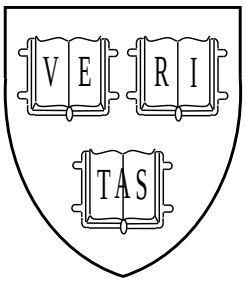
Transaction Support in a Log-Structured File System



Margo I. Seltzer

Harvard University

Division of Applied Sciences

Data Engineering 1993

Outline

- Introduction
- Implementation
- Performance
- Conclusions

Introduction

- **Technology $\Rightarrow$ I/O Bottleneck**
- **Caching $\Rightarrow$ Write Performance Critical**
- **Write Performance $\Rightarrow$ Write-optimizing file system.**

Log-Structured File Systems

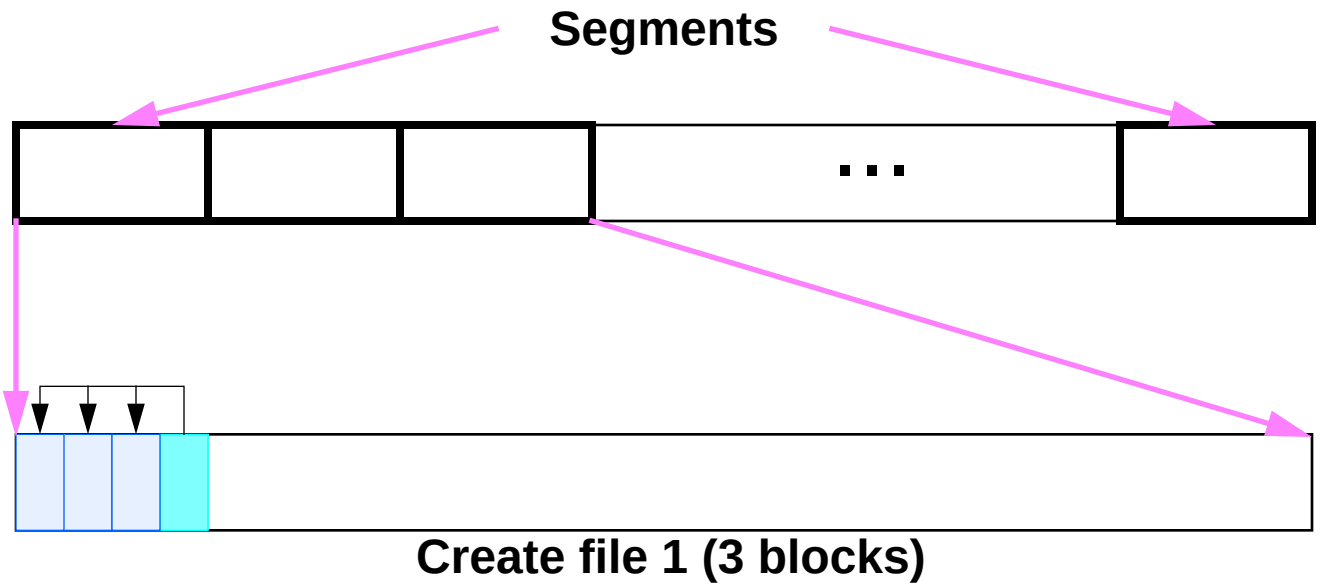
Mendel Rosenblum, John Ousterhout
"The Design and Implementation of a Log-Structured File System"
Transactions on Computer Systems, February 1992

- **All writes are sequential.**
- **Append-only writes.**

  ⇒No-overwrite policy.

- **Uses database logging techniques for recovery.**

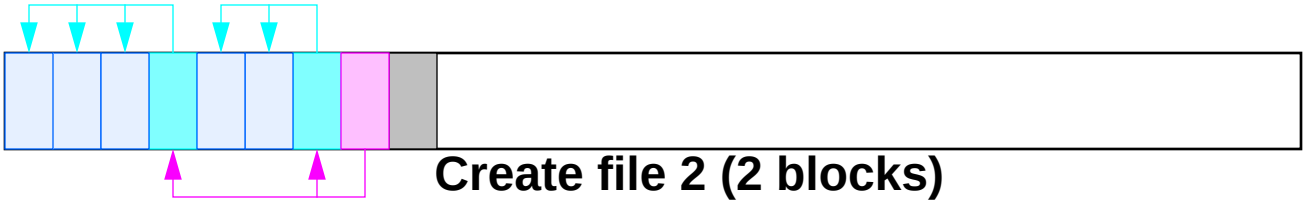
LFS Disk Layout (1)



