

A Deduplication File System & Course Review

Kai Li

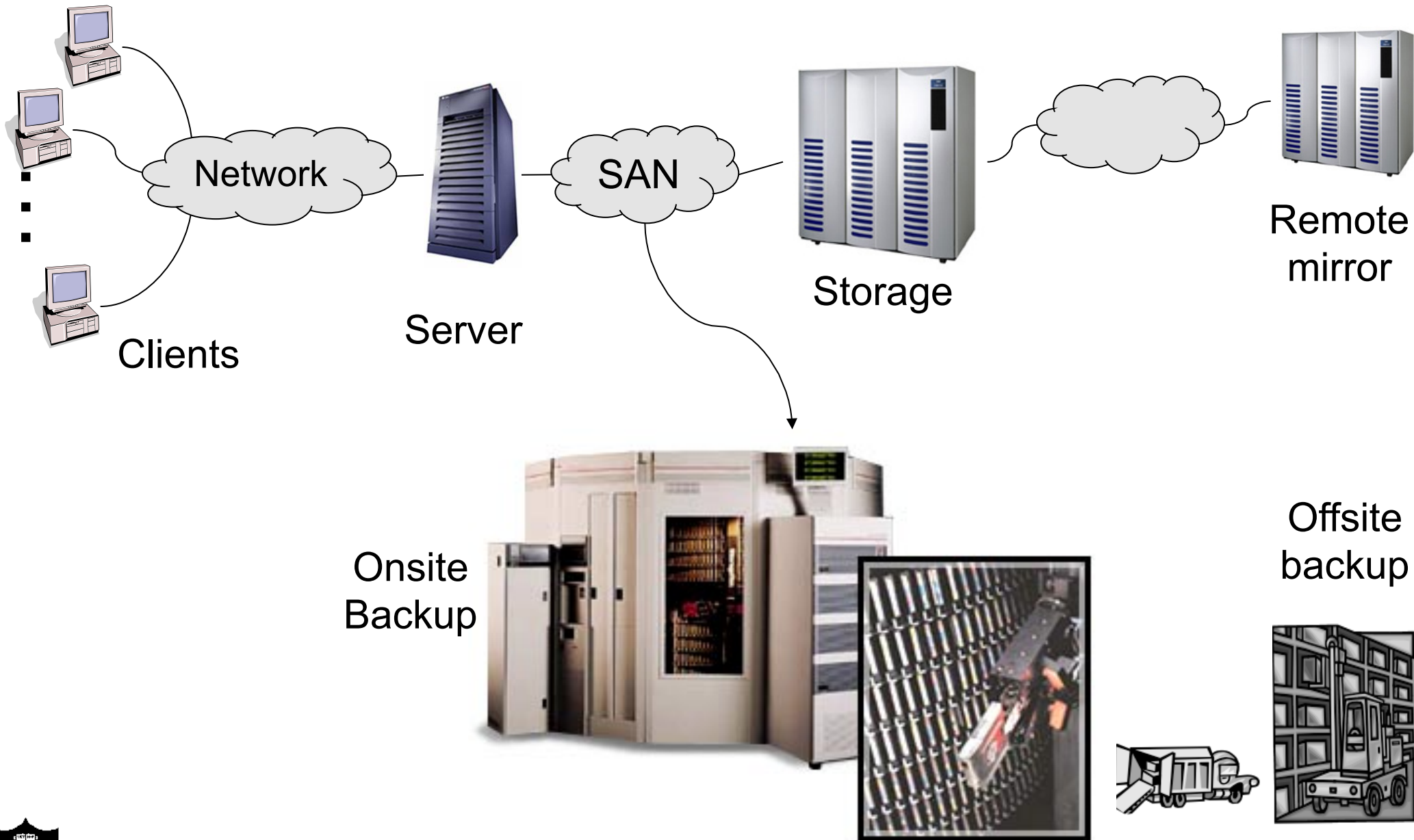


Topics

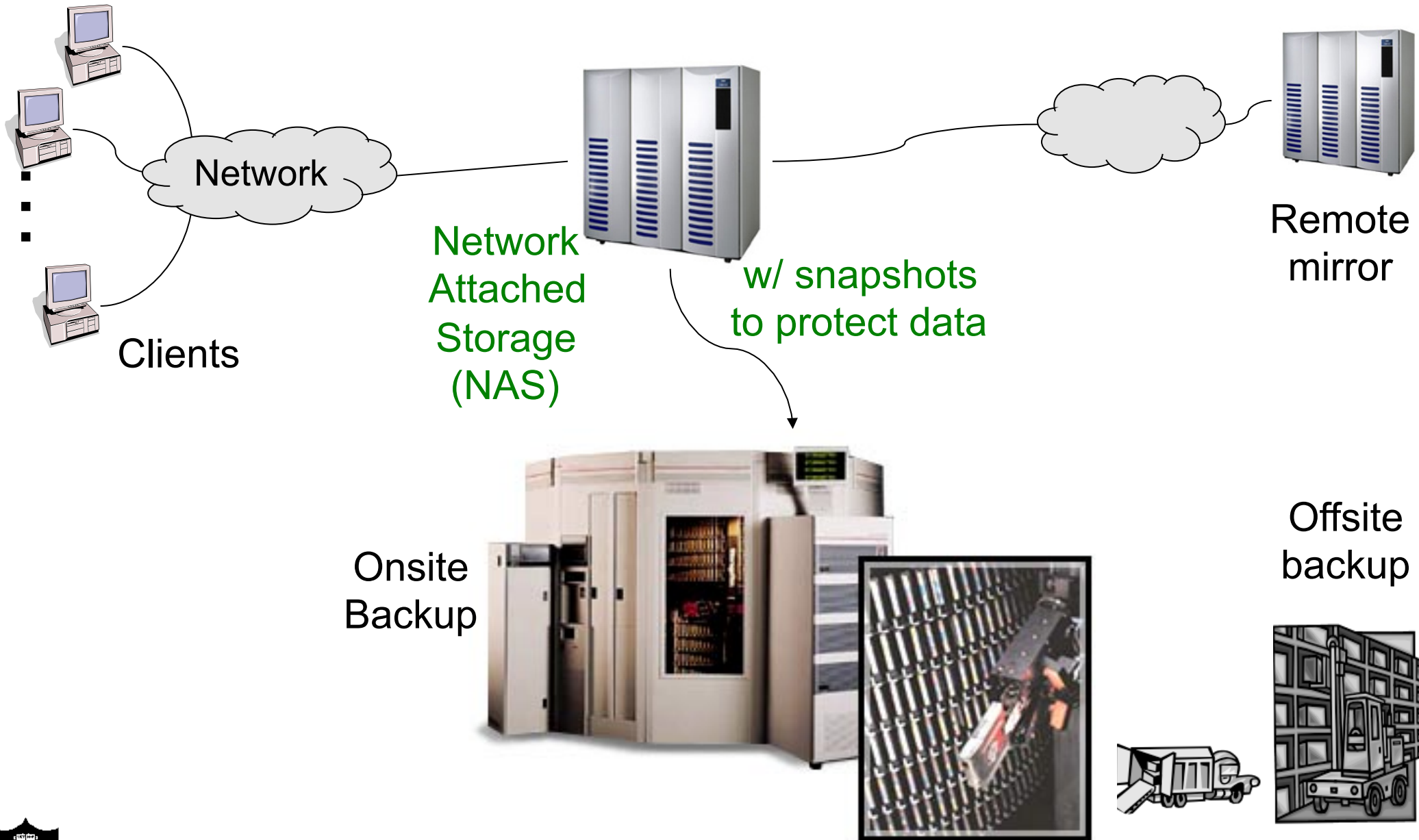
- ◆ A Deduplication File System
- ◆ Review



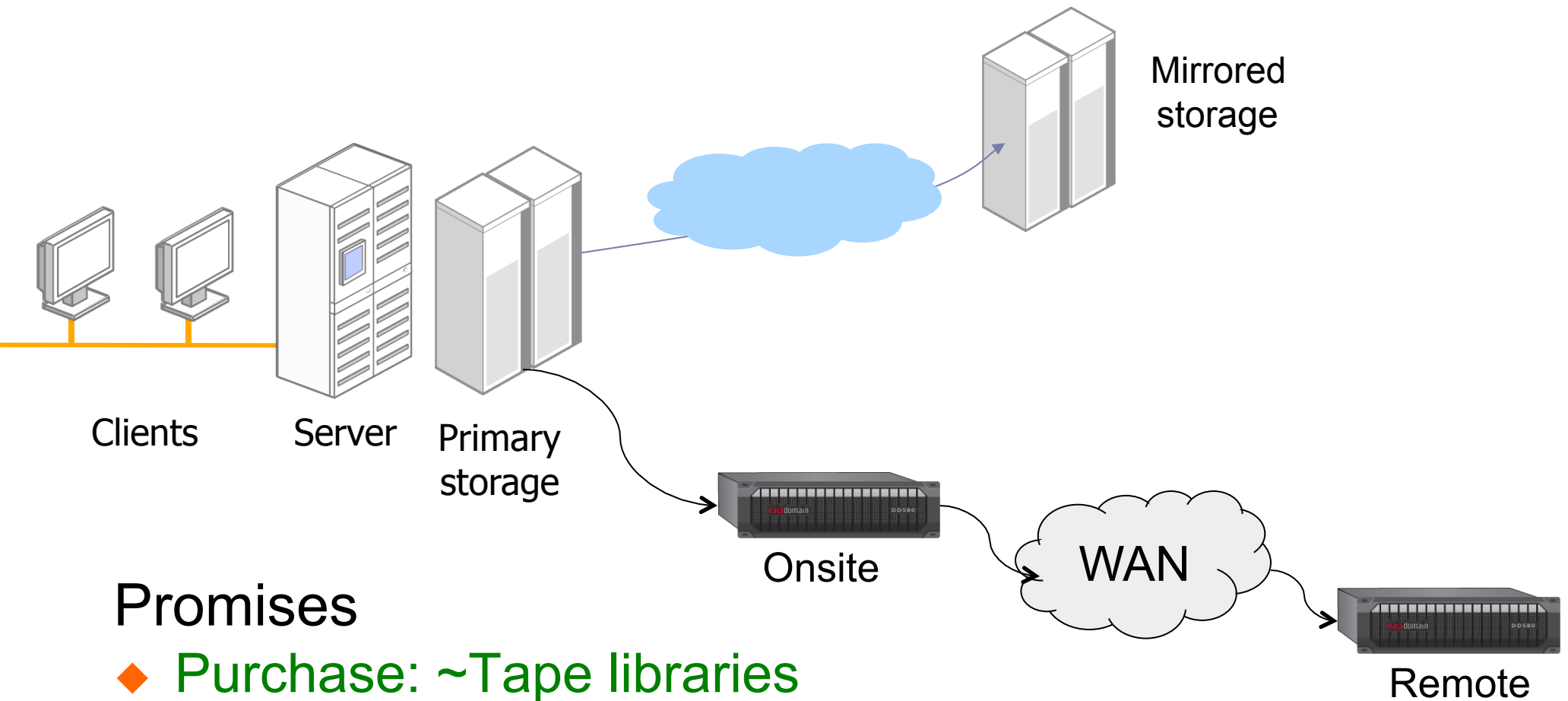
Traditional Data Center Storage Hierarchy



Evolved Data Center Storage Hierarchy



A Data Center with Deduplication Storage



Promises

- ◆ Purchase: ~Tape libraries
- ◆ Space: **10-30X** reduction
- ◆ WAN BW: **10-100X** reduction
- ◆ Power: ~**10X** reduction



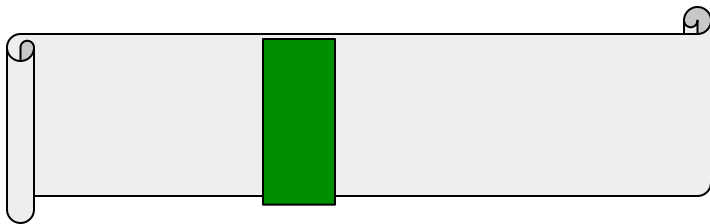
What Is “Deduplication?”

