The Linux ext2/3/4 Filesystem: Past, Present, and Future

Theodore Ts'o
IBM Linux Technology Center
September 11, 2006

Agenda

- A brief history of the ext2/3 filesystem
- The ext3 filesystem format
- Features added to ext3 in Linux 2.6
- New features planned for ext3/4
- Why ext4?
- Conclusion

A brief history of Linux filesystems

- The Minix filesystem (1991, used to bootstrap Linux)
  - Max FS size: 64MB
  - Max file size: 64MB
  - Max filename: 14/30 bytes (fixed-length directory entries)
  - Only supported modification timestamp
- First attempt to improve on Minixfs: the ext filesystem (1992)
  - Max FS size: 2GB
  - Max file size: 2GB
  - Max filename: 255 bytes
  - Still only one timestamp
  - Linked lists for free block/inodes caused performance problems

The xiafs and ext2fs filesystems

- Xiafs: minimal changes from minix (January 1993)
  - Max FS size: 2GB
  - Max file size: 64MB (instead of 2GB)
  - Max filename: 248 bytes (fixed-length directory entries)
  - ctime/mtime/atime timestamps
- Ext2fs – improvements to extfs (January 1993)
  - Max FS extended to 4TB
  - Variable block sizes
  - ctime/mtime/atime timestamps
  - Improved block/inode allocation using bitmaps and block groups

Competition between xiafs and ext2fs

- Since xiafs only made minor changes to minix, it was initially (appeared) more stable.
- Frank Xia tried to rename xiafs to Linuxfs – negative reaction to marketing-driven changes
- Ext2 had a larger development community (so more features added) and had a more scalable design. In the end it became the dominant “default” filesystem
- Features added to ext2 over the years
  - sparse superblocks
  - Large file support (> 2GB)
  - Extended attributes
  - ACL's

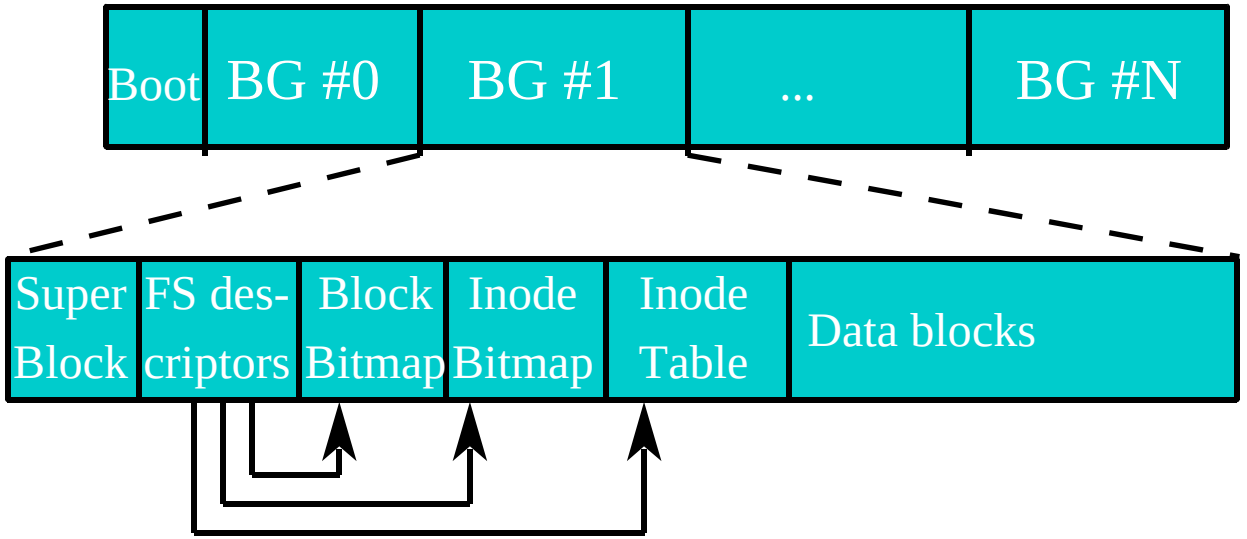
The ext3 filesystem

- Journalling added to ext2 in 2000 (work started in 1998).
- Since it required many changes to the code base, a new version of the filesystem code was created in the kernel.
  - Hence, ext3
  - But really just ext2 with the COMPAT_HAS_JOURNAL feature (from the filesystem format point of view)
- Other Journaling Filesystems
  - Reiserfs, JFS, XFS
- Advantages of ext3
  - Backwards compatibility with ext2
  - Robustness against hardware errors highest priority

