

























## 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

## 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)

## 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

## 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













## 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

## Delayed block allocation

Defer block allocation to write back time

Improve chances allocating contiguous blocks, reducing fragmentation

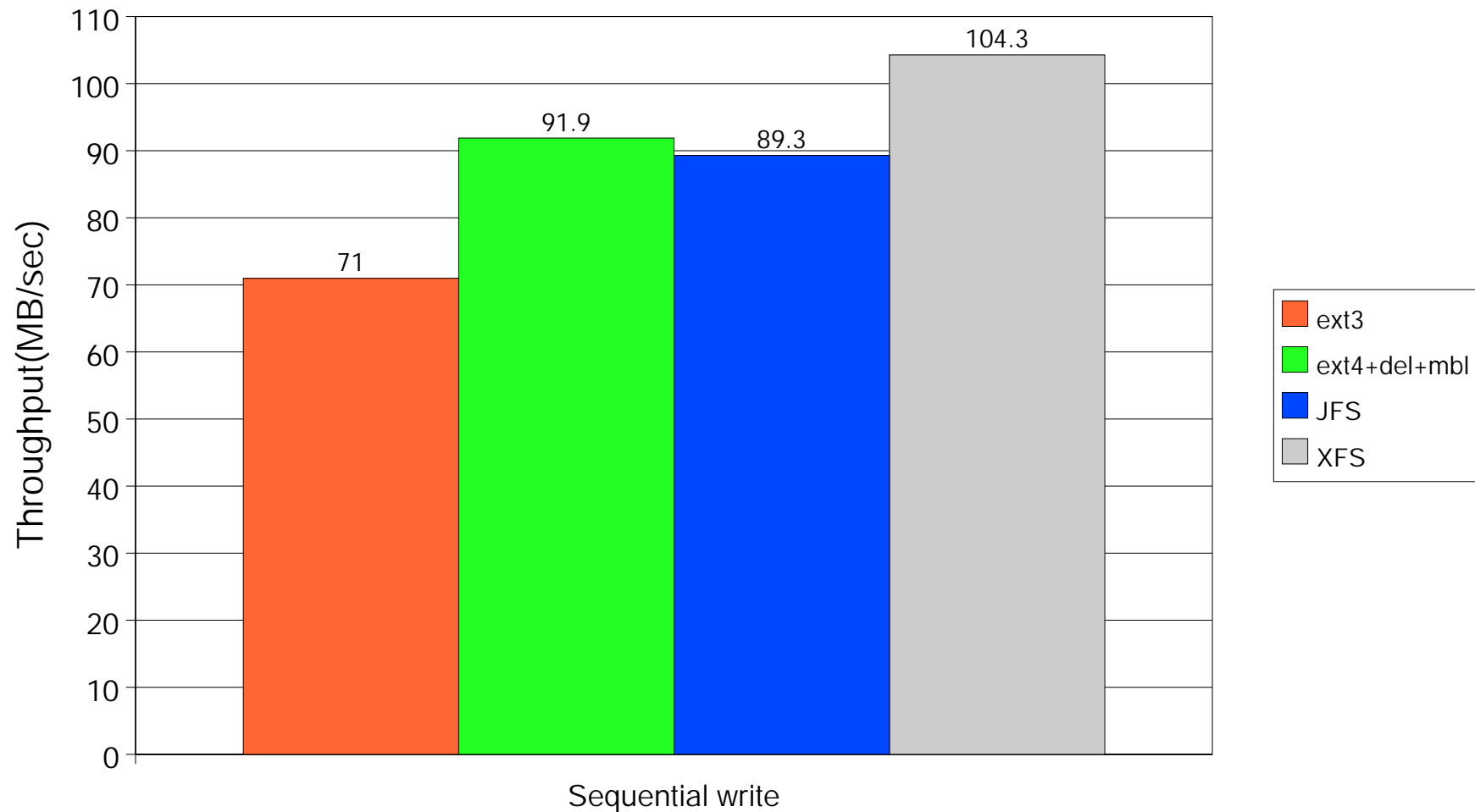
Blocks are reserved to avoid ENOSPC at writeback time:

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

## Large File Sequential Write Using FFSB



## 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



## Online defragmentation

Defragmentation is done in kernel, based on extent

Allocate more contiguous blocks in a temporary inode

Read a data block from 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

## 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

## 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

## 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.

## Metadata checksumming

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

## 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

## 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

## 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
defragmentation	No	yes
directory indexing	disabled	enabled
delayed allocation	No	yes
multiple block allocation	basic	advanced



## Getting involved

Mailing list: [linux-ext4@vger.kernel.org](mailto: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](irc://irc.oftc.net), /join #linuxfs





