DiskReduce: Making Room for More Data on DISCs

Wittawat Tantisiriroj

Lin Xiao, Bin Fan, and Garth Gibson

PARALLEL DATA LABORATORY

Carnegie Mellon University

GFS/HDFS Triplication

- GFS & HDFS triplicate every data block
  - Triplication: one local + two remote copies
- 200% space overhead to handle node failures
- RAID has been used to handle disk failures
- Why can't we use RAID to handle node failures?
  - Is it too complex?
  - Is it too hard to scale?
  - Can it work with commodity hardware?

RAID5 Across Nodes at Scale

- RAID5 across nodes can be done at scale
  - Panasas does it [Welch08]
- But, error handling is complicated

GFS & HDFS Reconstruction

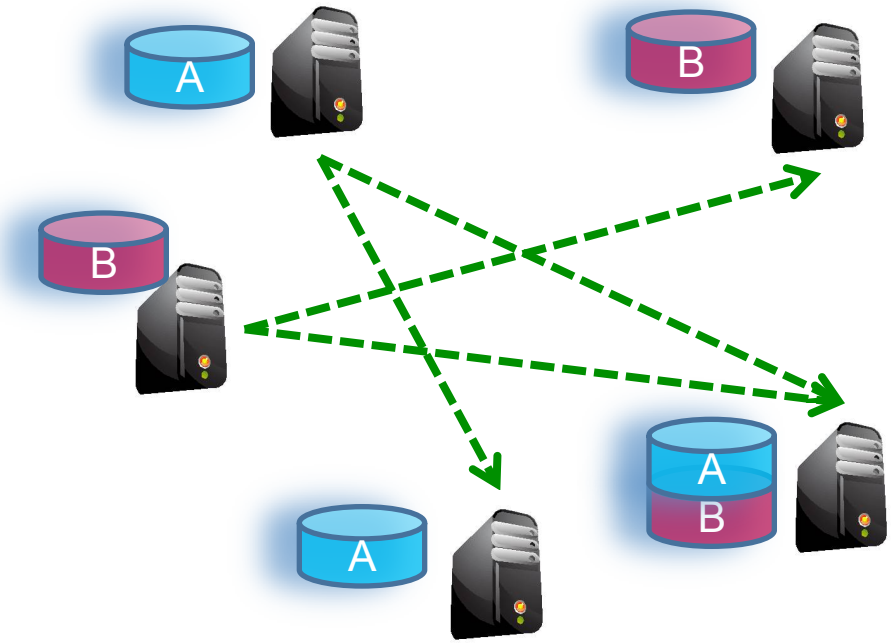
- GFS & HDFS defer repair
  - Background (asynchronous) process repairs copies
    - Notably less scary to developers

Outline

- Motivation
- DiskReduce basic (replace 1 copy with RAID5)
  - Encoding
  - Reconstruction
  - Design options
  - Evaluation
- DiskReduce V2.0
- Conclusion

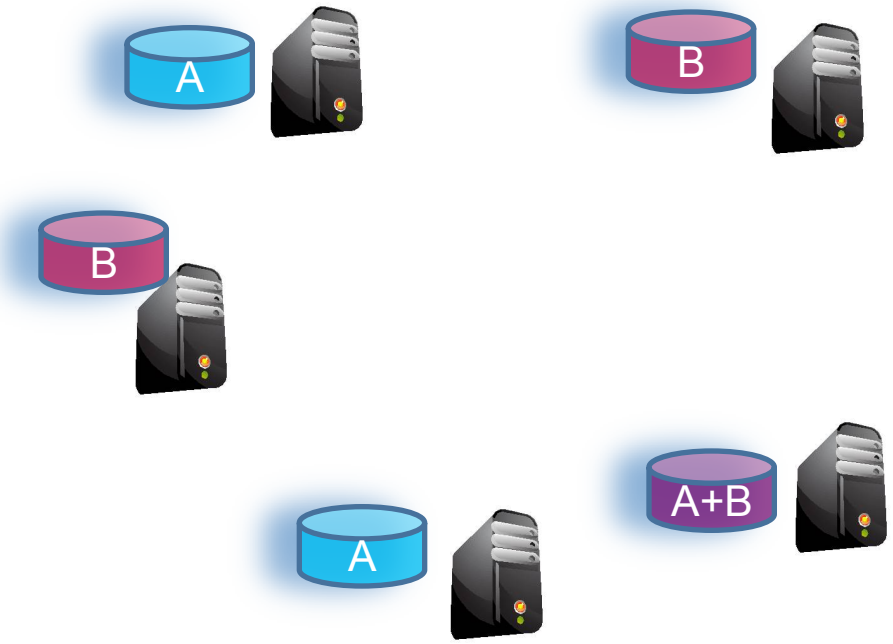
Triplication First

- Start the same: triplicate every data block
  - Triplication: one local + two remote copies
- 200% space overhead



