Ext4: The Next Generation of Ext2/3 Filesystem

Mingming Cao
Suparna Bhattacharya
Ted Tso
IBM

© 2007 IBM Corporation

Agenda

- Motivation for ext4
- Why fork ext4?
- What's new in ext4?
- Planned ext4 features

© 2006 IBM Corporation

Motivation for ext4

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

© 2006 IBM Corporation

Why fork ext4

- Many features require on-disk format changes
- Keep large ext3 user community unaffected
- 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

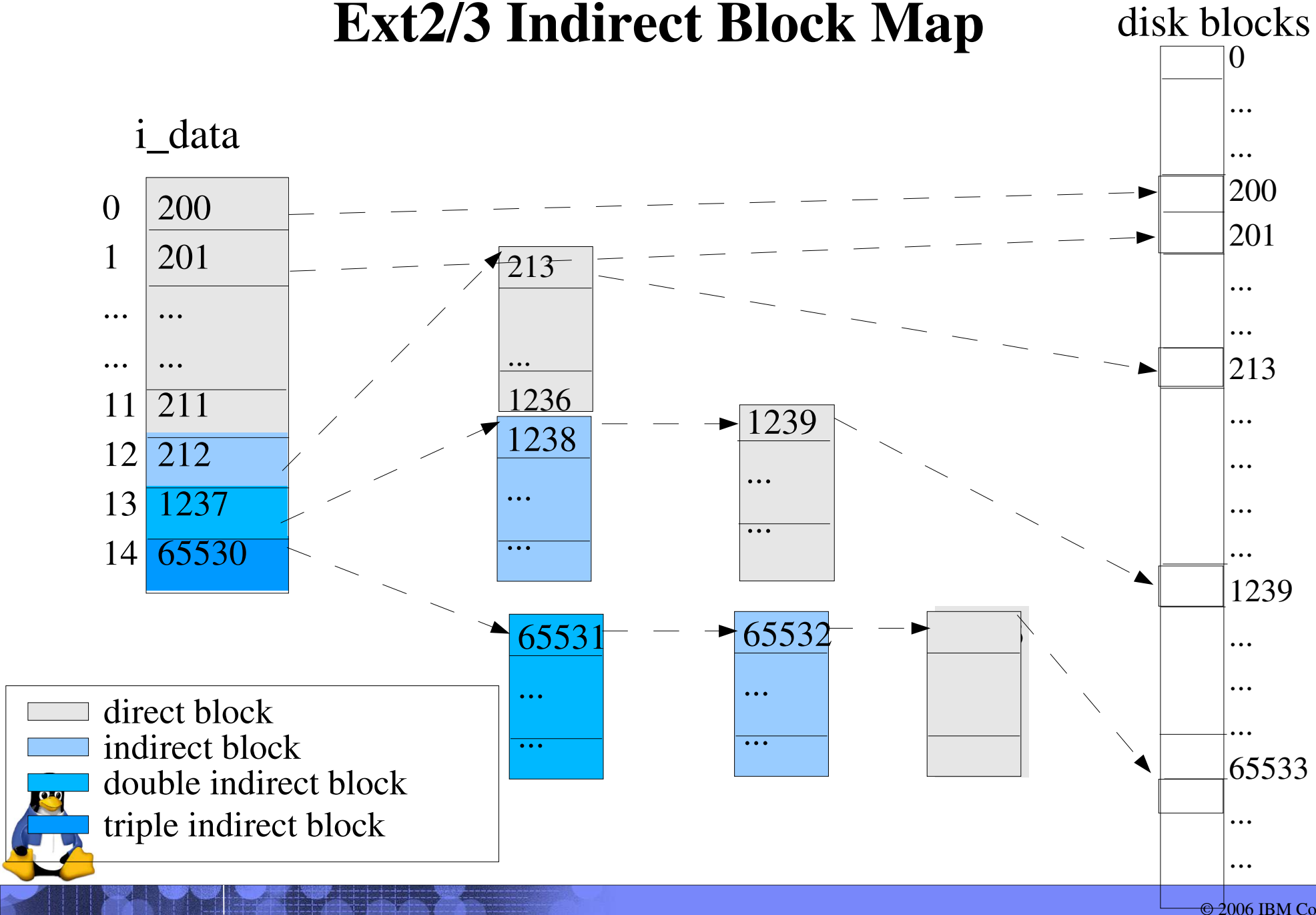
© 2006 IBM Corporation

What's new in ext4

- Ext4 was cloned and included in 2.6.19
- Replacing indirect blocks with extents
- Ability to address >16TB filesystems (48 bit block numbers)
- Use new forked 64-bit JBD2

© 2006 IBM Corporation

Ext2/3 Indirect Block Map



© 2006 IBM Corporation

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
- An extent is a single descriptor for a range of contiguous blocks
  - a efficient way to represent large file
  - Better CPU utilization, fewer metadata IOs

