EXT4

Theodore Ts'o

THE LINUX FOUNDATION

What's Good About Ext3

- **Most widely used filesystem for Linux**
- **Extremely diverse developer community**
  - Good in these days of the economic downturn
  - Distribution expertise needed if they are to feel comfortable supporting the filesystem

What's Not So Good About Ext3

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

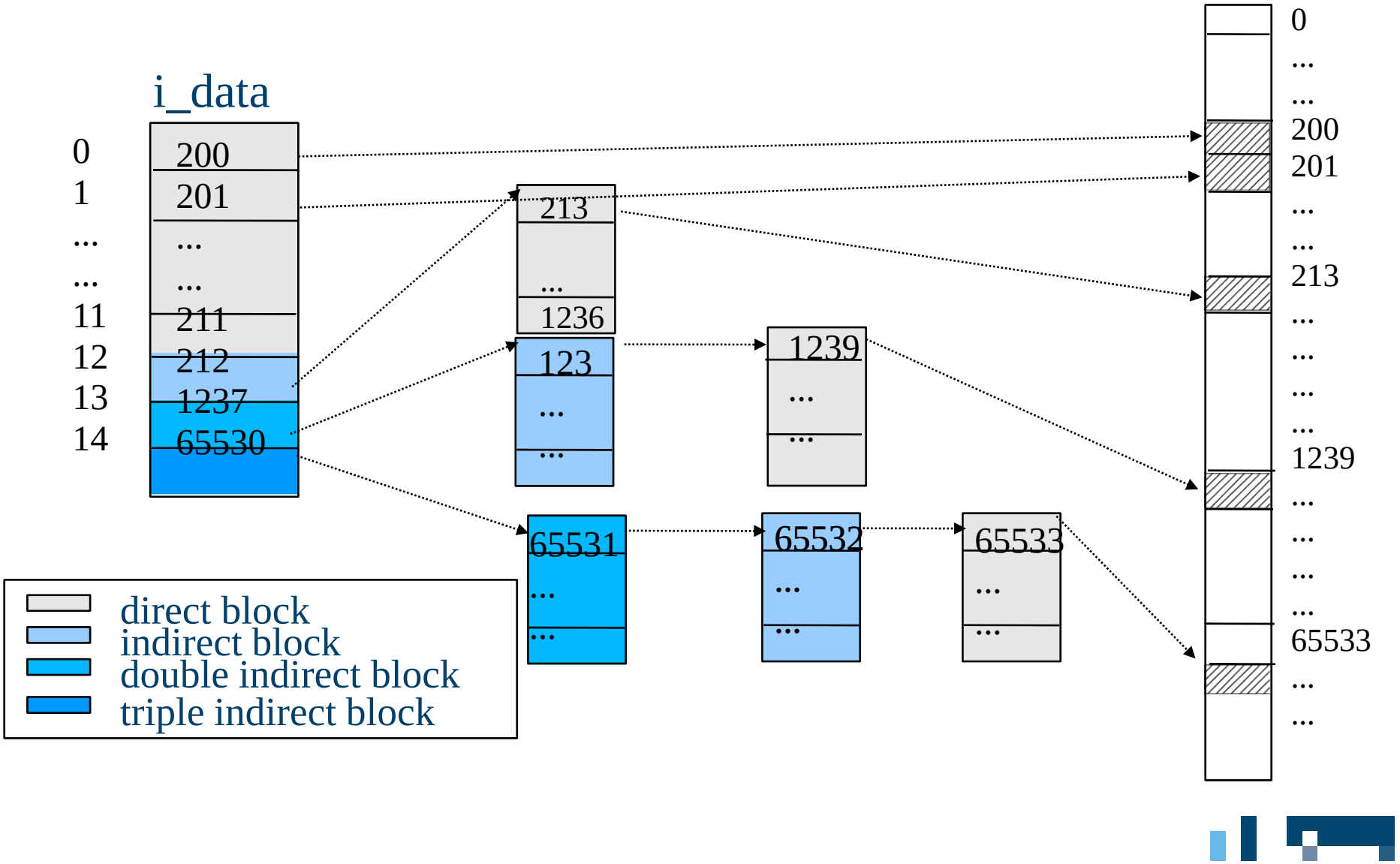
Is Ext4 really a new filesystem?

- **“Ext” in ext2/3/4 stands for “extended”**
- **The ext4 filesystem driver supports new filesystem features that together makes up what most people consider “ext4”**
  - Just as ext3 supports ext2 with some new features, such as “has_journal” and “dir_index”
  - Ext4 (as of 2.6.29) can even support filesystems that do not have the “has_journal” feature flag
- **Ext4 fork in 2.6.19 was to make sure that the large ext3 user community would not be affected while ext4 was under development**
  - Allowed more experimentation than work was done under ext4.
- **Renamed from ext4dev to ext4 in 2.6.28**

What's New in Ext4?

- **Many new features:**
  - Use of extents instead of indirect blocks
  - Delayed Allocation
  - Multiblock Allocation
  - Persistent Allocation
  - Subsecond timestamps
  - NFSv4 version id's for caching
  - Greater than 32000 subdirectories
  - Journal and group descriptor checksums
  - “Huge files” 16TB files on 4k block filesystems
  - ATA TRIM support
  - More intelligent metadata layout
- **The most important new feature is Extents**

Ext2/Ext3 Indirect Block Map



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
- **Extents is an efficient way to represent large file**
- **An extent is a single descriptor for a range of contiguous blocks**

