

CS E6998-10. Advanced Topics in Network Storage Systems Spring 2004

Kostas Magoutis

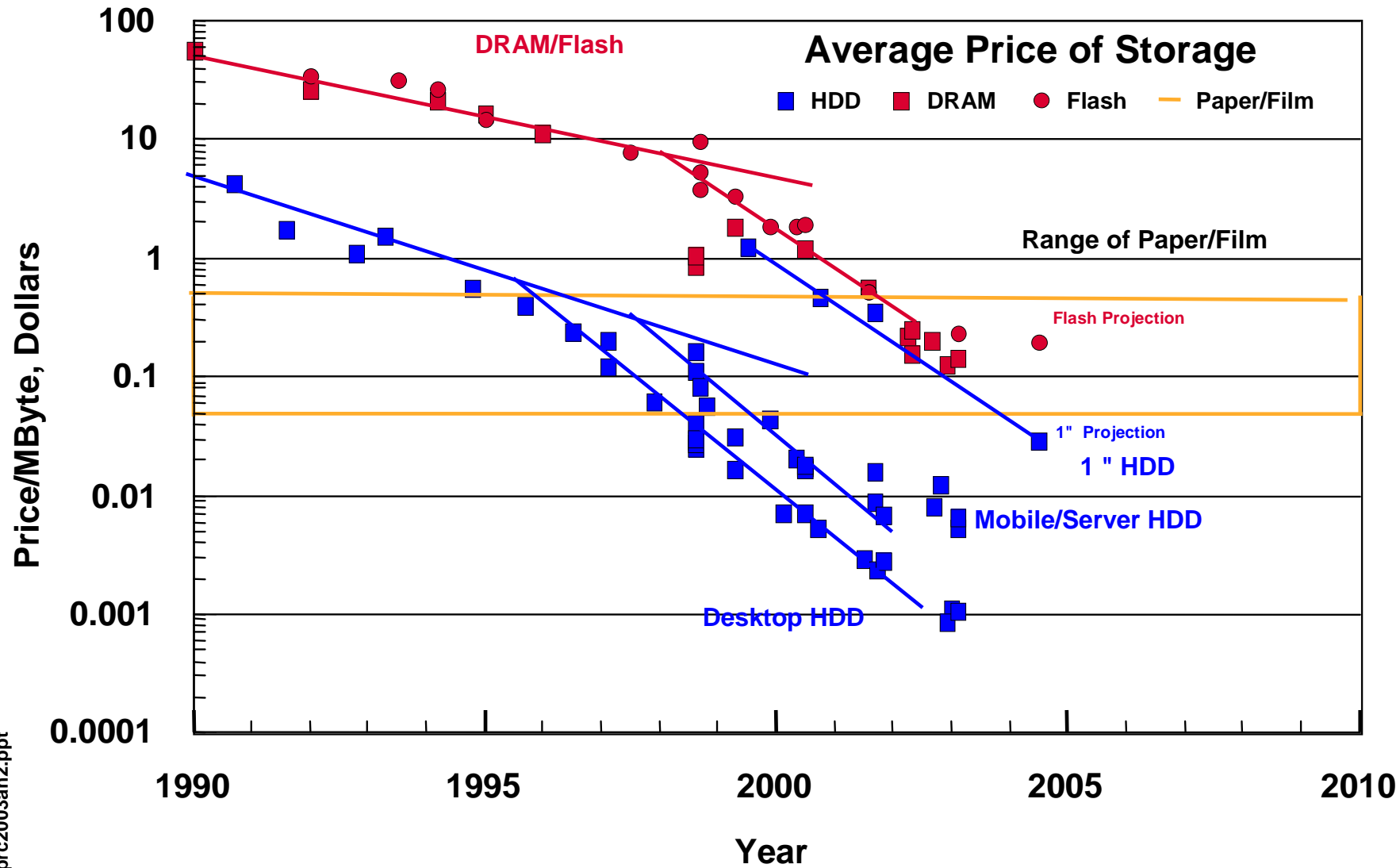
magoutis@cs.columbia.edu

<http://www.cs.columbia.edu/~cs699810>

Introduction

- Why is network storage an important field?
 - Storage demands are growing rapidly
 - Need scalable, reliable, available storage systems.
 - New applications
 - Physics, biology, etc.
- Evolution of the Hard Disk Drive (HDD)
 - Density improved by seven orders of magnitude since 1980
 - Price goes down (chart next slide)

Oemprc2003ah2.ppt



Ed Grochowski

Applications

- Enterprise (business)
- Biology (protein sequencing)
- High-energy physics
- Grid computations

[John Wilkes talk at FAST'03](#)

Themes

- Network storage system structure
- Network storage protocols
- Administration (provisioning, backup, etc.)
- Peer-to-peer
- Wide-area replication
- Application-specific designs

Assignments

- Readings
 - Short reviews due before beginning of class (by email).
- Two homeworks
 - Paper critique.
- Two quizzes
 - 15 minutes, previously announced, at the end of class.
- Project
 - System design and some experimental evaluation (e.g., microbenchmarks); Equivalent to a workshop paper.