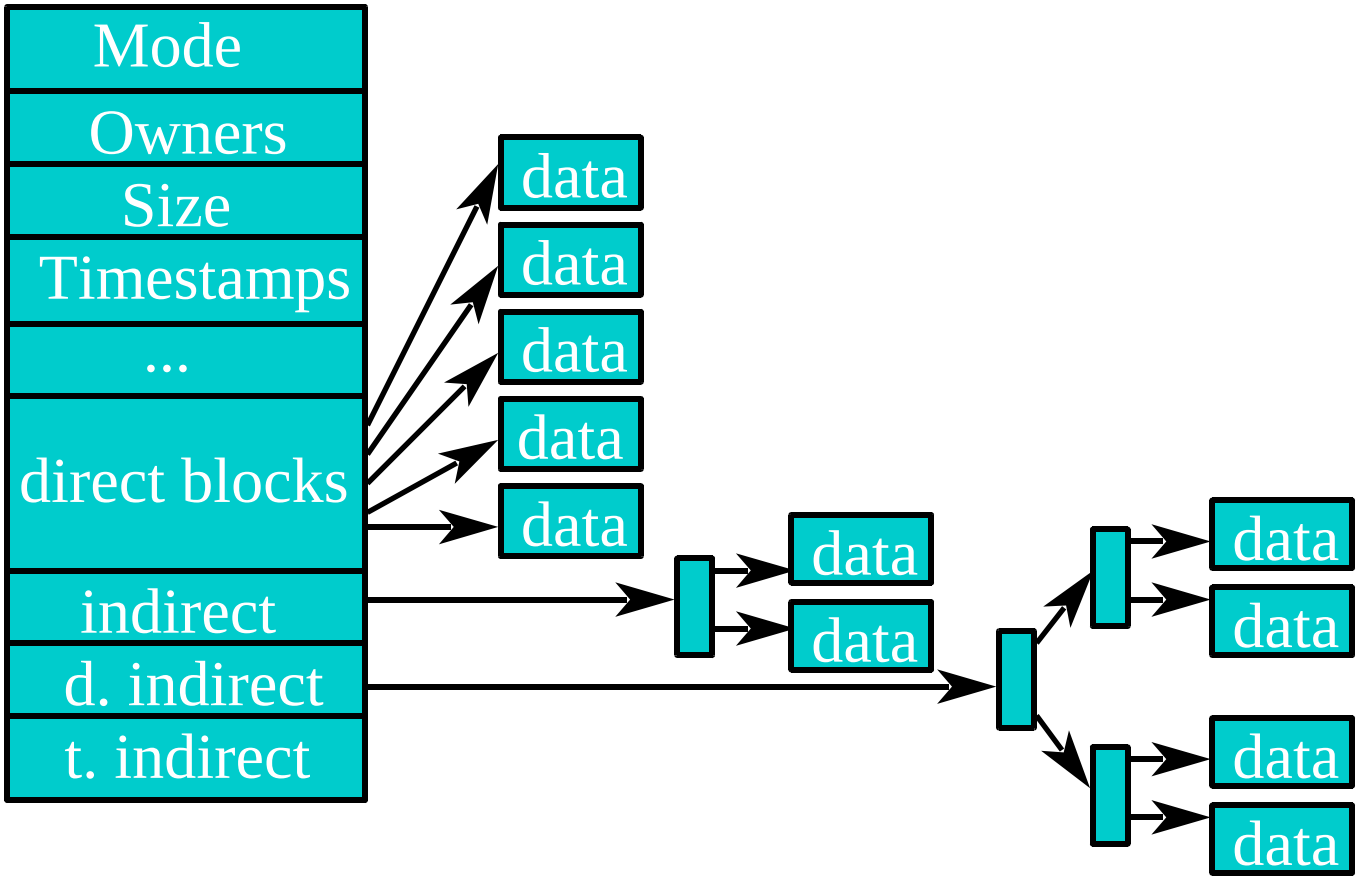
The ext2/3 filesystem format

- Verfy similar to the BSD FFS
- Cylinder groups have become “block groups”
- Compatibility feature sets allow controlled addition of new features via three bitmasks:
  - R/W Compat – The kernel may mount the filesystem even if it does not understand a feature in this bitmask. (E2fsck however will refuse to touch a filesystem it doesn't understand)
  - R/O Compat – The kernel may mount the filesystem read/only if it does not understand a feature in this bitmask
  - Incompat – The kernel must not mount the filesystem if it does not understand a feature in this bitmask

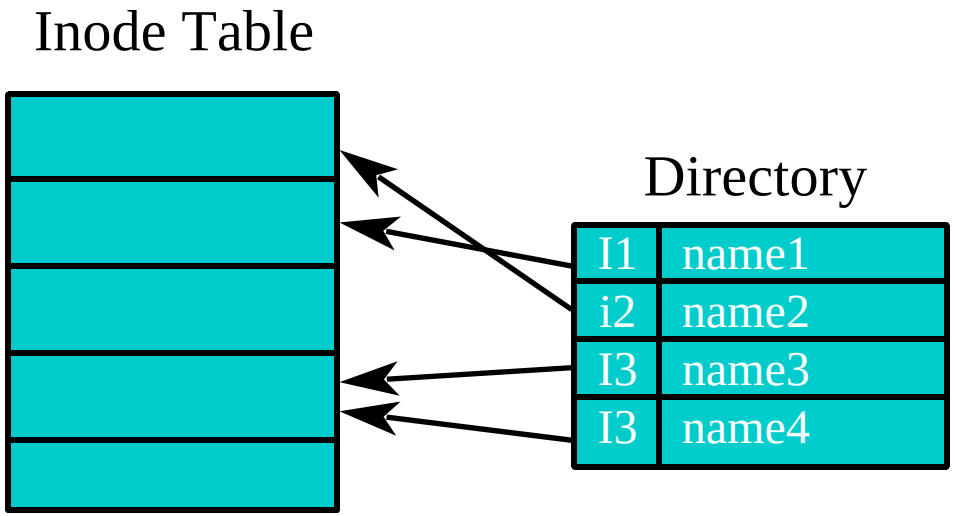
Ext2 Filesystem Layout



Ext2 Inode structure



Ext2 Directory Layout



Features added to Linux 2.6

- BKL removal and other scalability improvements (Andrew Morton, Alex Thomas)
- Directory Indexing (Daniel Phillips, Theodore Ts'o)
- Extended Attributes (Andreas Gruenbacher)
- Online resizing (Andreas Dilger, Stephen Tweedie)
- Reservation-based block preallocation (Mingming Cao, Andrew Morton, Stephen Tweedie, Badari Pulvarty)

Reducing Lock Contention

- Motivation: scaling issues for 2.4's ext3/jbd under workloads with concurrent I/O
- To address this problem:
  - replaced the per-filesystem superblock lock in ext3 with finer-grained locks
  - Removed the big (global) kernel lock from the JBD layer
- Result: SDET benchmark throughput improved by a factor of 10

Directory Indexing

- Motivation: large directories took a long time to search
- Solution: Add a search tree indexed by the hash of the filename to the directory
  - Variation of a B+tree
    - Directory entries stored in only leaf nodes
    - The use of fixed-length, 64-bit hashes as keys results in a high fanout factor
- Fully backwards compatible with older kernels
  - Interior nodes look like deleted directory entries
  - Older kernels will clear the directory indexed bit when they modify a directory, thus invalidating the interior nodes until they can be regenerated.

