Linux on zSeries Journaling File Systems

Volker Sameske (sameske@de.ibm.com)
Linux on zSeries Development
IBM Lab Boeblingen, Germany

Share Anaheim, California
February 27 – March 4, 2005
Session 9257

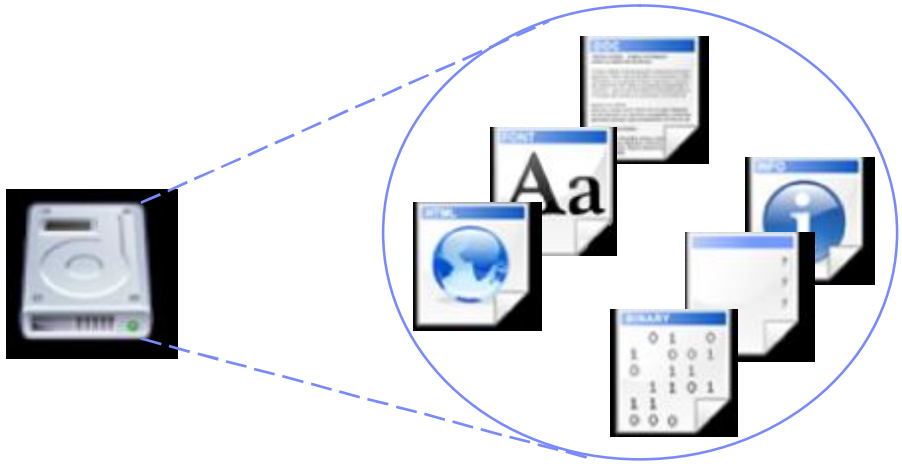
Agenda

- File systems.
  - Overview, definitions.
  - Reliability, scalability.
  - File system features.
  - Common grounds & differences.
- Volume management.
  - LVM, EVMS, MD.
  - Striping.
- Measurement results.
  - Hardware/software setup.
  - throughput.
  - CPU load.

A file system should...

- ...store data
- ...organize data
- ...administrate data
- ...organize data about the data
- ...assure integrity
- ...be able to recover integrity problems
- ...provide tools (expand, shrink, check, ...)
- ...be able to handle many and large files
- ...be fast
- ...

File system - definition



- Informally
  - The mechanism by which computer files are stored and organized on a storage device.

- More formally,
  - A set of abstract data types that are necessary for the storage, hierarchical organization, manipulation, navigation, access and retrieval of data.

Why a journaling file system?

- Imagine your Linux system crashs while you are saving an edited file:
  - The system crashs after the changes have been written to disk
    → good crash
  - The system crashs before the changes have been written to disk
    → bad crash
    but bearable if you have an older version
  - The sytem crashs just in the moment your data will be written:
    → very bad crash
    your file could be corrupted and in worst case the file system could be corrupted

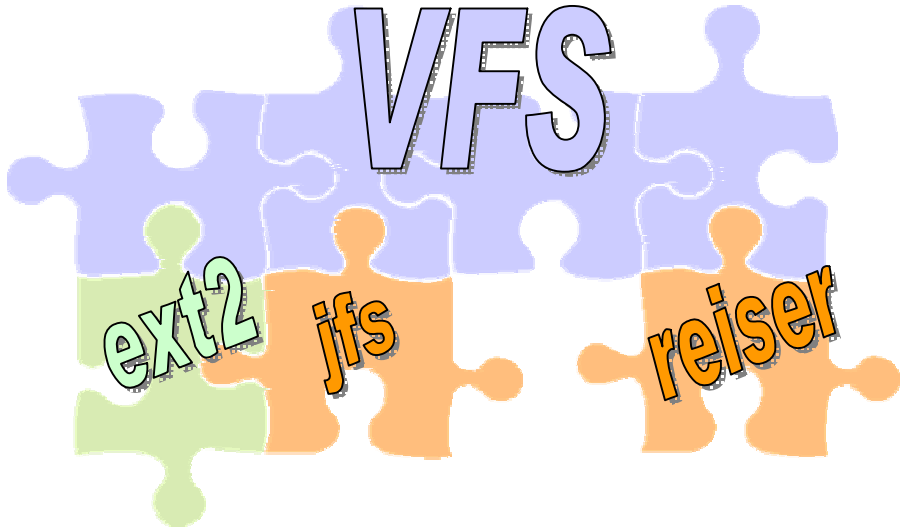
→ That's why you need a journal

Some file system terms

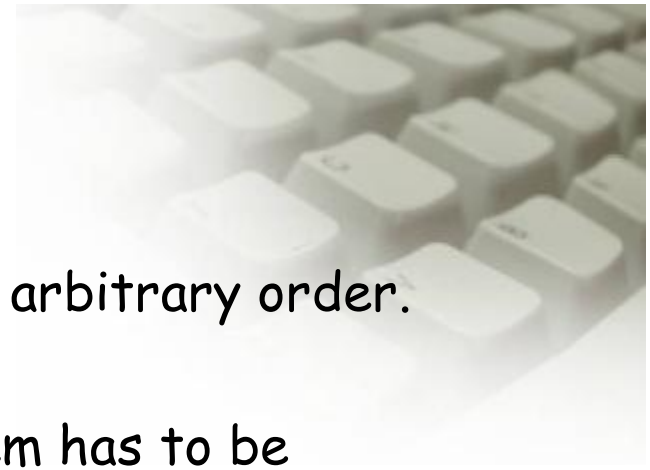
- Meta data
  - "Data about the data"
  - File system internal data structure (e.g. inodes)
  - Assures all the data on disk is properly organized and accessible.
- Inode
  - Contains various information about a file
  - E.g. Size or date and time of creation
  - Contains pointers to the data
- Journal
  - On-disk structure containing a log
  - Stores current meta data changes

Virtual file system (VFS)

- Software layer in the kernel.
- Provides the file system interface to user space programs.
- Allows coexistence of different file system implementations.
- File system independent.
- All file systems (ext2, ext3, jfs, reiser, …) provide certain VFS routines.
- Performs standard actions, equal for all file systems.



File systems – reliability



- Non-journaling file systems.
  - Data and meta-data is written directly and in arbitrary order.
  - No algorithm to ensure data integrity.
  - After crash, complete structure of file system has to be checked to ensure integrity.
  - File system check times depend on size of file system.

  → Long and costly system outages, risk of data loss.

- Journaling file systems.
  - Data integrity ensured.
  - In case of system crash journal has to be replayed to recover consistent file system structure.
  - File system check time depends on size of journal.

  → Shorter system outages, no data loss.

File systems – performance / scalability

- Non-journaling file systems.
  - File sizes, partition sizes and number of directory entries are limited.
  - Inadequate performance when handling huge storage capacities.
  - Block size – choose between space waste or performance.

  → Doesn't scale as expected, no journal overhead

- Journaling file systems.
  - Large files, larger directories, new organization.
  - Scale up easily with thousands of files.
  - Good behaviour with small and big files.
  - Dynamic i-node allocation (not in ext3 yet).

  → Scales with file system, journal overhead

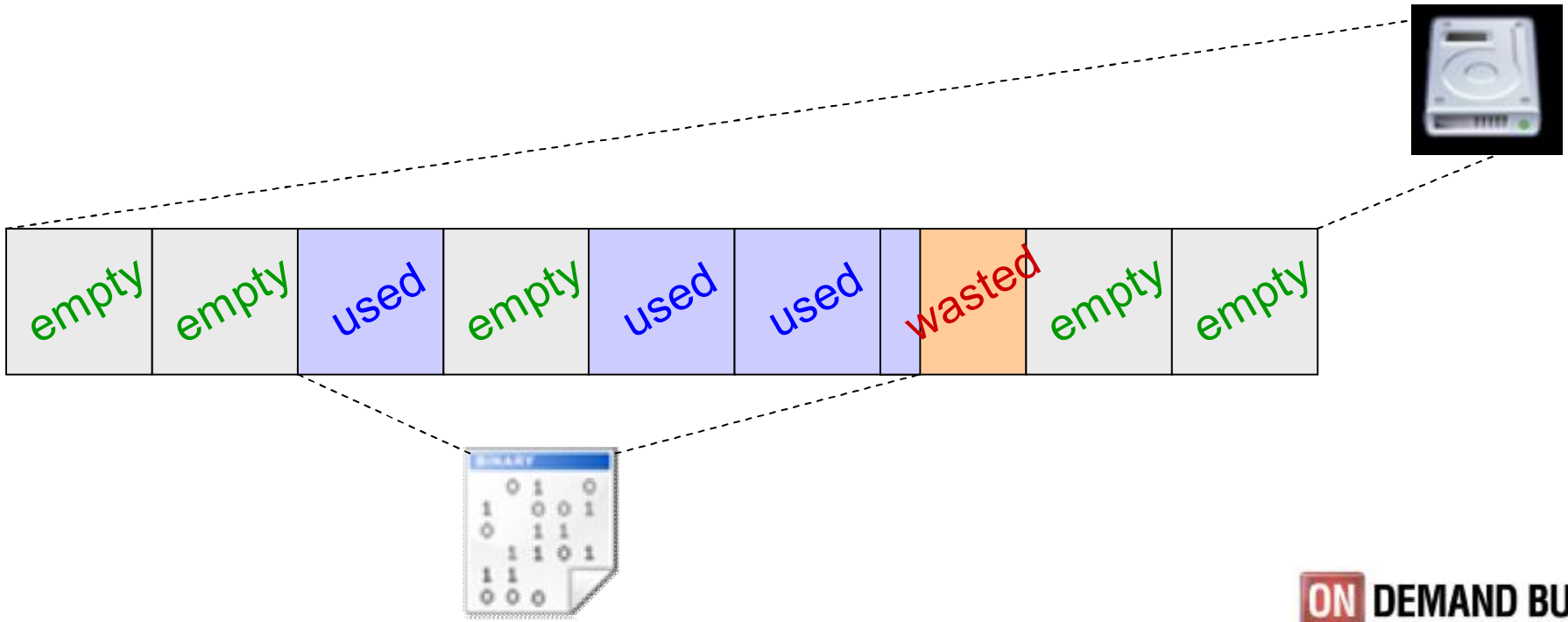
Issues with journaling file systems

- Stability.
  - Quite stable, but not so mature as other file systems.
- Migration.
  - Effort to spend.
  - Often one way migration (not ext2 ↔ ext3).
- Performance.
  - Some extra work to do.
  - Room for improvements.

→ Needs extra resources.

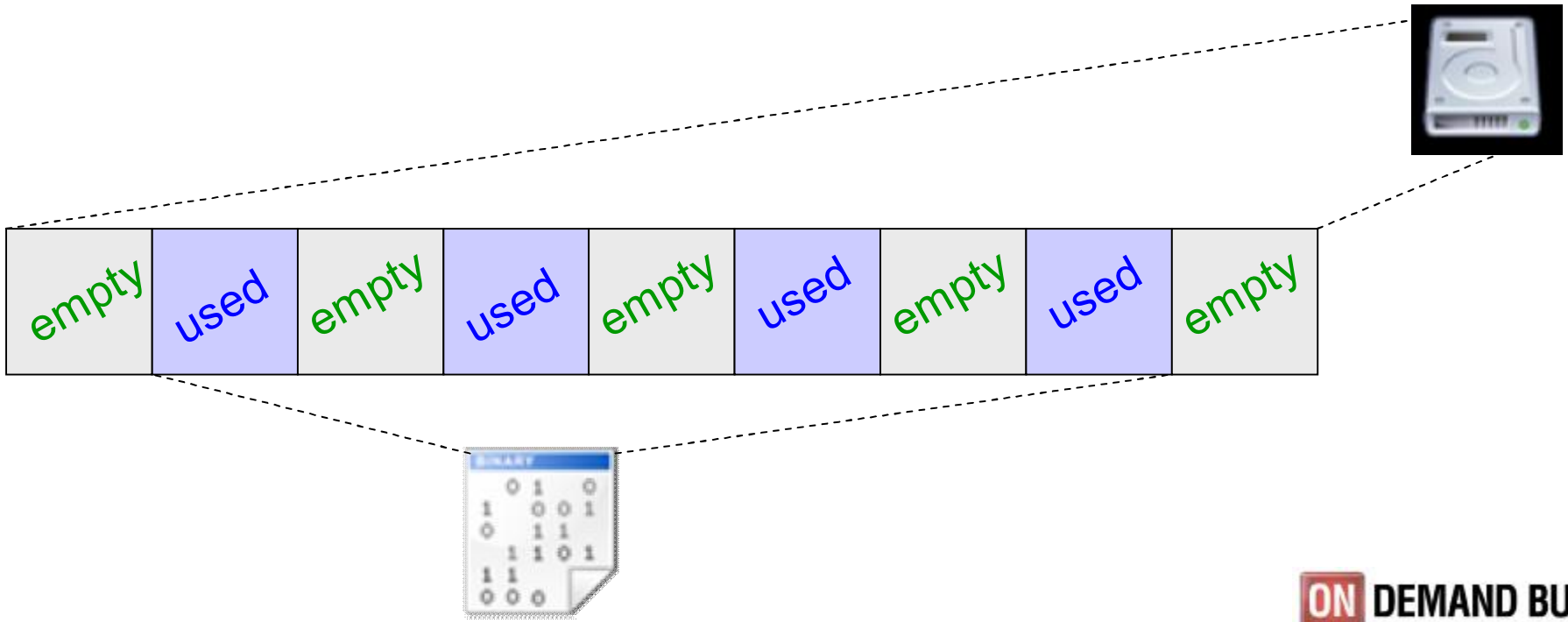
File system problems

- Internal fragmentation.
  - The logical block is the minimum allocation unit.
  - Average loss of space when blocking data into blocks of fixed size.
  - Problems with the ends of files not filling a whole block.



