15-712:
Advanced Operating Systems & Distributed Systems

A Fast File System for UNIX

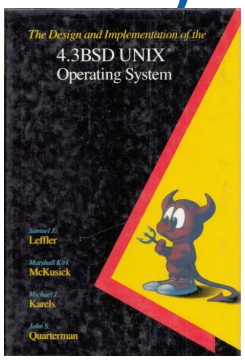
Prof. Phillip Gibbons

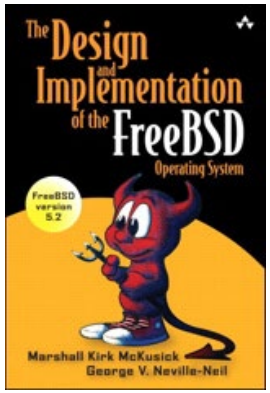
Spring 2023, Lecture 7

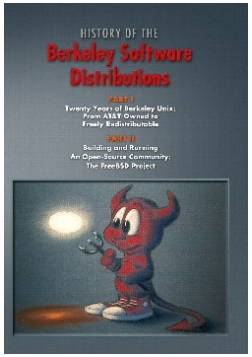
“A Fast File System for UNIX”

Marshall K. McKusick, William N. Joy, Samuel J. Leffler, Robert S. Fabry 1984

- Kirk McKusick (UC Berkeley, consultant firm, USENIX President)
- Bill Joy (UC Berkeley, Sun Microsystems co-founder/chief scientist)
  - Grace Hopper Award, NAE, AAAS
- Sam Leffler (UC Berkeley, LucasFilm, Pixar, consultant)
- Bob Fabry (UC Berkeley, started it all)







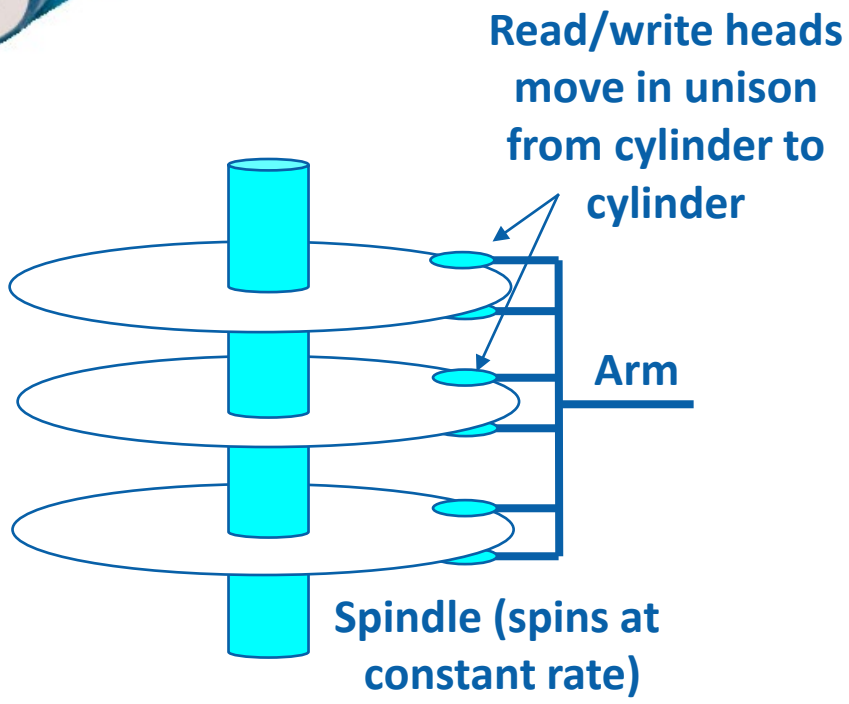
“A Fast File System for UNIX”