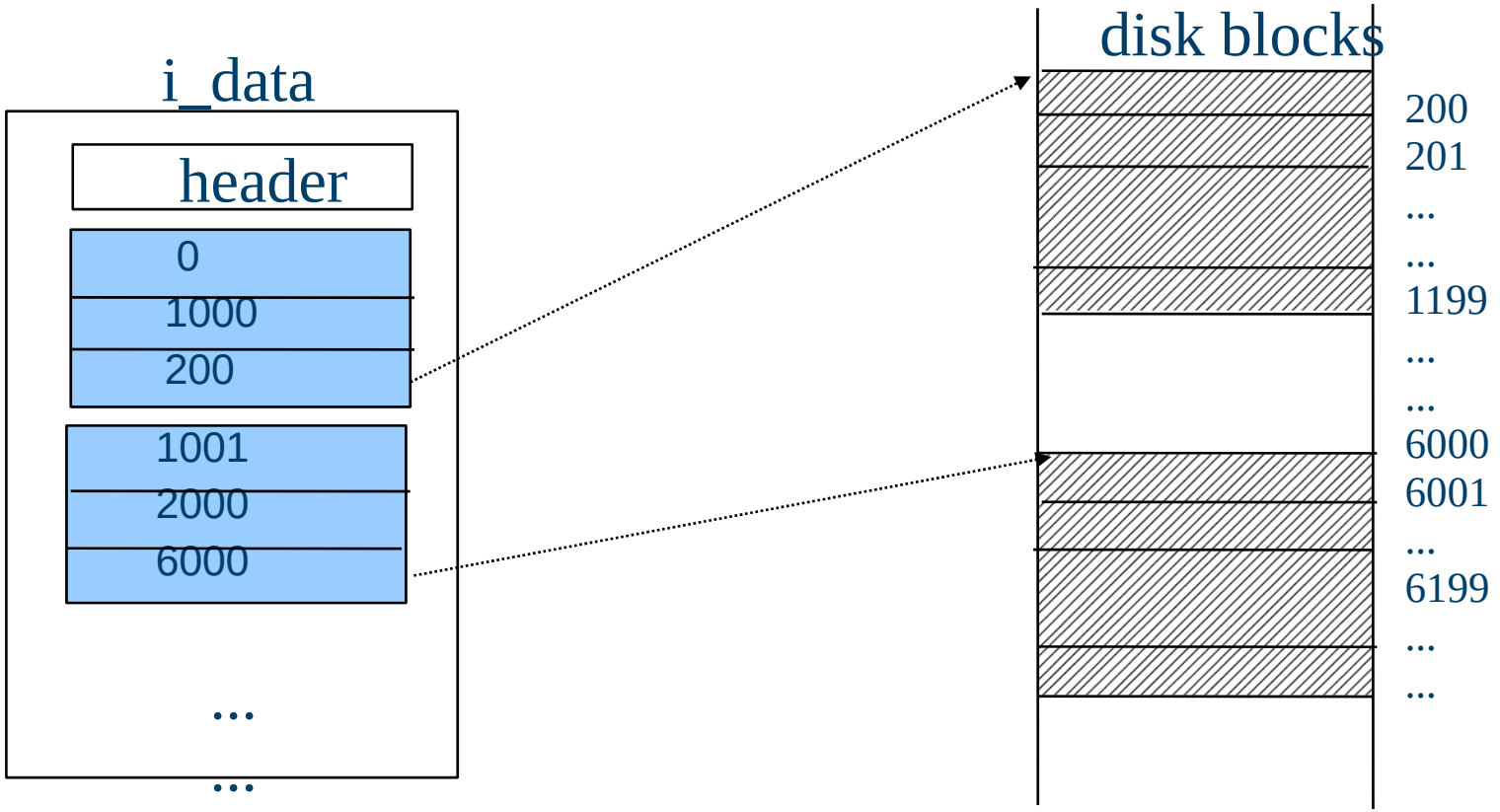
logical	length	physical
0	1000	200

On-disk Extents Format

- **12 bytes ext4_extent structure**
  - address 1EB filesystem (48 bit physical block number)
  - max extent 128MB (16 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 */
};
```

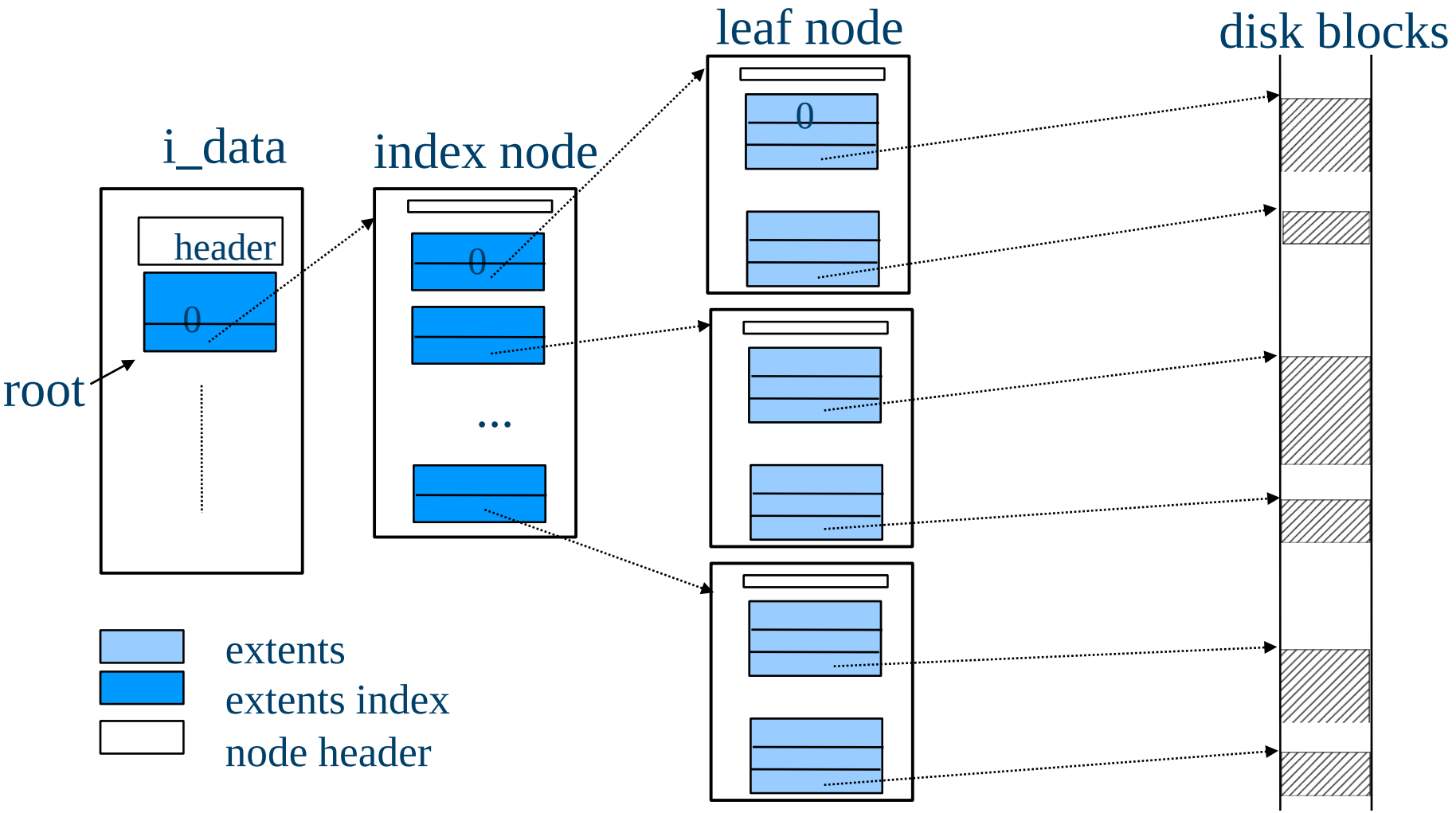
Ext4 Extent Map



Extents Tree

- **Up to 3 extents could stored in inode i_data body directly**
- **Convert to a B-Tree extents tree, for > 4 extents**
  - Tree root is stored in inode body (could be in EA or other block)
    - pointing to a index extents block
    - leaf extents block store extents (up to 340 extents)
  - extents look up
    - Leaf/Index extent block is sorted by logical block number
    - Binary search for extent lookup
  - extents insert
    - B-Tree split if leaf block is full
- **Last found extent is cached in-memory extents tree**

Ext4 Extent Tree



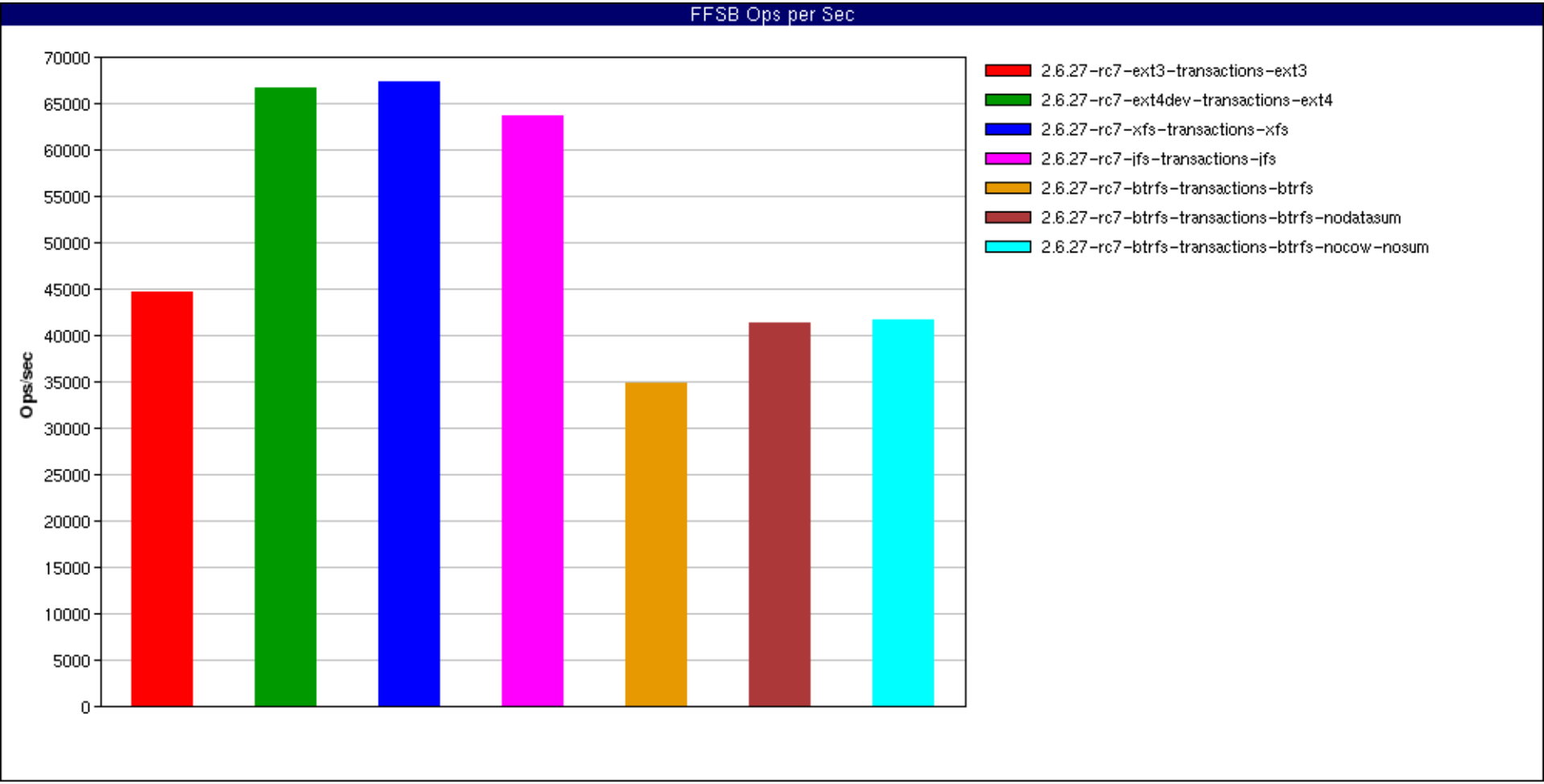
Block Allocator changes

