The Btrfs Filesystem

Chris Mason

Oracle

Sept 2007

Topics

Storage

- Drives are getting bigger, but not much faster
- File sizes are often small
- Fsck times can be very large
- Filesystem scaling issues
  - Performance
  - Management
  - Space utilization

A New Filesystem?

- Existing filesystems not well suited
- Disk format compatibility
- Upstream integration

ORACLE

Btrfs Design

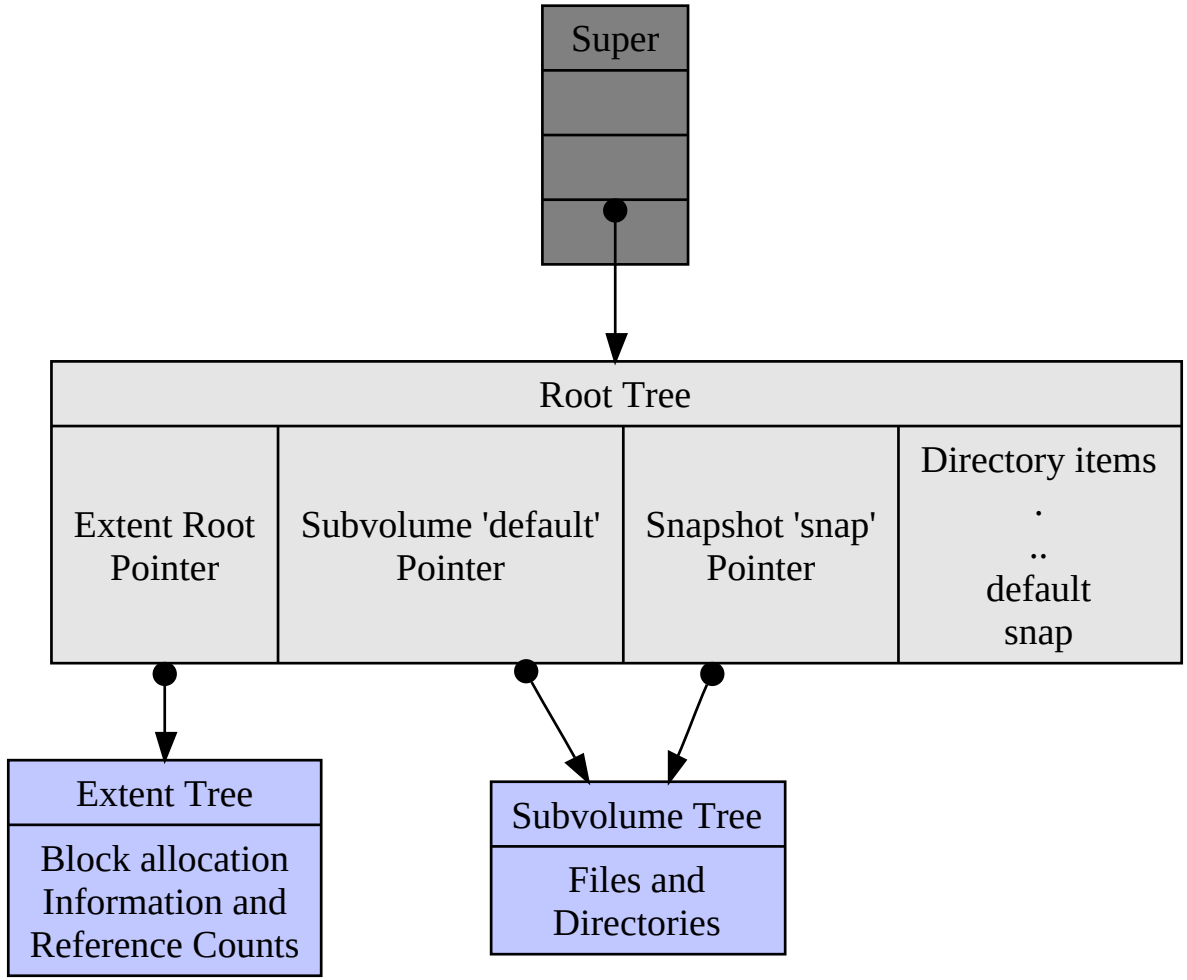
- Extent based file storage
- Space efficient packing of small files
- Space efficient indexed directories
- Integrity checksumming (data and metadata)
- Writable snapshots
- Efficient incremental backups
- Object level striping and mirroring
- Online filesystem check

ORACLE

Btrfs Btree Basics

- Simple copy on write tree
- Reference counting for filesystem trees
- Everything stored as a key/value pair
- Large file data extents stored outside the tree
- Defragments before writeback and via an ioctl
- Btree blocks allocated from metadata-only block groups

Btrfs Trees


Super
Root Tree
Extent Root Pointer
Subvolume 'default' Pointer
Snapshot 'snap' Pointer
Directory items
.
..
default
snap
Extent Tree
Block allocation Information and Reference Counts
Subvolume Tree
Files and Directories

Btrfs Files

- Small files stored inside the btree (inline)
- Large files stored in large extents
- Written via delayed allocation
- Variable sized checksums stored in the btree

Btrfs Directories

- Double indexed by name and inode number
- Compact: space reclaimed as names are removed
- Backpointers from the file to the directory
- Used to implement xattrs

Directory Layout Benchmarks

- Create one million files in a single directory (512 bytes or 16k)
- Time find, backup and delete operations

