Clustering, Logging, and Journaling:

Approaches to High Performance File System Design

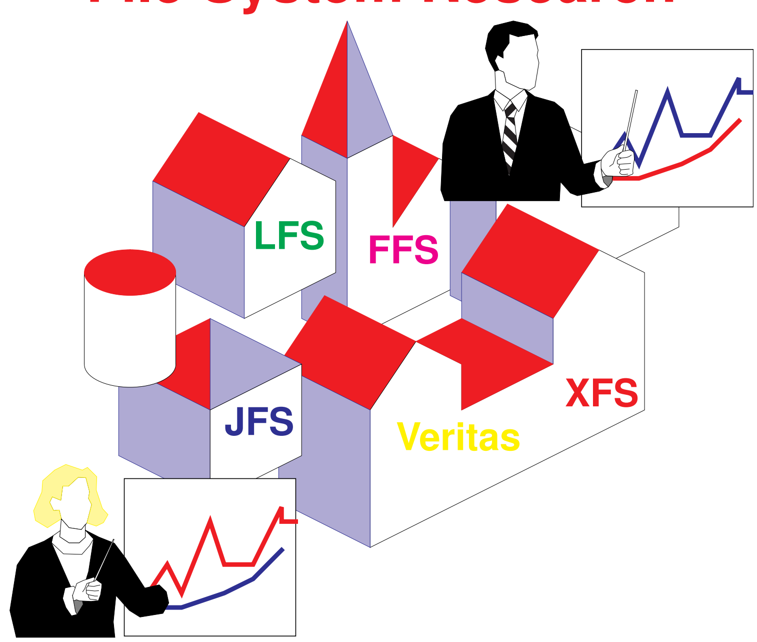
Margo I. Seltzer

Harvard University

Division of Applied Sciences

March 17, 1995

File System Research



The State of the World

- **There are a variety of file systems.**
- **Benchmarks exist to make any file system look excellent.**
- **Benchmarks exist to make any file system look bad.**
- **Commercial sector is moving to journaling file systems (IBM, Veritas, SGI).**
- **Still debating performance of FFS/LFS.**
- **Result: Total Confusion**

What I Will Talk About

- **Approaches to high performance.**
- **Performance comparisons.**
  - Emphasis on definition of stressful conditions.
  - Emphasis on performance trade-offs.
  - Emphasis on areas for improvement.
- **Conclusions**

Approaches to High Performance

- **Clustering: batch multiple I/Os into single disk request.**
- **Journaling: keep an auxiliary data structure to log synchronous operations.**
- **Log-structuring: perform all writes contiguously and asynchronously.**

Clustering

- **Historically, Unix file systems read/write one block at a time.**
- **Writing:**
  - If blocks are contiguous, delay writing.
  - Coalesce multiple blocks into single I/O.
- **Reading:**
  - Use aggressive read-ahead.
  - Read-ahead in *clusters*.
- **Provides file system throughput near disk maximum, for large files.**

Peacock—1988, McVoy—1991

Journaling

- Even with clustering, meta-data operations are still synchronous.
- Keep auxiliary data structure (log).
- Log meta-data operations.
- Make other writes asynchronous.
- Provides fast log-based recovery.

Hagmann—1987, Chang—1990, Chutani—1992

Log-Structured

- Make all writes asynchronous.
- Use a log-based representation on disk.
- Write data in large segments.
- Use garbage collector to reclaim space.
- Potential for read/write performance akin to clustering for large files.
- Solves both synchronousity problem and small file performance problem.
- Also provides fast, log-based recovery.

Ousterhout—1989, Rosenblum—1992

System Comparisons

	FFS	LFS	Journal
I/O	Clusters	Clusters	Clusters
Meta-Data	Sync; many independent writes	Asynch; bundled	Sync log write; async independent writes
Recovery	Slow	Fast	Fast
Pitfalls	Fragmenta-tion; Synchronous meta-data updates	Cleaning	Separate log device Individual I/Os on small files.
Code	**100%**	**60%(+40%)**	**100%(+23%)**

Performance comparisons

Test	What to Look For	Description
Ideal	Maximum throughput. Impact of synchronousity. Impact of write-bundling. Effects of fragmentation.	Create, read, write, delete multi-ple megabytes of data decomposed into files.
TPC	Impact of LFS cleaning in a hostile environment	Check processing.
NFS	Cleaning cost in workstation environment. Impact of NFS synchronous writes on LFS.	Simulate LFS on real NFS traffic.