- ◆ Deduplication is **global compression** that removes the redundant segments globally (across **many** files)
- ◆ **Local compression** tools (gzip, winzip, ...) encode redundant strings in a small window (within a file)



Idea of Deduplication

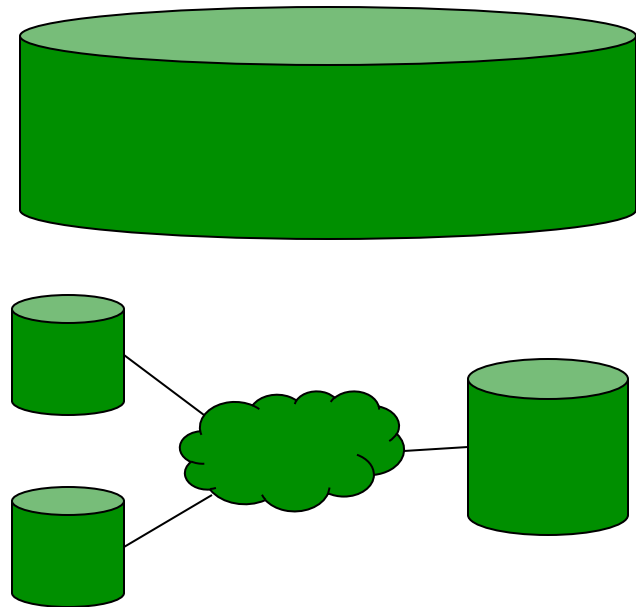
Traditional local
compression



Encode a sliding window
of bytes (e.g. 100K)
[Ziv&Lempel77]

~2-3X compression

Deduplication



~10-50X compression

Large window \Rightarrow more redundant data

Main Deduplication Methods

◆ Fingerprinting

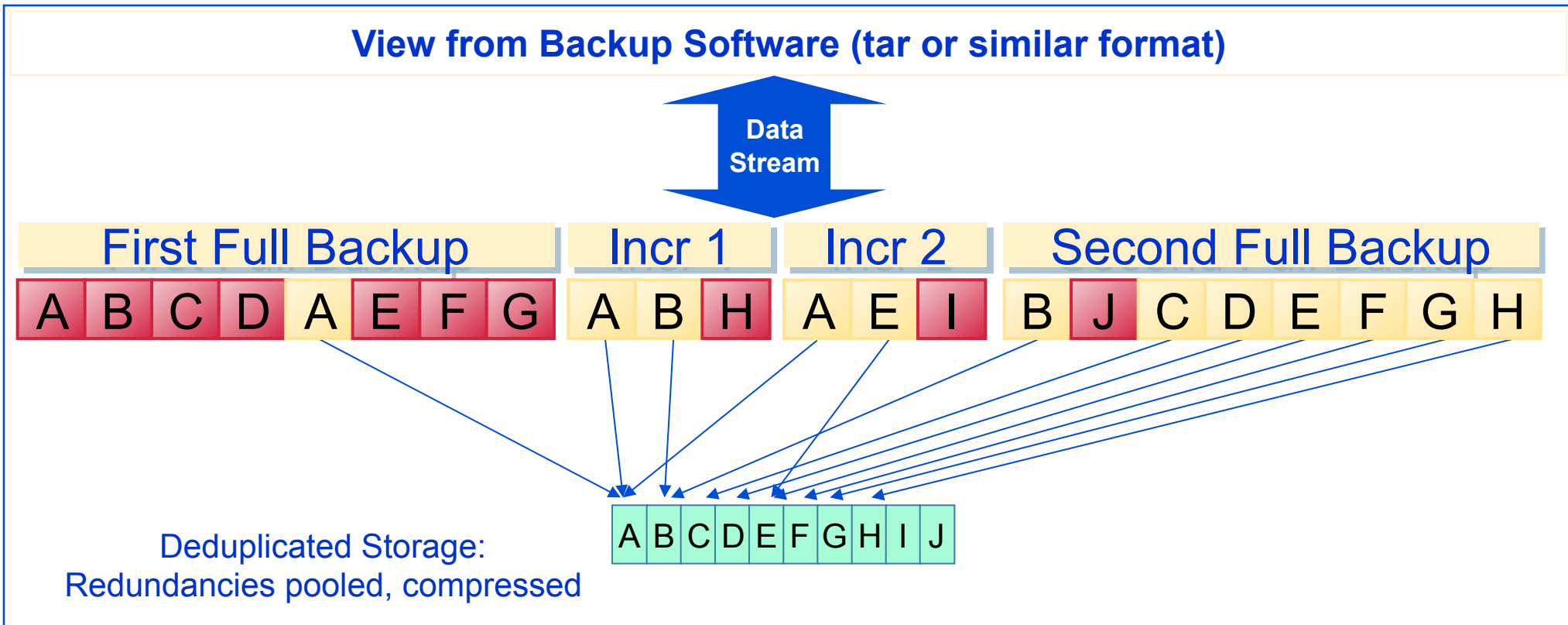
- Computing a fingerprint as the ID for each segment
- Use an index to lookup if the segment is a duplicate
- Is the fingerprint in the index?
- **Yes: duplicate**
- **No: new segment**




◆ Deltas

- Computing a sketch for each segment [Broder97]
- Find the most similar segment by comparing sketches
- **Yes:**
 - **Compute deltas with the most similar segment**
 - **Write delta and a pointer to the similar segment**
- **No: new segment**



Backup Data Example



-  = Unique variable segments
-  = Redundant data segments
-  = Compressed unique segments



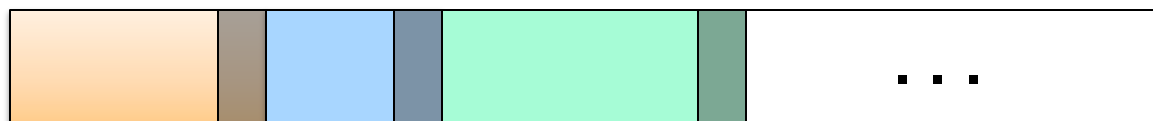
Two Segmentation Methods

- ◆ Fixed size

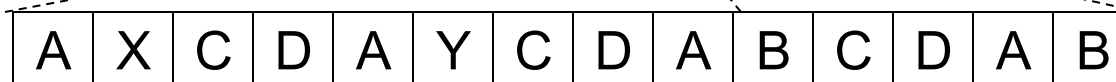


*Cannot handle
adds, deletes
(shifts) well*

- ◆ Content-based, variable size



*Independent of
adds, deletes
(shifts)*



fp = 10110110

fp = 10110100

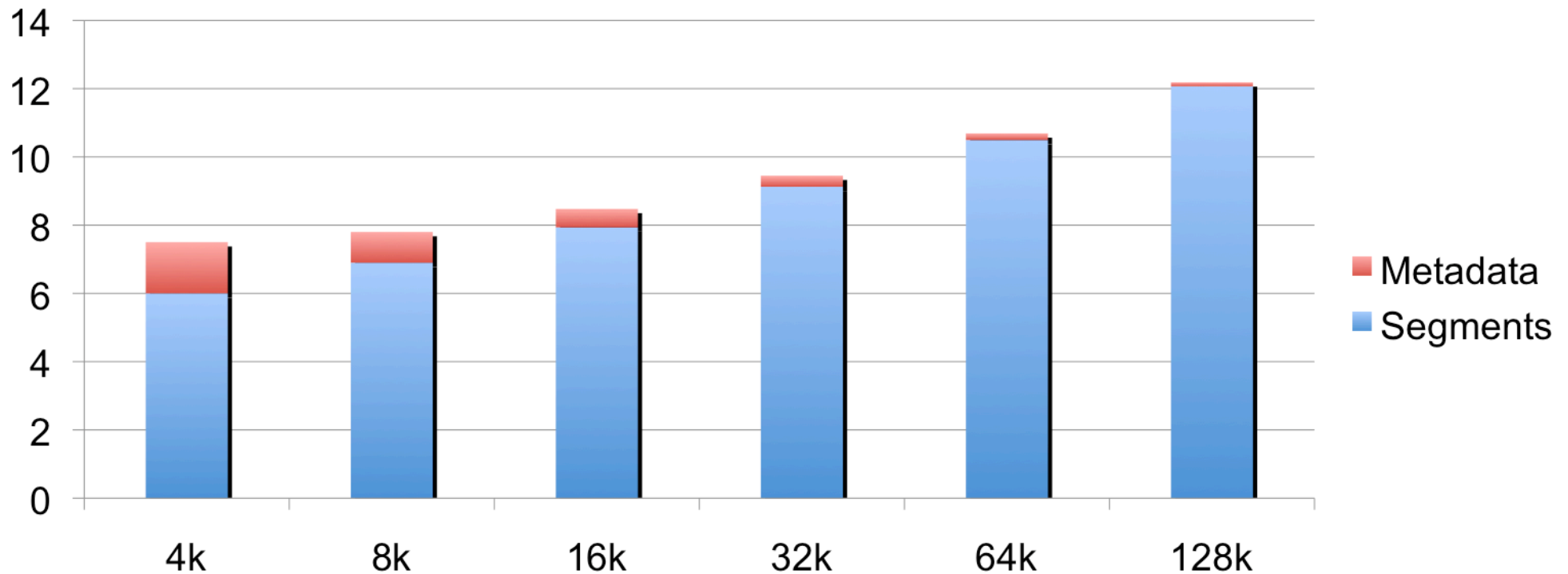
...

“Rolling fingerprinting”

