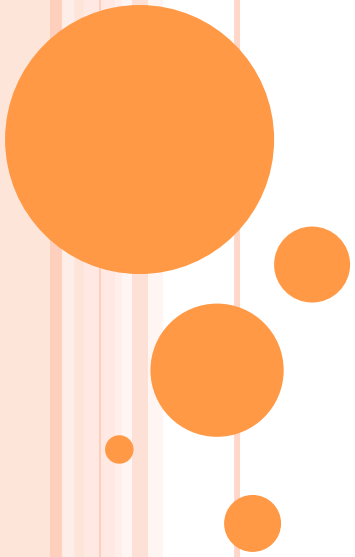


Unusual Disk Optimization Techniques

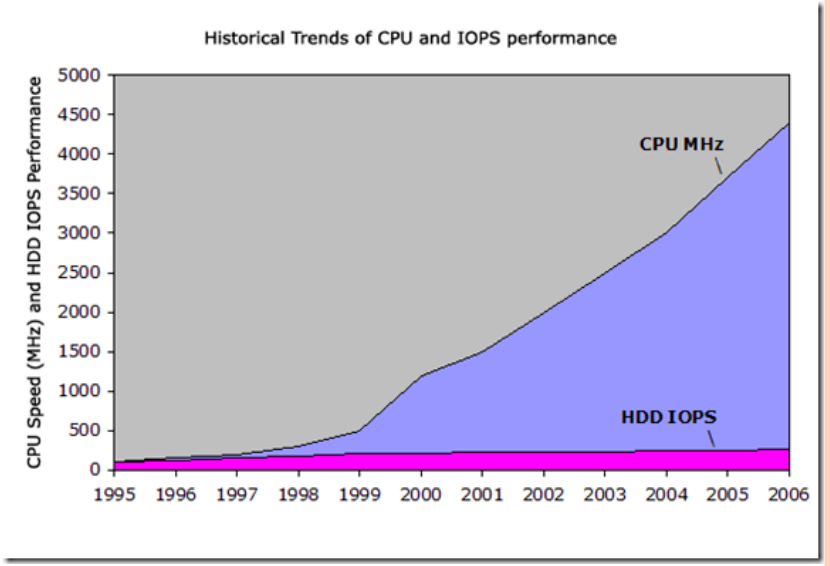
Andrew Kane

University of Waterloo - PhD Candidate – arkane@cs.uwaterloo.ca

October 28th 2009

1. Motivation

- Disk I/O is a scarce resource and often a bottleneck



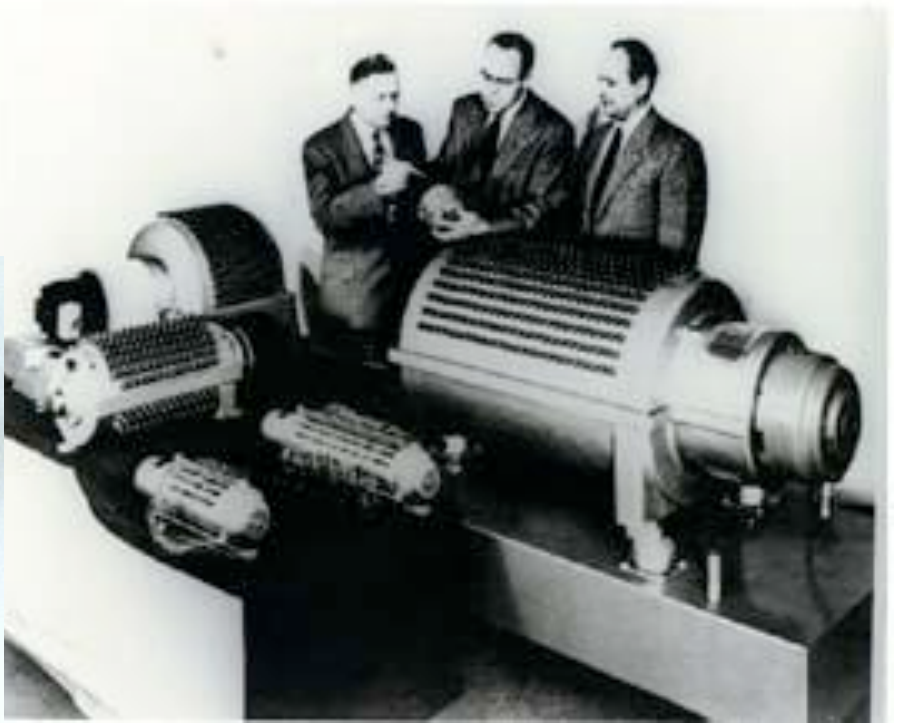
- Optimization Types:
  - Disk Efficiency (Usage Rate)
  - Low Latency Writes (Logging) or Reads (Cache)
  - Workload Smoothing (prefetching, speculative execution)

http://blogs.msdn.com/e7/archive/2009/01/25/disk-defragmentation-background-and-engineering-the-windows-7-improvements.aspx

Outline of Talk

- 1. Motivation
- 2. History
- 3. Modern I/O Stack
  - File Systems: Traditional, Journaling, Log-structured
- 4. Common Optimization Techniques
- 5. Unusual Optimization Techniques
  - 5.2 Freeblock scheduling
  - 5.3 Eager writing
  - 5.4 Low Latency Write-Ahead Log
  - 5.5 Virtual logs
  - 5.6 Dual-actuator disks
  - 5.7 Track-based logging
- 6. Conclusions