Extended Attributes

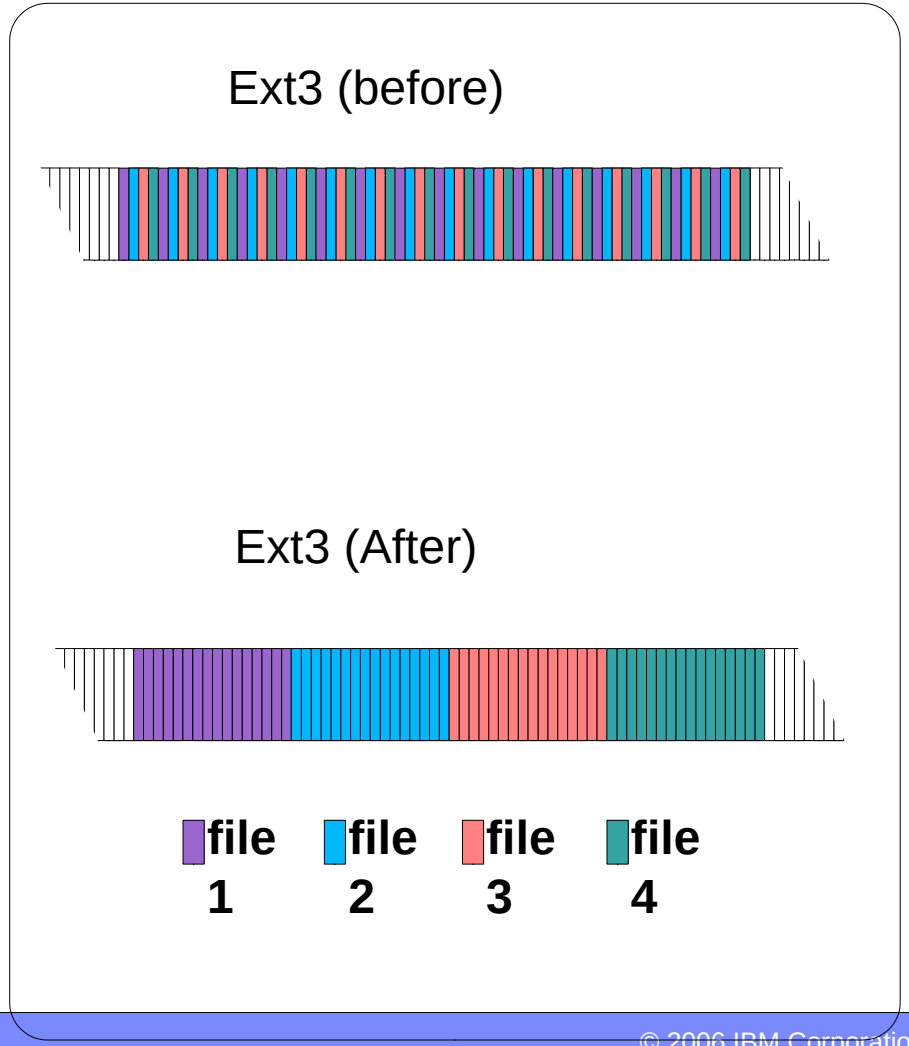
- Motivation: need to store small amounts of custom metadata which is associated with files or directories
  - Also needed to support Access Control Lists (ACL's)
- EAs are stored in a single EA block, which can be shared by inodes have same extended attributes
- In Linux 2.6.11+, EA's can be stored in the expanded inode as well.
  - This EA-in-inode makes the ext3 top filesystem on Samba4 benchmarks