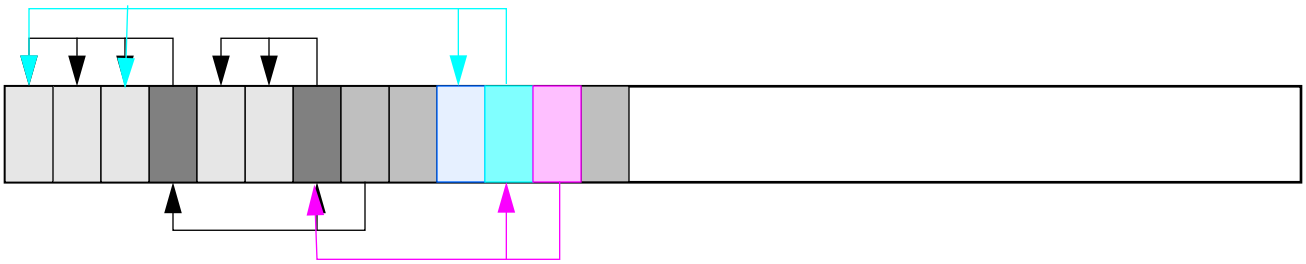
Data Block

Meta-data Block

LFS Disk Layout (2)



Create file 2 (2 blocks)



Update block 2, file 1

Data Block

Meta-data Block

File Map

Summary Block

Why LFS for Transactions

Traditionally	Using LFS
Update files in place	Update by sequential write
Use a separate log file	Use LFS's logging
Force log for commit	Use segments to impose atomicity

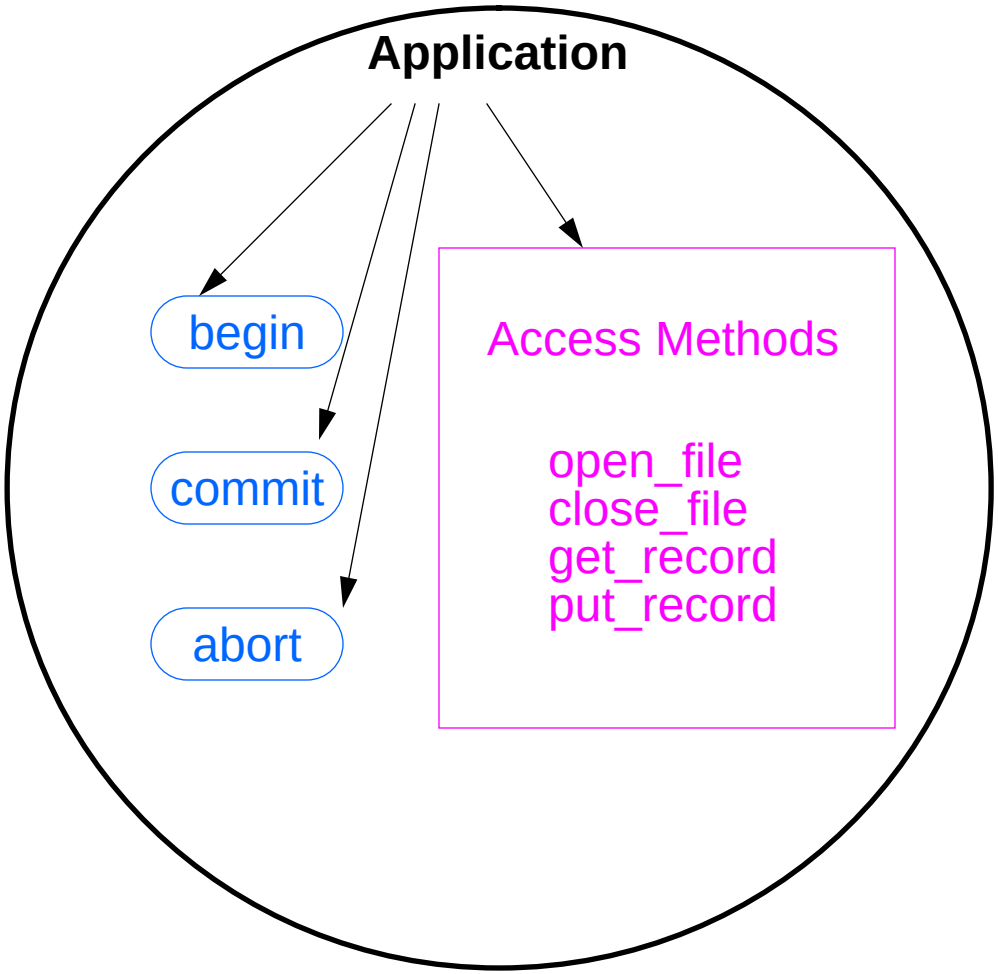
It's basically free!

