
# DiskReduce v2.0 for HDFS

## Open Cirrus Summit, Jan 28, 2010

Garth Gibson
Carnegie Mellon University and Panasas Inc
garth@cs.cmu.edu

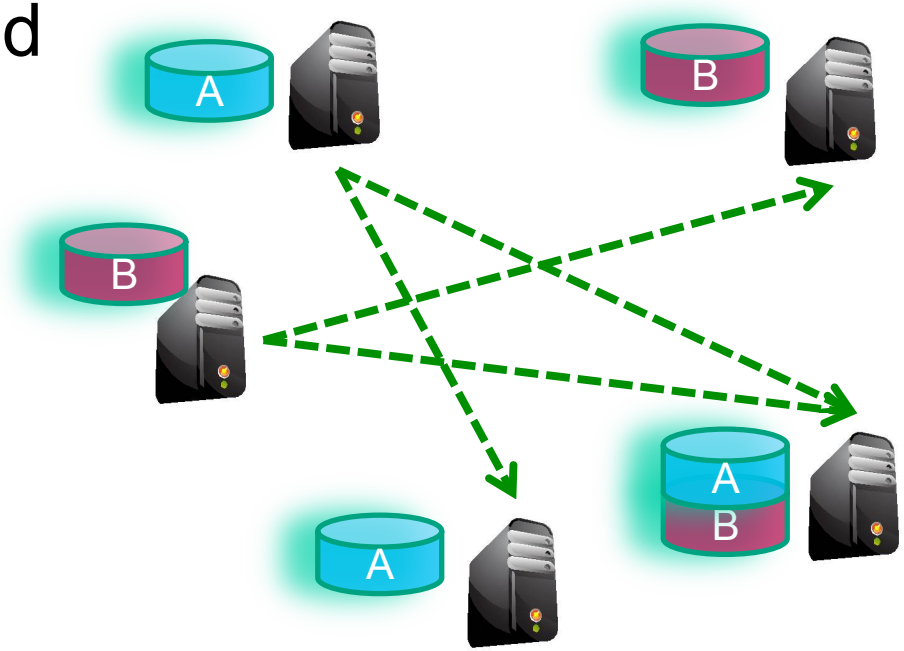
Bin Fan, Wittawat Tantisiriroj, Lin Xiao
Carnegie Mellon University

Carnegie Mellon
Parallel Data Laboratory

YAHOO!
pdsi
Office of Science

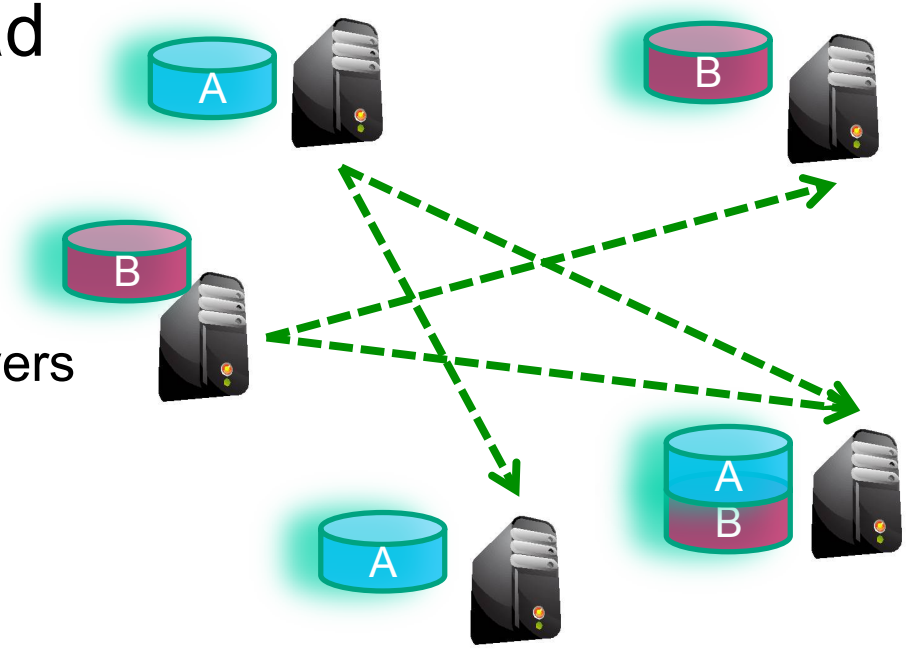
# Revisit HDFS Triplication

- GFS & HDFS triplicate every data block
  - Triplication: one local + two remote copies
- 200% space overhead
  - But RAID5 is simple?



# Revisit HDFS Triplication

- GFS & HDFS triplicate every data block
  - Triplication: one local + two remote copies
- 200% space overhead
  - But RAID5 is simple?
  - Can be done at scale
    - Panasas does it
      Object RAID over servers