fp = 10110000



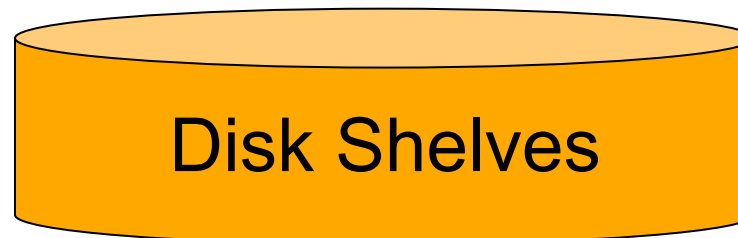
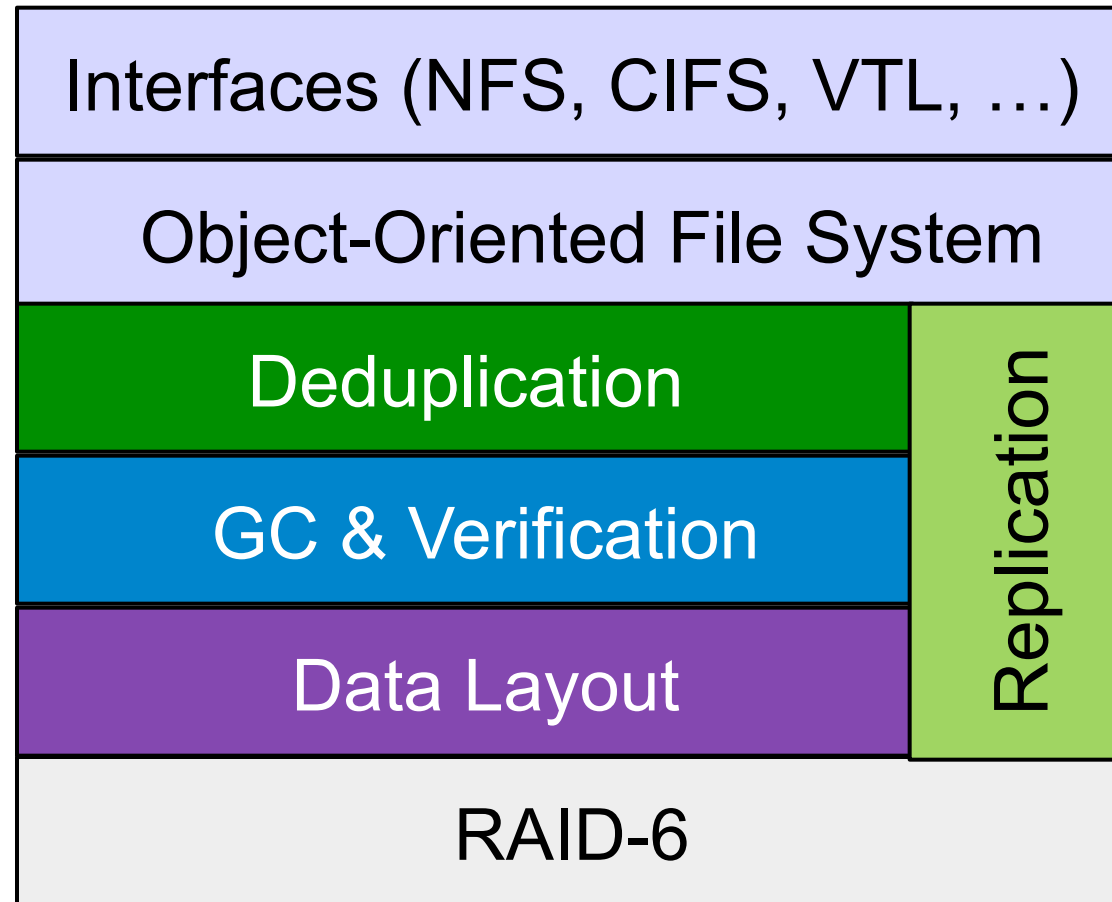
Segment Sizes



- ◆ Double segment size
 - ◆ Increase space for unique segments by 15%
 - ◆ Decrease most metadata by about 50%
 - ◆ Reduce disk I/Os for writes and reads
- ◆ Use the right size for compression ratio and speed



Components in Data Domain Deduplication File System

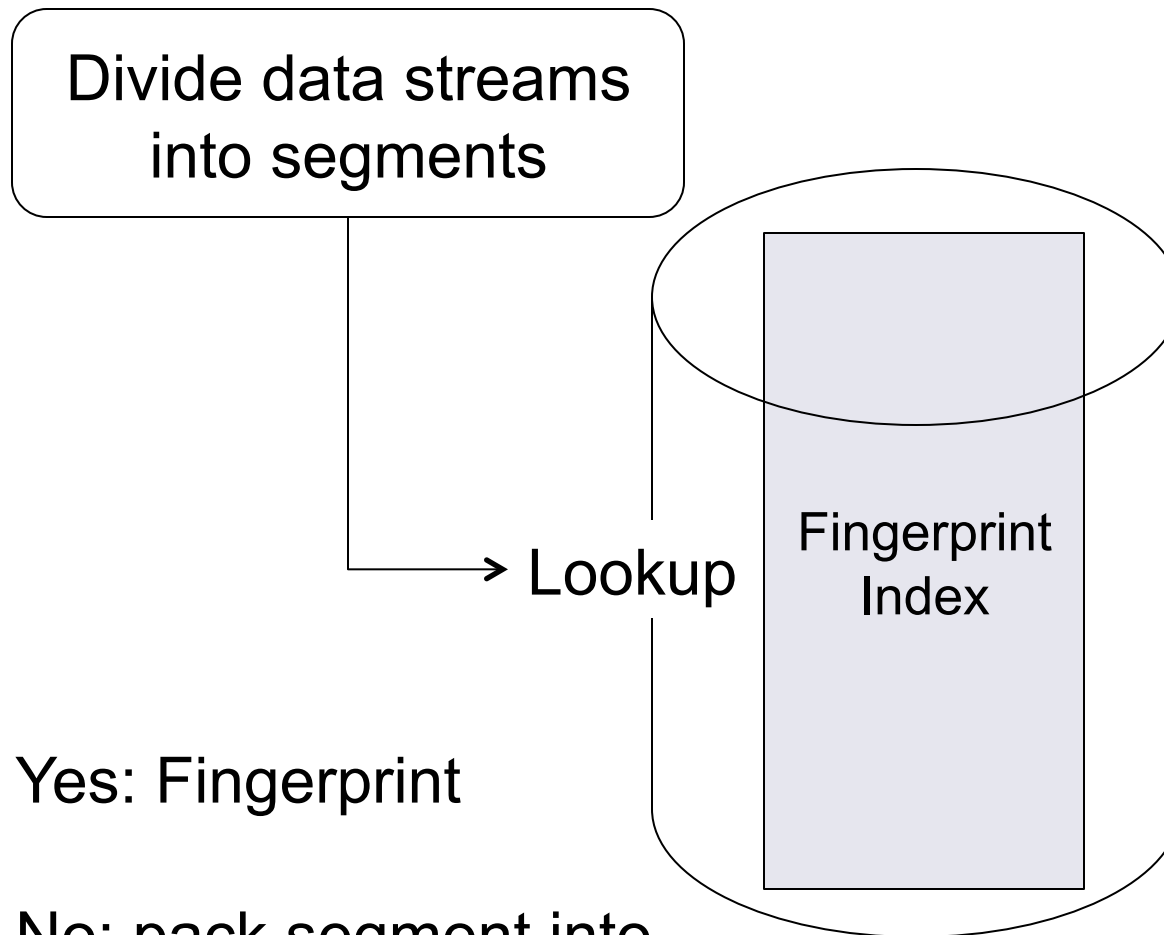


Design Challenges

- ◆ Extremely reliable and self-healing
 - Corrupting a segment may corrupt multiple files
 - NVRAM to store log (transactions)
 - Invulnerability features:
 - Frequent verifications
 - Metadata reconstruction from self-describing containers
 - Self-correction from RAID-6
- ◆ High-speed high-compression at low HW cost
 - Why high speed: data 2X/18 months and 24 hours/day
 - Why high compression: low cost and fewer disks
 - Use commodity server hardware



Revisit the Deduplication Process (Fingerprinting)



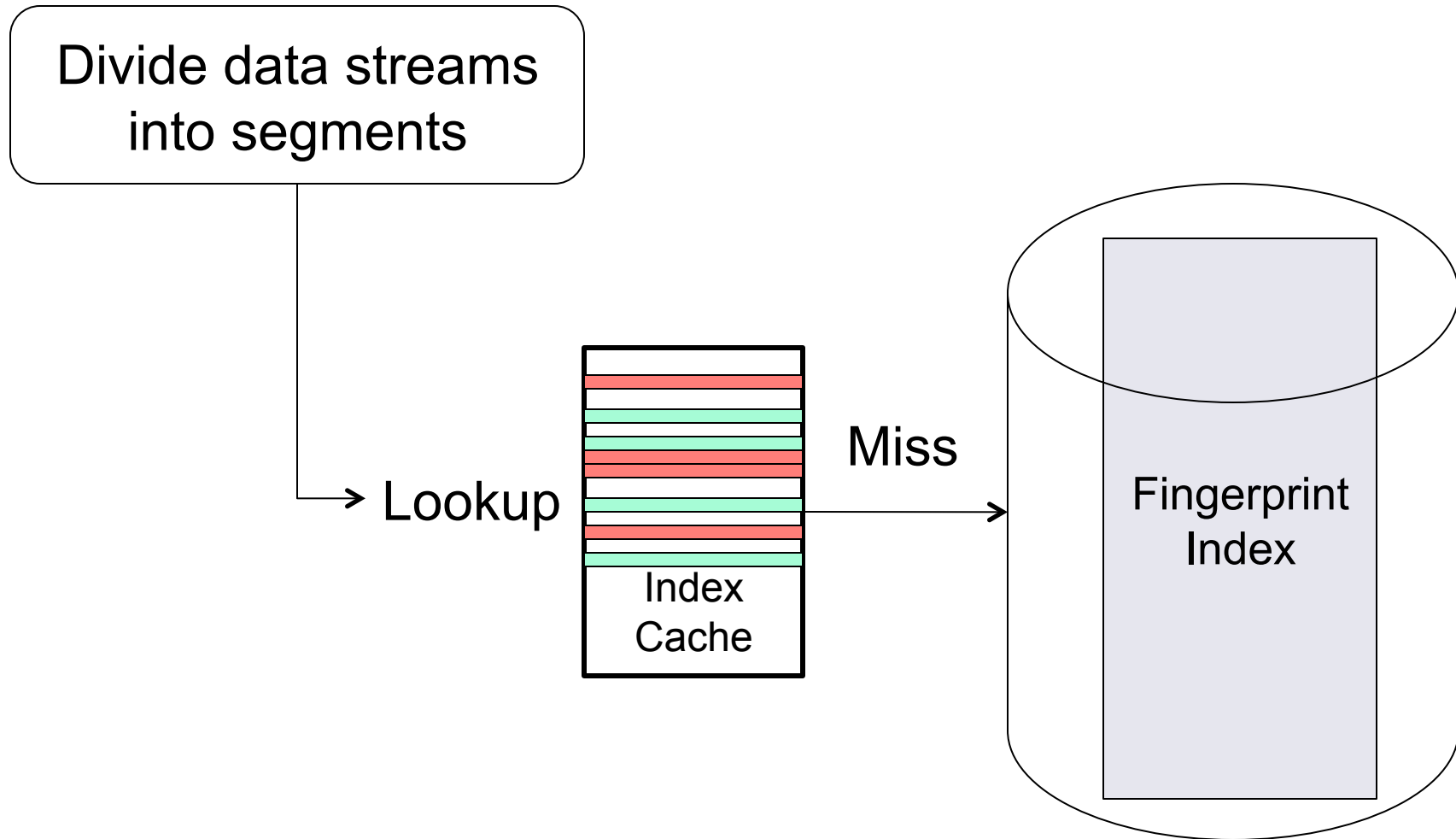
Index size for 80TB
w/ 8KB segments
= $(80\text{TB}/8\text{KB}) * 20\text{B}$
= **200GB!**