logical	length	physical
0	1000	200

© 2006 IBM Corporation

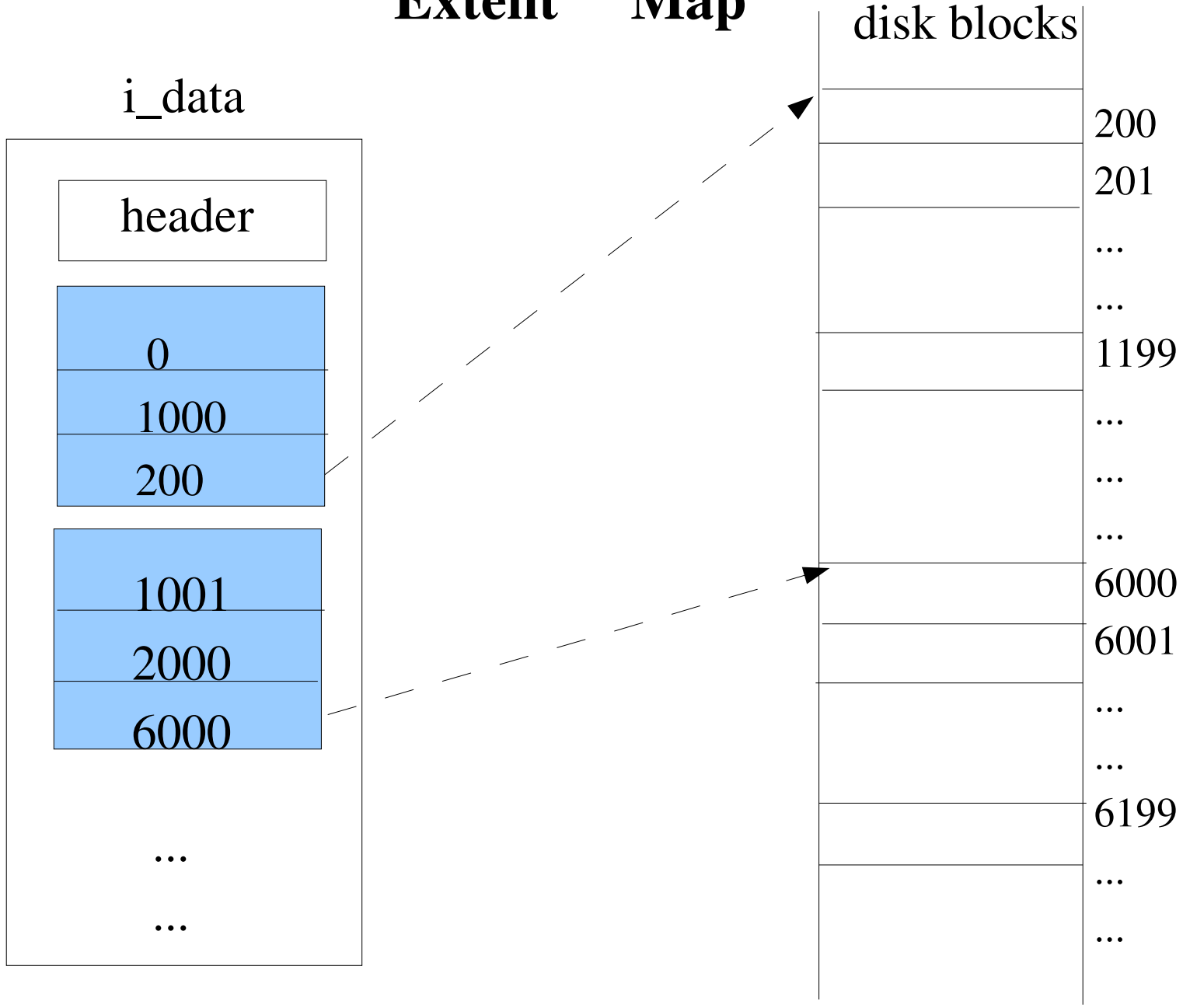
On-disk extents format

- 12 bytes ext4_extent structure
  - address 1EB filesystem (48 bit physical block number)
  - max extent 128MB (15 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 */
};
```

© 2006 IBM Corporation

Extent Map



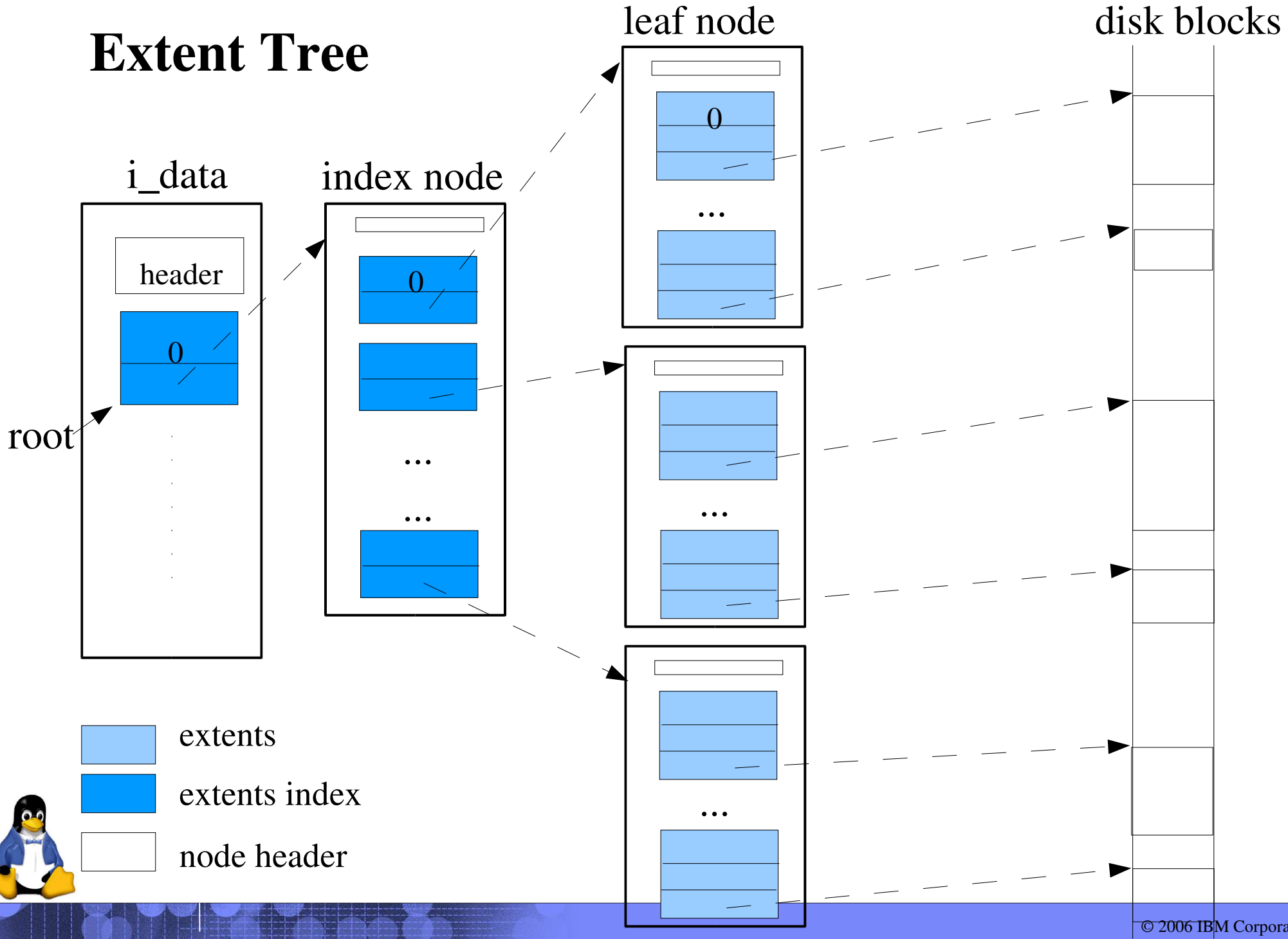
© 2006 IBM Corporation

Extents tree

- Up to 3 extents could stored in inode i_data body directly
- Use a inode flag to mark extents file vs ext3 indirect block file
- Convert to a B-Tree extents tree, for > 3 extents
- Last found extent is cached in-memory extents tree

© 2006 IBM Corporation

Extent Tree



© 2006 IBM Corporation

48-bit block numbers

- Part of the extents changes
  - 32bit ee_start and 16 bit ee_start_hi in ext4 extent struct
- Why not 64-bit
  - 48-bit is enough for a 2**60 (or 1EB) filesystem
  - Original lustre extent patches provide 48-bit block numbers
  - More packed meta data, less disk IO
  - Extent generation flag allow adapt to 64-bit block number easily

© 2006 IBM Corporation

64-bit meta data changes

- In kernel block variables to address >32 bit block number
- Super block fields: 32 bit -> 64 bit