# Revisit HDFS Triplication

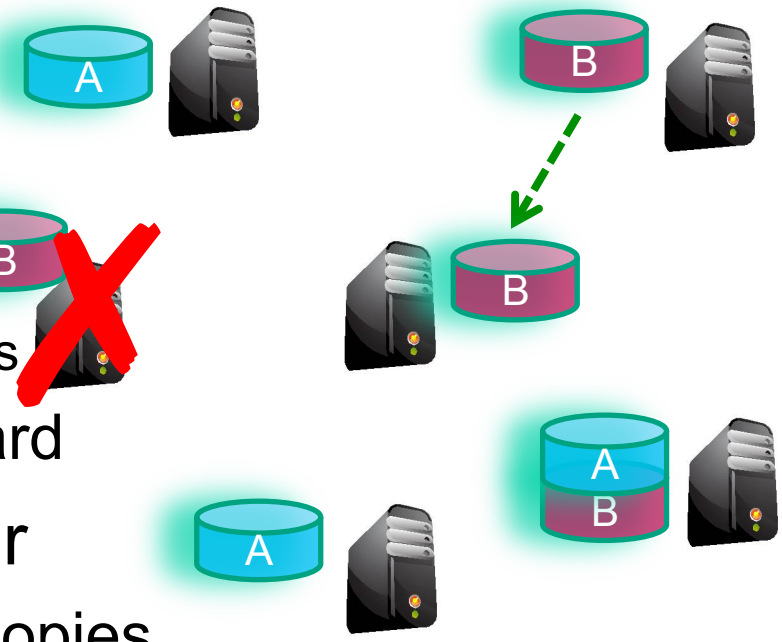
- GFS & HDFS triplicat
  - Triplication: one local -
- 200% space overhead
  - But RAID5 is simple?
  - Can be done at scale
    - panasas – Panasas does it
      >PF, >50 GB/s, >10K clients
  - But sync error handling hard



pdsi

# Revisit HDFS Reconstruction

- GFS & HDFS triplicate every data block
  - Triplication: one local + two remote copies
- 200% space overhead
  - But RAID5 is simple?
  - Can be done at scale
    - Panasas does it
      >PF, >50 GB/s, >10K clients
  - But sync error handling hard
- GFS & HDFS defer repair
  - Background task repairs copies
    - Notably less scary to developers



# Since June 09 Summit talk

- Hadoop HDFS (0.22.0) implemented a version of DiskReduce v1 (two copies + RAID 5 encoding)
- Thanks to Druba Borthakur & the HDFS team

## HDFS Raid

- Start the same: triplicate every data block
- Background encoding
  - Combine third replica of blocks from a single file to create parity block
  - Remove third replica
  - Apache JIRA HDFS-503
- DiskReduce from CMU
  - Garth Gibson research