Yes: Fingerprint

No: pack segment into
container, apply
local compression,
write out to disk



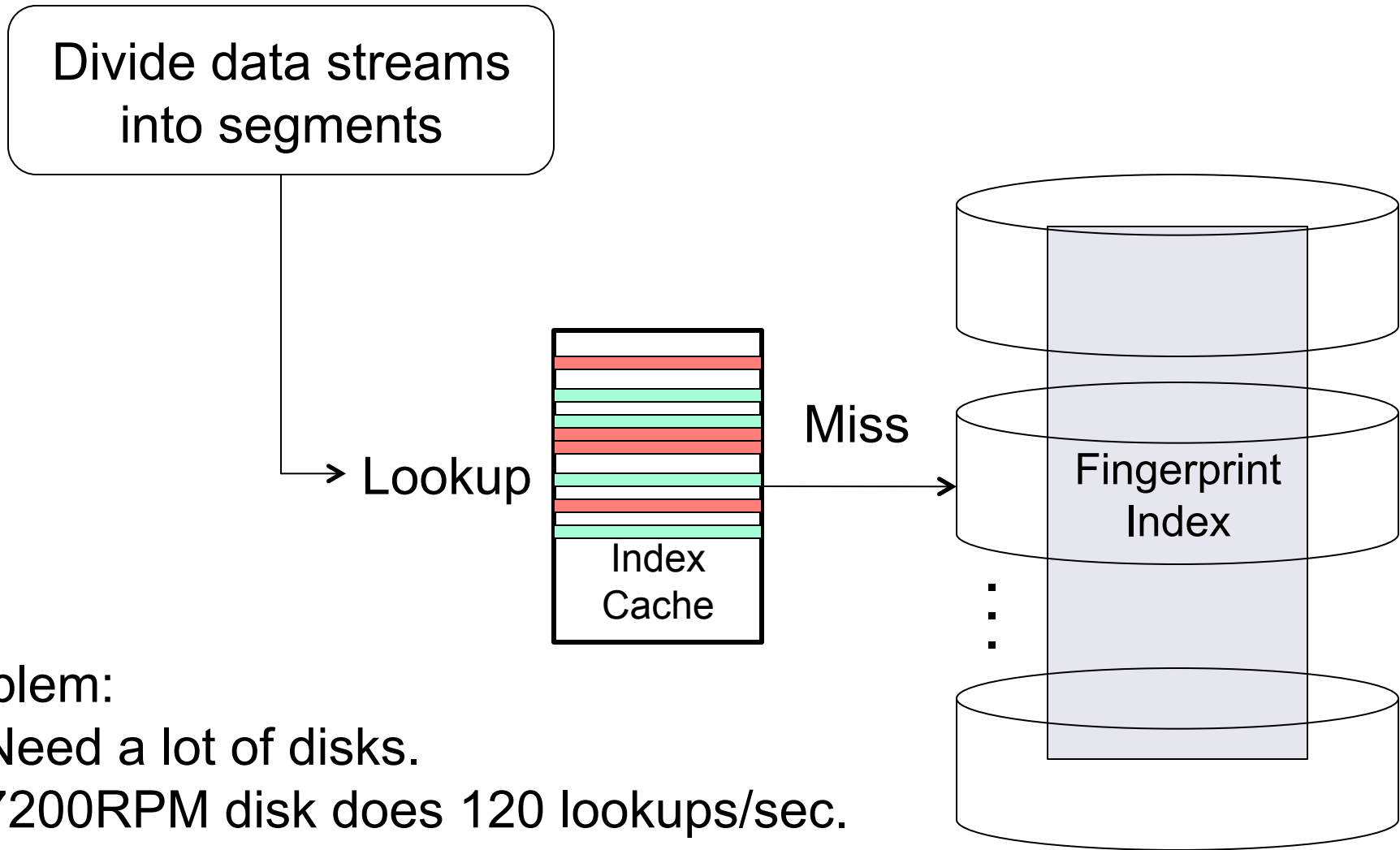
Problematic Alternative 1: Caching



Problem: **No locality.**



Problematic Alternative 2: Parallel Index [Venti02]



Problem:

Need a lot of disks.

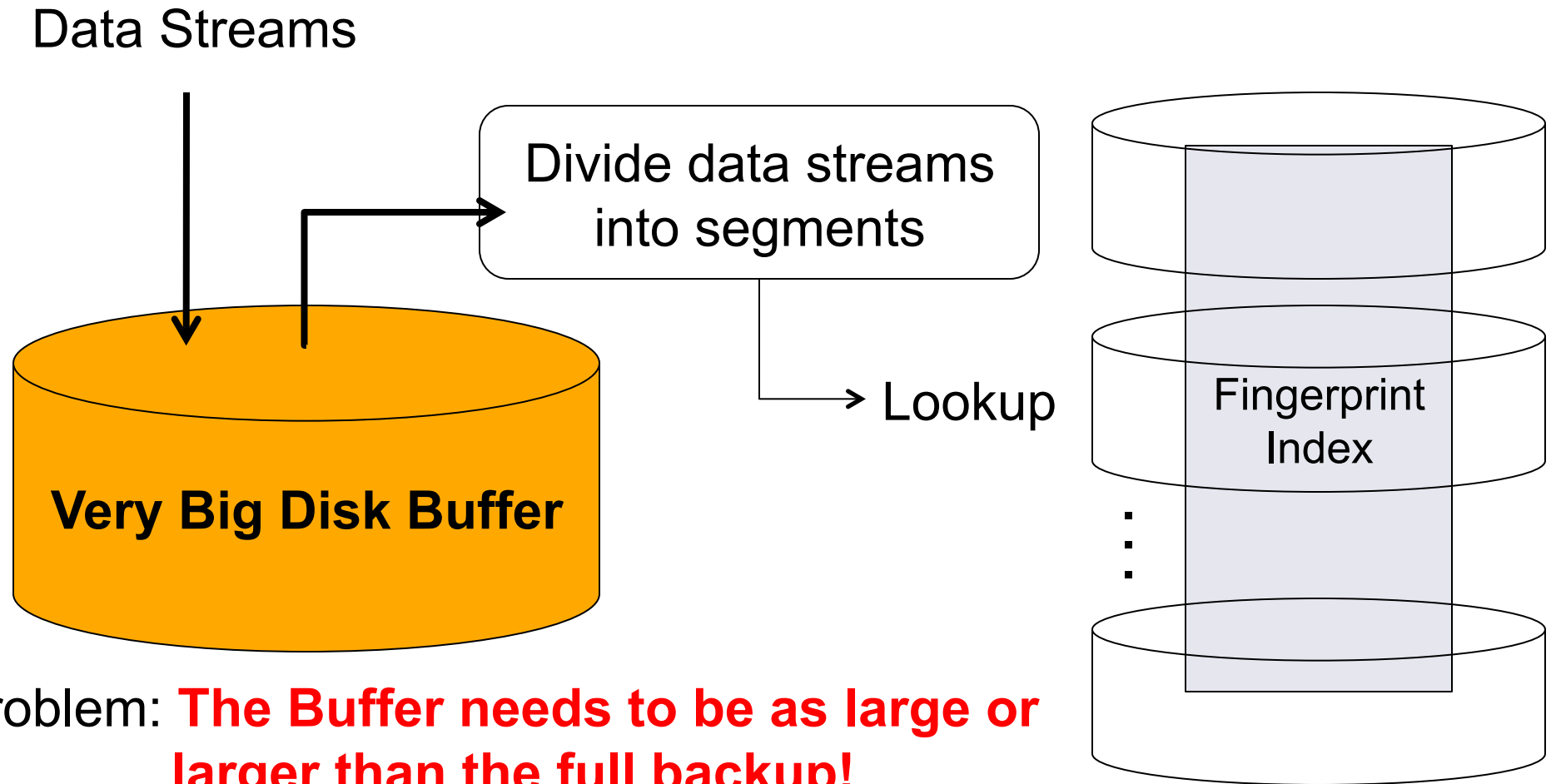
7200RPM disk does 120 lookups/sec.

1MB/sec with 8KB segment per disk

1GB/sec needs 1,000 disks!



Problematic Alternative 3: Staging



Problem: **The Buffer needs to be as large or larger than the full backup!**
Big delay and may still never catch up

High-Speed High Compression at Low HW Cost

- ◆ Layout data on disk with “duplicate locality”
- ◆ A sophisticated cache for the fingerprint index
 - Summary data structure for new data
 - “locality-preserved caching” for old data
- ◆ Parallelized deduplication architecture to take full advantage of multicore processors

Benjamin Zhu, Kai Li and Hugo Patterson. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. In Proceedings of The 6th USENIX Conference on File and Storage Technologies (FAST'08). February 2008

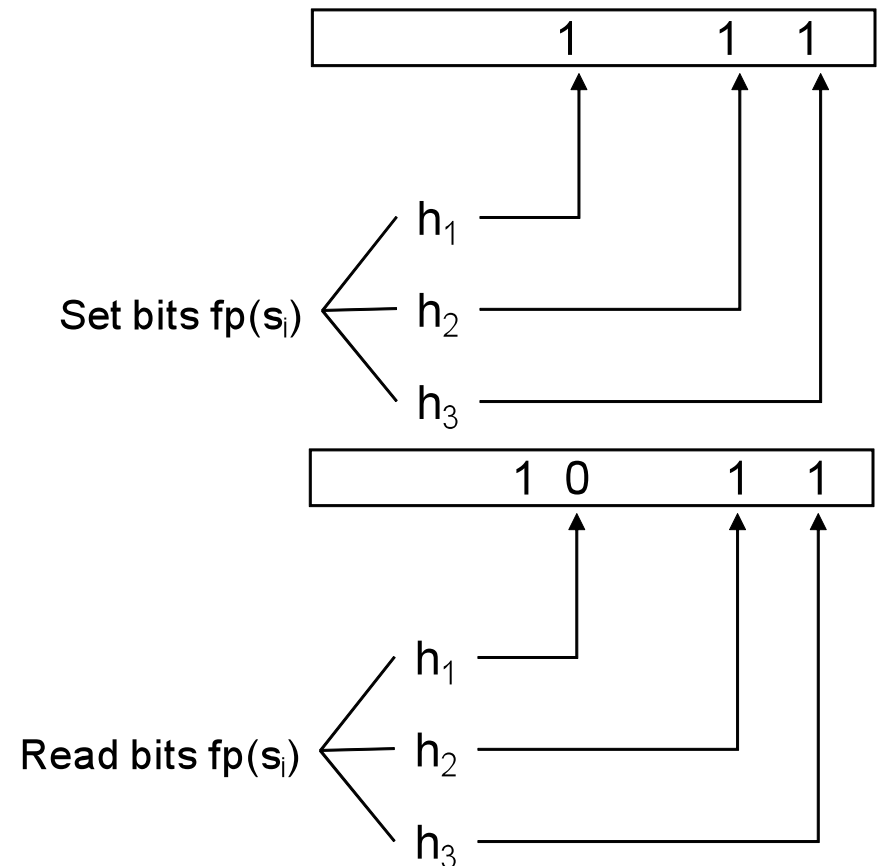
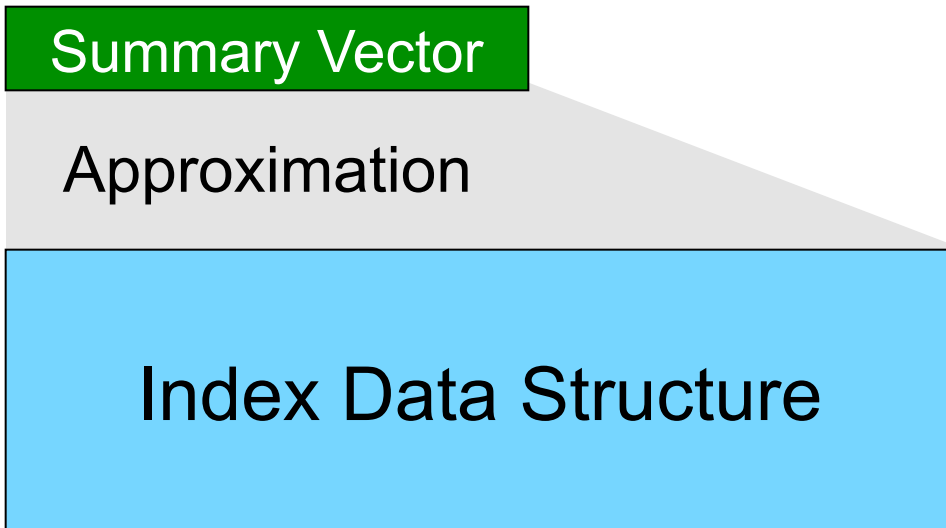


Summary Vector

Goal: Use minimal memory to test for new data

⇒ Summarize what segments have been stored, with Bloom filter (Bloom'70) in RAM

⇒ If Summary Vector says no, it's new segment



Known Analysis Results

- ◆ Bloom filter with m bits k independent hash functions
- ◆ After inserting n keys, the probability of a false positive is:

$$p = \left(1 - \left(1 - \frac{1}{m} \right)^{kn} \right)^k \approx \left(1 - e^{-kn/m} \right)^k$$

- ◆ *Examples:*

- $m/n = 6, k = 4: p = 0.0561$
- $m/n = 8, k = 6: p = 0.0215$
- ...