- **Needed to best support extents**
  - Extents work best if files are contiguous
  - Delayed allocation and allocating multiple blocks at a time makes this much more likely
  - Responsible for most of ext4's performance improvements
- **Persistent preallocation allows blocks to be assigned to files without initializing first**
  - Most useful for databases and video files
  - Also useful for files that grow gradually via small append operations (i.e., Unix mail files and log files)
  - Can access via posix_fallocate(), but for some applications, direct access to the Linux system call in glibc is needed.
    - When you don't want to fall back to manual initialization
    - When you don't want to change i_size.

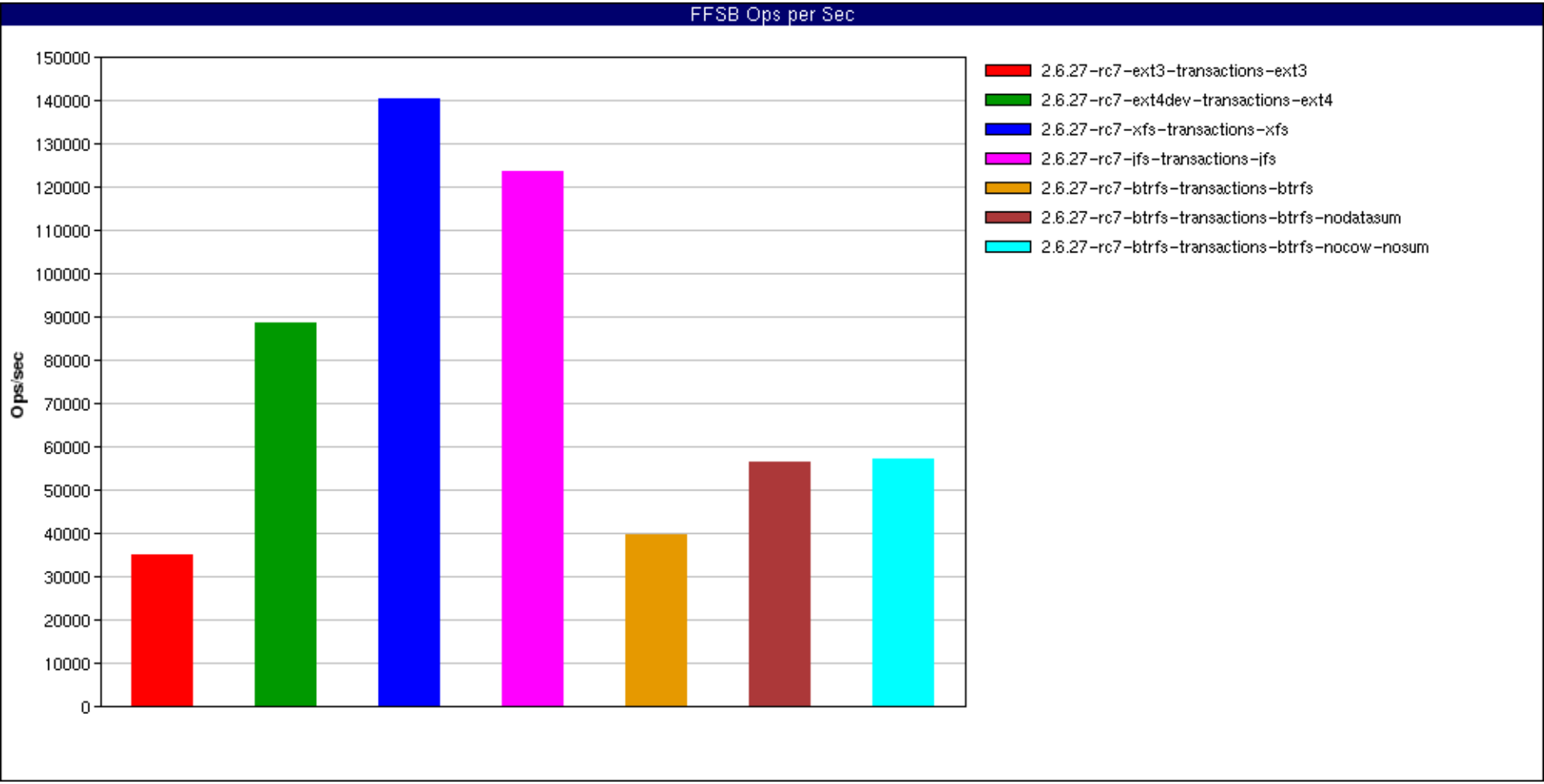
Some performance charts....

- **Lies, d*mn lies, and benchmarks...**
- **What to insist before you believe benchmarks**
  - Are the benchmarks “fair”?
  - Are the benchmarks “repeatable”?
  - Do the benchmarks fairly represent the workload that you care about?