A file with three blocks A, B and C

http://hadoopblog.blogspot.com/2009/08/hdfs-and-erasure-codes-hdfs-raid.html

# DiskReduce v2: RAID6 Encoding

Step0: N0 picks a codeword (x,N1,N2,N3,N4) randomly

Step1: N0 creates D1 and sends to N1, N2

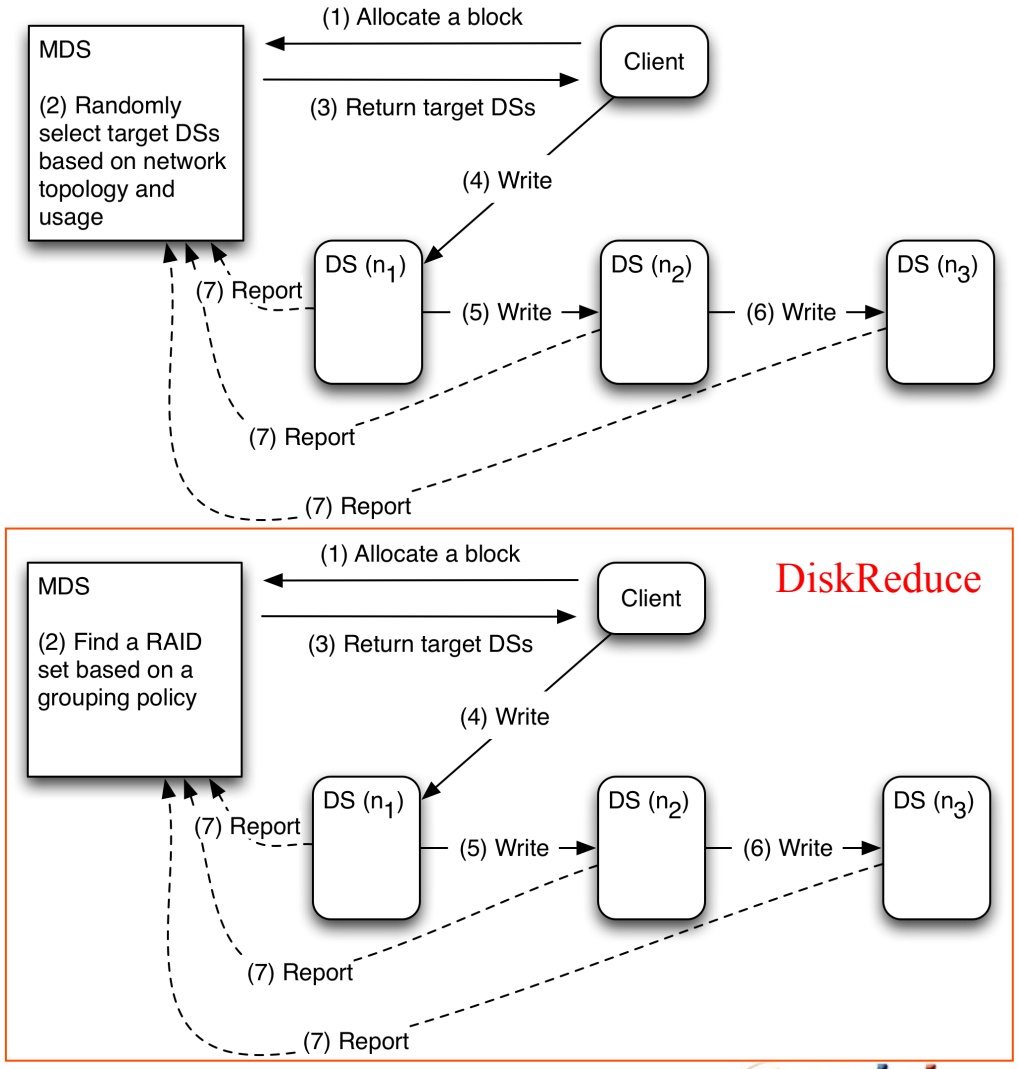

Step2: N0 creates D2 and sends to N1, N3


Step3: N0 creates D3 and sends to N1, N4


Step4: N0 and N1 encode D1,D2 and D3
and P1=f1(D1,D2,D3), P2=f2(D1,D2,D3)

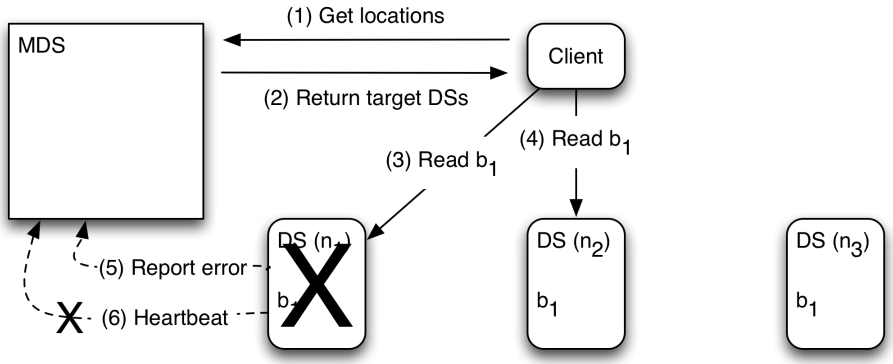
# Implementation - Write

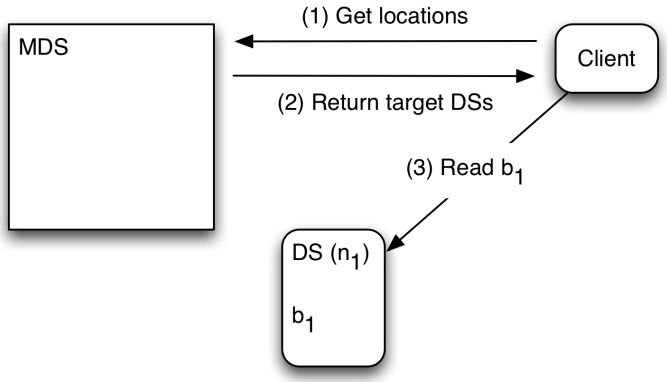
- Write unchanged
- Except policy for selecting location of replicas
- A key design principle is that initial writing is unchanged, starting with triplication

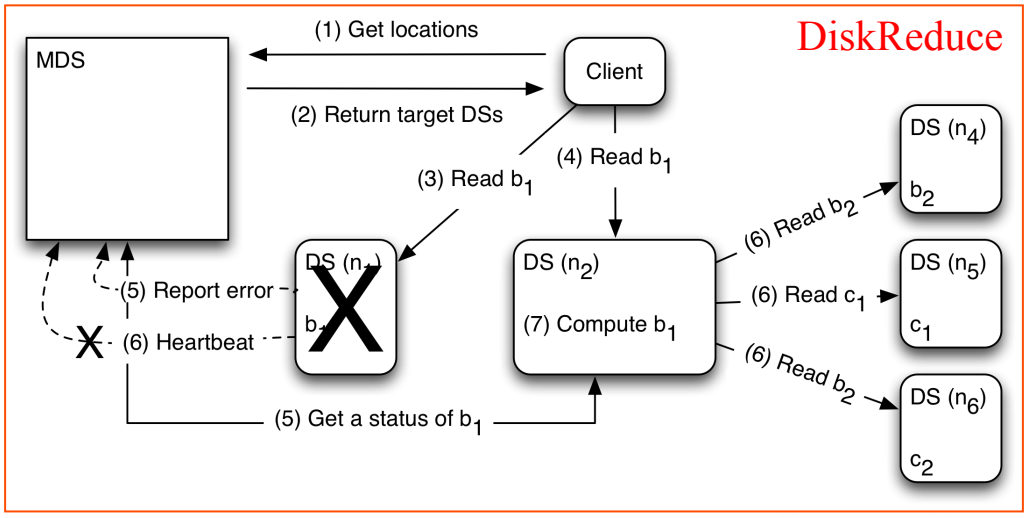


# Implementation - Read

- HDFS Read unchanged
- Except if block not found, then 2nd data server implements reconstruction
- HDFS client code unchanged !

