FUJITSU

shaping tomorrow with you

Performance Improvement of Btrfs

Miao Xie <miaox@cn.fujitsu.com>
Li Zefan <lizf@cn.fujitsu.com>

Agenda

FUJITSU

- Comparison between Btrfs and Ext3/4
- Issue analysis (We have investigated)
  - Small file sequential read
  - Large file random write (Direct I/O and fsync)
  - File creation/deletion
- Future work

Comparison between Btrfs and Ext3/4

FUJITSU

- Performance test environment
  - Hardware
    - CPU ： Xeon(TM) X5260 3.33G X 2 （4 cores）
    - Memory ： 4GB
    - Disk ： 20GB
  - Software
    - OS ： RHEL6(x86_64)
    - Kernel ： 2.6.38
    - Glibc ： 2.12
    - Btrfs-progs ： 0.9
    - Sysbench: 0.4.12

Comparison between Btrfs and Ext3/4

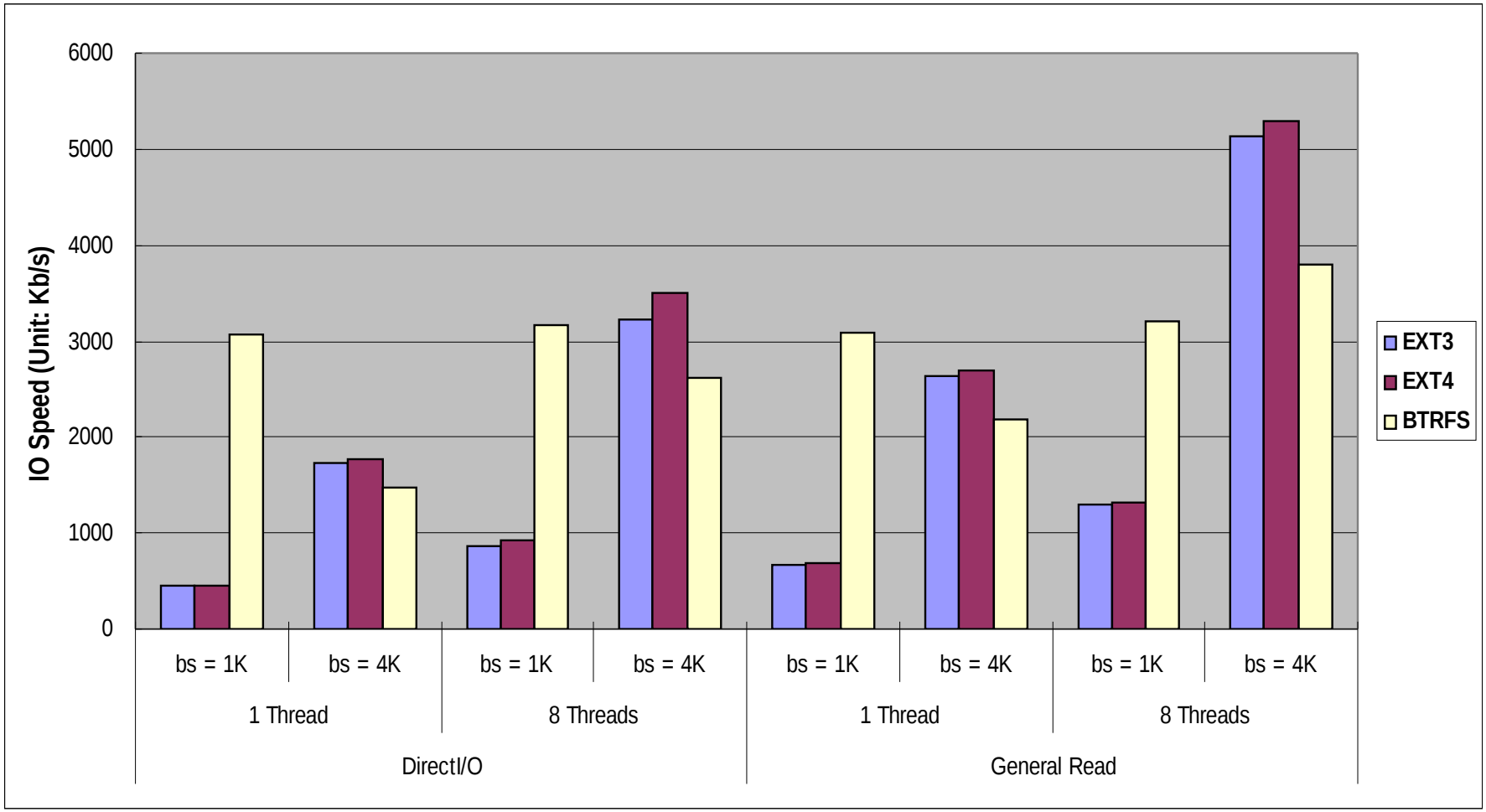
FUJITSU

- 73 cases in total
  - 72 file I/O cases, mix the following conditions:
    - Small file / Large file
    - Write / Read
    - Random / Sequential
    - Sync / Async / Direct I/O
    - Single-thread / Multi-thread
    - Different block size (1Kb, 4Kb, 32Kb) *
  - File creation/deletion
    - Measure the speed of empty file creation/deletion

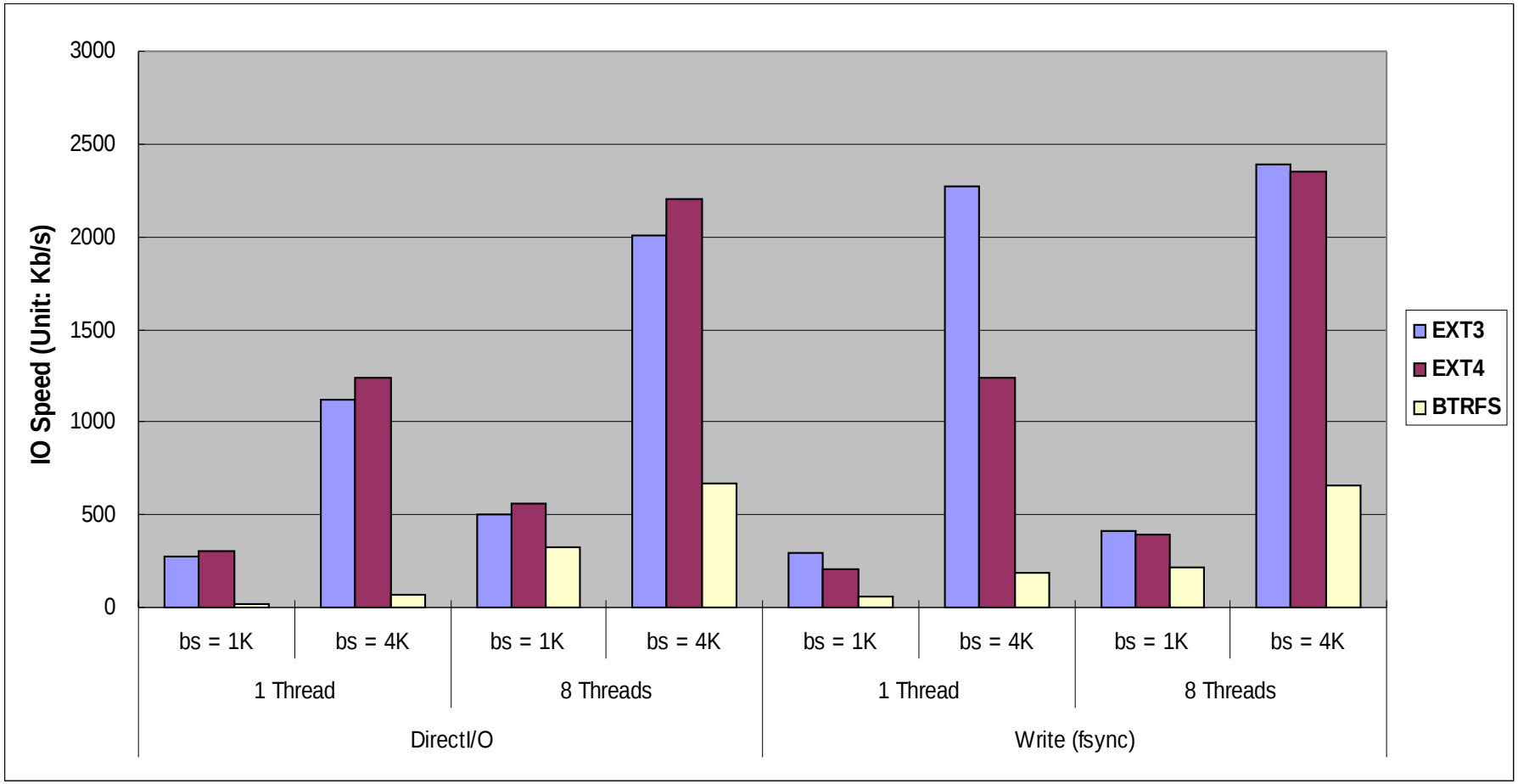
* Block size (bs): read or write BYTES bytes at a time.