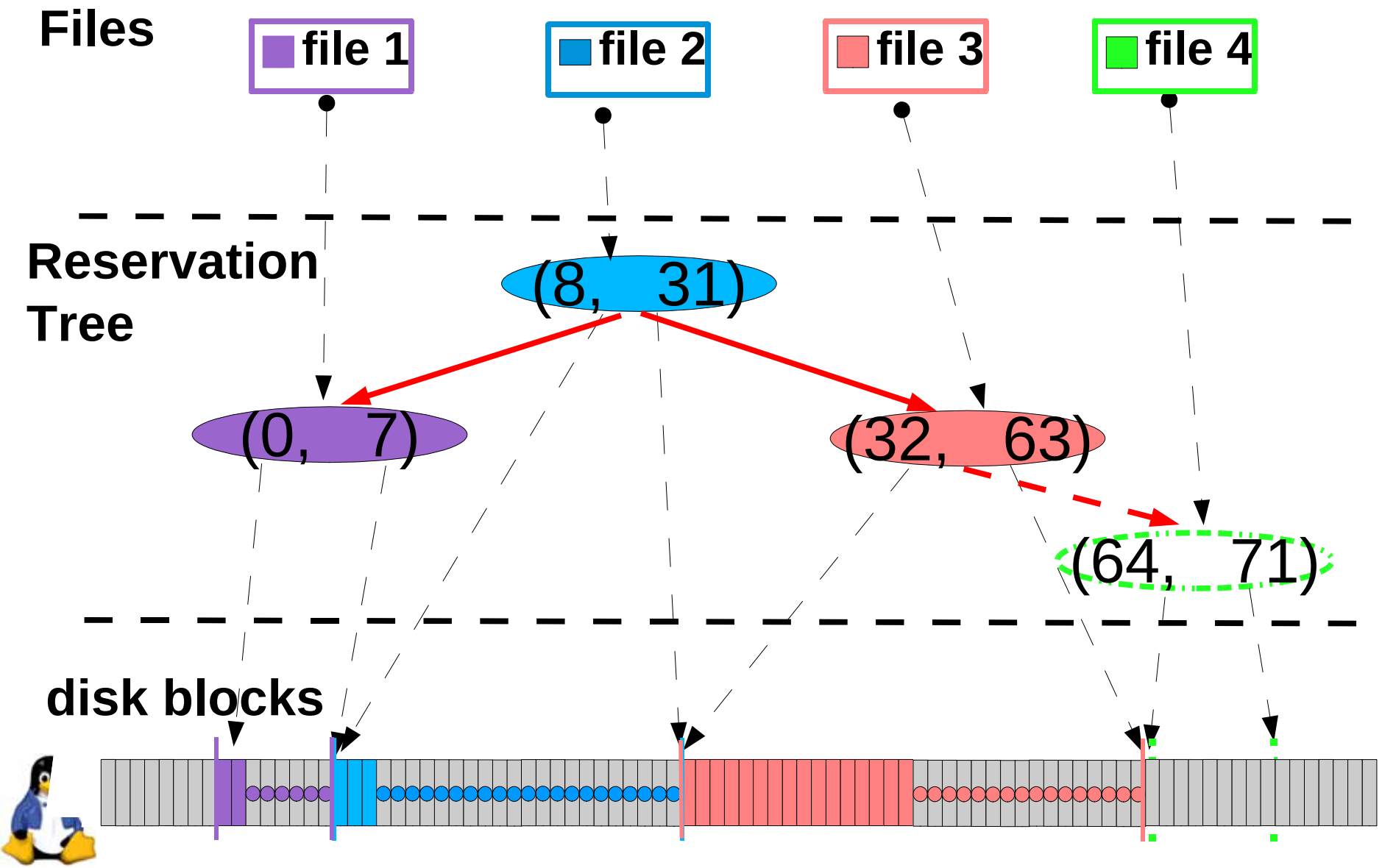
Online Resizing

- Motivation: Taking advantage of new disk space after a logical volume has been grown by the LVM subsystem without needing to unmount the filesystem
- Solution: Reserve space so that the number of blocks needed for the block group descriptors can be grown
  - An additional 4k block is required for every 32 block groups
  - Block group descriptors must be contiguously stored after the superblock
- Integrated into the kernel as of 2.6.10 and e2fsprogs 1.39