- **One recent good effort: http://btrfs.boxacle.net**
  - Done by Steven Pratt (who happens to be a member of IBM's Performance Benchmarking team)
  - Hardware and software configurations are documented in detail; multiple configuration are tested.

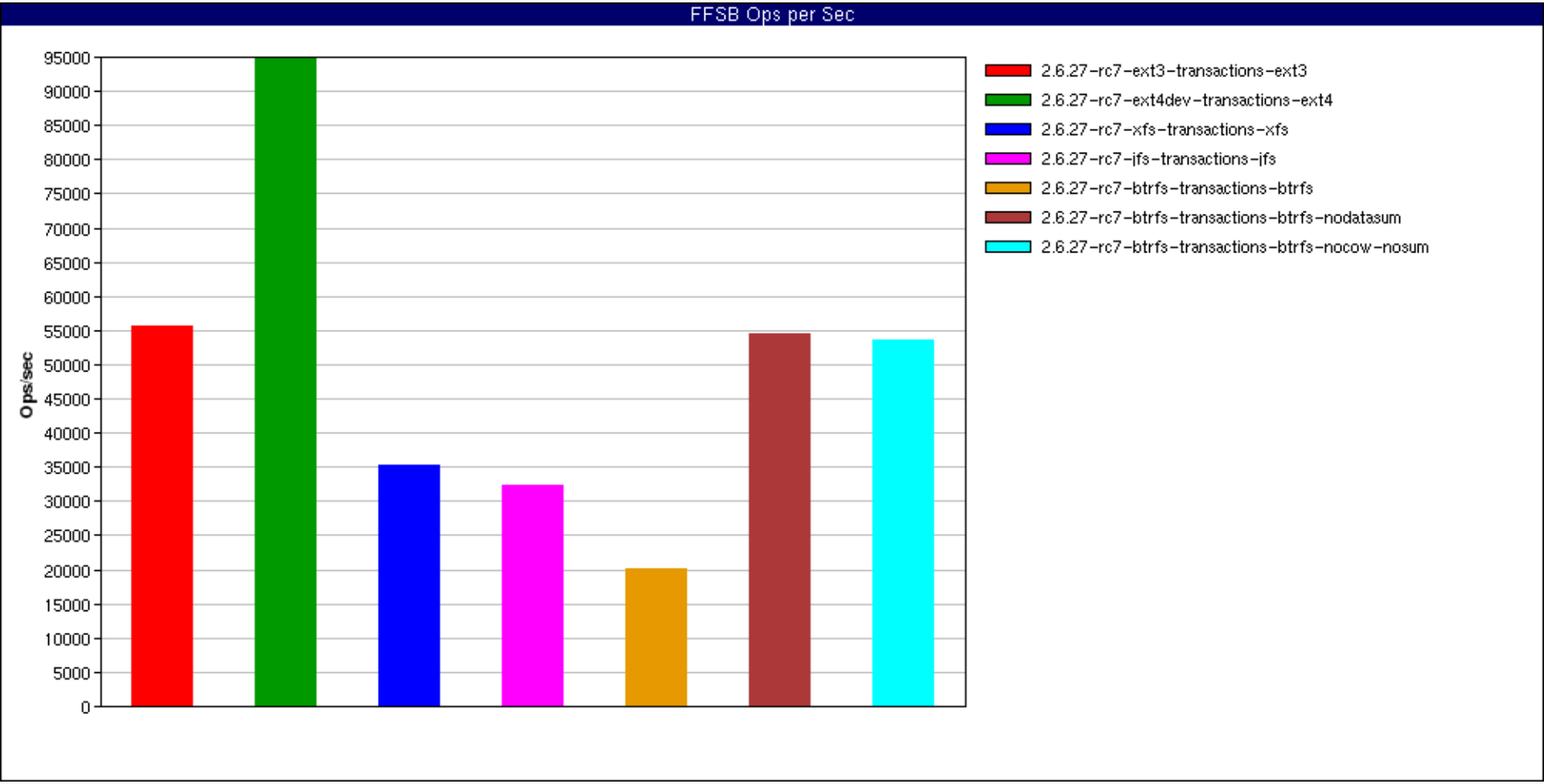
Large File Creates, Raid, 1 Threads



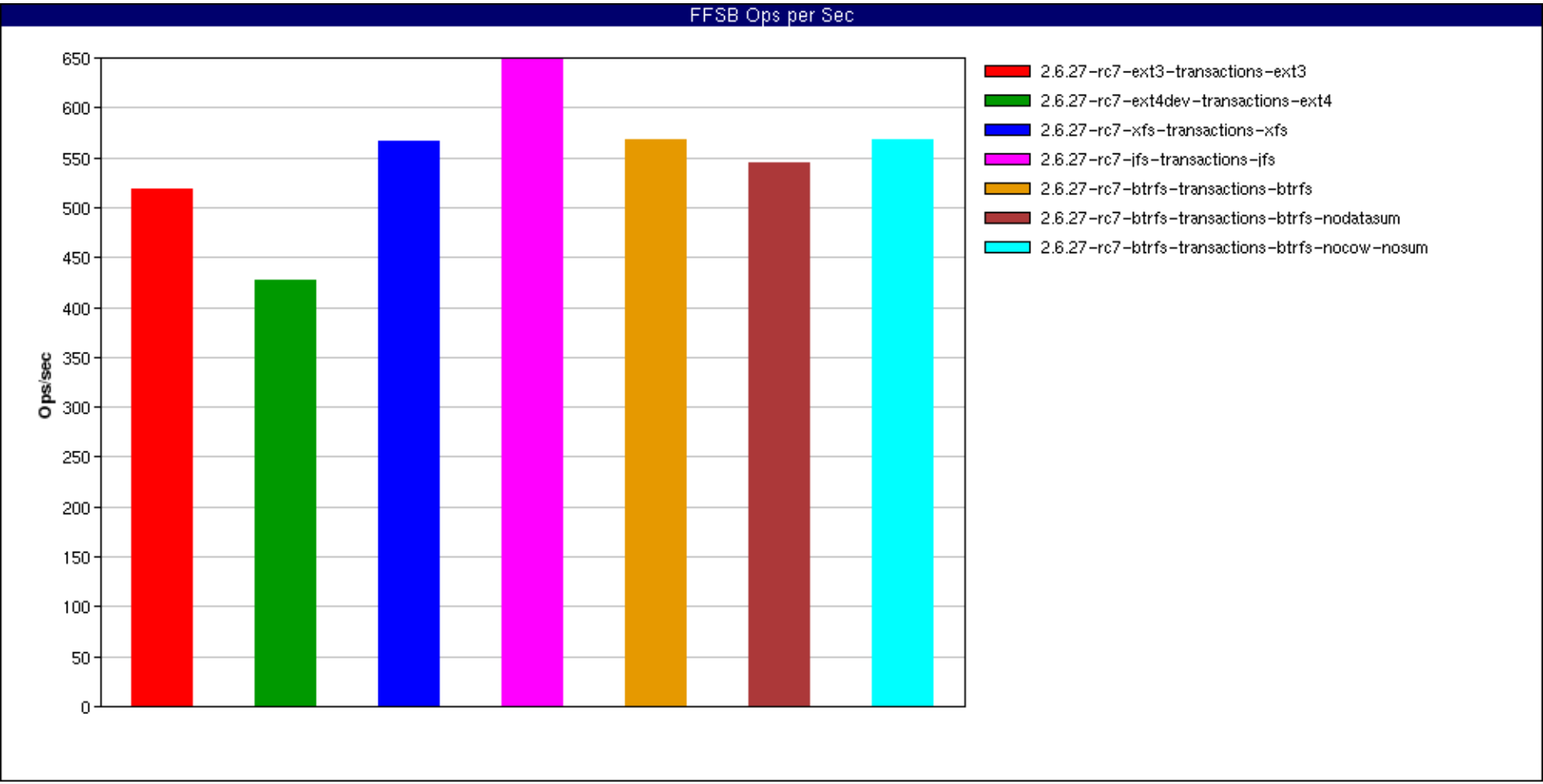
Large File Creates, Raid, 16 Threads



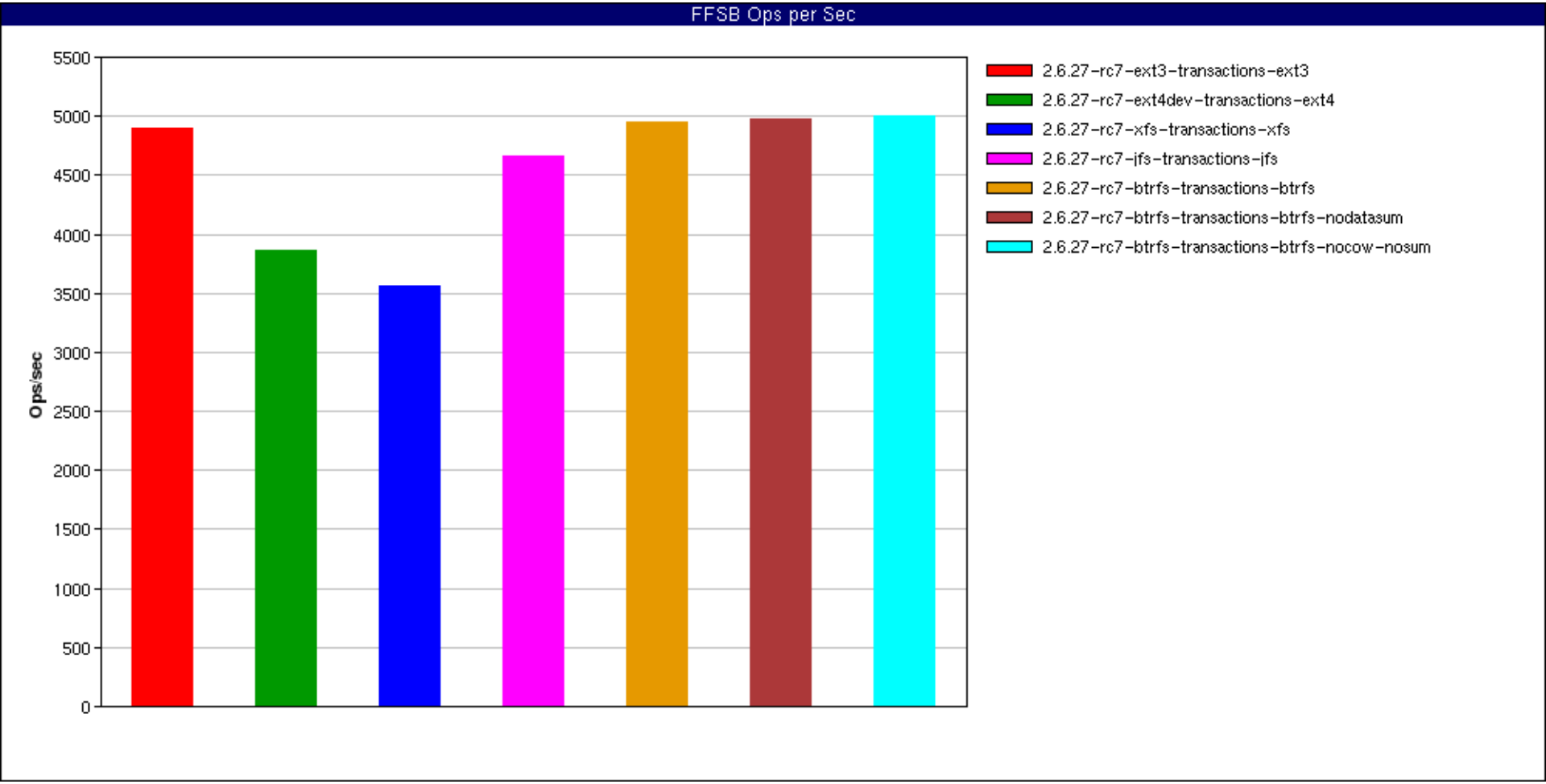
Large File Creates, RAID, 128 threads



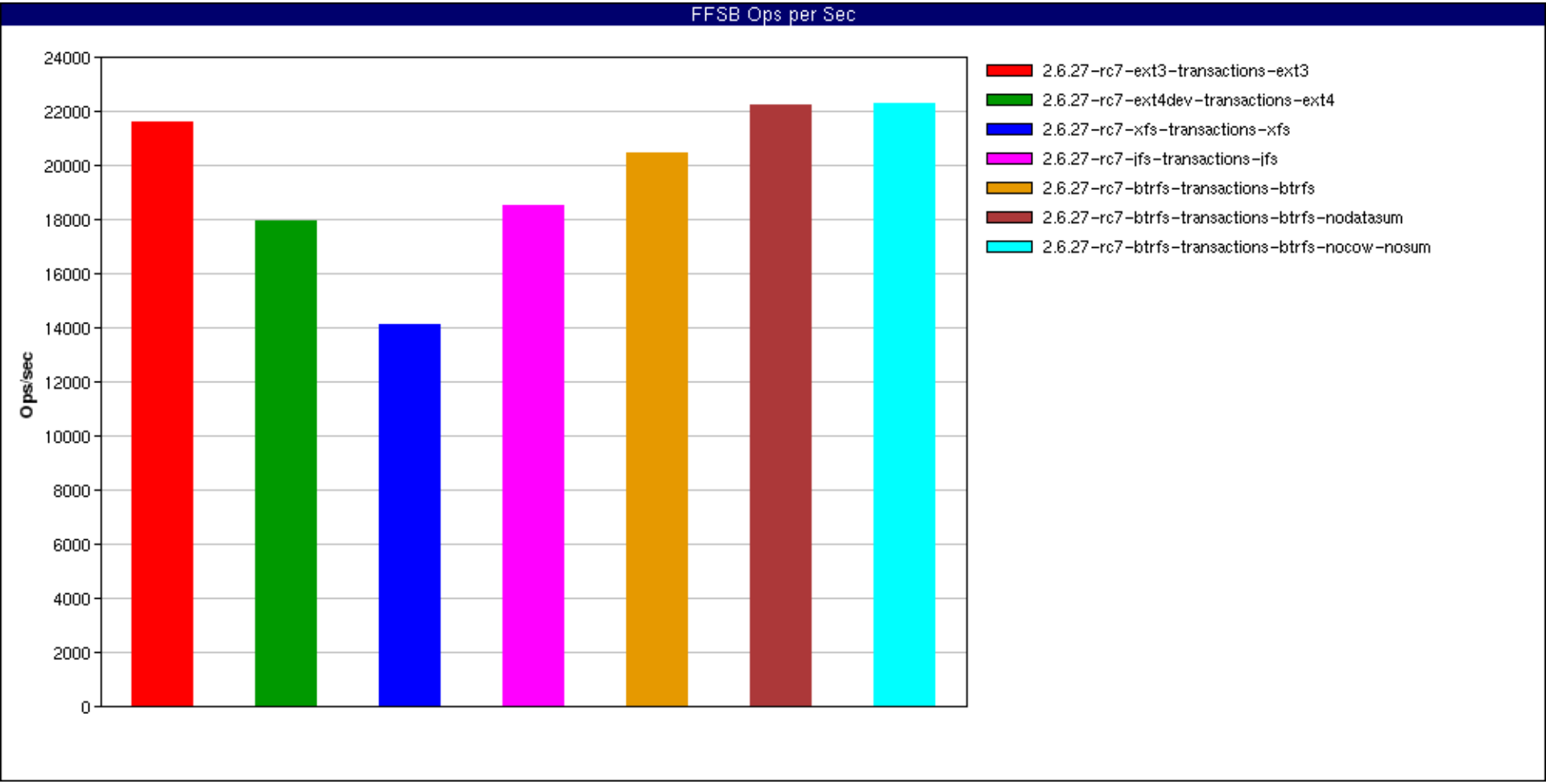
Large File Random Reads, RAID, 1 thread



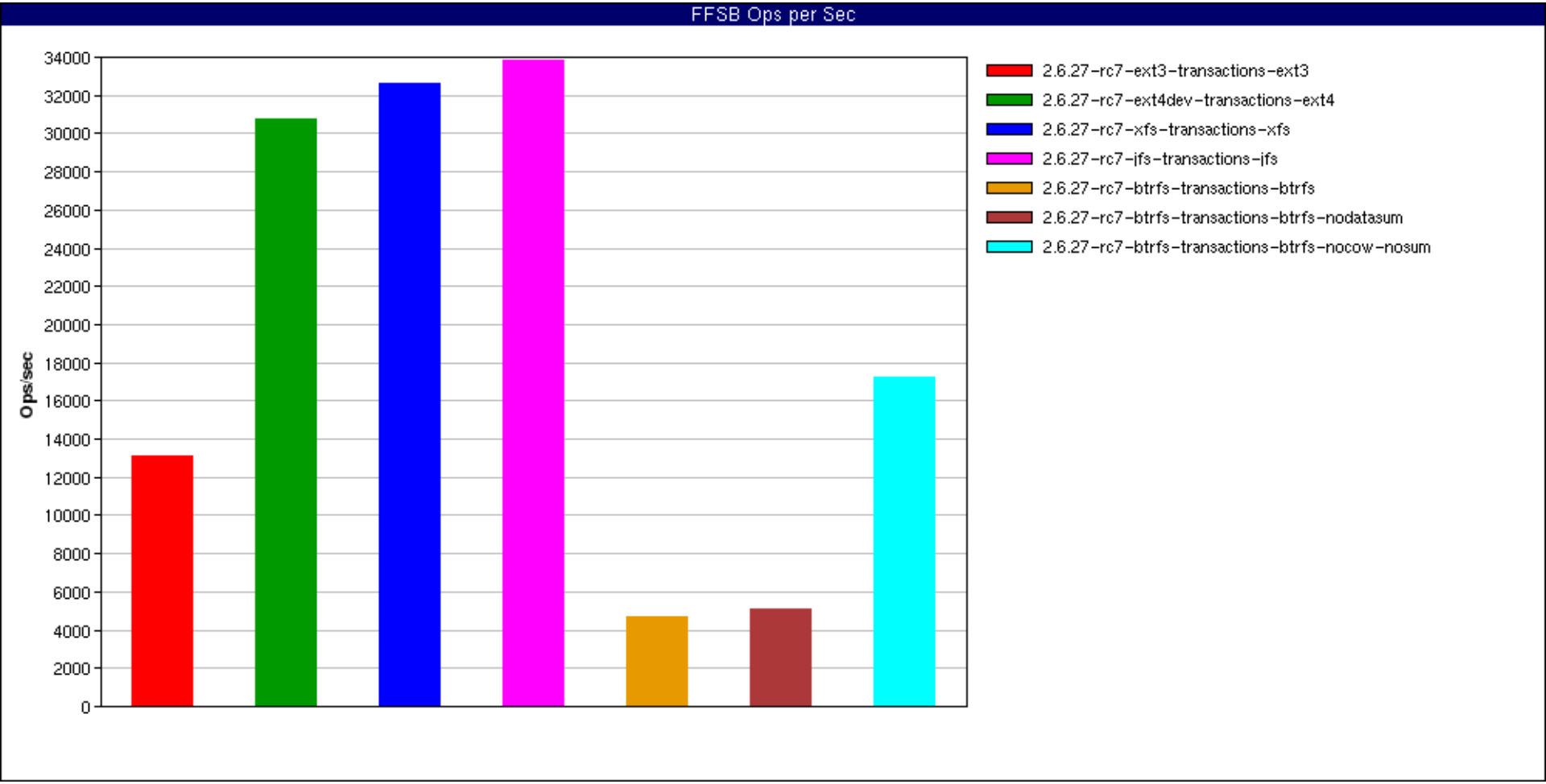
Large File Random Reads, RAID, 16 threads



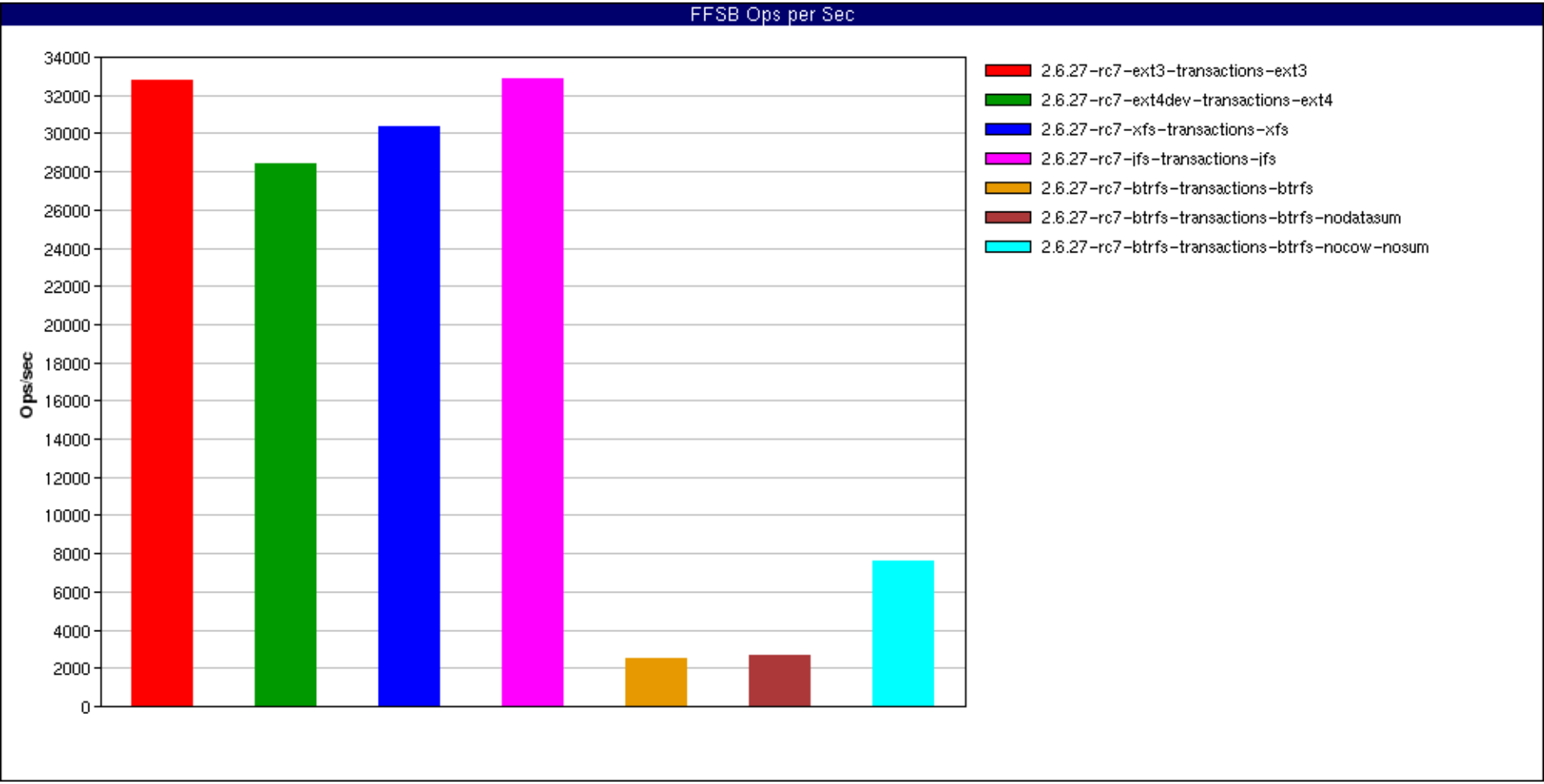
Large File Random Reads, RAID, 128 threads



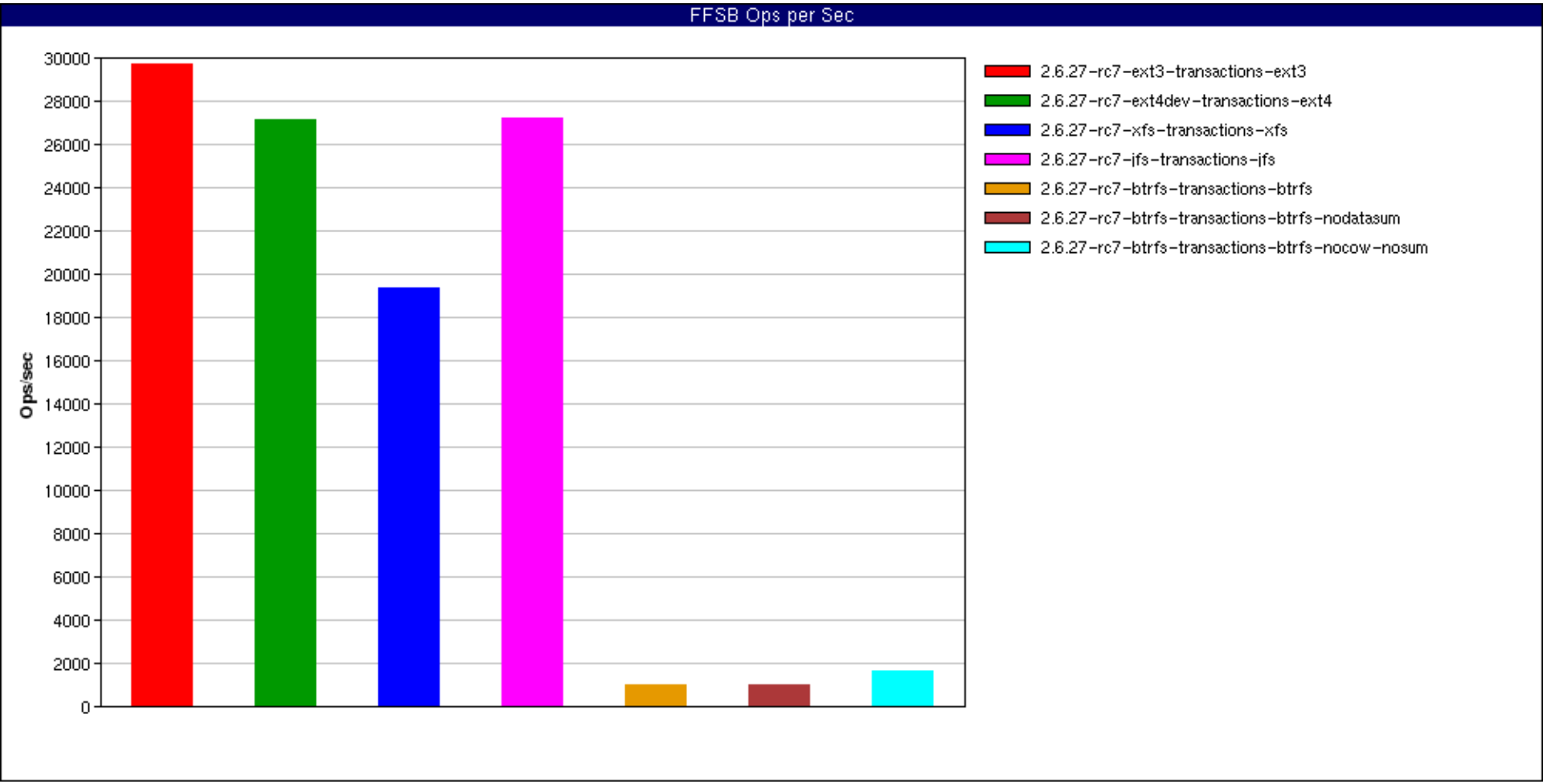
Large File Random Writes, RAID, 1 thread



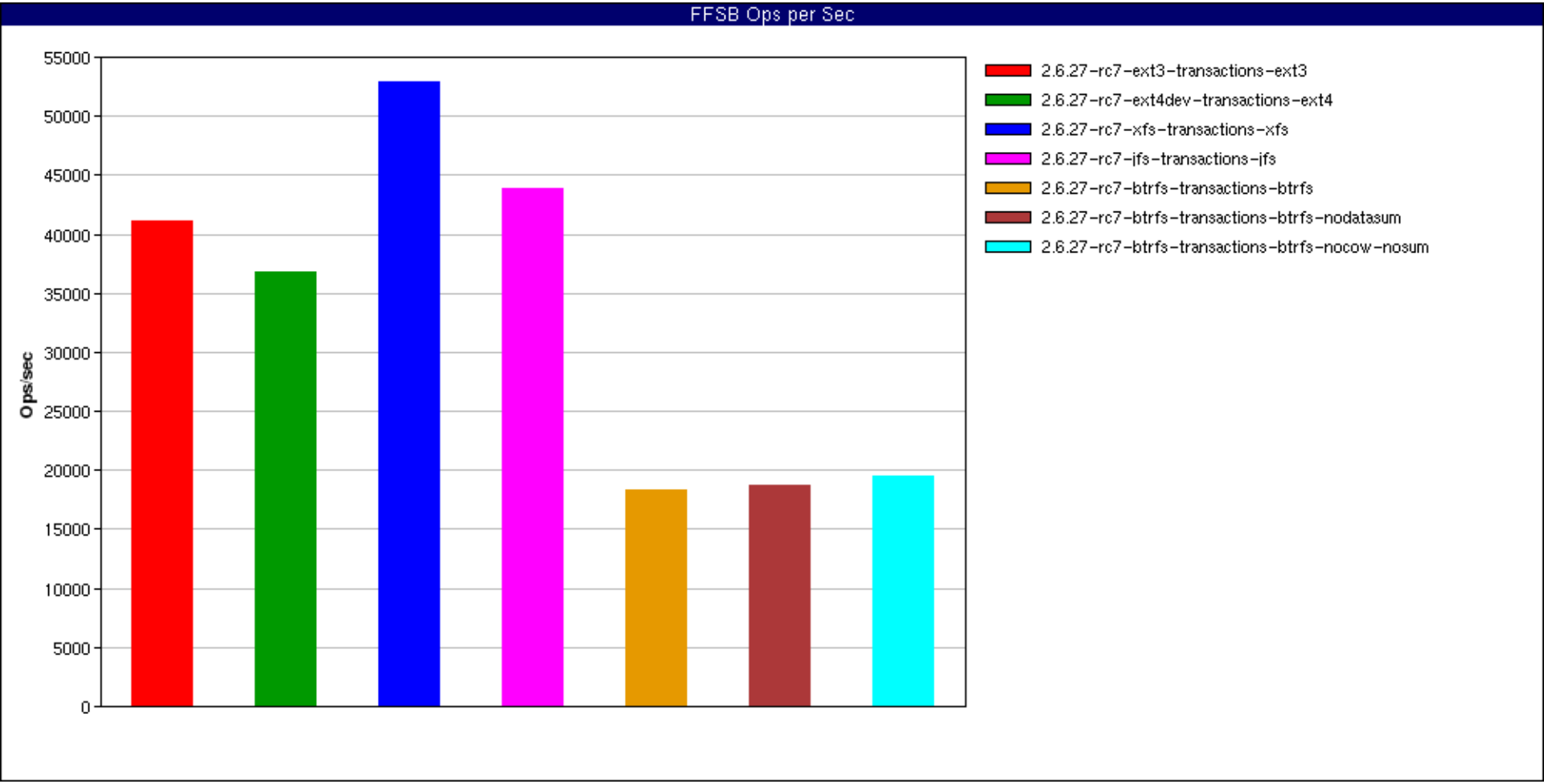
Large File Random Writes, RAID, 16 threads



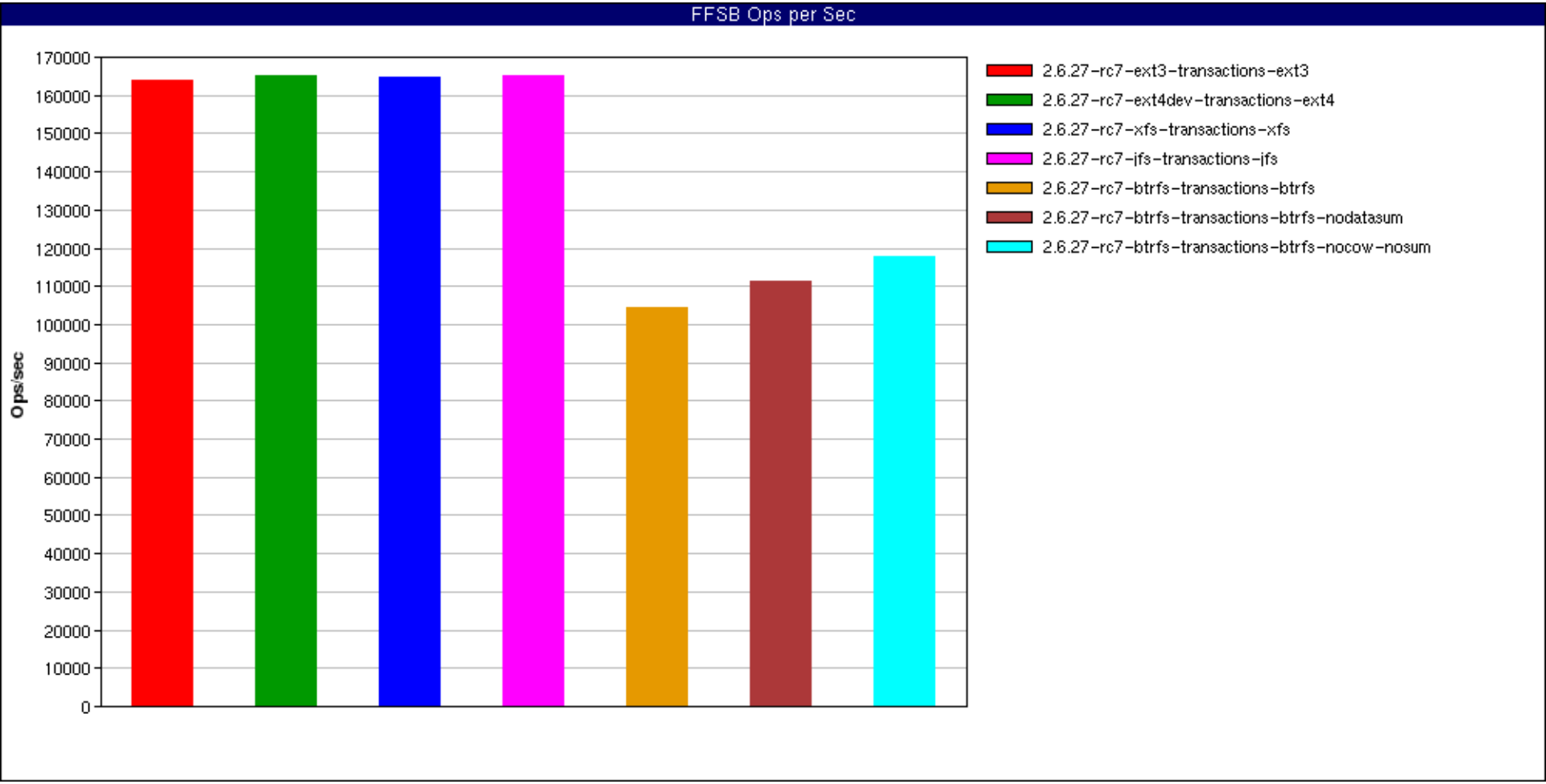