Small file random read performance

FUJITSU

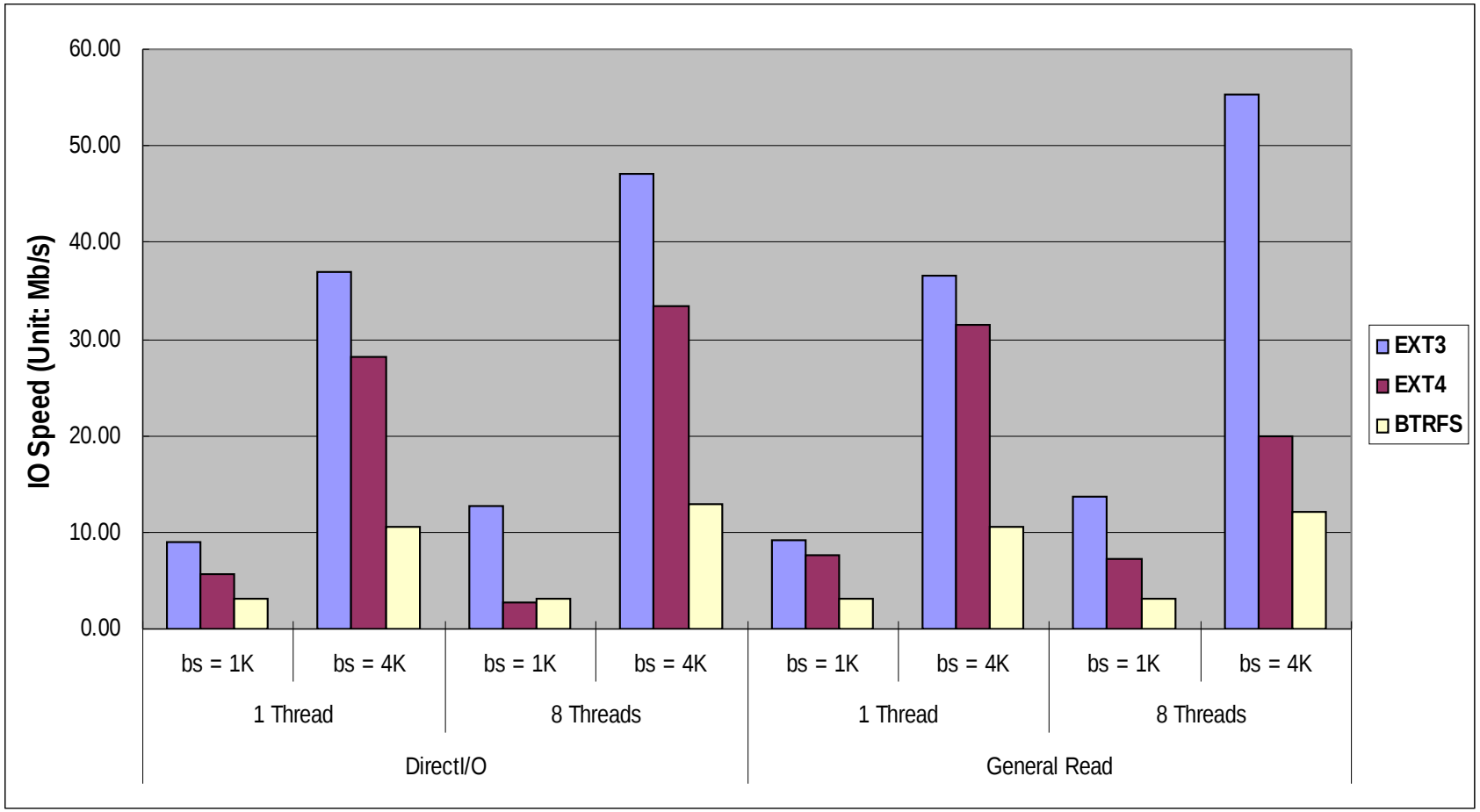


Small file random write performance



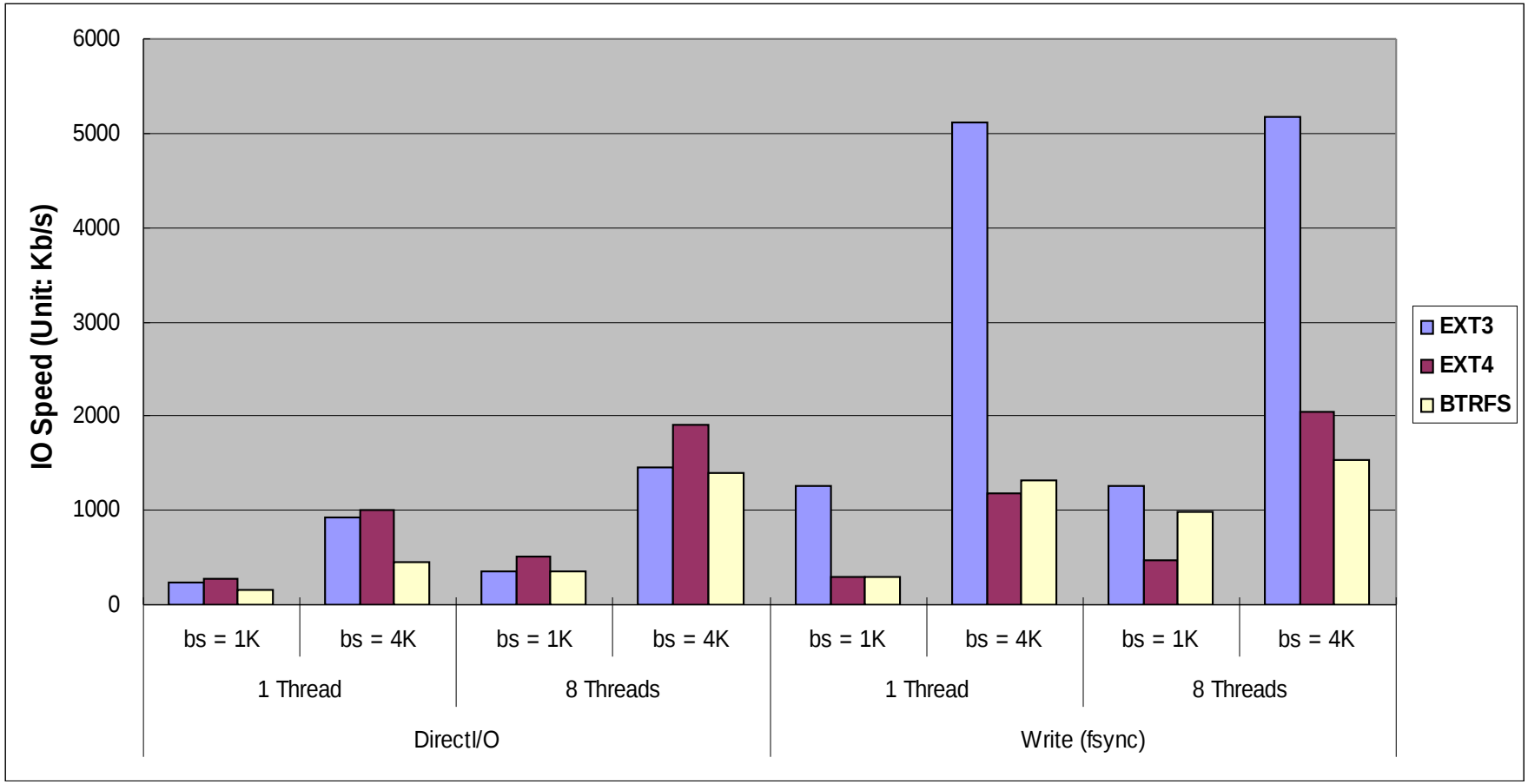
Write (fsync): write data into the file, and do fsync every 100 requests

