



ORACLE[®]

XFS - The High Performance Enterprise File System

<jeff.liu@oracle.com> Jeff Liu



Agenda

- ① About XFS
- ② Design
- ③ Journal
- ④ Self-describing metadata
- ⑤ Performance
- ⑥ What's new && in progress

General information

- ▶ Created by SGI in 1993, ported to Linux since 2001
- ▶ Full 64-bit journaling file system, the best choice for >16TB storage
- ▶ Extent based and delayed allocation
- ▶ Maximum file system size/file size: 16 EiB/8EiB
- ▶ Variable blocks sizes: 512 bytes to 64 KB
- ▶ Freeze/Thaw to support volume level snapshot - `xfs_freeze`
- ▶ Online file system/file defragmentation - `xfs_fsr`
- ▶ Online file system resize – `xfs_growfs`
- ▶ User/Group/Project disk quota support
- ▶ Dumping and Restoring – `xfsdump/xfsrestore`

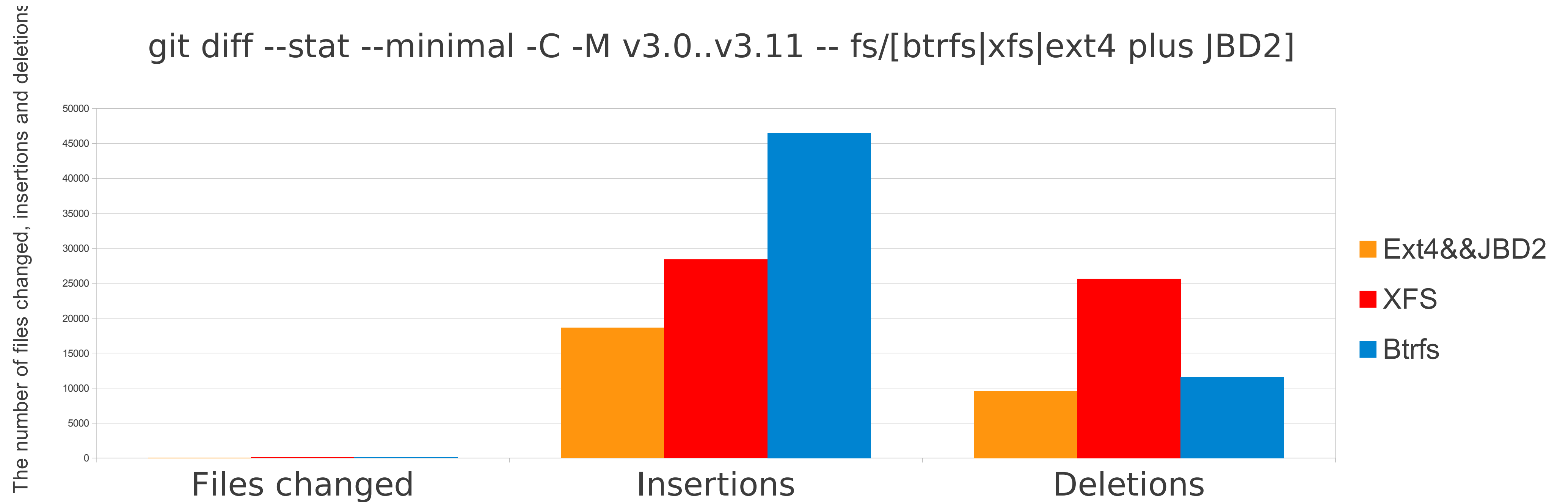
Linux distributions support

- ▶ Available on more than 15 Linux distributions
- ▶ SuSE has include XFS support since SLES8
- ▶ Will be the default file system on RHEL7
- ▶ Premier support on Oracle Linux 6
- ▶ Ceph OSD daemons recommend XFS as the underlying file system for production deployment

Community Efforts

The statistics of code changes between Linux 3.0~3.11

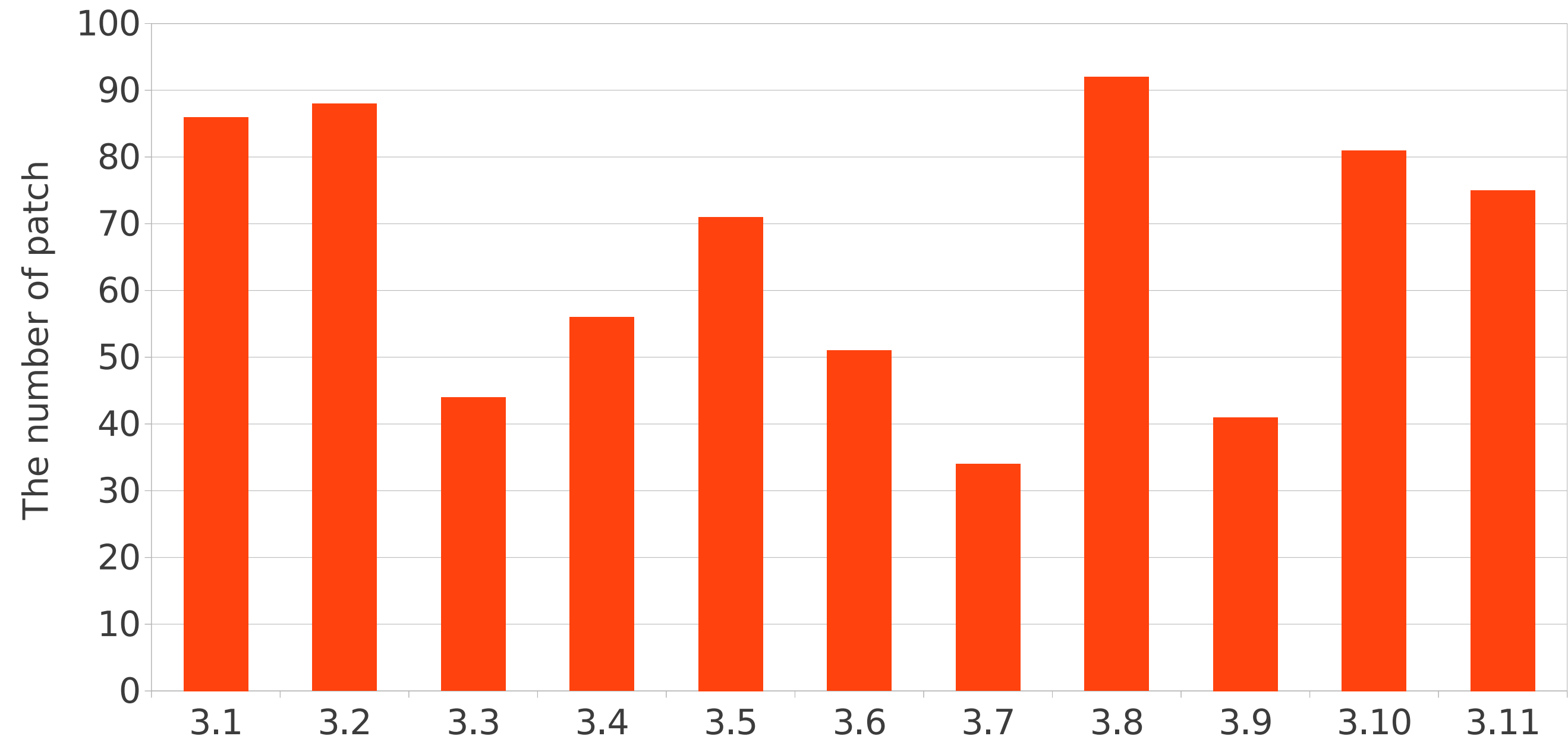
```
git diff --stat --minimal -C -M v3.0..v3.11 -- fs/[btrfs|xfs|ext4 plus JBD2]
```



Community Efforts

- Totally 719 patches
- Mainly on infrastructure, performance improvements as well as bug fixes