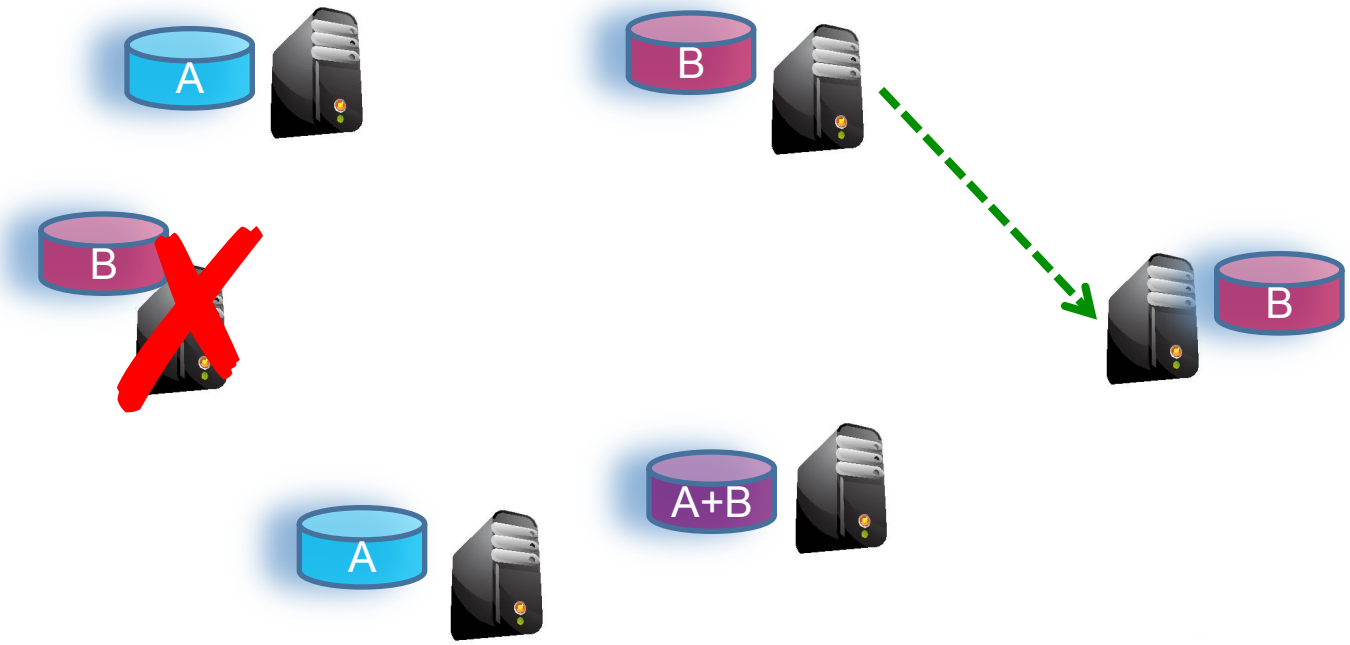
Background Encoding

- Goal: a parity encoding and two copies
- Asynchronous background process to encode
- In coding terms:
  - Data is A, B
  - Check is A, B, f(A,B)=A+B



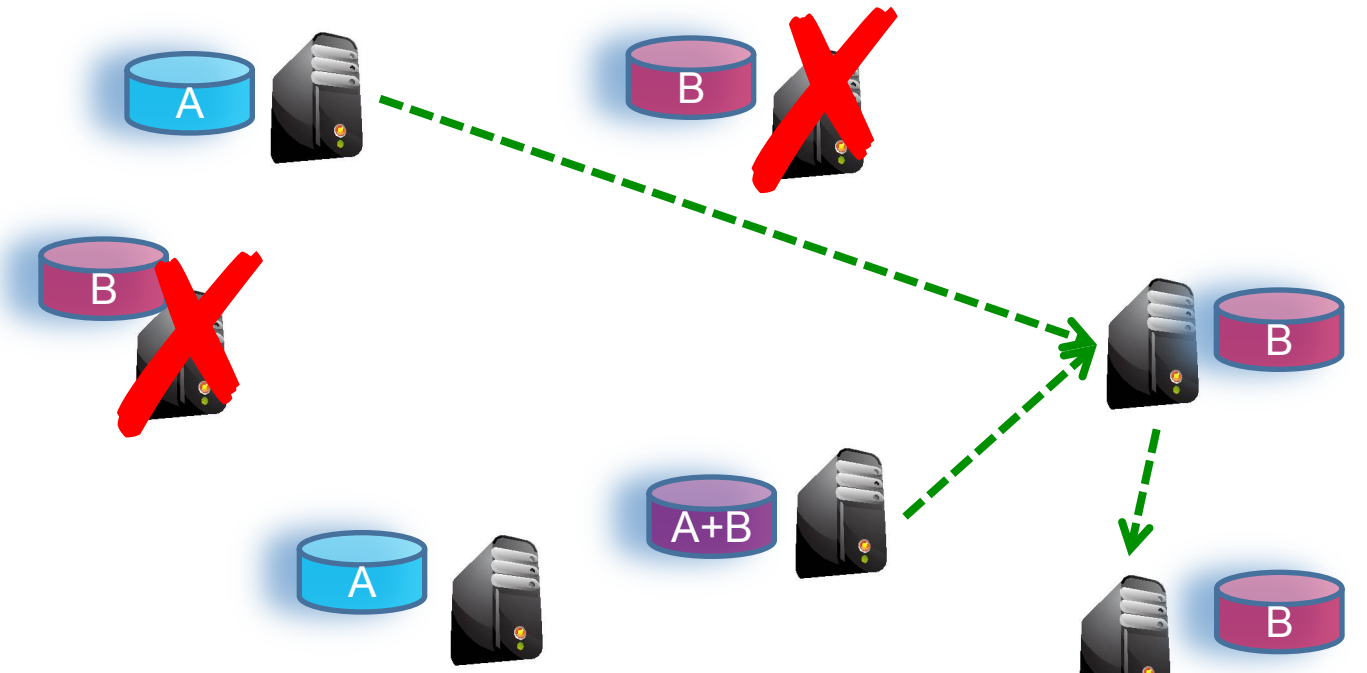
Background Repair (Single Failure)