Small file sequential read performance



Small file sequential write performance

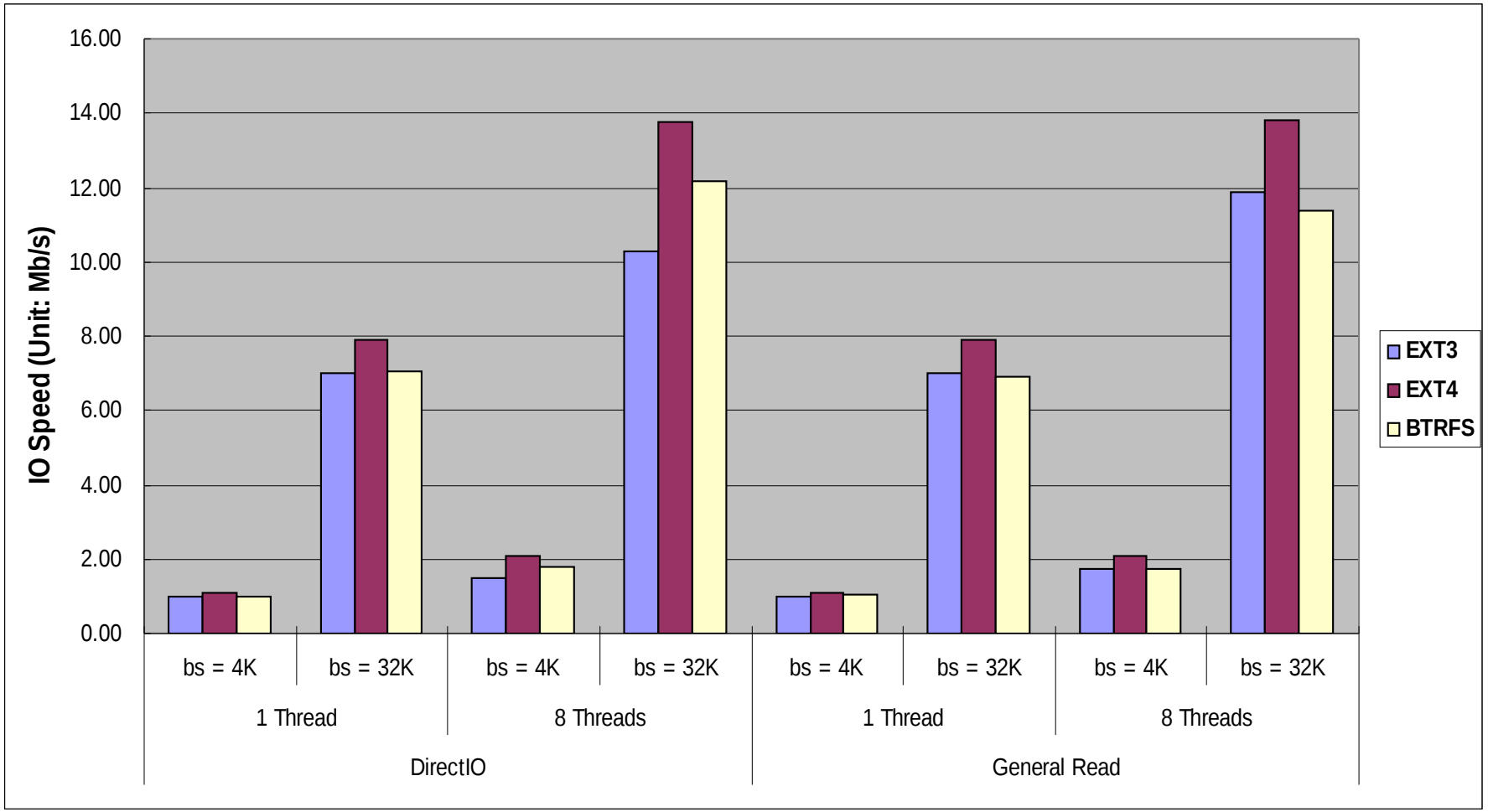
FUJITSU



Write (fsync): write data into the file, and do fsync every 100 requests

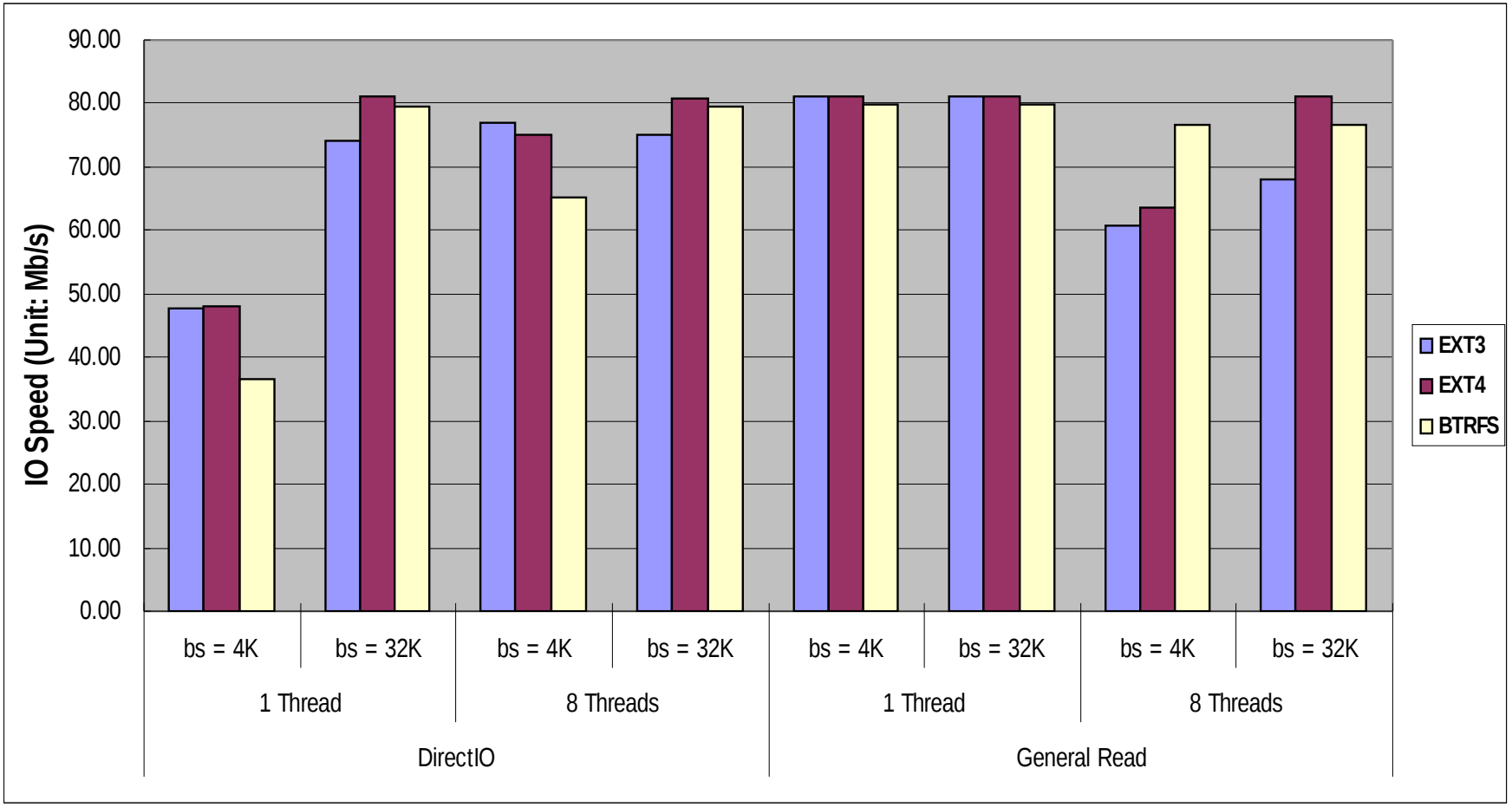
Large file random read performance

FUJITSU



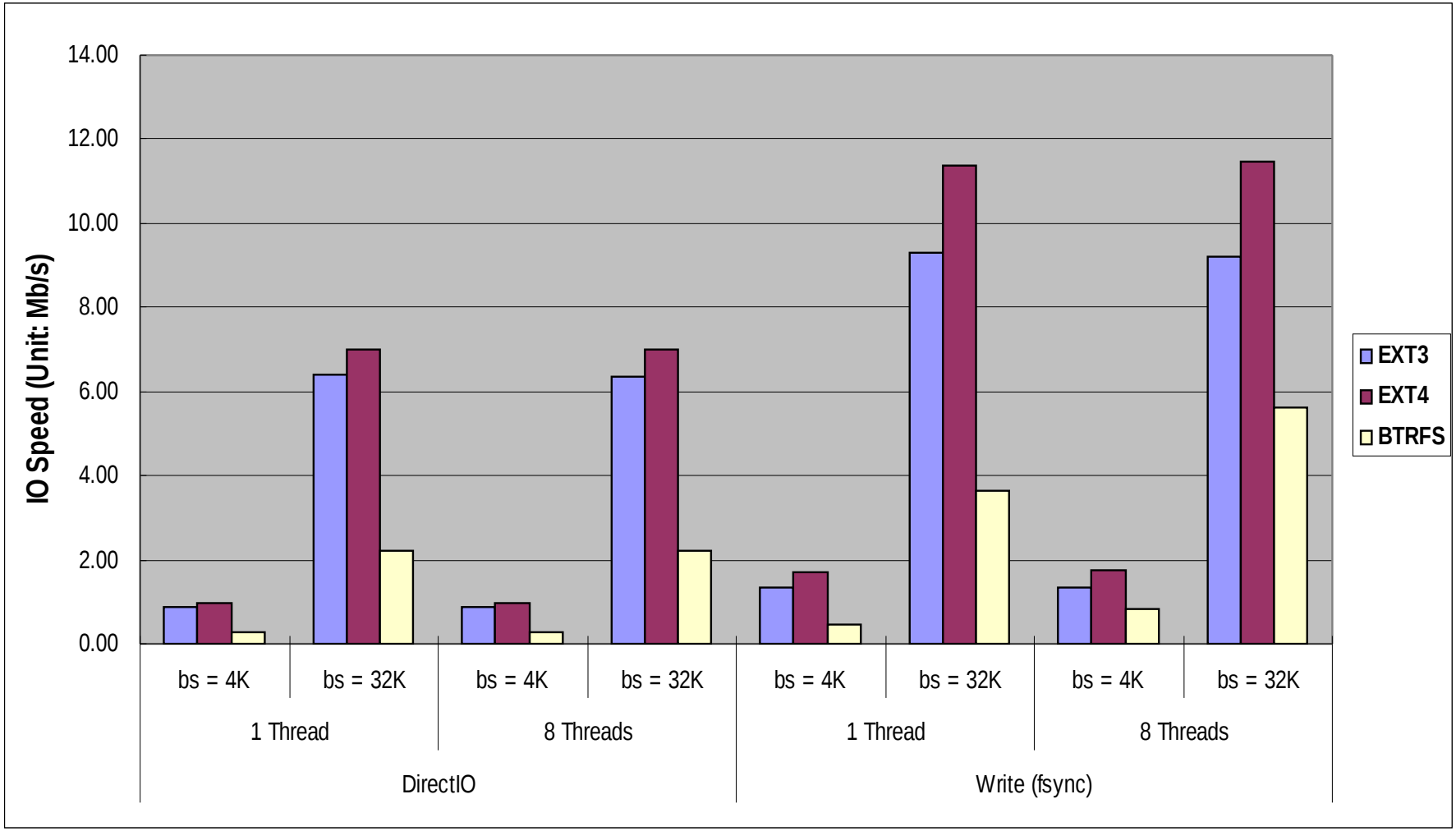
Large file sequential read performance

FUJITSU



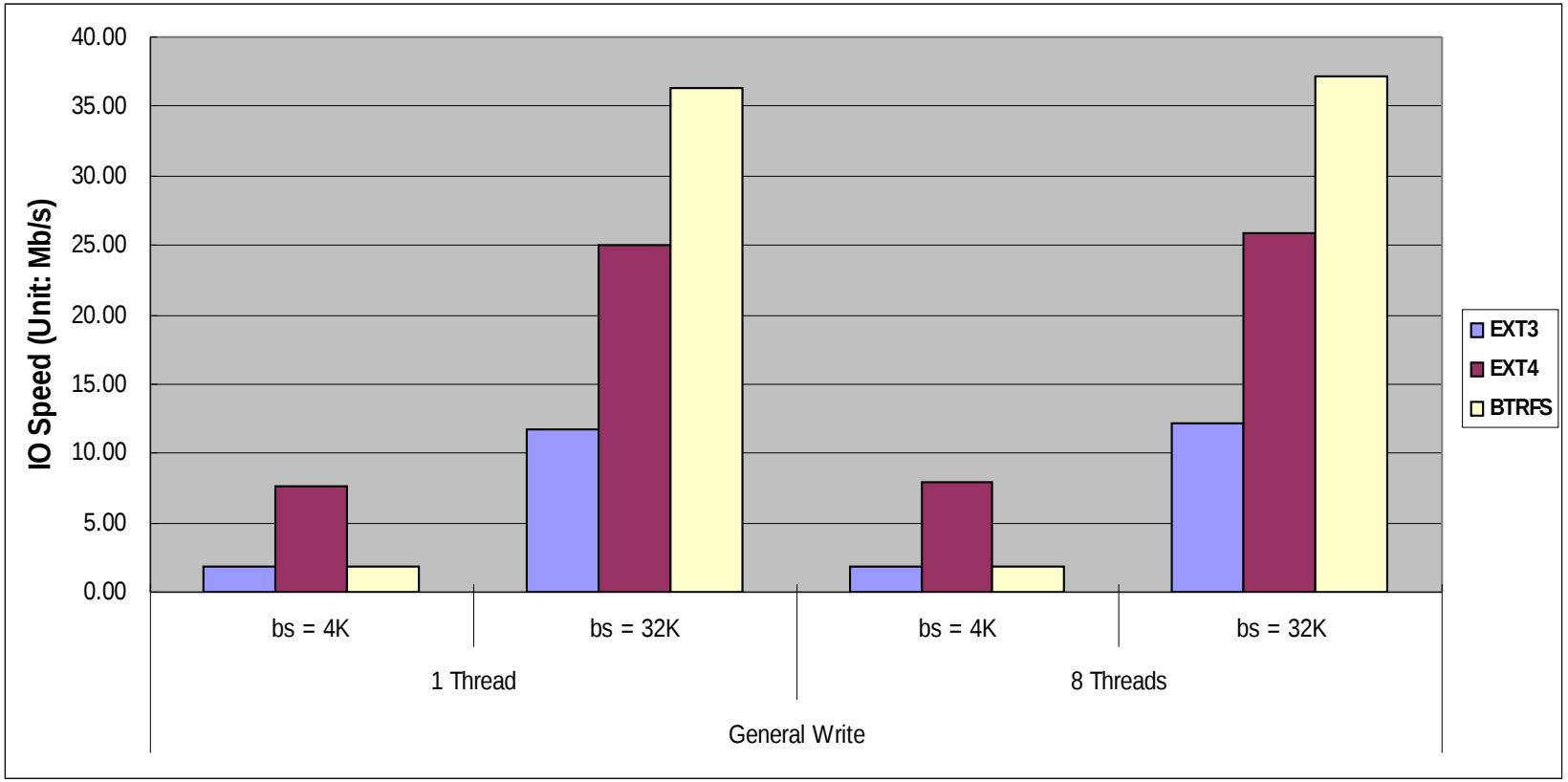
Large file random write performance (1/2)