2. History
2.1 Magnetic Drum Memory



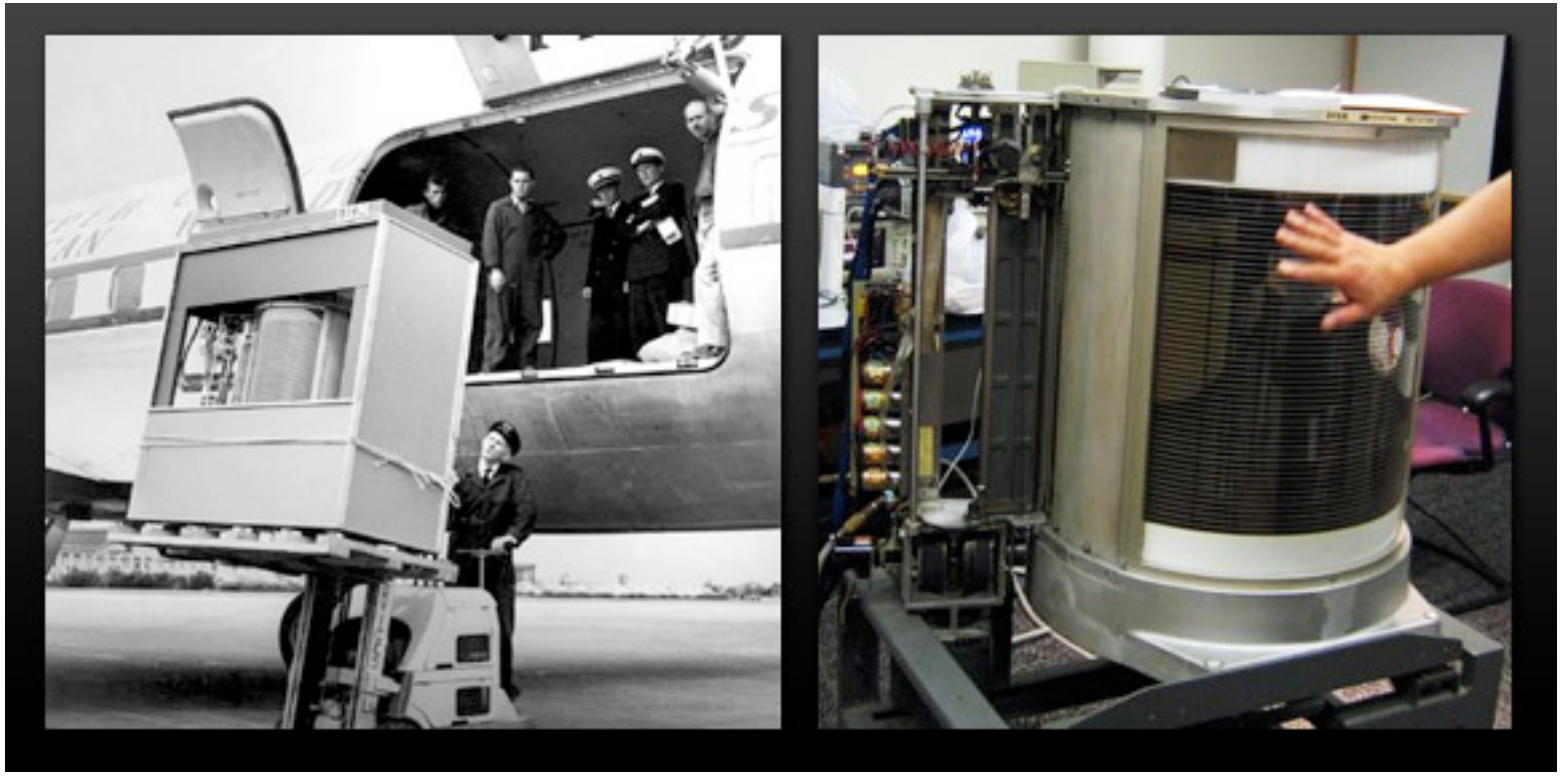
Widely used in the 1950s & 60s as the main working memory.

Above left: A 16-inch-long drum from the IBM 650 computer, with 40 tracks, 1 head per track, 10 kB of storage space, and 12,500 RPM.

2. History
2.1 Magnetic Drum Memory

- Acting as main memory means CPU is waiting for reads => we need low latency
  - Stride operations on the drum so that the next operation is under the read head when the CPU needs it
  - Fixed heads so no seek time
- This is memory, but random access is not a fixed cost

2. History
2.2 Hard Disk Drives



The first hard disk drive was the IBM Model 350 Disk File in 1956. It had 50 24-inch discs with a total storage capacity of 5 MB.

2. History
2.2 Hard Disk Drives

- Movable heads
  - Seek and rotational latency
  - So, don't use this for main memory
  - Read by block and cache results in memory so the disk is not part of the CPU execution cycle
  - Much larger storage sizes