Large File Random Writes, RAID, 128 threads



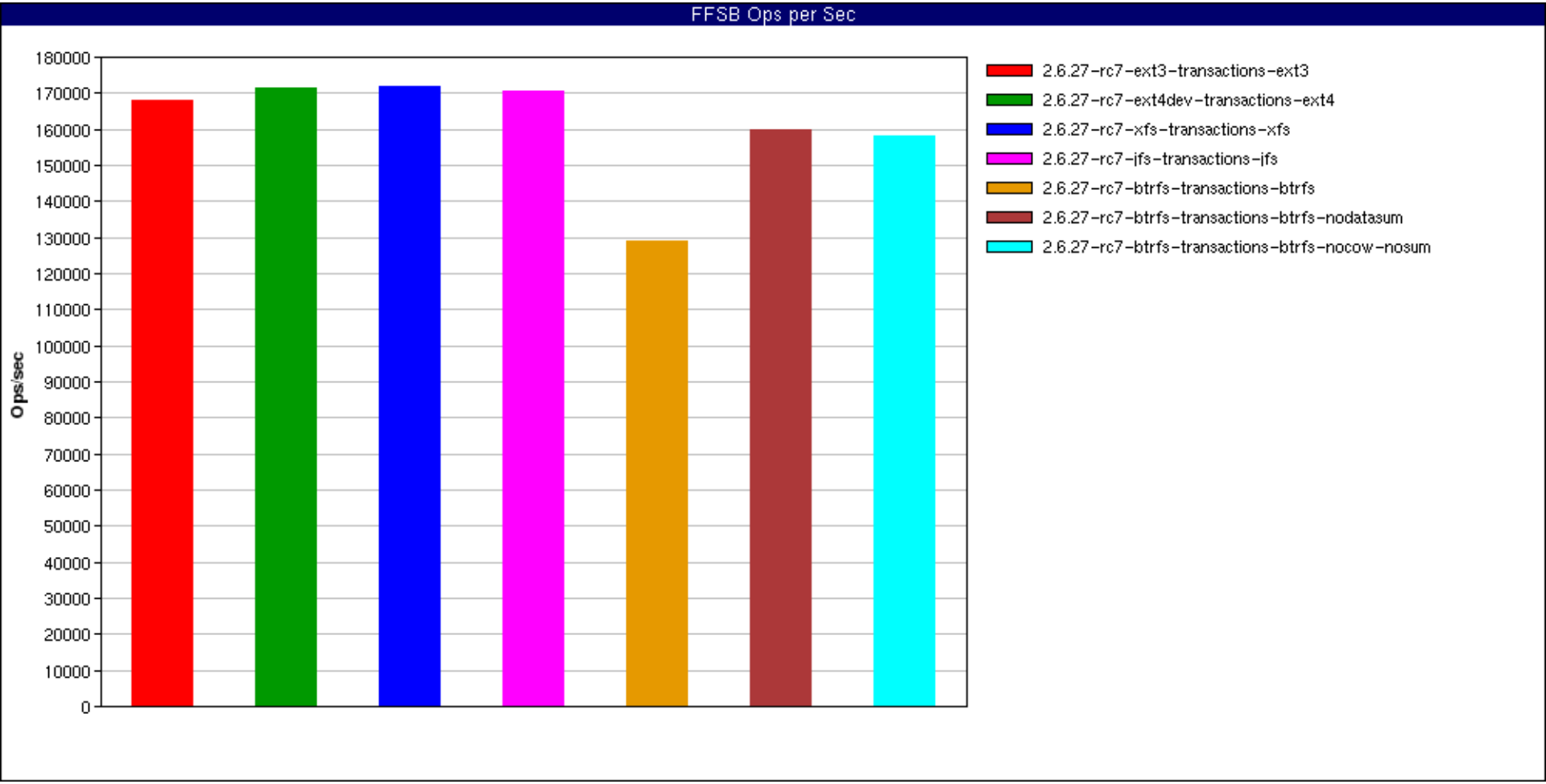
Large File Sequential Reads, RAID, 1 thread



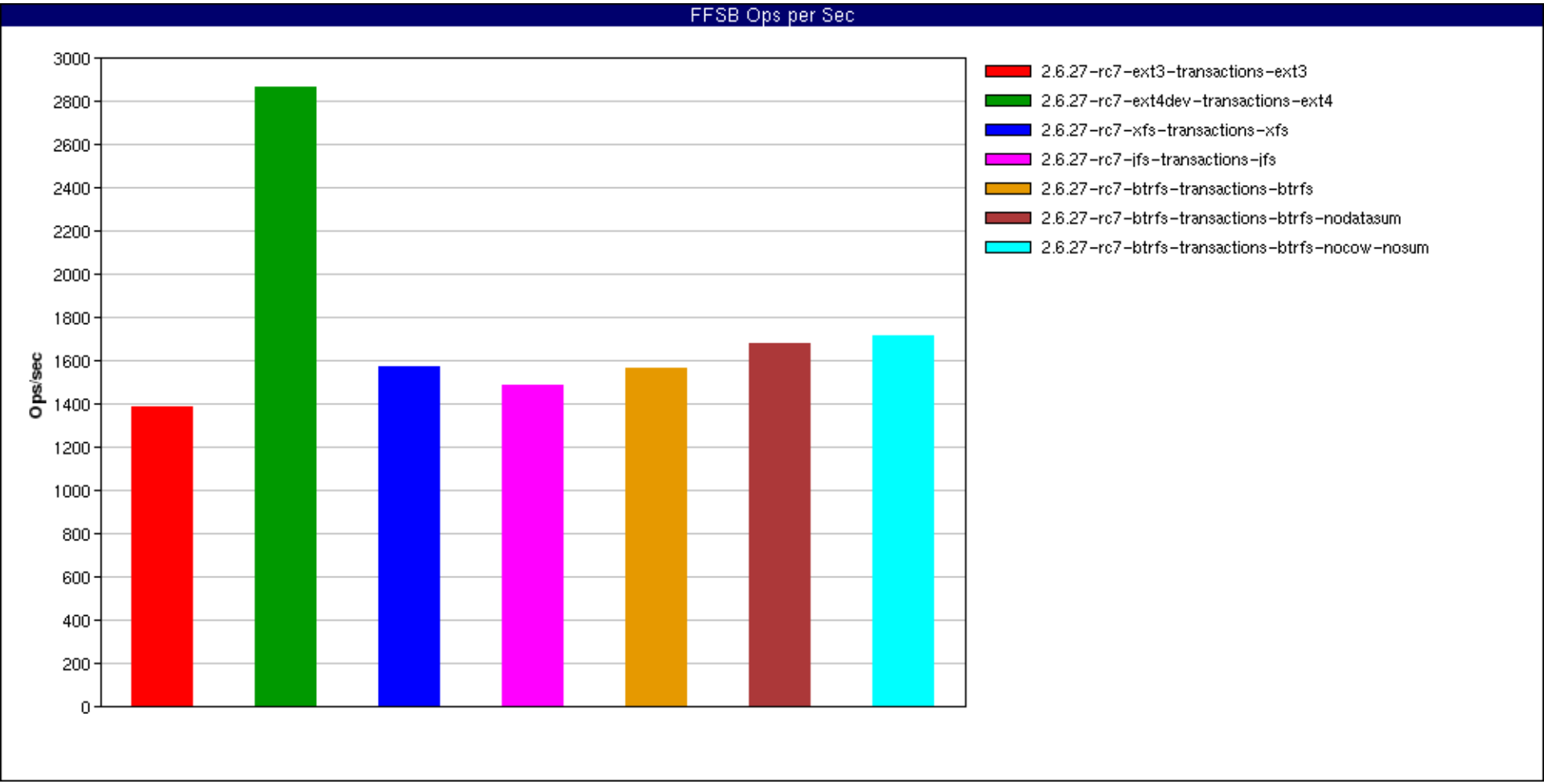
Large File Sequential Reads, RAID, 16 threads



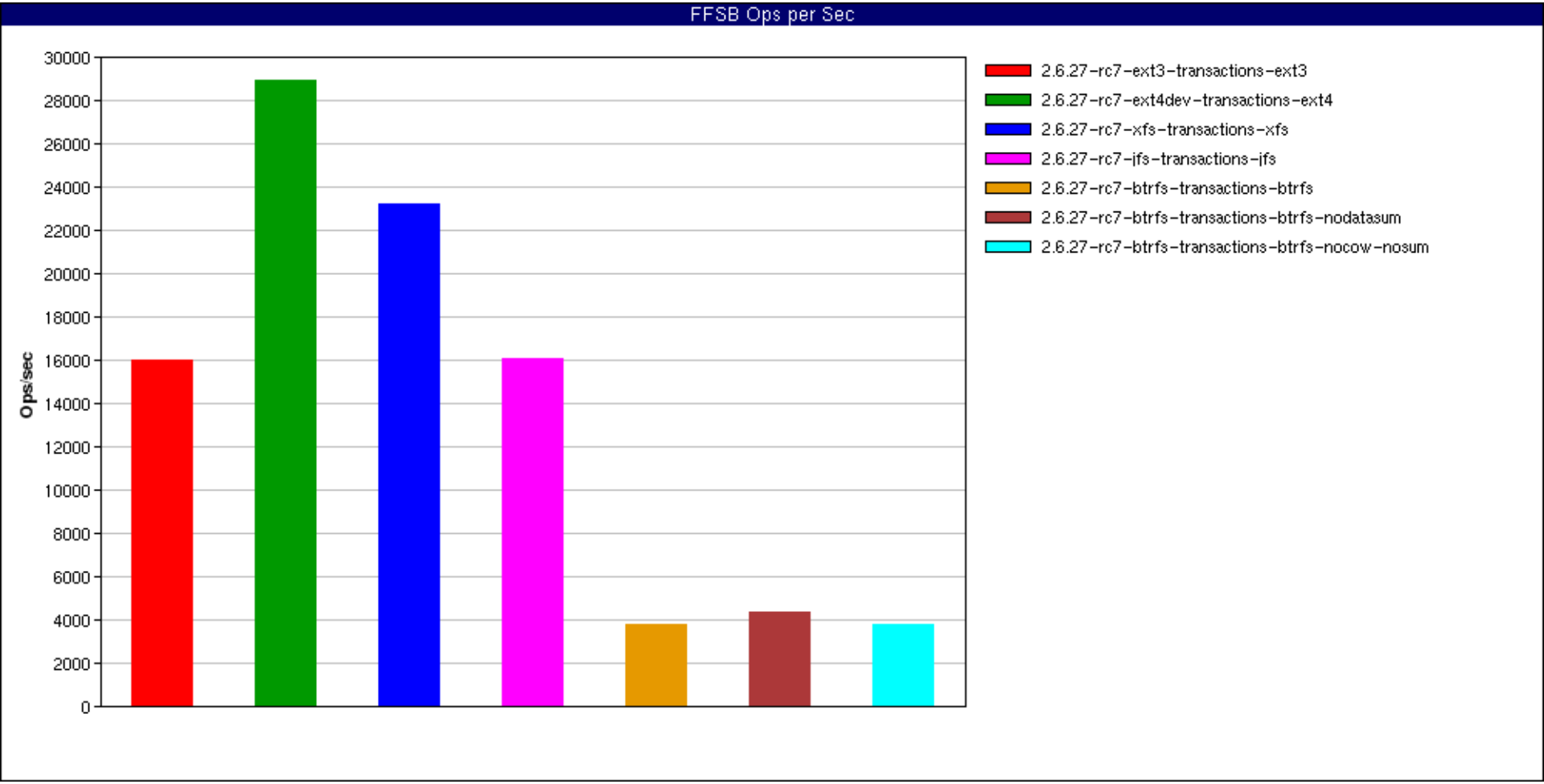
Large File Sequential Reads, RAID, 128 threads



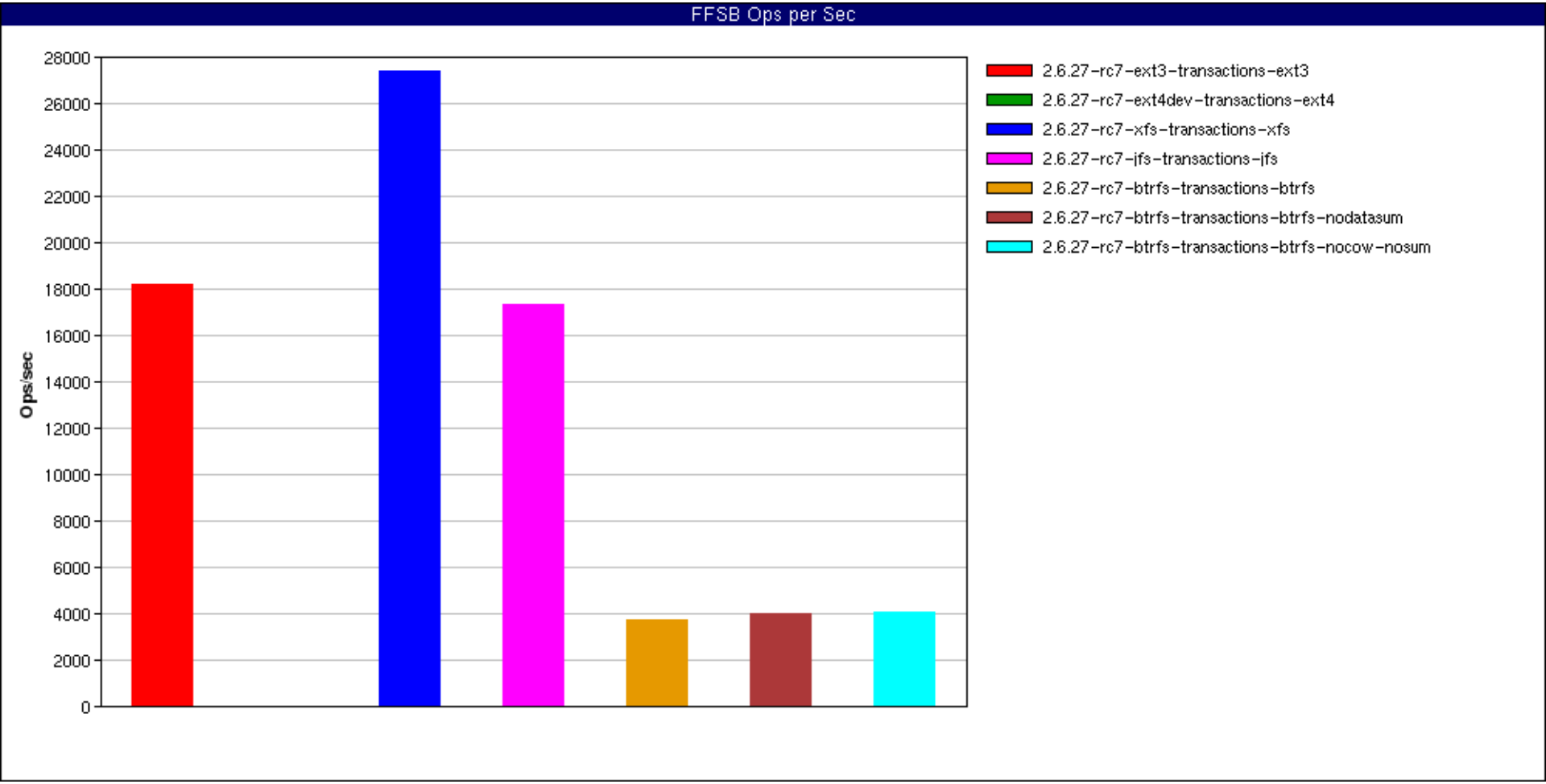
Mail Server Simulation, RAID, 1 thread



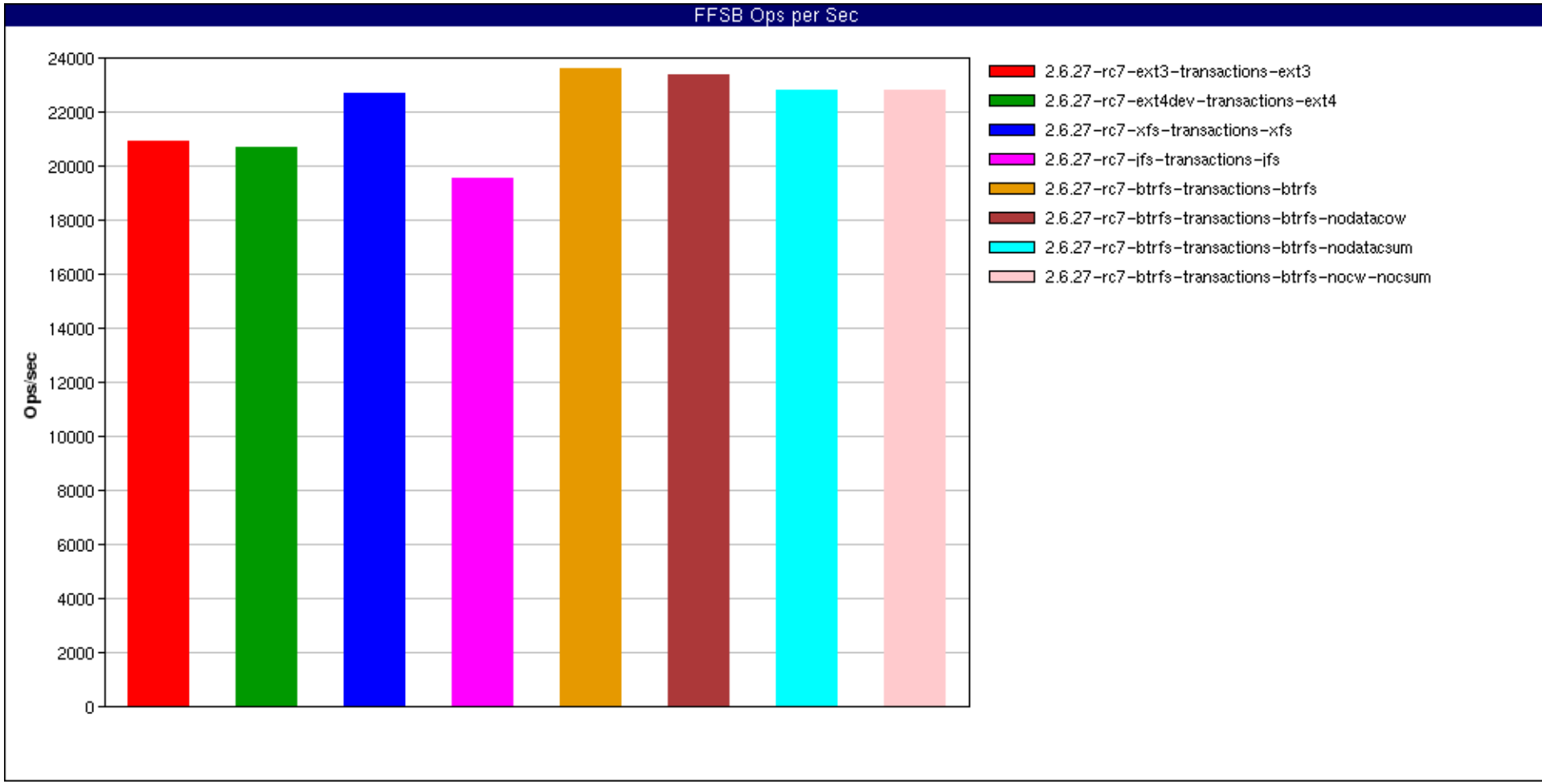
Mail Server Simulation, RAID, 16 threads



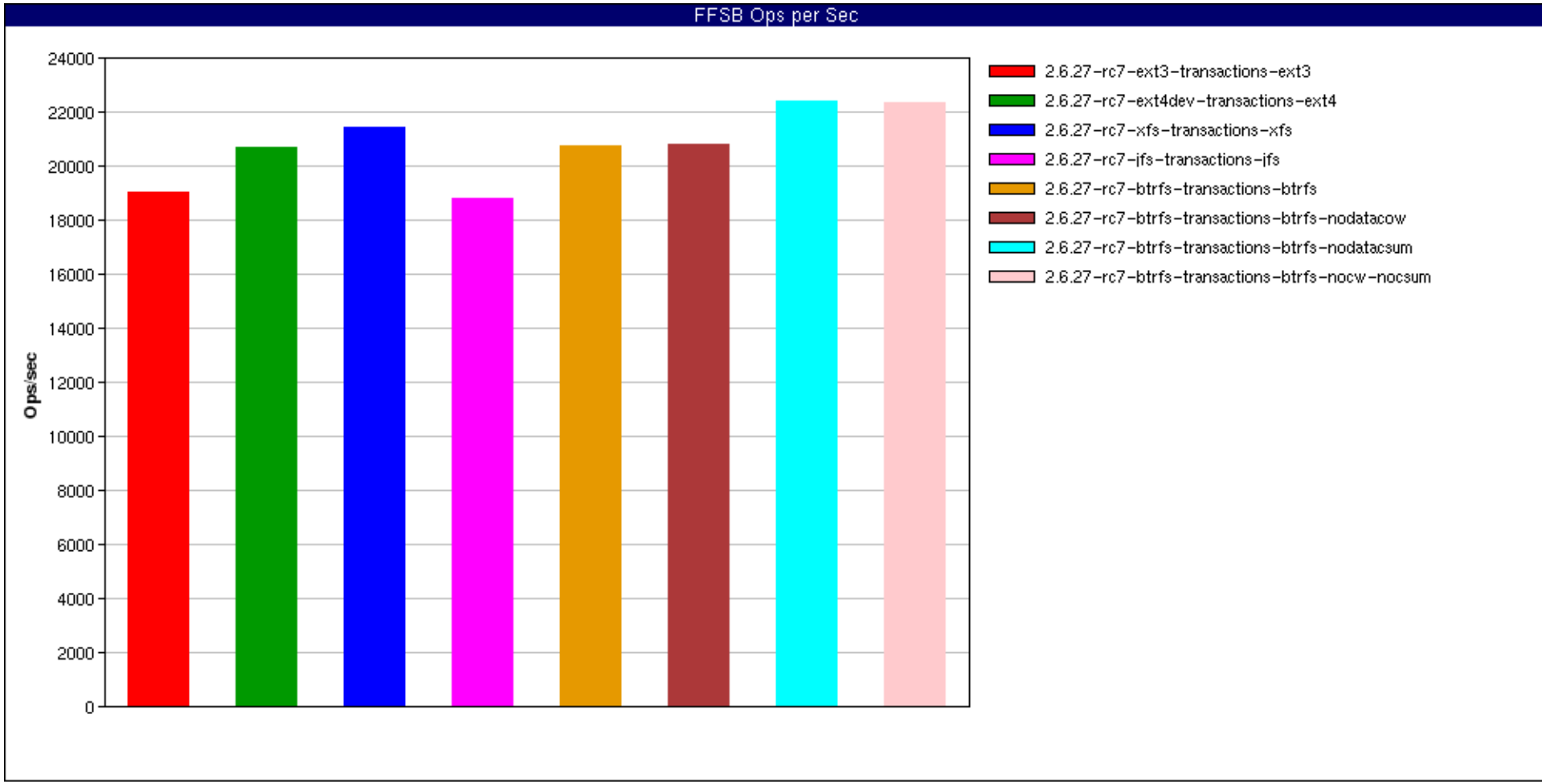
Mail Server Simulation, RAID, 128 threads



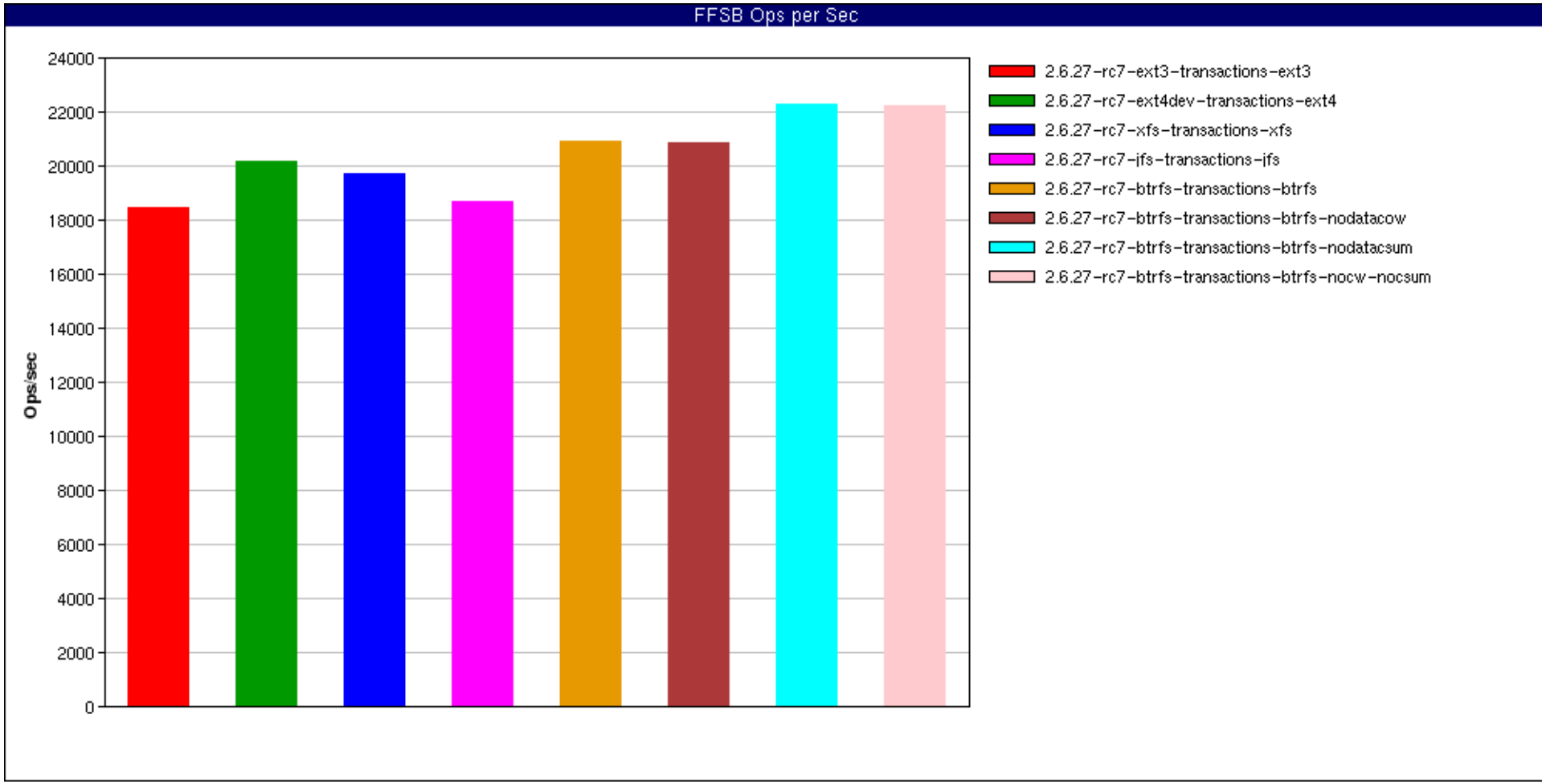
Large File Creates, Single Disk, 1 thread



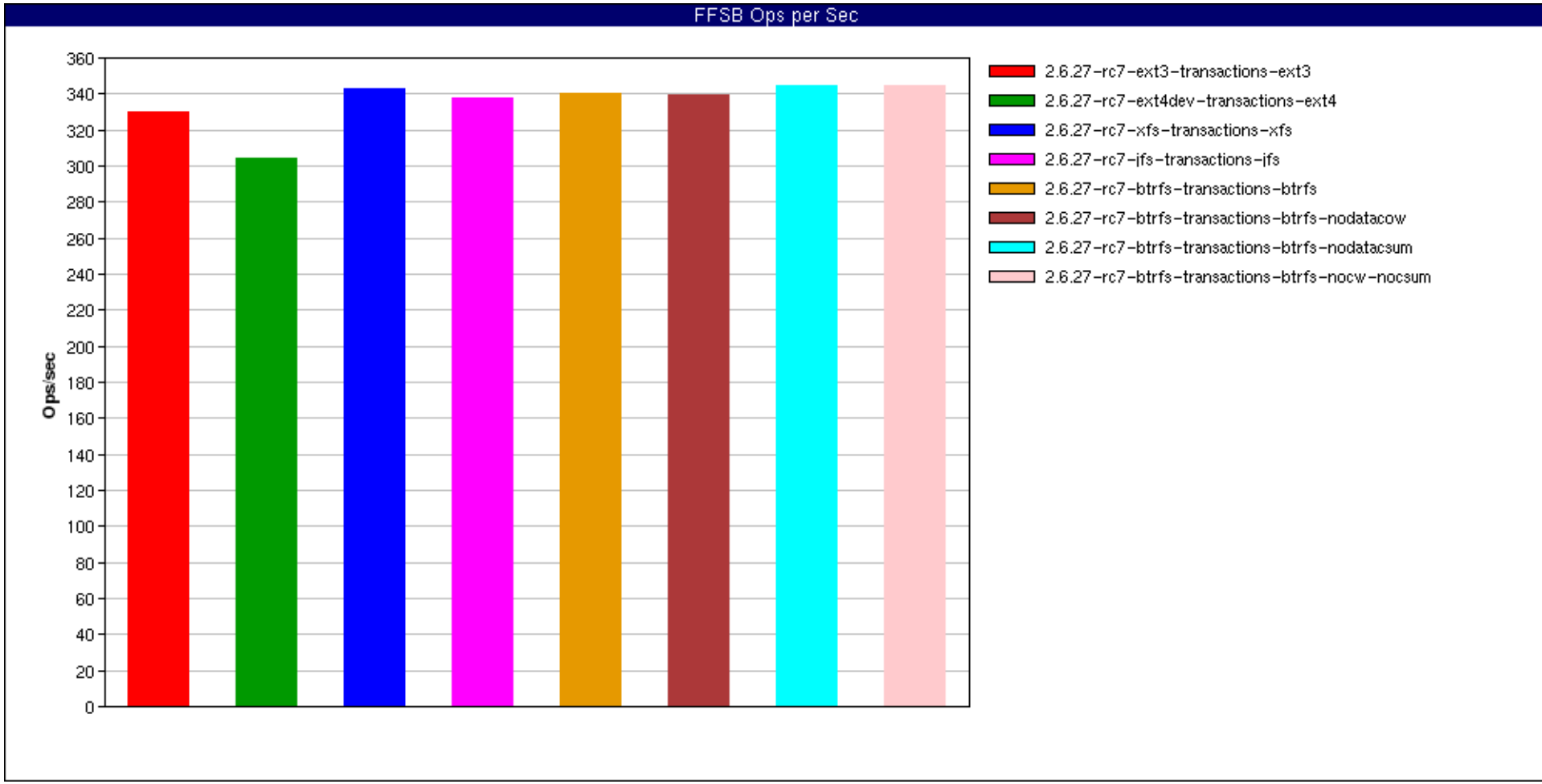
Large File Creates, Single Disk, 8 threads



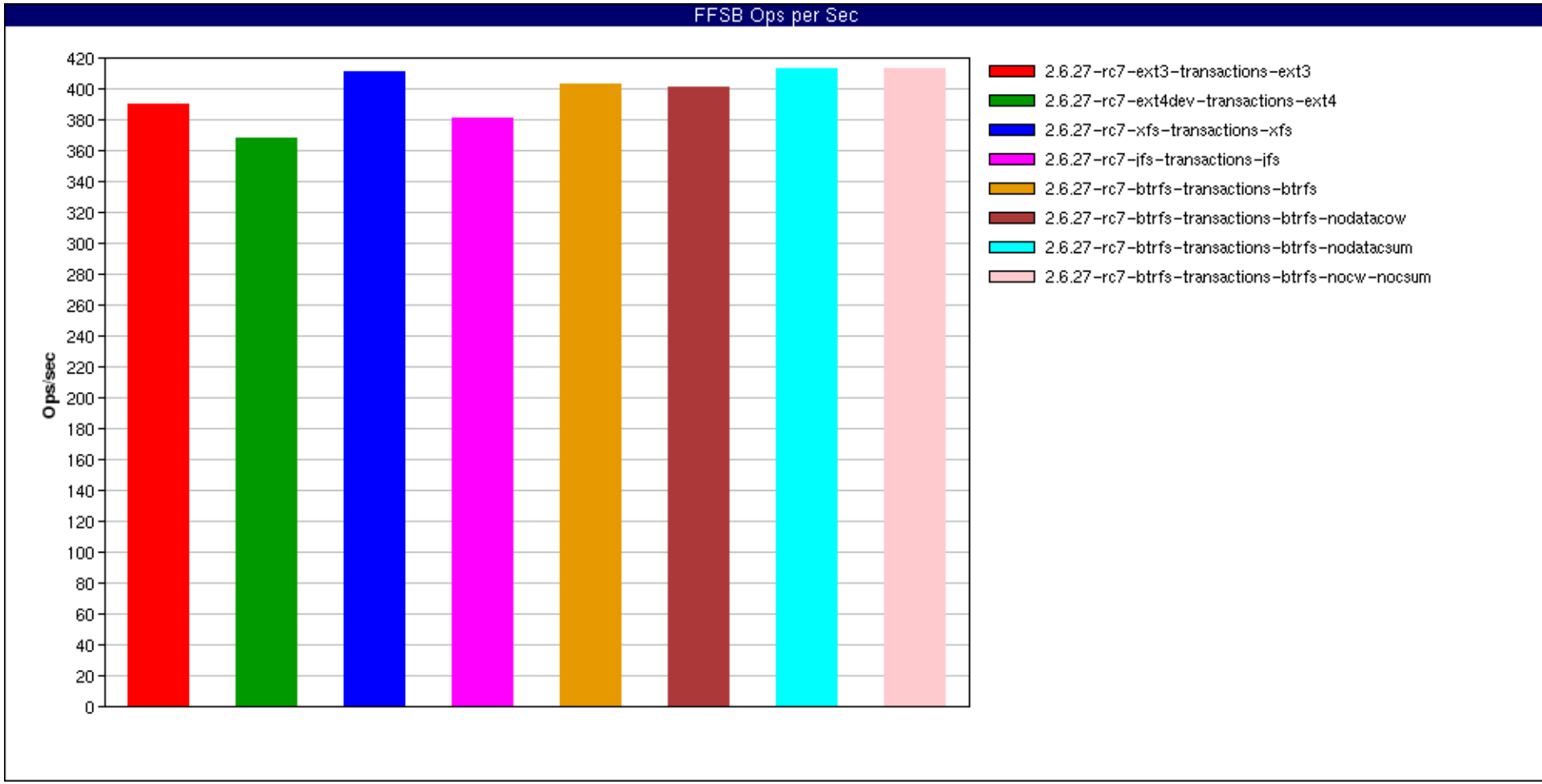
Large File Creates, Single Disk, 32 threads



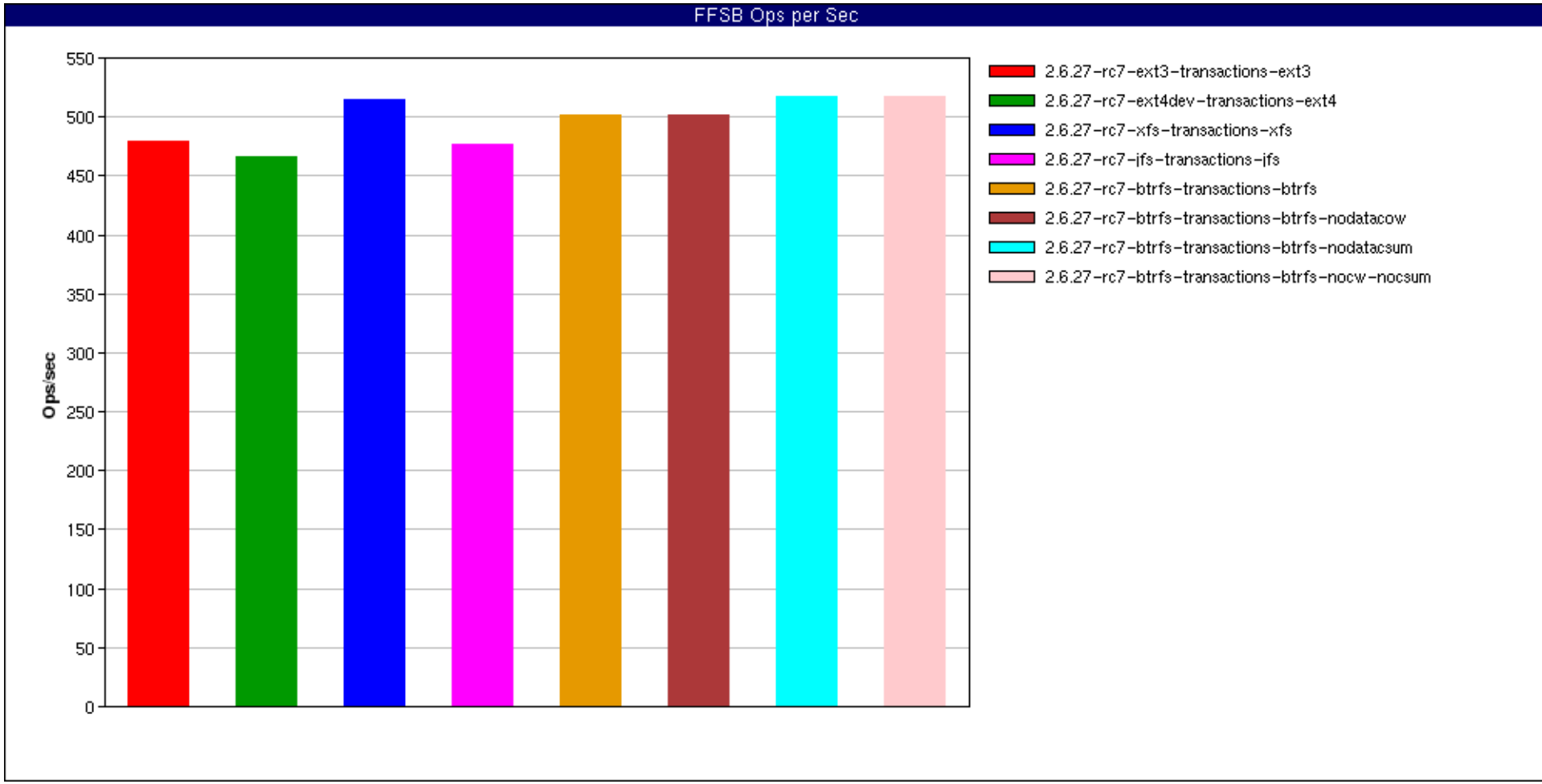
Large File Random Reads, Single Disk, 1 thread



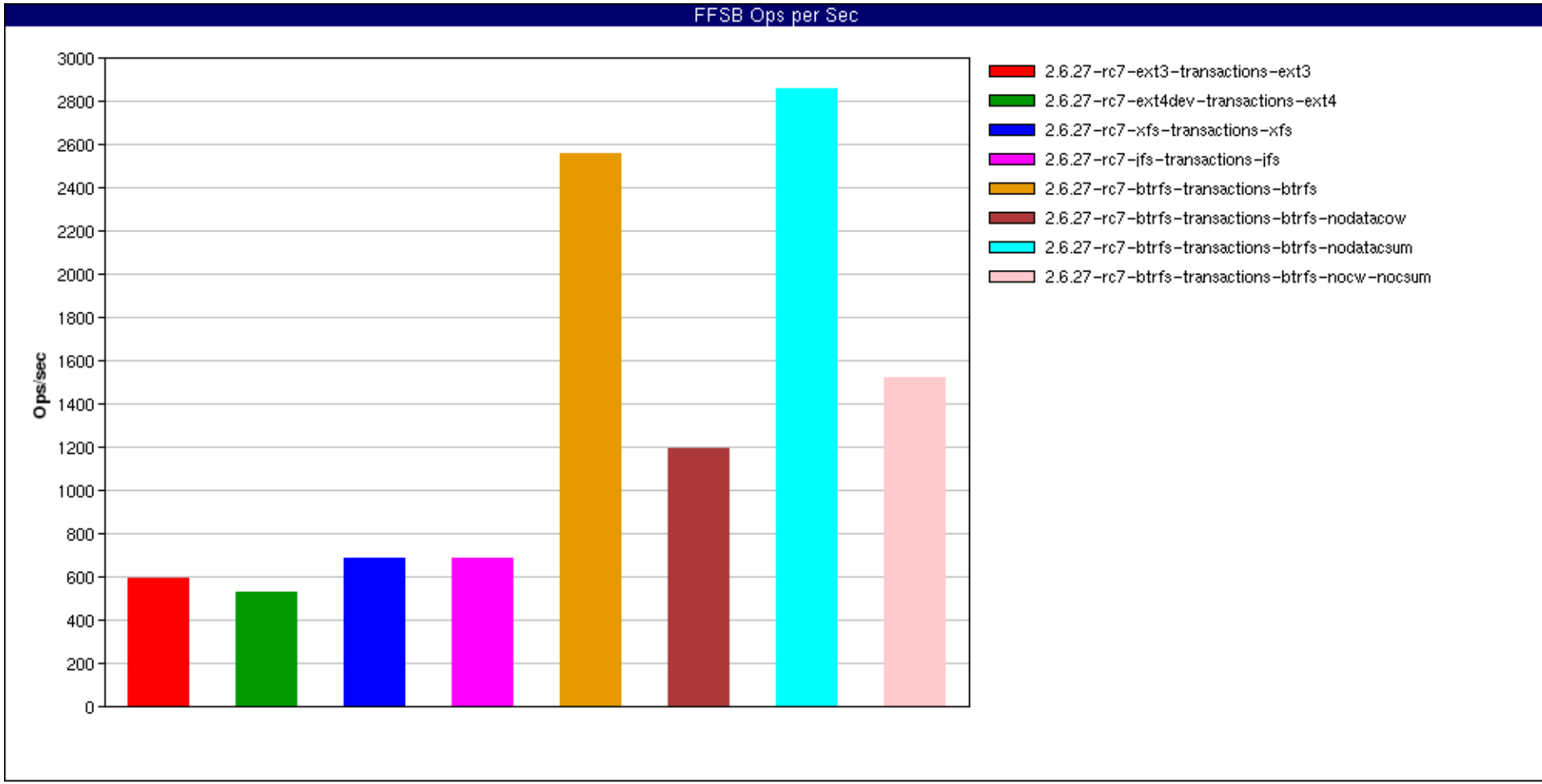