XFS Mainline Progress Between Linux 3.0~3.11



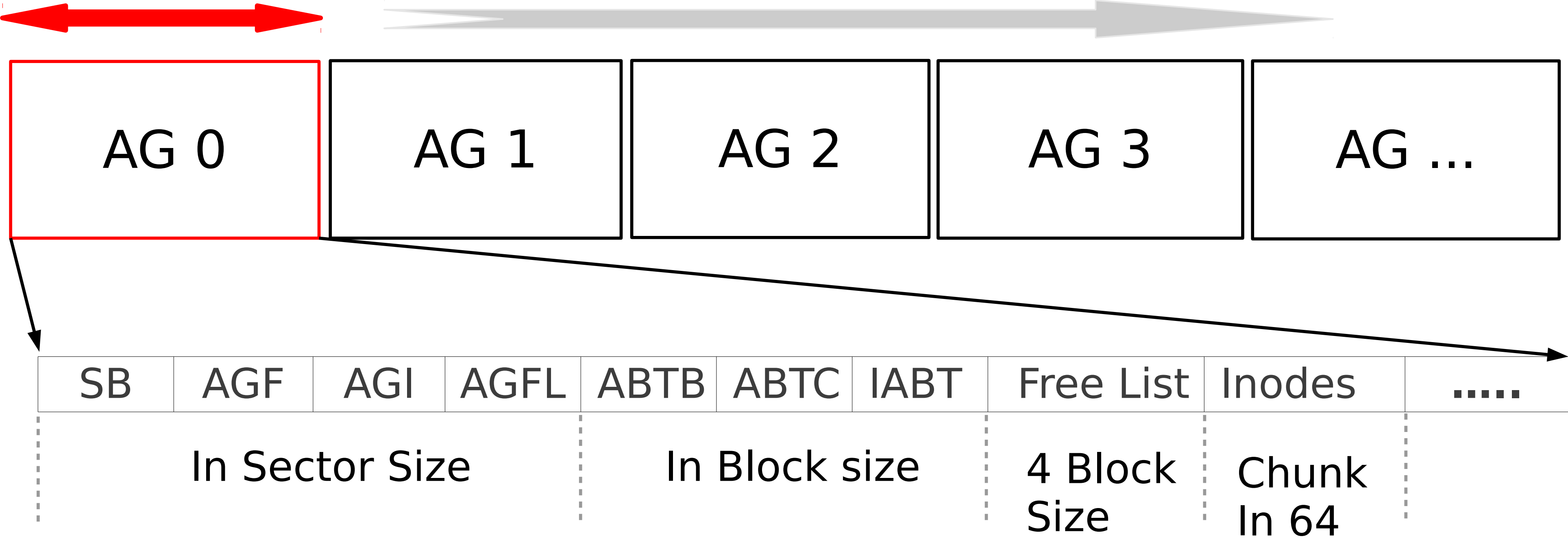
Design – Allocation group

- ▶ Allocation group
 - Can be thought as an individual file system
 - Primary AG and secondary AGs
- ▶ Each AG contains
 - Super block
 - Free space management
 - Inode allocation and tracking
- ▶ Benefits
 - Allows XFS to handle most operations in parallel without degrading performance

Design - AG layout

Primary AG

Other AGs



Design – Generic AG B+Tree Structures

► Generic btree header

```
struct xfs_btree_block {
    __be32      bb_magic;      /* magic number for block type */
    __be16      bb_level;     /* 0 is a leaf */
    __be16      bb_numrecs;   /* current # of data records */
    __be32/64   bb_leftsib;
    __be32/64   bb_rightsib;
    ...
}
```

► Data record/key structure

```
typedef struct xfs_alloc_rec {
    __be32      ar_startblock; /* starting block number */
    __be32      ar_blockcount; /* count of free blocks */
} xfs_alloc_rec_t, xfs_alloc_key_t;
```

► Node pointers

```
typedef __be32  xfs_alloc_ptr_t;
```

Design – AG Free Space Management

- ▶ Tracks free space in an AG using two B+tree
 - One B+tree tracks space by block number
 - Another by the count of the free space block
 - To quickly find free space near a given block or a give size
- ▶ All block numbers, indexes and counts are AG relative
- ▶ AG Free Space Block – 'AGF' locates in the 2rd sector of an AG

Design – AGF Free Space B+trees (1 order)

AGF
agf_roots[2]
agf_levels[2] = 1

[0] **xfs_btree_block**

bb_magic = XFS_ABTB_MAGIC
bb_level = 0 (leaf)
bb_numrecs = rec size
bb_leftsib/rightsib = NULL

recs[bb_numrecs] **xfs_alloc_rec_t**

startblock	startblock	blockcount
	startblock	blockcount

[1] **xfs_btree_block**

bb_magic = XFS_ABTC_MAGIC
bb_level = 0 (leaf)
bb_numrecs = rec size
bb_leftsib/rightsib = NULL

