Lecture Notes, 9/2/98

UNIX Fast File System
Measurements of a Distributed File System

I. A Fast File System for UNIX

Original UNIX file system was simple and elegant, but slow.

Could only achieve about 20 KB/sec/arm; ~2% of 1982 disk bandwidth

Problems:

- blocks too small
- consecutive blocks of files not close together
- i-nodes far from data
- i-nodes of directory not close together

Aspects of new file system:

- 4096 or 8192 byte block size (why not larger?)
- large blocks and small fragments
- disk divided into cylinder groups
- each contains superblock, i-nodes, bitmap of free blocks, usage summary info
- keeps i-node near file, i-nodes of a directory together
- cylinder groups ~ 16 cylinders, or 7.5 MB
- cylinder headers spread around so not all on one platter

Two techniques for locality:

- don't let disk fill up in any one area
- paradox: to achieve locality, must spread unrelated things far apart
- note: new file system got 175KB/sec because free list contained sequential blocks (it did generate locality), but an old system has randomly ordered blocks and only got 30 KB/sec

Specific application of these techniques:

- goal: keep directory within a cylinder group, spread out different directories
- goal: allocate runs of blocks within a cylinder group, every once in a while switch to a

new cylinder group (jump at 1MB).

- layout policy: global and local
- global policy allocates files & directories to cylinder groups. Picks "optimal" next block for block allocation.
- local allocation routines handle specific block requests. Select from a sequence of alternative if need to.

Results:

- 20-40% of disk bandwidth for large reads/writes.
- 10-20x original UNIX speeds.
- Size: 3800 lines of code vs. 2700 in old system.
- 10% of total disk space unusable (except at 50% perf. price)

Could have done more; later versions do.

Enhancements made to system interface: (really a second mini-paper)

- long file names (14 -> 255)
- advisory file locks
- symbolic links (contrast to hard links)
- atomic rename capability
- disk quotas

3 key features of paper:

- parameterize FS implementation for the hardware it's running on.
- measurement-driven design decisions
- locality "wins"

A major flaws:

- measurements derived from a single installation.
- ignored technology trends

A lesson for the future: don't ignore underlying hardware characteristics.

Contrasting research approaches: improve what you've got vs. design something new.

Lecture 2

II. Measurements of a DFS

History:

- two cs262 projects yield original BSD study in 1985
- follow-on cs262 project in 1990,
- Baker *et al*. work.

Goals: analysis paper

- Understand how access patterns have changed (it at all) since 1985.
- Understand behavior of distributed caches in network file systems.

BSD comparison results: *Which of these do you believe will hold in the future?*

- Throughput 20x higher; process migration can add another 6x

  Why has throughput gone up 20x?

  - bigger binaries
  - bigger projects; more files to link, grep, etc.
  - studied user community does "big file" things.
- Why did they get unexpected process migration results?
  - really just remote execution
  - only compute-intensive things get migrated
- Accesses still mostly sequential
- Files short (80% < 10KB), but significant numbers of large files (50% of all bytes are transferred to files > 500 KB).
- File lifetimes: 30-50% of bytes die within 30 sec.
- Note Hartmann letter correcting paper results.

Caching results:

- Caches not as effective as predicted in BSD study (40% read misses with 5-10 MB caches): why?
  - large files tend to flush caches
  - per-client caches less effective
- Paging is a significant portion of server traffic (35%)
- Client caches are big enough:
  - Block lifetimes in client caches ~ 1 hr or more
  - All write-backs for integrity, not for block replacement
    (70% 30-sec. delay, 16% fsync, 12% recall)
- Write sharing doesn't happen often, but happens often enough to affect many users.

- Simplest consistency mechanism seems good enough.

Good figure technique: individual data points explained as examples.

An interesting assertion: local disks considered “harmful”! Why? Network is faster than disk.

Some key features of the paper:

- Validated (most of) the original BSD study results for 1990 timeframe.
- Data caching works well, but not as well as originally predicted.
- Consistency for write sharing must be supported, but a simple implementation is good enough.

A flaw:

- How general is the measured user community and the workload it generates?

A lesson: Understanding the workload of your system is important.