- Standard single failure recovery
  - Use the 2nd data block to reconstruct



Background Repair (Double Failure)

- Use the parity and other necessary data blocks to reconstruct
- Continue with standard single failure recovery

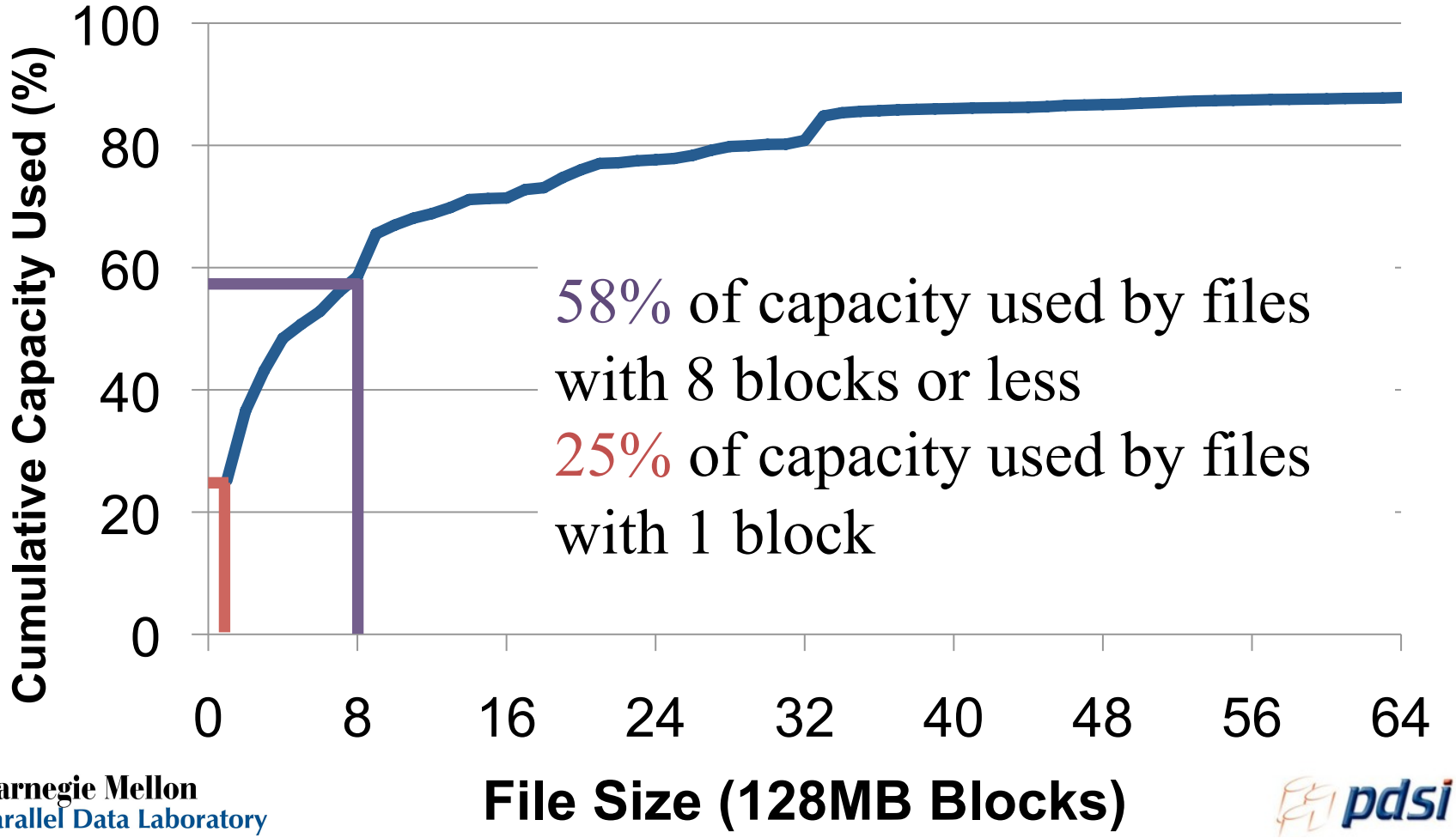


Design Options

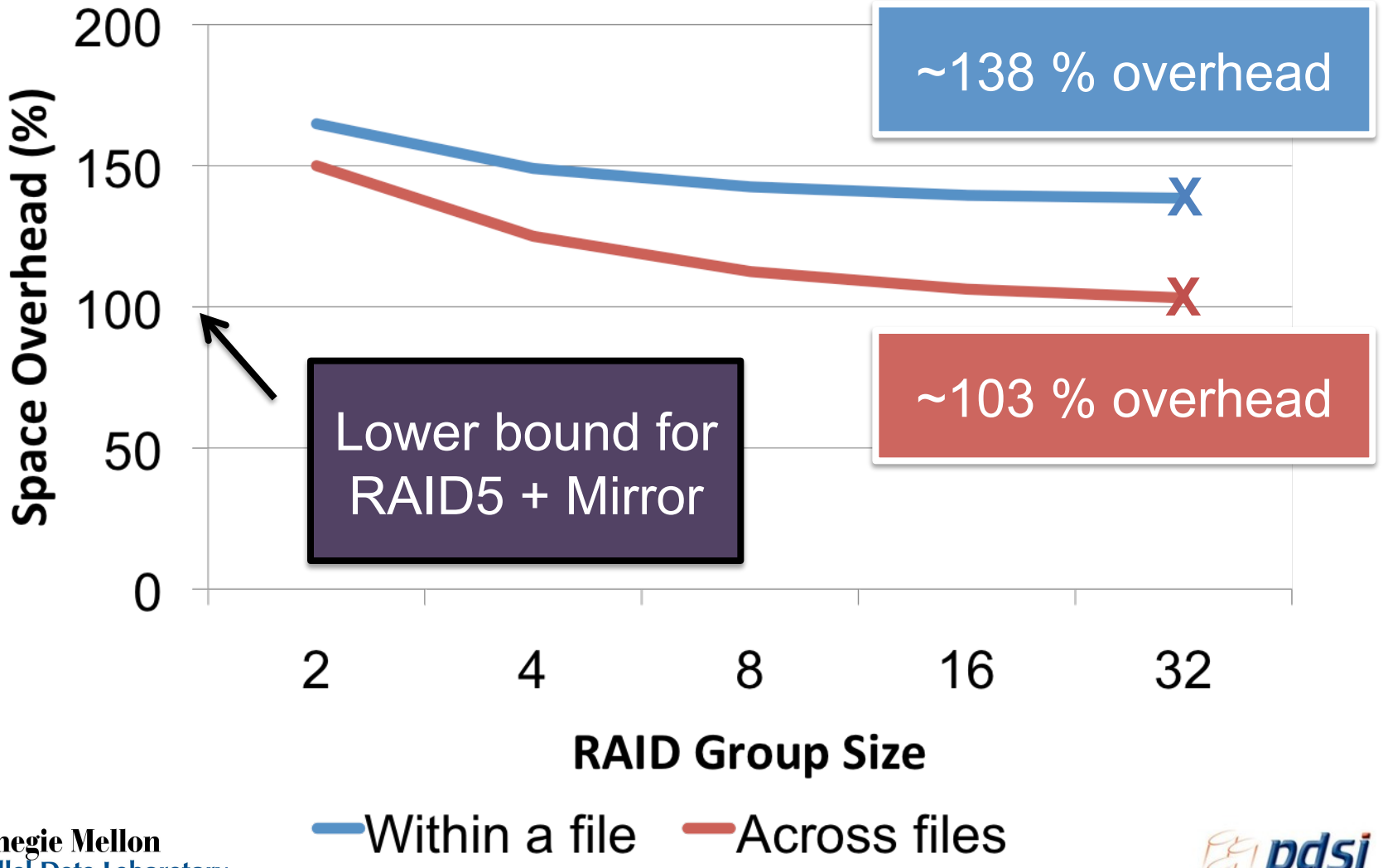
- Encode blocks within a single file
  - Pro: Simple deletion
  - Con: Not space efficient for small files
- Encode blocks across files
  - Pro: More space efficient
  - Con: Need to clean up deletion

Cloud File Size Distribution (Yahoo! M45)

