corrupted, to recreate the hash tree data structures
    - To optimize all directories if the -D option was passed to e2fsck

Speeding up fsck for ext4

- Using extents instead of indirect blocks
- High watermark for each block group's inode table
- Directory block allocation algorithm

Ext2/Ext3 Indirect Block Map



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
  - Every single indirect block must be read by e2fsck
- **Extents are a more efficient way to represent large files**
- **An extent is a single descriptor for a range of contiguous blocks**

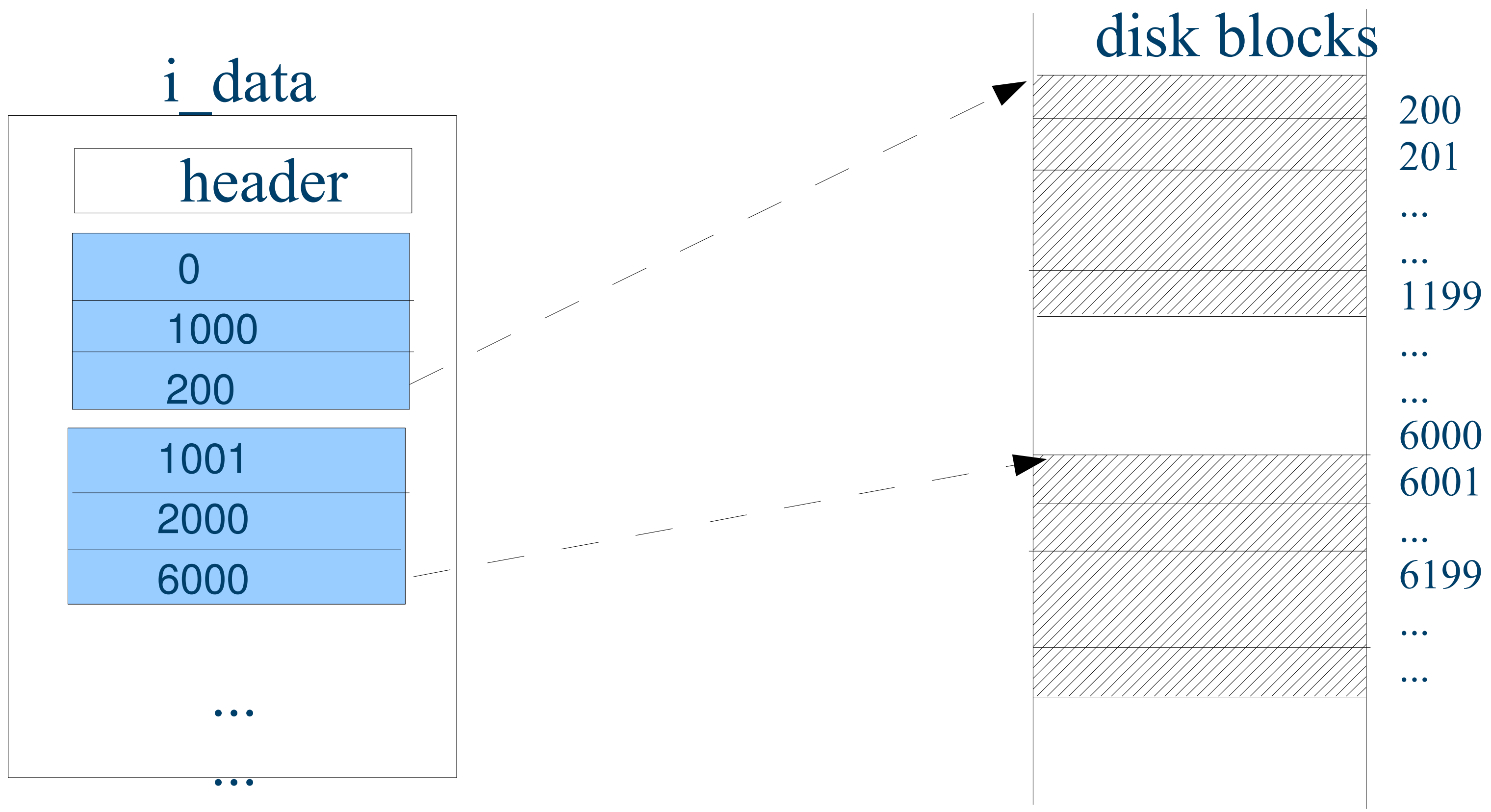
logical	length	physical
0	1000	200

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)

```
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 */
};
```