FUJITSU



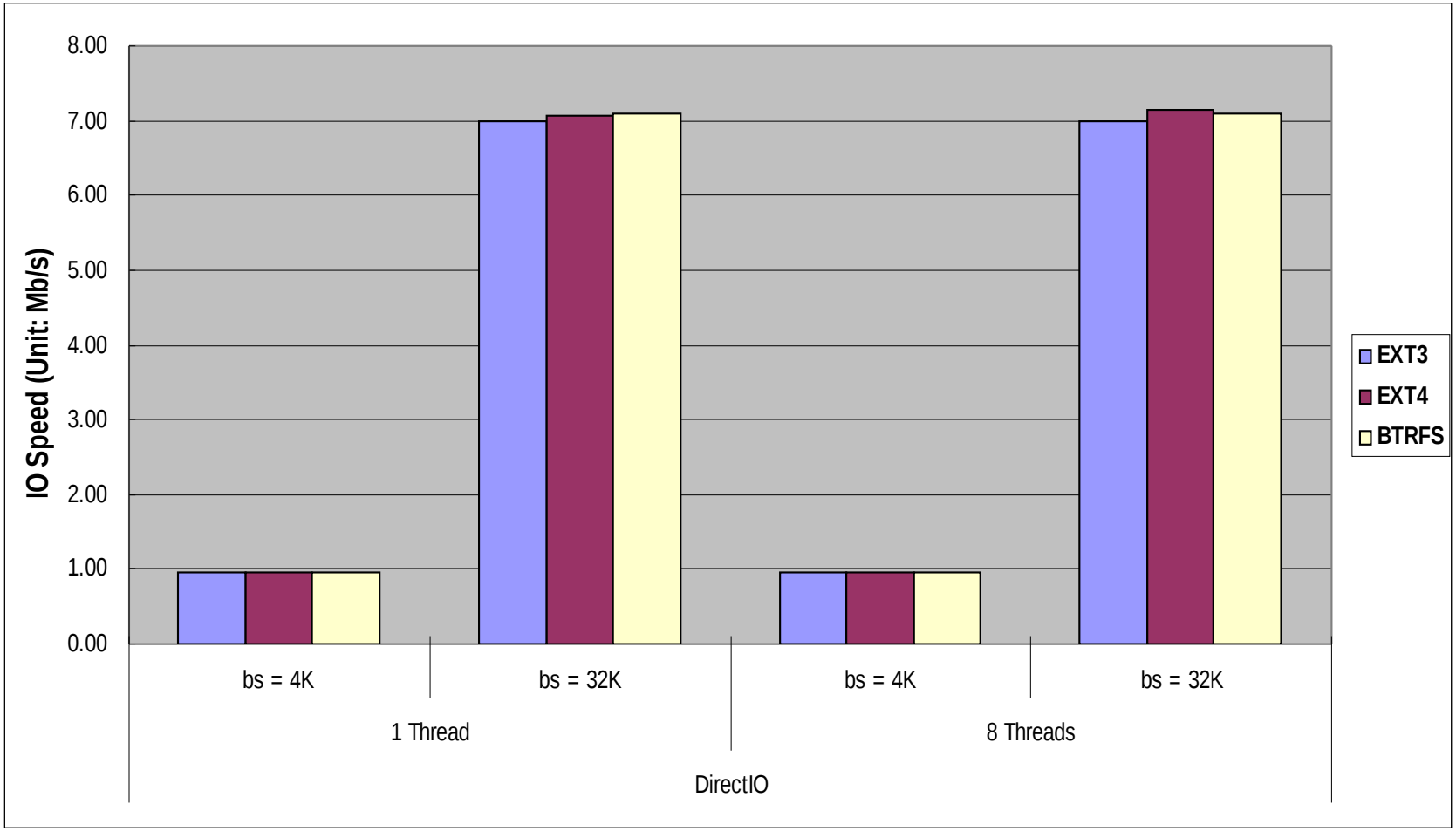
Large file random write performance
(2/2)

FUJITSU



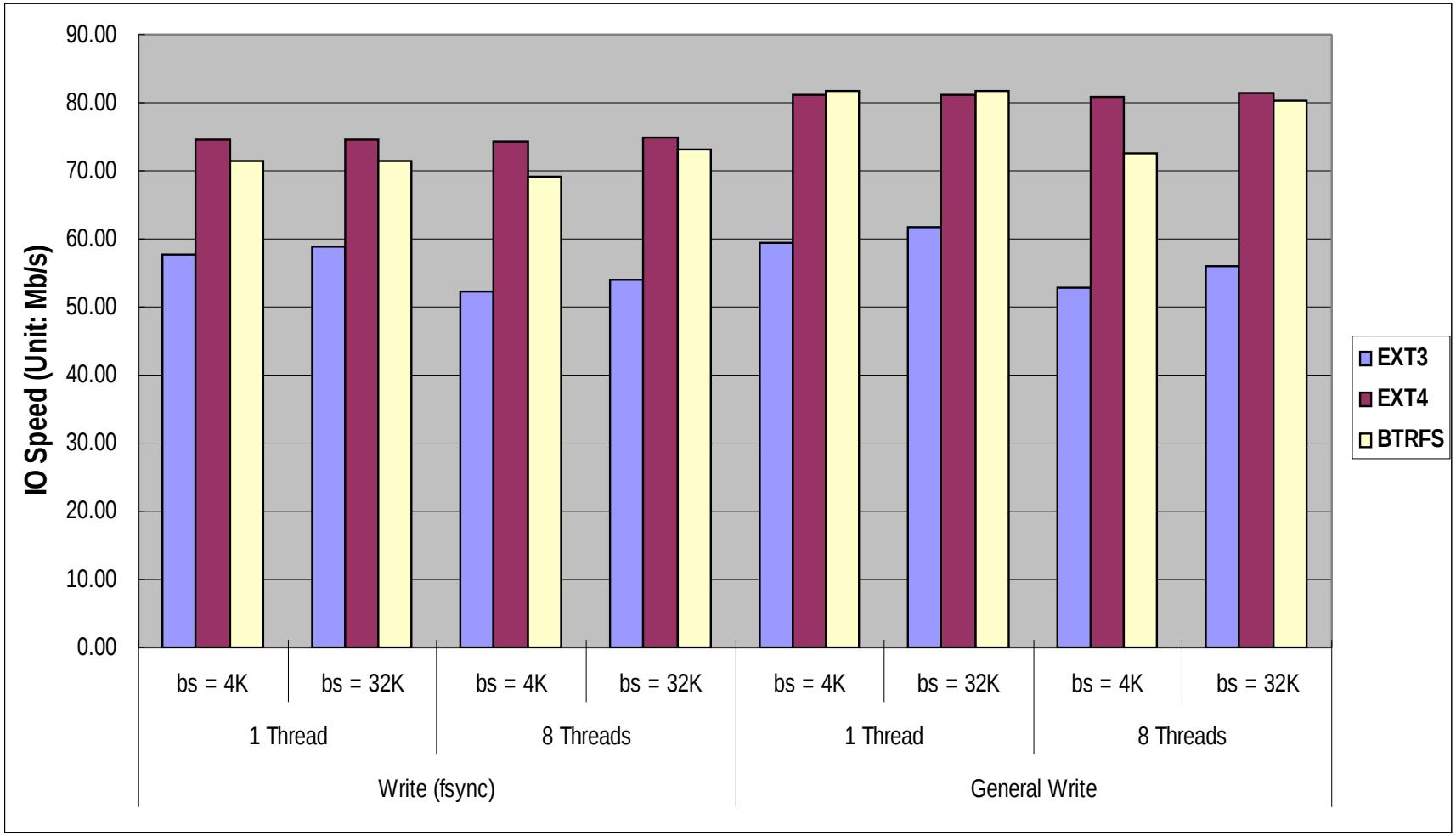
Large file sequential write performance (1/2)

FUJITSU



Large file sequential write performance (2/2)