recs[bb_numrecs] **xfs_alloc_rec_t**

blockcount	startblock	blockcount
	startblock	blockcount

Design – AG Inode Information

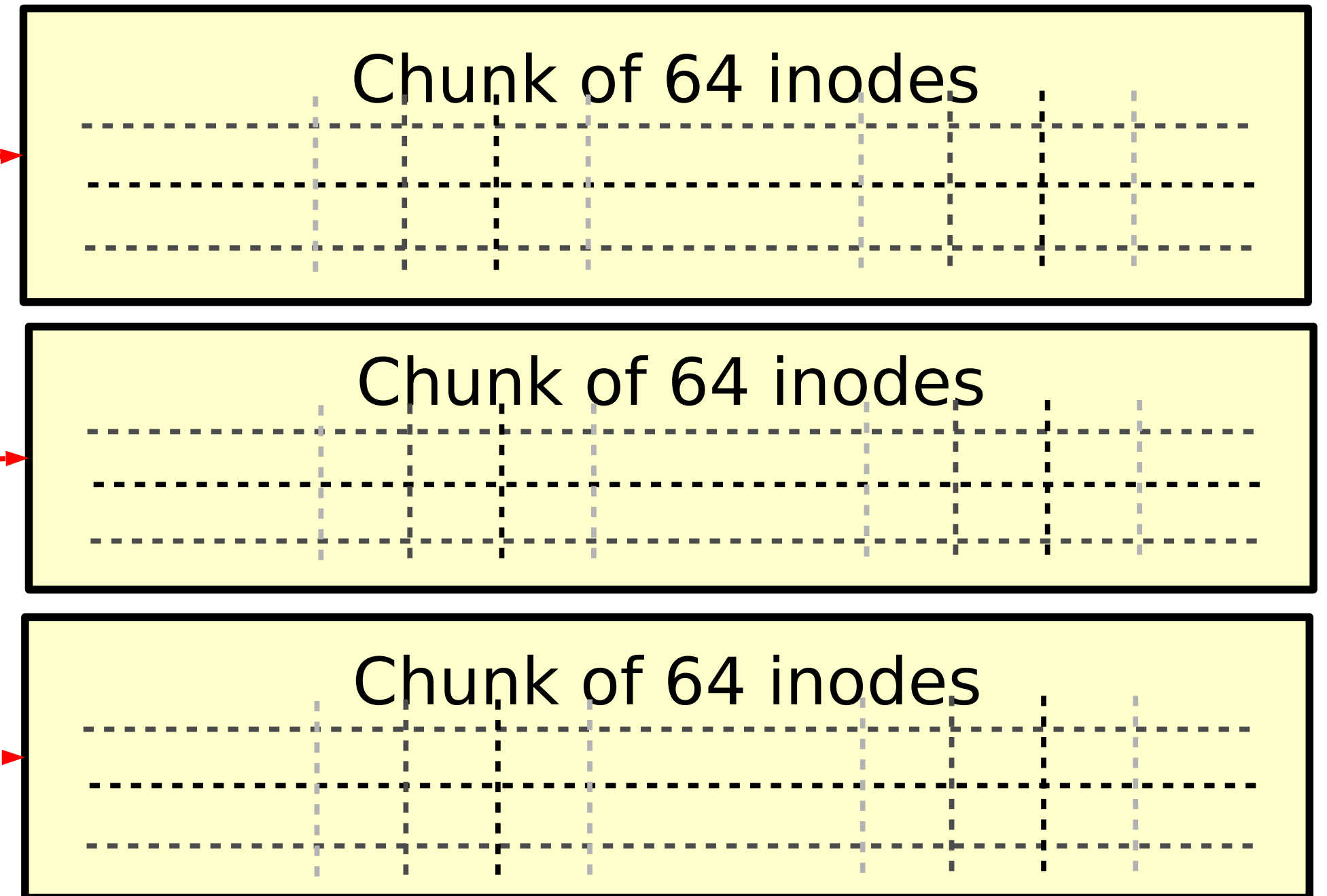
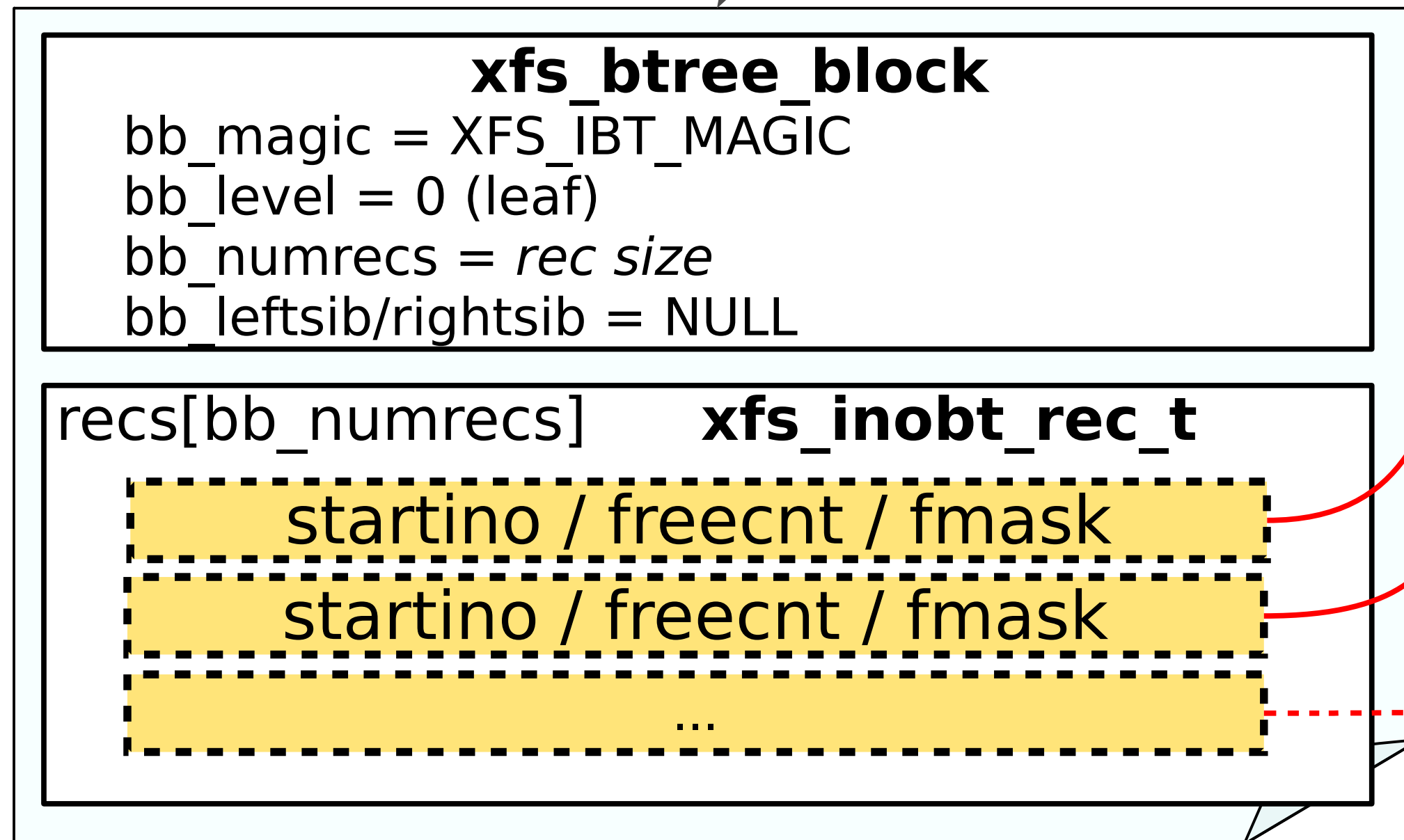
- ▶ Locates at 3rd sector of AG, known as “AGI”
- ▶ Inodes are allocated in chunk of 64, and a B+tree is used to track these chunks of inodes as they are allocated and freed
- ▶ B+tree header – same as AGF header
- ▶ B+tree leaves
- ▶ Node's key/pointer pairs

```
typedef struct xfs_inobt_rec {  
    __be32      ir_startino;    /* starting inode number */  
    __be32      ir_freecount;   /* count of free inodes (set bits) */  
    __be64      ir_free;       /* free inode mask */  
} xfs_inobt_rec_t;
```

```
typedef struct xfs_inobt_key {  
    __be32      ir_startino;    /* starting inode number */  
} xfs_inobt_key_t;  
typedef __be32 xfs_inobt_ptr_t;
```

Design – AGI B+tree (1 order)