- Large portion of space used by files with a small number of blocks



Across-file RAID Saves More Capacity

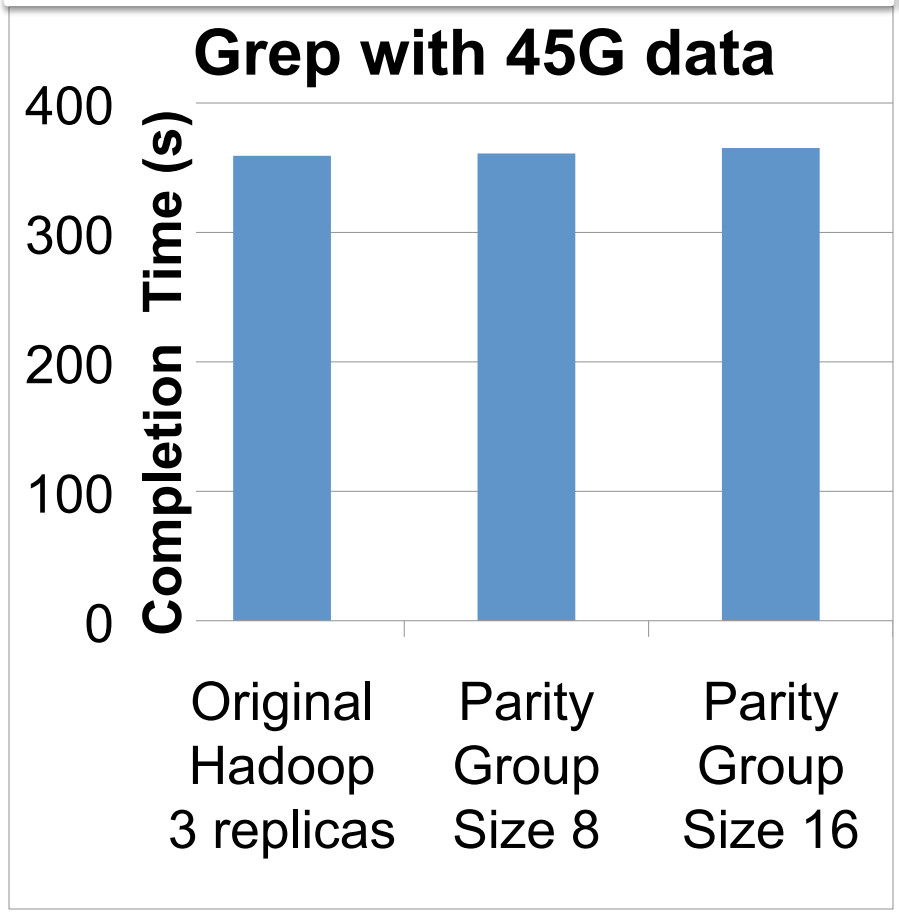


Evaluation

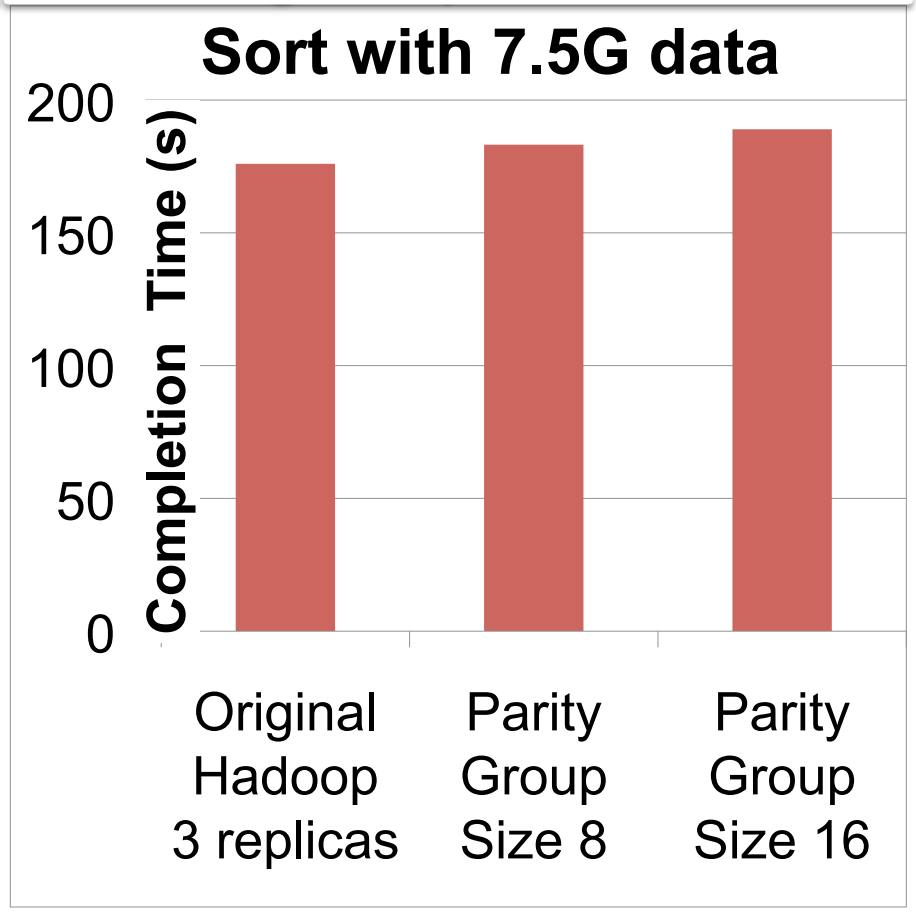
- Testbed
  - 16 nodes, PentiumD dual-core 3.00GHz
  - 4GB memory, 7200 rpm SATA 160GB disk
  - Gigabit Ethernet
- Implementation specification:
  - Hadoop/HDFS version 0.17.1
- Test conditions
  - Benchmarks modeled on Google FS paper
  - Benchmark input after “all parity groups are encoded”
  - Benchmark output has “encoding in background”
  - No failures during tests