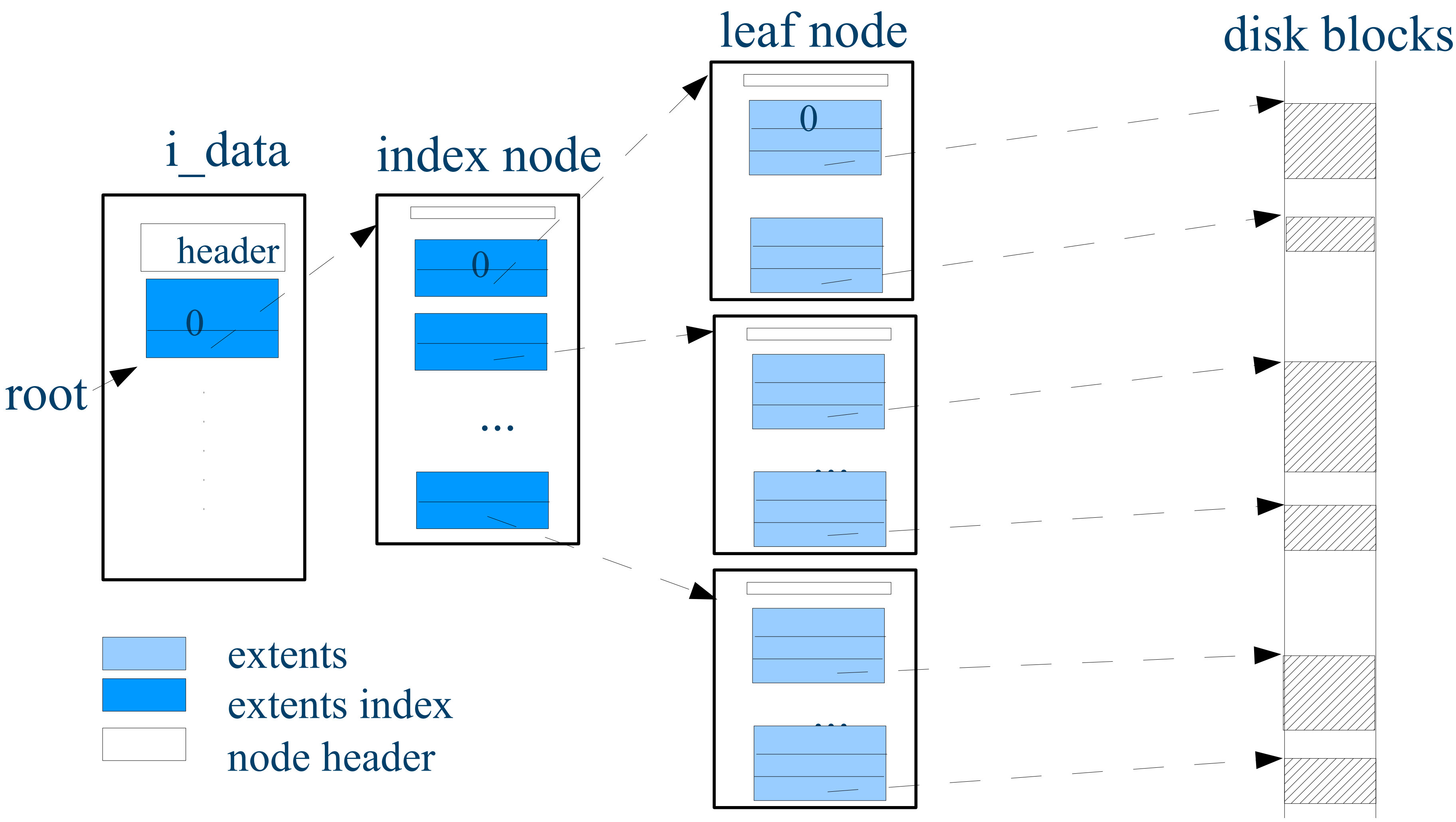
Ext4 Extent Map



Extents Tree

- **Up to 4 extents can be stored in inode i_data body directly**
- **Convert to an extents tree for > 4 extents**
  - Tree root is stored in inode body (could be in EA or other block)
    - pointing to a index extents block
    - leaf extents block store extents (up to 340 extents)
  - extents lookup
    - 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**

Ext4 Extent Tree



Test image for doing e2fsck comparisons

- **Sample filesystem taken from an Ubuntu 9.04 laptop**
  - 70GB, originally taken from an SSD drive filesystem
  - 41% blocks used, 18% inodes used
  - 6.7% directories, 89.0% regular files, 4.0% symlinks, 0.3% devices
  - Copied via rsync to freshly created ext3 and ext4 filesystem
    - Used same partition for both ext3 and ext4 tests

      5400 rpm laptop drive, raw speed measured via hdparm: 71.55 MB/s
- **Summary fsck times**
  - Ext3: 211.0 seconds, 1588 MB read, 7.52 Mb/s
  - Ext4: 18.75 seconds, 466 MB read, 24.85 Mb/s