File system problems

- External fragmentation.
  - Logical blocks of a file are scattered all over the disk.
  - Operations over that file will be slower.
  - More hard-disk header movements are needed.



Considered file systems

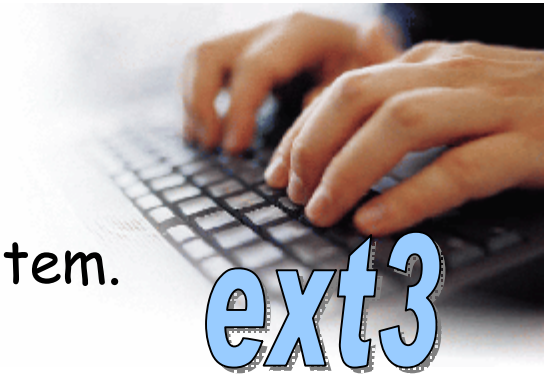
- Second extended file system (ext2).
- Third extended file system (ext3).
- Journaling file system (jfs).
- Reiser file system (reiserfs).
- Extended file system (xfs).

- All file systems have more or less a fairly extensive set of user space tools - for dumping, restoring, repairing, growing, snapshotting, tools for using ACLs and disk quotas, and a number of other utilities.

Second Extended File System (ext2)

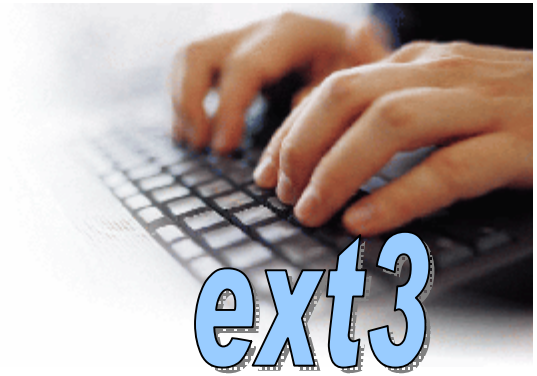
- ext2 was most popular Linux file system for years.
- "Old-timer."
- Heavily tested ("rock-solid").
- Faster than other file systems.
- Easy upgradeability.
- e2fsck analyzes file system data after system outage.
  - Meta data is brought into a consistent state (lost+found).
  - E2fsck analyzes the entire file system.
  - Recovery takes significantly longer than checking a journal.
  - Recovery time depends on file system size.
- No choice for any server that needs high availability.
- http://e2fsprogs.sourceforge.net/ext2.html

Third Extended File System (ext3)



- Developed by Stephen Tweedie and others.
- ext3 is a journaling extension to the ext2 file system.
- Shares ext2 on-disk and meta data format.
- Seamless migration from ext2.
- Large number of existing ext2 systems.
- Supports full data journaling.
- Resizing (only with unmount) possible.
- Easy downgrade from ext3 to ext2 (remount).
- Converting from ext2 to ext3 involves two separate steps:
  - Creating the journal.
    - tune2fs -j creates an ext3 journal with the default parameters.
    - Other parameters are size= and device= (man page).
  - Specifying the file system type in /etc/fstab.

Third Extended File System (ext3) – Cont.



- 3 modes, specified at mount time or with tune2fs.
- data=journal mode.
  - Maximum security and data integrity.
  - Both meta data and data are journaled.
- data=ordered mode (default).
  - Ensures both data and meta data integrity, but journaling only for meta data.
  - Data blocks are written before meta data update.
- data=writeback.
  - Data blocks are written after meta data update.
  - Performance advantages, but recovery "problems".
- http://www.zipworld.com.au/~akpm/linux/ext3/

ON DEMAND BUSINESS™

Journaling File System (JFS)

- Developed by IBM Austin Lab.
- Ported from OS/2 Warp Server.
- "meta data only" approach.
- Possibility to save data and journal to separate disks.
- Dynamic inode allocation, relatively large inodes (512 byte).
- Stores as much info as possible within inodes.
- Supports large files and partitions.
- Modifiable journal size.
- Can ignore case differences on file names
- Two different directory organizations.
  - For small directories, it allows the directory's content to be stored directly into its inode.
  - For larger directories, it uses B-Trees.
- http://www.ibm.com/developerworks/oss/jfs/index.html

Reiser File System (reiserfs)

- First journaling file system under Linux
- Developed by a group around Hans Reiser (Namesys, SUSE).
- Default choice when installing SUSE Linux Enterprise Server
- Only meta data journaling.
- Entire file system is one big fast B-Tree.
- No traditional inodes, but B-tree nodes.
- Dynamic node allocation.
- Stores file, directory and other data in the leaf nodes.
- Storage allocation in portions of the exact size.
- File data and meta data are stored next to each other.
- Small files can be stored as part of the meta data.

Reiser File System – Cont.

- Variable journal size and variable journal transactions size.
- Key assets.
  - Disk space utilization.
  - Disk access performance.
  - Fast crash recovery.
- Disk space optimization algorithm ("tail merging").
- Extra calculations to re-arrange data when file size changes.
- Online enlargement of file system.
- Three performance tuning options at mount time.
  - no unhashed relocation.
  - hashed relocation.
  - no border allocation.
- http://www.namesys.com/

Reiser4 File System

- Published 08/2004
- Important parts are rewritten from scratch.
- Atomic file system.
- Based on plug-ins.
- Space efficient, squishes small files together.
- Different tree handling.
  - Dancing trees instead of balanced trees.
  - Performance improvements.
- DARPA founded development.
  - Defense Advanced Research Projects Agency
  - Architected for high grade security.
- It will take some time until it will become stable as necessary for production.

Extended File System (XFS)

- Originally file system for SGI IRIX OS, early 1990s.
- Good at manipulating large files.
- Complete toolset, e.g. defragmentation utility
- XFS has variable sized inodes to store non-meta data.
- Only meta data journaling.
- Allocation groups.
  - "File systems in a file system."
  - Eight or more linear regions of equal size.
  - Own inode and free disk space management per AG.
  - Allocation groups are independent.
  - Simultaneously addressing.

Extended File System (XFS) – Cont.

- Use of B-trees for free space and inodes inside allocation groups.
- "Delayed allocation".
  - Pending transactions are stored in memory.
  - Delayed decision, where to store data.
  - Delayed until the last possible moment.
  - Increased performance.
  - Data loss after a crash during a write is more severe.
- Preallocation to avoid file system fragmentation.
- Possibility to use a separate log device.
- IRIX XFS disks could be used with Linux.
- http://oss.sgi.com/projects/xfs/index.html

Expand And Shrink

- Expanding and shrinking volumes are common volume operations on most systems.

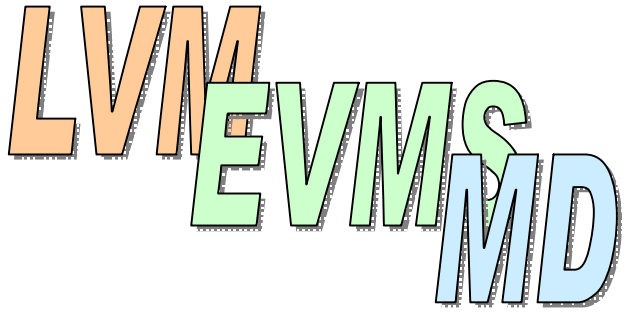
File System	Shrinks	Expands
ext2	Offline only	Offline only
ext3	Offline only	Offline only
JFS	No	Online only
ReiserFS	Offline only	Offline and online
XFS	No	Online only

Unique File System Features

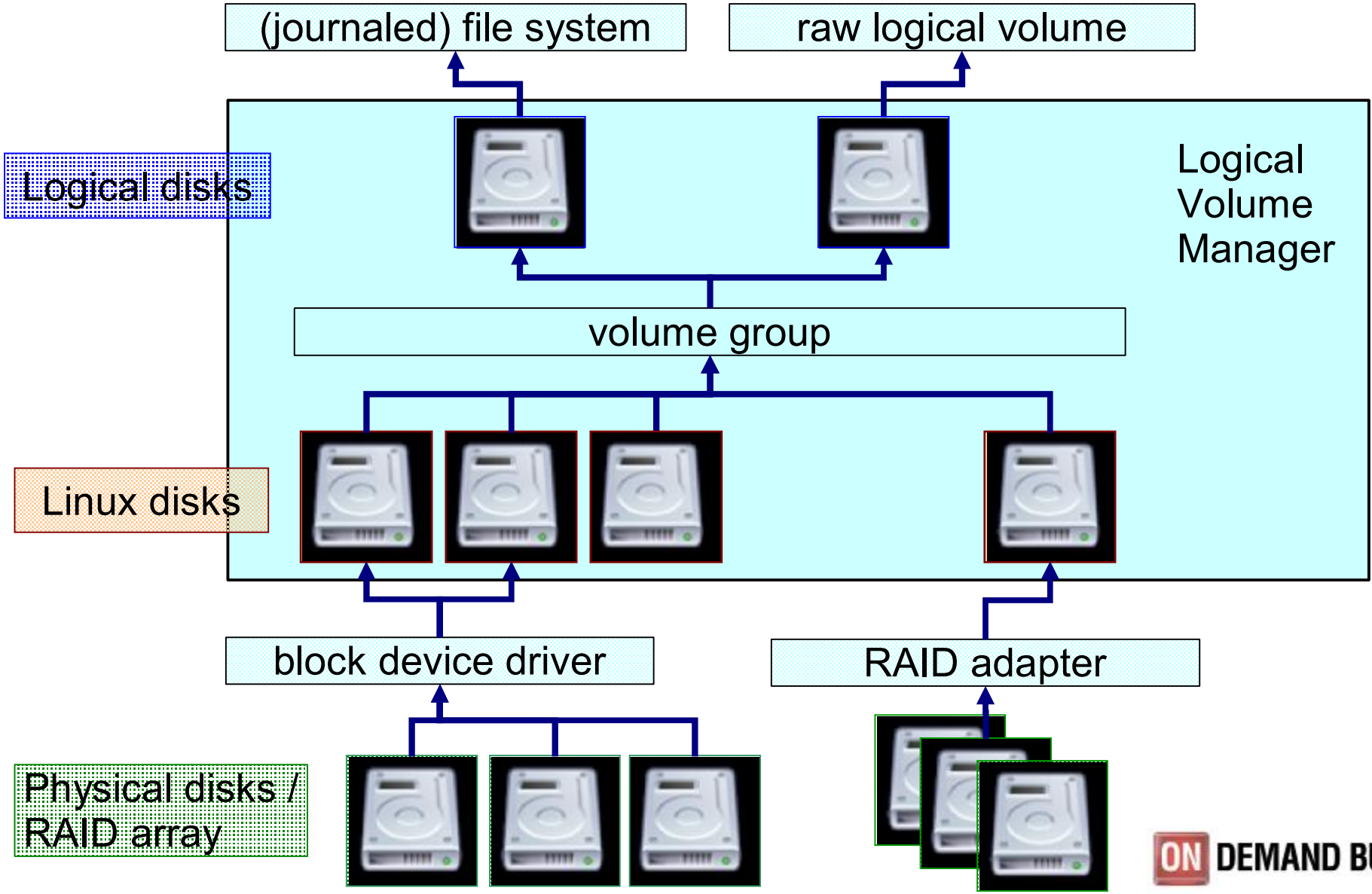
- ext2:
  - No journal, but fast and robust.
- ext3:
  - Small, simple, robust, forward/backward compatibility.
  - Data journaling.
- JFS:
  - Large inodes to store non-meta data (subdirectory names, symlinks, EAs, etc.)
  - Two different directory organizations.
- Reiserfs:
  - Space-efficient for small files (tail merging).
- XFS:
  - Designed to support large multi-processor systems.
  - Allocation groups.