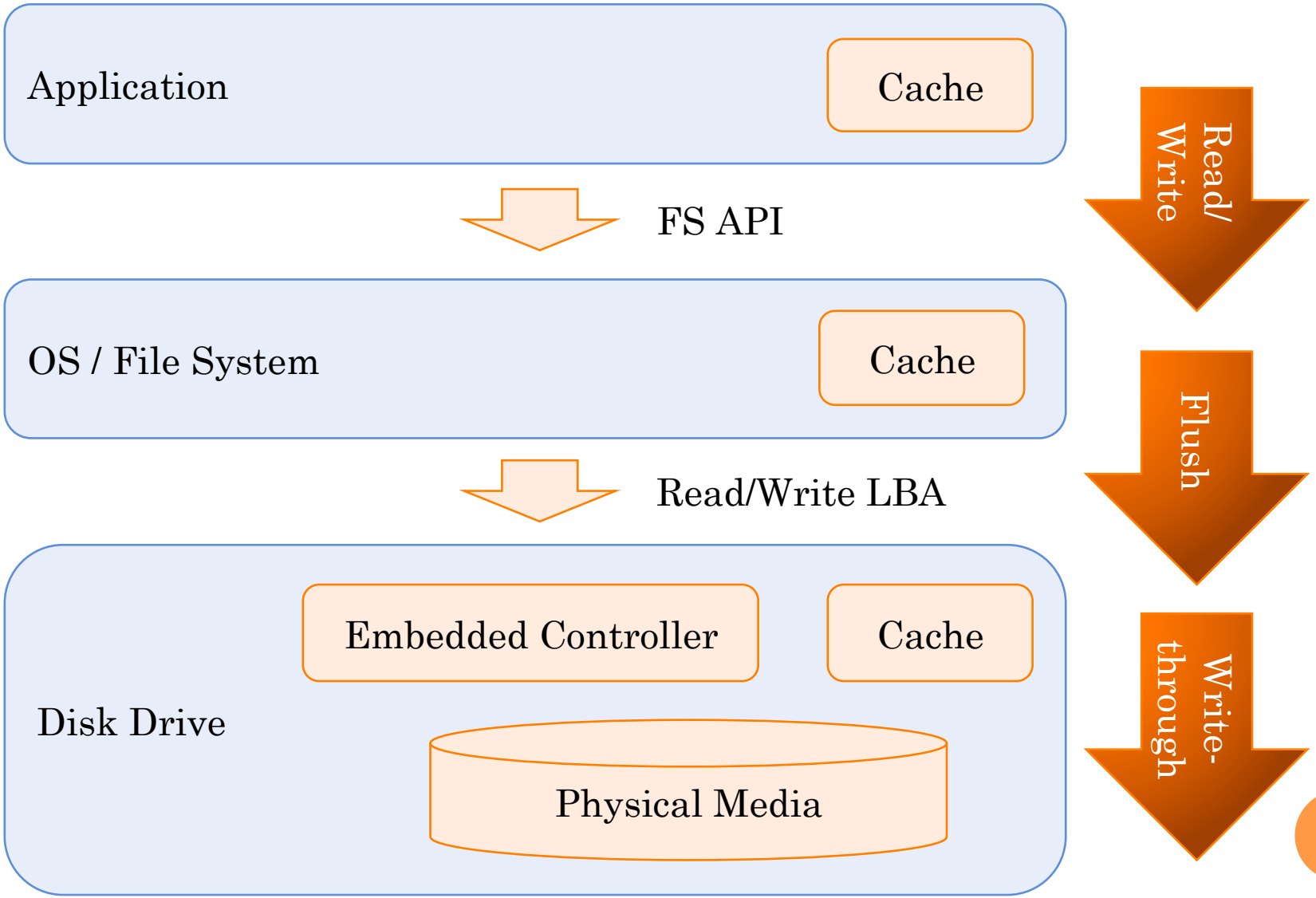
- Combine Drum and Hard Disk…

2. History
2.3 Combine Fixed & Movable Heads

- Fixed and moving heads within hard disk
  - IBM/VS 1.3 writes to Write Ahead Data Set (WADS) (1982).
    - One forced write to each track of the fixed head portion, means write where head is currently located
    - In parallel, block writes of all data to the movable head portion
    - Reads handled by disk cache and movable head portion

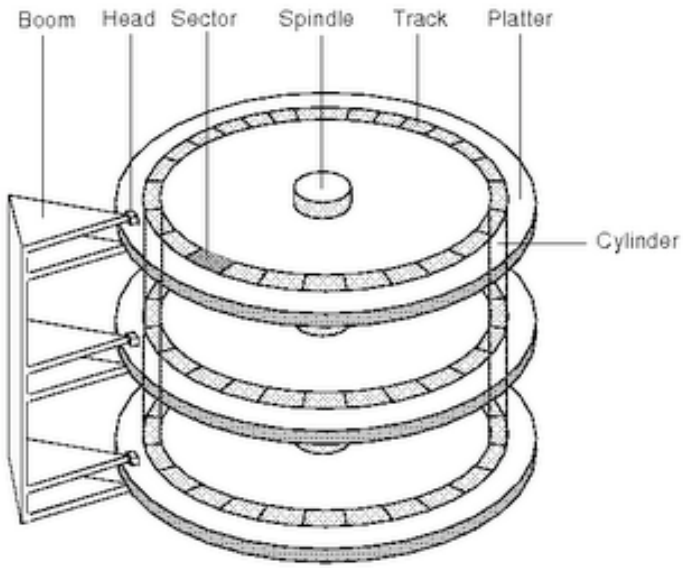
[1] Strickland, J. P., Uhrowczik, P. P., Watts, V. L. IMS/VS: An evolving system. *IBM System Journal*, 21, 4 (1982).
[2] Peterson, R. J., Strickland, J. P. Log write-ahead protocols and IMS/VS logging. *In Proceedings 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Atlanta, Ga., March 1983).
[3] US Patent 4507751 - Method and apparatus for logging journal data using a log write ahead data set. 1985.

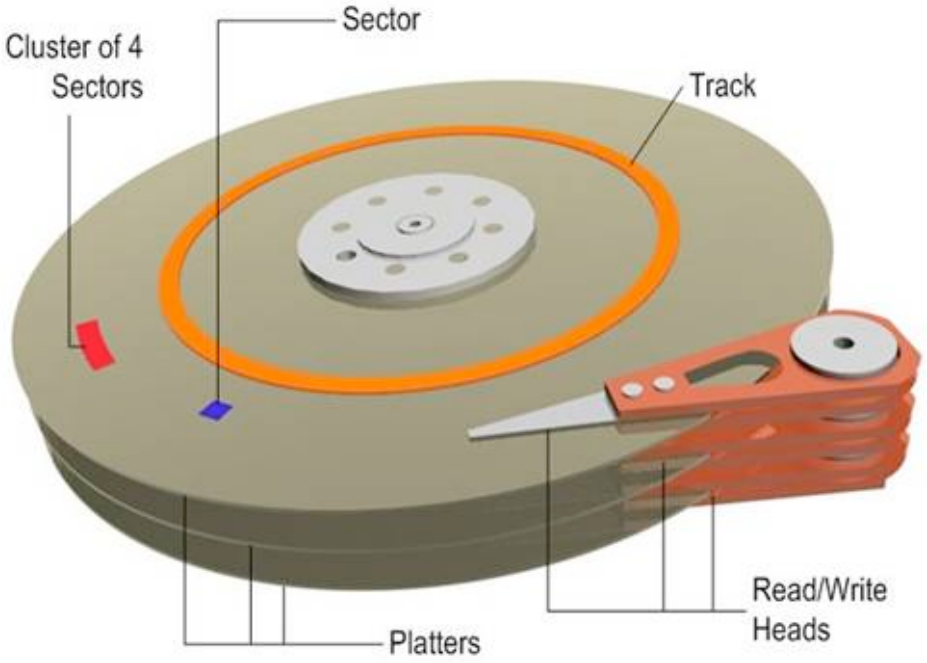
3. Modern I/O Stack



[4] Farley, M. Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and Filing Systems. Chapter 4. *Cisco Press*, 2004.