- Larger block group descriptors (required doubling their size)
- extended attributes block number (32 bit -> 48 bit)

© 2006 IBM Corporation

64-bit JBD2

- Forked from JBD to handle 64-bit block numbers
- Could be used for 32bit journaling support as well
- Added JBD2_FEATURE_INCOMPAT_64BIT

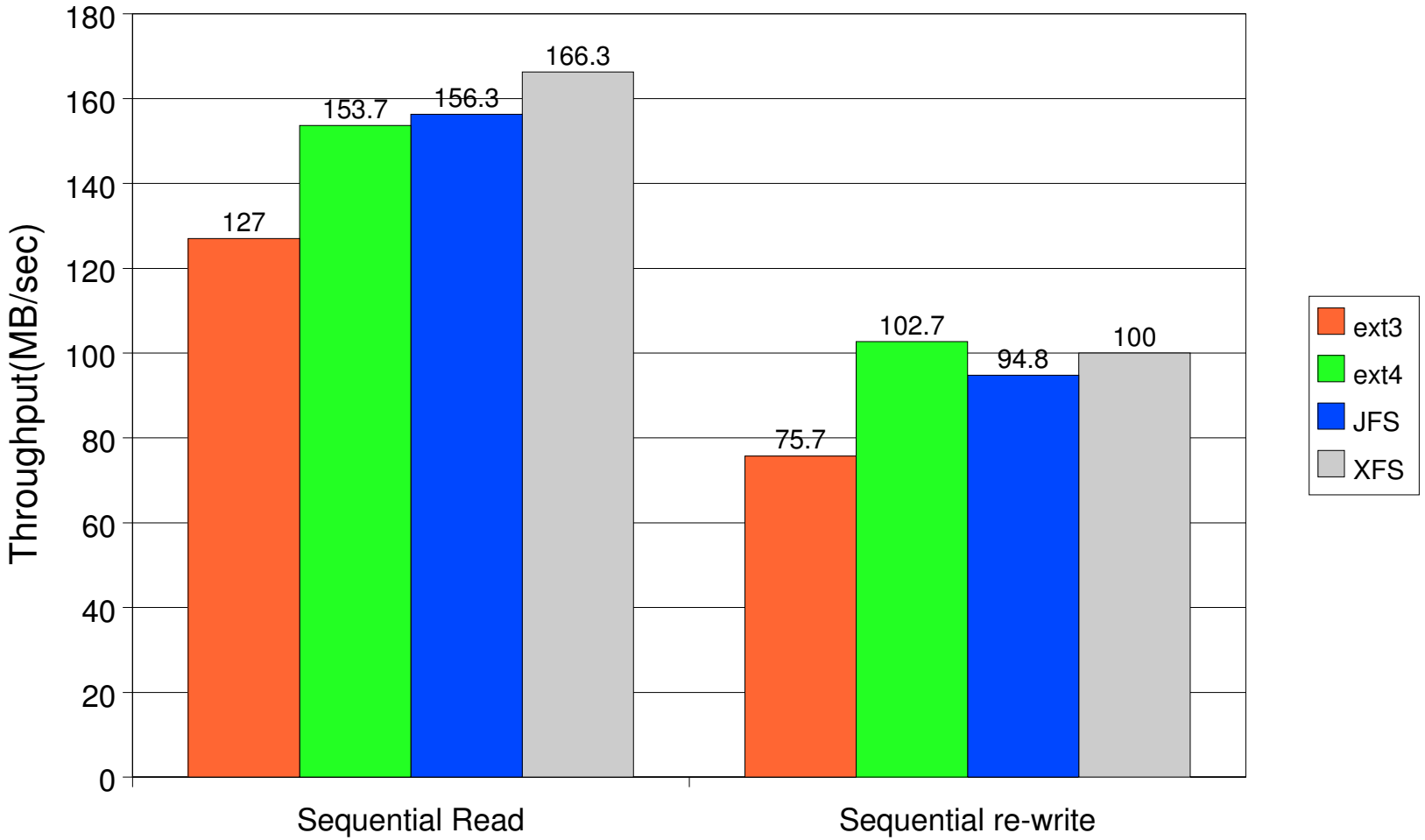
© 2006 IBM Corporation

Testing ext4

- Mount it as ext4dev
  - mount -t ext4dev
- Enabling extents
  - mount -t ext4dev -o extents
  - compatible with the ext3 filesystem until you add a new file
- ext4 vs ext3 performance
  - improve large file read/rewrite/unlink

© 2006 IBM Corporation

Large File Sequential Read &Rewrite Using FFSB



© 2006 IBM Corporation

New defaults for ext4

- Features available in ext3, enable by default in ext4
- directory indexing
- resize inode
- large inode (256bytes)

© 2006 IBM Corporation

Planned new features for ext4

- Work-in-progress: patches available
  - More efficient multiple block allocation
  - Delayed block allocation
  - Persistent file allocation
  - Online defragmentation
  - Nanosecond timestamps

© 2006 IBM Corporation

Others planned features

- Allow greater than 32k subdirectories
- Metadata checksumming
- Uninitialized groups to speed up mkfs/fsck