Volume Management And Multipathing

- SLES8
  - LVM – Logical Volume Manager
- SLES9
  - Device Mapper subsystem in 2.6 kernel
  - EVMS – Enterprise Volume Management System
  - LVM – Logical Volume Manager
- RHEL3
  - MD
  - mdadm – multiple device administration

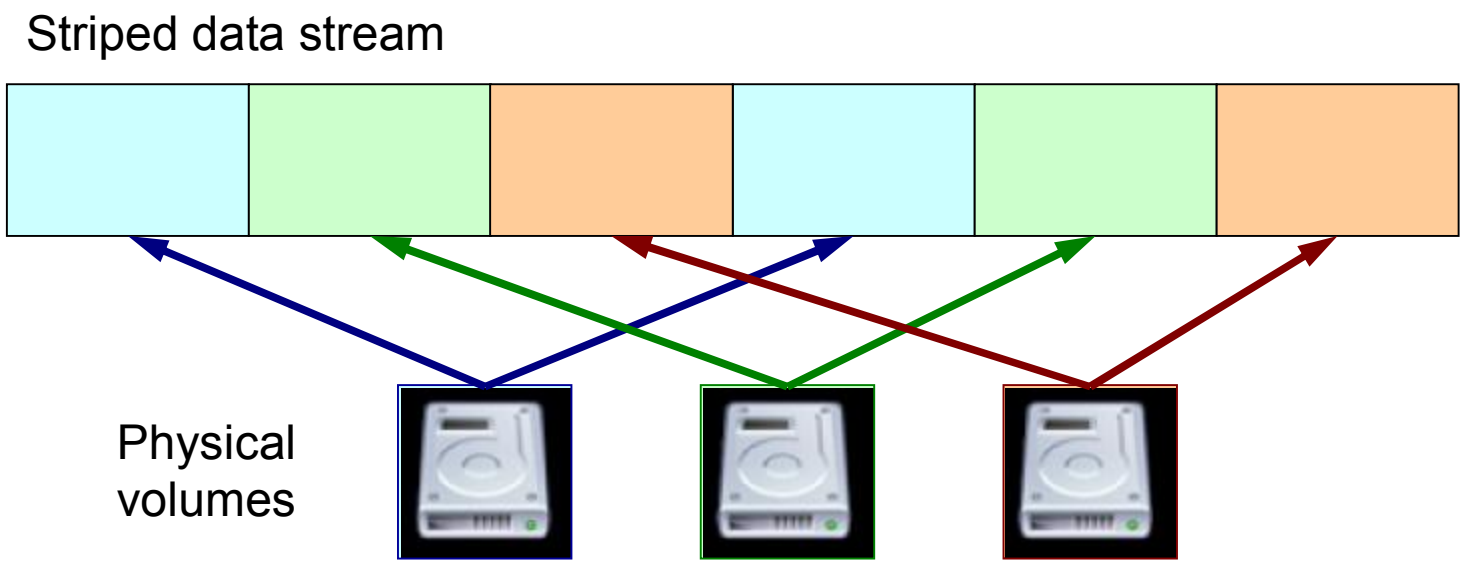


LVM System Structure



Improving Disk Performance With Striping.

- Technique for spreading data over multiple disks.
- With LVM and striping parallelism is achieved.



Measurement Setup

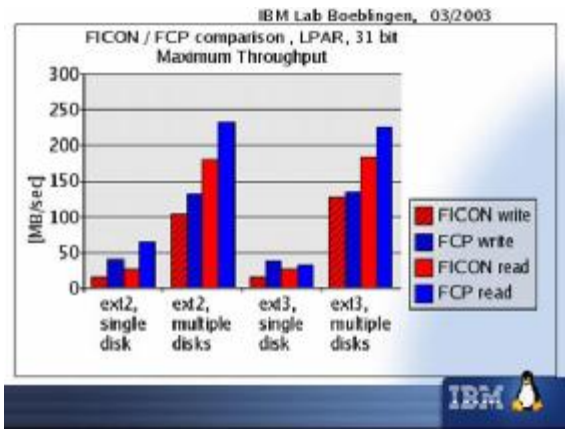
- SUSE SLES9 RC5
- Dbench 2.1

- **2084-316 (z990)**
  - 0.83ns (1200MHz)
  - 2 * 32 MB L2 Cache (shared)
  - 96 GB
  - 4 FCP channels
- **2105-F20 (ESS)**
  - 384 MB NVS
  - 16 GB Cache
  - 128 * 36 GB disks
  - 10.000 RPM
  - FCP

ON DEMAND BUSINESS™

Measurement Setup: Dbench File I/O

- File system benchmark.
- Version 2.1.
- Generates load patterns similar to Netbench.
- It does no networking calls.
- Does not require a lab of load generators to run.
- De-facto standard for generating load on the Linux VFS.



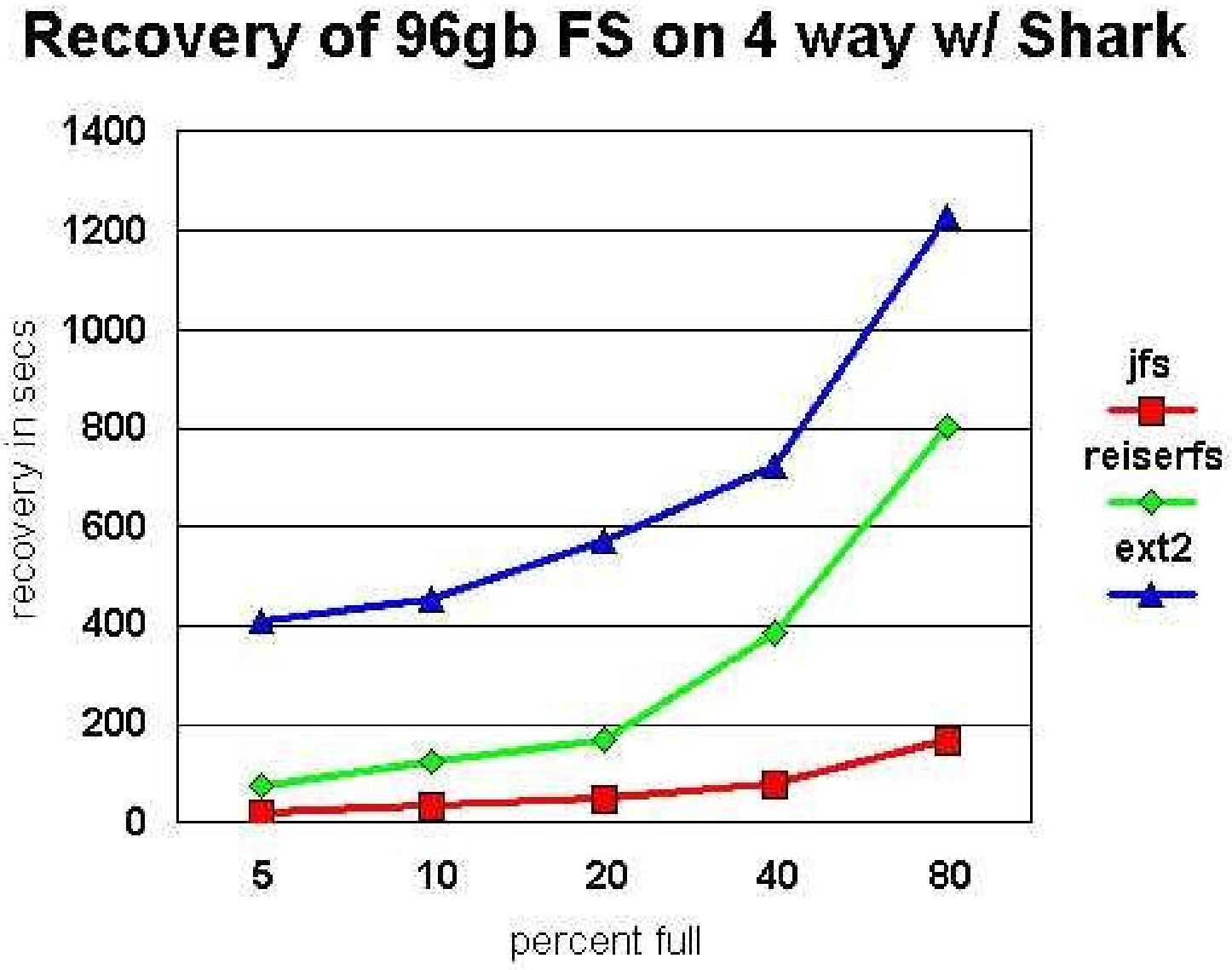
- Author: Andrew Tridgell.
- Released under the GNU Public License.

Things To Keep In Mind

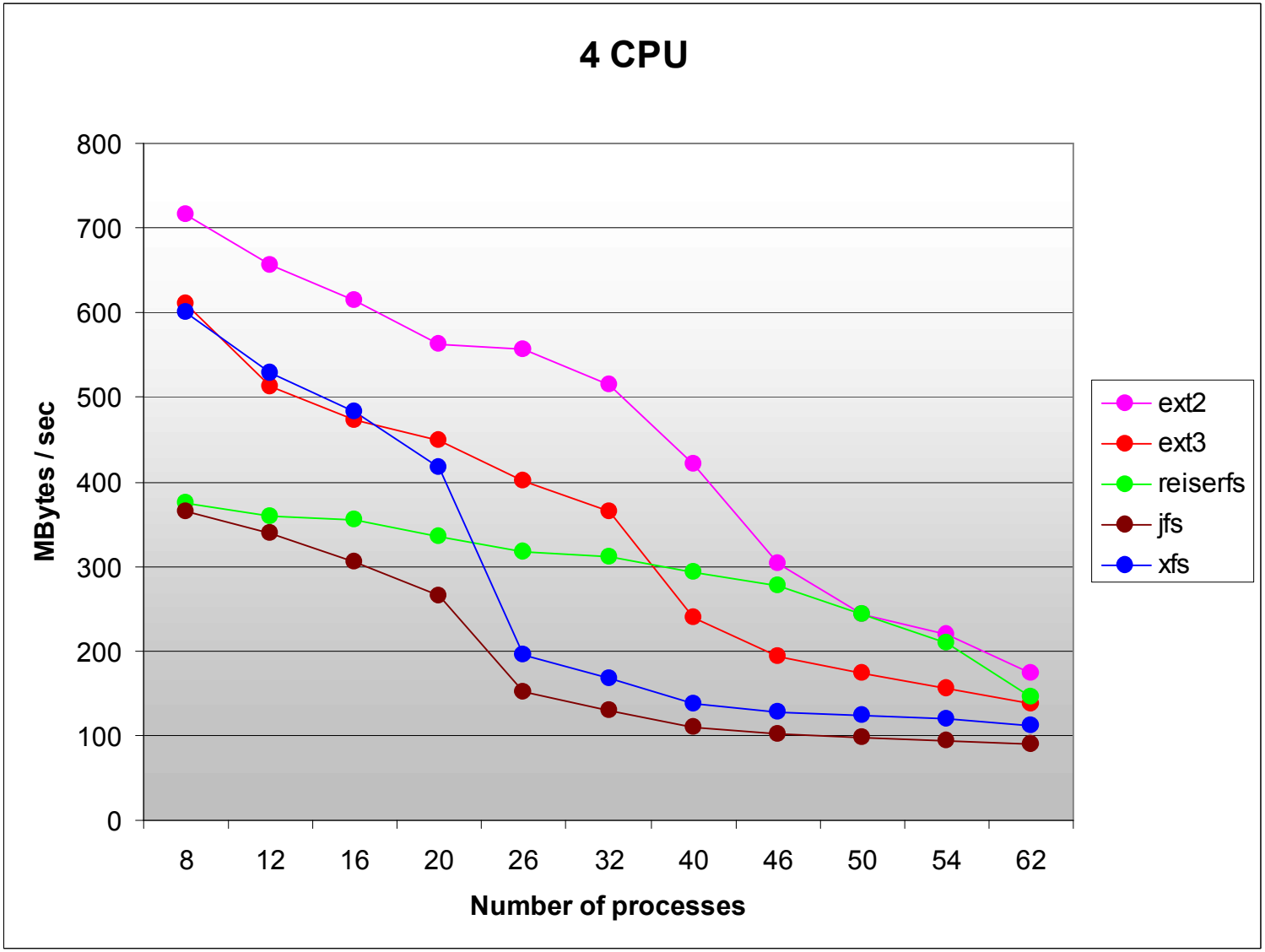
- Dbench is only one benchmark.
- Dbench throughput is a „mixed" throughput.
- We have used only one special hardware/software setup. There are many possibilities.
- Disk access under VM is slightly slower.
- Volume managers with striping are faster than single disks.
- Lots of different file system options.