As Expected, Little Performance Degradation

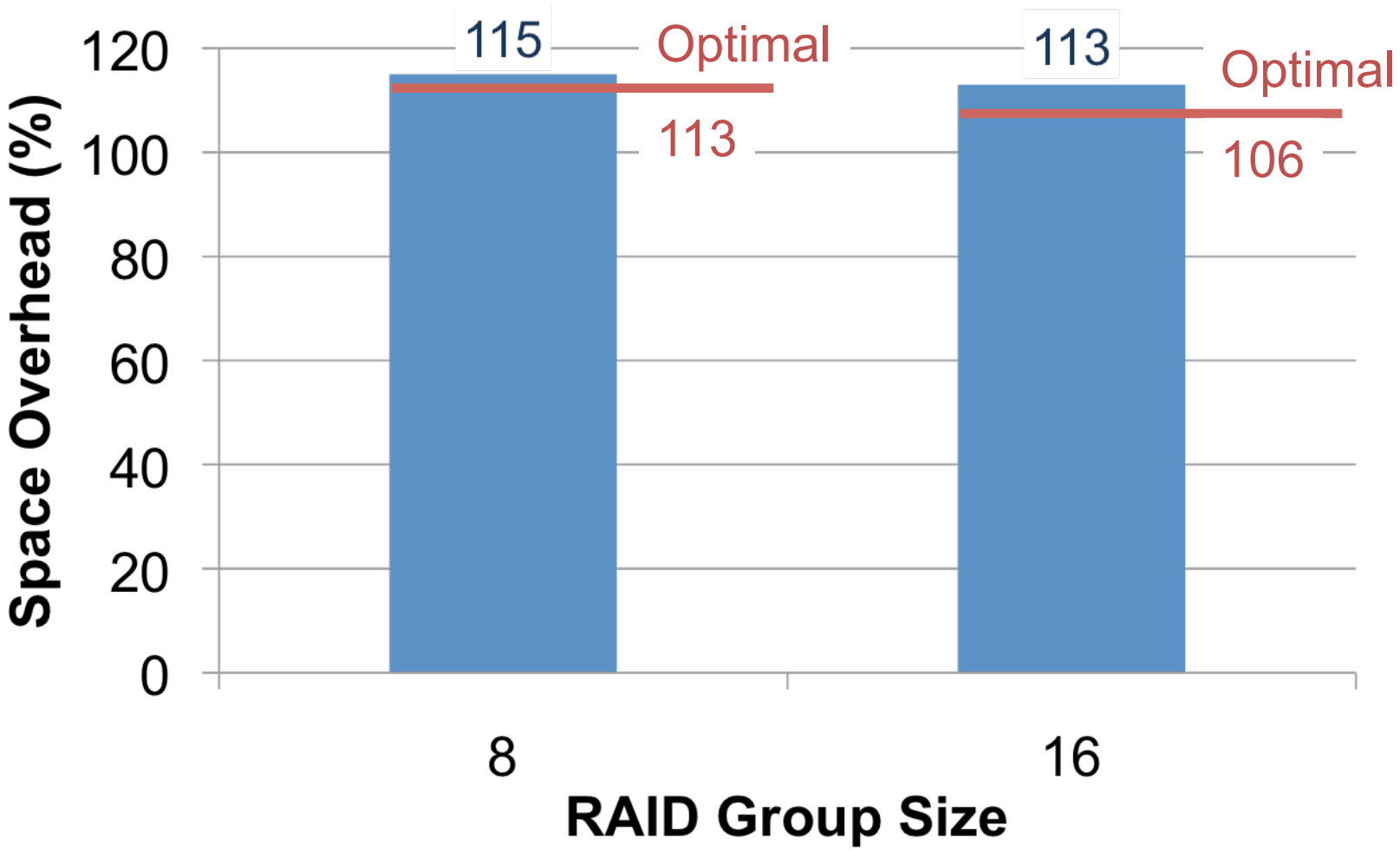
~1 % degradation
No write



~7% degradation
Encoding competes with write



Reduce Overhead to Nearly Optimal



- Optimal only when blocks are perfectly balanced

DiskReduce in the Real World!

- Based on a talk about DiskReduce v1
- An user-level of RAID5 + Mirror in HDFS [Borthakur09]
  - Combine third replica of blocks from a single file to create parity blocks & remove third replica
- Apache JIRA HDFS-503 @ Hadoop 0.22.0

FRIDAY, AUGUST 28, 2009

HDFS and Erasure Codes (HDFS-RAID)

The Hadoop Distributed File System has been great in providing a cloud-type file system. It is robust (when administered correctly :-)) and highly scalable. However, one of the main drawbacks of HDFS is that each piece of data