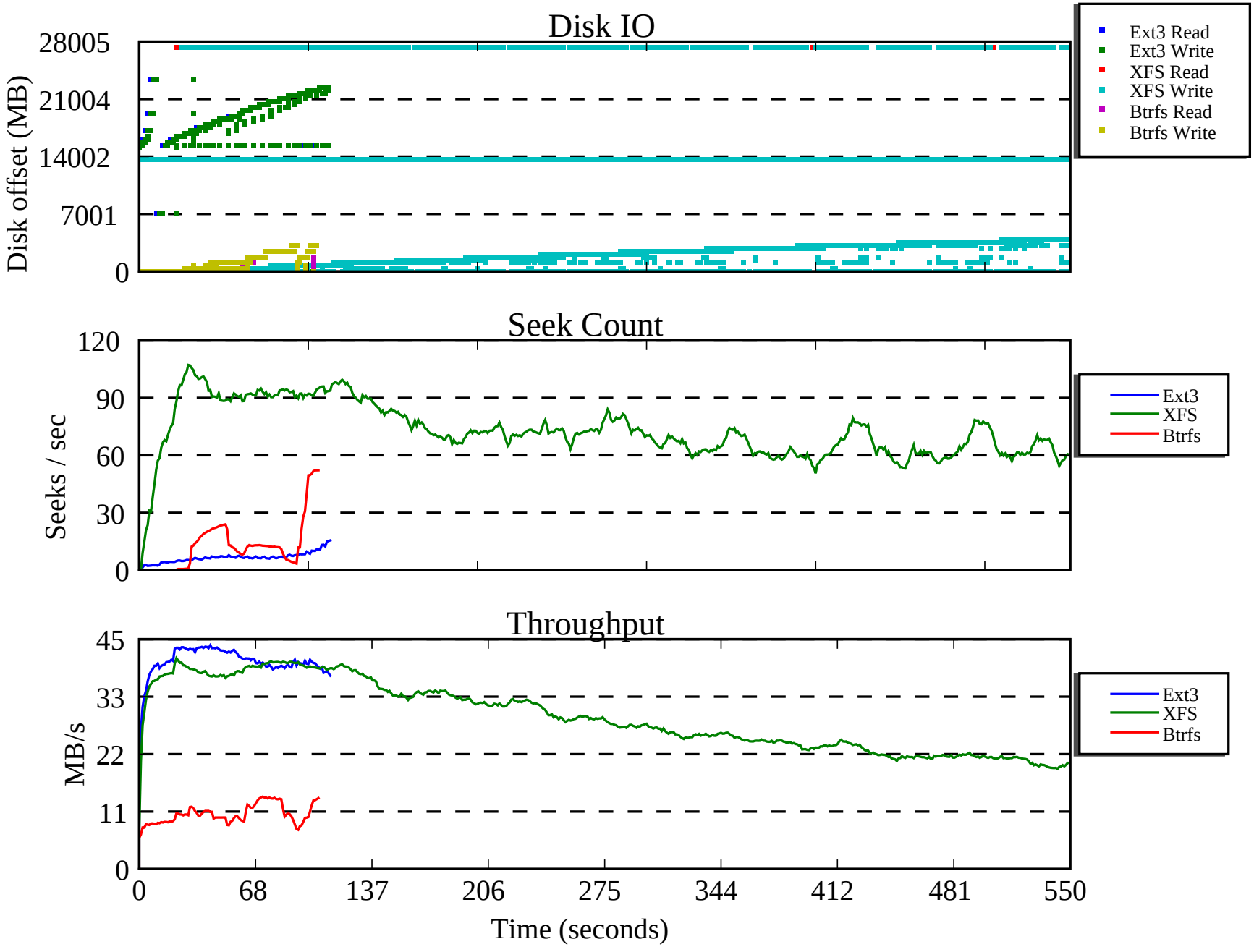


Figure: Create one million 512 byte files

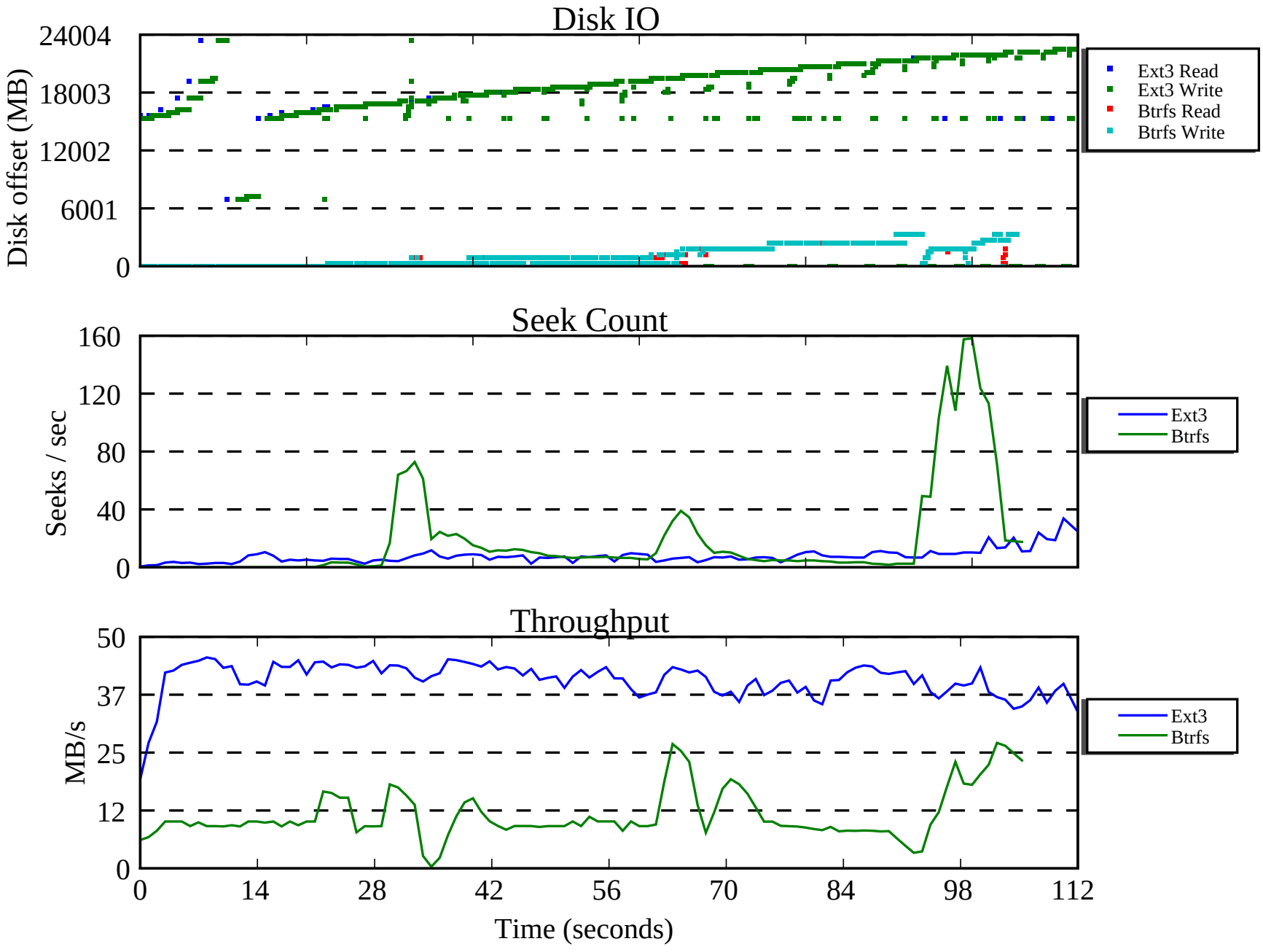


Figure: Create one million 512 byte files

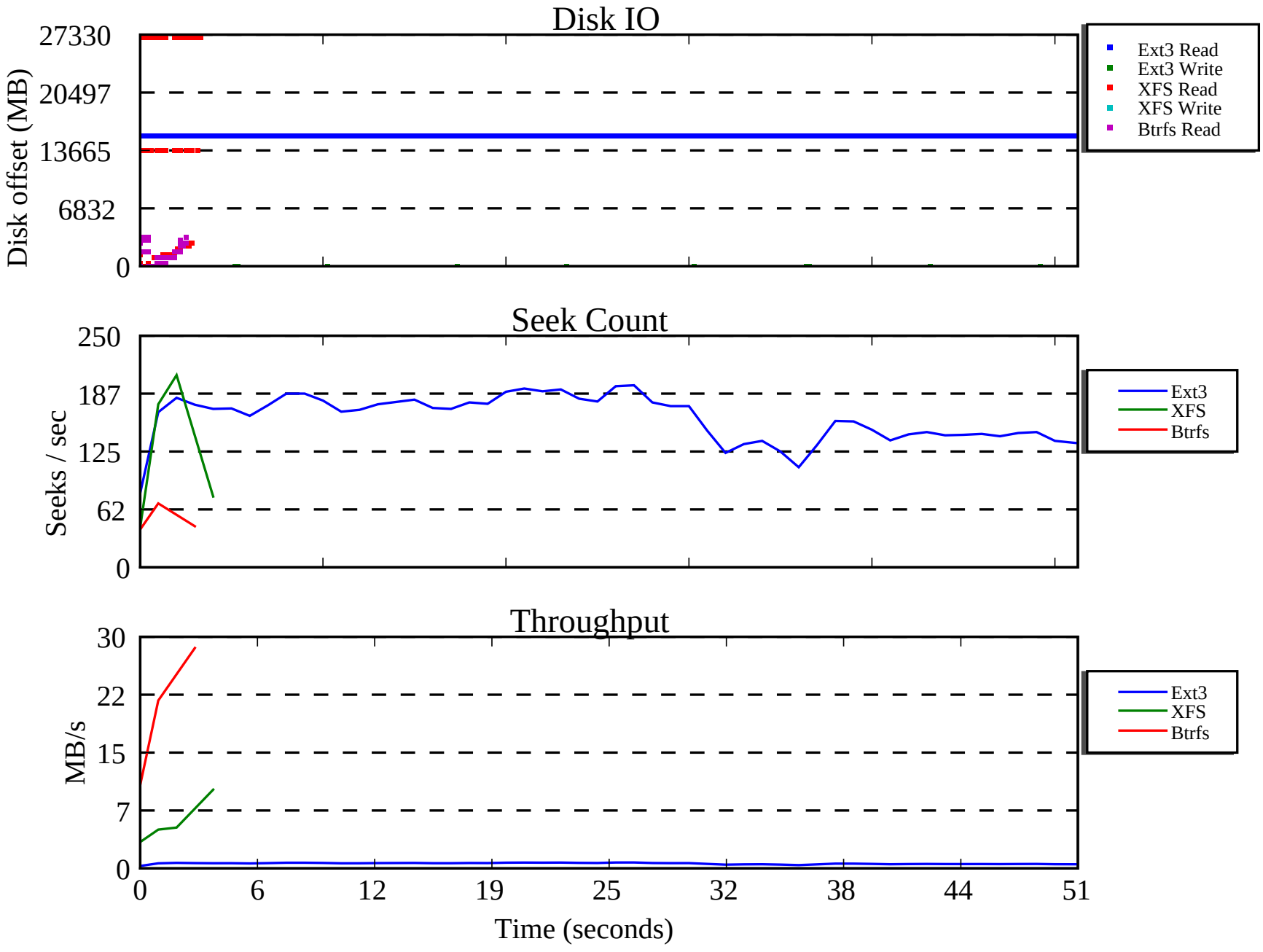