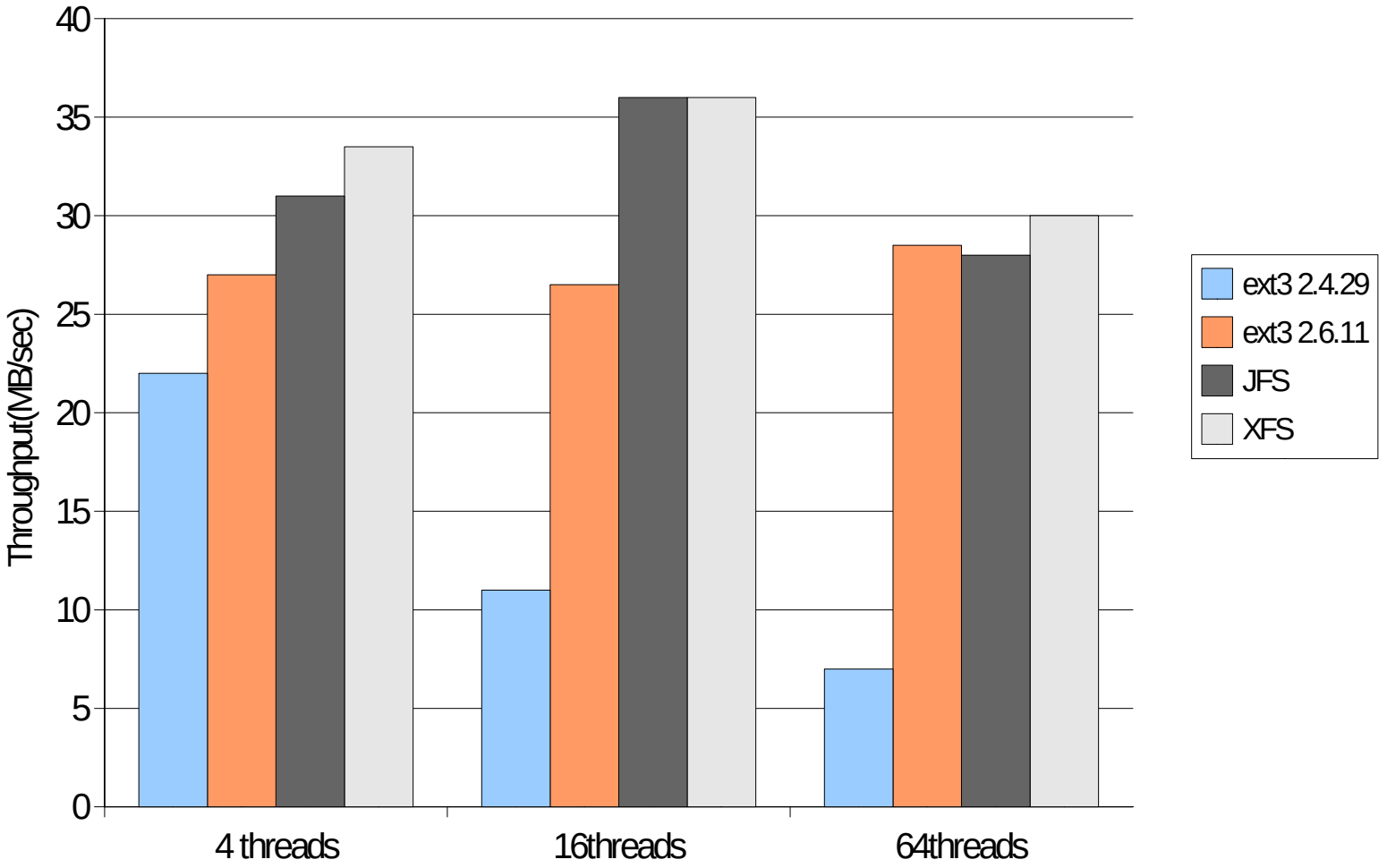
Reservation based block preallocation

- Block preallocation helps reduce file fragmentation caused by concurrent allocation
- Ext3 added block preallocation since 2.6.10 kernel.
- Ext3 uses in-memory block reservation to support a large preallocation





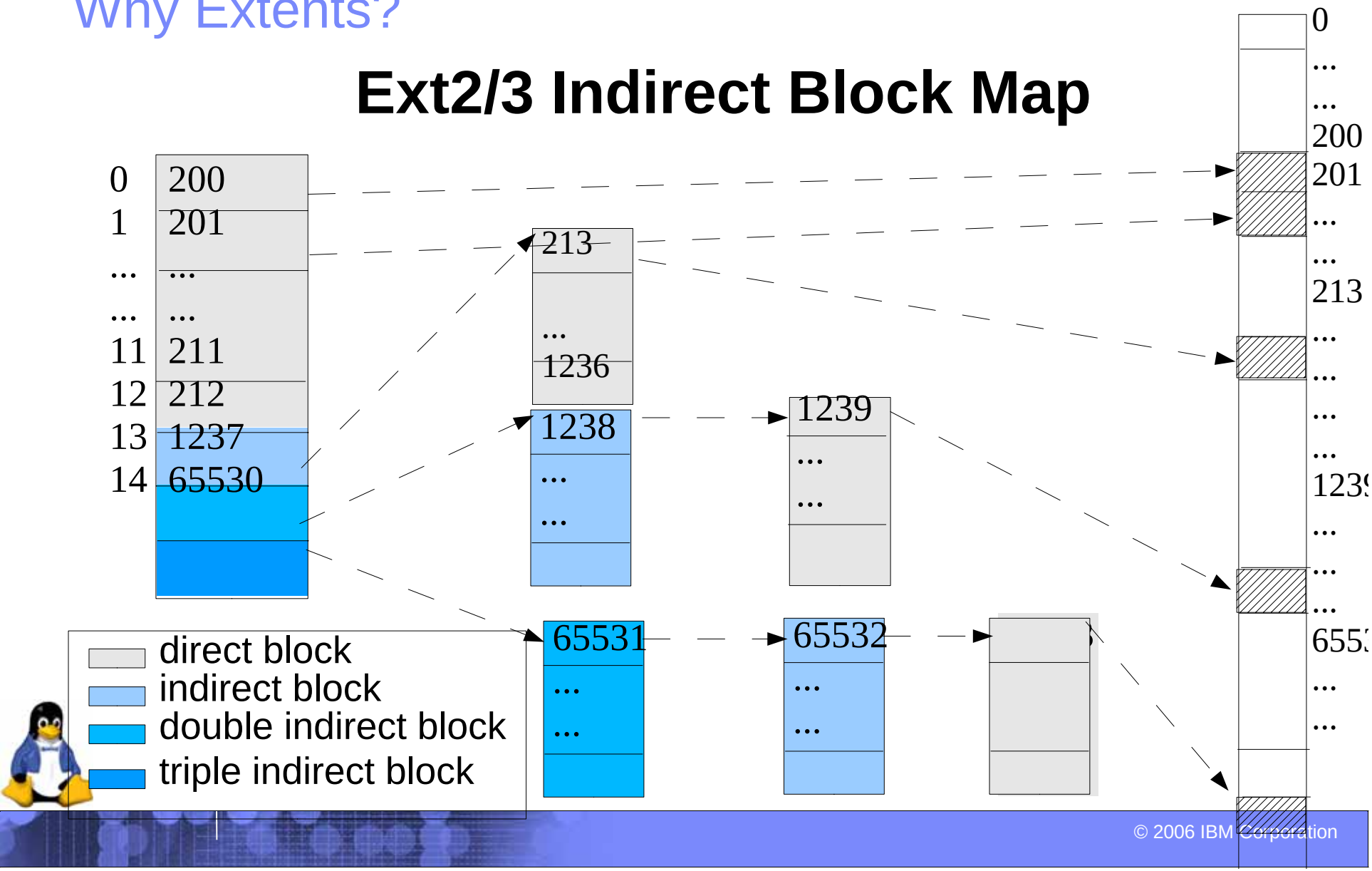
tiobench sequential write



Features planned for ext3/4

- Extents
- Support for large disks (48 and 64 bit block numbers)
- Fine-grained timestamps
- Asynchornous (background) unlink/truncate
- Support > 32,000 subdirectories
- Finer grained locking to support parallel directory operations

Why Extents?