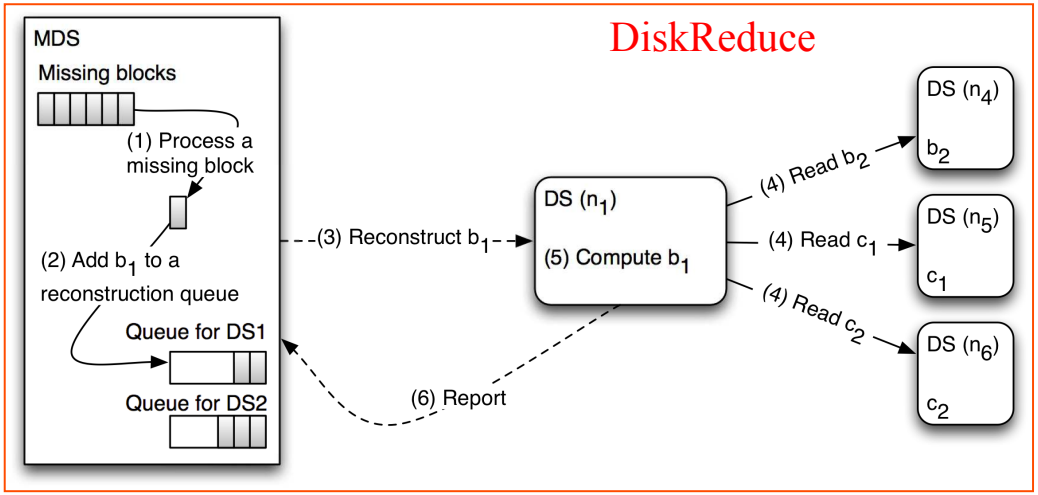


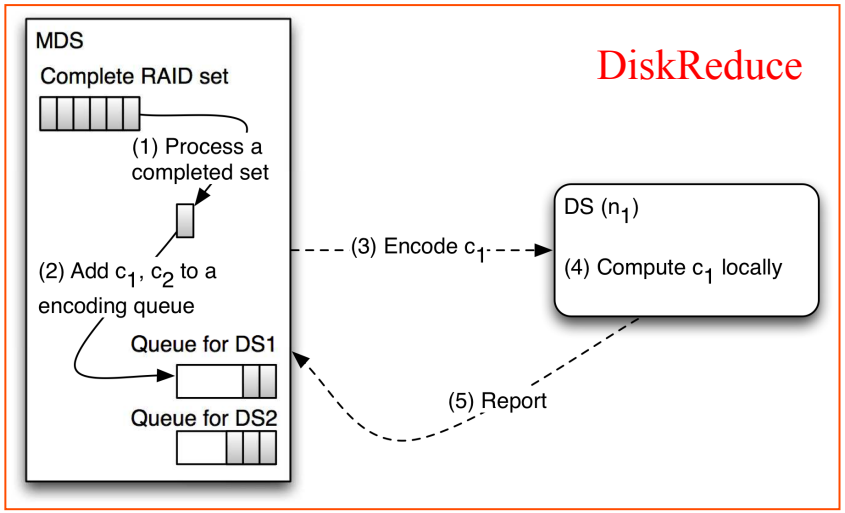



# Implementation – Recovery & Encoding

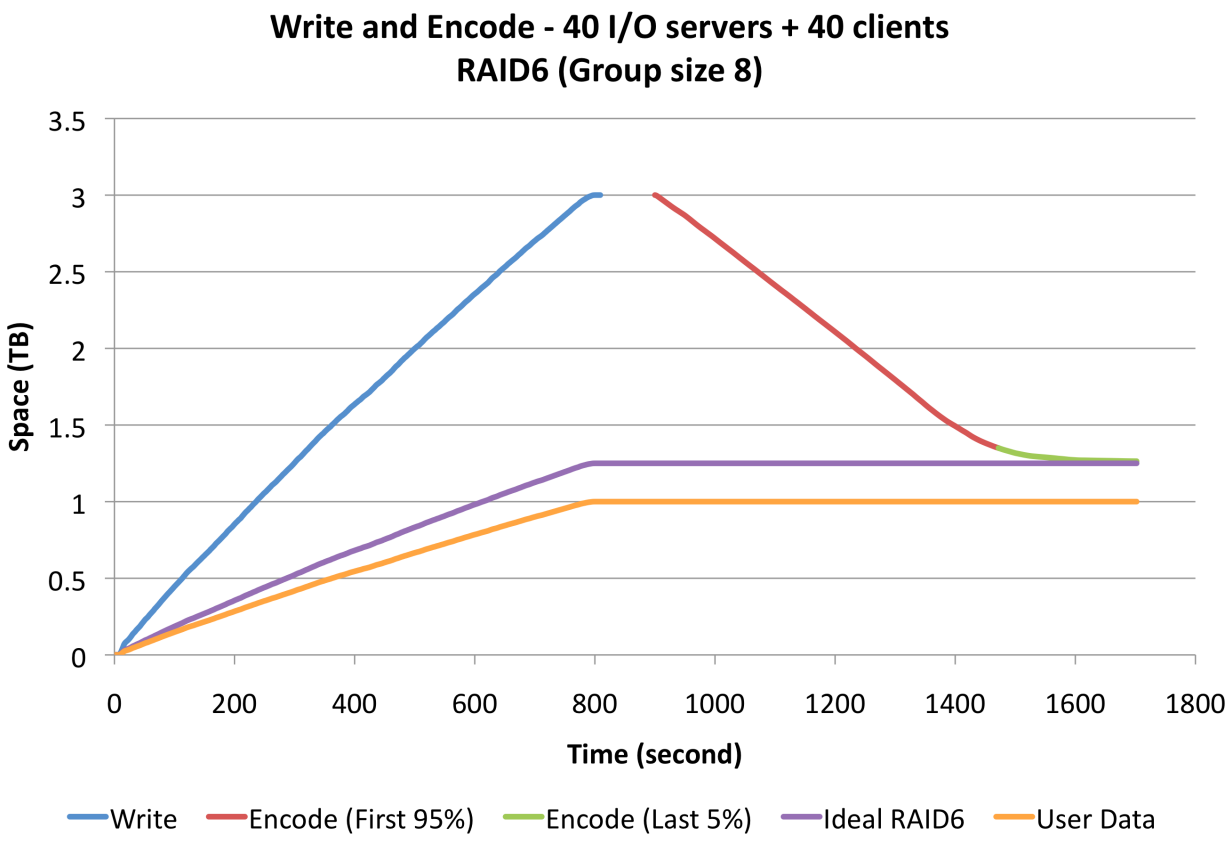
- Recovery extended
- A missing block is queued for recovery as in original but data server does RAID reconstruct
- Encoding is triggered using same queuing but computing check block can be all local if triplication of blocks in RAID set chosen appropriately