Large File Random Reads, Single Disk, 8 threads



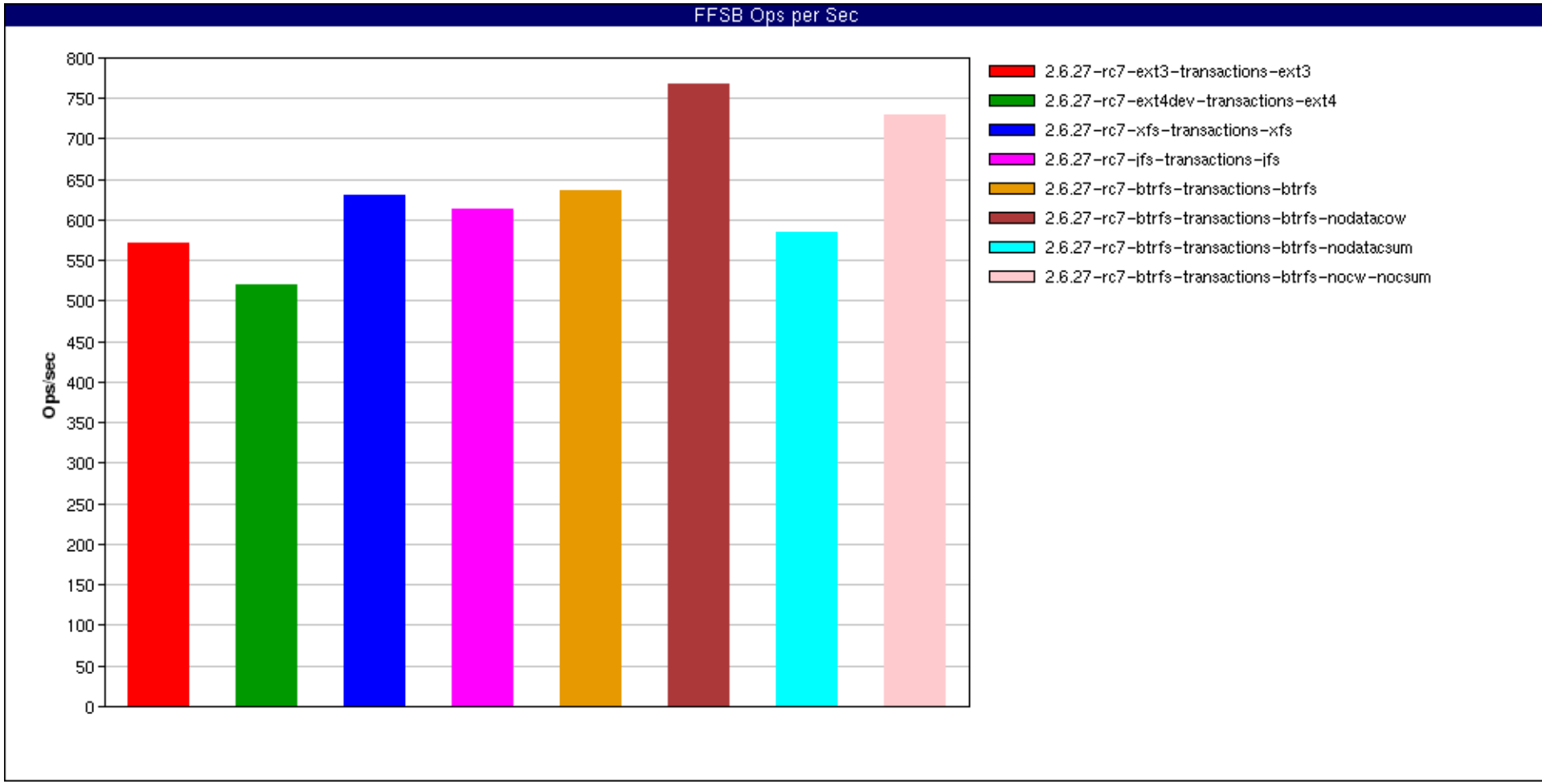
Large File Random Reads, Single Disk, 32 threads



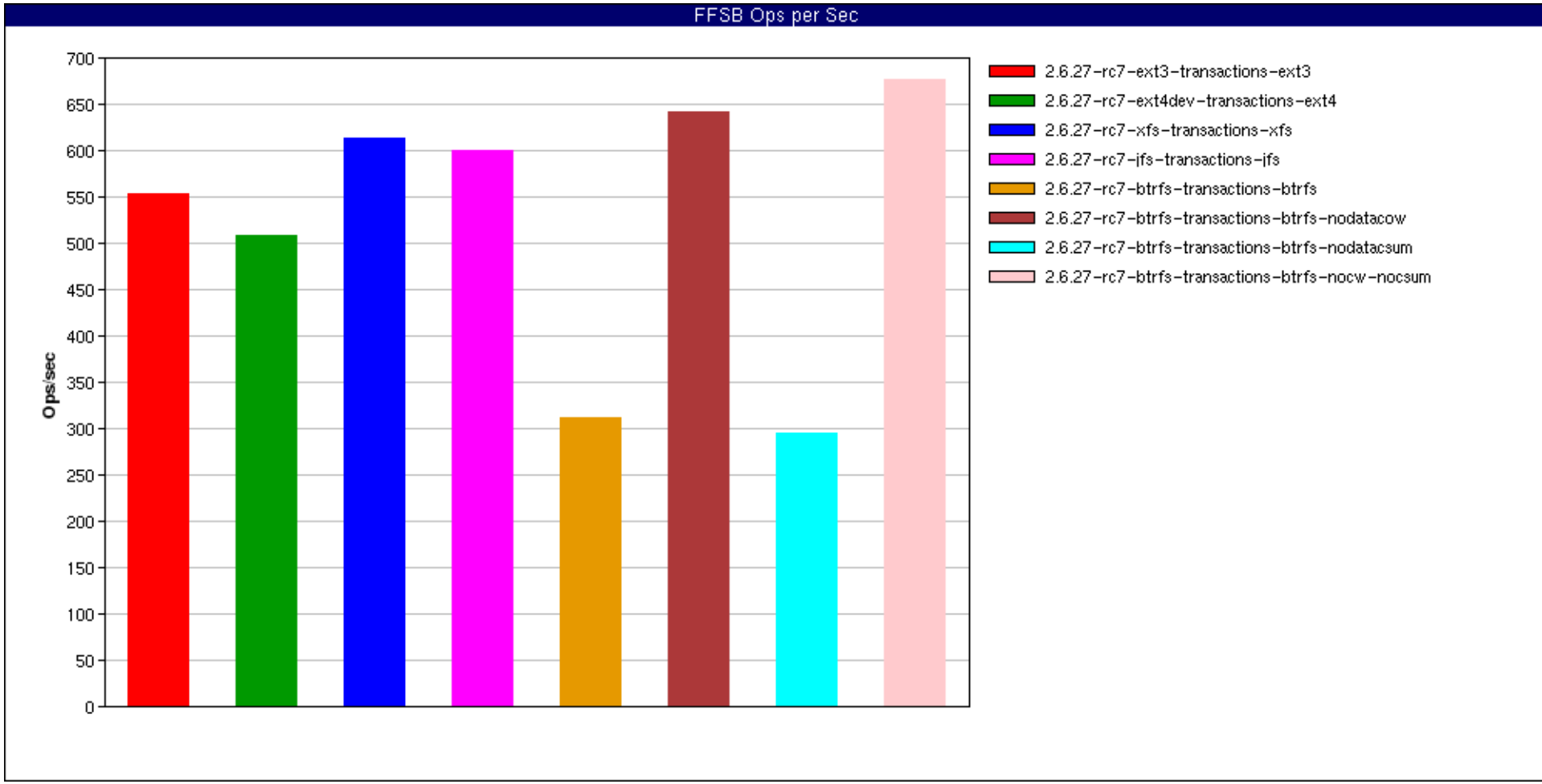
Large File Random Writes, Single Disk, 1 thread



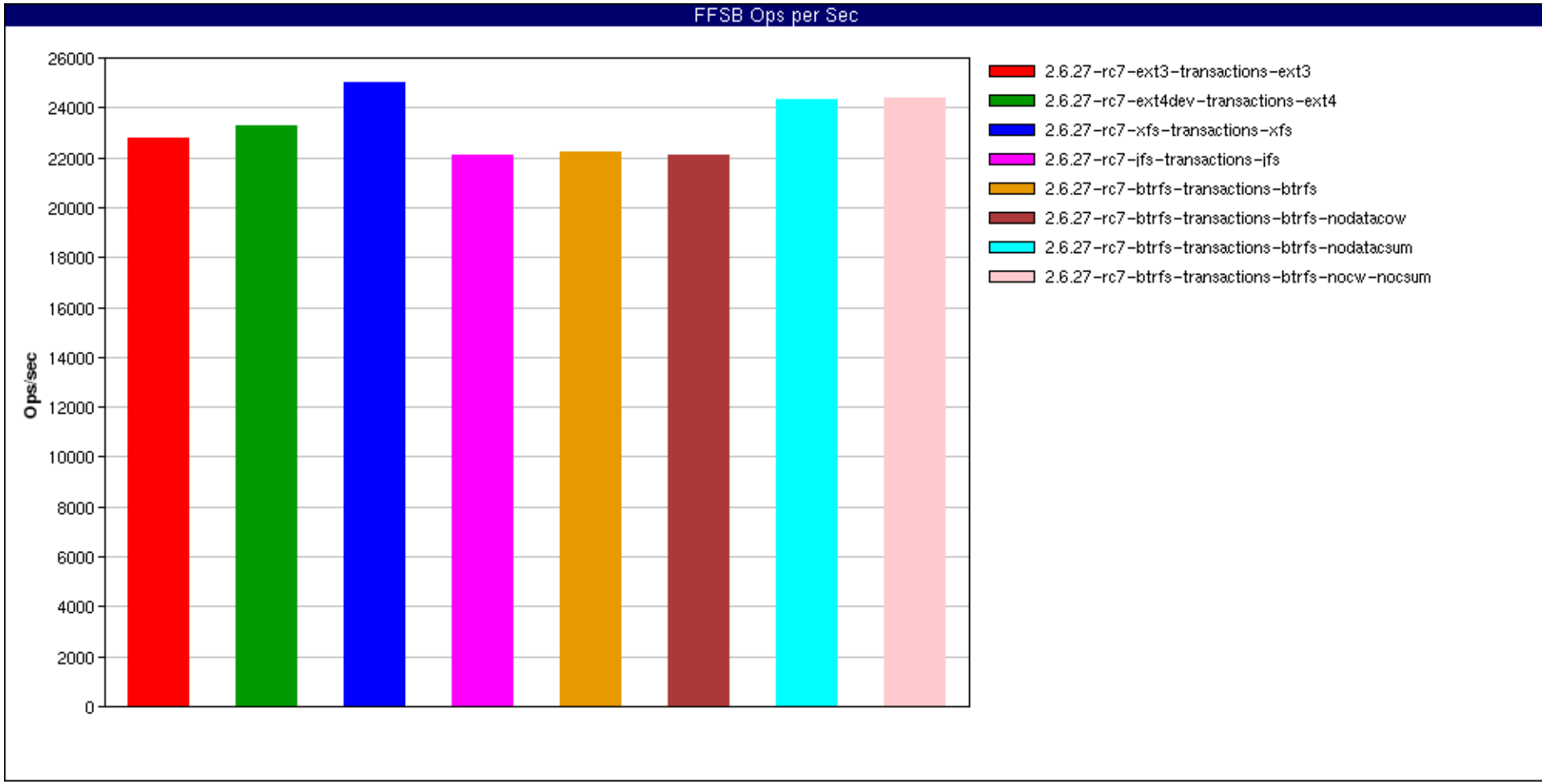
Large File Random Writes, Single Disk, 8 threads



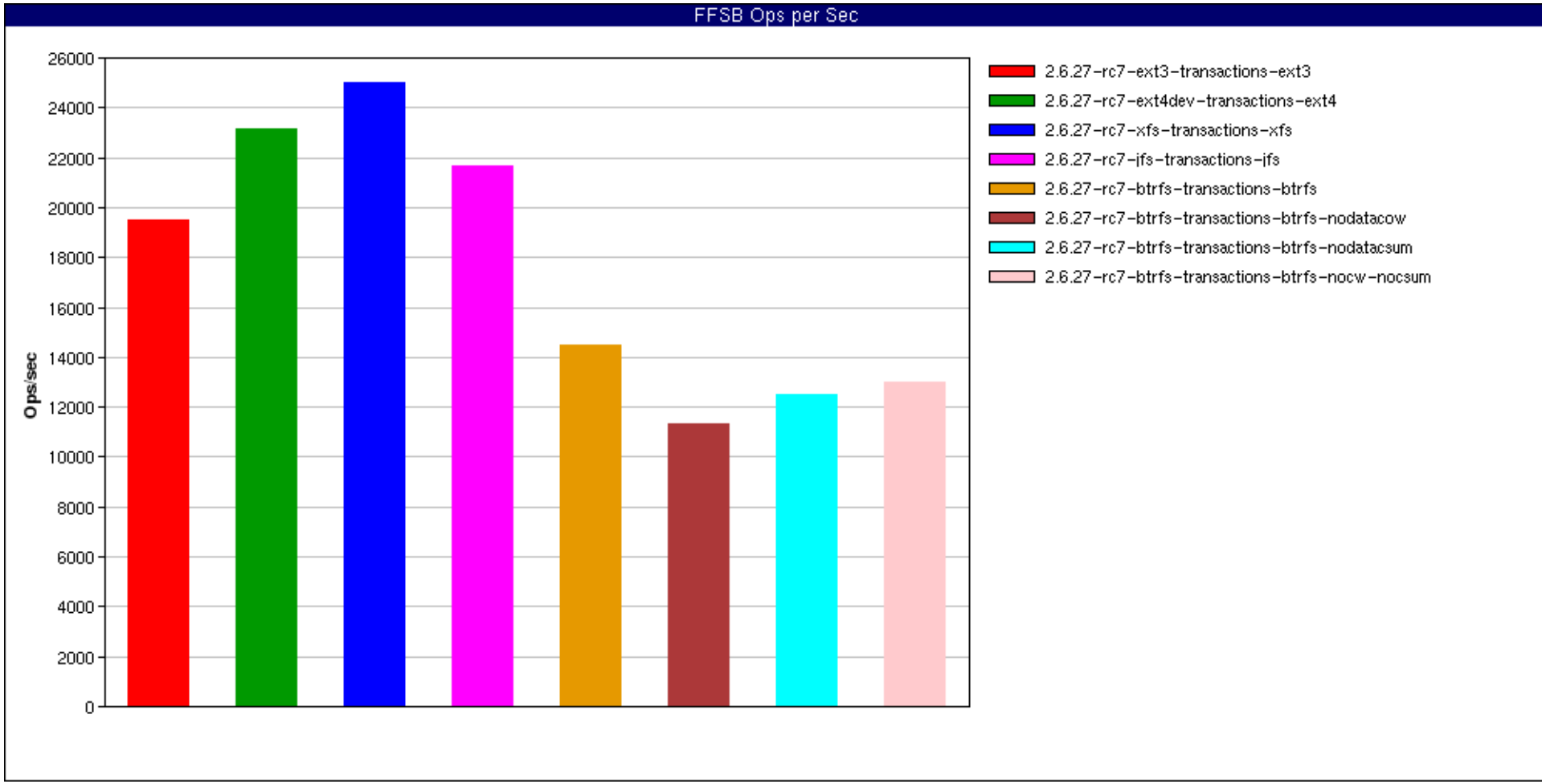
Large File Random Writes, Single Disk, 32 threads



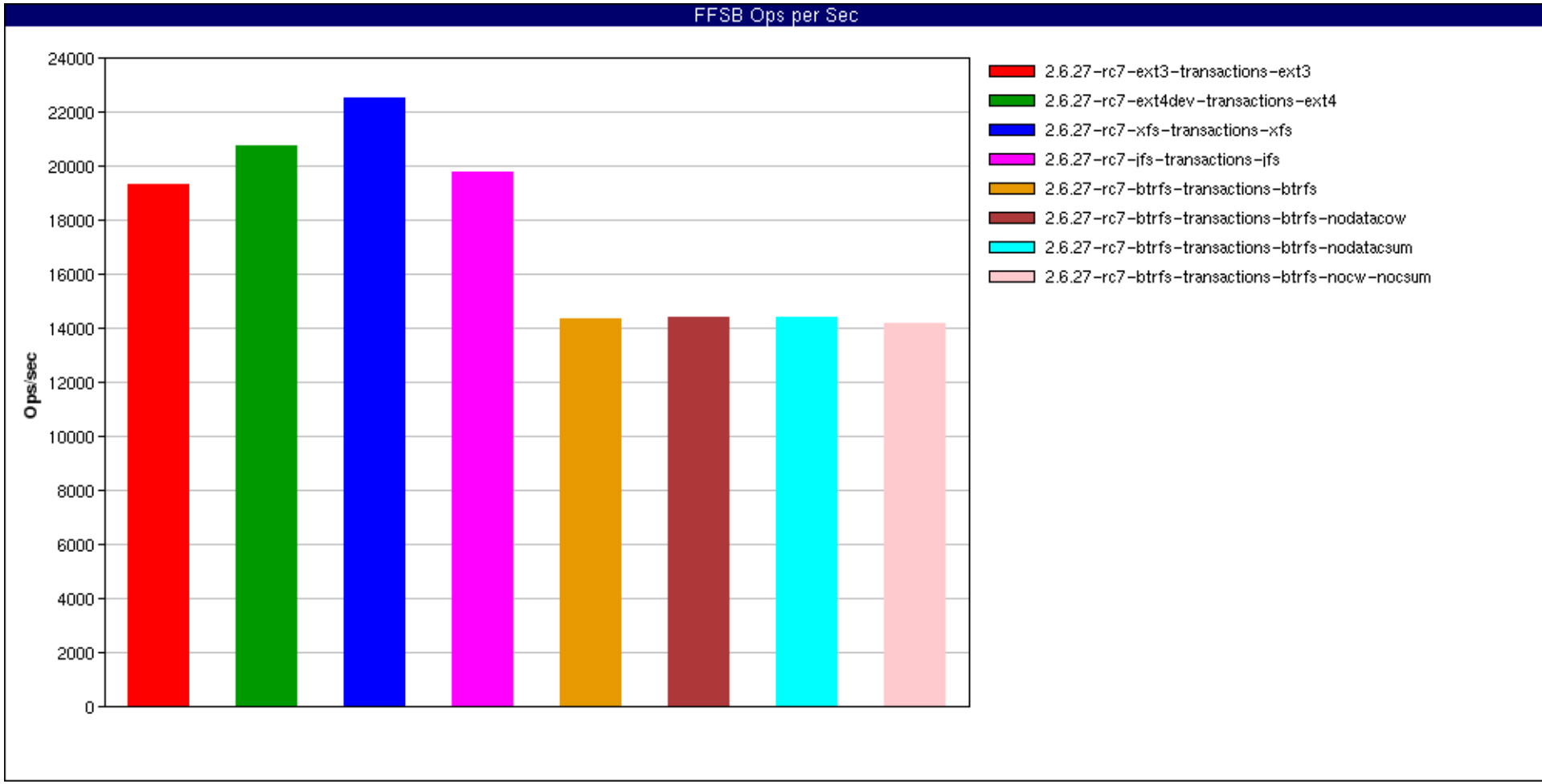
Large File Seq Reads, Single Disk 1 thread



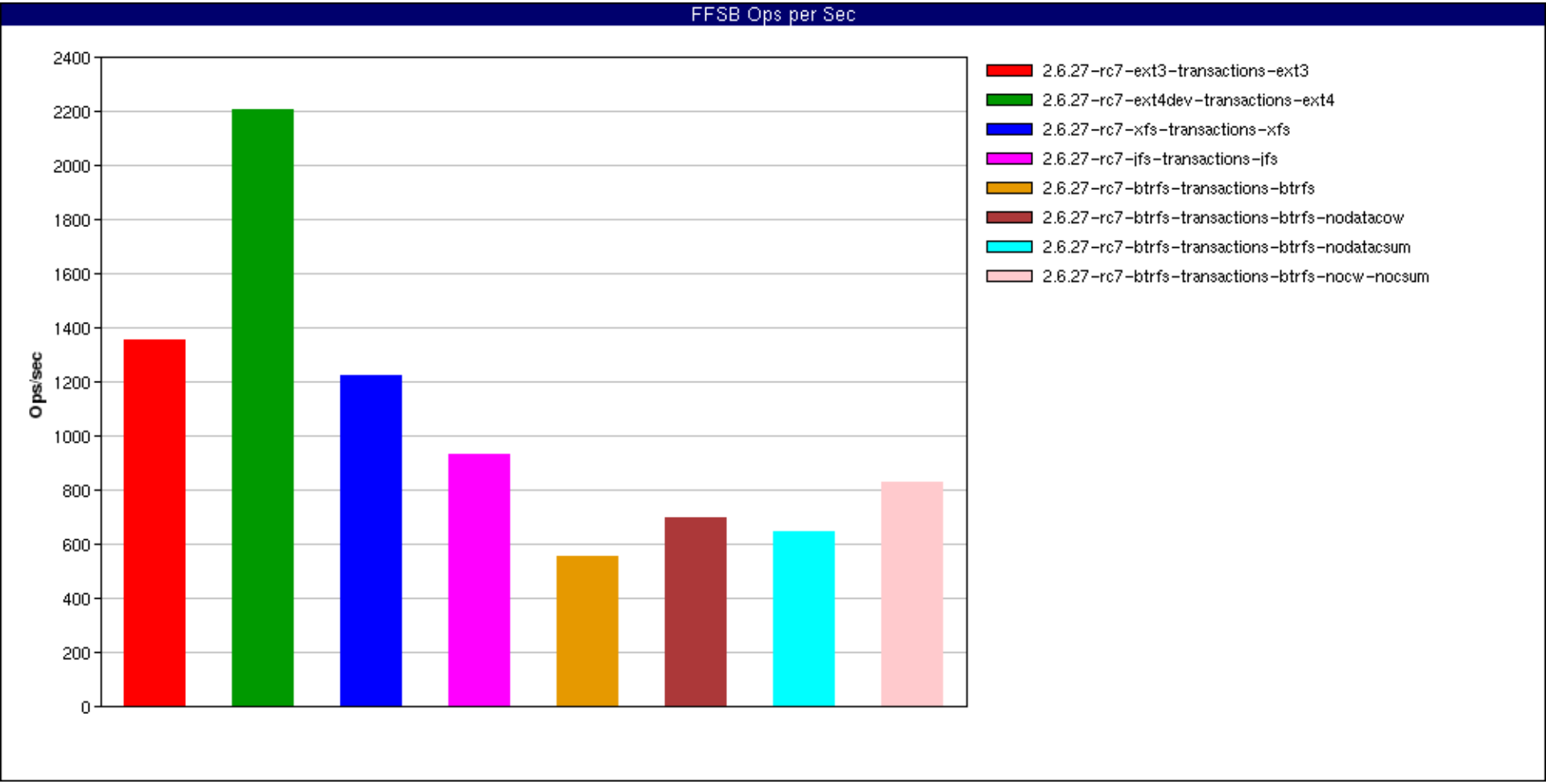
Large File Seq Reads, Single Disk 8 threads



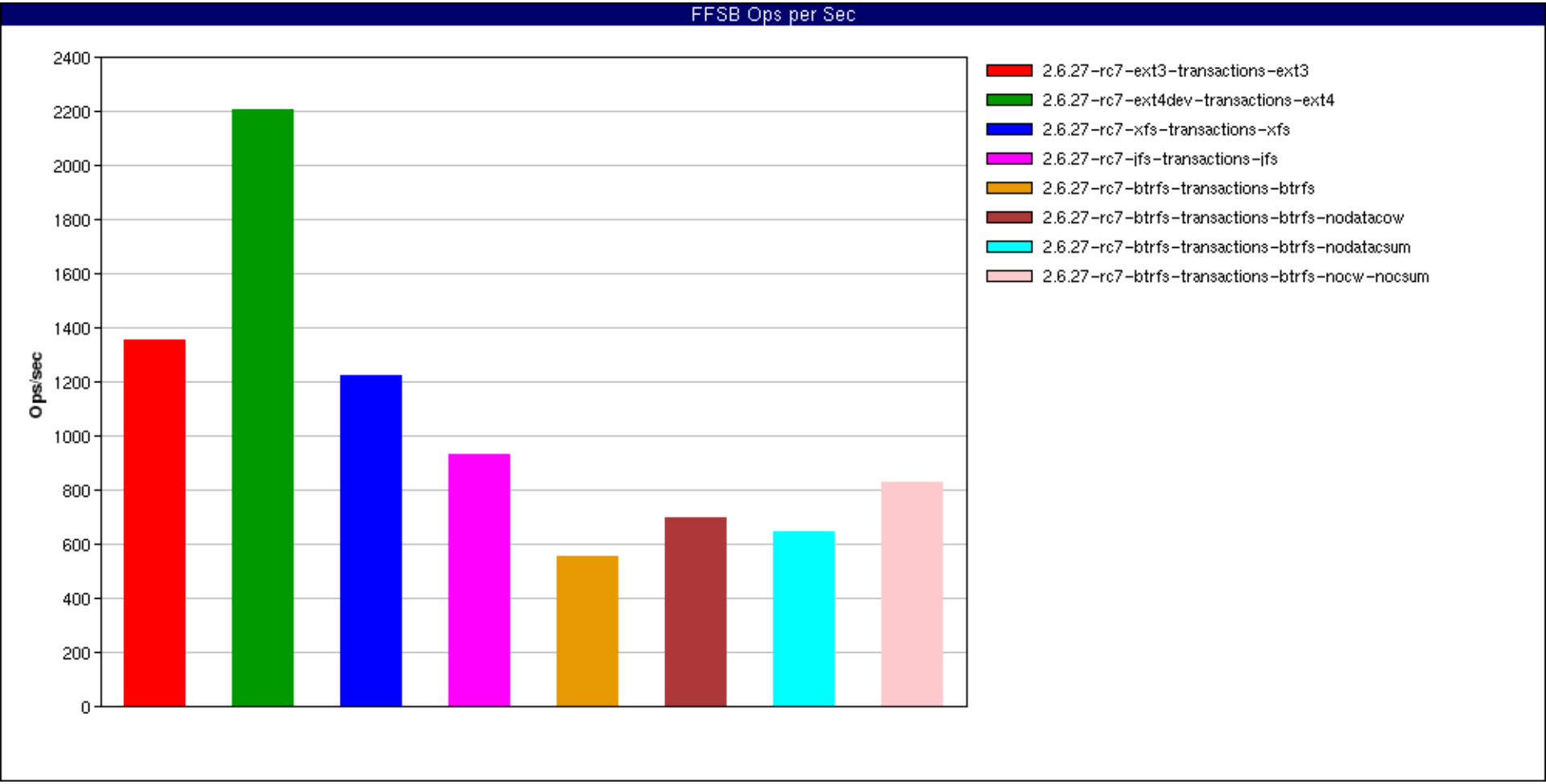
Large File Seq Reads, Single Disk 32 threads



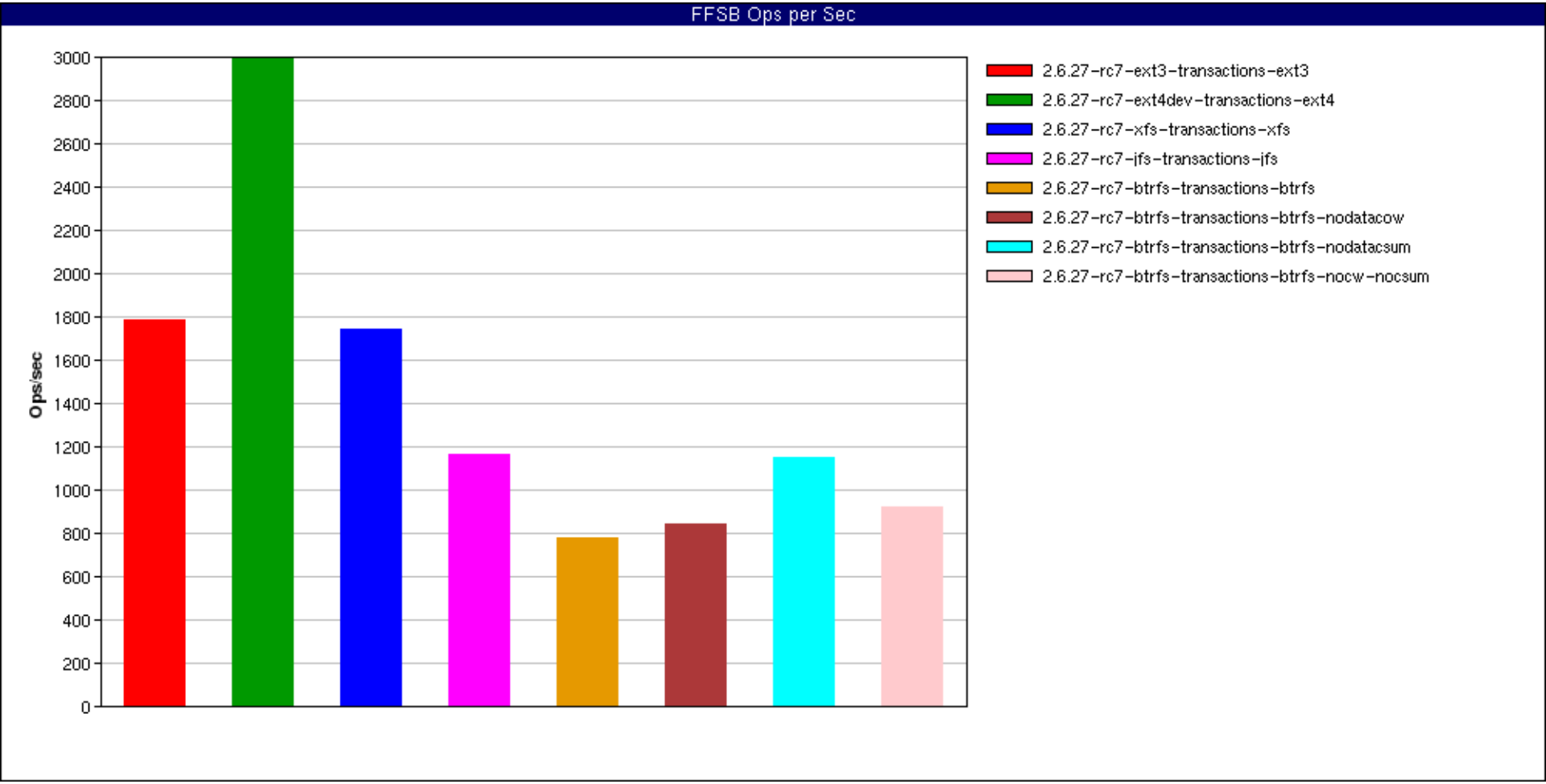
Mail Server Simulation, Single Disk, 1 thread



Mail Server Simulation, Single Disk, 8 threads



Mail Server Simulation, Single Disk, 32 threads



e2fsck Performance

- **Not something that we had explicitly engineered**
- **Improvements from**