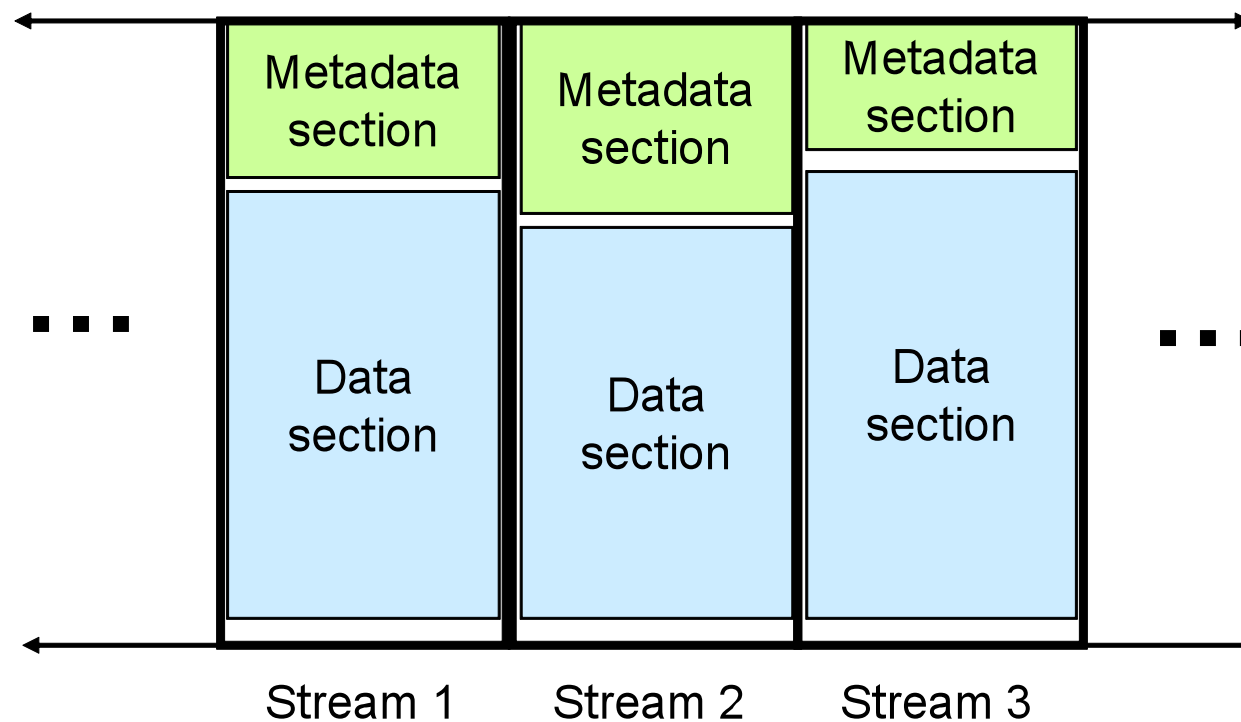
- ◆ Experimental data validate the analysis results



Stream Informed Segment Layout

Goal: Capture “duplicate locality” on disk

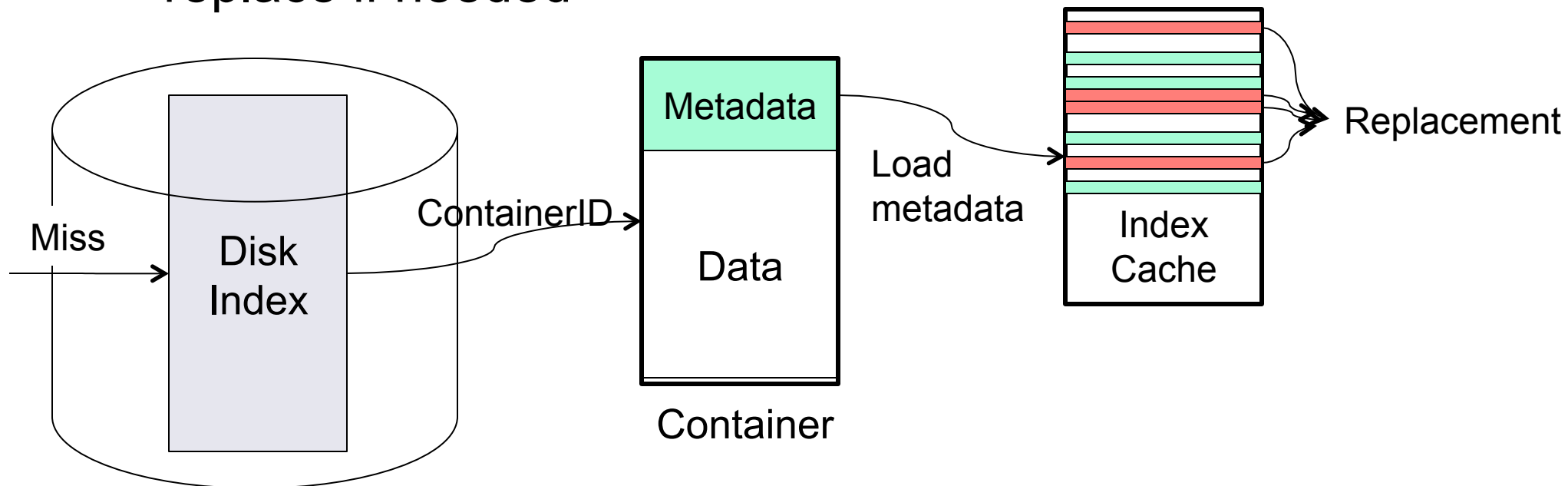
- Segments from the same stream are stored in the same “containers”
- Metadata (index data) are also in the containers



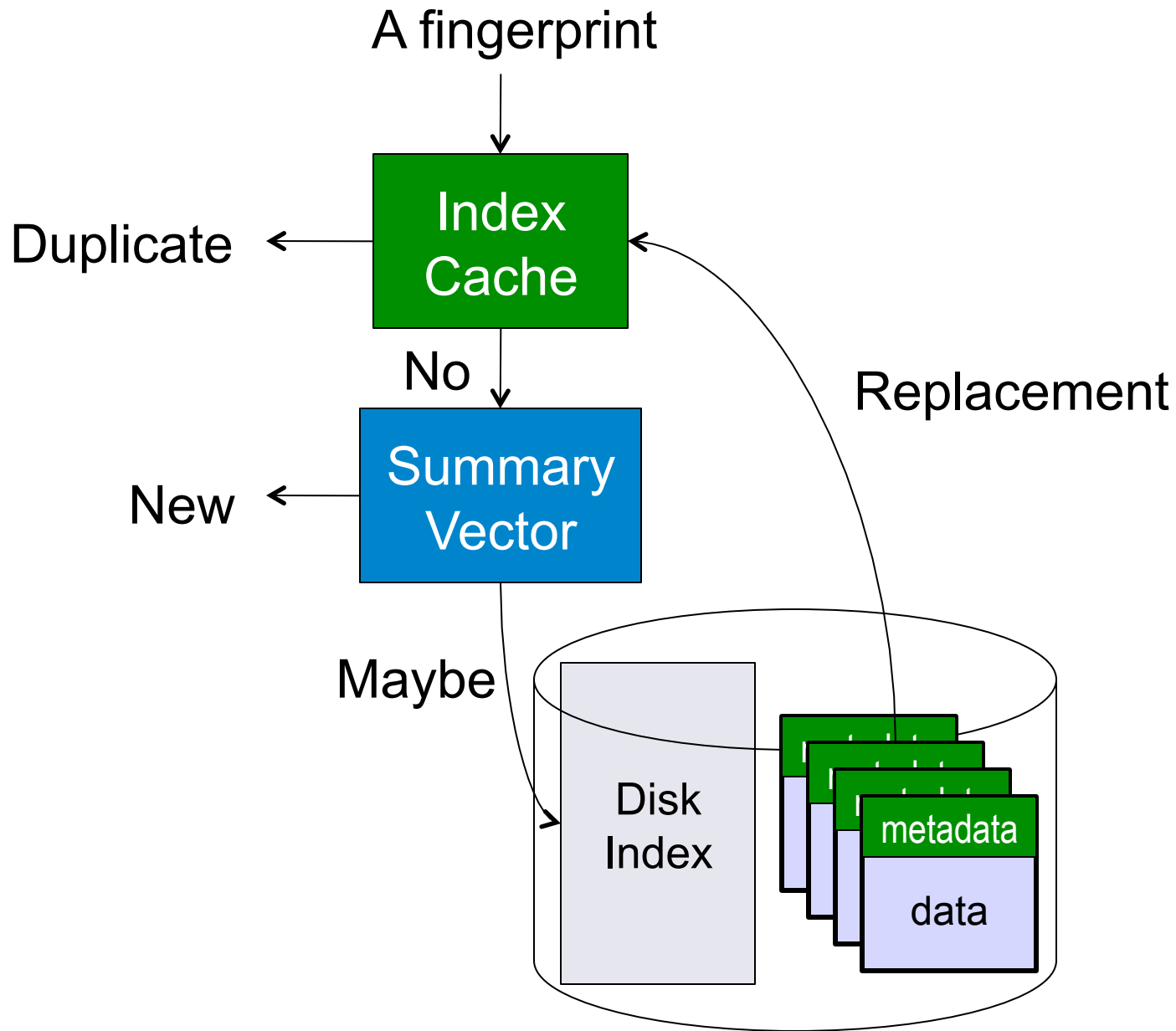
Locality Preserved Caching (LPC)

Goal: Maintain “duplicate locality” in the cache

- Disk Index has all <fingerprint, containerID> pairs
- Index Cache caches a subset of such pairs
- On a miss, lookup Disk Index to find containerID
- Load the metadata of a container into Index Cache, replace if needed



Putting Them Together



Evaluation

- ◆ What to evaluate
 - Disk I/O reduction
 - Write and read throughput
 - Deduplication results
- ◆ Platform (DD880)
 - 4 × Quad 2.9Ghz Xeon CPUs, 32GB RAM, 10GE NIC, 2 x 1GB NVRAM, 96TB 7,200 RPM ATA disks