Marshall K. McKusick, William N. Joy, Samuel J. Leffler, Robert S. Fabry 1984

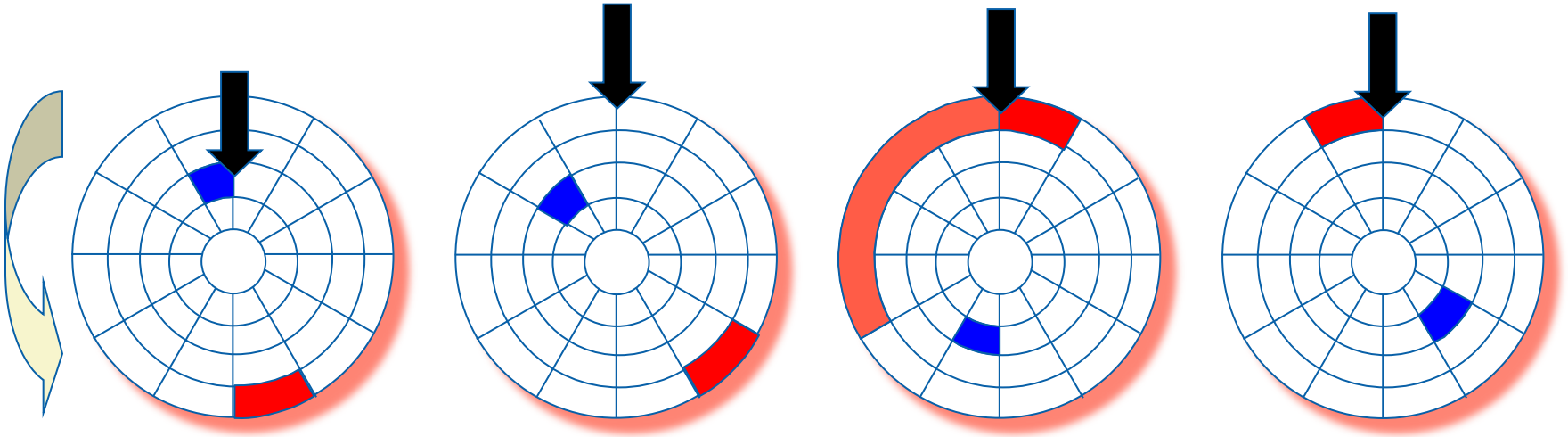
SigOps HoF citation (2015):

This paper introduced techniques to make the file system “disk aware”, thus demonstrating the importance of understanding the interplay between hardware technology and file-system design. The structuring concept of the cylinder group, while simple, is found in some form in many current systems (including the widely-deployed Linux ext* family) and serves as an excellent example of the importance of locality in storage. The paper also introduced numerous functionality improvements, including symbolic links and atomic rename, which have since become commonplace features in modern file systems.

Refresher: Hard Disk Drive (HDD)



Reading from a HDD: Time Components



Which time component(s) dominated in 1984? Today?

Problem with "Traditional" File System

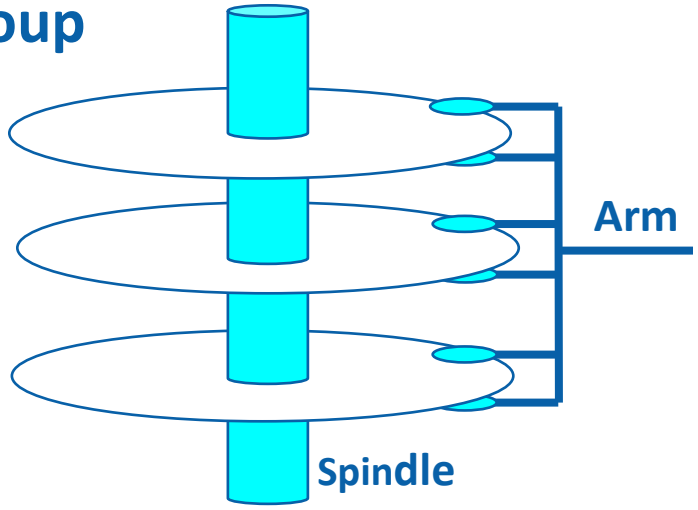
- **Poor data throughput: 2% of max disk bandwidth. Why?**
  - Long seek from file's inode to its data
  - Inodes for the files in a directory are scattered
  - Data blocks (unit of transfer) is too small (512 bytes)
  - Consecutive data blocks in a file are scattered across cylinders

- **"Old" File System (after first round of BSD improvements)**
  - Doubled block size to 1024 bytes
  - Improved reliability (added fsck)

Throughput still only 4% of max disk bandwidth

Improvements in New File System

- **Increased block size to 4096 bytes. Why 4096?**
  - Files up to 2^{32} bytes only need 2 levels of indirection
- **Divides each disk partition into "cylinder groups" (consecutive cylinders on a disk). Why?**
  - Low seek times between nearby cylinders
- **Bit map of available blocks in cylinder group (replaces free list)**
- **Cylinder group bookkeeping info at varying offsets. Why?**
  - Reliability: Not all on first platter