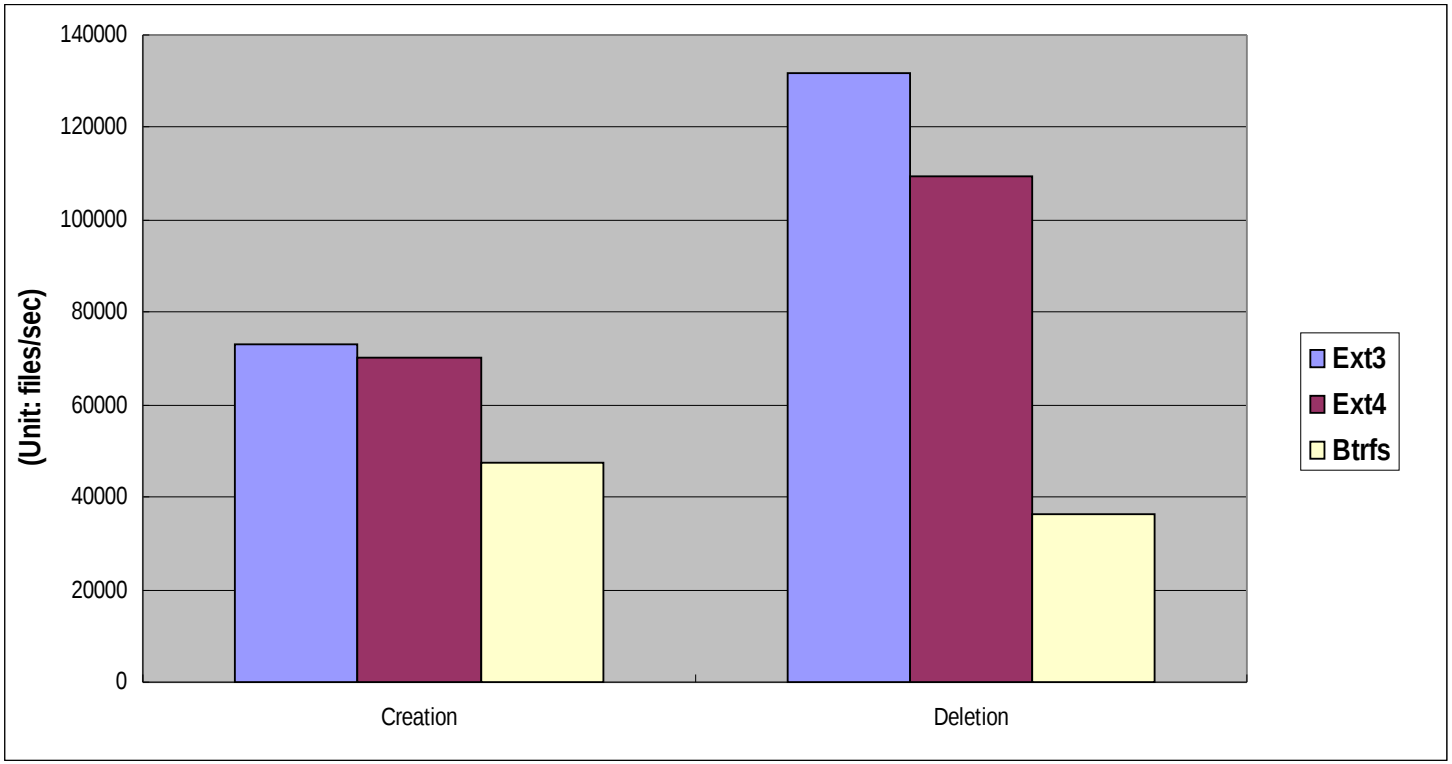
FUJITSU



File creation/deletion performance

FUJITSU

- Create/delete lots of empty files to measure the speed of file creation and deletion.



Comparison between Btrfs and Ext3/4

FUJITSU

- The performance of Btrfs is quite poor in the following cases (> 20% lower than Ext3/4)
  - Small file random read (Not inline file)
  - Small file sequential read
  - Small file random/sequential write
  - Large file random write (Direct I/O and fsync)
  - Large file random write (general write, bs = 4Kb)
  - File creation and deletion

Agenda

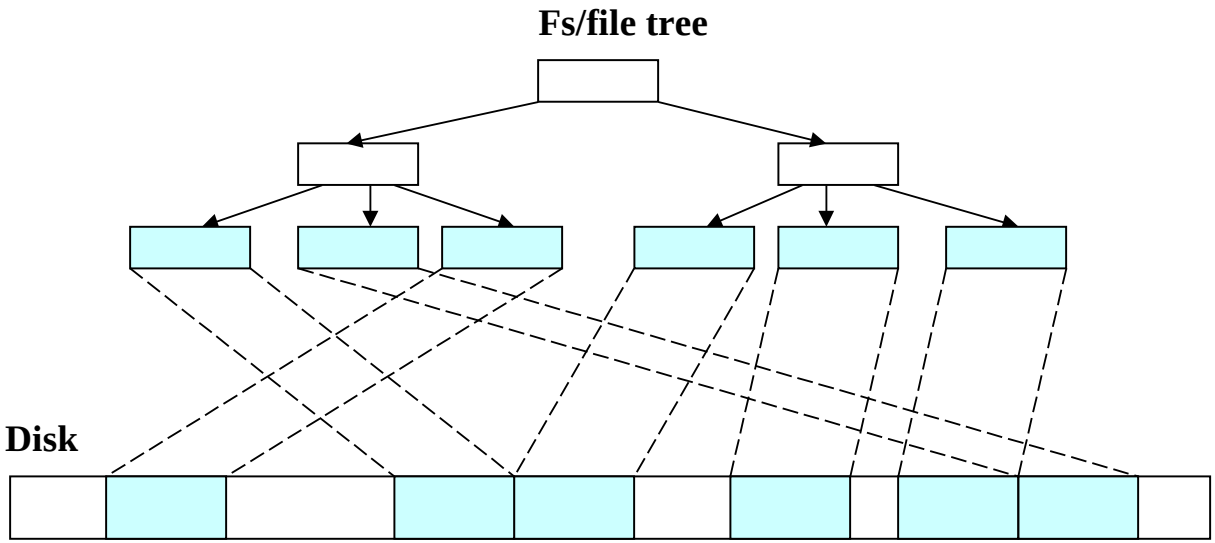
FUJITSU

- Comparison between Btrfs and Ext3/4
- Issue analysis (We have investigated)
  - Small file sequential read
  - Large file random write (Direct I/O and fsync)
  - File creation/deletion
- Future work

Small file sequential read

FUJITSU

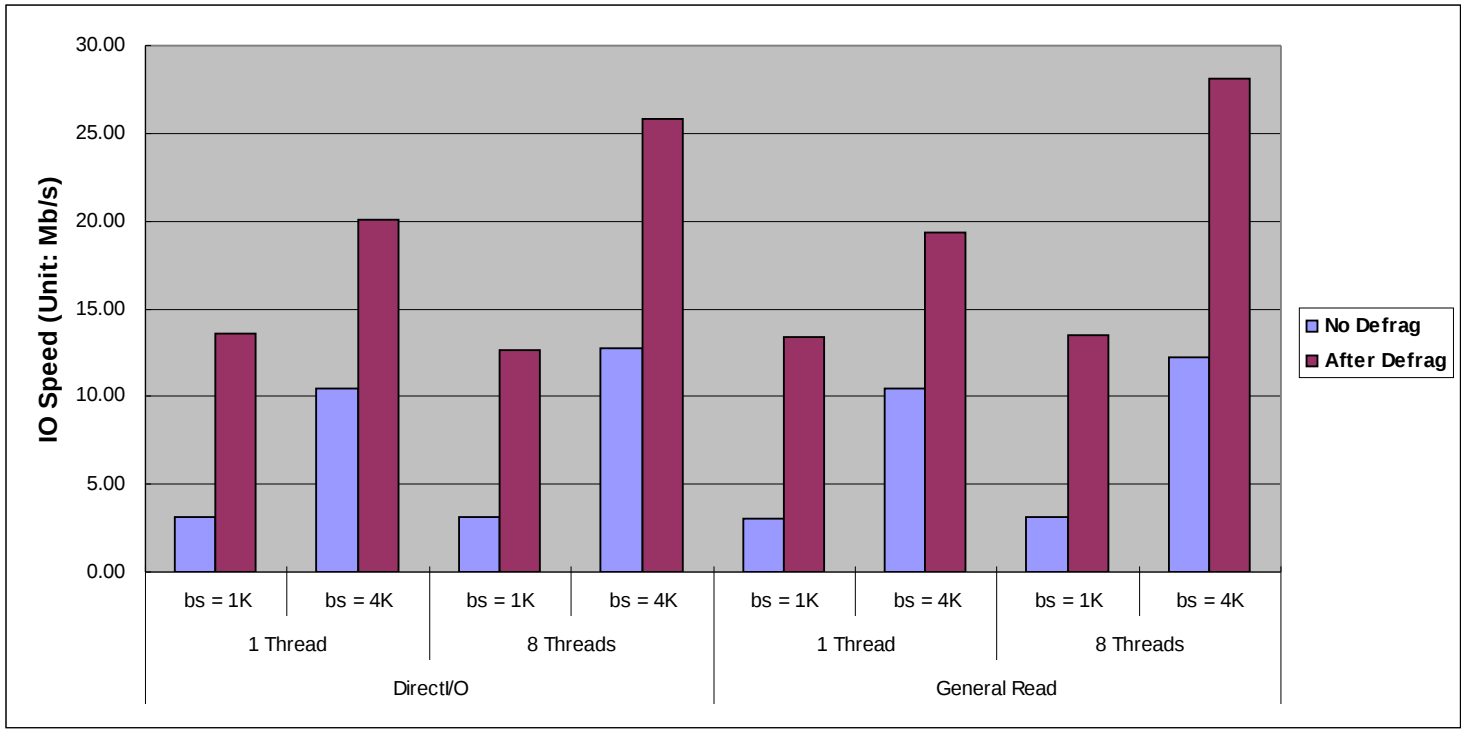
- Reasons
  - Metadata fragment -> The file extent reading latency -> The delay of file data reading
    - Btrfs must read file extent before reading file data (no matter the small file is inlined or not), but the disk has to reposition the reading offset frequently because of the fragment, and the readahead function can’t work well. So …



Small file sequential read

FUJITSU

- Reason verification
  - Do small file sequential read after defragment



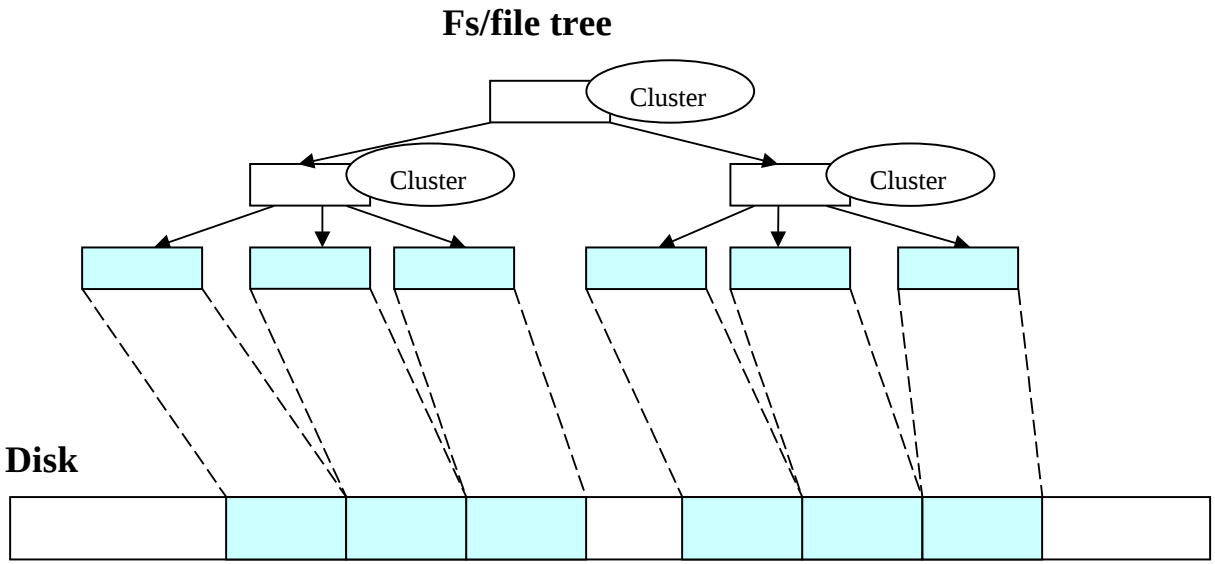
Small file sequential read

FUJITSU

Solution

- Pre-allocation for b+ tree: Introduce free space clusters for each node in the tree, then we can allocate contiguous free space from the parent node's cluster to store the sibling leaves closely