Figure: Find one million 512 byte files

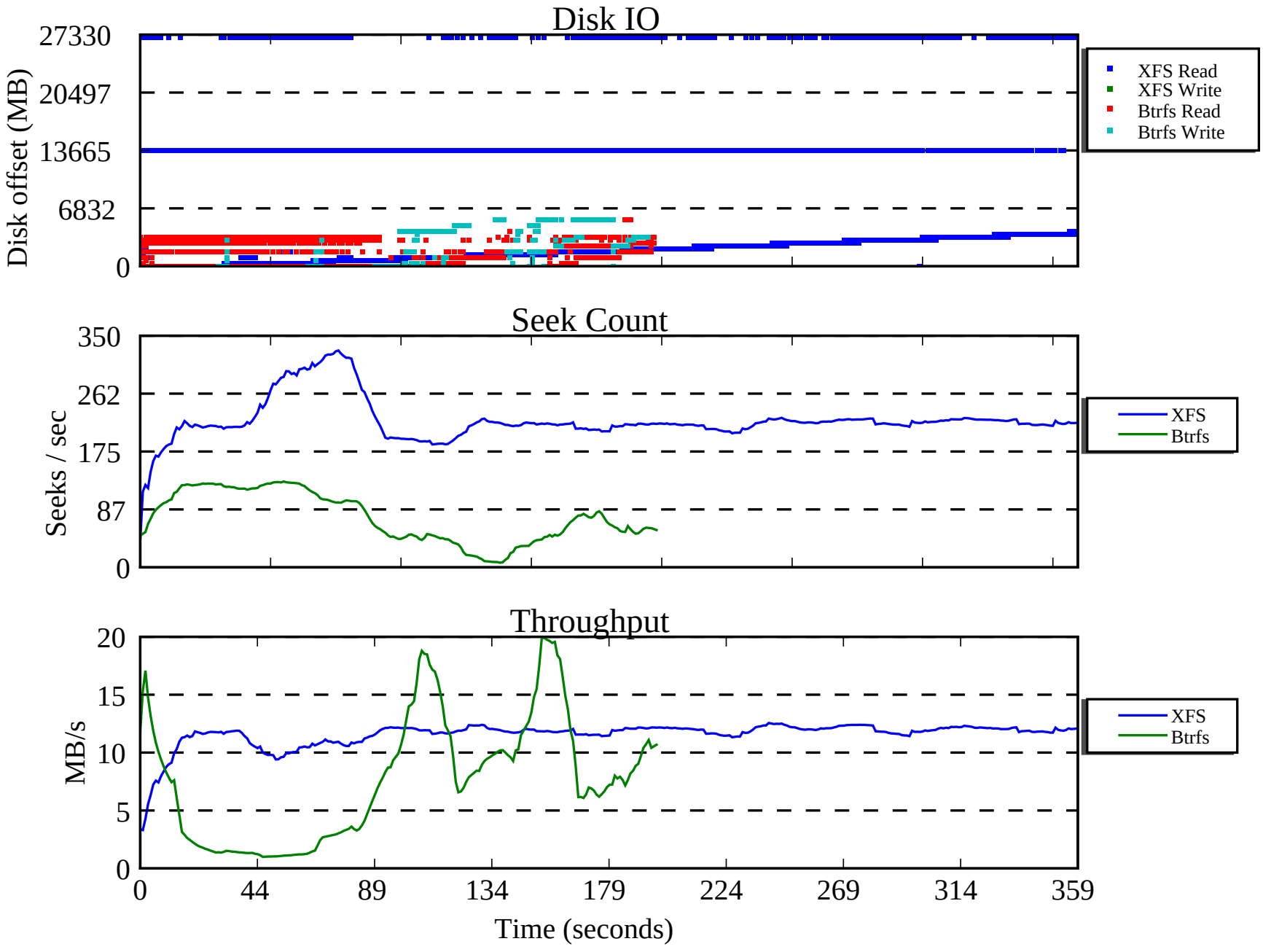


Figure: Read one million 512 byte files

Btrfs Snapshots

- Subvolume trees and file extents are reference counted
- Snapshots are writable and can be snapshotted again