- Larger file (16TB)
- Extending Extended Attributes limit
- Caching directory contents in memory

© 2006 IBM Corporation

And maybe scales better?

- 64 bit inode number
  - challenge: user space might in trouble using 32bit stat()
- Dynamic inode table
- More scalable free inode/free block scheme
- fsck scalability issue
- Larger block size

© 2006 IBM Corporation

Multiple block allocation

- Multiple block allocation
  - Allocate contiguous blocks together
    - Reduce fragmentation, extent meta-data and cpu usage
    - Stripe aligned allocations
- Buddy free extent bitmap generated from on-disk bitmap
- Status
  - Patch available

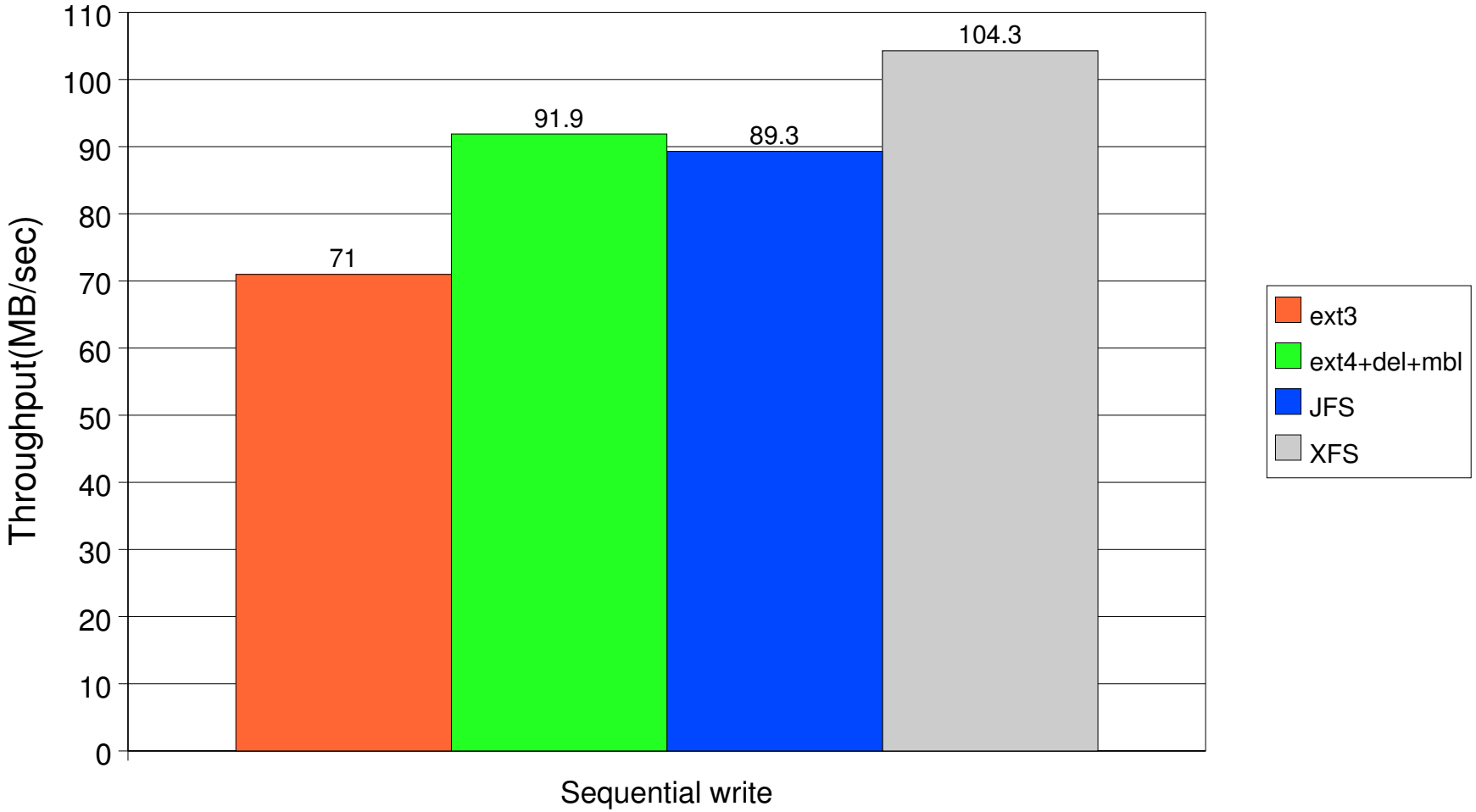
© 2006 IBM Corporation

Delayed block allocation

- Defer block allocation to write back time
  - Improve chances allocating contiguous blocks, reducing fragmentation
- Blocks are reserved to avoid ENOSPC at writeback time:
  - At prepare_write() time, use page_private to flag page need block reservation later.
  - At commit_write() time, reserve block. Use PG_booked page flag to mark disk space is reserved for this page
- Trickier to implement in ordered mode

© 2006 IBM Corporation

Large File Sequential Write Using FFSB