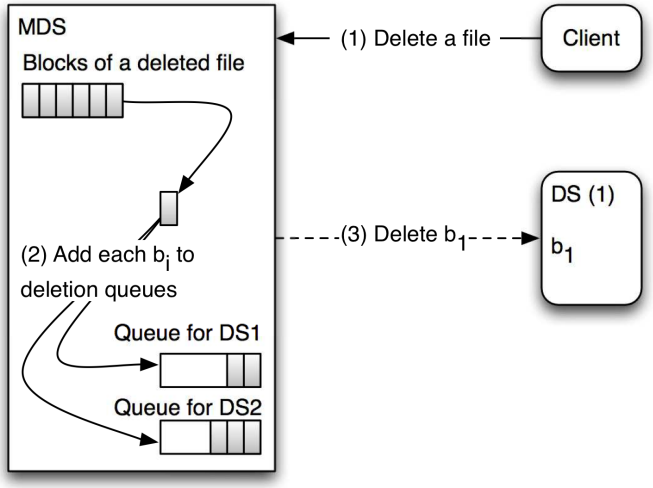
# Basic Implementation Working

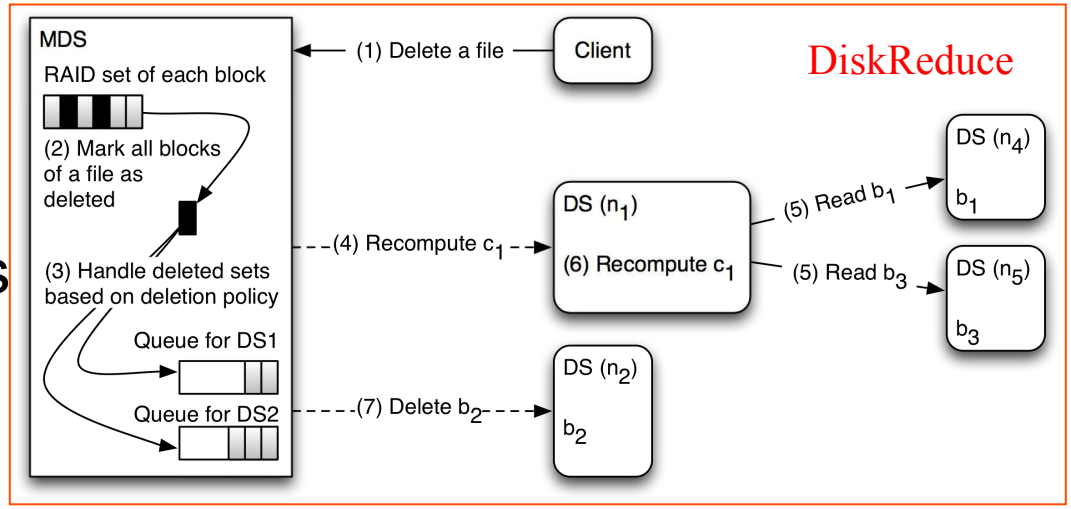
- Write 1TB (40 x 25GB) flat out: 1.25 user GB/s
- Flush cache and enable encoding: "compress" at 1.6 user GB/s for first 95% of data
- Background encoding is comparable in duration to initial write
- If "idle" time per "day" equal to "write" time, encoding is free