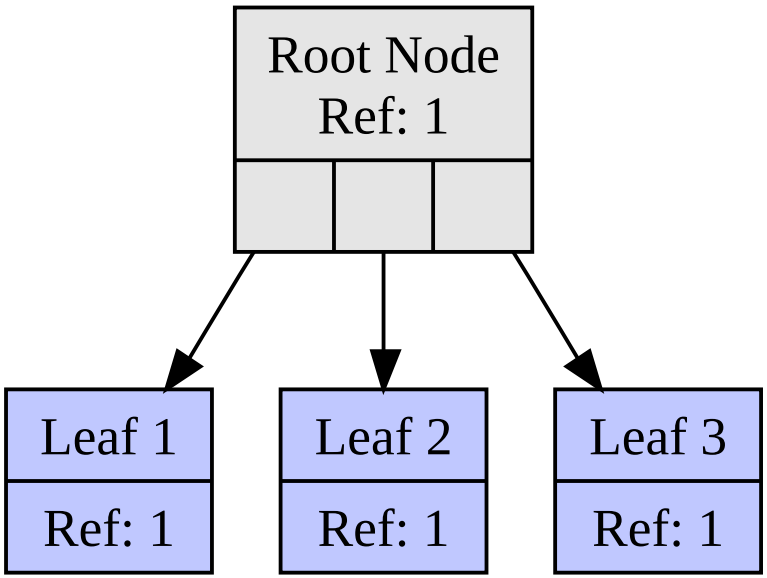


Figure: Before COW

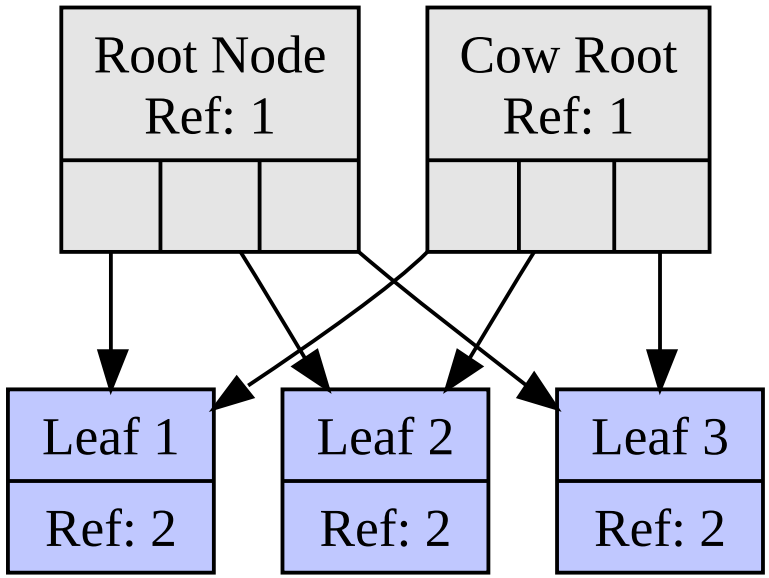


Figure: After COW

Btrfs Snapshots

- Transactions are snapshots that are deleted after commit

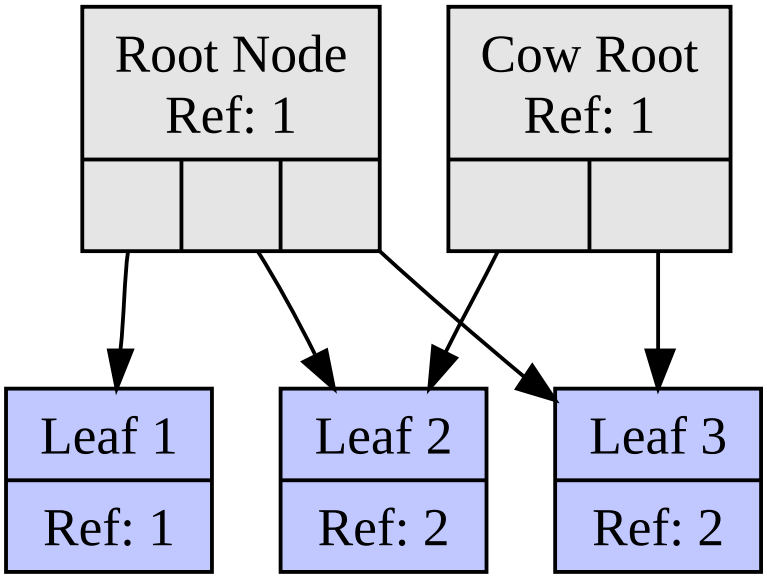


Figure: Modification after COW

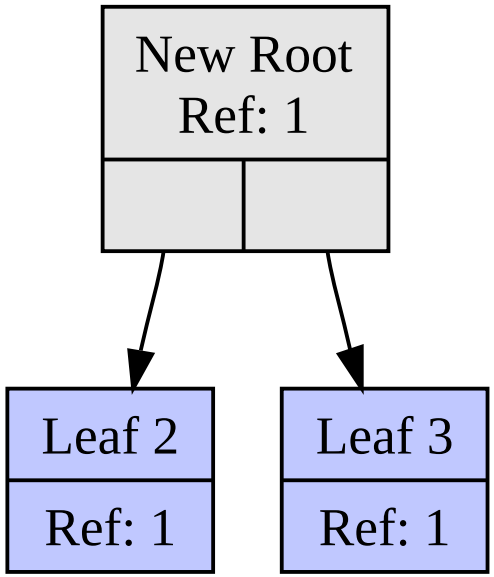


Figure: After Deletion

Btrfs Allocation Trees

- Will help model underlying storage (DM integration)
- Record reference counts for all extents
- Contain resizable block groups
- Btrees can pull from one or more allocation trees

ORACLE

Btrfs Benchmarking

- Single threaded, focus on layout
- mkfs.xfs -d agcount=3 -l version=2,logsize=128m
- mount -t xfs -o logbsize=256k
- mount -t ext3 -o data=writeback
- Single SATA drive, no barriers used
- atimes left on
- Graphs from `http://oss.oracle.com/~mason/seekwatcher`