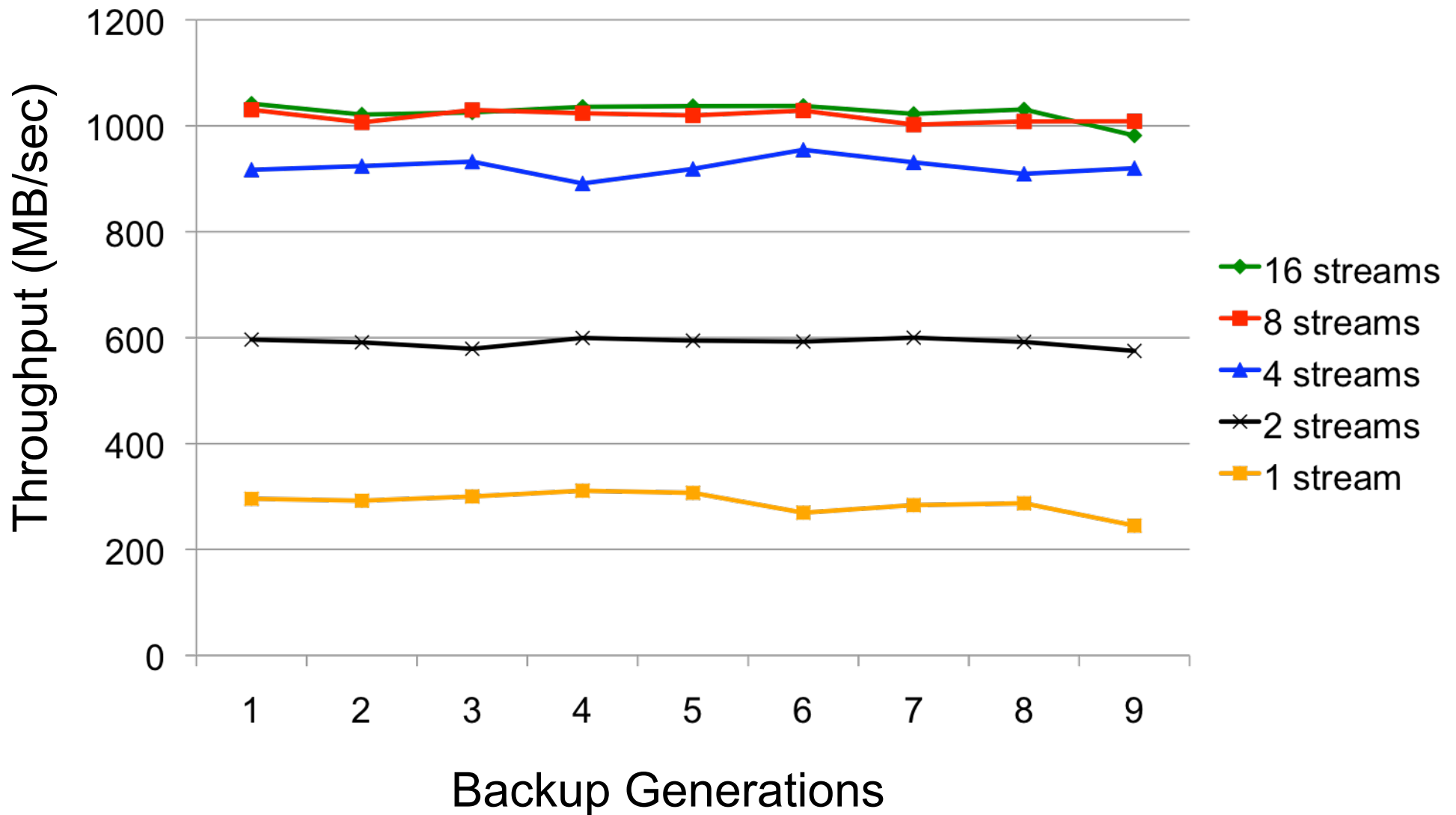


Disk I/O Reduction Results

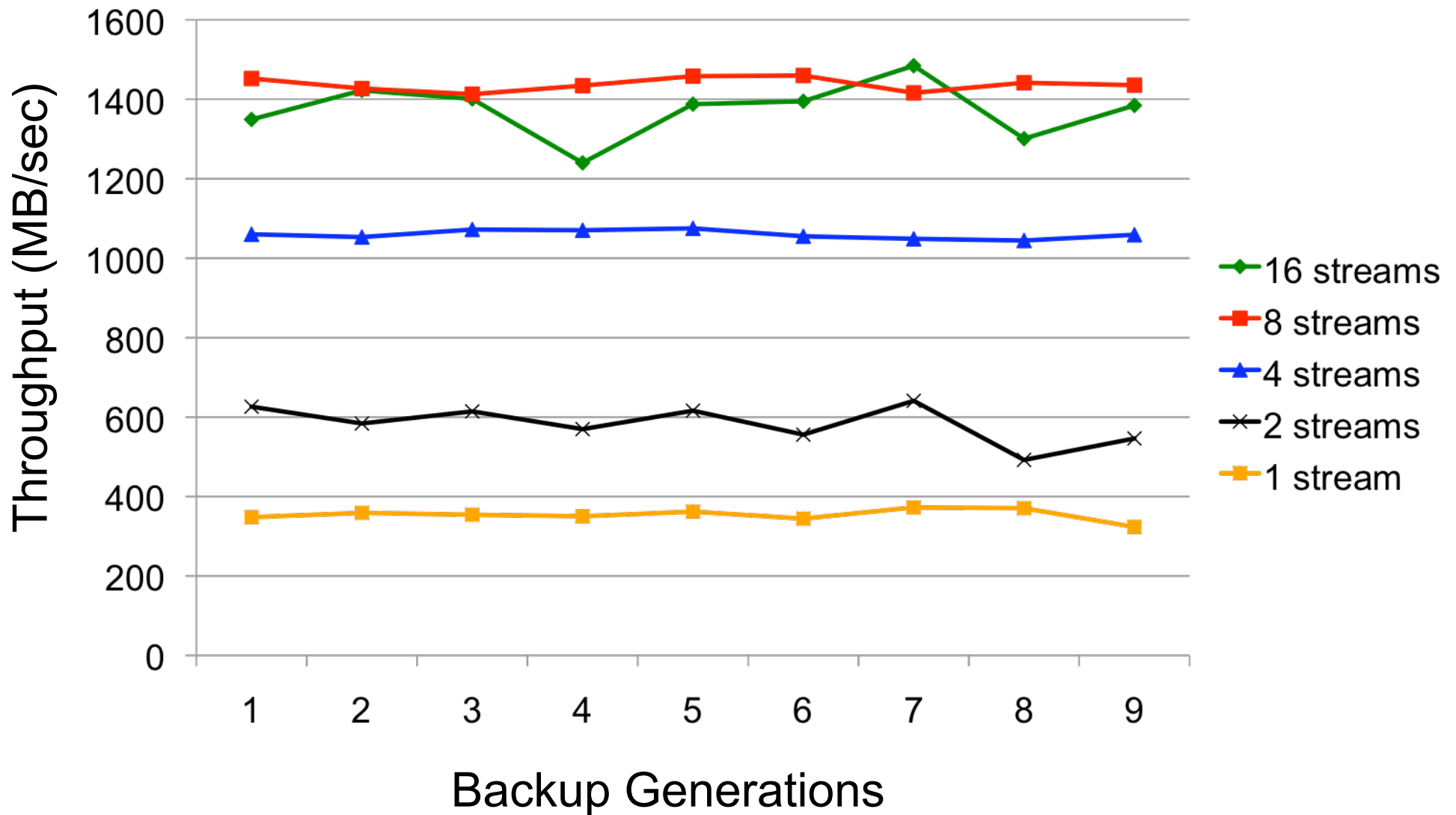
| | Exchange data (2.56TB) <i>135-daily full backups</i> | | Engineering data (2.39TB) <i>100-day daily inc, weekly full</i> | |
|-----------------------------------|---|--------------|--|--------------|
| | # disk I/Os | % of total | # disk I/Os | % of total |
| No summary, No SISL/LPC | 328,613,503 | 100.00% | 318,236,712 | 100.00% |
| Summary only | 274,364,788 | 83.49% | 259,135,171 | 81.43% |
| SISL/LPC only | 57,725,844 | 17.57% | 60,358,875 | 18.97% |
| Summary & SISL/LPC | 3,477,129 | 1.06% | 1,257,316 | 0.40% |