Net savings for using extents

- **Ext3 filesystem**
  - Number of inodes with indirect blocks: 40,860
  - Number of inodes with double indirect blocks: 394
  - Amount of indirect block metadata: 199MB
  - Difference in pass 1 times between e2fsck of an empty and populated file system: 170.56 seconds
    - 21.74 seconds, 51.76 Mb/s vs. 192.30 seconds, 6.89 Mb/s
- **Ext4 filesystem**
  - Number of inodes with a depth > 1 extent tree: 60
  - Difference in pass 1 times between e2fsck of an empty and populated file system: 9.80 seconds
    - 0.07 seconds, 56.57 Mb/s vs 9.87 seconds, 20.56 MB/s

Skipping unused inodes in the inode table

- **Ext 2/3/4 uses fixed inode table**
  - Advantages: robustness (always know where inodes can be found)
  - Disadvantages: wastes space (typical metadata overhead 1.82%), slows down mke2fs and e2fsck
- **If we can reliably know how much of the inode table is actually in use, we can skip the unused portion**
  - Requires checksummed block group descriptors for safety
  - Eventually can speed up mke2fs time as well
- **Net savings for skipping unused inodes**
  - Pass 1 time for an empty file system with ext3: 21.74 seconds
  - Pass 1 time for an empty file system with ext4: 0.07 seconds
  - Pass 5 time for an empty file system with ext3: 6.56 seconds
  - Pass 5 time for an empty file system with ext4: 2.24 seconds
  - Actual time saved will depend on how much of the inode table is actually in use

File system layout improvements

- **In the traditional ext3 layout, the metadata for each block group (inode table, block/inode allocation bitmaps) is located at the beginning of each block group**
  - With 4k file system blocks, block groups are 128 Mb each
  - This means that files > 128 MB can not be contiguous
- **In ext4, the block groups are grouped together into "flex_bg groups"**
  - By default mke2fs uses 16 block groups/flex_bg group (must be power of 2)
  - The inode table and bitmaps are placed at the beginning of the flex_bg group (in the first block group)
- **Idea for improving pass 2 times, reserve the first block group in each flex_bg group for extent tree blocks and directory blocks**
  - This reduces seek times when reading the directory blocks
  - Reduces pass 2 times by 46% (11.81 seconds vs. 6.34 seconds)

Overall e2fsck performance summary

- **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
  - Directory blocks are allocated so they are grouped together to speed up pass #2

	e2fsck on ext3			e2fsck on ext4	
	time	**MB read**		**time**	**MB read**
Pass 1	192.3	1324		9.87	203
Pass 2	11.81	260		6.34	261
Pass 3	0.01	1		0.01	1
Pass 4	0.13	0		0.18	0
Pass 5	6.56	3		2.24	2
Total	211.1	1588		18.75	466

Summary of ext4 improvements

- **Better Performance**
  - Extents help performance for large files
  - Better block and inode allocation
  - More efficient journal commits
  - File preallocation
- **Increased protection for data integrity**
  - Barriers on by default
  - Metadata checksums
- **New file system features**
  - Fine grained time stamps
  - Better support for NFSv4
- **Better file system scalability**
  - File system sizes up to 1 exabyte
  - > 32000 sub directories
- **Faster file system checks**

How to use ext4

- **Shipping in some community distributions**
  - Fedora 11
  - Ubuntu 9.04 (but must upgrade to a mainline kernel)
  - Technology previews in latest SLES and RHEL update releases
- **To roll your own**
  - Need e2fsprogs 1.41.9
  - Need 2.6.27 kernel or newer. Strongly recommend 2.6.31
  - Need a file system to mount

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

Getting involved

- **Mailing list: linux-ext4@vger.kernel.org**
- **latest ext4 patch series**
  - git://git.kernel.org/pub/scm/linux/kernel/git/tytso/ext4.git
  - http://www.kernel.org/pub/scm/linux/kernel/git/tytso/ext4.git
  - ftp://ftp.kernel.org/pub/linux/kernel/people/tytso/ext4-patches
- **Wiki: 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
- **Weekly conference call; minutes on the wiki**
  - Contact us if you'd like dial in
- **IRC channel: irc.oftc.net, /join #ext4**

The Ext4 Developers

- Alex Thomas (Sun)
- Andreas Dilger (Sun)
- Theodore Tso (IBM/Linux Foundation)
- Mingming Cao (IBM)
- Dave Kleikamp (IBM)
- Aneesh Kumar (IBM)
- Eric Sandeen (Red Hat)
- Jan Kara (SuSE)
- Akira Fujita (NEC)
- Curt Wohlgemuth (Google)
- Frank Mayhar (Google)

Legal Statement

- 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.