Wasted Space as Function of Block Size

Space used (Mbytes)	% Waste	Organization
775.2	0.0	Data only, no separation between files
807.8	4.2	Data only, each file starts on 512-byte boundary
828.7	6.9	Data + inodes, 512-byte block UNIX file system
866.5	11.8	Data + inodes, 1024-byte block UNIX file system
948.5	22.4	Data + inodes, 2048-byte block UNIX file system
1128.3	45.6	Data + inodes, 4096-byte block UNIX file system

Solution? Divide block into fragments

Bits in map	XXXX	XXOO	OOXX	OOOO
Fragment numbers	0–3	4–7	8–11	12–15
Block numbers	0	1	2	3

Fig. 1 Example layout of blocks and fragments in a 4096/1024 file system.

Utilization: 4096/1024 in New ≈ 1024 in Old. Throughput: 4x higher

Discussion: Summary Question #1

- **State the 3 most important things the paper says.** These could be some combination of their motivations, observations, interesting parts of the design, or clever parts of their implementation.

File System Parameterization

- **Processor characteristics**
  - Does I/O Channel require processor intervention
  - Expected time to service interrupt & schedule new disk transfer
- **Characteristics of each disk**
  - Number of blocks/track
  - Rotational rate (RPMs)
- **Any HW support for mass storage transfers**

Goal: Seek- and Rotationally- optimal block placement

Layout Policies

- **Global Policies**
  - Placement of new directories, new files, large files
  - Goals: (i) Localize concurrently accessed data
    (ii) Spread out unrelated data
  - For each directory, place all its file inodes in same cylinder group
    For new directory, place in lightly-loaded cylinder group
  - Redirect block allocation to a different cylinder group when file > 48 KB, and every MB thereafter

"Cost of one long seek per megabyte is small"

Maintain ≥10% free space

- **Local Policy priority**
  - 1. requested block, 2. rotationally-closest, 3. cylinder in same group, 4. quadratic hash to new group, 5. exhaustive search

Performance Improvements

(Systems in production for 1 month+)

Table IIa. Reading Rates of the Old and New UNIX File Systems

Type of file system	Processor and bus measured	Speed (Kbytes/s)	Read bandwidth %	% CPU
Old 1024	750/UNIBUS	29	29/983 3	11
New 4096/1024	750/UNIBUS	221	221/983 22	43
New 8192/1024	750/UNIBUS	233	233/983 24	29
New 4096/1024	750/MASSBUS	466	466/983 47	73
New 8192/1024	750/MASSBUS	466	466/983 47	54

Table IIb. Writing Rates of the Old and New UNIX File Systems

Type of file system	Processor and bus measured	Speed (Kbytes/s)	Write bandwidth %	% CPU
Old 1024	750/UNIBUS	48	48/983 5	29
New 4096/1024	750/UNIBUS	142	142/983 14	43
New 8192/1024	750/UNIBUS	215	215/983 22	46
New 4096/1024	750/MASSBUS	323	323/983 33	94
New 8192/1024	750/MASSBUS	466	466/983 47	95

Discussion: Summary Question #2

- **Describe the paper's single most glaring deficiency.** Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.

Discussion of Performance Results

- **10%+ free is 2X BW over when file system is full**
  - Is this a problem today?
- **40% of time spent copying from OS disk buffers to User buffers**
  - Is this a problem today?
- **Inability to coalesce multiple requests into one long request limits throughput to 50% of disk BW**
  - But would require rewriting all the disk drivers
  - Is this a problem on HDDs today?