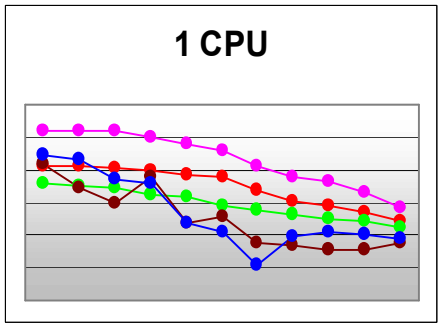
→ Many, many switches to tweak.

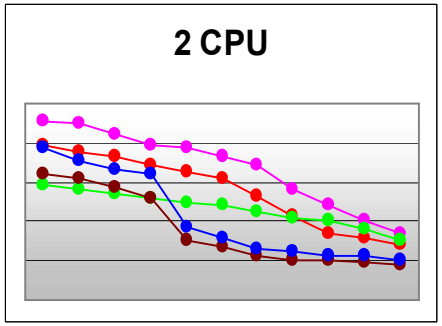
Recovery Times

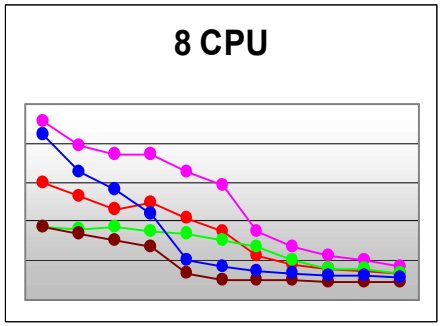


Throughput - FS Comparison - SCSI, LVM, 256MB



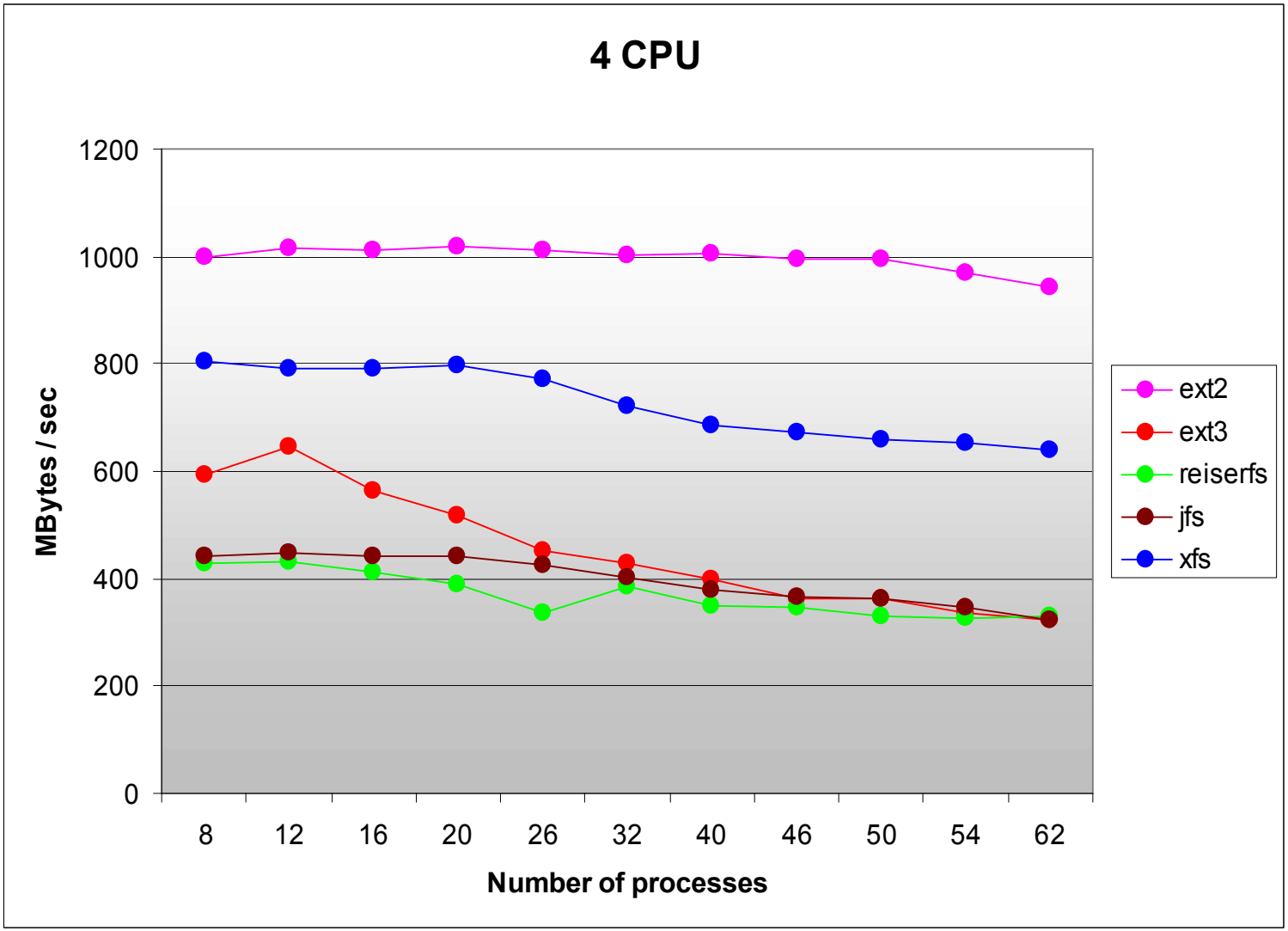


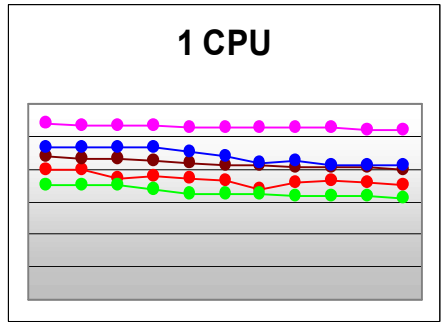


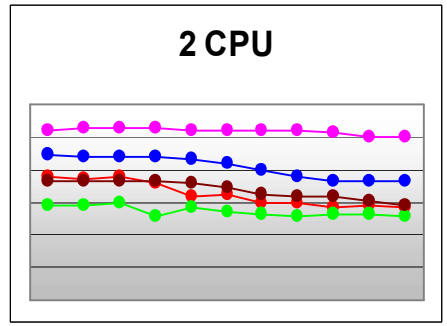


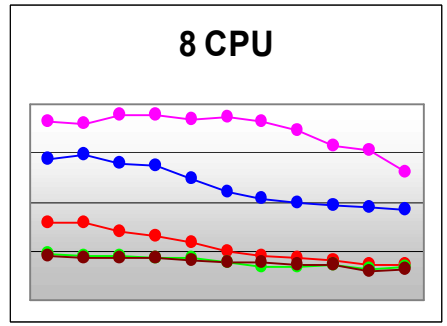
ON DEMAND BUSINESS™

Throughput - FS Comparison - SCSI, LVM, 2 GB



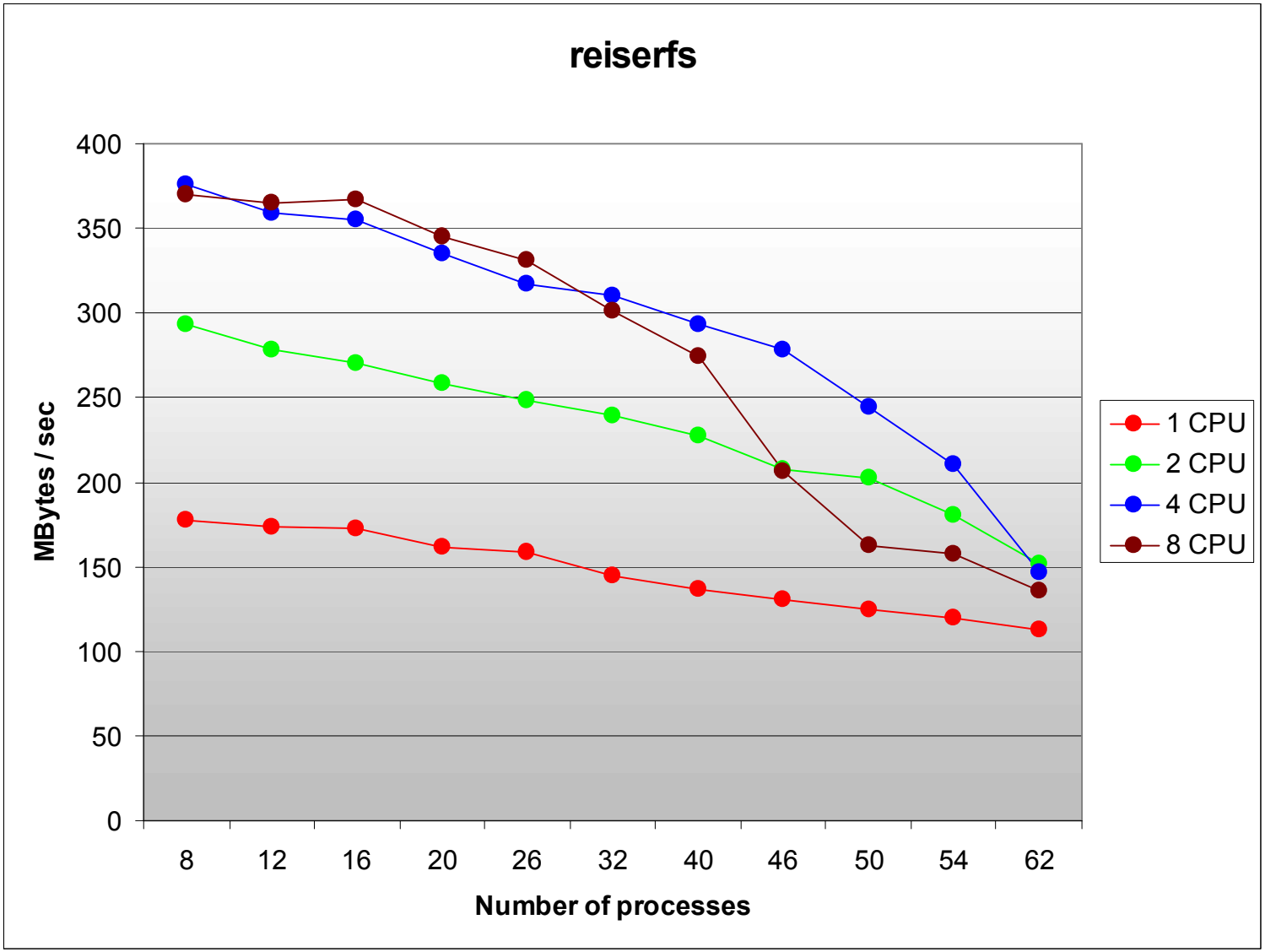


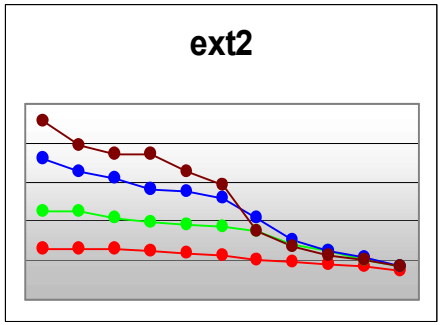


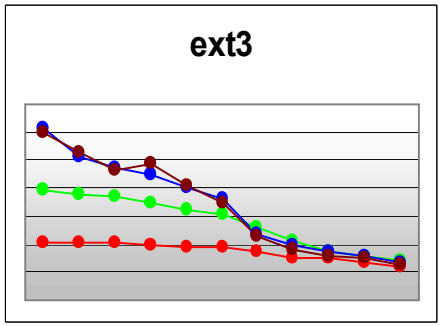


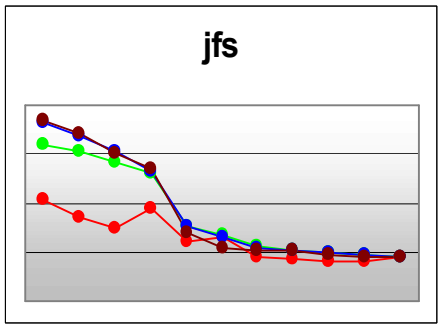
ON DEMAND BUSINESS™

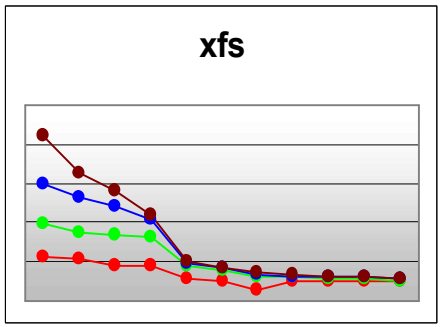
Throughput - SCSI, LVM, 256 MB



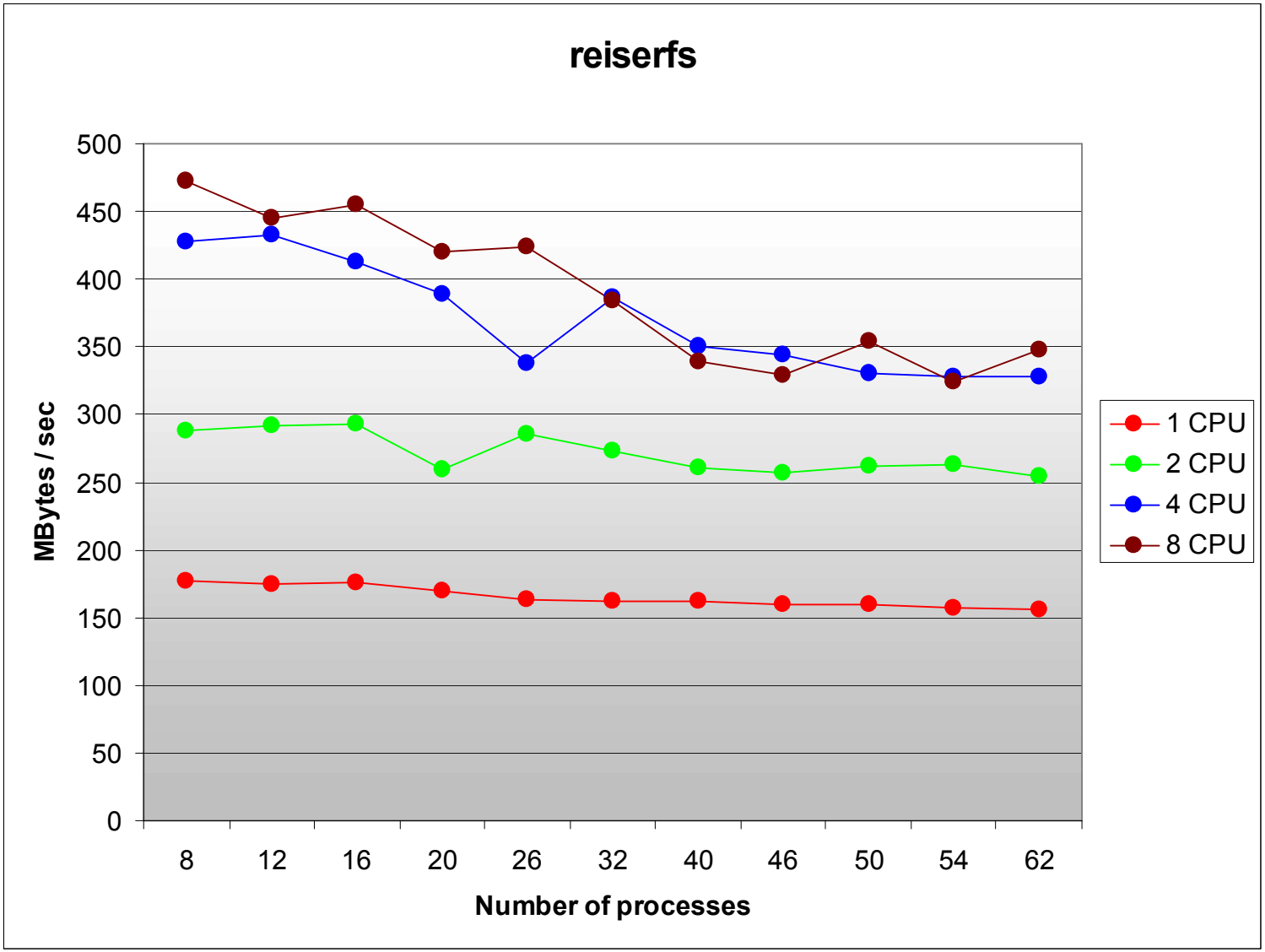


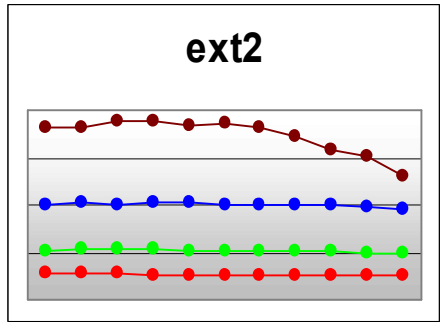


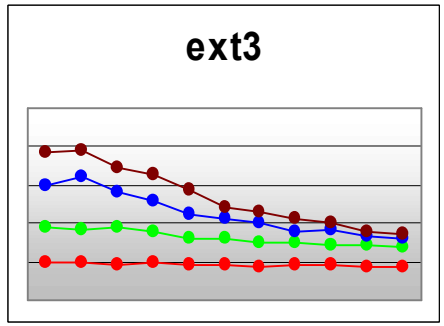


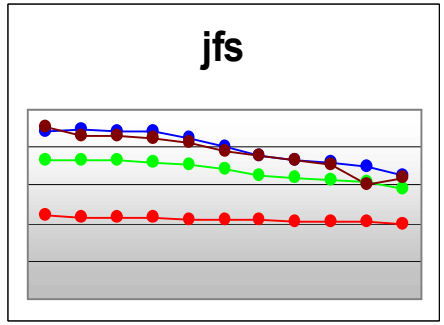


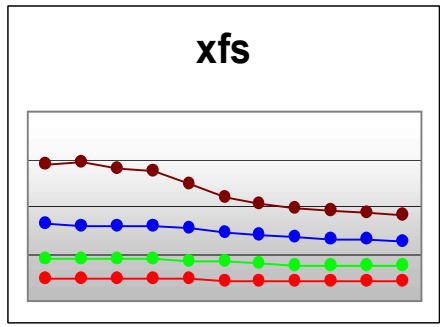
Throughput – SCSI, LVM, 2 GB



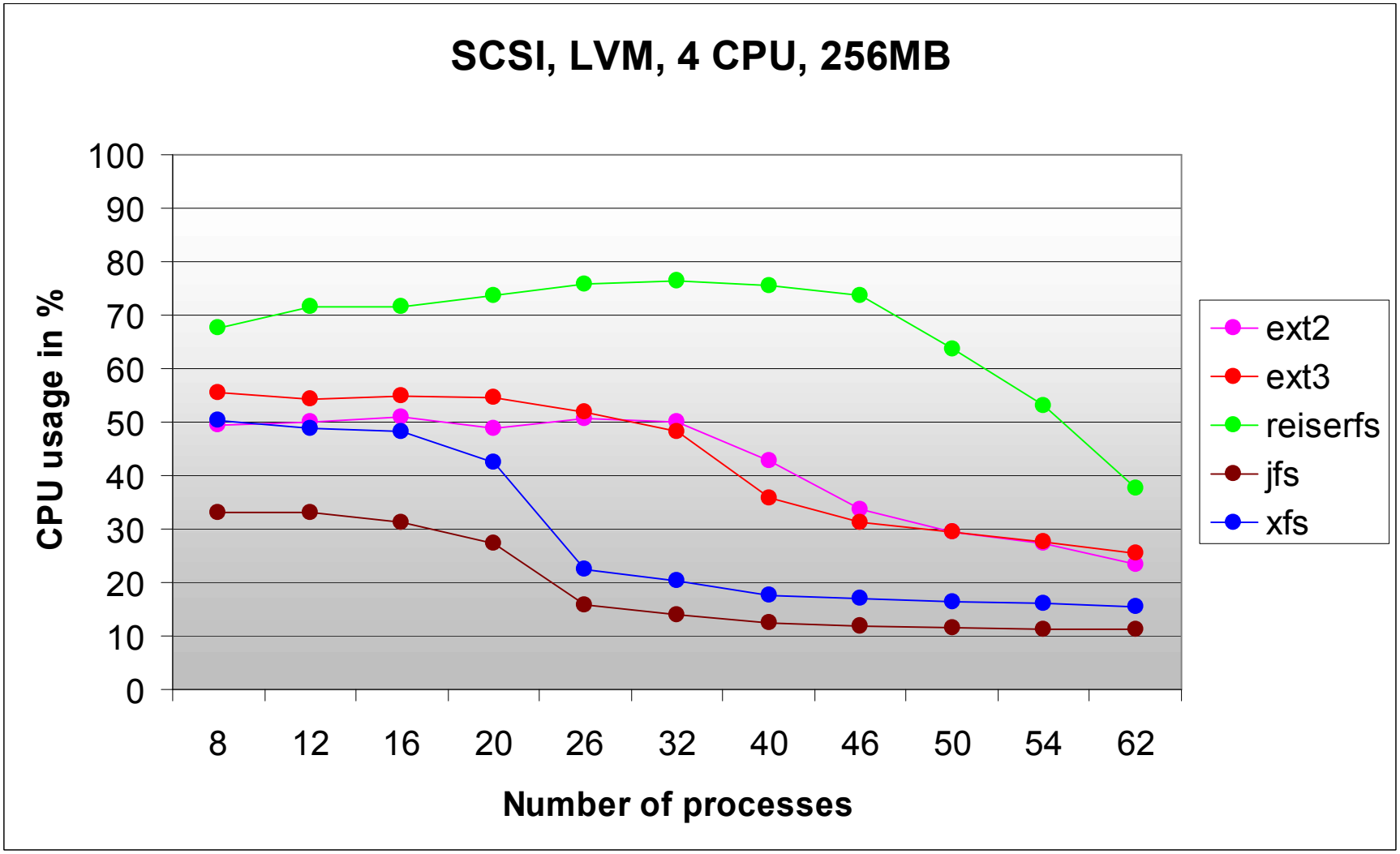




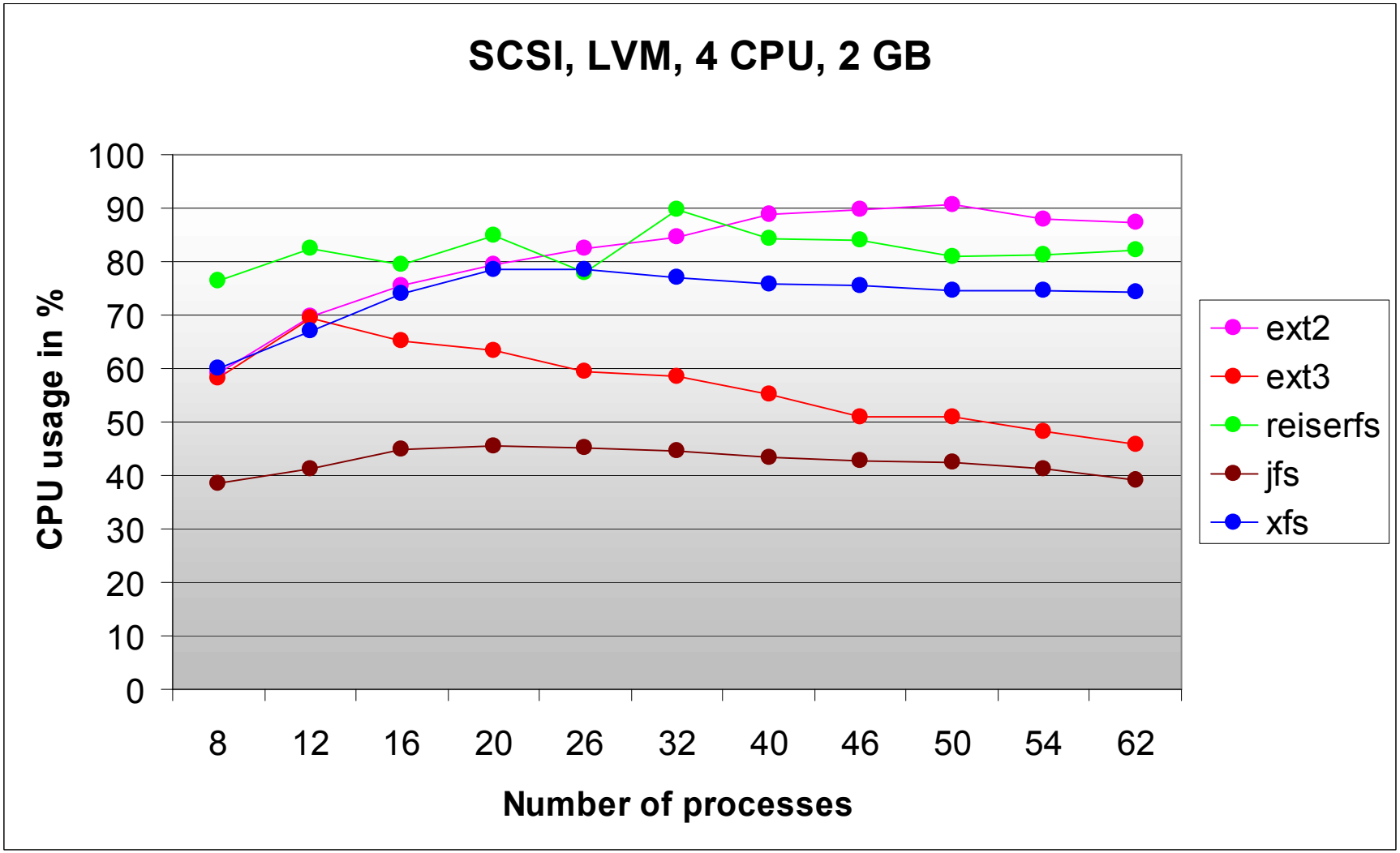