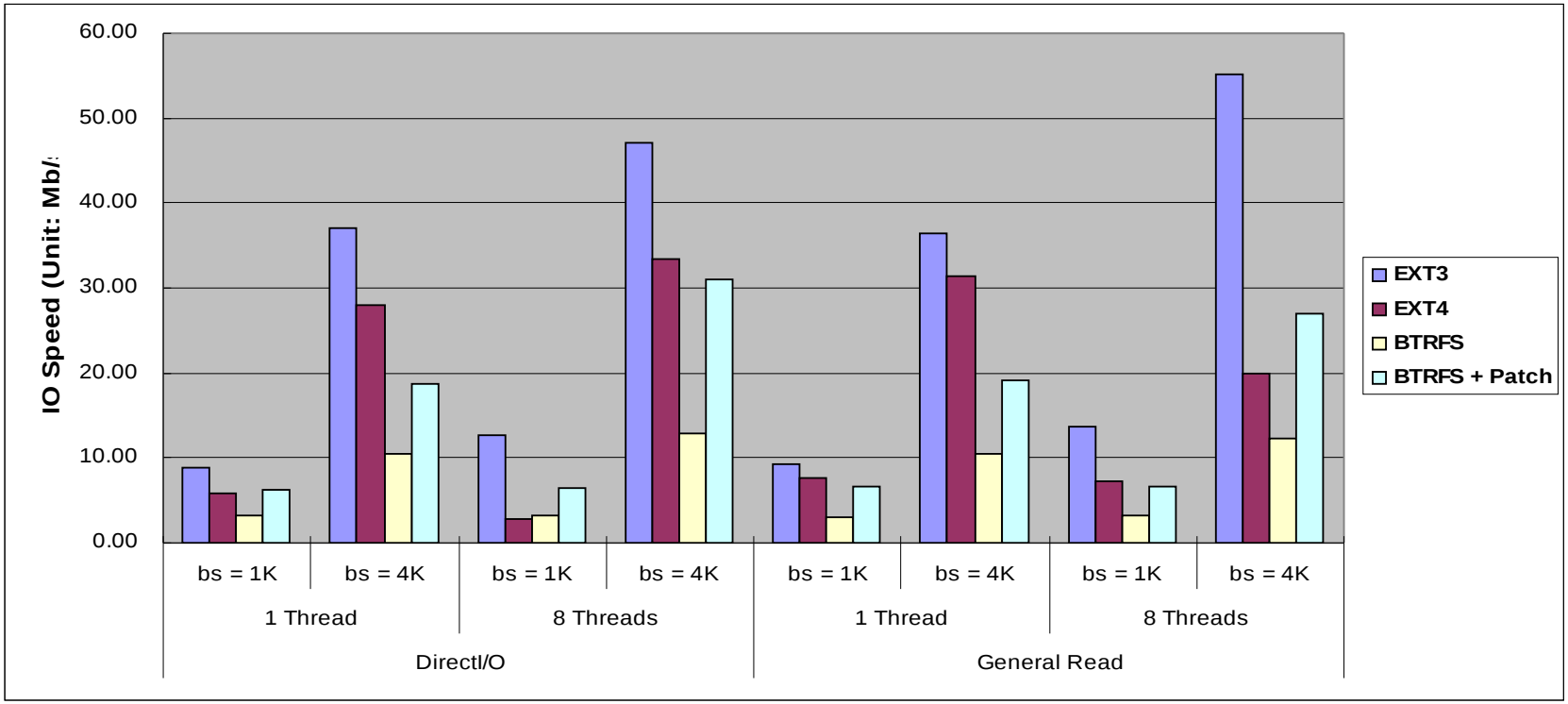
  (The patch of this solution is still under test, hasn't be posted)



Small file sequential read

FUJITSU

- Improvement result



- Further Improvement
  - Introduce the auto defragment for metadata
  - Apply the new metadata readahead API written by Arne

Agenda

FUJITSU

- Comparison between Btrfs and Ext3/4
- Issue analysis (We have investigated)
  - Small file sequential read
  - Large file random write (Direct I/O and fsync)
  - File creation/deletion
- Future work

Large file random write (Direct IO and fsync)

FUJITSU

- Background – What is tree logging?

  Tree logging is a special write ahead log of dirty metadata.

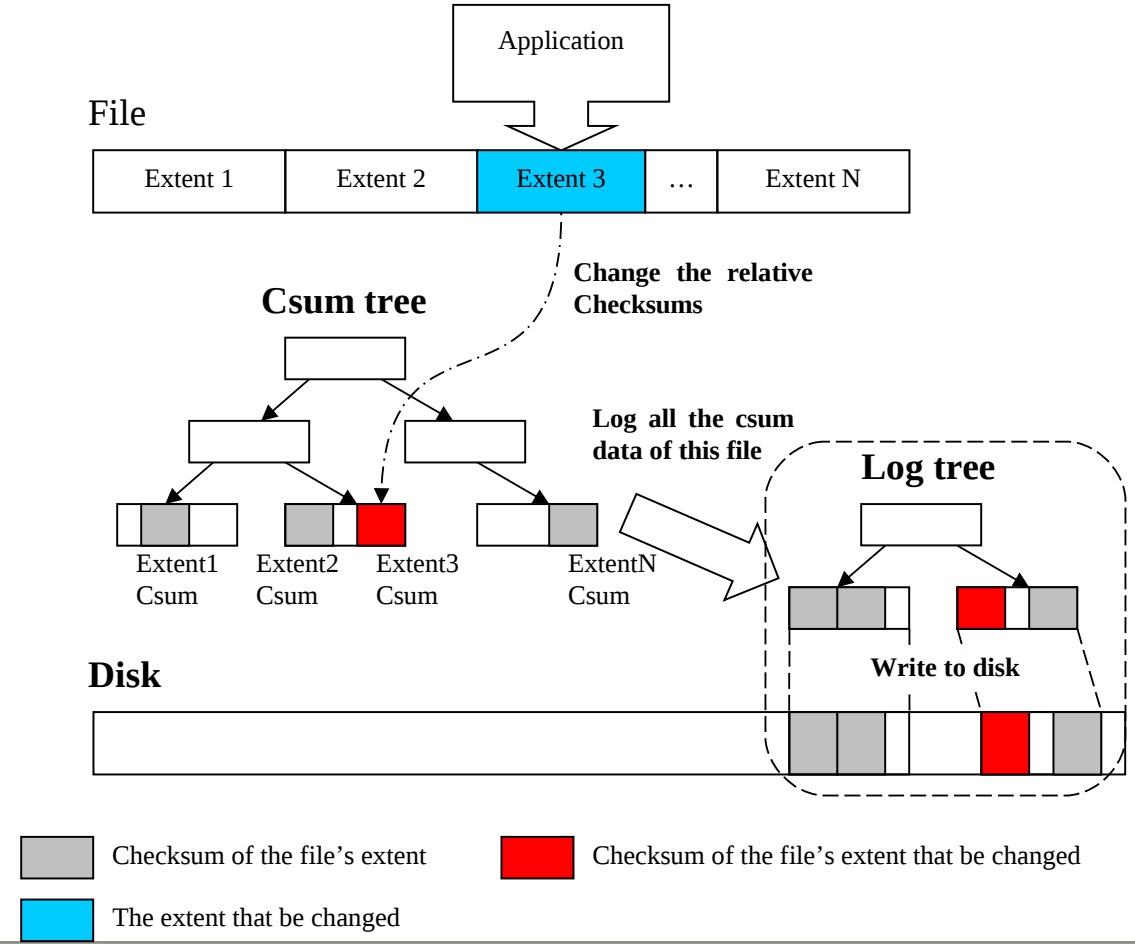
  - Purpose: Reduce the write requests of the metadata when fsyncs and O_SYNCs happen.

  - Implementation: Copy the changed items into a special tree (log tree, one per fs/file tree), and then write that tree to disk. After a crash, Btrfs recover the fs/file tree by that tree.

Large file random write (Direct IO and fsync)

FUJITSU

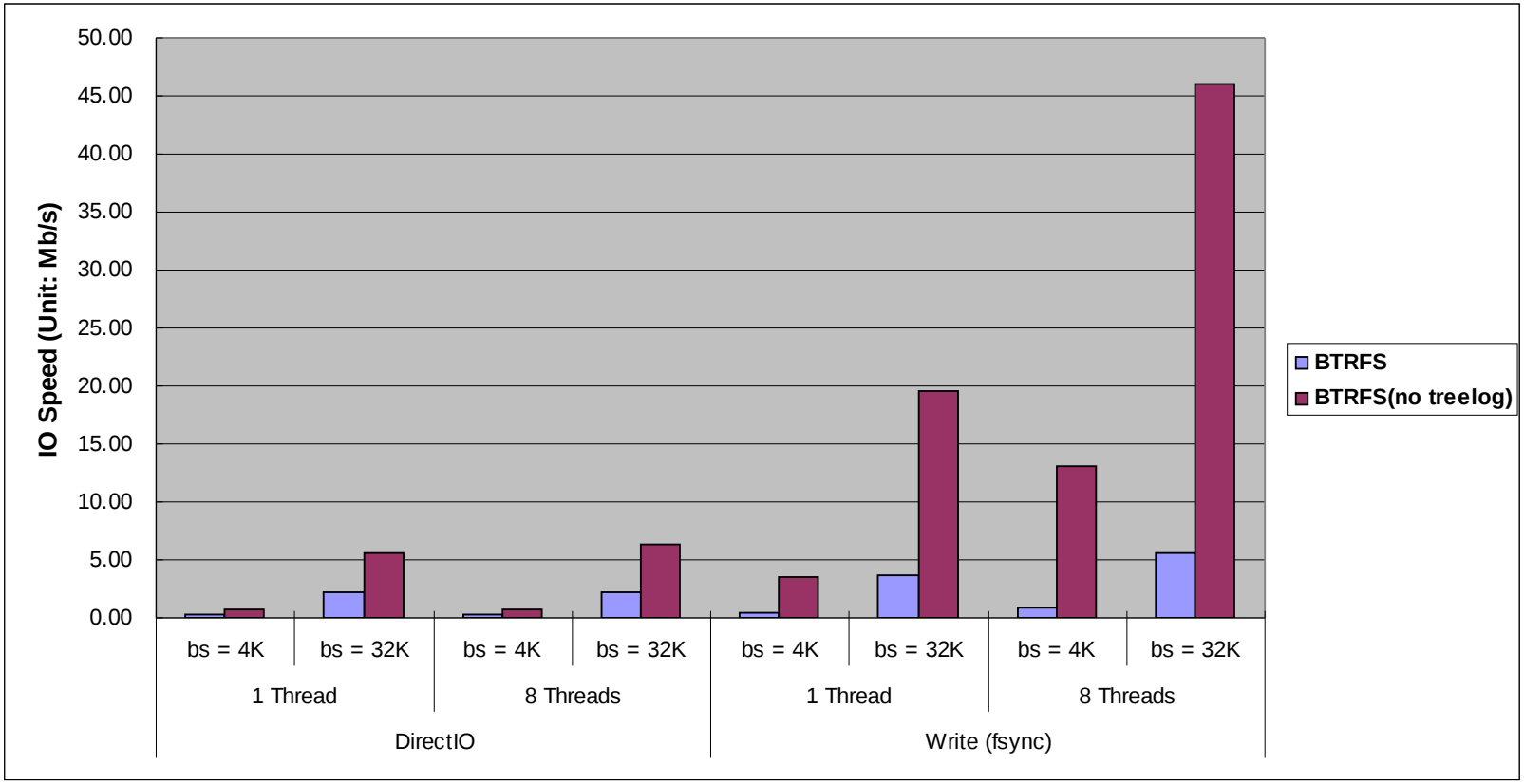
- Reasons
  - Log lots of unchanged metadata (Ex. Csum, File extent)