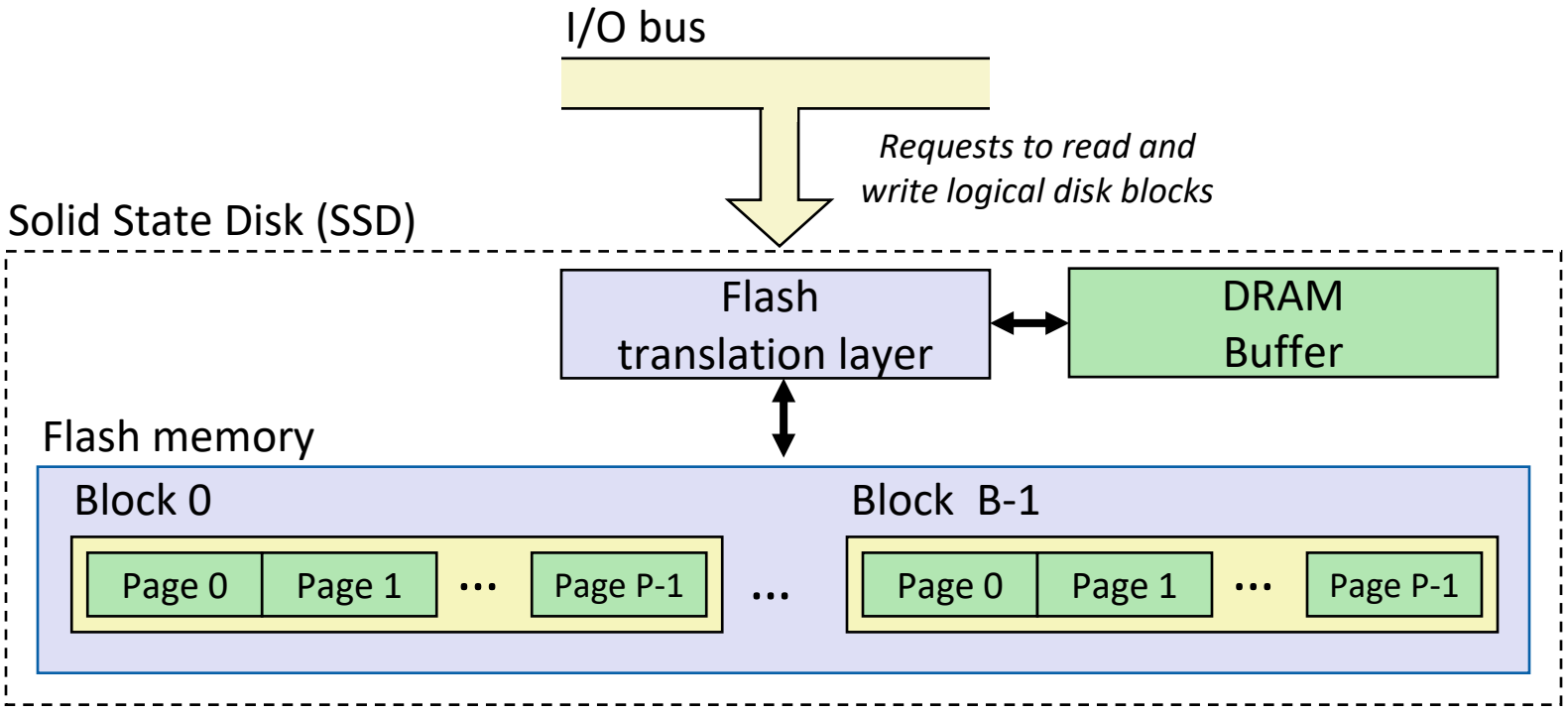
Functional Enhancements

- Long File Names
- File Locking
- Symbolic Links
- Rename
- Quotas

File Locking

- Advisory only
- Shared or Exclusive
- Only on open files
- Blocks by default, but can choose to return error instead
- Only deadlock detection is: can’t apply lock of same type twice

Solid State Disks (SSDs)



- **Pages: 512KB to 4KB, Blocks: 32 to 128 pages**
- **Data read/written in units of pages.**
- **Page can be written only after its block has been erased.**
- **A block wears out after about 100,000 repeated writes.**

Discussion: Summary Question #3

- **Describe what conclusion you draw from the paper as to how to build systems in the future.** Most of the assigned papers are significant to the systems community and have had some lasting impact on the area.

Consider both systems with HDDs & systems with only SSDs

Monday's Paper

File Systems and Disks (II)

"Scale and Performance in a Distributed File System"

John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, Michael J. West 1988

Optional Further Reading:

"Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency"

Cary G. Gray, David R. Cheriton 1989

BACKUP SLIDES

SSD Performance Characteristics

- **Benchmark of Samsung 940 EVO Plus**

https://ssd.userbenchmark.com/SpeedTest/711305/Samsung-SSD-970-EVO-Plus-250GB

Sequential read throughput	2,126 MB/s	Sequential write tput	1,880 MB/s
Random read throughput	140 MB/s	Random write tput	59 MB/s

- **Sequential access faster than random access**

- **Random writes are somewhat slower**
  - Erasing a block takes a long time (~1 ms).
  - Modifying a block page requires all other pages to be copied to new block.
  - Flash translation layer allows accumulating series of small writes before doing block write.