3. Modern I/O Stack
3.1 Disk Drive





3. Modern I/O Stack
3.1 Disk Drive

- Access physical media via (Cylinder, Track, Sector) = CTS
  - Remap damaged sectors
- Costs: seek (2-6 ms, minimum 0.6 ms), rotational (4-8 ms), head switch, transfer latencies + queuing delay
  - Seek cost varies non-linearly
- Cache for reading and writing
  - Up to 30 second delay before write to cache is executed on the physical media
  - Reorder operations to reduce latencies
- Zoned-bit recording varies density on tracks
  - Fastest throughput for outermost tracks
- Partitions are assigned from outermost track inwards

[4] Farley, M. Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and Filing Systems. Chapter 4. *Cisco Press*, 2004.

3. Modern I/O Stack
3.2 File System Interface

- The file system keeps track of files organized into a directory structure
  - Traditionally for one disk partition
  - Metadata (file structure, data location and other information) + data (what's in the file)

- Deals with the disk drive via Logical Block Addressing (LBA), a single flat address space of blocks
  - This makes optimizations harder at this level
  - Allows the disk to do its own optimizations
  - Allows the disk to be more reliable via remapping

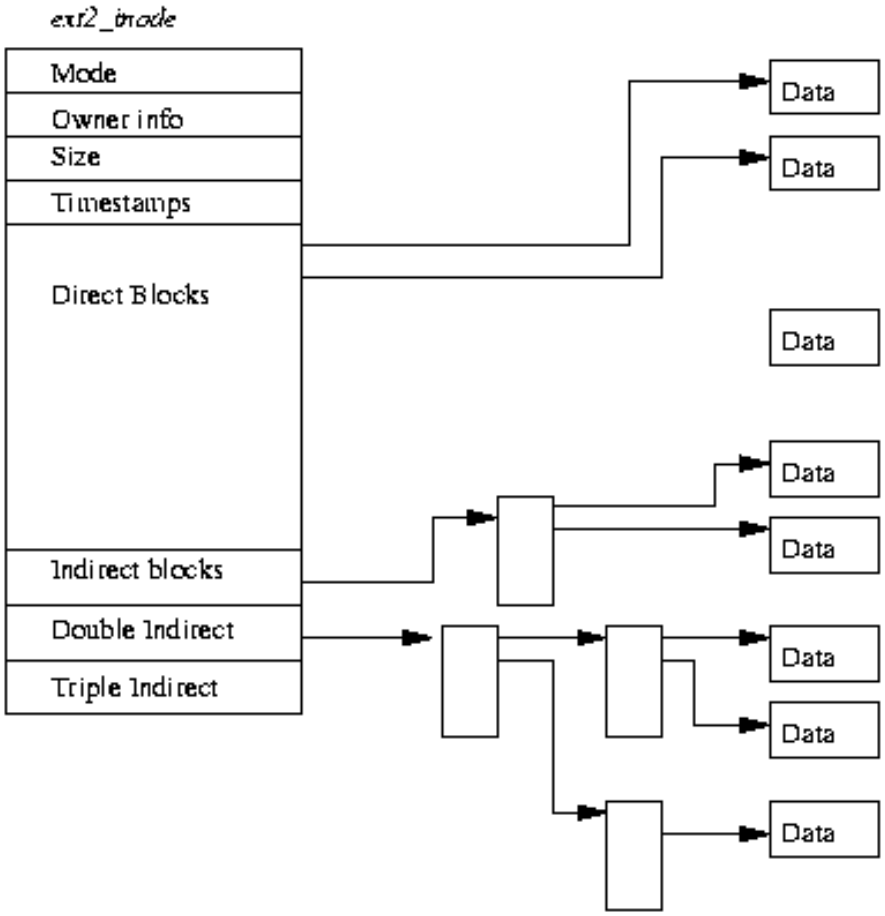
[4] Farley, M. Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and Filing Systems. Chapter 4. *Cisco Press*, 2004.

3. Modern I/O Stack
3.3 Traditional File Systems

- Idea: Store metadata in tree of directory nodes and inodes where leaves are blocks of data for the files
  - Try to sequentially allocate blocks to a file so that reading is faster
  - Writes to existing blocks of a file are executed to that exact location on disk
  - Delayed writes can cause corruption on failure
  - Example: ext2

[5] McKusick, M. K., Joy, W. N., Leffler, S. J., Fabry, R. S. A fast file system for UNIX. *ACM Transactions on Computer Systems (TOCS)*, v.2 n.3, p.181-197, Aug. 1984.

3. Modern I/O Stack
3.3 Traditional File Systems



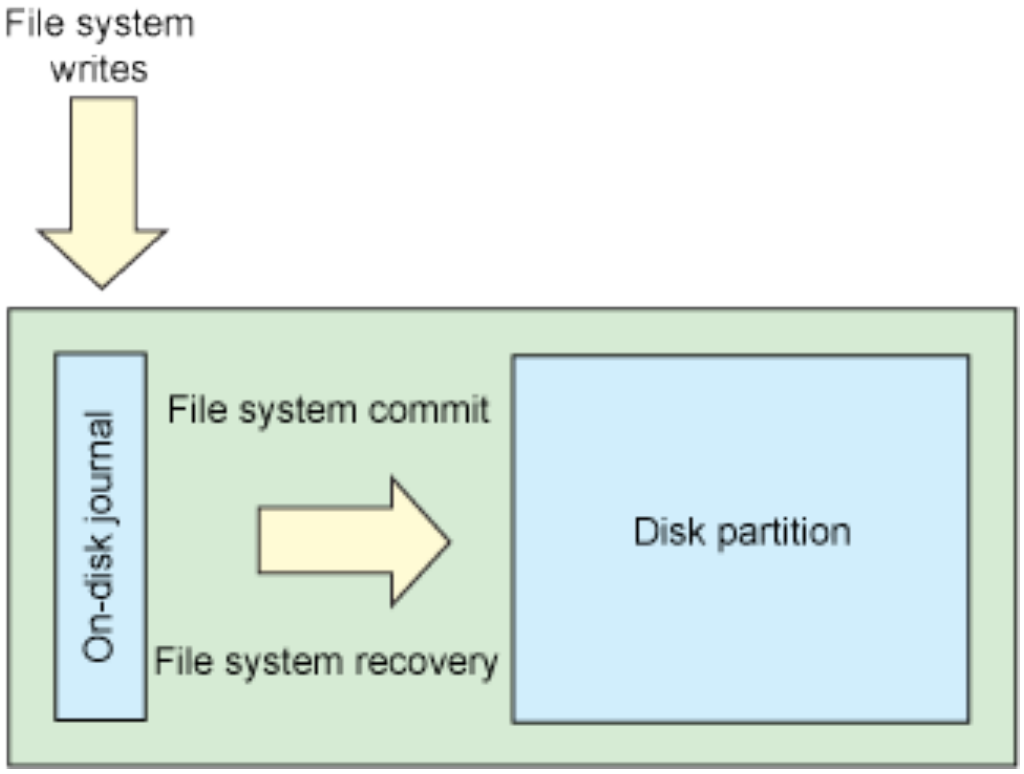
http://www.zimbio.com/Linux/articles/738/Part+II+Object+File+Systems+Legacy+Unix+Linux