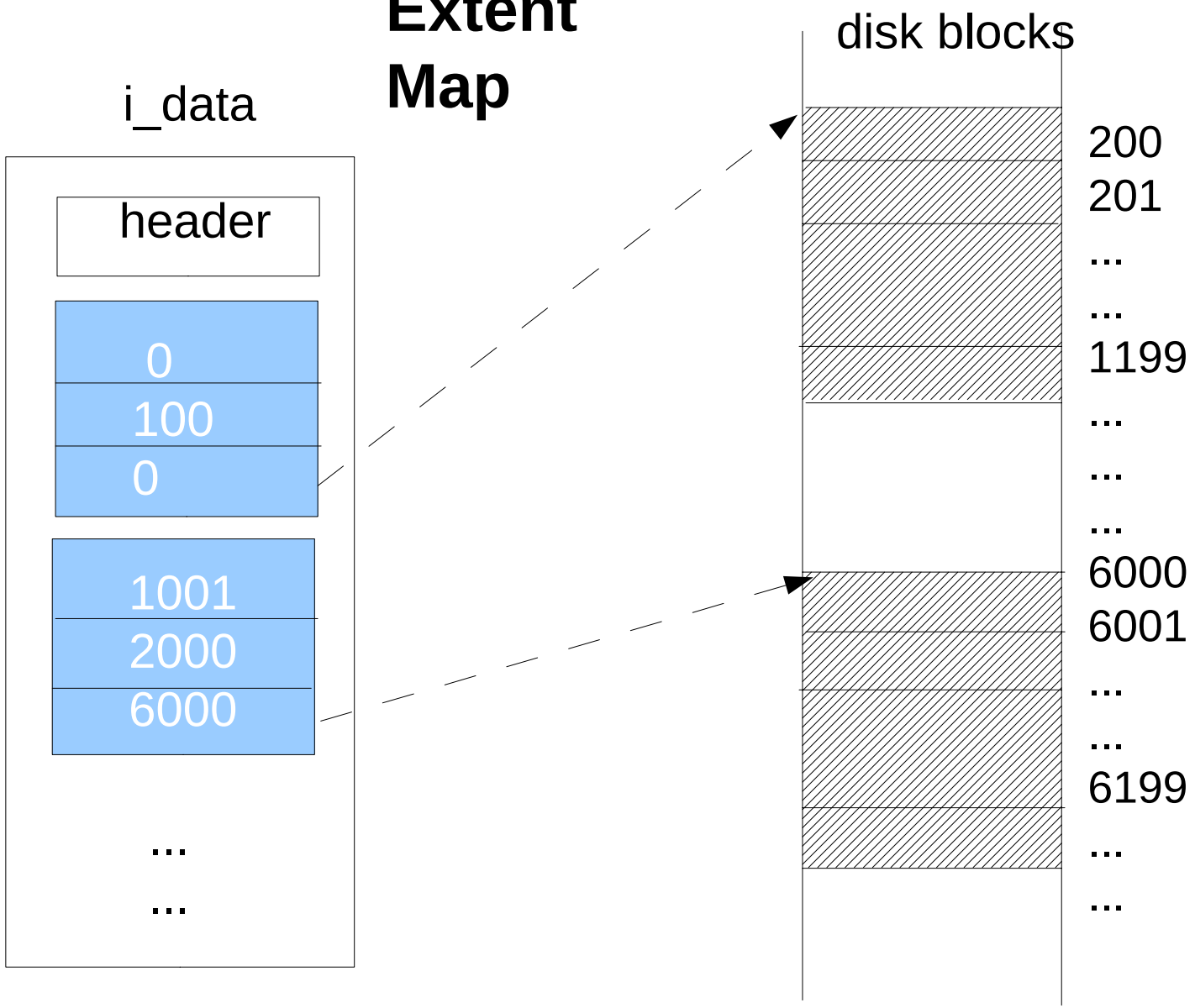


Extents

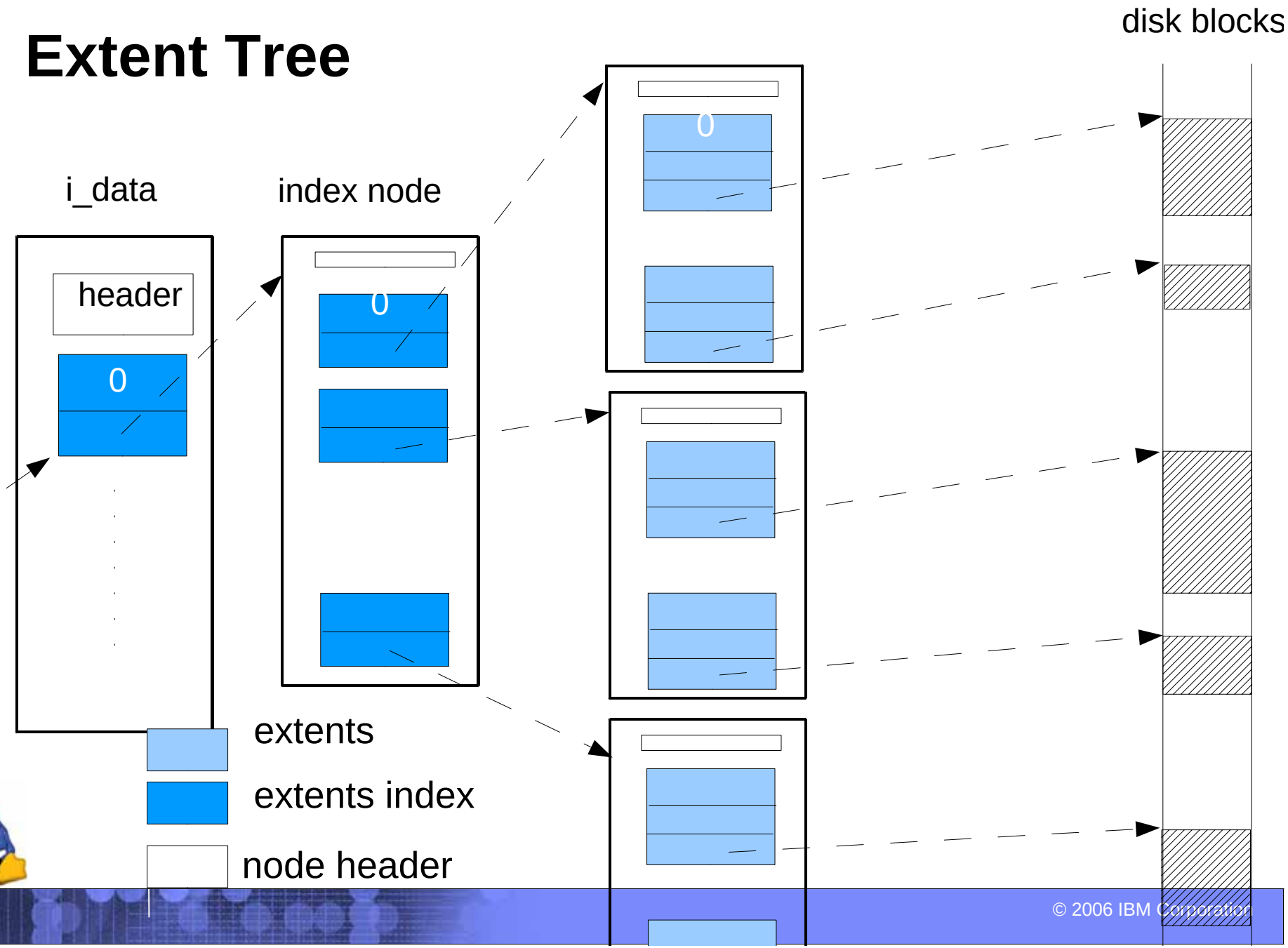
- Extents are an efficient way to represent large files
- An extent is a single descriptor for a range of contiguous blocks

logical	length	physical
0	1000	200

Extent Map



Extent Tree