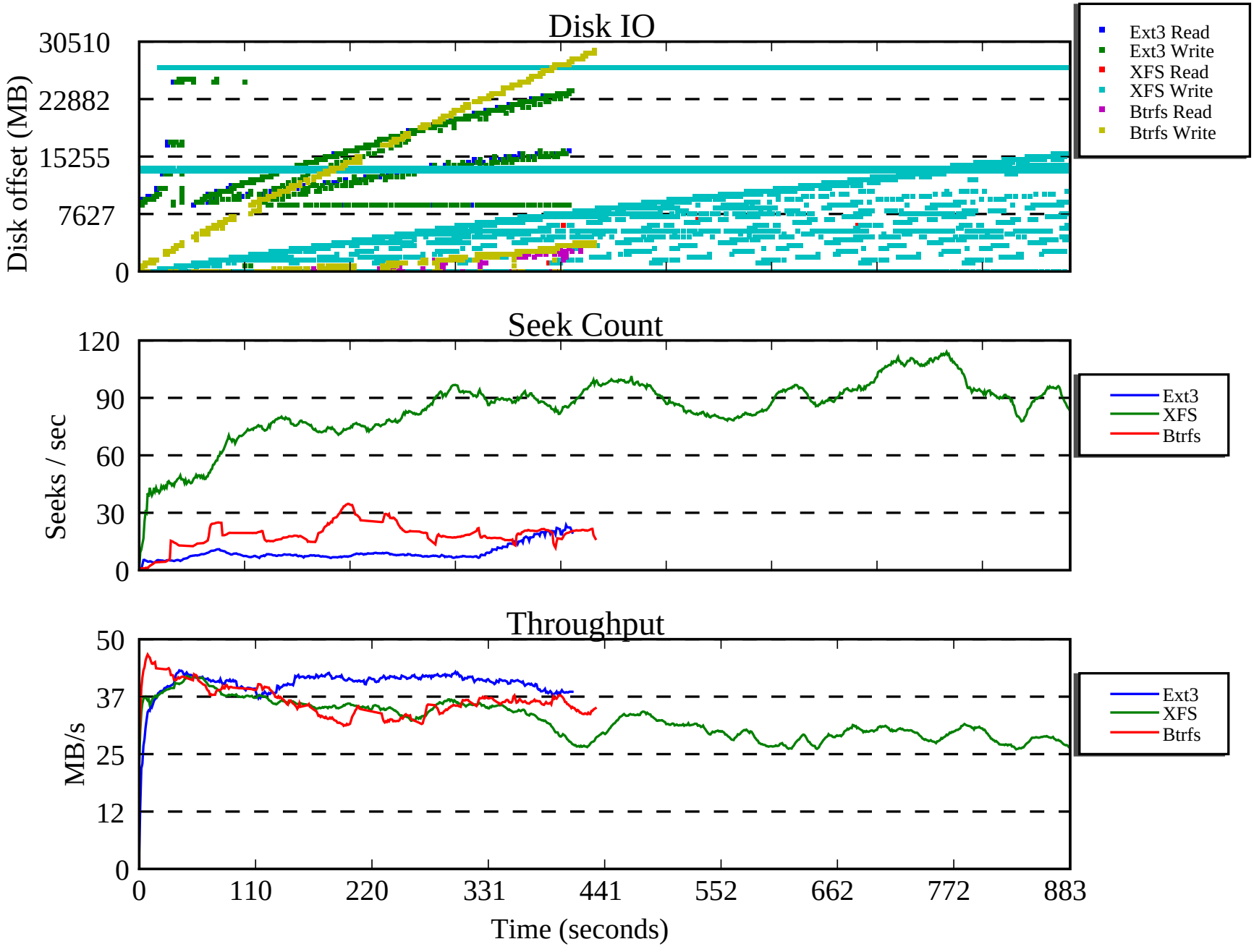


Figure: Create one million 16 KB files

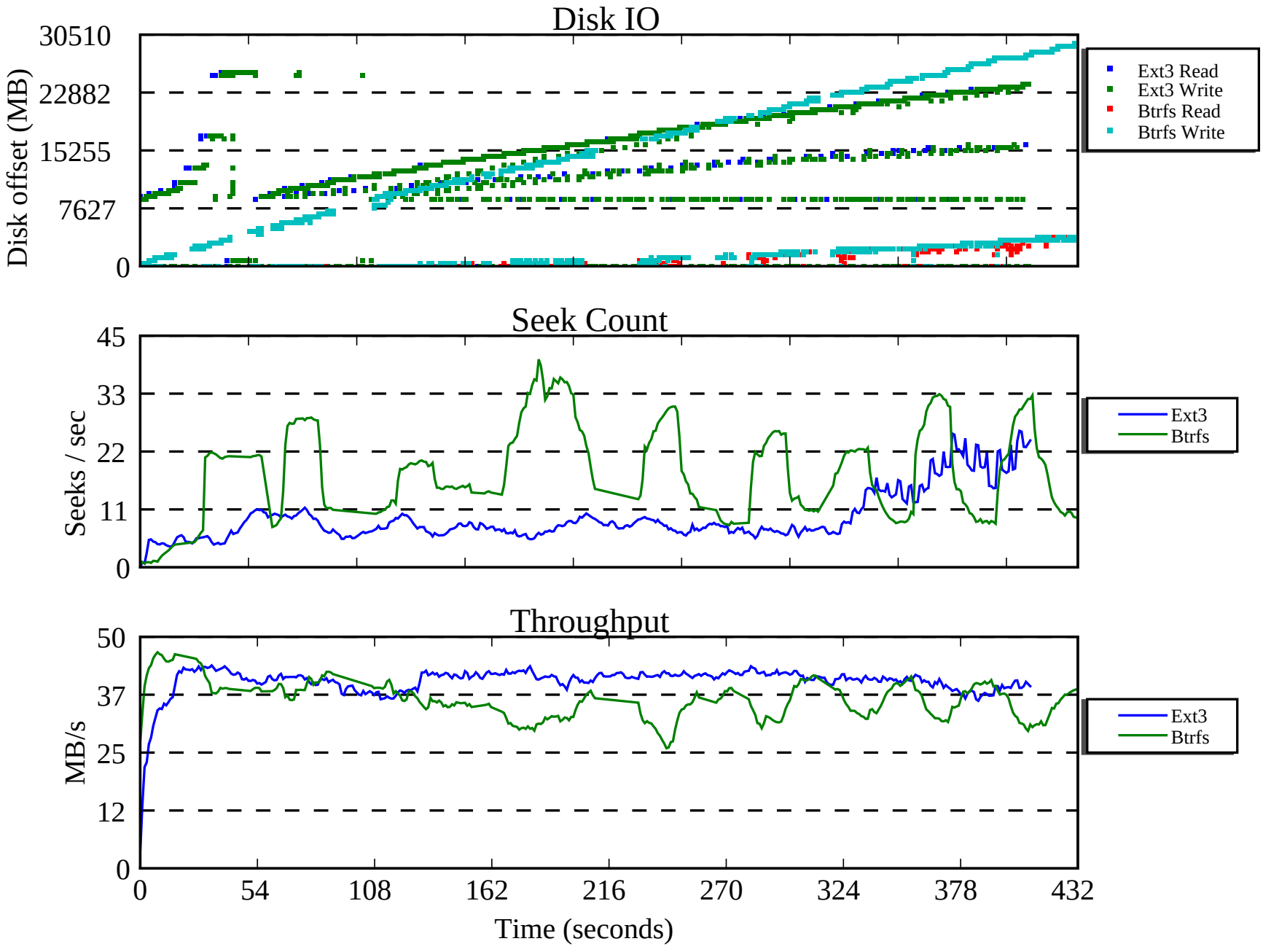


Figure: Create one million 16 KB files

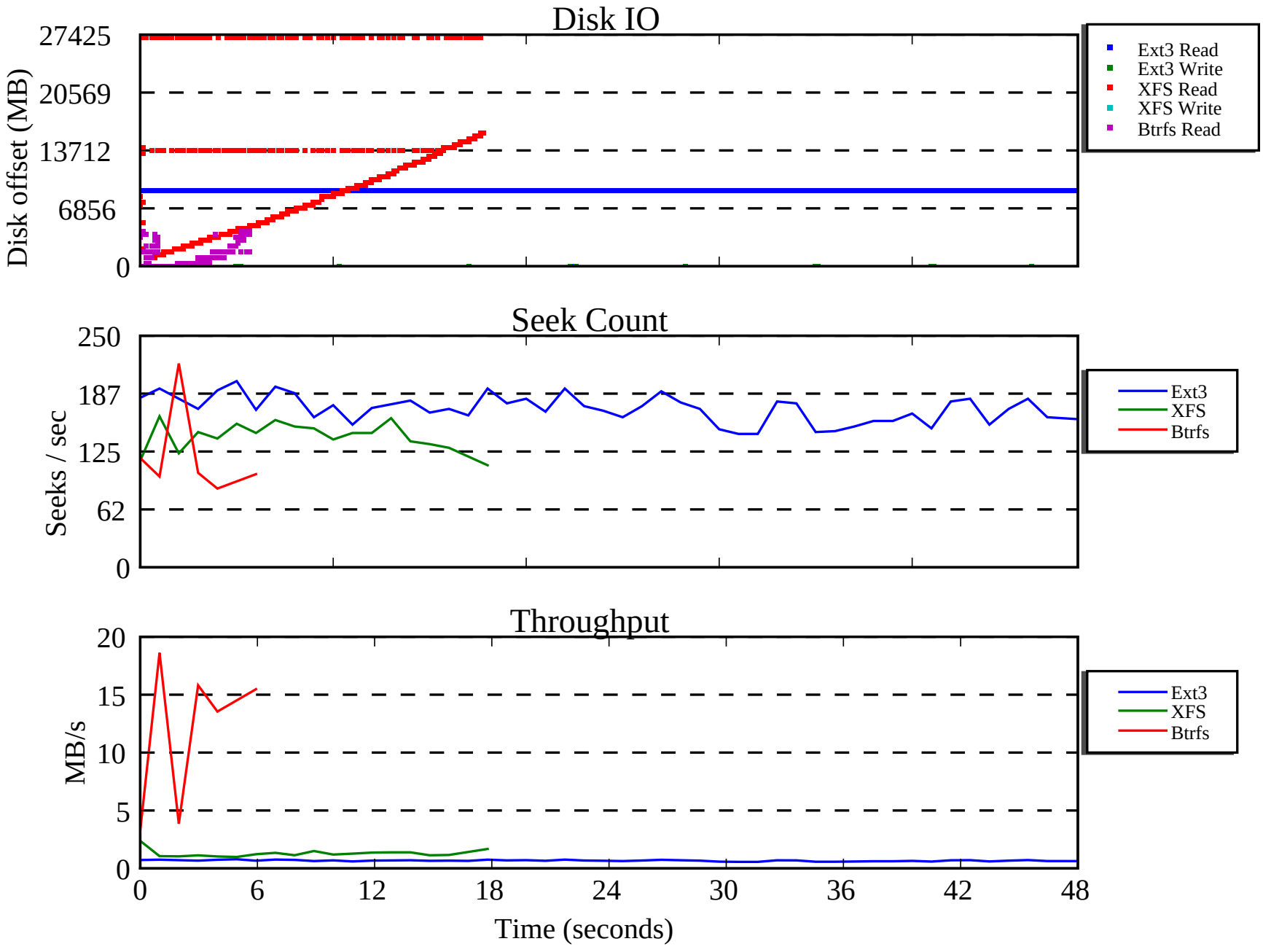


Figure: Find one million 16 KB files

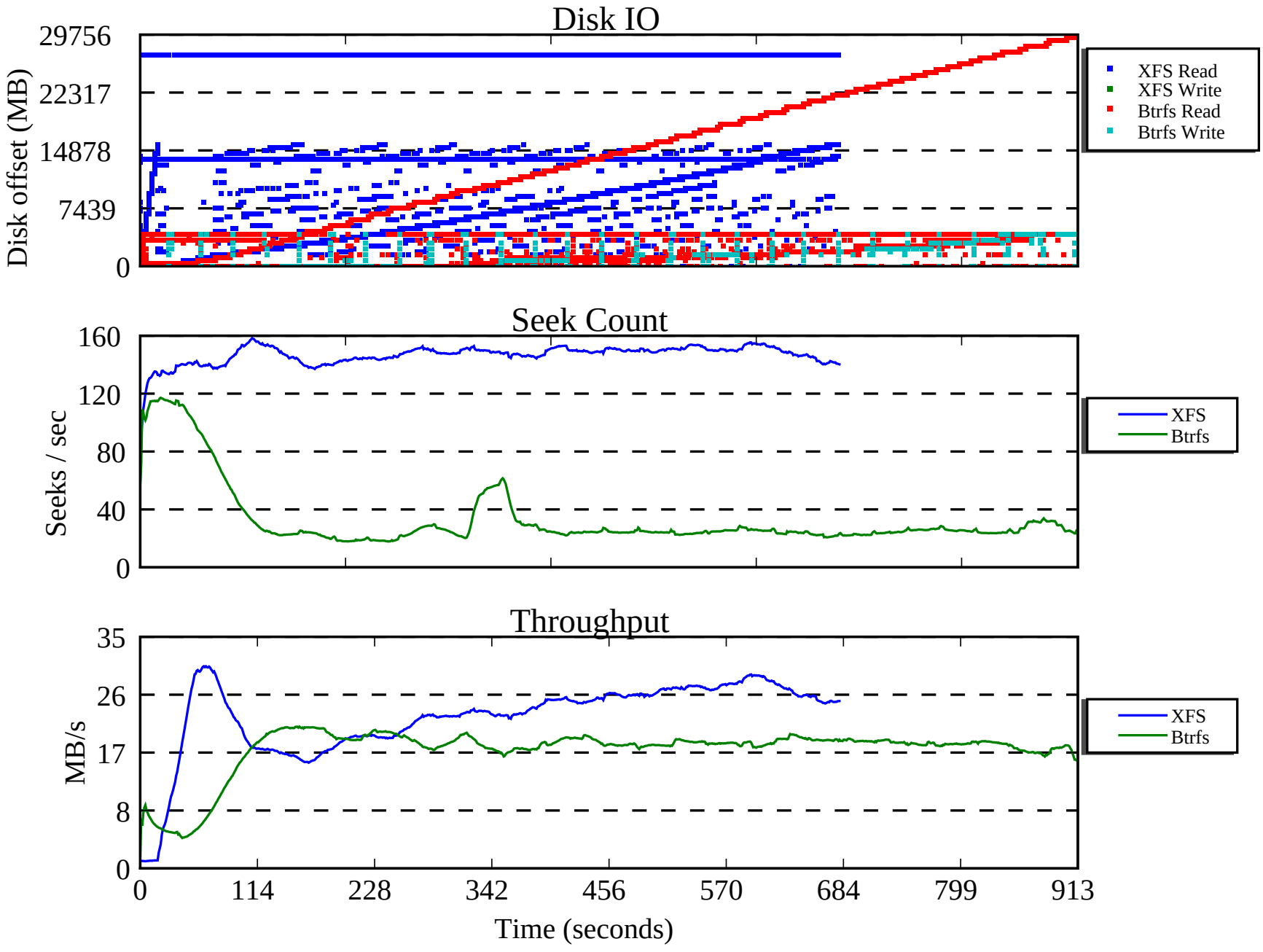


Figure: Read one million 16 KB files

Compilebench

- Simulates operations on kernel trees
- File sizes and names collected from real kernel trees
- Randomly create, compile, patch, clean, delete
- Intended to age and fragment the FS
- Created 90 initial trees and ran 150 operations
- `http://oss.oracle.com/~mason/compilebench`

Compilebench

	Btrfs	Ext3	XFS
Initial Create	33.78 MB/s	10.31 MB/s	15.30 MB/s
Create	14.39 MB/s	12.47 MB/s	8.94 MB/s
Patch	3.77 MB/s	4.48 MB/s	3.19 MB/s
Compile	20.56 MB/s	15.28 MB/s	21.61 MB/s
Clean	69.96 MB/s	43.84 MB/s	134.15 MB/s
Read	5.37 MB/s	8.66 MB/s	7.05 MB/s
Read Compiled	11.31 MB/s	13.22 MB/s	14.40 MB/s
Delete	15.77s	18.34s	16.63s
Delete Compiled	22.96s	33.35s	21.38s
Stat	15.86s	11.35s	7.25s
Stat Compiled	21.82s	14.17s	8.99s