is replicated in three places. This is acceptable because disk storage is cheap and is becoming cheaper by the day; this isn't a problem if you have a relatively small to medium size cluster. The price difference (in absolute terms) is not much whether you use 15 disks or whether you use 10 disks. If we consider the cost of $1 per GByte, the price difference between fifteen 1 TB disk and ten 1 TB disk is only $5K. But when the total size of your cluster is 10 PBytes, then the costs savings in storing the data in two places versus three is a huge ten million dollars!

FACEBOOK BADGE

Dhruba Borthakur

facebook

Name:
Dhruba Borthakur

Outline

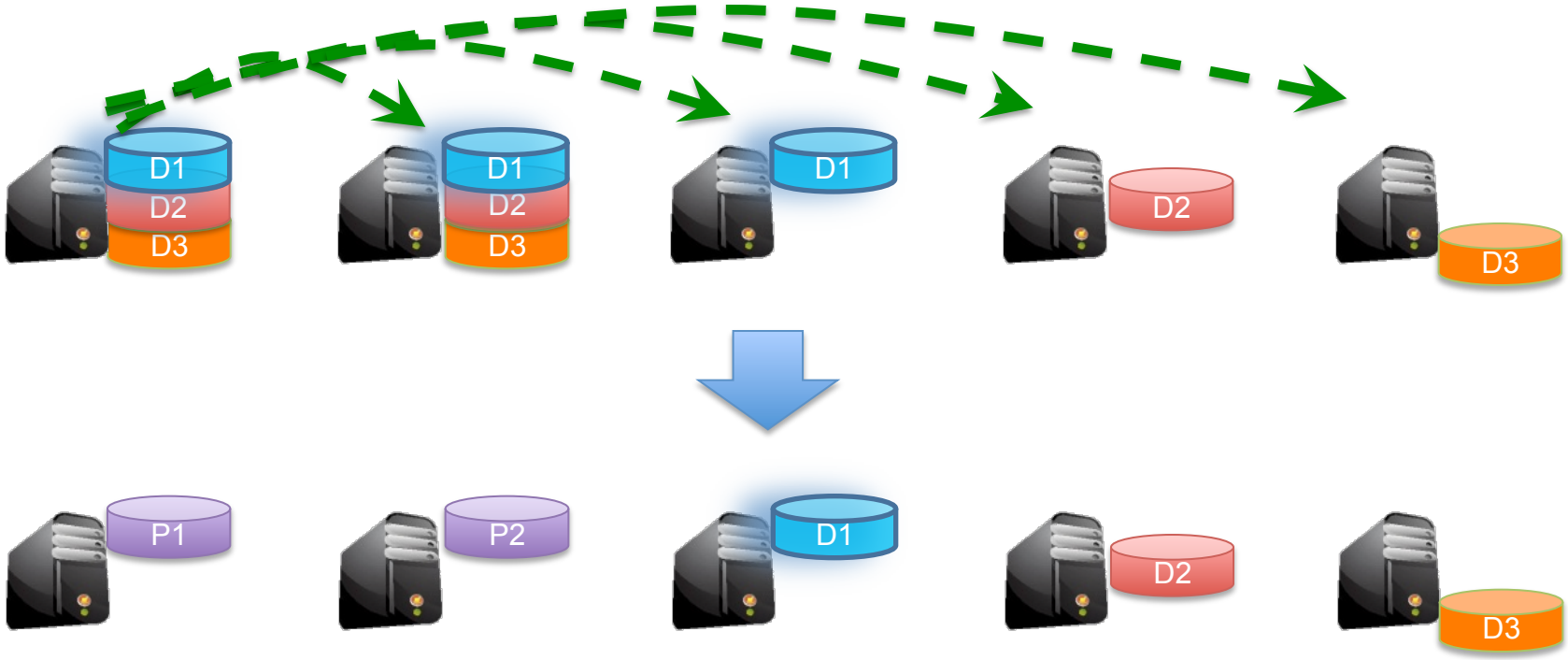
- Motivation
- DiskReduce Basic (Apply RAID to HDFS)
- DiskReduce V2.0
  - Goal
  - Delayed Encoding
- Conclusion