3. Modern I/O Stack
3.4 Journaling File Systems

- Idea: Add a journal (log) of changes that you are going to make to the files system before you make them
  - Better recovery and fault tolerance
  - Reads use the normal file system
  - Writes happen twice (journal + normal file system), but the journal is sequential and batched for group commit
    - Could journal only the metadata (common) which is small
  - Example: ext3

[6] Tweedie, S. C. Journaling the Linux ext2fs File System. *In the Fourth Annual Linux Expo*, Durham, North Carolina, May 1998.

3. Modern I/O Stack
3.4 Journaling File Systems



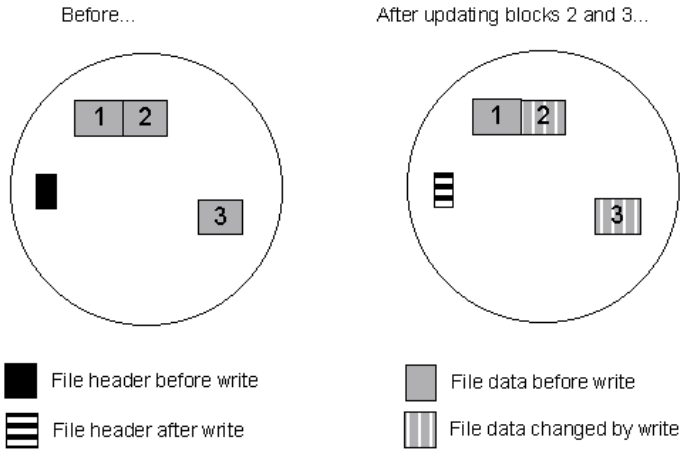
http://www.ibm.com/developerworks/linux/library/l-journaling-filesystems/

3. Modern I/O Stack
3.5 Log-Structured File Systems

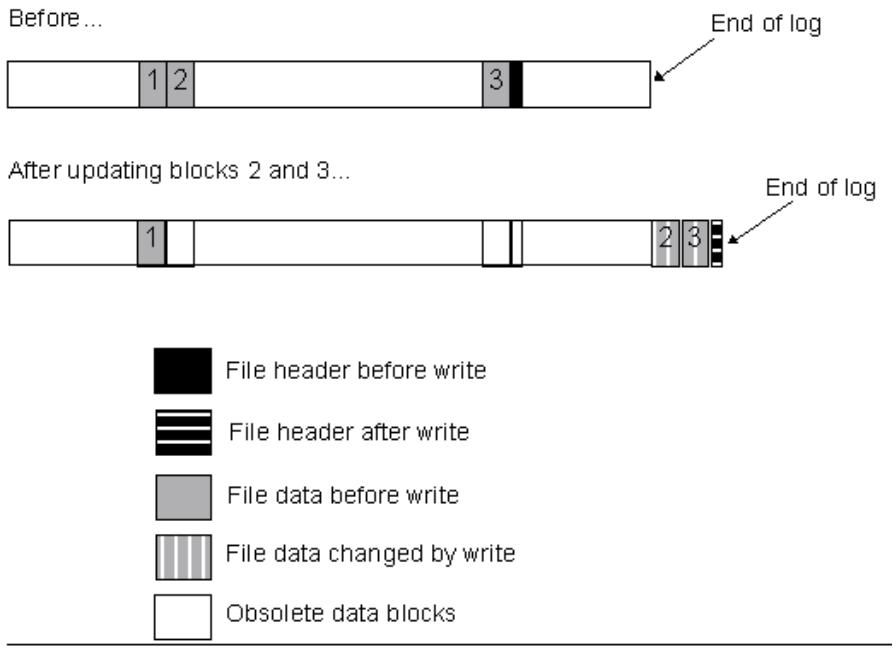
- Idea: Treat the entire disk as one log and put writes to files at the end of the log
  - Need cleanup and compaction to allow the log to loop around
  - Fast writes because of batching and group commit to end of log
  - Fragmentation of file on read (cache may solve this)

[7] Rosenblum, M. and Ousterhout, J. K. The design and implementation of a log-structured file system. ACM Transactions on Computer Systems, 10, 1 (1992), 26-52.

3. Modern I/O Stack
3.5 Log-Structured File Systems



Normal File System



Log-Structured File System

http://www.outflux.net/projects/lfs/what_lfs_is.html

4. Common Optimization Techniques

- Caching reads
  - Removes or postpones lots of issues with fragmentation
  - Do different levels of cache work well together?
- Reorder operations
- Prefetching
- Replicas of data (even on a single disk)

- Buffering/batching writes
  - Potential data loss on failure
  - If writes are transactional, then you're trading latency for throughput

- Short-stroking disk
  - Use only the outer tracks of the disk to reduce seek time
  - Align with zoned-bit recording increases throughput
  - Usually implemented using partitions

- Use non-volatile memory (most common is flash)
  - Solid state drives (SSD)
  - Hybrid drives = flash + hard disk
- Use multiple disks
  - NAS/SAN/RAID includes extra cache memory