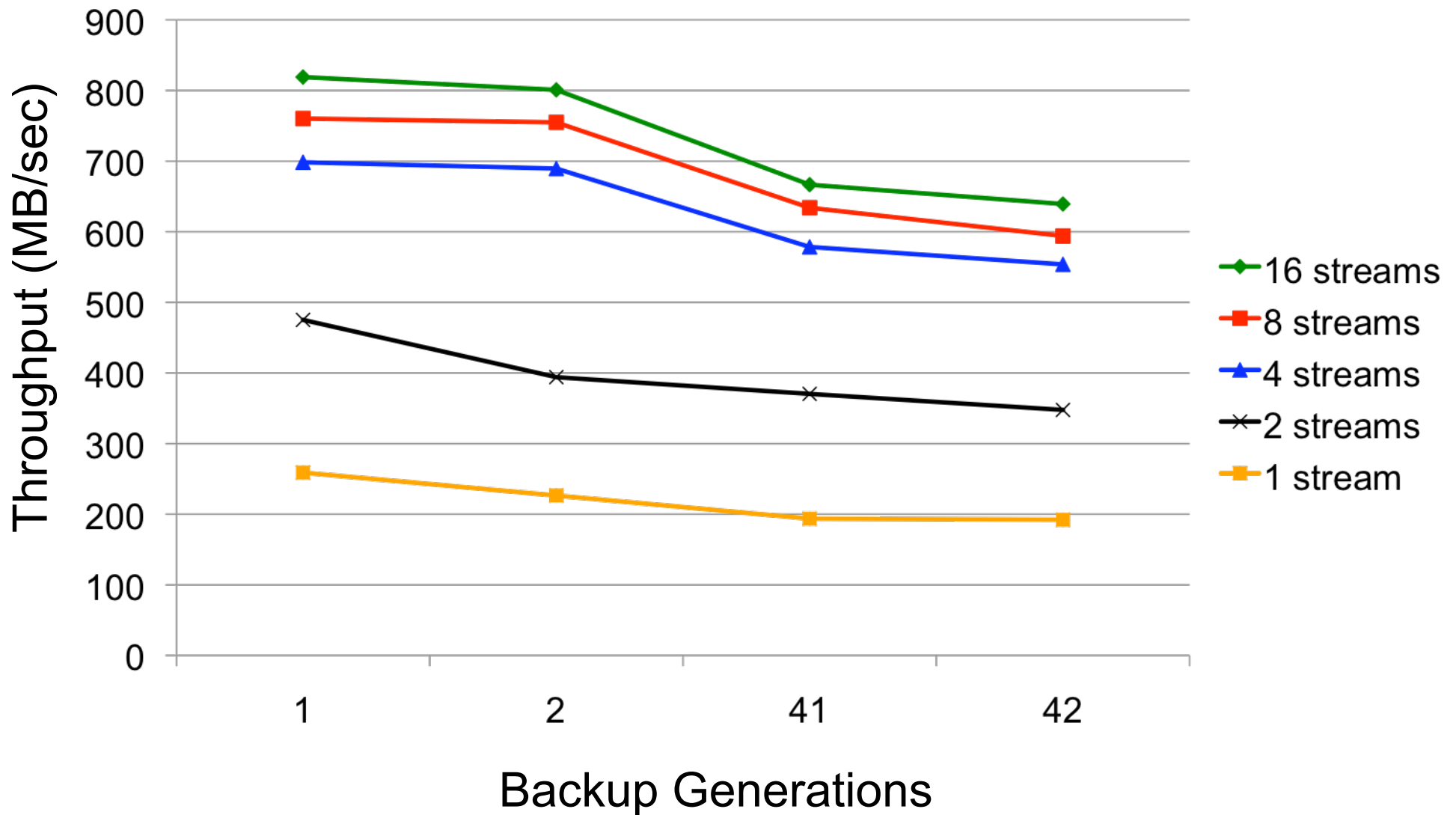
NFS Deduplication Write



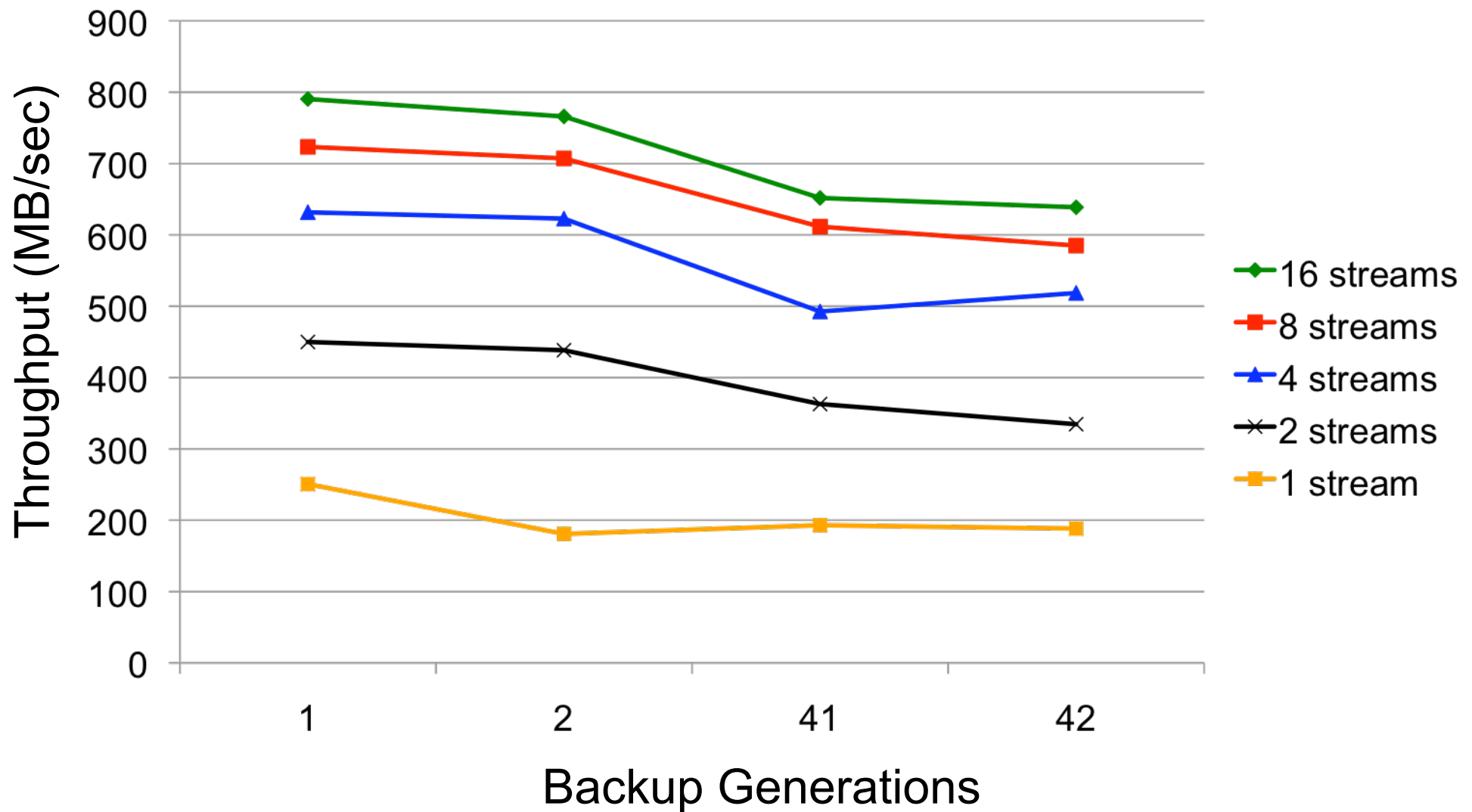
NetBackup OST Deduplication Write



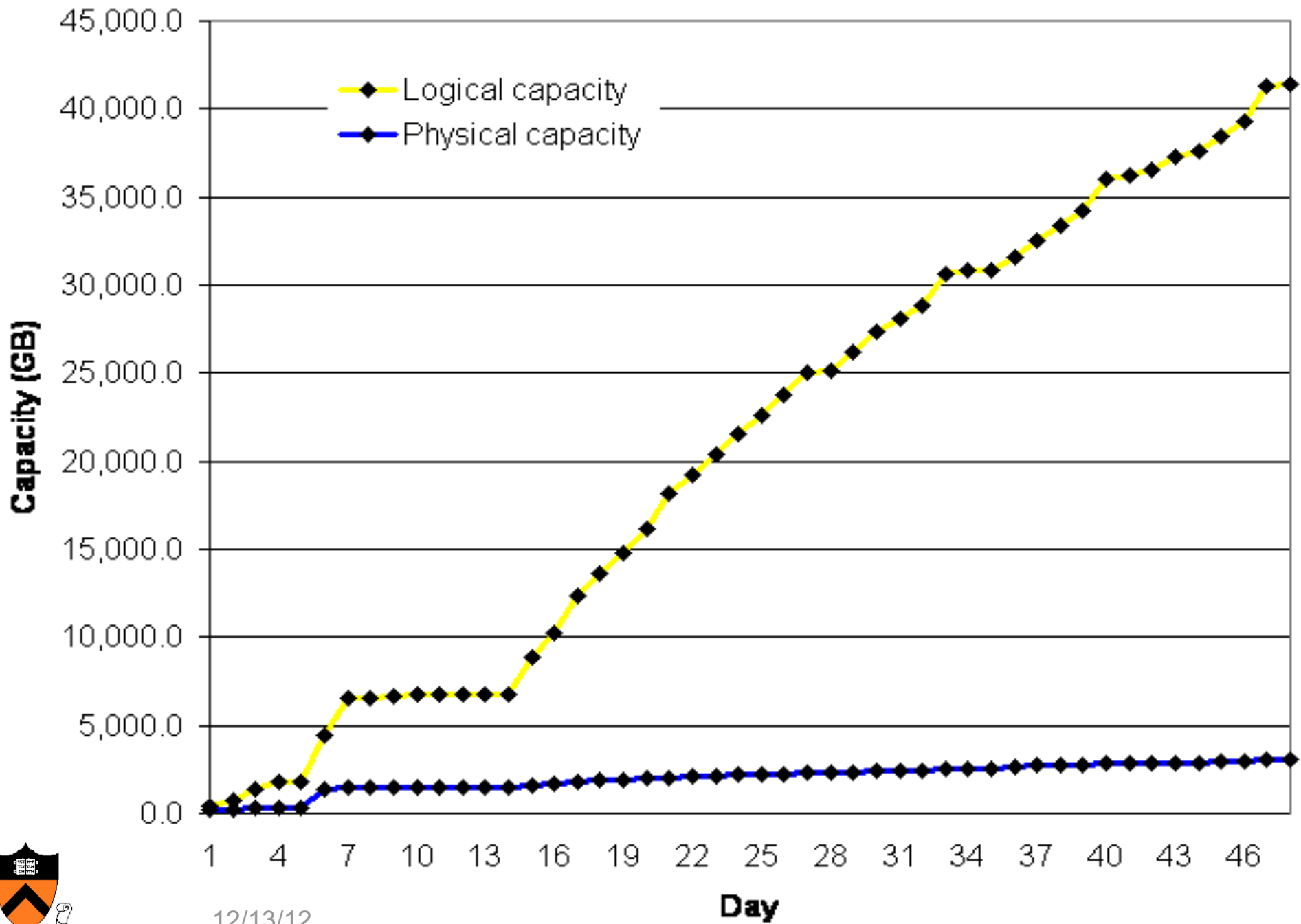
NFS Deduplication Read



NetBackup OST Deduplication Read

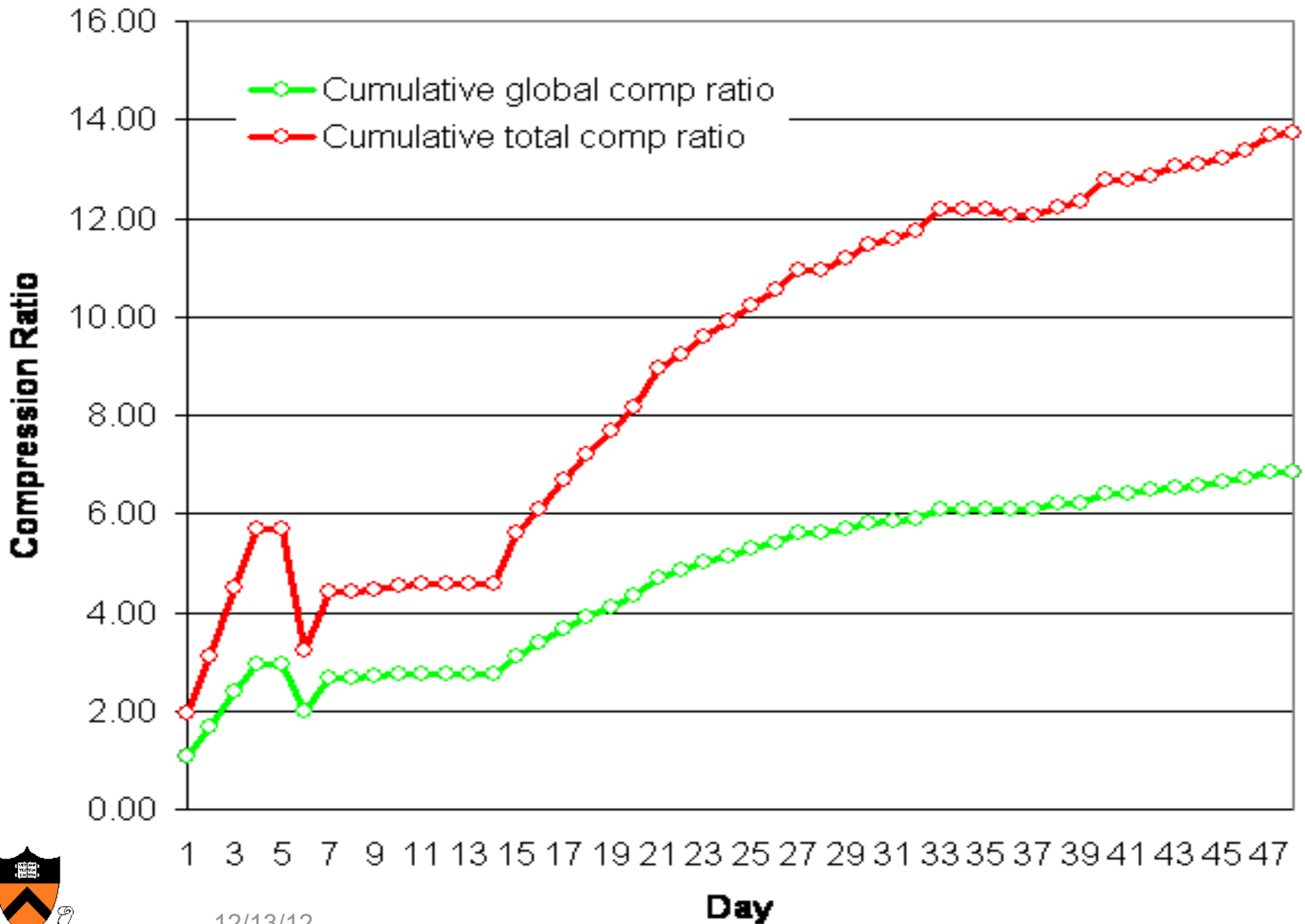


Real World Example at Datacenter A



12/13/12

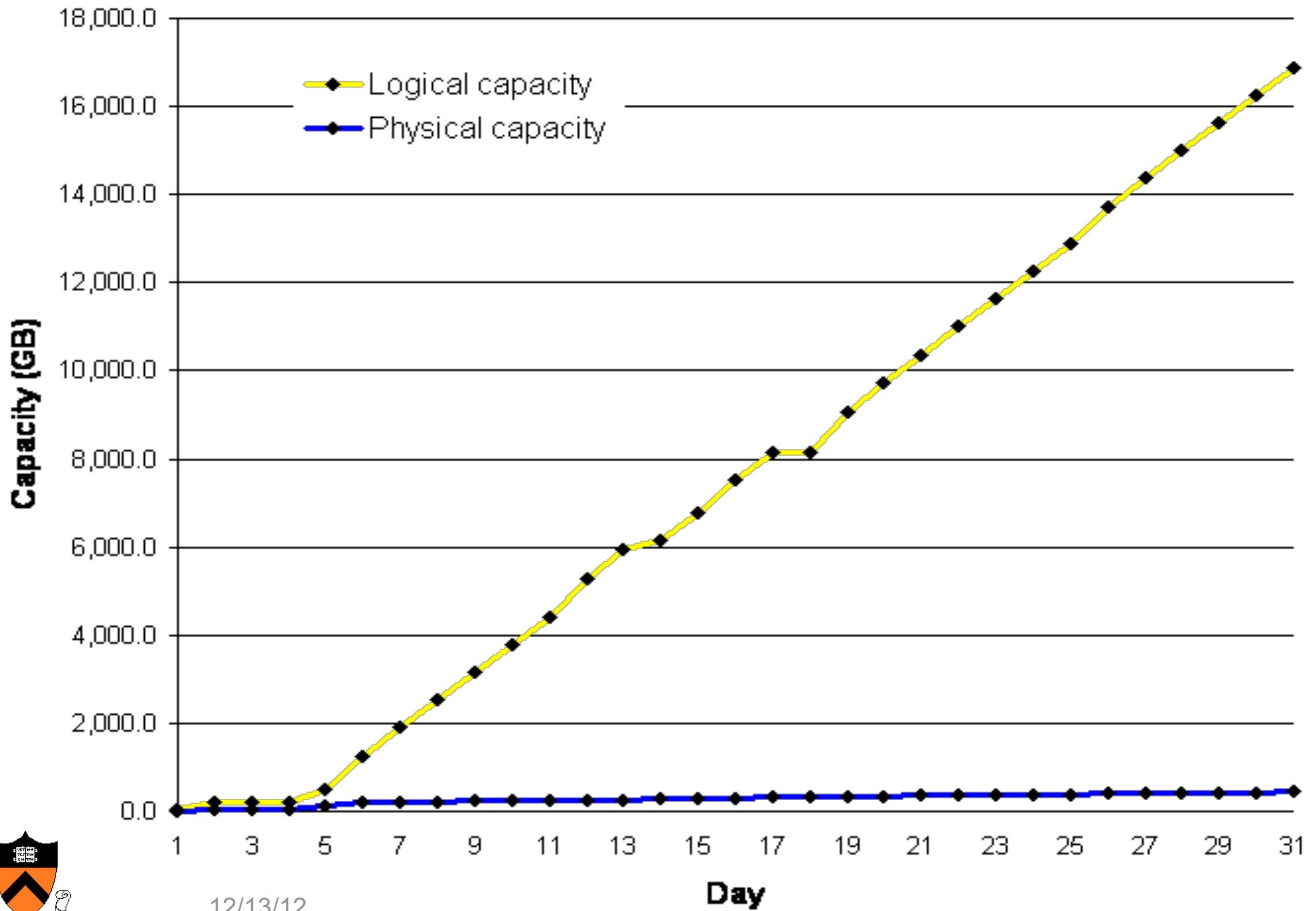
Real World Compression at Datacenter A



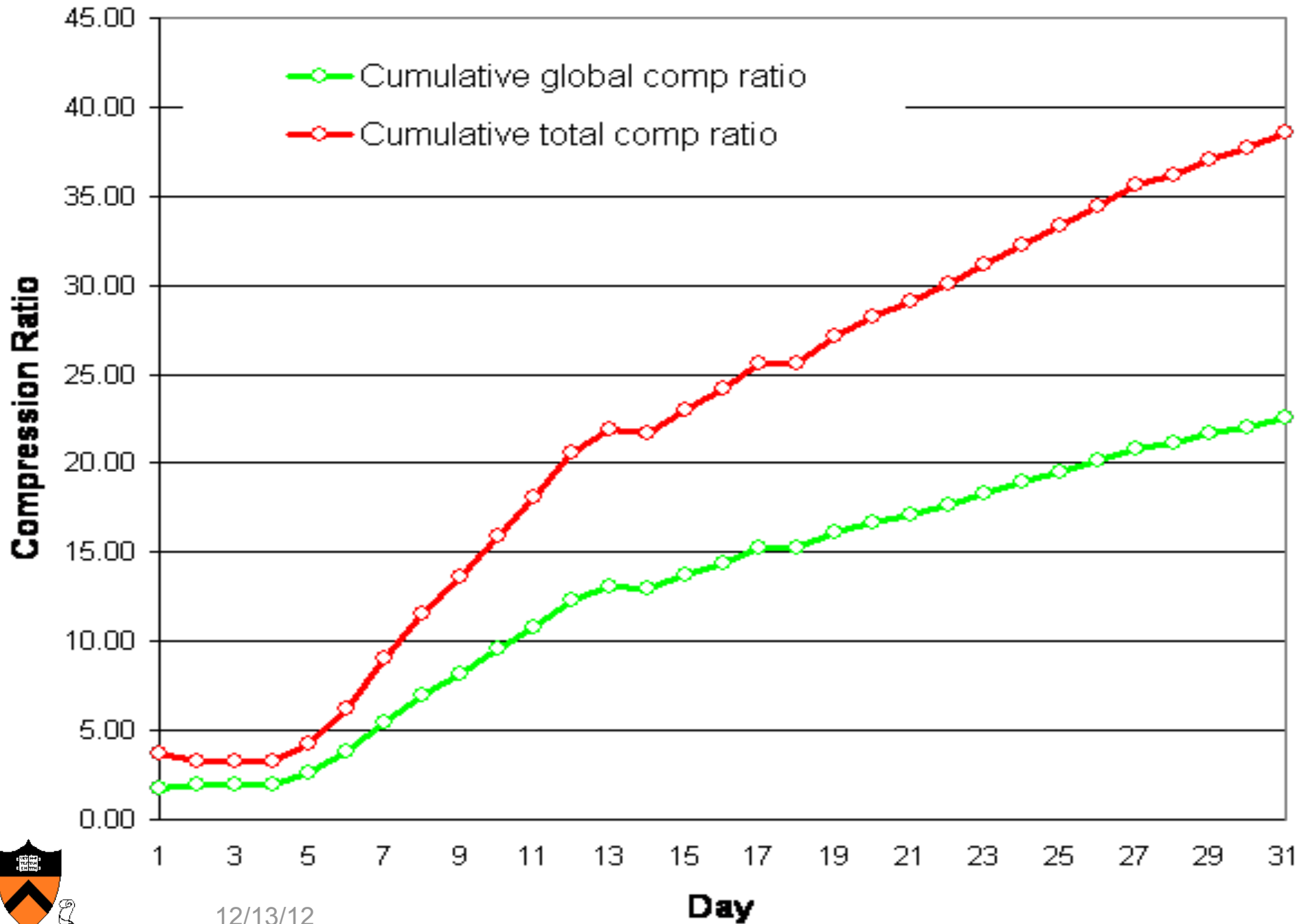
12/13/12



Real World Example at Datacenter B



Real World Compression at Datacenter B



Summary

- ◆ Deduplication removes redundant data globally
- ◆ Advanced deduplication file system
 - Has become a de facto standard to store highly redundant data because of reduction in cost, performance, power, space, ...
 - Scalable performance with multicore CPUs
- ◆ Use cases
 - Backup, nearline, archival and flash



Review Topics

- ◆ OS structure
- ◆ Process management
- ◆ CPU scheduling
- ◆ I/O devices
- ◆ Virtual memory
- ◆ Disks and file systems
- ◆ General concepts



Operating System Structure

- ◆ Abstraction
- ◆ Protection and security
- ◆ Kernel structure
 - Layered
 - Monolithic
 - Micro-kernel
- ◆ Virtualization
 - Virtual machine monitor



Process Management

- ◆ Implementation
 - State, creation, context switch
 - Threads and processes
- ◆ Synchronization
 - Race conditions and inconsistencies
 - Mutual exclusion and critical sections
 - Semaphores: P() and V()
 - Atomic operations: interrupt disable, test-and-set.
 - Monitors and Condition Variables
 - Mesa-style monitor
- Deadlocks
 - How deadlocks occur?
 - How to prevent deadlocks?



CPU Scheduling

- ◆ Allocation
 - Non-preemptible resources
- ◆ Scheduling -- Preemptible resources
 - FIFO
 - Round-robin
 - STCF
 - Lottery



I/O Devices

- ◆ Latency and bandwidth
- ◆ Interrupts and exceptions
- ◆ DMA mechanisms
- ◆ Synchronous I/O operations
- ◆ Asynchronous I/O operations
- ◆ Message passing



Virtual Memory

◆ Mechanisms

- Paging
- Segmentation
- Page and segmentation
- TLB and its management

◆ Page replacement

- FIFO with second chance
- Working sets
- WSClock



Disks and File Systems

- ◆ Disks
 - Disk behavior and disk scheduling
 - RAID5 and RAID6
- ◆ Flash memory
 - Write performance
 - Wear leveling
 - Flash translation layer
- ◆ Directories and implementation