AGI
agf_root
agf_level = 1



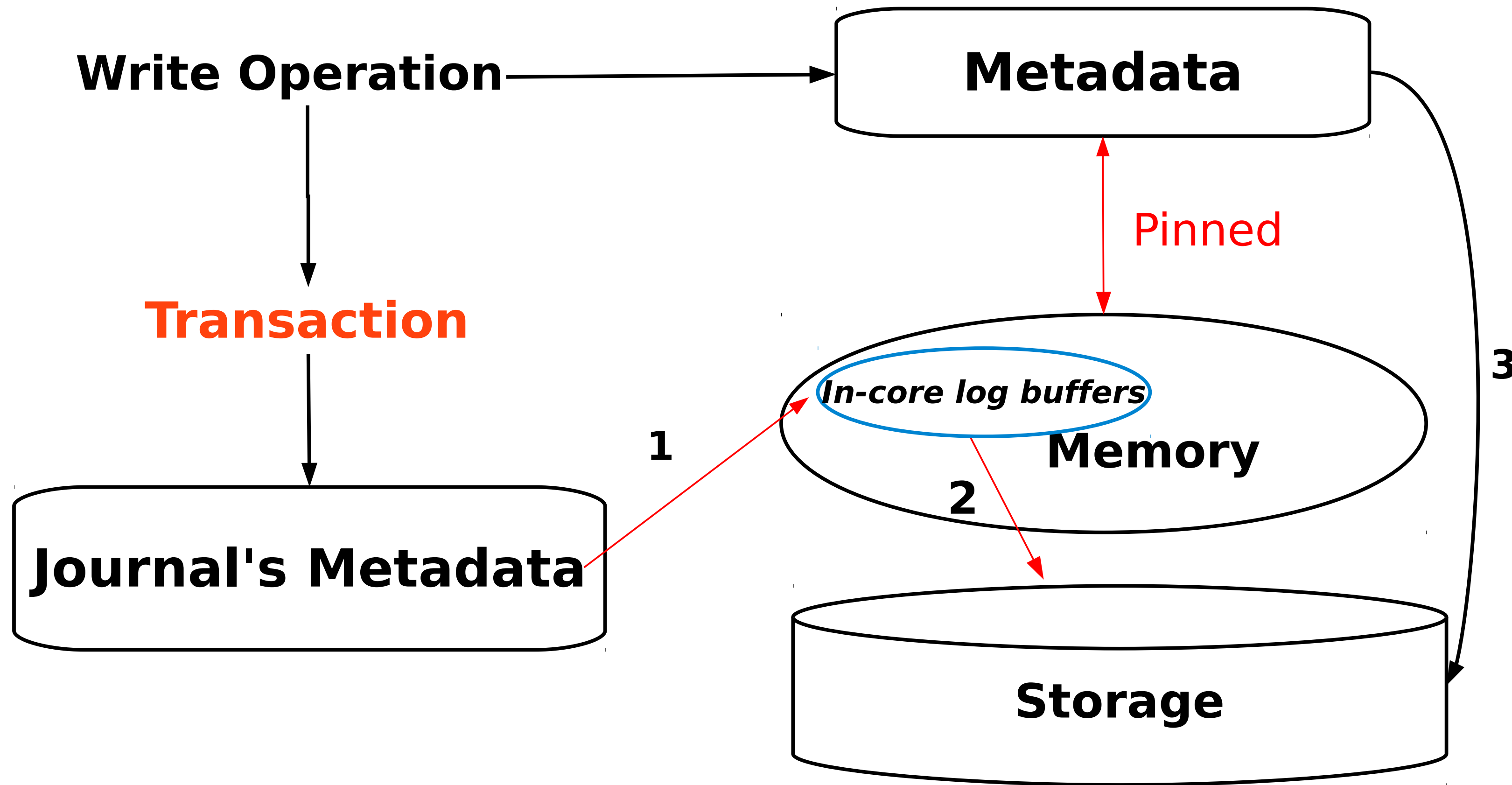
Journal

- ▶ Guarantee file system consistency in the event of power failure or system crash
- ▶ Need not conventional UNIX fsck
- ▶ Logical/physical mixed journaling
 - Log items which are logged in logical format contains changes to the in-core structures rather than the on-disk structures
 - Typical buffer are logged in physical format
- ▶ Automatically perform log recovery at mount time
- ▶ Quick crash recovery
 - Independent to the size of the storage
 - Dependent to the activities of the file system at the mount of disasters

Journal

- ▶ Internal log blocks start near the middle of the disk
- ▶ External log volume support
- ▶ Maximum log size just under 2GB (2038 MB)
- ▶ File system update are written into journal before the actual disk blocks are updated
- ▶ Incore log and on-disk log
 - Journals metadata are writing to in-core log buffers by first
 - The in-core log buffers are flushed to on-disk log asynchronously
 - Metadata are pinned in memory a transaction is committed to the on-disk log

Journal



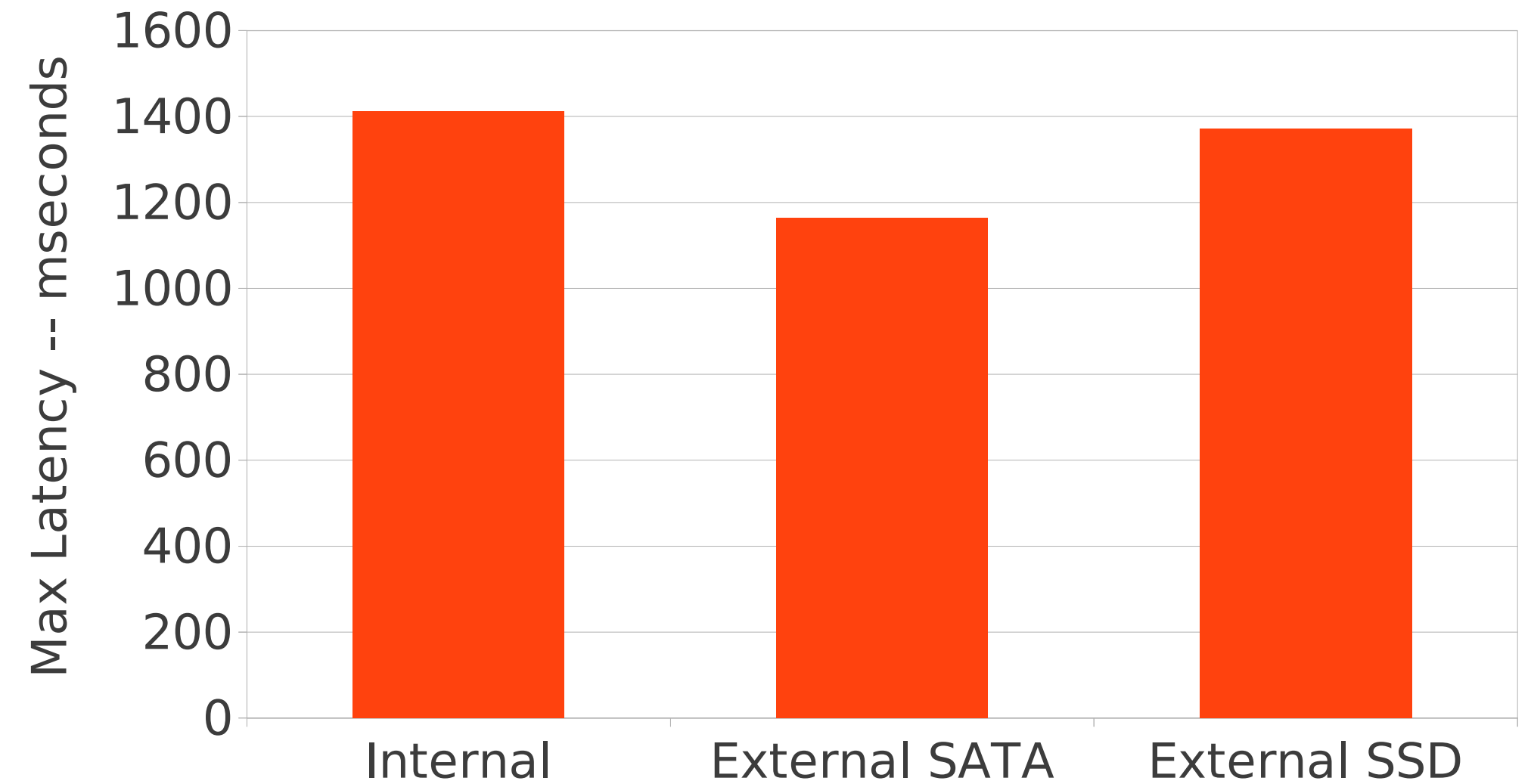
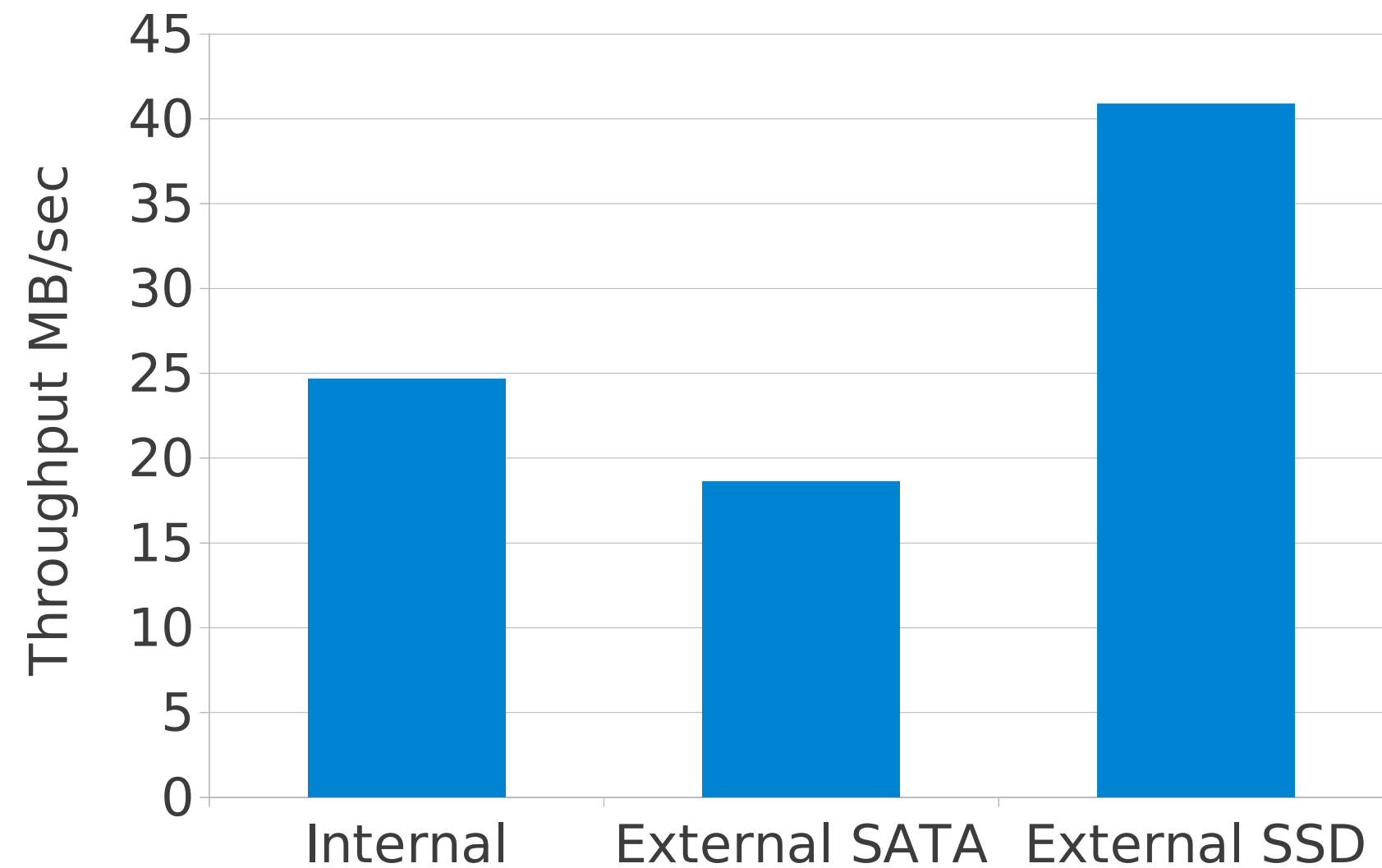
► Delayed logging
Is the only mode
Beginning from
Linux 3.3