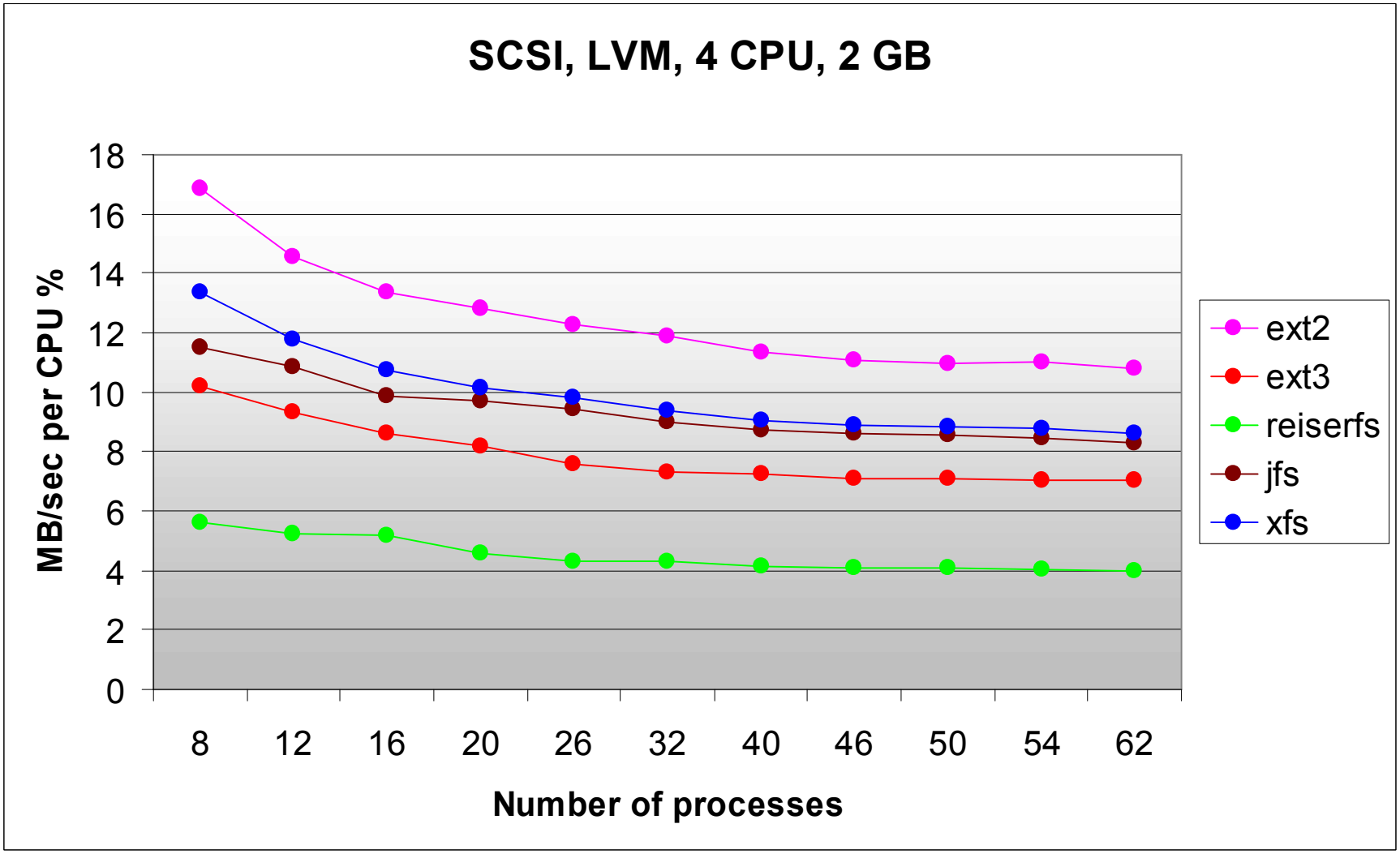
CPU Load – Comparison 265 MB



CPU Load – Comparison 2 GB



Throughput Per CPU Usage %



Summary

- Journaling file systems increase data integrity significantly.
- Journaling file systems dramatically reduce system outage times.
- Performance cost is at least 30%.
- Journaling file systems profit from LVM.
- 2.6 brings more improvements (increased throughput, reduced CPU load).
- File system choice requires detailed understanding of the write characteristics of your application:
  - only appends information to files,
  - writes to the middle of files,
  - creates and deletes many files.

→ In any case it is a tradeoff of consistency guarantees versus speed

Linux For zSeries Journaling File Systems

Questions ?

Trademarks

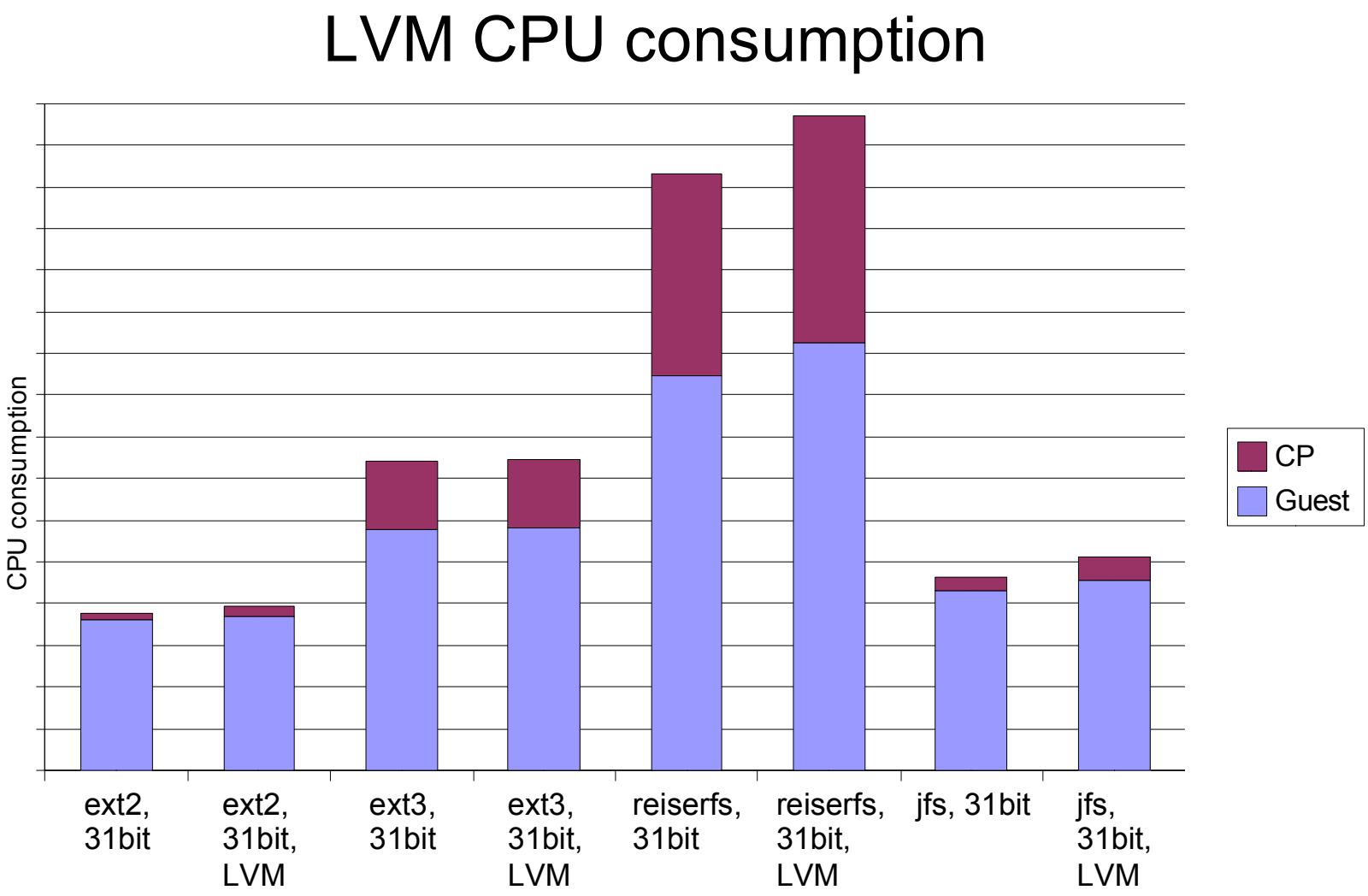
- **The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.**
  - AIX, e-business logo, on-demand logo, IBM, IBM logo, OS/390, PR/SM, z900, z990, z800, z890, zSeries, S/390, z/OS, z/VM, FICON, ESCON
- **The following are trademarks or registered trademarks of other companies.**