# Implementation - Delete

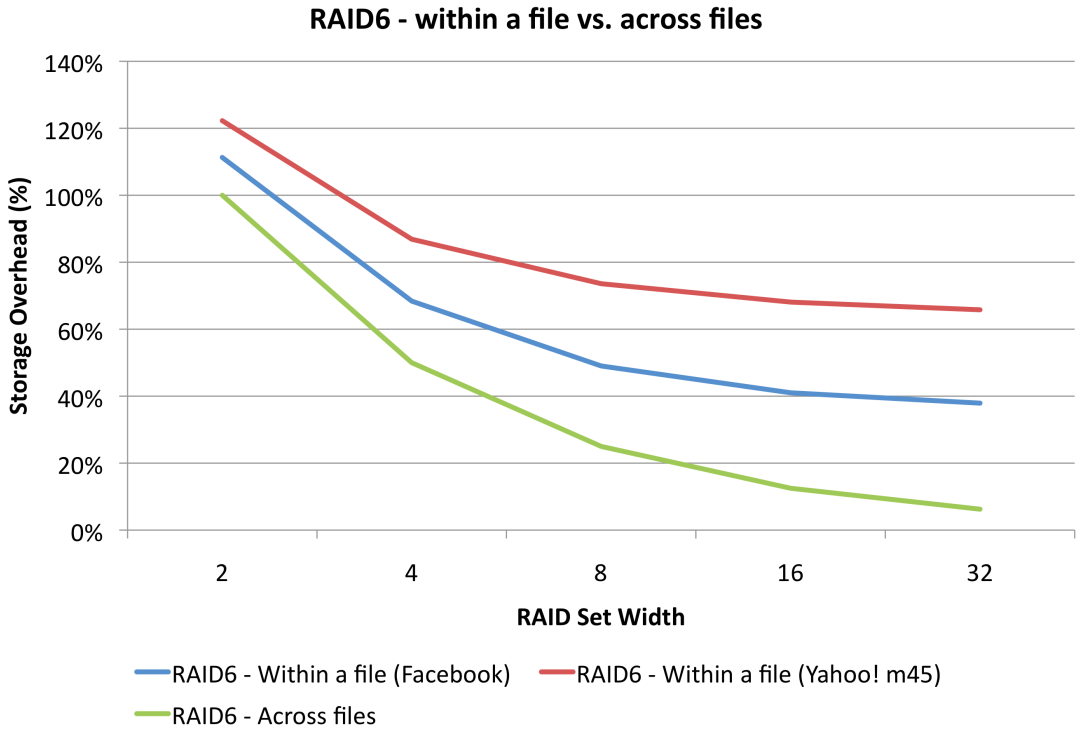
- Delete can be harder
- HDFS async deletes each block in a file
- In DiskReduce if a deleted block was in a RAID set with blocks that are not deleted check codes become wrong when block is gone – check blocks must be recomputed to recover capacity



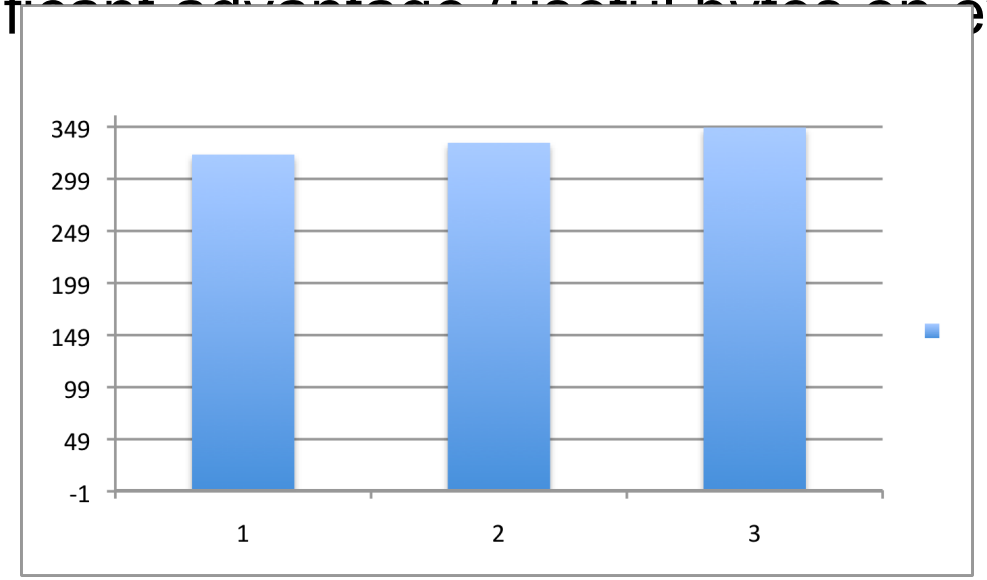
# Why not restrict RAID set to one file?

- In Hadoop clusters, files are mostly much smaller (in blocks) than the desired RAID set size
- Restriction of RAID set to one file simplifies delete, but costs significant overhead:
  3% -> up to 60%
- Traces from Yahoo M45 and from Facebook



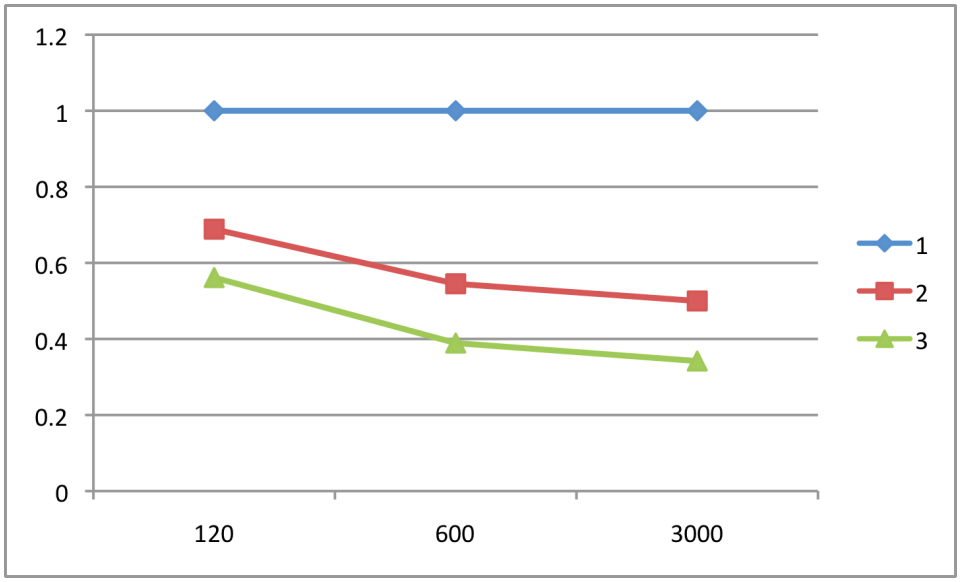
# Delaying Encoding – Advantages?