Journal

- ▶ Place journal on external faster device to get performance improvements

Dbench Benchmark - Run 600sec Threads 8

AMD FX(tm)-8120 Eight-Core Processor, 16G RAM



Journal

► External log volume support

```
# mkfs.xfs -f /dev/sda7 -l logdev=/dev/sda8,size=512m
meta-data=/dev/sda7 isize=256      agcount=4, agsize=655360 blks
.....
log      =/dev/sda8 bsize=4096     blocks=131072, version=2
         =          sectsz=512     sunit=0 blks, lazy-count=1
.....
# mount -o logdev=/dev/sda8 /dev/sda7 /xfs
# mount | grep xfs
/dev/sda7 on /xfs type xfs (rw,logdev=/dev/sda8)
```

Self-describing Metadata

- ▶ Solution with additional validation information
 - CRC32c
 - File system identifier
 - The owner of a metadata block
 - Logical sequence number of the most recent transaction

The typical on-disk structure

```
struct xfs_ondisk_hdr {  
    __be32  magic;  
    __be32  crc;  
    uuid_t  uuid;  
    __be64  owner;  
    __be64  blkno;  
    __be64  lsn;  
}
```

Self-describing Metadata

The largest scalability problem facing XFS is not one of algorithmic scalability, but of verification of the file system structure...

- Kernel document

Self-describing Metadata

- ▶ Primary purpose
 - Minimize the time and efforts required for basic forensic analysis of PB scale file system
 - Detect common types of errors easily and automatically
- ▶ Existing XFS forensic analysis utilities
 - xfs_repair (xfs_check is effectively deprecated)
 - xfs_db
 - Need pretty much efforts to analysis large storage manually
- ▶ Problems with the current metadata format
 - Magic number is the only way
 - Even worse, lack of magic number identifying in AGFL, remote symlinks as well as remote attribute blocks

Self-describing Metadata - CRC enabled inode format

```
xfs_db> inode 67
xfs_db> p
core.magic = 0x494e
....
core.version = 3
....
next_unlinked = null
v3.crc = 0x90bc8bba
v3.change_count = 1
v3.lsn = 0x100000002
v3.flags2 = 0
v3.crttime.sec = Thu Oct 10 14:40:53 2013
v3.crttime.nsec = 543419982
v3.inumber = 67
v3.uuid = b7232b63-95fc-475e-a454-6f50e2725053
```

```
xfs_db> inode 67
xfs_db> p
core.magic = 0x494e
....
core.version = 2
....
core.filestream = 0
core.gen = 1
next_unlinked = null
```

- ▶ CRC increase the v3 inode size to 512 bytes, while v2 is 256 bytes by default

Self-describing Metadata - v3 inode format

```

# blkid /dev/sda8
/dev/sda8: UUID="b7232b63-95fc-475e-a454-6f50e2725053" TYPE="xfs"

xfs_db> type text
xfs_db> p
000:  49 4e 81 a4 03 02 00 00 00 00 00 00 00 00 00 00 00  IN.....
010:  00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 01  .....
020:  52 56 4b f5 20 63 ee 4e 52 56 4b f5 20 63 ee 4e  RVK..c.NRVK..c.N
030:  52 56 4b f5 20 63 ee 4e 00 00 00 00 00 00 00 00  RVK..c.N.....
040:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
050:  00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00  .....
060:  ff ff ff ff 90 bc 8b ba 00 00 00 00 00 00 00 00 01  .....
070:  00 00 00 01 00 00 00 02 00 00 00 00 00 00 00 00  .....
080:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
090:  52 56 4b f5 20 63 ee 4e 00 00 00 00 00 00 00 43  RVK..c.N.....C
0a0:  b7 23 2b 63 95 fc 47 5e a4 54 6f 50 e2 72 50 53  ...c..G..ToP.rPS

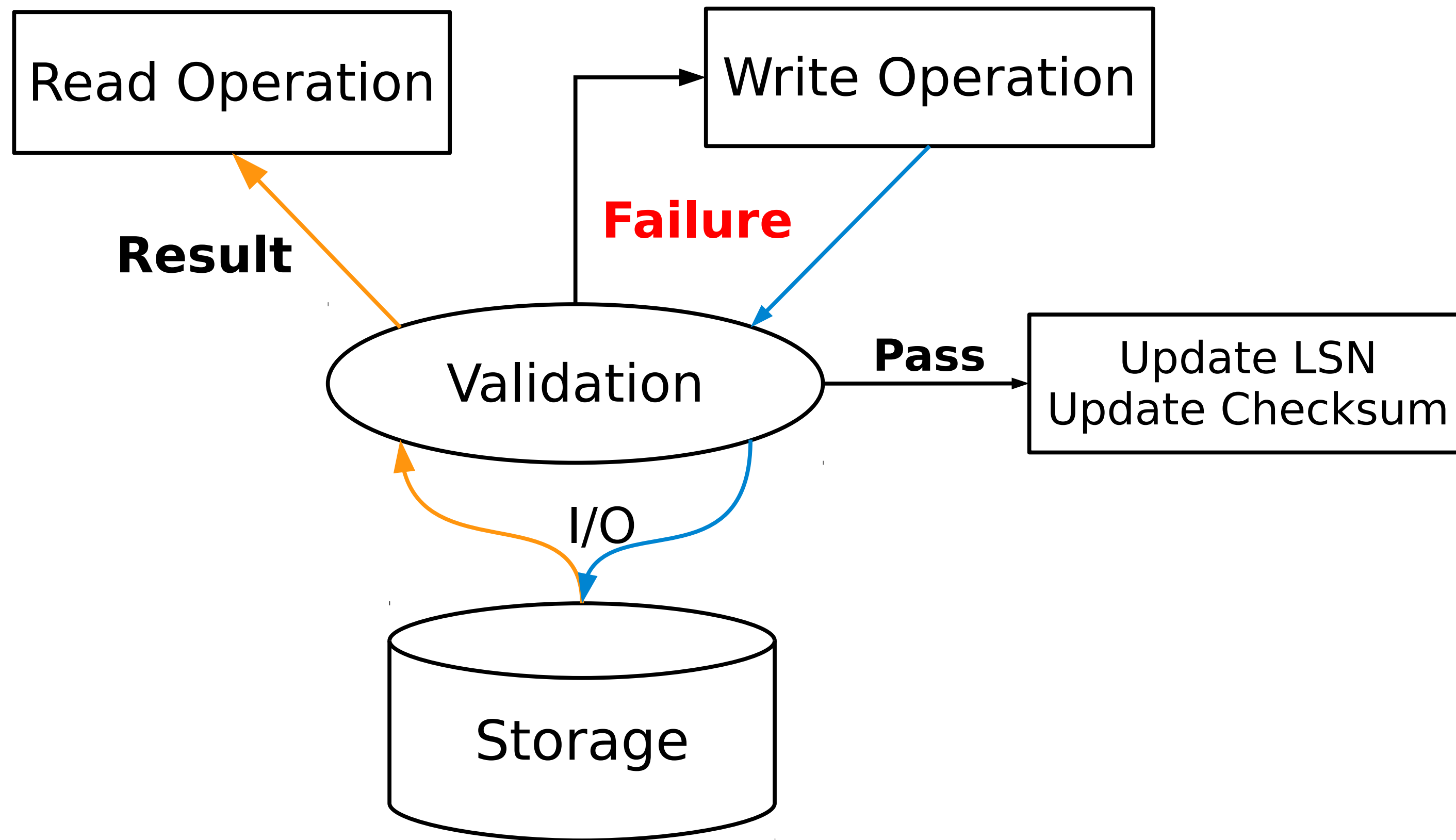
```