  - LINUX is a registered trademark of Linus Torvalds
  - Penguin (Tux) complements of Larry Ewing
  - Tivoli is a trademark of Tivoli Systems Inc.
  - Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
  - UNIX is a registered trademark of The Open Group in the United States and other countries.
  - SMB, Microsoft, Windows are registered trademarks of Microsoft Corporation.
- * All other products may be trademarks or registered trademarks of their respective companies.
- Notes:
  - Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
  - IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
  - All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved.
  - Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
  - This publication was produced in Germany. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
  - All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
  - Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

ON DEMAND BUSINESS™

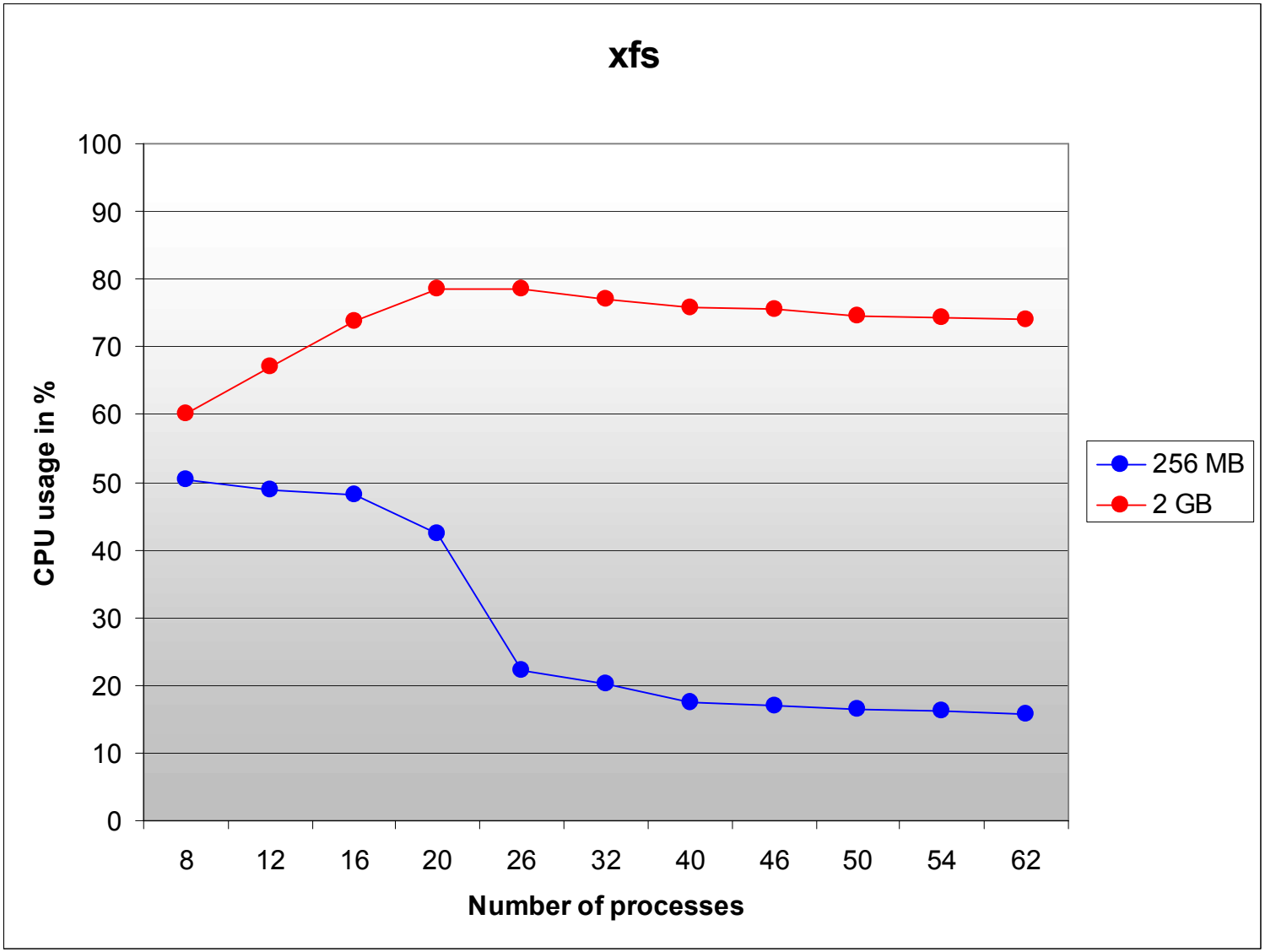
Linux For zSeries Journaling File Systems

