A Brief History of UNIX File Systems

Val Henson
IBM, Inc.
vhenson@us.ibm.com

Summary

- Review of UNIX file system concepts
- File system formats, 1974-2004
- File system comparisons and recommendations
- Fun trivia
- Questions and answers (corrections ONLY during talk)

VFS/vnode architecture

- VFS: Virtual File System: common object-oriented interface to fs's
- vnode: virtual node: abstract file object, includes vnode ops
- All operations to fs's and files done through VFS/vnode interface
- S.R. Kleiman, "Vnodes: An Architecture for Multiple File System Types in Sun UNIX," Summer USENIX 1986

Some Definitions

superblock: fs summary, pointers to other information
inode: on-disk structure containing information about a file
indirect block: block containing pointers to other blocks
metadata: everything that is not user data, including directory entries

Disk characteristics

- Track - contiguous region, can be read at maximum speed
- Seek time - time to move the head between different tracks
- Rotational delay - time for part of track to move under head
- Fixed per I/O overhead means bigger I/Os are better

In the beginning: System V FS (S5FS) (c. 1974)

- First UNIX file system, referred to as "FS"
- Disk layout: superblock, inodes, followed by everything else
- 512-1024 byte block size, no fragments
- Super simple - and super slow! 2-5% of raw disk bandwidth

Berkeley Fast File System (FFS or UFS) (c. 1984)

- Metadata spread throughout the disk in "cylinder groups"
- Block size 4KB minimum, frag size 1KB (to avoid 45% wasted space)
- Physical disk parameters taken into account
- Much better performance - 14-47% of raw disk bandwidth
- Also 256 byte file names, file locks, symlinks, rename(), and quotas

Improvements to FFS/UFS

- 198?: Logging
- 1991: Improvements to block allocation and read ahead policy (Larry McVoy and Steve Kleiman)
- 1999: Soft updates (Kirk McKusick and Greg Ganger)

Log-Structured File System (LFS) (1991)

- All on-disk data in the form of sequential log entries
- Crash recovery rolls forward from last checkpoint
- 70% of raw disk write bandwidth, but FFS can do as well
- Mendel Rosenblum and John K. Ousterhout, "The Design and Implementation of a Log-Structured File System," 13th ACM SOSP

ext2 and ext3 (1993 - present)

- FFS-like, no fragments
- ext3 adds journaling, removes need for fsck on crash
- Primary attributes are simplicity, performance, and recoverability
- ext2 often beats all other file systems in performance tests

Write Anywhere File Layout (WAFL) (1994)

- Copy-on-write, checksummed, always consistent on-disk format
- Metadata in files
- Intent log in NVRAM satisfies NFS sync semantics and performance
- Revolutionary and completely underappreciated
- Dave Hitz, et al., "File System Design for an NFS File Server Appliance," USENIX Winter 1994

SGI's XFS (1996)

- EFS was FFS with extents - pointers to contiguous lists of blocks
- XFS adds B+ trees to track free space, index directories, locate file blocks and inodes, also first 64-bit file system in wide use
- Focus on scalability and streaming I/O - 90-95% raw disk bandwidth
- Dynamic inode allocation, logging, volume manager, multi-threaded read/writes

JFS (2000)

- Similar to XFS with fewer B+ trees

- Currently undergoing major hype

Veritas File System (VxFS) (c. 1988 - present)

- Version 1: Similar to FFS plus logging
- Version 5: Similar to XFS/JFS
- Main claim to fame: ported to many architectures
- Leader in special purpose performance hacks

Reiserfs v. 1 - 4 (c. 1998 - present)

- Designed for good small file and large directory performance
- Constantly changing (4 on-disk formats in 6 years)
- A little of everything: logging, B+ trees, COW, tail-packing
- Second only to ext2 in performance, but poorer recovery
- Understood by few, no good papers

Zettabyte File System (ZFS) (2004)

- First fs to improve on WAFL, available in a future Solaris 10 update
- Built-in volume manager, file systems share storage
- First 128-bit file system
- Self-healing data, dynamic striping, multiple block sizes, unlimited constant-time r/w snapshots...

Solutions to common design problems

- On-disk consistency in face of system crash
- Large contiguous allocations
- Metadata allocation

On-disk consistency

- Repair after the fact - fsck (FFS, ext2) item Journal replay - write log, redo writes
- Soft updates - per-pointer roll forward and back
- Always consistent - copy-on-write

Large contiguous allocations

- Block clustering
- Extents
- Multiple block sizes

Metadata allocation

- Fixed number, all together
- Fixed number, statically spread throughout the disk
- Fixed number, located anywhere
- Dynamically allocated, in files

File system feature table

	FFS	UFS	ext2	ext3	VxFS	XFS	JFS	WAFL	reiser4	ZFS
Dynamic inodes					X	X	X	X	X	X
64/128 bit						X	X	X		X
no fsck on boot	X	X		X	X	X	X	X	X	X
logging		X		X	X	X	X	X	X	X
always consistent								X		X
checksums								X		X
fs snapshots	X	X			X			X	X	X
online resize					X	X	X	X	X	X
fast & reliable			X							

Q&A

vhenson@us.ibm.com