Choosing and Tuning Linux File Systems

Finding the right file system for your workload

Val Henson
<val_henson@linux.intel.com>

With help from #linuxfs on irc.oftc.net

Structure of talk

- Understanding your workload
- File system performance basics
- Differences between file systems
- Example workloads and file system choices
- Q & A

What is the One True File System?

What is the One True File System?

ZFS

What is the One True File System?

~~ZFS~~

ext3

What is the One True File System?

~~ZFS~~

~~ext3~~

chunkfs

What is the One True File System?

~~ZFS~~

~~ext3~~

~~chunkfs~~

$ONE_TRUE_FS

How to cut through the hype?

The Answer

The Answer

The Answer

It depends.

Understanding your workload

- Size of file system, files, directories, reads, writes
- Pattern of file operations
- Caching - application or operating system level?
- What data consistency guarantees do you need?

File system performance basics

- No single "best file system" - workload dependent
- Disk characteristics usually dominate file system performance
- Disks go faster with large, sequential I/Os
- Fixed cost per I/O limits I/Os per second (iops)

File system performance basics

- On-disk format determines cold-cache performance
- In-memory format determines warm-cache performance

How file systems like to be treated

- Mostly reads
- Large, contiguous I/Os on block boundaries
- File size 4-128 KB
- Directories with 10-1000 entries
- I/Os near the beginning of the file
- Few metadata operations
- Clean unmount

How to abuse your file system

- Create one directory with a million files
- Create huge file till ENOSPC
- Randomly create and delete small files
- Randomly read and write single bytes
- Add and remove extended attributes and ACLs
- Now yank the power plug... slowly

(may result in non-functional machine)

Differences between file systems

- File system and file size
- Number of files and directories (inodes)
- Directory size and lookup algorithm
- File data read/write performance
- File create/delete performance
- Space efficiency
- Special features - direct I/O, execute in place, etc.

Differences between file systems

- Data consistency guarantees
- Crash recovery method
  - fsck
  - journal replay
  - copy-on-write
- Ease of repair
- Stability
- Support

Quick summary of local file systems

- ext2 - simple, fast, stable, easy to repair, but slow recovery
- ext3 - rock stable, fast recovery, but slow metadata ops
- XFS - best for large files, big directories, big file systems, but slow repair
- reiserfs (v3) - best for small files, but less stable, poor repair, less support

Common workloads and recommendations

- Embedded - small, read-mostly
- Laptop - frequent crashes, low traffic
- Desktop - middle of the road
- File server - high concurrency, bandwidth
- Mail server - many small file operations
- Database server - many small random I/Os
- Video server - large write-once read-many files

Embedded devices

- What is your "disk" - flash, ram, thumb drive?
- ext2 for memory-based file systems
- ext3 for disk
- jffs2, LogFS for flash
- Avoid writing flash - modern flash may do wear-leveling - but poorly

Laptop

- ext3
- Needs to withstand frequent crashes, some corruption, low performance demands
- Eliminate writes as much as possible

```
# mount -o {noatime,relatime}
```

- Group writes using laptop mode, read:

  /usr/src/linux/Documentation/laptop-mode.txt

Desktop

- ext3
- Large file working set? Increase number of inodes cached in memory, see:

  /usr/src/linux/Documentation/sysctl/fs.txt

File server

- ext3
- XFS for everything ext3 can't handle
- ext3: data=writeback trades speed for data integrity after a crash, data=journal reduces latency of sync NFS writes
- Change journal, block size if needed
- Consider ext2

Mail server

- mbox format (all mail in one big file) => ext3
- maildir format (each mail in one file) => XFS
- ext3 with small blocks, high inode-to-file ratio for maildir too
- Don't cut any corners on your mail server

Database server

- ocfs2 for clustered Oracle databases
- Support for direct I/O is good
- Database tuning: an arcane art

Video server

- Large files, write-once, read-many
- XFS is clear winner
- Vast number of tuning options:

http://oss.sgi.com/projects/xfs/training/index.html

```
$ man xfs_admin
```

Simple Summary

- Use ext3 unless you know you need something else
- XFS - Big, lots
- reiserfs for small files (if you refuse to use database)
- jffs2 or LogFS for flash
- ocfs2 for clustered databases

Questions & (possibly) Answers

Thanks to linux-fsdevel@vger.kernel.org, #linuxfs on irc.oftc.net, David Chinner, Jörn Engel, Theodore Y. T'so, Zach Brown, Ric Wheeler, Ard Biesheuvel, and many many others

What about reiserfs?

- reiser4 a research file system
- reiserfs (v3) not widely supported (NameSys doesn't care, SuSE likely to move to ext3)
- Difficult to repair
- Only file system that stores small files efficiently (in space, "notail" option often recommended for performance)
- Better to alter file usage pattern (use a database, fewer files per directory)

Quick NFS tuning tips

- Raise read/write size

```
# mount -o rsize=8192,wsize=8192
```

- Use NFSv3 and TCP (not UDP)
- async option raises write performance but could cause problems in the event of a server crash (Note: default recently changed from async to sync)

A note on distributed file systems

- Central tradeoff of latency versus consistency
- Most distributed file systems are buggy and slow
- Only use distributed file systems optimized for one case
  - NFS - multiple reader, single writer
  - OCFS2 - database
  - GoogleFS - append-mostly workload (not available)