- Delaying encoding – performance advantages from having multiple copies to read from?
- Simple test: 29 nodes, 116 files each 4GB, 64MB blocks, read each byte once via Hadoop in Y seconds
- Three cases: one, two or three copies of each block
- No signif[illegible]very disk)



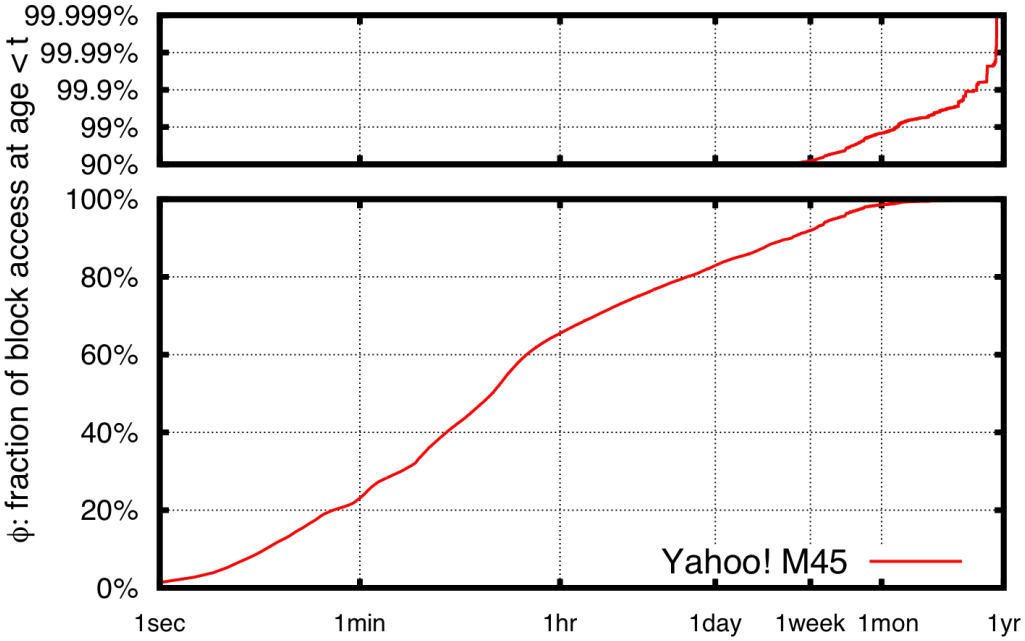
# Delaying Encoding – Advantages II?

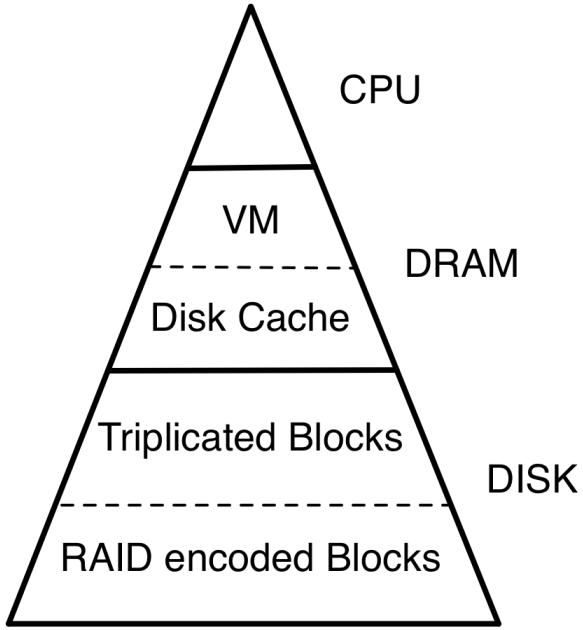
- Delaying encoding – performance advantages from having multiple copies to read from ??
- Try harder: small hot files: 512MB file in one 512MB block, read redundantly by X maps (30 nodes)
- Two copies faster by 25% - 50%, three copies faster by 40% - 60%
- There are significant performance benefits from replication, but harder to get than we expected



# Lets Cache!

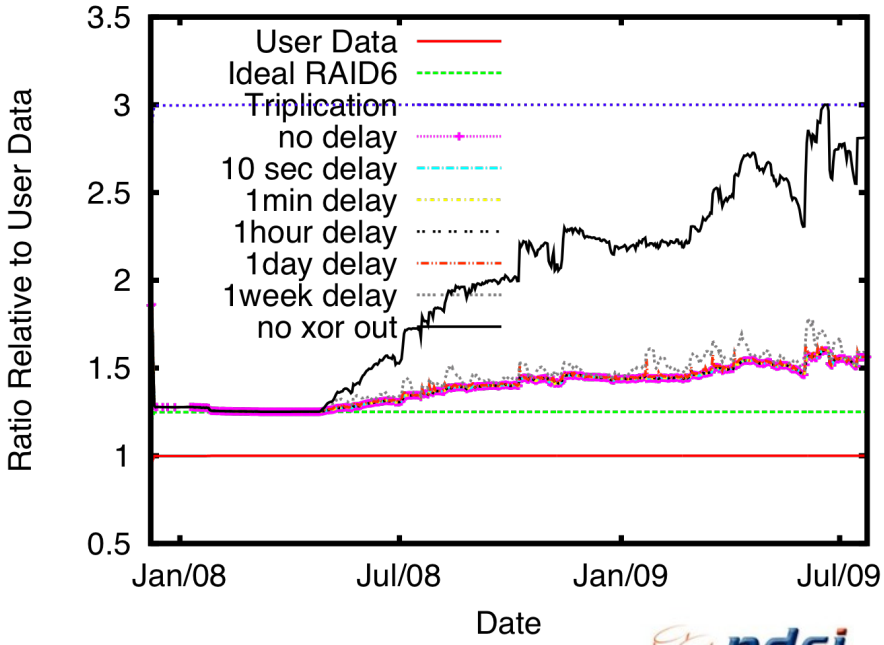
- Recently written data is triplicated, so delay encoding and treat two copies as performance improvement
- 80% of reads "hit" on 3 copies with 1 day delay
- Implementation underway