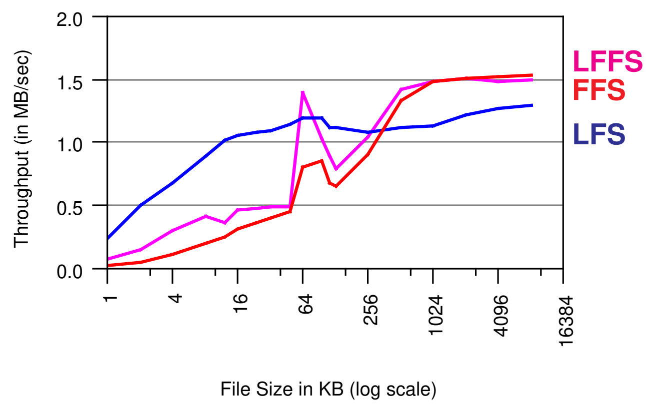
Benchmark Platform

- **20 Mhz Sparcstation I.**
- **Fujitsu M2694EXA 5400 RPM disk.**
  - 15 platters, 1818 cylinders.
  - 34-55 KB/track.
  - 9.5 ms average seek.
  - 512KB track buffer.
- **4.4BSD-Lite.**
  - LFS fixes: ordered blocks and vnodes, fragments.
  - Journaling: support asynchronous FFS mounts and journaling file system (WAFS).
  - 8 KB block size.
  - FFS has rotational delay = 0.

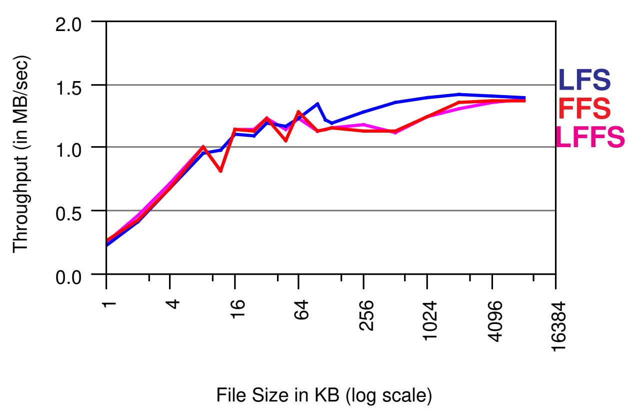
Ideal Operating Conditions

- **Begin with disk empty .**
- **No cleaner running.**
- **The Benchmark**
  - Based on the Sprite-LFS benchmarks.
  - 16 MB of data (or 10 files, whichever is bigger).
  - Vary file size (1K - 8 MB).
  - 100 files / directory.
  - 4 tests: Create, Delete, Read, Write.
  - All file accesses in create order.

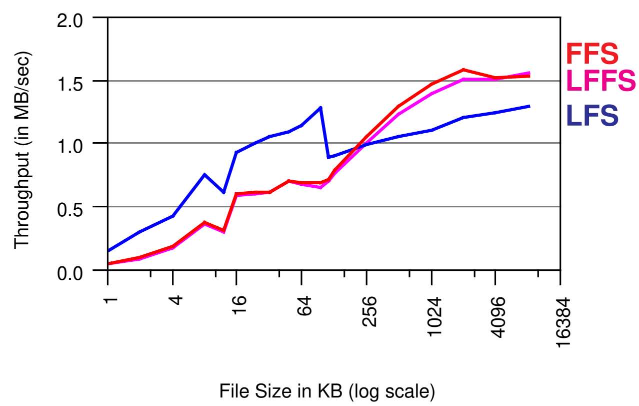
Create Performance



Read Performance



Write Performance

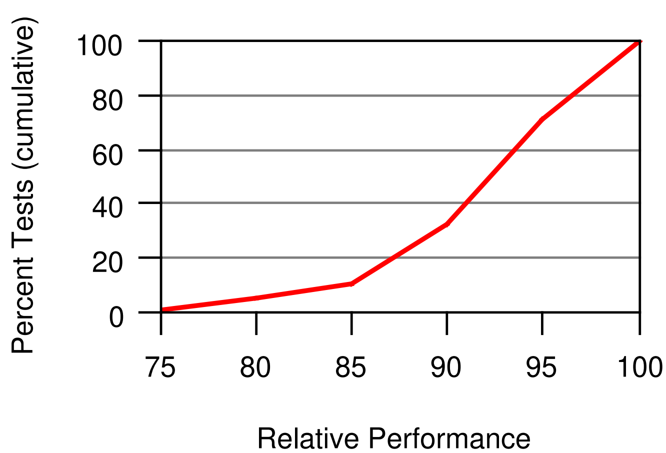


Delete Performance

Fragmentation—!Ideal

- **Apply benchmark to fragmented FFS.**
- **Data Collection**
  - Nightly snapshots for 11 months on 51 file systems.
  - Snapshot includes all allocation information, file system size, file ages.
- **Benchmark**
  - Goal: evaluate performance of a file system over time.
  - Recreate file system as it existed on a previous date.
  - Reduce files/directory to 25 .
  - Simulate SunOS 4.1.3 File Systems.
  - Numbers for: 9 file systems, 4 dates, 3 file sizes.
  - Express results as fraction of empty performance.