Implementation Goals

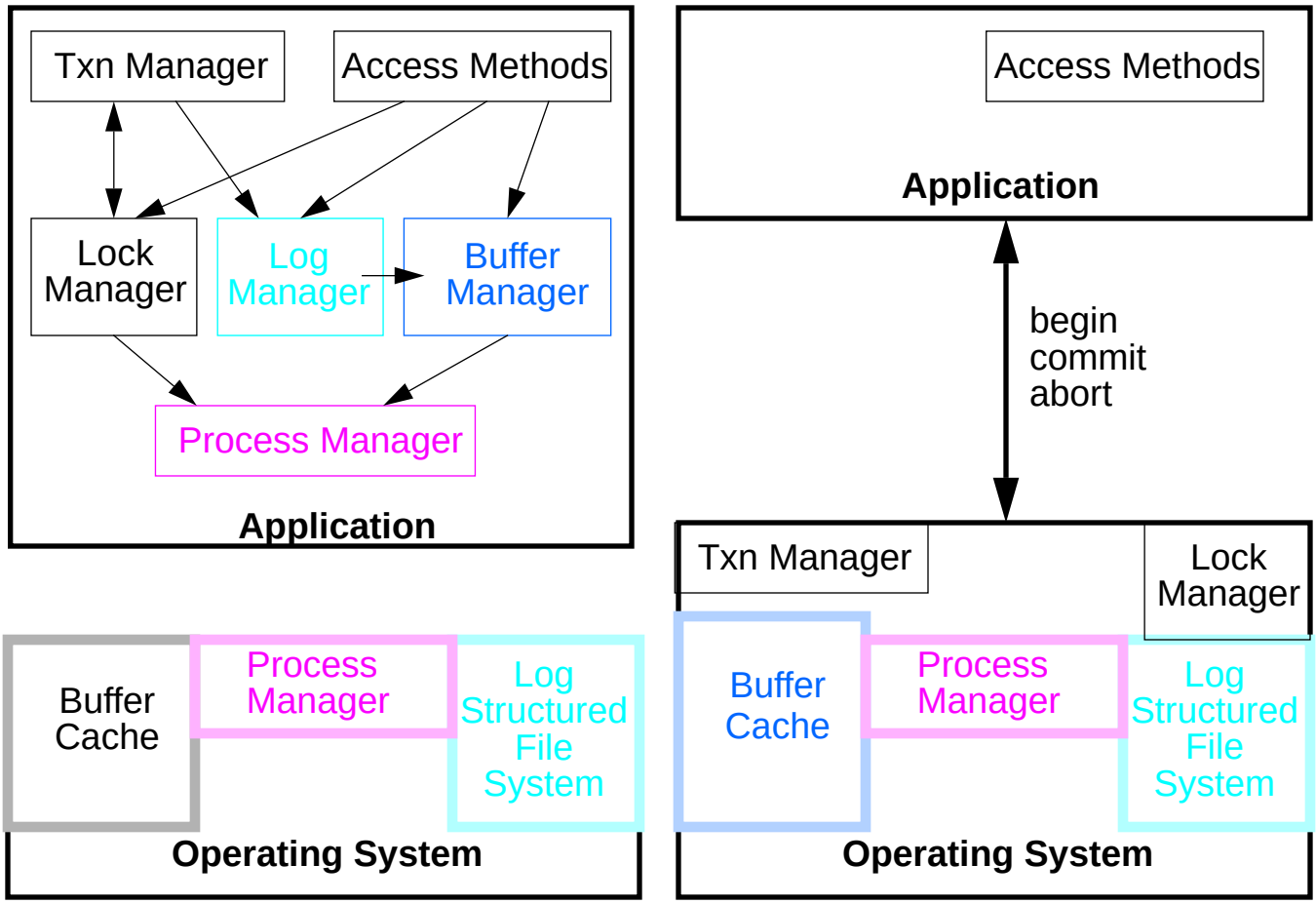
- **Compare transaction performance on LFS to transaction performance on a traditional file system.**
- **Compare user-level transaction performance to LFS-embedded performance.**

Can we provide transactions as a file system primitive with little or no overhead?