Large file random write (Direct IO and fsync)

FUJITSU

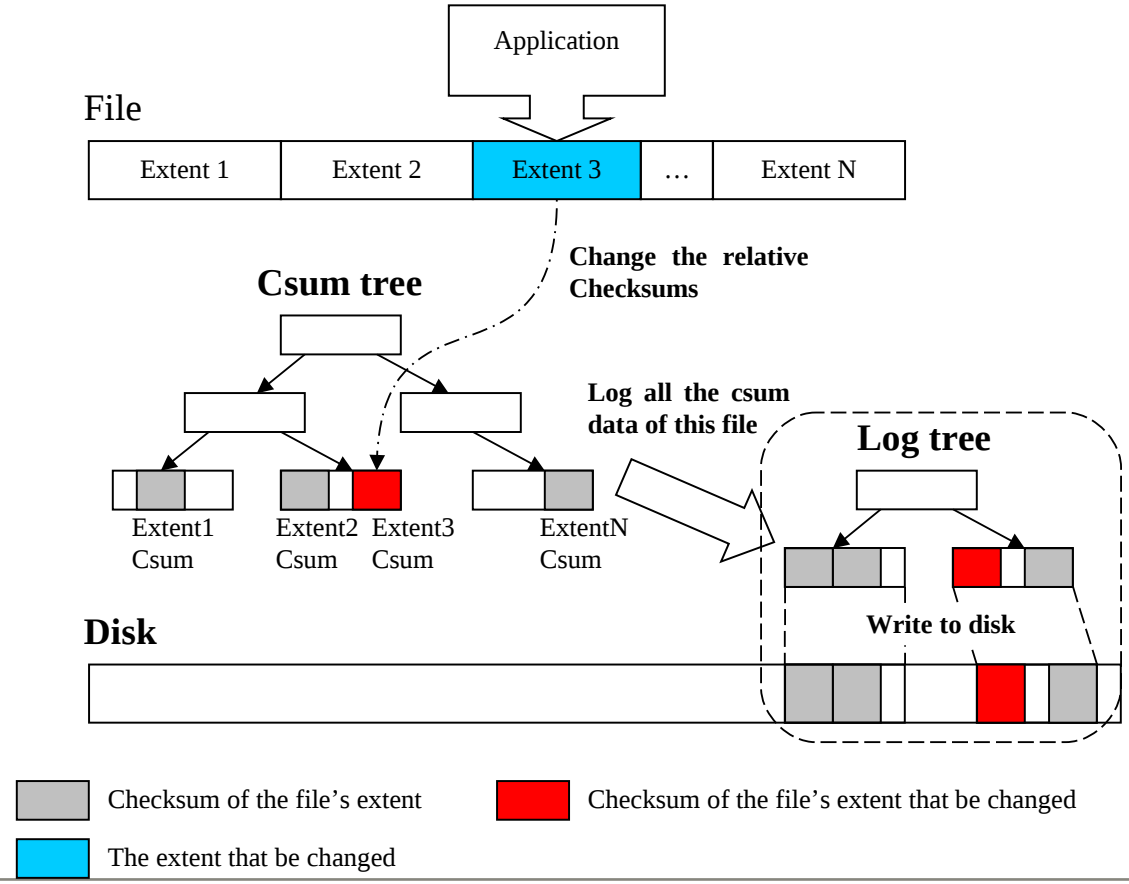
- Reason verification
  - Do large file random write test after closing tree log function (mount with -o notreelog)



Large file random write (Direct IO and fsync)

FUJITSU

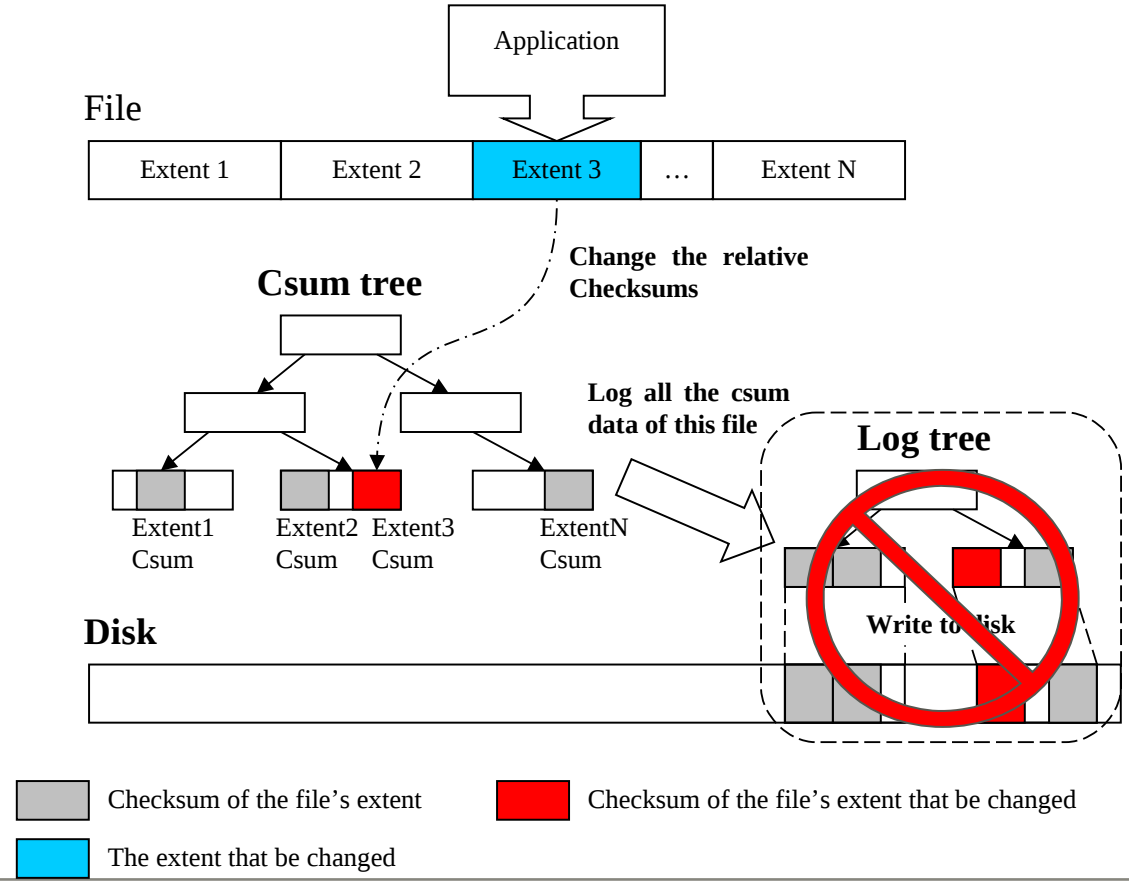
- Solution
  - Don’t log unchanged metadata: Introduce sub-transaction id to filter the unchanged metadata (v2.6.41)



Large file random write (Direct IO and fsync)

FUJITSU

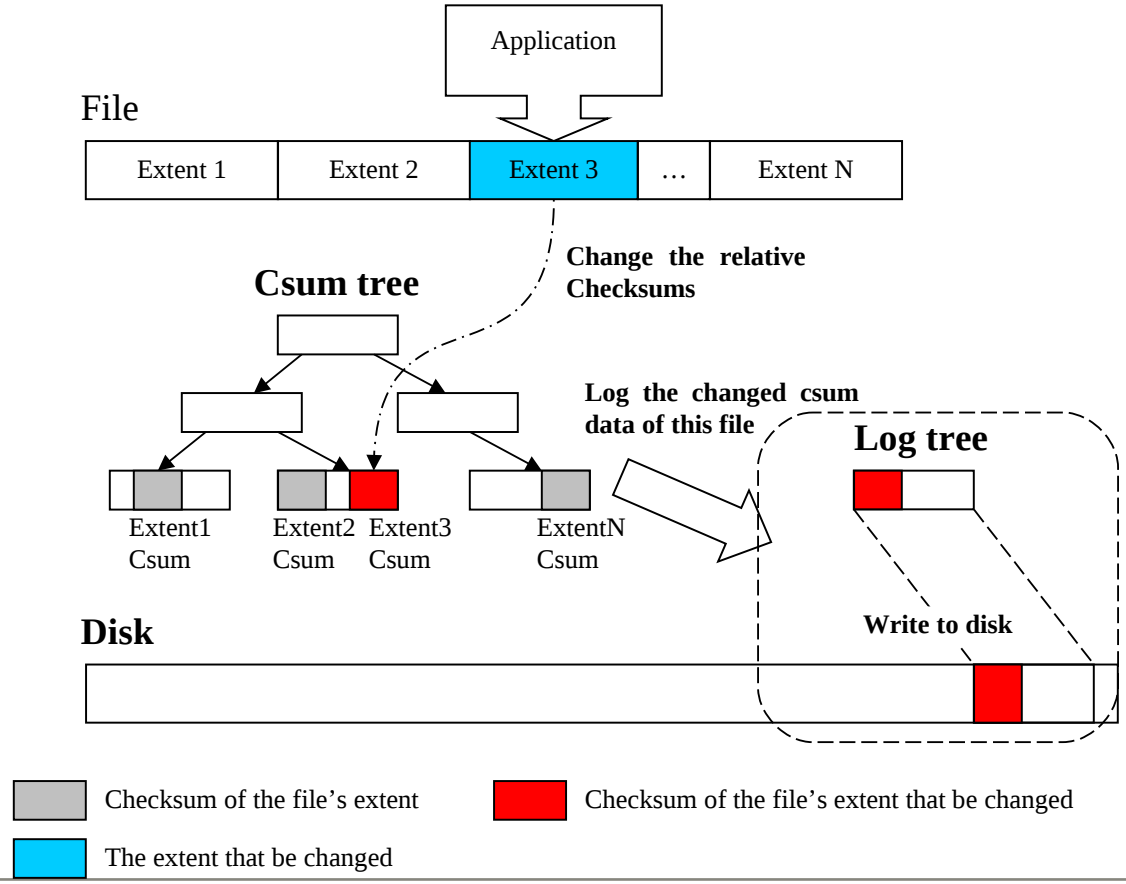
- Solution
  - Don’t log unchanged metadata: Introduce sub-transaction id to filter the unchanged metadata (v2.6.41)