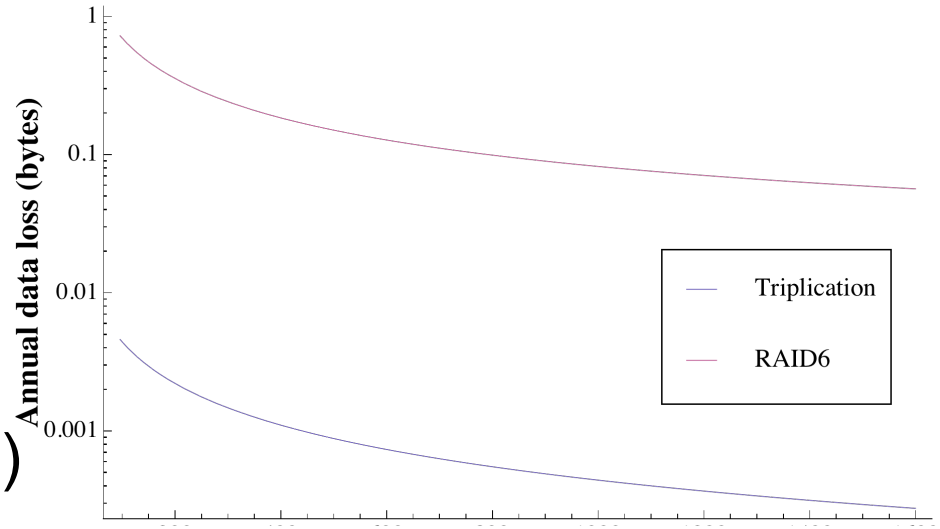
# Lets also Delay Delete

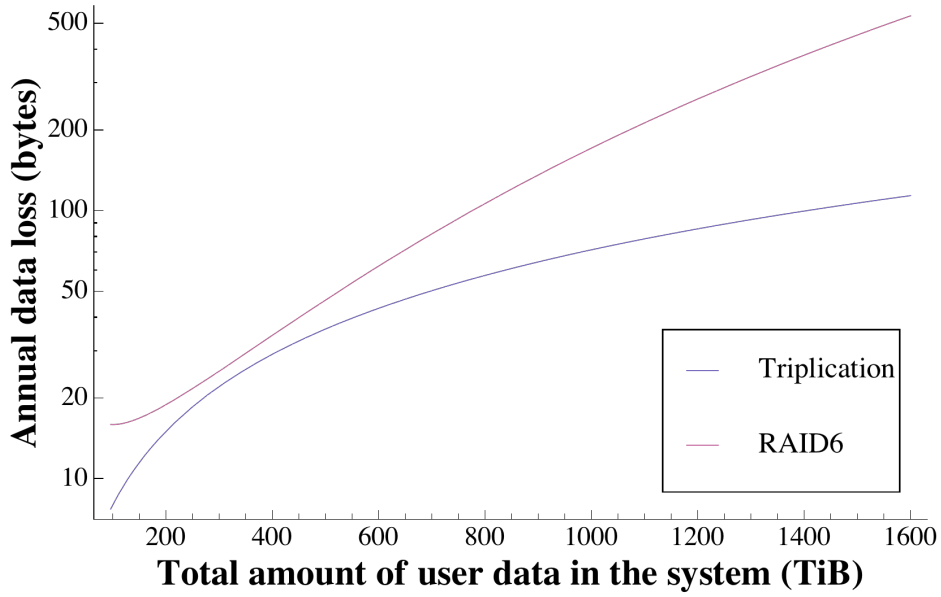
- Deleting a block in a RAID set forces check codes to be recomputed in order to recover block's space
- Delaying delete to avoid recompute (xor below) comes with a capacity penalty
- Penalty huge if wait for all blocks in RAID set to die
- Need to recompute to recover space, but can shift to "idle" time
- Interesting choices of which blocks in a RAID set to improve temporal locality of deletion



# What about Reliability Differences?

- Two fault tolerant is not the whole story
- Three copies more reliable
- Bigger systems less likely to have >2 blocks lost in any RAID set (8+2)/triple (3)
- Bigger systems "repair" in parallel faster (declustered)
- Triplication has ~3X disks for same user data, so ~3X faster repair
- Assume .8TB/disk used, 64MB blocks, 1% AFR disk fail, exponential repair is either 12/N or 0.5+12/N (Markov model)



# Closing

- DiskReduce for HDFS
  - Give users ~3X more stored data
  - Exploit async encode/delete for performance
  - Exploration of complexity in storage stack
  - Fragmentation, the never beaten annoyance

- CMU Open Cirrus, Open Cloud & DCO
  - Data-Intensive Scalable Computing resources
    - Utility for CMU science, testbed for PDL+
  - Broader agenda is “The Unreasonable Effectiveness of Data” for science and commerce