Background

- Disk drive
- SCSI bus
- RAID arrays
- File system data structures and consistency
- Atomic transactions
 - Logging.
 - Two-phase commit.

Disk drive

- Geometry
- Disk I/O is performance bottleneck
 - Moore's law.
 - Amdahl's law.
- RAID
 - Parallelism improves performance.
 - Redundancy improves reliability.

SCSI

- Parallel bus
 - Limited length, width, load.
- Asynchronous operation
 - <initiator, target, LUN>
 - SCSI IDs fixed, determine bus arbitration.
 - Can be used for communication between hosts.
- Evolution: Fibre Channel, SSA.

Fibre Channel

- Interconnect topologies
 - Point-to-point.
 - Fabric (switch).
 - Arbitrated loop (ring).
- Physical characteristics
 - Fiber optic connections up to 10kms.
 - 100 MB/s, 200 MB/s and 400 MB/s growth path.
 - Loops up to 126 ports; Switches up to 16 ports.
- Protocol layers
 - FC-0 through 4; FC-4 maps upper-level protocol to FC.
 - Can be used to transfer both SCSI and IP traffic.

RAID

- Low reliability without redundancy
 - Assuming independent failures, MTTF goes down to a few thousand hours.
- Levels
 - RAID-0 (Striped)
 - RAID-1 (Mirrored)
 - RAID-5 (Parity)
- Small-write problem with RAID-5
 - Need up to 4 I/Os (to update parity)

RAID (continued)

- Availability, failover
 - Rebuild on hot spare
 - Operation in “degraded” mode
- MTTF of RAID
 - Excellent! But..
- Reality is different
 - System crashes result to parity inconsistency.
 - Uncorrectable bit errors.
 - Correlated disk failures result to data loss.

File system data structures

- Metadata
 - Directories
 - i-nodes
 - Indirect blocks
 - Free lists
 - Superblock
- Metadata operations involve several steps
 - E.g., `rmdir`
- Updates should preserve metadata consistency (i.e., no stale pointers) in case of a crash

File system consistency

- Older FSES use synchronous writes to ensure order
 - Requires scavenging, e.g., `fsck`
 - Can take hours (or days!) for large file systems
- Need atomic updates
- Logging
 - Use (atomic) write I/O to write “intentions” to a *log* before issuing the actual (in-place) I/Os.
 - Easy recovery: just replay the log.
 - Good performance: Sequential writes (appends), saves I/Os.
 - Example: writes to stripe unit and parity in RAID.

Two-phase commit (2PC)

- Distributed atomic transactions
- Modifications to objects are made on volatile versions at several nodes
- When an action commits, the system must ensure that
 - It either commits everywhere or aborts everywhere, and
 - Its effects are made permanent by writing the modified atomic objects to stable storage.

Two-phase commit (2PC)

Coordinator

Sends "prepare" message

Write "committed" to the log

Send "commit" messages

Write "done" to the log

Participants

Write modified objects to the log

Write "prepared" to the log

Send "prepared" message

Write "committed" record to the log

Perform the in-place updates

Reply "committed"

Baker 1991: Measurements of a Distributed File System (Sprite)

- Validate results of BSD study (1985)
 - File accesses are largely sequential
 - Vast majority of file accesses are to small files
 - Most bytes transferred belong to large files
 - Files are typically open for only a fraction of a second
 - File lifetimes are short

- What changed?
 - File throughput per user increased substantially (x20) and is more bursty
 - Large files have become an order of magnitude larger

Sprite

- Sprite has a client-server structure
 - Enables diskless clients.
 - Clients cache in main memory.
 - Remote paging (client cache shared between FS and VM).
 - Process migration for load balancing.
- Offers “perfect” cache consistency
 - Each read operation returns most recently written data.
 - Multiple-reader/single-writer caching algorithm.
 - Decides whether to cache in the client or not at open time.
 - Caching disabled for concurrent writes.
 - Was that a good choice?