- CRC
- LSN
- INUMBER
- UUID

Self-describing Metadata

► Runtime Validation

- Immediately after a successful read from disk
- Immediately prior to write IO submission



Self-describing Metadata

► Enable CRC validation

```
# mkfs.xfs -f -m crc=1 /dev/sda8
meta-data=/dev/sda8          isize=512    agcount=4, agsize=1402050 blks
                        =               sectsz=512   attr=2, projid32bit=1
                        =               crc=1
data        =               bsize=4096    blocks=5608198, imaxpct=25
                        =               sunit=0     swidth=0 blks
naming      =version 2       bsize=4096   ascii-ci=0
log         =internal log    bsize=4096   blocks=2738, version=2
                        =               sectsz=512   sunit=0 blks, lazy-count=1
realtime    =none           extsz=4096   blocks=0, rtextents=0
Version 5 superblock detected. xfsprogs has EXPERIMENTAL support enabled!
Use of these features is at your own risk!
```

Self-describing Metadata

- ▶ Does it cause noticeable overhead? Basically No

Compilebench	Runs	Non-CRC		CRC	
Initial create	30	21.18	MB/s	19.70	MB/s
Create	14	13.91	MB/s	13.09	MB/s
Patch	15	4.85	MB/s	4.72	MB/s
Compile	14	38.14	MB/s	37.29	MB/s
Clean	10	235.85	MB/s	245.28	MB/s
Read tree	11	10.00	MB/s	8.98	MB/s
Read compiled tree	4	16.56	MB/s	16.12	MB/s
Delete tree	10	8.60	Sec	9.23	Sec
Delete compiled tree	4	11.05	Sec	11.82	Sec
Stat tree	11	4.83	Sec	4.52	Sec
Stat compiled tree	7	6.80	Sec	6.97	Sec

Self-describing Metadata

► Compatibility

- Old file system does not support the new disk format
- Old kernel and userspace can not read the new format
- Kernel and userspace that support the new format works just fine with the old non-CRC check enabled format
- Support two different incompatible disk formats from this point onwards
- **Will not provide tools to convert the format of existing file system?**

Performance & Scalability

XFS performs very well with large files but very poor to handle lots of small files - From the Internet

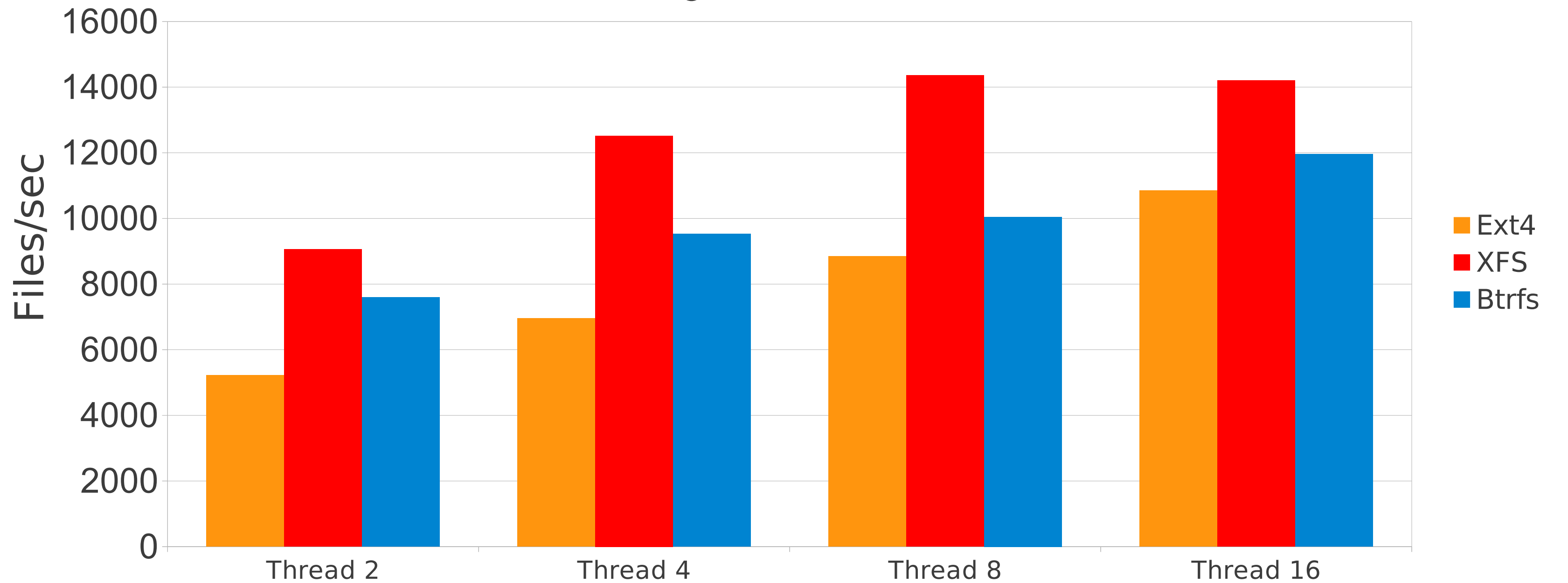
Maybe that's an old story!