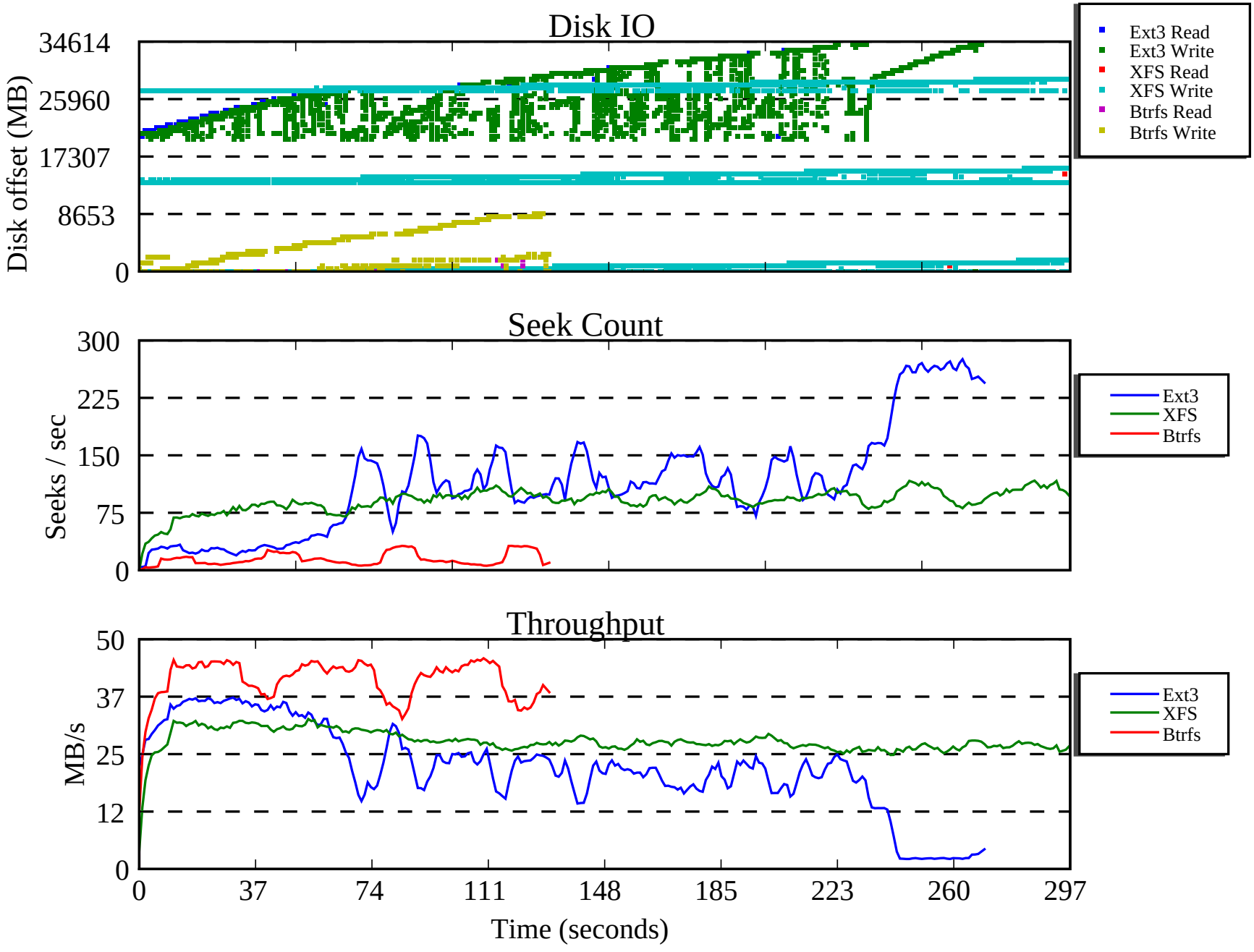


Figure: Create 20 kernel trees

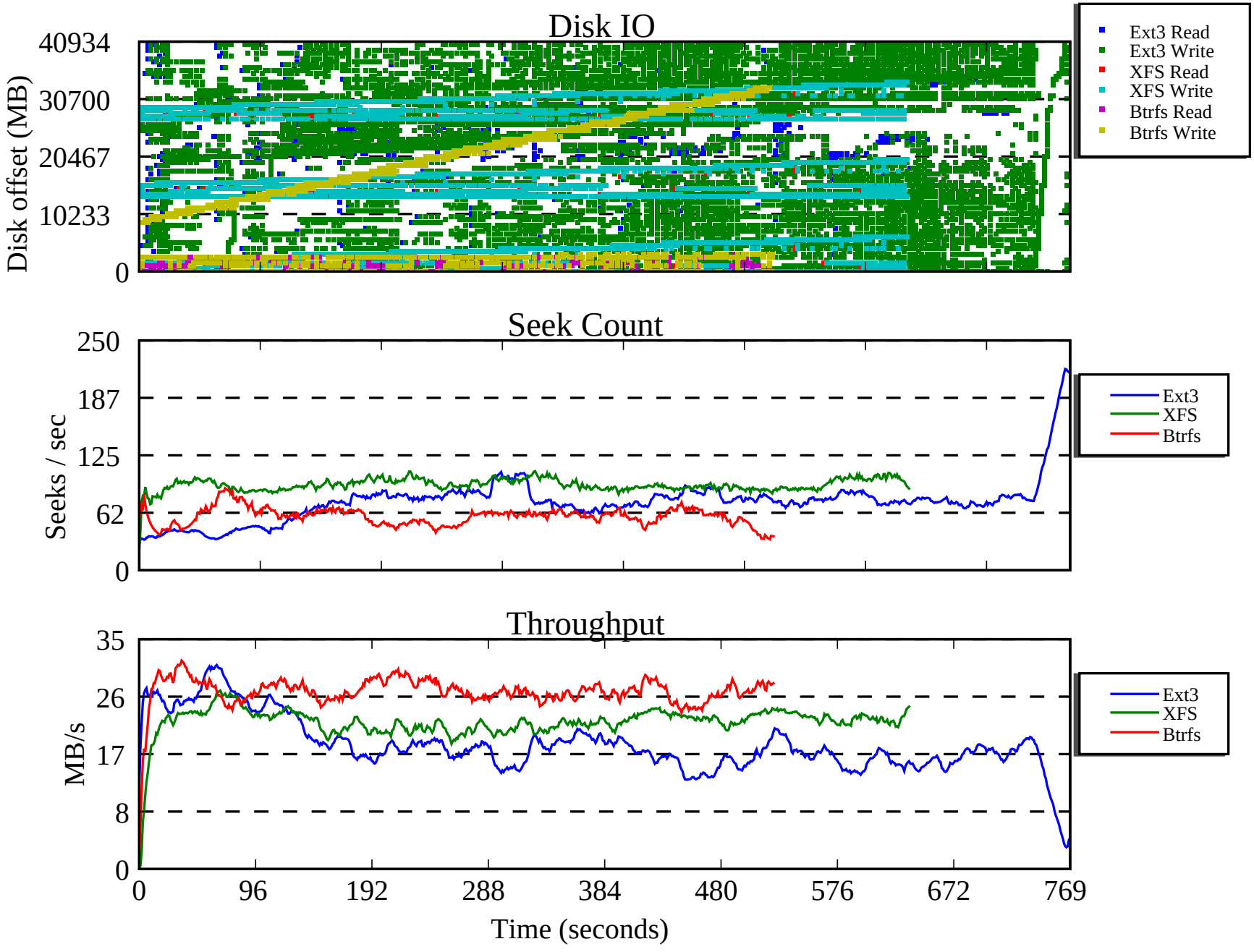


Figure: Make -j simulated 20 kernel trees

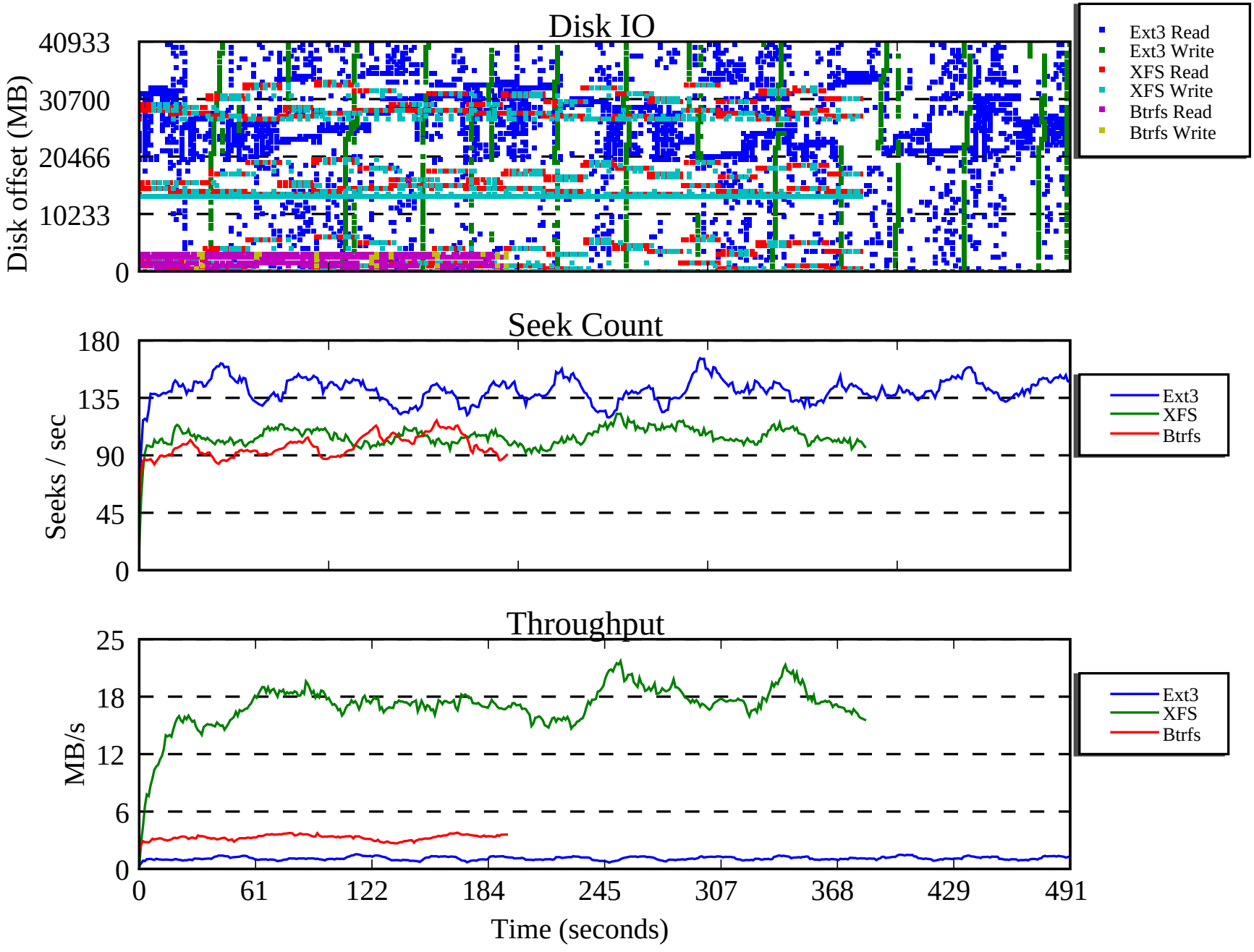


Figure: Delete 20 kernel trees

TODO

- Concurrency (tree locking)
- Multiple extent allocation trees
- OSYNC and fsync performance
- Online fsck
- Many many more