Why DiskReduce V2.0?

- Goal: Save more space with stronger codes
- Challenge
  - Simple search used in DiskReduce V1.0 to find feasible groups cannot be applied for stronger codes
- Solution
  - Pre-determine placement of blocks

Example: Block Placement

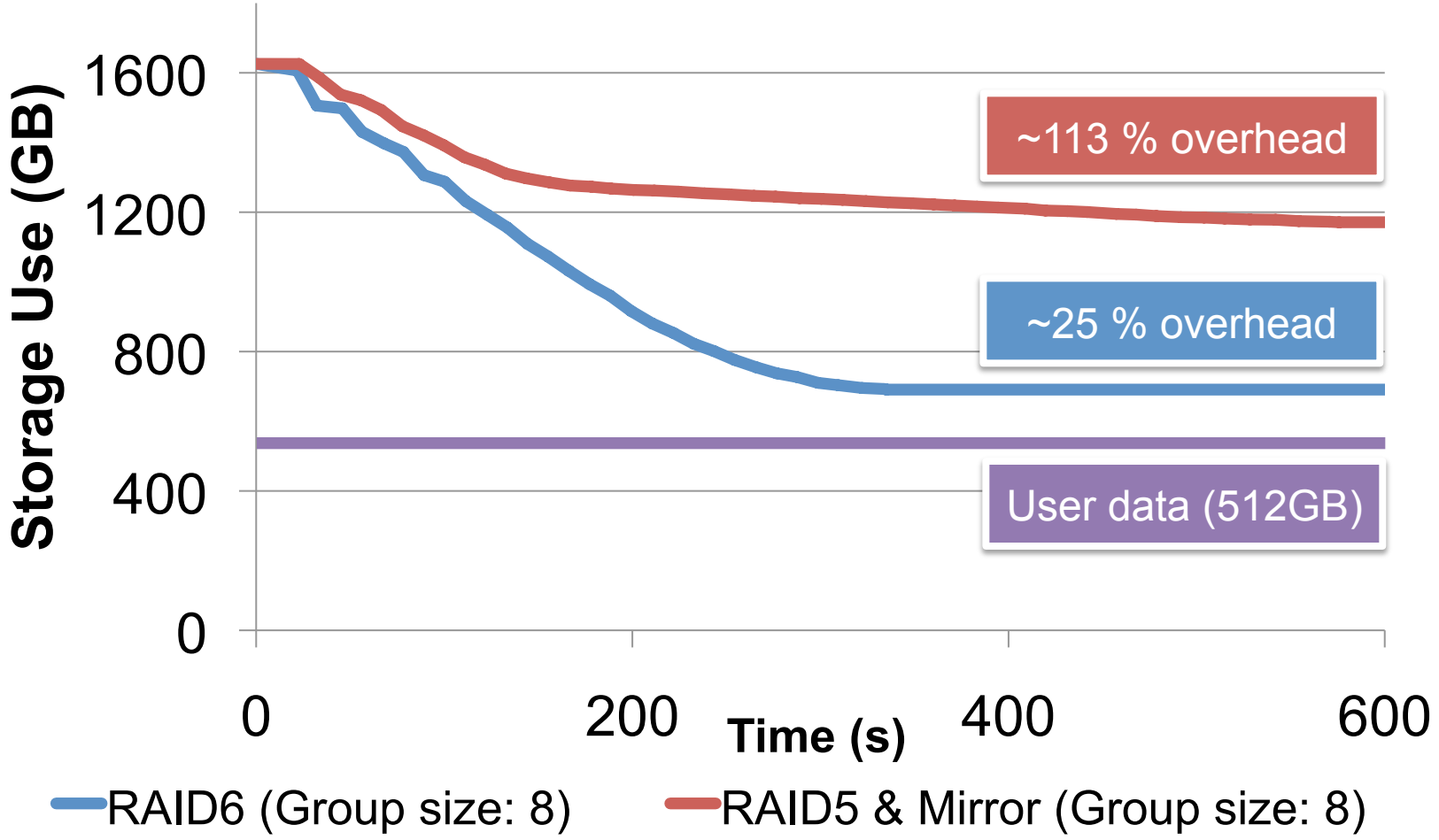
- Codeword drawn from any erasure code
- All data in codeword created at one node
  - Pick up codeword randomly



Prototype Evaluation

- Testbed
  - 32 nodes, two quad-core 2.83GHz Xeon
  - 16GB memory, 4 x 7200 rpm SATA 1TB disk
  - 10 Gigabit Ethernet
- Implementation:
  - Hadoop/HDFS version 0.20.0
  - Encoding part is implemented
  - Other parts are work-in-progress
- Test
  - Each node write a file of 16 GB into a DiskReduce modified HDFS
  - 512GB of user data in total

DiskReduce v2.0 Prototype Works!