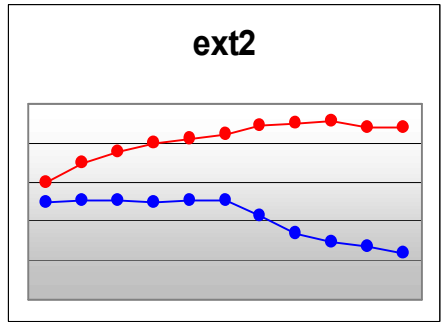
Backup charts

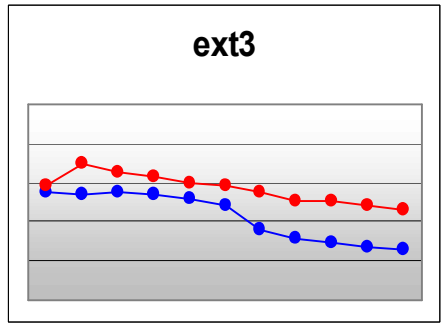
z/VM Overhead

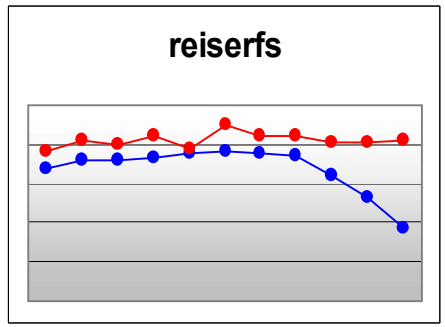


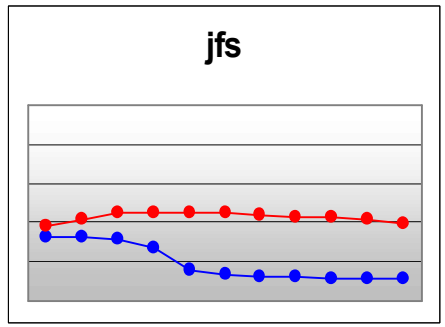
CPU Load – SCSI, LVM, 4 CPU



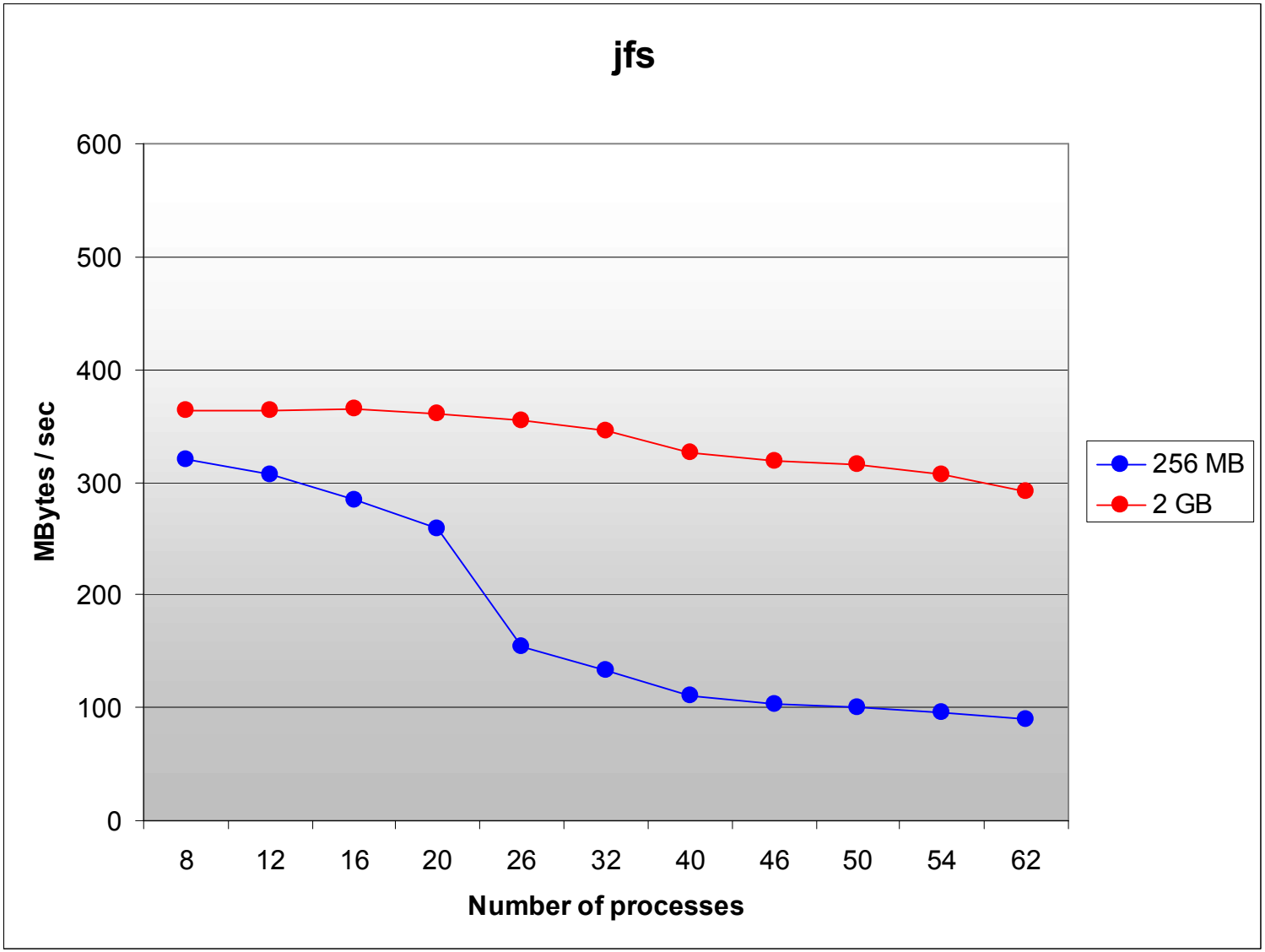


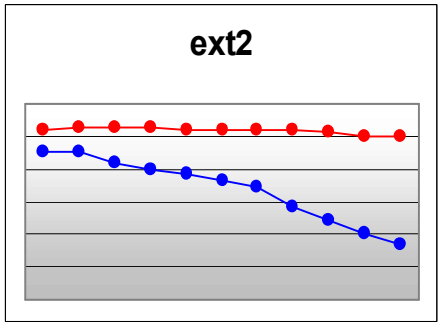


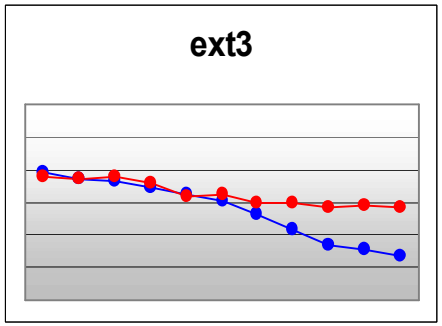


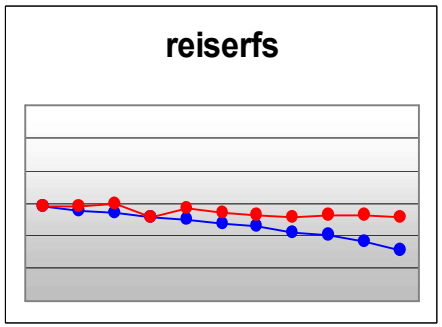


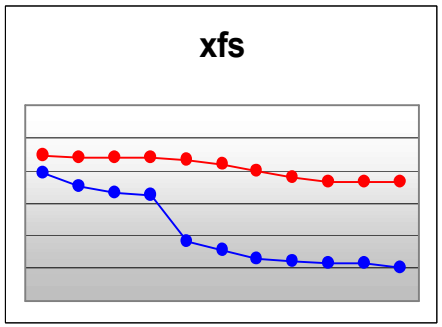
Throughput - SCSI, LVM, 2 CPU



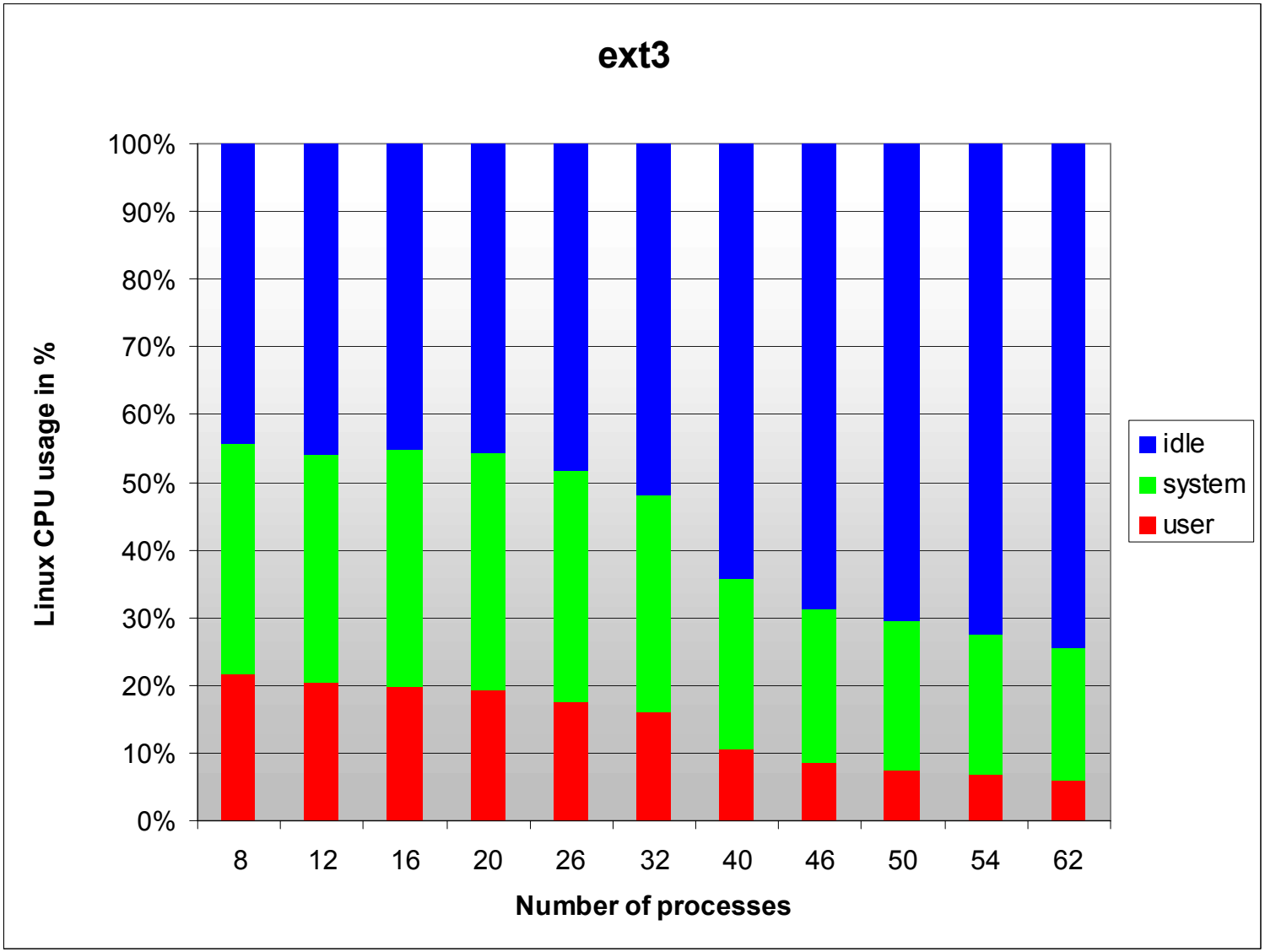


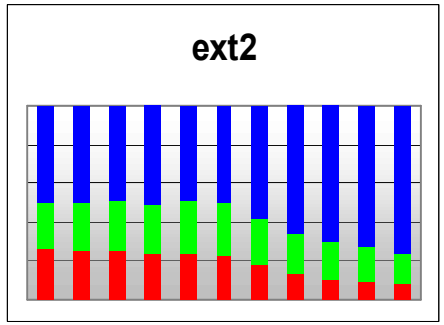