Large file random write (Direct IO and fsync)

FUJITSU

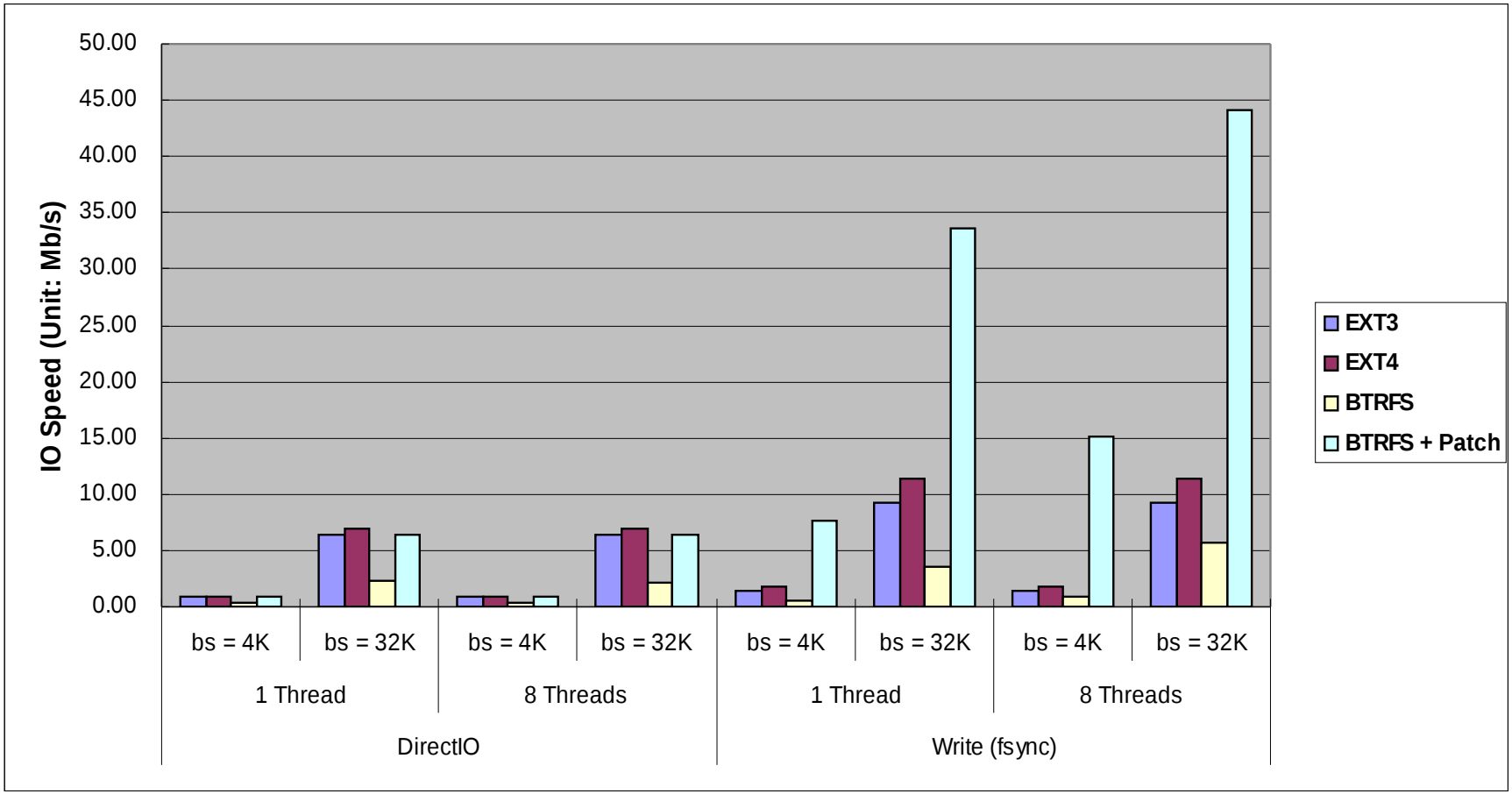
- Solution
  - Don’t log unchanged metadata: Introduce sub-transaction id to filter the unchanged metadata (v2.6.41)



Large file random write (Direct IO and fsync)

FUJITSU

- Improvement result



Agenda

FUJITSU

- Comparison between Btrfs and Ext3/4
- Issue analysis (We have investigated)
  - Small file sequential read
  - Large file random write (Direct I/O and fsync)
  - File creation/deletion
- Future work

File creation/deletion

FUJITSU

- Reasons
  - Btrfs does more metadata insertion and deletion.

	Btrfs	Ext4(Not Sure)
File Creation	inode name back reference ACL directory item directory name index	inode ACL directory entry
File Deletion	inode inode back reference ACL directory item directory name index logged directory item logged directory name index	inode ACL directory entry

  - Btrfs must search the b+ tree to look up the place, where the inode will be stored, when updating inode.

    (Time complexity: O(log(n)), But Ext3/4 is O(1))
  - Searching nodes/leaves in the rb-tree spends lots of time

File creation/deletion

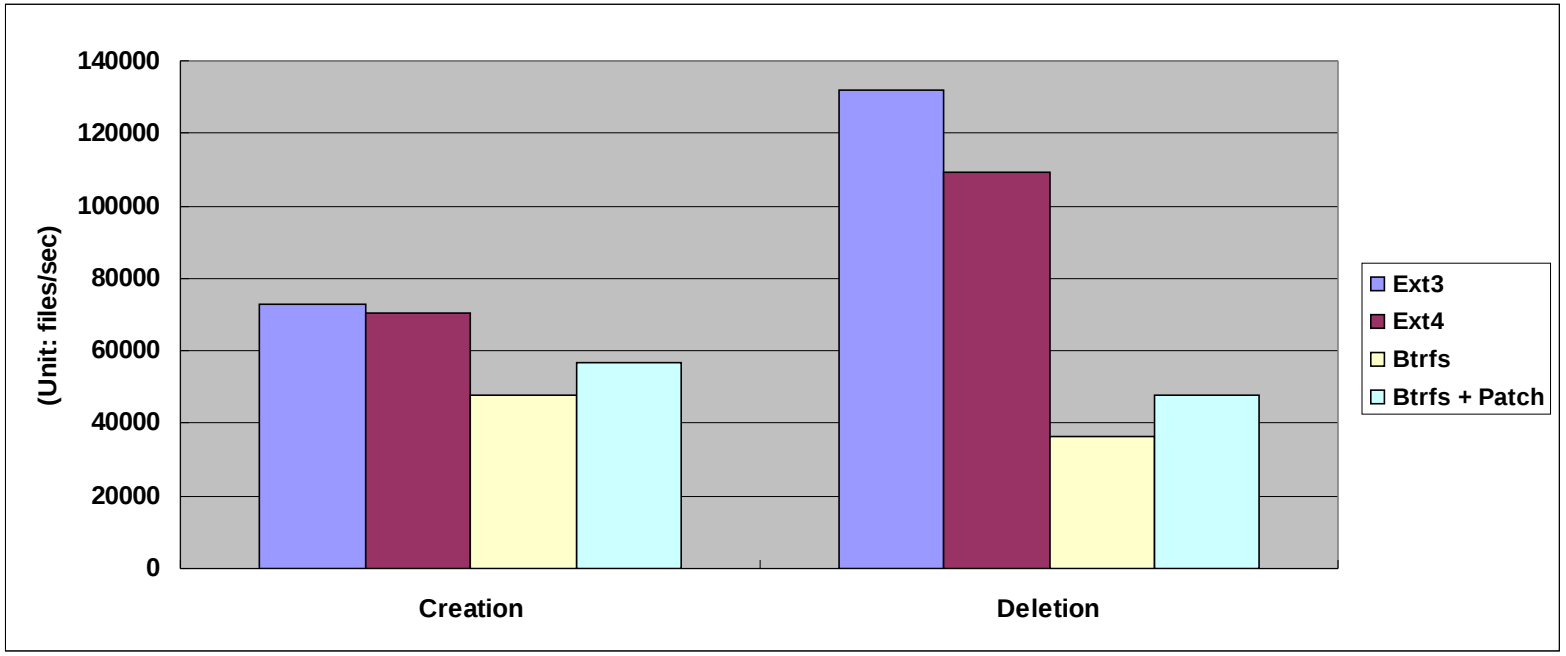
FUJITSU

- Solution
  - Batch operation -- Insert/delete a batch of the directory name indexes (v2.6.40)
  - Delay operation -- Delay to update the inode information in the b+ tree (v2.6.40)
  - Using radix tree instead of rb-tree (v2.6.37)

File creation/deletion

FUJITSU

- Improvement result
  - Create/delete lots of empty files to measure the speed of file creation and deletion.



Agenda

FUJITSU

- Comparison between Btrfs and Ext3/4
- Issue analysis (We have investigated)
  - Small file sequential read
  - Large file random write (Direct I/O and fsync)
  - File creation/deletion
- Future work

Future work

FUJITSU

- Improve small file sequential read performance further
- Improve small file random read performance (Not inline file)
- Improve small file random/sequential write performance
- Do other benchmarks and improve bad cases

FUJITSU

Q/A