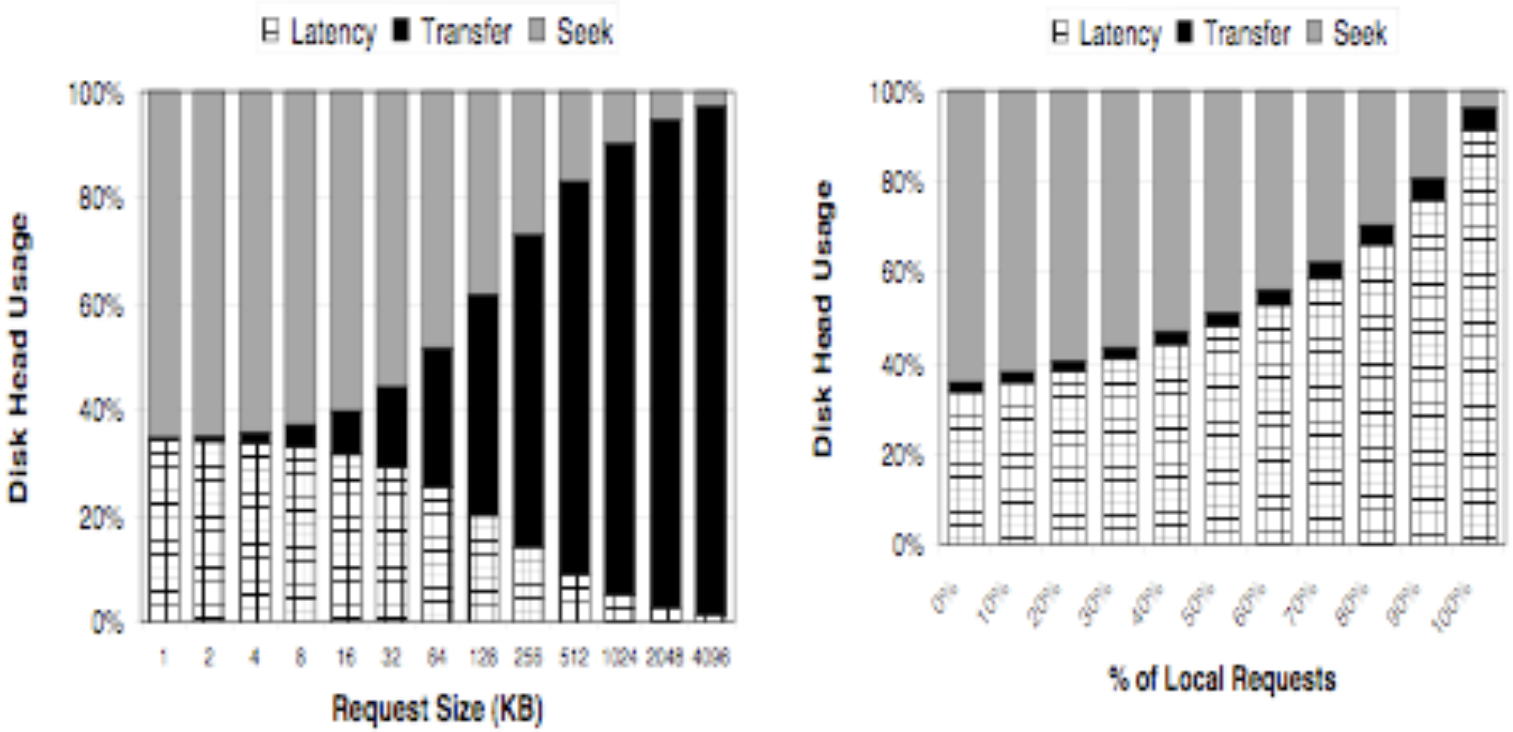
[8] Hsu, W. and Smith, A. J. The performance impact of I/O optimizations and disk improvements. IBM Journal of Research and Development, March 2004, Volume 48, Issue 2, 255-289.

5. Unusual Optimization Techniques
5.1 Modeling the Disk in Software

- Need to know how the disk is laid out
  - Go from LBA to CTS addressing
  - Include remapping of sectors
- Need to know where the disk head is located
  - Can be done in software
  - When return from new read/write you know where the head is (+ processing time)
  - Keep this accurate by issuing new reads/writes as needed
- Model scheduling algorithm
  - Predict order of execution of operations sent to the disk

5. Unusual Optimization Techniques
5.2 Freeblock Scheduling

- Idea: Replace a disk drive's rotational latency delays with useful background media transfers



[9] Lumb, C. R., Schindler, J., Ganger, G. R., Nagle, D. F. and Riedel, E. Towards higher disk head utilization: extracting free bandwidth from busy disk drives. *In Anonymous OSDI'00: Proceedings of the 4th Conference on Operating System Design & Implementation*. (San Diego, California), 87-102. 2000.

[10] Lumb, C. R., Schindler, J., Ganger, G. R. Freeblock Scheduling Outside of Disk Firmware. *In Proceedings of the First USENIX Conferenceon on File and Storage Technologies (FAST'02)*, Monterey, CA, January 2002.

5. Unusual Optimization Techniques
5.2 Freeblock Scheduling

- Applications
  - Segment cleaning (e.g. LFS)
  - Data mining (e.g. indexing for search)

- In firmware (OSDI 2000)
  - 20-50% of disk's bandwidth can be provided to background applications
  - 47 full disk scans per day on an active 9 GB disk (last 5% takes 30% of the time)

- In software (FAST 2002)
  - 15% of disks potential bandwidth can be provided to background applications
  - 37 full disk scans per day on active 9 GB disk

5. Unusual Optimization Techniques
5.3 Eager Writing

- Idea: Execute writes in free sectors near the disk head to reduce write latency
  - Usually used for transactional writes

- Issues
  - How to ensure there are free sectors near the head
  - Fragmentation for reads (cache may hide this)
    - Linking fragments together
    - Defragmentation method
  - Garbage collection of sectors
  - Wasted portions of disk (how much?)
  - Recovery from system failure

5. Unusual Optimization Techniques
5.4 Low Latency Write-Ahead Log

- Idea: Write log entries using eager writing and reconstruct their order using a log sequence number (LSN)
  - Write in a cylinder until you reach a usage threshold, then move to the next cylinder to guarantee free sectors near disk heads
  - Scan used cylinders on recovery
  - Use for logging disk of transactional system

[11] Hagmann, R. Low Latency Logging. Technical Report CSL-91-1, *Xerox Corporation*, Palo Alto, CA, February 1991.