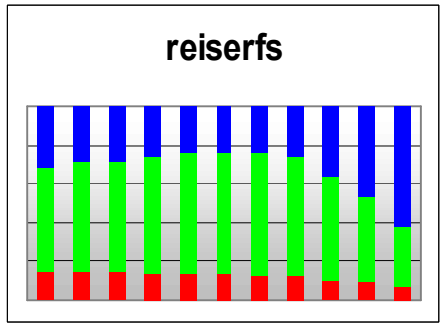


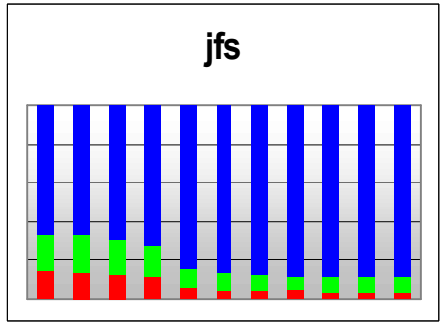


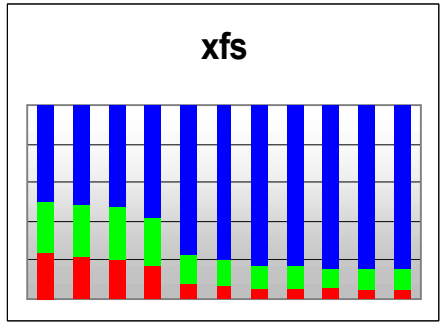
CPU Load – SCSI, LVM, 4 CPU, 256 MB



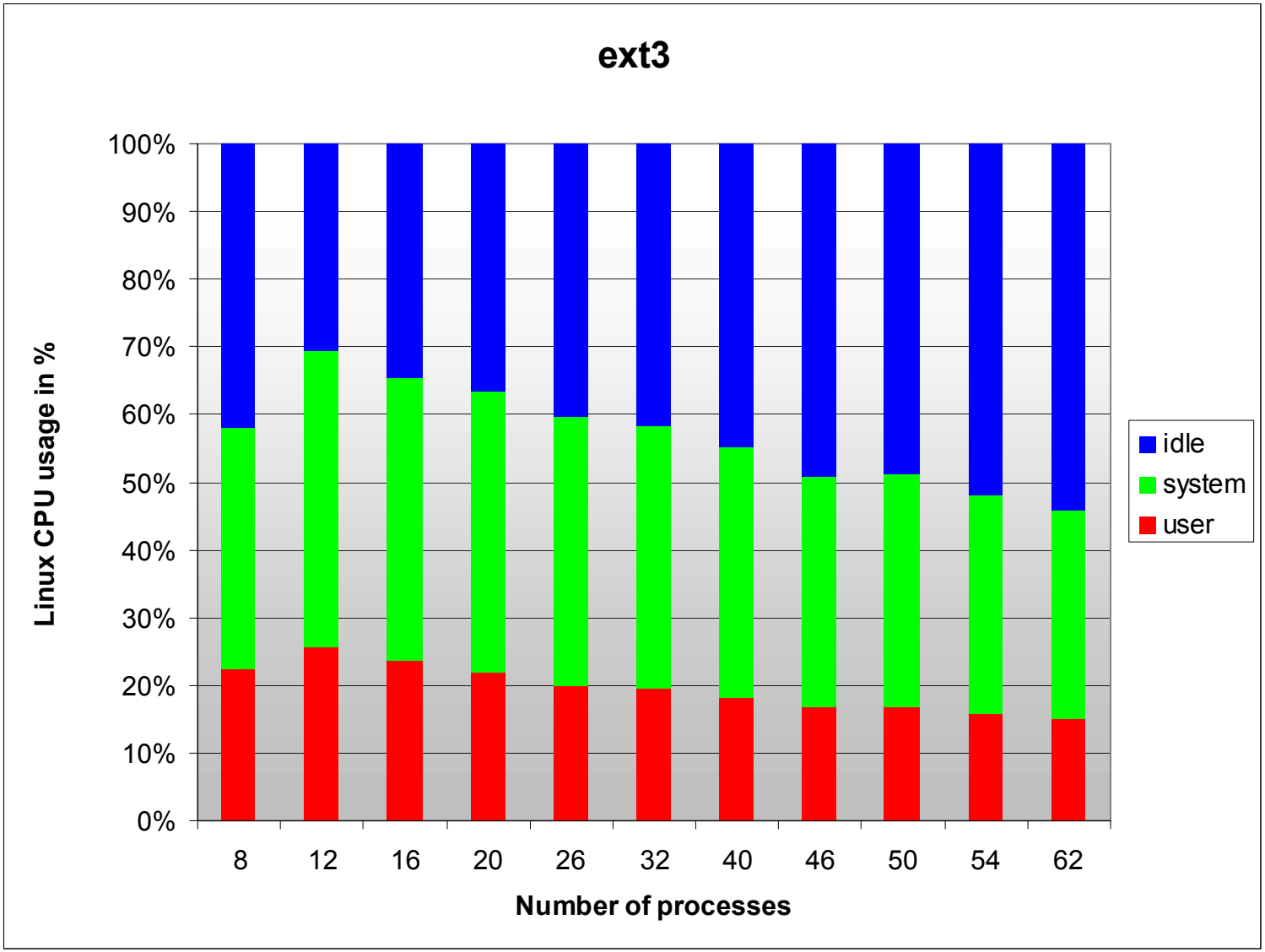


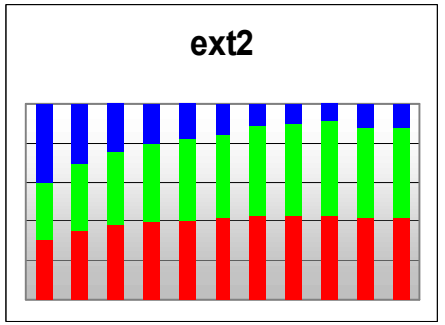


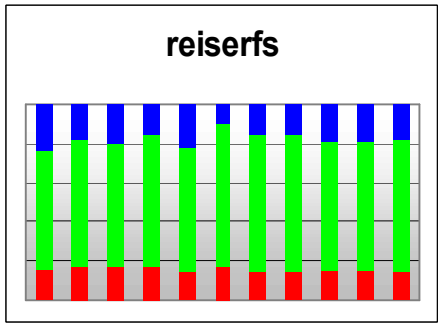


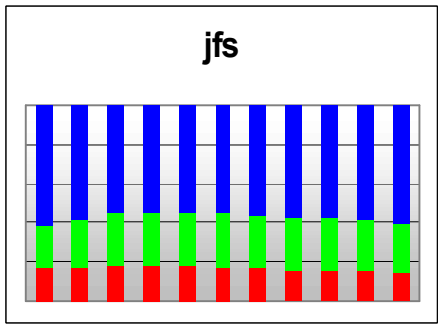


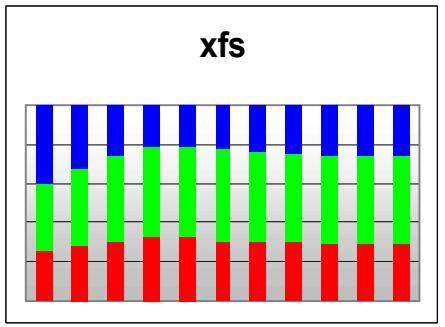
CPU Load – SCSI, LVM, 4 CPU, 2 GB



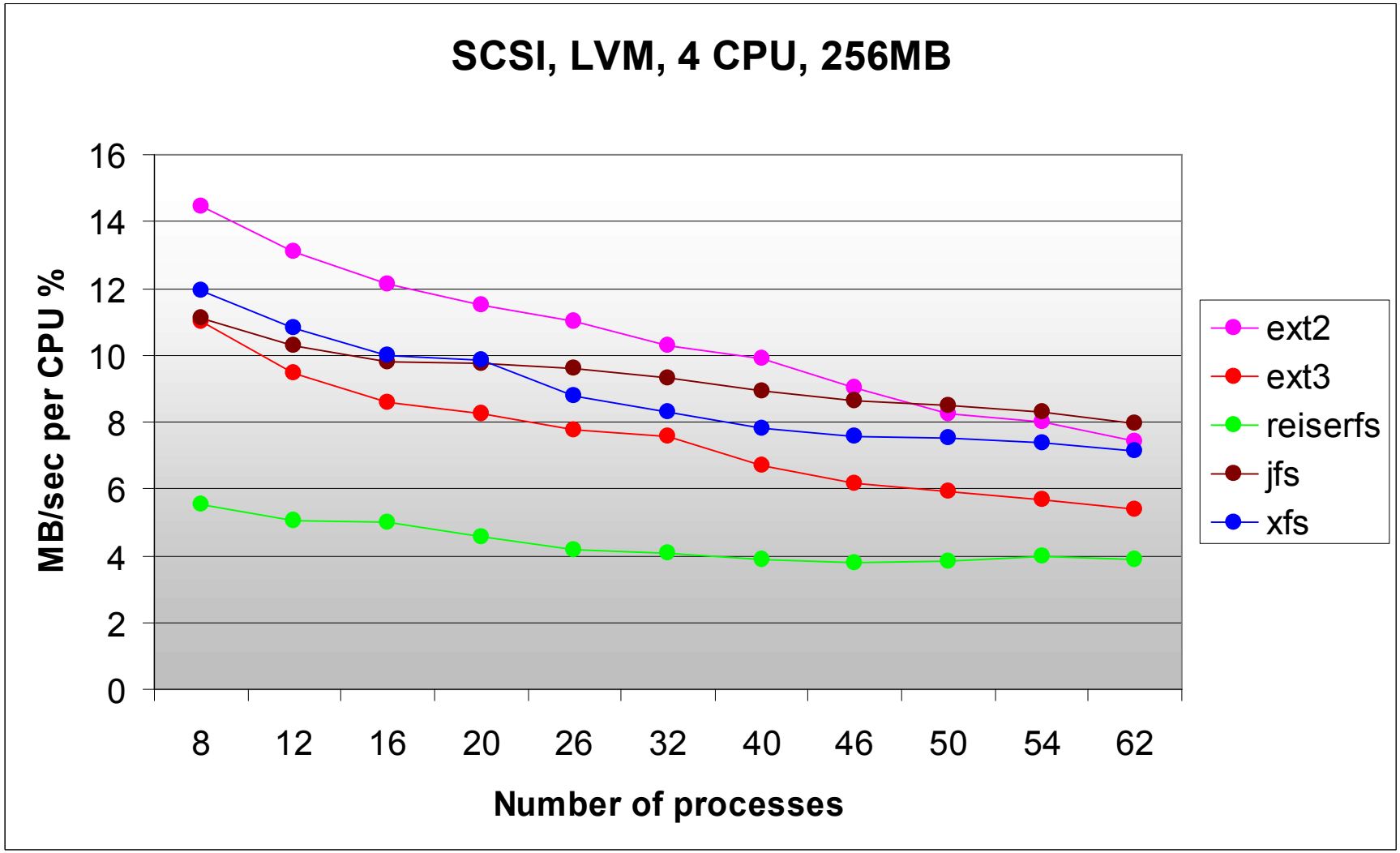








Throughput Per CPU Usage %



CPU Usage % In Relation To Throughput