Application Structure

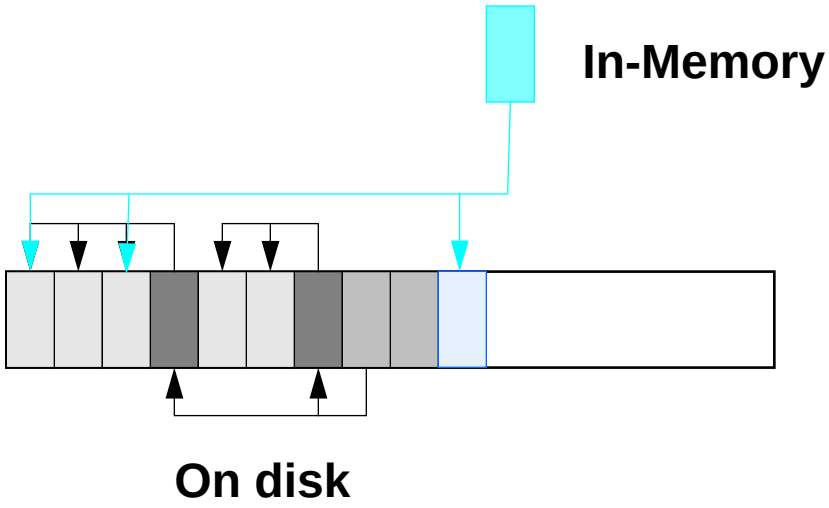


Architecture



Implementation Techniques

- **Commit: Force policy**
- **Abort: No-Steal**
- **Abort: Shadow Files**

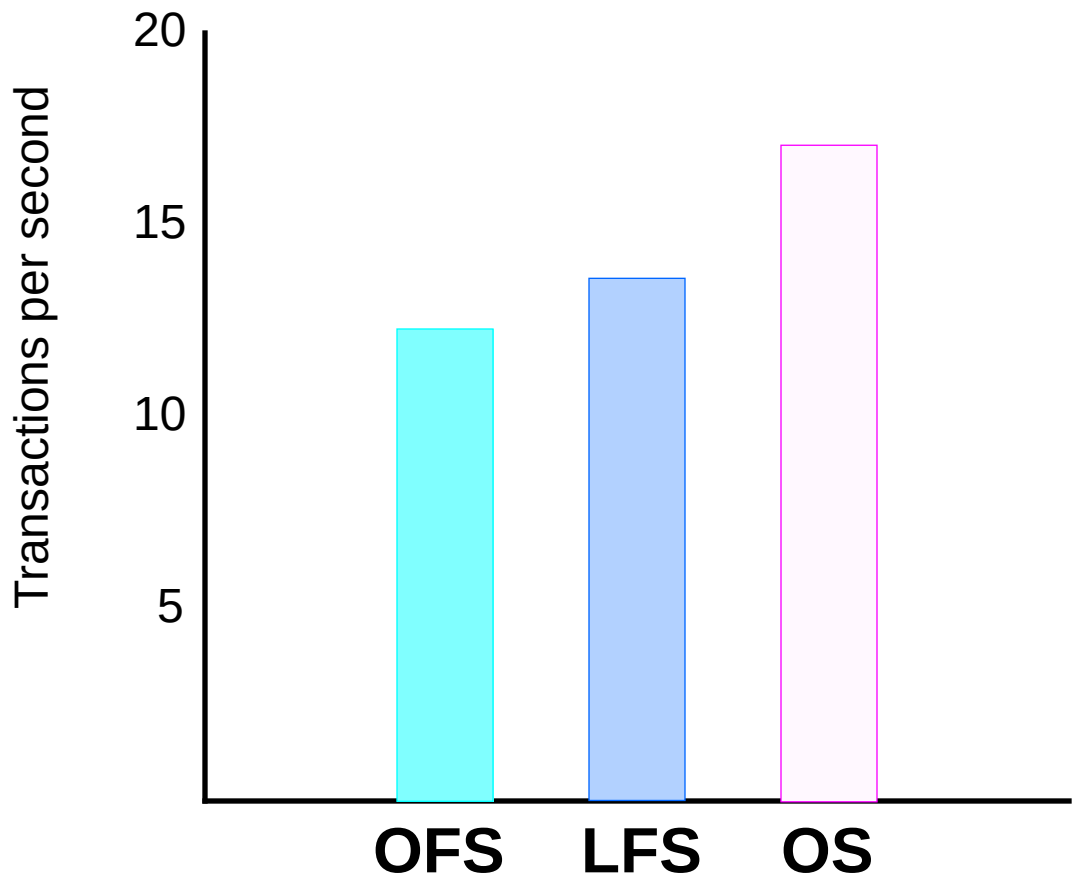


Performance

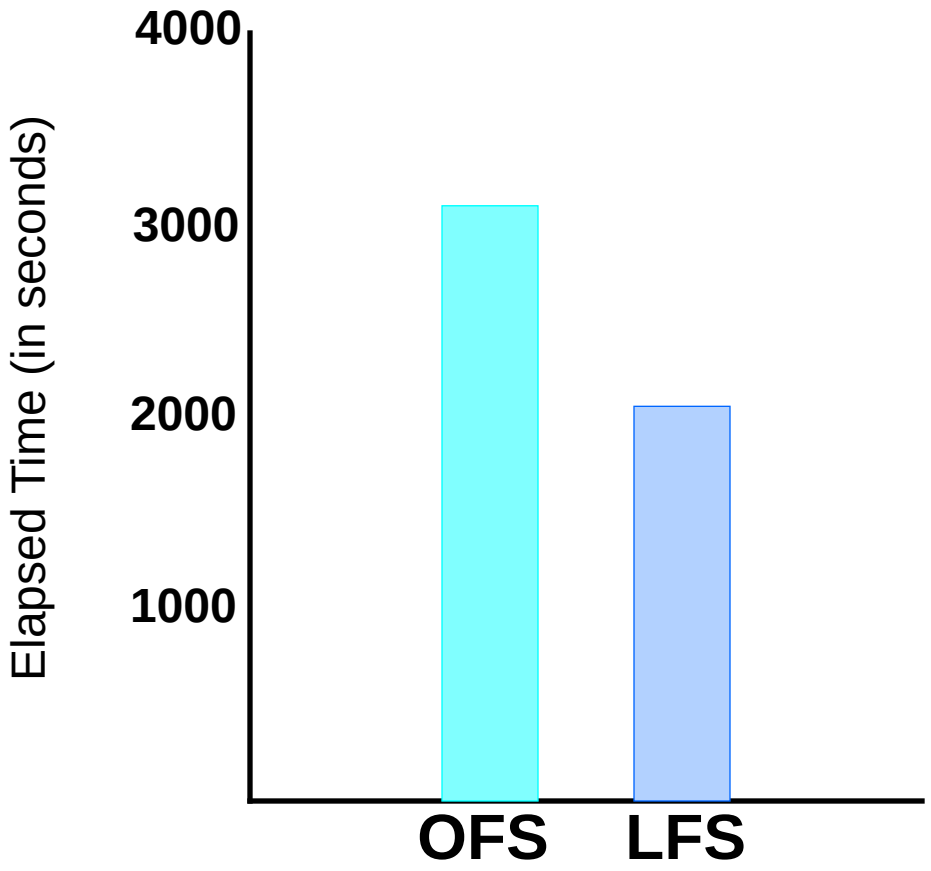
- **DECstation 5000 (15 Mips), 300 MB SCSI drive, 32 MB memory, Sprite Operating System.**
- **Modified TPCB Benchmark**

  10 TPS Scaling
  Single-user (worst case)
  No replicated log
  No think time between transactions
  Measure throughput only

Single-User TP Throughput



Sequential Performance



Conclusions

- LFS attractive for transactions.
- Embedded support is feasible.
- Sequential performance is not terrific.
- Need to experiment with alternative cleaning strategies to improve sequential write performance.