Fragmentation Results

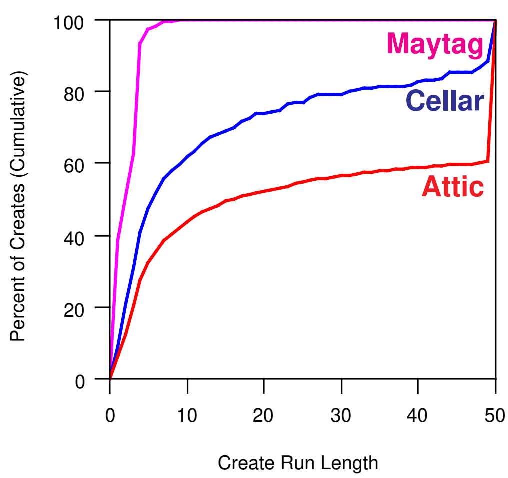


- **Less degradation than expected.**

Performance Correlation

- **No consistent correlation between disk utilization and performance.**
  - Some file systems show positive correlation.
  - Some file systems show negative correlation.
- **No correlation between age and performance.**
- **Moderate correlation between file layout and performance.**

Reasonable Benchmark?



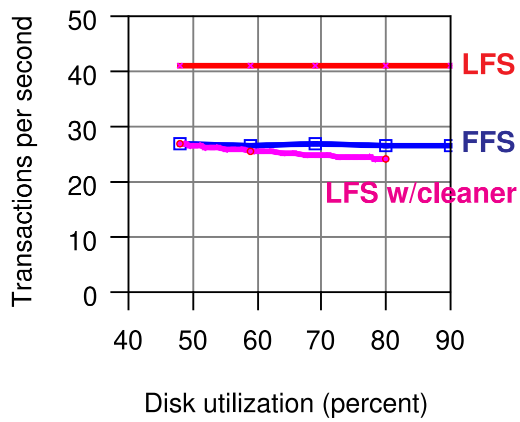
TPC—Check Cashing

- **Cleaning in a hostile environment.**
- **Simulates a Debit/Credit transaction.**
- **Four files.**
  - account: 1,000,000 records.
  - branch: 10 records.
  - teller: 100 records.
  - history: append-only.
- **Transaction.**
  - Select account at random.
  - Update account, teller, branch.
  - Write history record.

The Test Configuration

- **40 Mhz SparcStation II.**
- **DSP3105 5400 RPM disk.**
  - 14 platters, 2568 cylinders.
  - 57 512-byte sectors/track.
  - 9.5 ms average seek.
  - 256 KB track buffer.
  - 2.3 MB/sec bus bandwidth.
  - 2.5 MB/sec disk bandwidth.
- **4.4BSD-Lite.**

TPC/B Performance



- **Performance sensitive to cleaning algorithm.**

Cleaning—Why?

- **Very different results than SOSP91.**
- **SOSP showed random workload with greedy cleaner; we use cost-benefit.**
- **Age calculation is crucial.**
- **SOSP91 simulated calculation**
  - log(number of writes performed)
- **Sprite-LFS implementation**
  - age of segment in minutes
- **BSD-LFS implementation**
  - age of segment in seconds
- **Need a cleaner that dynamically adapts.**

NFS Traces

- **Cleaning in a friendly environment.**
- **3 busy NFS file systems.**
  - 2 at Harvard (61 clients on 2 Ethernets)
  - 1 at Network Appliance (33 clients on 1 FDDI ring)
  - Home directories, s/w projects, ...
- **Trace by snooping NFS requests off Ethernet/FDDI.**
- **Feed traces to simulator.**
- **NFS Server (both sync and async).**
- **LFS with 1 MB segments (1-8 GB).**

Good News!

- **Simple heuristics effective.**
- **Can clean in idle disk cycles.**
- **Can predict idle disk cycles well.**
  - Wait one second before beginning cleaning.
  - 95% of the time, cleaning completed before a user request needed the disk.
  - 99% of the time, an idle gap occured before running out of disk space.
  - Most of the time, less than 4 MB of data written between cleaner invocations.
  - Necessary reserve: much less than 10%.
- **BSD-LFS almost does this (looks at load average instead of disk idle).**

Conclusions

- **Under ideal circumstances**
  - Synchronous meta-data kills FFS.
  - Separate I/Os penalizes LFFS.
  - LFS realizes most of potential (still drops disk revolutions due to hardware constraints).
- **Fragmentation**
  - Surprisingly little degradation.
- **Cleaning**
  - Very workload dependent.
  - Algorithm must adjust to workload.
  - Need to understand correlation and aging.

Ongoing Work

- **Tune LFFS performance.**
- **Use LFFS framework to apply Ganger's ordered updates . [Ganger, 1994 (OSDI)]**
- **Cleaning Algorithms .**
- **File System Benchmarks .**
- **Applications of WAFS .**

Acknowledgements

- **FFS v.s. LFS Comparisons**
  - Hari Balakrishnan, Venkata Padmanabhan, Jacqueline Chang, Sara McMains (U. C. Berkeley)
  - Keith Smith (Harvard)
- **NFS Tracing and Simulation**
  - Trevor Blackwell, Jeff Harris (Harvard)
- **Journaled FFS**
  - Clam Associates
  - Kirk McKusick (U. C. Berkeley)

To Find Out More

- **Excuse the long URL, but:**

http://das-www.harvard.edu/users/faculty/Margo_Seltzer/papers.html

  - Papers
  - File System Code
  - Benchmarks
- **Send email to:**
  - margo@das.harvard.edu