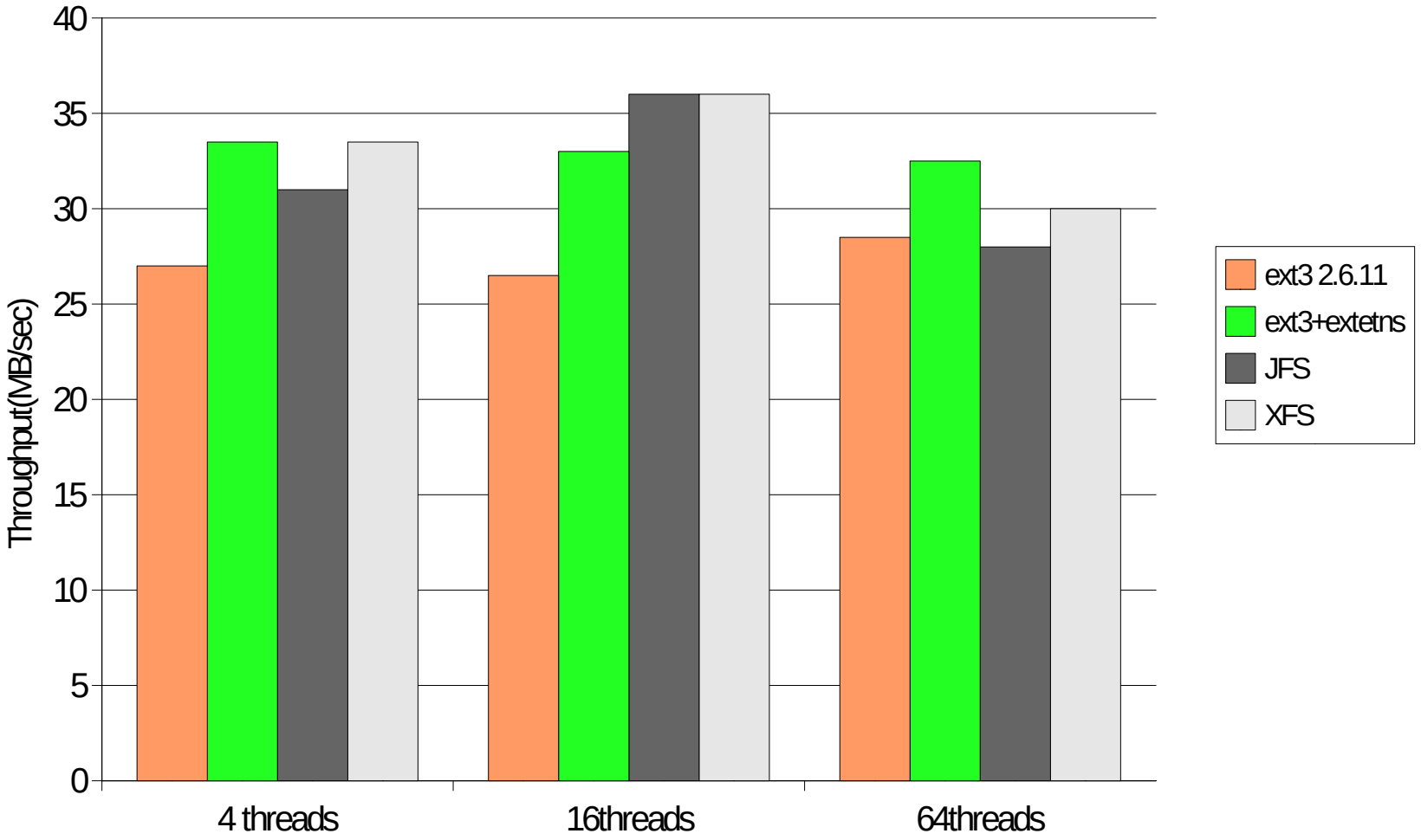
Extent Related Works

- Multiple block allocation
  - An efficient way to allocating a chunk of contiguous blocks at a time
- Delayed allocation
  - Enable multiple block allocation by deferring and clustering single block allocation