5. Unusual Optimization Techniques
5.5 Virtual Logs

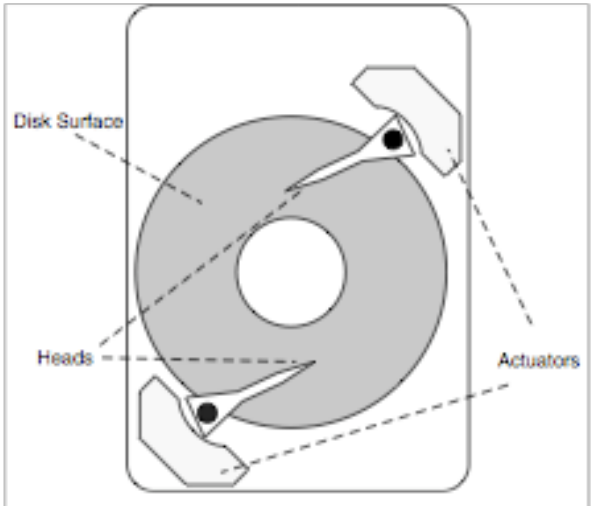
- Idea: Use eager writing to extend performance of LFS
  - Use back chaining to connect portions of the log
    - Extend to a tree to allow skipping obsolete entries

  - Compact free space using disk idle bandwidth
    - Copy chunks from one track to holes in another
    - A new empty track is the end result

[12] Wang, R. Y., Anderson, T. E., Patterson, D. A. Virtual log based file systems for a programmable disk. *In Proceedings of the Symposium on Operating Systems Design and Implementation*, 1999, pp. 29–43.

5. Unusual Optimization Techniques
5.6 Dual Actuator Disks

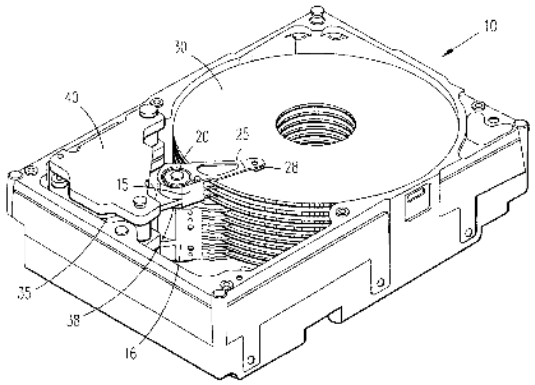
- Use a log-structure file system
- Reads and writes on a single actuator disk force lots of seeks
  - Use one actuator for writing to the end of the log, use the other for reads
- Benefits:
  - Approximately 0 seeks per write
  - Reads and writes do not cause lots of seeks
  - Heads can be smaller and simpler if they do only one function
  - Combine with eager writing to reduce rotational latency
- Issues:
  - Costs of disk
  - Can't move two actuators at once

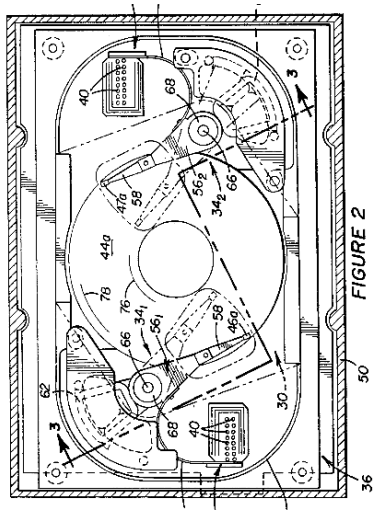


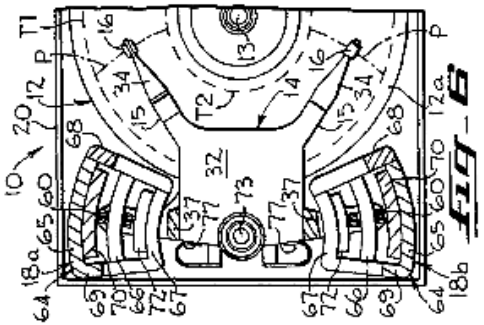
[13] Chandy, J. A. Dual actuator logging disk architecture and modeling. Journal of System Architecture, 53, 12 (2007), 913-926.

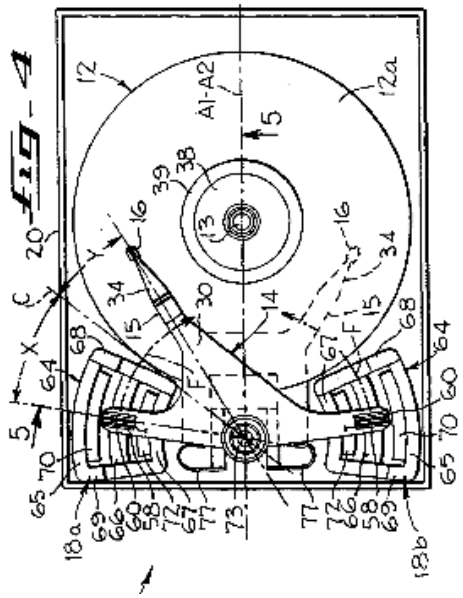
5. Unusual Optimization Techniques
5.6 Dual Actuator Disks

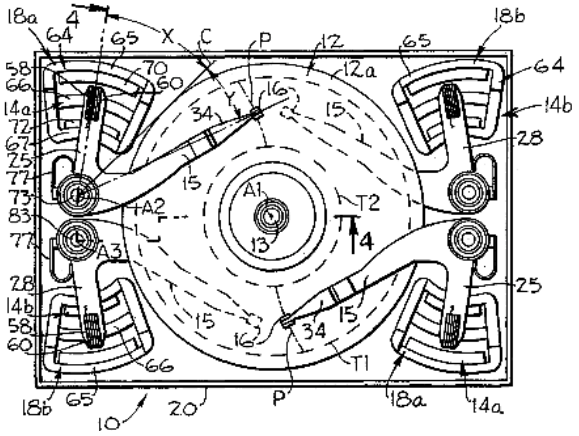
- There are many patents for multiple actuator disks

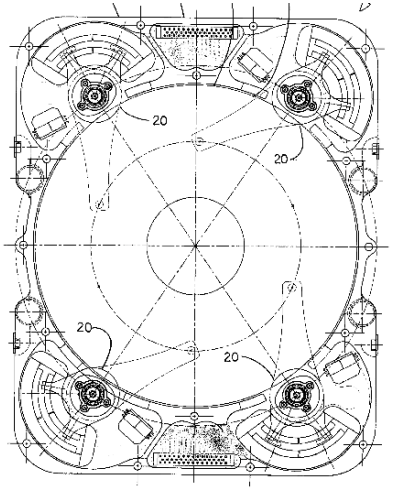












5. Unusual Optimization Techniques
5.7 Track-Based Logging

- Idea: Use one small disk which executes log writes one per track for low latency writes, combined with a normal disk for reads.