Performance & Scalability

Fs_mark ZERO file creation benchmark -- Linux 3.10

Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 8G RAM, Normal SATA Disk

- Keep: Yes
- Sync Method: 0
- Directory: 10
- File: 1000
- Byte Count: 0
- Thread: 2/4/8/16

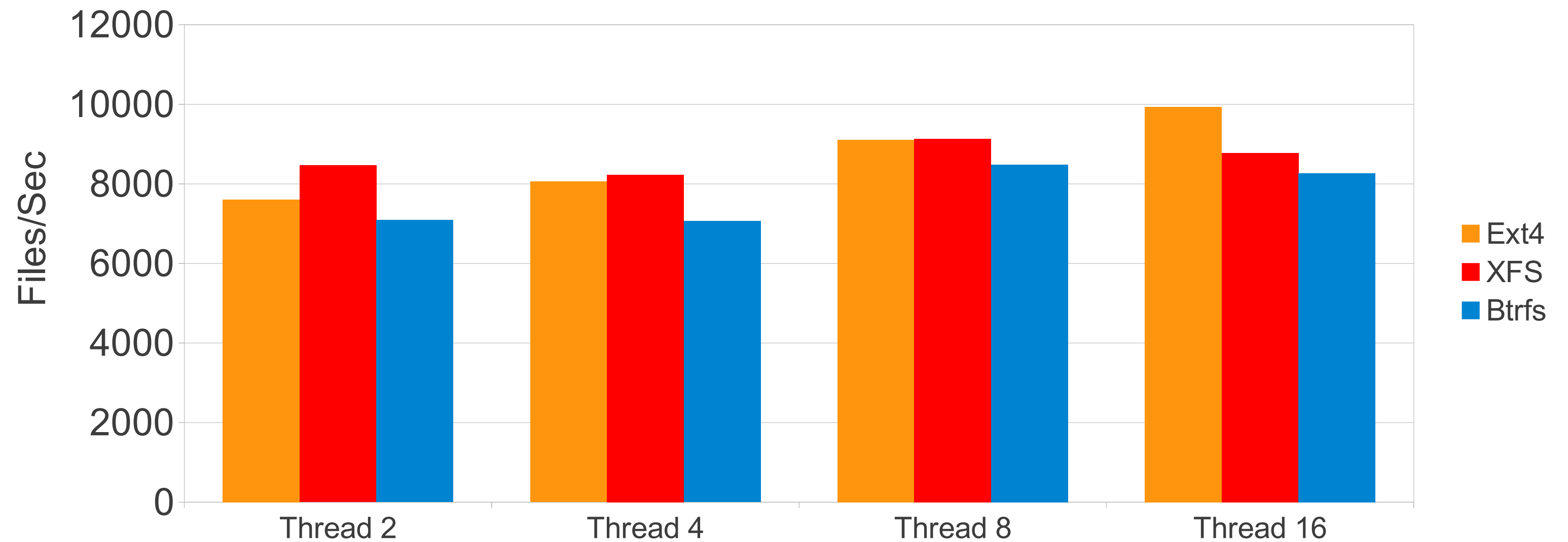


Performance & Scalability

Fs_mark Creation/Deletion Benchmark -- Linux 3.10

Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 8G RAM, Normal SATA Disk

- Keep: No
- Sync Method: 0
- Directory: 10
- File: 1000
- Byte Count: 0
- Thread: 2/4/8/16

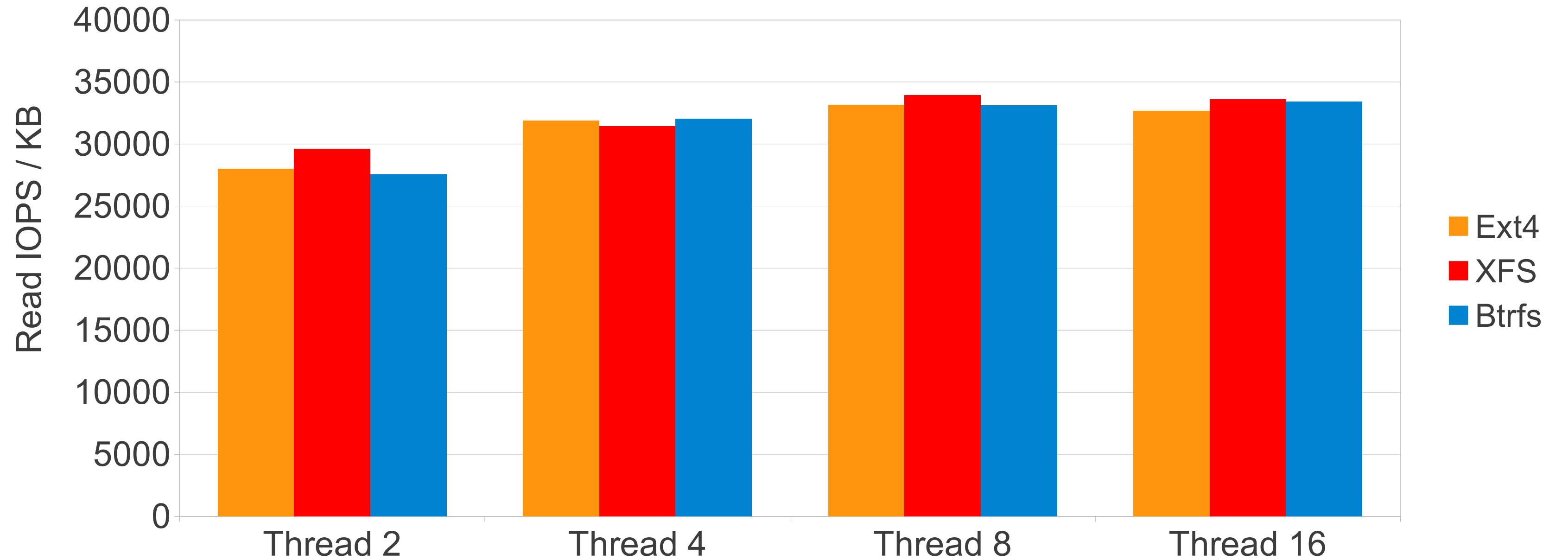


Performance & Scalability

Fio Read Benchmark - Linux 3.10

Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 8G RAM, Normal SATA Disk

- RW: Random
- Block Size: 4K
- File Size: 1GB
- Runtime: 120 Sec
- NumJobs: 2/4/8/16