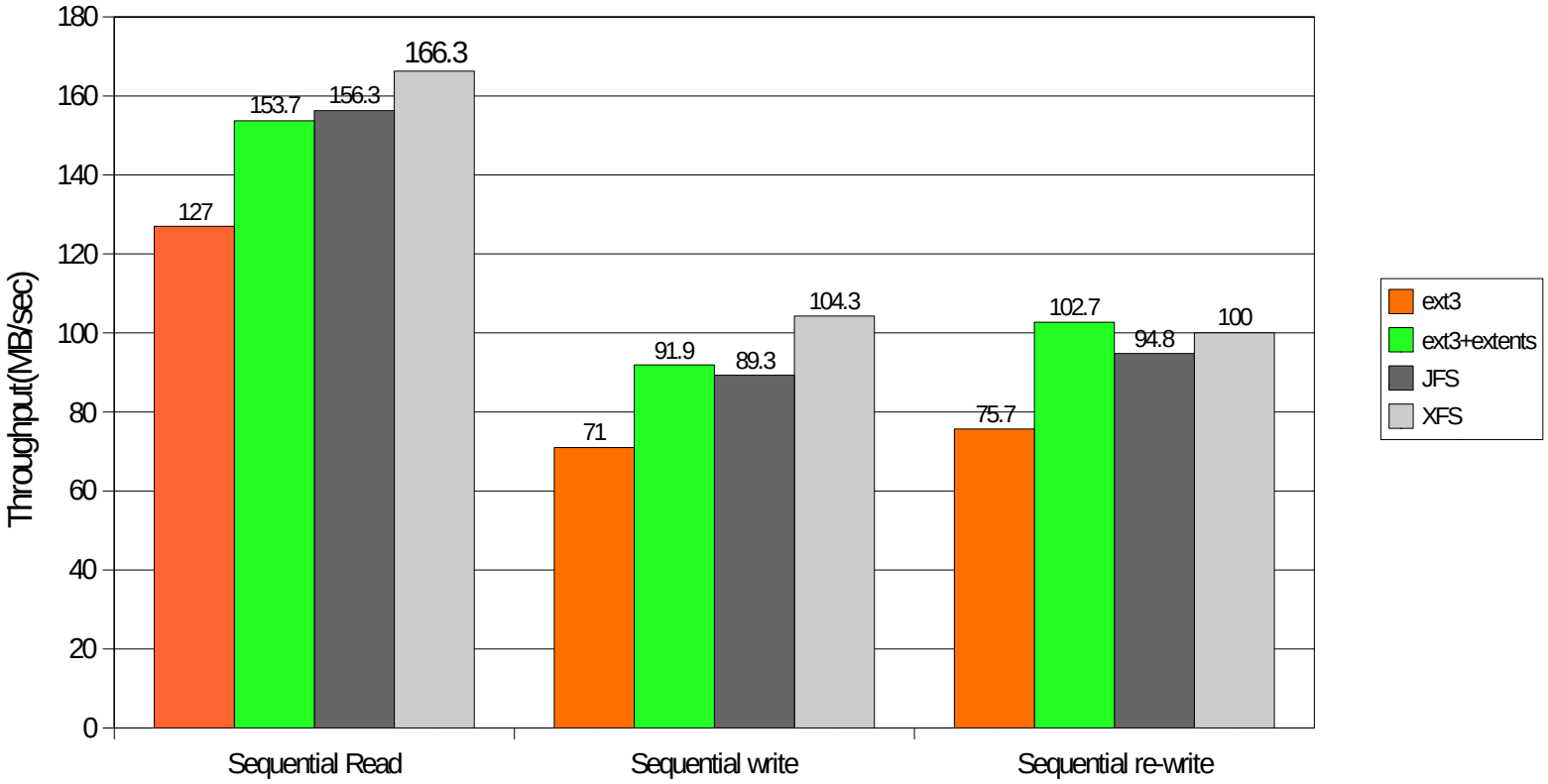
Evaluation of Extents Patches

- Improvements for large file creation/removal/sequential read/sequential rewrite
- Benchmarks used: dbench, tiobench, FFSB filemark, sqlbench, iozone, etc.

Tiobench Sequential Write Comparison With Extents



Large File Sequential I/O Comparison Using FFSB



Ext4: The next-generation ext3

- When initial versions of the extents patches were sent out for comment, some Linux kernel developers expressed concern:
  - Ext3 was too important to risk destabilizing code quality
  - Backwards incompatible extensions could cause user confusion
- After much discussion, a consensus on moving forward
  - Ext3 cleanup patches would be applied
  - The ext3 code base would be forked to fs/ext4, with the filesystem name ext4-dev
  - New work would happen in ext4-dev, and when the feature set for ext4 is stablized it would be renamed from ext4-dev to ext4.

Conclusion

- The ext2/3/4 filesystem is oldest filesystem which is still being actively developed in Linux
- Has served the Linux community well for over 10 years
- With new improvements being constantly being proposed, implemented, and placed into production, ext3/4 development continues to remain vital and exciting!