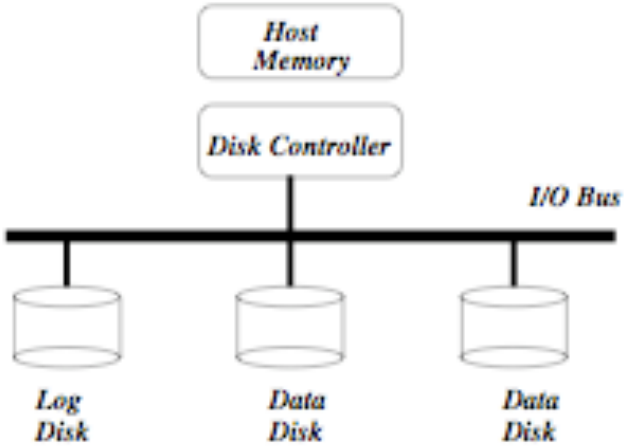


Figure 1. *Hardware Organization of the Trail architecture, which consists of a log disk and one or multiple data disks that it serves. The Trail driver implements track-based logging using a portion of the host memory and the log disk.*

[14] Chiueh, T. C. Trail: A track-based logging disk architecture for zero-overhead writes. In Proceedings of the International Conference on Computer Design, 1993, pp. 339–343.
[15] Chiueh, T. C. Huang L. Track-based disk logging. In Proceedings of the International Conference on Dependable Systems and Networks, 2002, pp. 429–438.

5. Unusual Optimization Techniques
5.7 Track-Based Logging

- Similar to IMS/VS 1.3 WADS setup, but using a non-fixed head disk for logging.
  - Free sectors are always under disk head so no waiting to write
- Transaction processing workload (ICCD 1993)
  - Write latency >10x improvement
  - Read latency better in all cases
- TPC-C (DSN 2002)
  - Throughput is 62.7% higher
  - DB logging related disk I/O overhead is reduced by 42%

5. Unusual Optimization Techniques
5.8 Restrict Locations to Write

- Idea: Append write data to one of X files based on which is closest to the disk head.
  - X is a tunable setting
  - Efficient use of disk space because only X files and each is compact
  - Recovery is fast, because you only need to read X files

I submitted something like this to FAST 2010, though without a good background in the area.

6. Conclusions / Take Away Points

- Freeblock scheduling can do useful background work on an active disk without affecting foreground processes.

- Eager writing can be very valuable, but maintaining good performance can be tricky
  - Why are these systems not used in practice?

REFERENCES

History:

[1] Strickland, J. P., Uhrowczik, P. P., Watts, V. L. IMS/VS: An evolving system. *IBM System Journal*, 21, 4 (1982).

[2] Peterson, R. J., Strickland, J. P. Log write-ahead protocols and IMS/VS logging. *In Proceedings 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Atlanta, Ga., March 1983).

[3] US Patent 4507751 - Method and apparatus for logging journal data using a log write ahead data set. 1985.

Modern IO Stack:

[4] Farley, M. Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and Filing Systems. Chapter 4. *Cisco Press*, 2004.

File Systems:

[5] McKusick, M. K., Joy, W. N., Leffler, S. J., Fabry, R. S. A fast file system for UNIX. *ACM Transactions on Computer Systems (TOCS)*, v.2 n.3, p.181-197, Aug. 1984.

[6] Tweedie, S. C. Journaling the Linux ext2fs File System. *In the Fourth Annual Linux Expo*, Durham, North Carolina, May 1998.

[7] Rosenblum, M. and Ousterhout, J. K. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10, 1 (1992), 26-52.

Normal Disk Optimizations:

[8] Hsu, W. and Smith, A. J. The performance impact of I/O optimizations and disk improvements. *IBM Journal of Research and Development*, March 2004, Volume 48, Issue 2, 255-289.

Freeblock Scheduling:

[9] Lumb, C. R., Schindler, J., Ganger, G. R., Nagle, D. F. and Riedel, E. Towards higher disk head utilization: extracting free bandwidth from busy disk drives. *In Anonymous OSDI'00: Proceedings of the 4th Conference on Operating System Design & Implementation*. (San Diego, California). USENIX Association, Berkeley, CA, USA, 87-102. 2000.

[10] Lumb, C. R., Schindler, J., Ganger, G. R. Freeblock Scheduling Outside of Disk Firmware. *In Proceedings of the First USENIX Conferenceon on File and Storage Technologies (FAST'02)*, Monterey, CA, January 2002.

Low Latency Write-Ahead Log:

[11] Hagmann, R. Low Latency Logging. Technical Report CSL-91-1, *Xerox Corporation*, Palo Alto, CA, February 1991.

Virtual Logging:

[12] Wang, R. Y., Anderson, T. E., Patterson, D. A. Virtual log based file systems for a programmable disk. In Proceedings of the Symposium on Operating Systems Design and Implementation, 1999, pp. 29–43.

Dual Actuator Disks:

[13] Chandy, J. A. Dual actuator logging disk architecture and modeling. *Journal of System Architecture*, 53, 12 (2007), 913-926.

Track-based Logging:

[14] Chiueh, T. C. Trail: A track-based logging disk architecture for zero-overhead writes. In Proceedings of the International Conference on Computer Design, 1993, pp. 339–343.

[15] Chiueh, T. C. Huang L. Track-based disk logging. In Proceedings of the International Conference on Dependable Systems and Networks, 2002, pp. 429–438.

Questions?