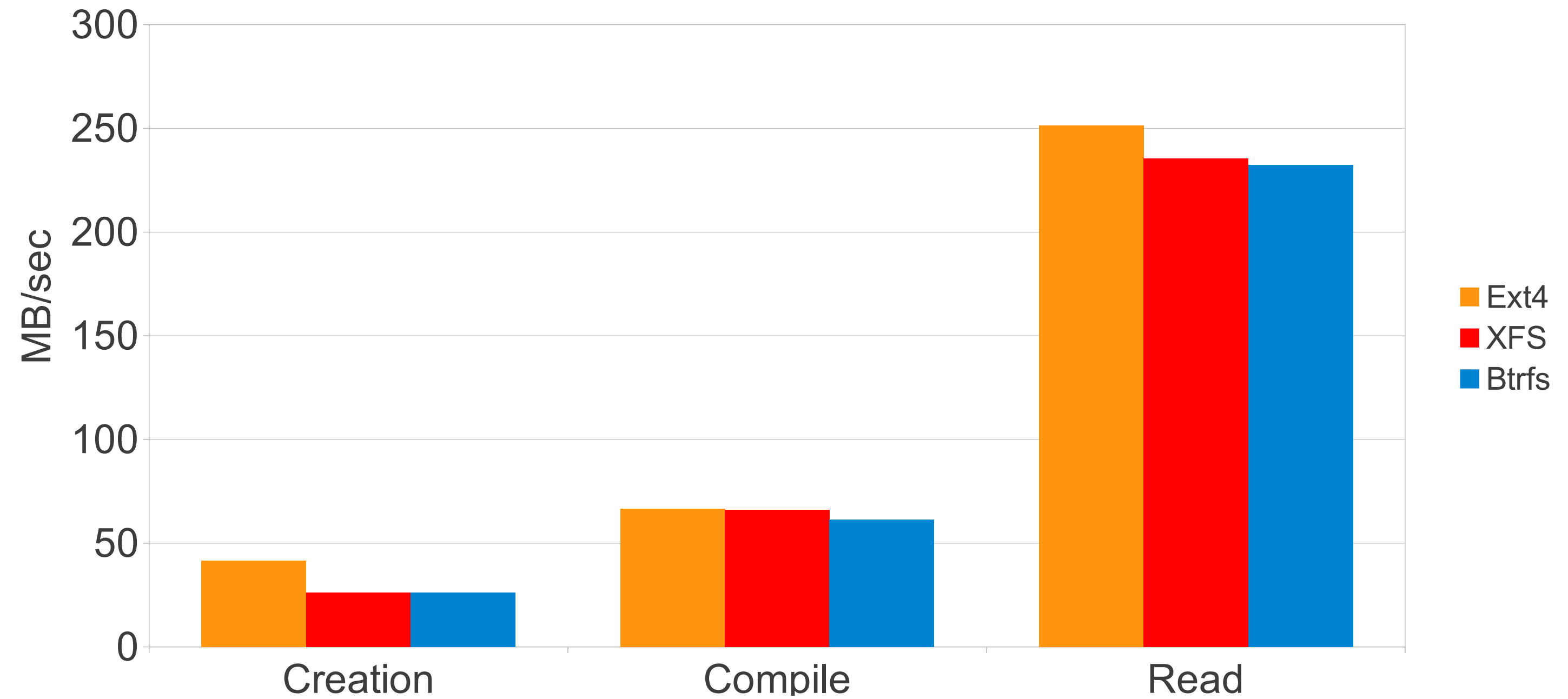


Performance & Scalability

Compilebench -- Linux 3.10

Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 8G RAM, Normal SATA Disk

- Initial Dir: 10
- Makej: Yes

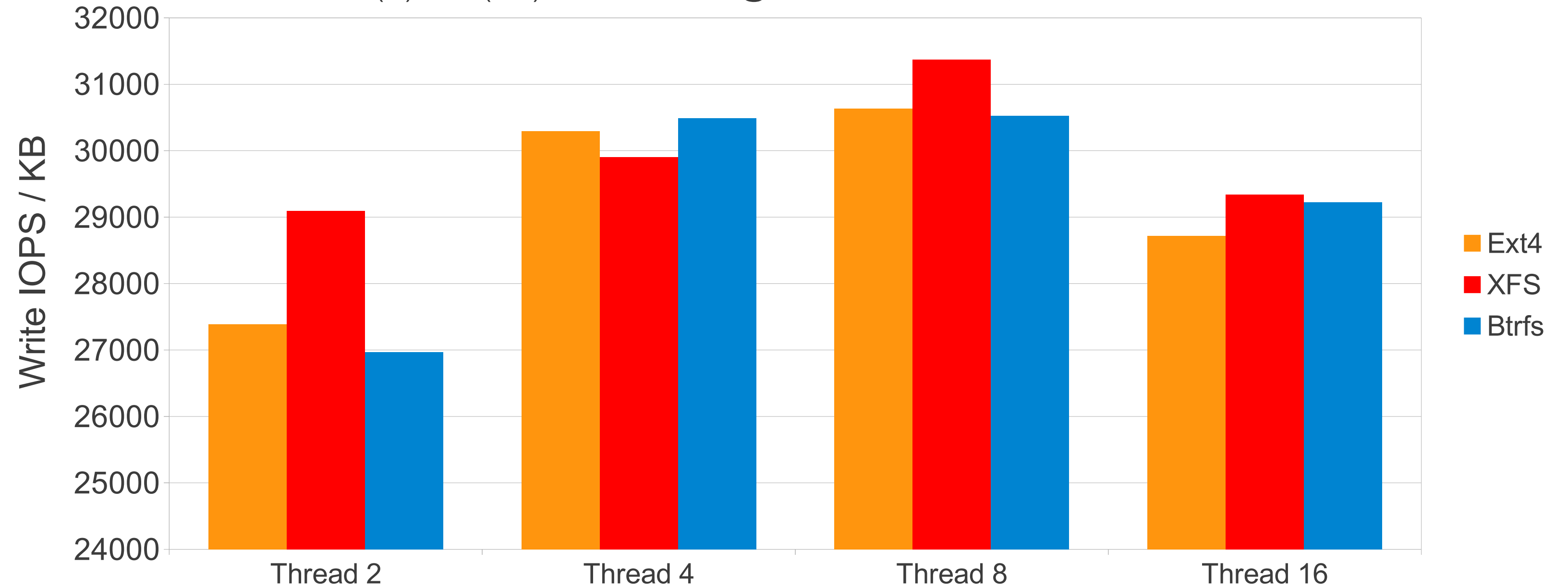


Performance & Scalability

Fio Write Benchmark -- Linux 3.10

Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 8G RAM, Normal SATA Disk

- RW: Random
- Block Size: 4K
- File Size: 1GB
- Runtime: 120 Sec
- NumJobs: 2/4/8/16



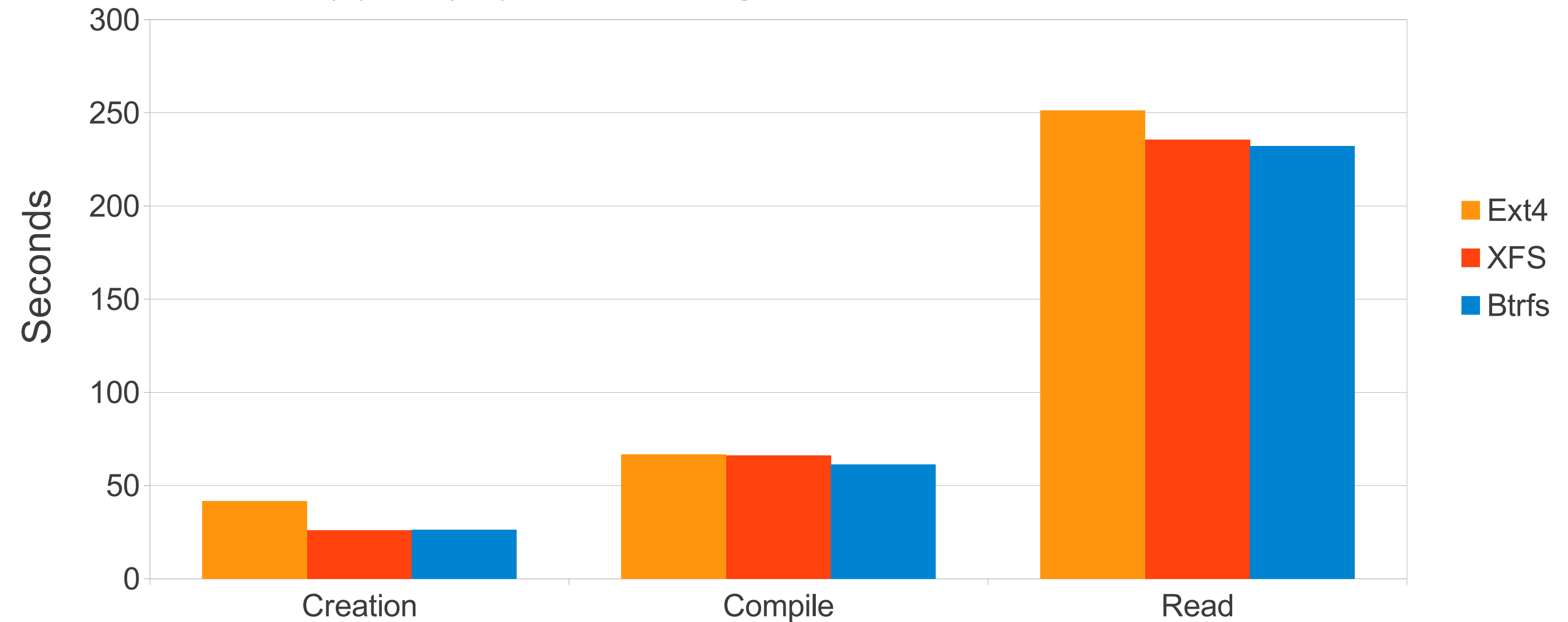
Performance & Scalability

A more realistic case of creation/deletion

time/tar/rm
Uncompress Linux
3.10-bz2
package and remove

Creation/Deletion Benchmark for small files -- Linux 3.10

Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 8G RAM, Normal SATA Disk



What's new && Wait in progress

- ▶ CONFIG_XFS_WARN
 - Less overhead than the old CONFIG_XFS_DEBUG option
 - Suitable to production environment
- ▶ Separate project disk quota inode
 - Can be enabled with group quota at the same time
- ▶ User namespace support for Linux container (LXC)
- ▶ Online file system shrink support (WIP)
- ▶ The free inode btree (WIP)

Thank You!