  - Fewer extent tree blocks to read instead of indirect blocks
  - Uninitialized block groups means we don't have to read portions of the inode table

	e2fsck on ext3			e2fsck on ext4	
	time	MB read		time	MB read
Pass 1	192.3	1324		9.87	203
Pass 2	11.81	260		6.34	261
Pass 3	0.01	1		0.01	1
Pass 4	0.13	0		0.18	0
Pass 5	6.56	3		2.24	2
Total	211.1	1588		18.75	466

80 gig filesystem on a laptop drive

Using ext4

- **Need e2fsprogs 1.41.8**
- **Need 2.6.27 kernel or newer. Strongly recommend 2.6.30**
- **Need a filesystem to mount**
  - Can use existing unconverted ext3 (or ext2) filesystem.
  - Can convert an existing ext3 filesystem:
    - Tune2fs -O extents,huge_file,dir_nlink,dir_isize /dev/sdXX
    - Optional: can add uninit_bg and dir_index to the above, but then you must run “e2fsck -pD /dev/sdXX
  - Can create a fresh ext4 filesystem mke2fs -t ext4 /dev/sdXX
- **Shipping in some community distributions**
  - Fedora 11
  - Ubuntu 9.04 (but must upgrade to a mainline kernel)

Getting involved

- **Mailing list: linux-ext4@vger.kernel.org**
- **latest ext4 patch series**
  - git://git.kernel.org/pub/scm/linux/kernel/git/tytso/ext4.git
  - http://www.kernel.org/pub/scm/linux/kernel/git/tytso/ext4.git
  - 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
  - Import and improve content from http://kernelnewbies.org/Ext4
- **Weekly conference call; minutes on the wiki**
  - Contact us if you'd like dial in
- **IRC channel: irc.oftc.net, /join #ext4**

The Ext4 Development Team

- Alex Thomas (Sun)
- Andreas Dilger (Sun)
- Theodore Tso (IBM/Linux Foundation)
- Mingming Cao (IBM)
- Suparna Bhattacharya (IBM)
- Dave Kleikamp (IBM)
- Badari Pulavarathy (IBM)
- Avantikia Mathur (IBM)
- Aneesh Kumar (IBM)
- Eric Sandeen (Red Hat)
- Jan Kara (SuSE)

Legal Statement

- This work represents the view of the author(s) and does not necessarily represent the view of IBM or of the Linux Foundation.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.