- ◆ File layout
- ◆ Buffer cache
- ◆ Transaction and its implementation
- ◆ NFS and Stateless file system
- ◆ Snapshot
- ◆ Deduplication file system



Implementation

- ◆ BeginTransaction
 - Start using a “write-ahead” log on disk
 - Log all updates
- ◆ Commit
 - Write “commit” at the end of the log
 - Then “write-behind” to disk by writing updates to disk
 - Clear the log
- ◆ Rollback
 - Clear the log
- ◆ Crash recovery
 - If there is no “commit” in the log, do nothing
 - If there is “commit,” replay the log and clear the log
- ◆ Assumptions
 - Writing to disk is correct (recall the error detection and correction)
 - Disk is in a good state before we start



An Example: Atomic Money Transfer

- ◆ Move \$100 from account S to C (1 thread):

BeginTransaction

$S = S - \$100;$

$C = C + \$100;$

Commit

- ◆ Steps:

- 1: Write new value of S to log
- 2: Write new value of C to log
- 3: Write commit
- 4: Write S to disk
- 5: Write C to disk
- 6: Clear the log

- ◆ Possible crashes

- After 1
- After 2
- After 3 before 4 and 5

- ◆ Questions

- Can we swap 3 with 4?
- Can we swap 4 and 5?

C = 110
S = 700

C = 110
S = 700

S=700 C=110 Commit



Questions

- ◆ Do the following transactions behave the same?

BeginTransaction

$C = C + \$100;$

$S = S - \$100;$

Commit

BeginTransaction

$S = S - \$100;$

$C = C + \$100;$

Commit

- ◆ Yes, this is why transactions are good
 - Flushing buffer cache without worrying about the order of writes
 - Group transactions in data base systems
 - Many more convenient programming



Major Concepts

- ◆ Locality
 - Spatial and temporal locality
- ◆ Scheduling
 - Use the past to predict the future
- ◆ Layered abstractions
 - Synchronization, transactions, file systems, etc
- ◆ Caching
 - TLB, VM, buffer cache, etc



Operating System as Illusionist

Physical reality

- ◆ Single CPU
- ◆ Interrupts
- ◆ Limited memory
- ◆ No protection
- ◆ Raw storage device

Abstraction

- ◆ Infinite number of CPUs
- ◆ Cooperating sequential threads
- ◆ Unlimited virtual memory
- ◆ Each address has its own machine
- ◆ Organized and reliable storage system



Future Courses

◆ Spring

- COS 461: computer networks
- COS 598C: Analytics and systems of big data

◆ Fall:

- COS 432: computer security
- COS 561: Advance computer networks or (or COS 518: Advanced OS)
- Some grad seminars in systems