© 2006 IBM Corporation

Persistent file preallocation

- Allow preallocating blocks for a file without having to initialize them
  - Contiguous allocation to reduce fragmentation
  - 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
- API for preallocation
  - Current implementation uses ioctl
    - EXT4_IOC_FALLOCATE cmd, the offset and bytes to preallocate

© 2006 IBM Corporation

Online defragmentation

- Defragmentation is done in kernel, based on extent
- Allocate more contiguous blocks in a temporary inode
- Read a data block form the original inode, move the corresponding block number from the temporary inode to the original inode, and write out the page
- Join the ext4 online defragmentation talk for more detail

© 2006 IBM Corporation

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

© 2006 IBM Corporation

High resolution timestamps

- Address NFSv4 needs for more fine granularity time stamps
- Proposed solution used 30 bits out of the 32 bits field in larger inode (>128 bytes) for nanoseconds
- Performance concern: result in additional dirtying and writeout updates
  - might batched by journal

© 2006 IBM Corporation

Unlimited number of subdirectories

- Each subdirectory has a hard link to its parent
- Number of subdirectories under a single directory is limited by type of inode's link count(16 bit)
- Proposed solution to overcome this limit:
  - Not counting the subdirectory limit after counter overflow, storing link count of 1 instead.

© 2006 IBM Corporation

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

© 2006 IBM Corporation

Uninitialized block groups

- Add flags field to indicate whether or not the inode and bitmap allocation bitmaps are valid
- Add field to indicate how much of the inode table has been initialized
- Useful to create a large filesystem and fsck a not-very-full large filesystem

© 2006 IBM Corporation

Extend EA limit

- Allow EA data larger than a single filesystem block
- The last entry in EA block is reserved to point to a small number of extra EA data blocks, or to an indirect block

© 2006 IBM Corporation

ext3 vs ext4 summary

	ext3	ext4dev
filesystem limit	16TB	1EB
file limit	2TB	16TB
limit	2**32	2**32
default inode size	128 bytes	256 bytes
block mapping	indirect block map	extents
time stamp	second	nanosecond
sub dir limit	2**16	unlimited
EA limit	4K	>4K
preallocation	in-core reservation	for extent file
deframentation	No	yes
directory indexing	disabled	enabled
delayed allocation	No	yes
multiple block allocation	basic	advanced

© 2006 IBM Corporation

Getting involved

- Mailing list: linux-ext4@vger.kernel.org
- latest ext4 patch series
  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
- Weekly conference call; minutes on the wiki
  - Contact us if you'd like dial in
- IRC channel: irc.oftc.net, /join #linuxfs

© 2006 IBM Corporation

The Ext4 Development Team

- Alex Thomas
- Andreas Dilger
- Theodore Tso
- Stephen Tweedie
- Mingming Cao
- Suparna Bhattacharya
- Dave Kleikamp
- Badari Pulavarathy
- Avantikia Mathur
- Andrew Morton
- Laurent Vivier
- Alexandre Ratchov
- Eric Sandeen
- Takashi Sato
- Amit Arora
- Jean-Noel Cordenner
- Valerie Clement

© 2006 IBM Corporation

Conclusion

- Ext4 work just beginning
- Extents merged, other patches on deck

© 2006 IBM Corporation

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of IBM.

IBM and the IBM logo are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.

Lustre is a trademark of Cluster File Systems, Inc.

Unix is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

This document is provied ``AS IS," with no express or implied warranties. Use the information in this document at your own risk.

© 2006 IBM Corporation