- Both schemes can achieve close to optimal

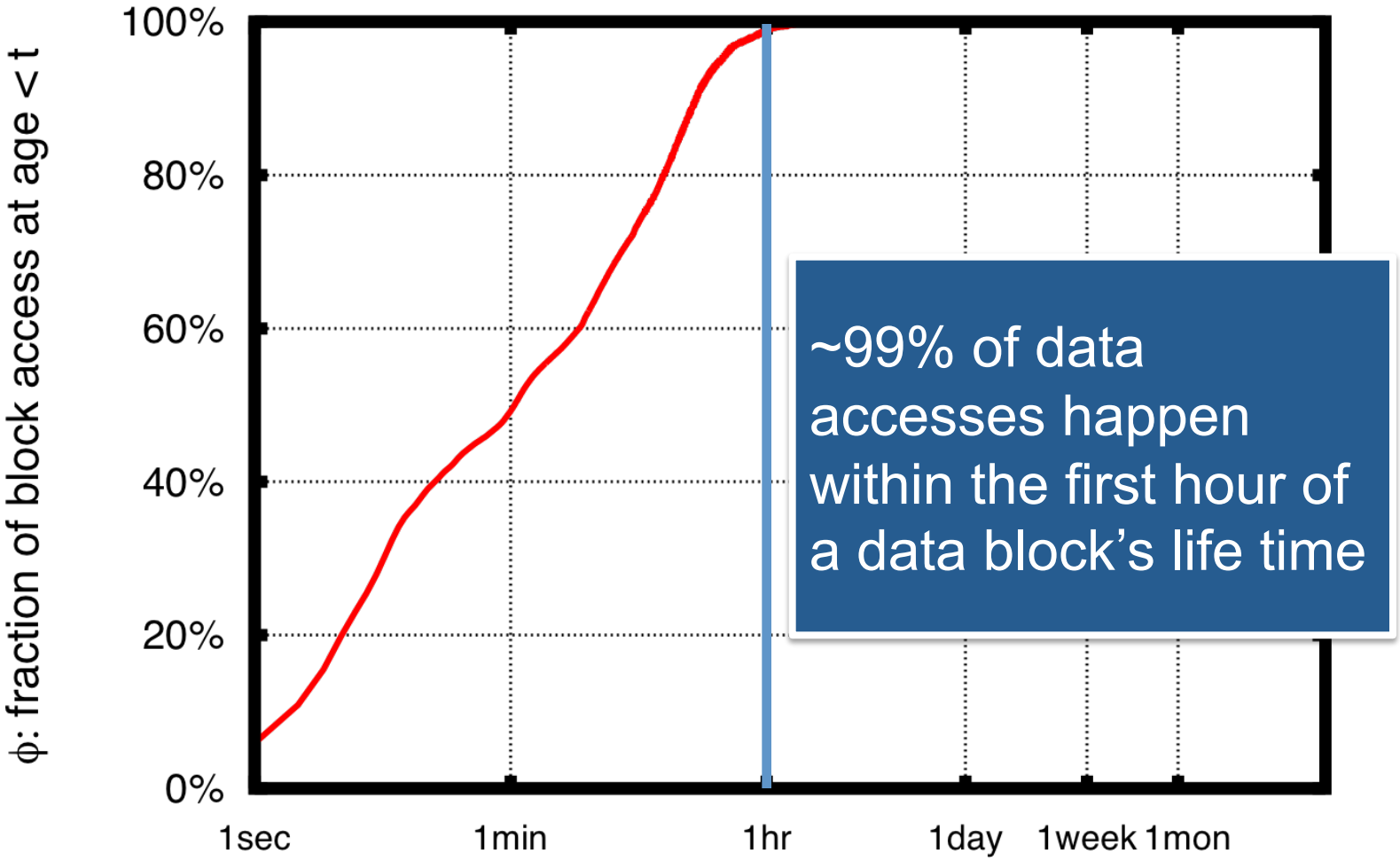
Possible Performance Degradation

- When does more than one copy help?
  - Backup tasks
    - More data copies may help schedule the backup tasks on a node where it has a local copy
  - Hot files
    - Popular files may be read by many jobs at the same time
  - Load balance and local assignment
    - With more data copies, the job tracker has more flexibility to assign tasks to nodes with a local data copy

Delayed Encoding

- Encode blocks when extra copies are likely to yield only small benefit
  - For example, only blocks that have been created for at least one day can be encoded
- How long should we delay?

Age of Block Accesses Distribution (Yahoo! M45)



How Long Should We Delay?

- Fixed delay (ex. 1 hour)
  - Benefit
    - ~99% of data accesses get benefits from multiple copies
  - Cost
    - For a workload of continuous writing
      - 25MB/s per disk
      - ~90GB/hour per disk
      - < 5% of each disks' capacity (a 2TB disk)

Conclusions and Future Work

- RAID can be applied to HDFS
  - Dhruba Borthakur of Facebook has implemented a variant of RAID5 + Mirror in HDFS
- RAID can bring overhead down from 200% to 25%
- Delayed encoding helps avoid performance degradation
- We are currently working on...
  - Reduce clean up work for deletion
  - Analyze additional traces