A Tale of Two Erasure Codes in HDFS

Mingyuan Xia, Mohit Saxena
Mario Blaum, David Pease

IBM Research Almaden & McGill University
Really Big Data Today

All Global Data in Zettabytes

1ZB = 1,126,000,000,000,000,000,000,000 bytes (approx)

UNECE data
Big Data Storage

GFS
Three-way replication

2003

Storage overhead: 3x
Big Data Storage

- GFS (2003): Three-way replication
- Facebook HDFS (2011): Reed-Solomon
- GFS v2 (2012): Reed-Solomon
- Microsoft Azure Storage (2011): LRC
- Facebook HDFS (2014): LRC, Hitchiker

Storage overhead: 3x

Erasure Coded Storage saves millions of $ for capital cost

1.4x~1.5x
Erasure Coding 101

Facebook HDFS: Reed Solomon (14,10)
Erasure Coding 101

Facebook HDFS: Reed Solomon (14,10)
• Compute 4 parities per 10 data blocks
• Storage overhead: \(1.4 \times 14/10\)
Erasure Coding 101

Facebook HDFS: Reed Solomon (14,10)
• Compute 4 parities per 10 data blocks
• Storage overhead: \(1.4x = \frac{14}{10}\)
• For a single failure, RS needs any 10 blocks over network from other nodes to recover
Erasure Coding 101

Facebook HDFS: Reed Solomon (14,10)

- Compute 4 parities per 10 data blocks
- Storage overhead: 1.4x
- For a single failure, RS needs any 10 blocks over network from other nodes to recover

Problems:
- Degraded read
- Data reconstruction
Problem 1: Degraded Read

Causes
- Software errors (>90%)
  - Rolling updates
  - Hot-spot effects
- Hardware errors
Problem 1: Degraded Read

Causes
- Software errors (>90%)
- Rolling updates
- Hot-spot effects
- Hardware errors

Extra latency for recovery

HDFS client

Read exception

HDFS

Data node

Data node

Data node
Problem 2: Data Reconstruction

**Causes**

- Disk or node failure
- Decommissioned nodes

[Diagram showing a MapReduce reconstruction job and data nodes in HDFS]
Problem 2: Disk/node Reconstruction

Production Clusters

• New data: 500TB~900TB/day
• Failure: lose ~100k blocks/day
• Reconstruction traffic: 180TB/day

[SIGCOMM’14] Rashmi et al. A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers
Recovery Cost vs. Storage Overhead

Facebook: RS(14,10)

Google: RS(9,6)

Single Coded Systems
Recovery Cost vs. Storage Overhead

- **Facebook**: RS(14,10)
  - Recover faster?
  - Use less storage?

- **Google**: RS(9,6)

---

Google: RS(9,6)
Facebook: RS(14,10)

Recovery Cost vs. Storage Overhead graph
HDFS Data Access Skew

10% of the data gets the majority of the accesses

90% data is accessed only a few times

File access frequency

File rank by decreasing access frequency

Four Cloudera customers and one Facebook workload

[VLDB’12] Chen et al., Interactive Query Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads
Two Erasure Codes

Coded with a **fast code** with low recovery cost

Coded with a **compact code** with high storage efficiency
Adaptive Coding in HDFS

3-way Replication
- Write cold
- Recently created

Reed Solomon

HDFS

Compact Code
- Read hot

Fast Code
- Read cold

3-way Replication
- Write cold
- Recently created

HACFS
Popular code families

- Product Code
- Local Reconstruction Code
- Reed-Solomon Code (MDS code)
- Partial MDS Code
- HoVer Code

Low recovery cost codes
Popular code families

Product Code
- Local Reconstruction Code
- Reed-Solomon Code (MDS code)
- Partial MDS Code
- HoVer Code

Low recovery cost codes
Fast and Compact Product Codes

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D6</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>D10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>P4</td>
<td>P5</td>
<td>P6</td>
<td>P7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fast code**  
(Product Code 2x5)

Storage overhead: **1.8x** (18/10)
Fast and Compact Product Codes

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D6</td>
<td></td>
<td></td>
<td>D8</td>
<td>D9</td>
<td>D10</td>
<td>P2</td>
</tr>
<tr>
<td>P3</td>
<td>P4</td>
<td>P5</td>
<td>P6</td>
<td>P7</td>
<td></td>
<td>P8</td>
</tr>
</tbody>
</table>

Fast code
(Product Code 2x5)
Recovery cost: 2
Storage overhead: 1.8x (18/10)
Fast and Compact Product Codes

<table>
<thead>
<tr>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D6</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>D10</td>
<td>P2</td>
</tr>
<tr>
<td>P3</td>
<td>P4</td>
<td>P5</td>
<td>P6</td>
<td>P7</td>
<td>P8</td>
</tr>
</tbody>
</table>

**Fast code**
(Product Code 2x5)

Recovery cost: 2
Storage overhead: 1.8x

**Compact code**
(Product Code 6x5)

Recovery cost: 5
Storage overhead: 1.4x
HACFS with Product Codes

![Graph showing Recovery Cost vs Storage Overhead for Single Coded Systems, with points labeled Compact code and Fast code.](image-url)
HACFS with Product Codes

![Graph showing Recovery Cost vs Storage Overhead for Single Coded Systems]
HACFS with Product Codes

Cold data accounts for the majority
HACFS with Product Codes

- Hot data is accessed more often

Diagram:
- Recovery Cost vs. Storage Overhead
- Single Coded Systems
HACFS with Product Codes

Recovery Cost vs. Storage Overhead

Optimal point: lowest recovery cost and storage overhead
Storage Bound

- **Storage Bound**
- **Recovery Cost**
- **Storage Overhead**

**Google Colossus**
(1.5x storage)

**Single Coded Systems**
Upcoding/Downcoding

- 3-way Replication
  - Write cold
  - Recently created

- Reed Solomon
  - Write cold

HDFS

HACFS

- 3-way Replication
  - Write cold
  - Recently created

- Compact Code
  - Read cold

- Fast Code
  - Read hot

Over the bound

Upcoding
Upcoding/Downcoding

- Recently created
- 3-way Replication
- Reed Solomon
- Write cold

HDFS

HACFS

- Recently created
- 3-way Replication
- Write cold

- Read cold
- Compact Code
- Over the bound
- Read hot
- Downcoding

- Read cold
- Below the bound
- Fast Code
- Upcoding
Upcoding for Product Codes

Fast code
PC(2x5)

Compact code
PC(6x5)

Parity-only Conversion
- Horizontal parties require no re-computation
- Vertical parities require no data block transfer
- All parity updates can be done in parallel and in a distributed manner
Efficient Up/Down-coding

• Popular code families with efficient up/down-coding
  ✓ Product Code
  ✓ Local Reconstruction Code
  ✓ Reed-Solomon Code (MDS code)
  ✓ Partial MDS Code
  ✓ HoVer Code

HACFS implementation

Applicable to other codes as well
Evaluation

• HACFS Implementation
  – Extension to Facebook’s HDFS
  – 3k LOC: three new modules

• Methodology
  – Five workloads: four Cloudera customers, one Facebook [VLDB’2012]
  – HDFS cluster: 11 nodes
  – Each node: 24 cores, 6 disks, 1 Gbps network
Experiment metrics

• Degraded read latency
  – Foreground read request delay
  – Caused mostly by software issues

• Reconstruction time
  – Background recovery for failures
  – Caused mostly by hardware failures

• Storage overhead (bounded)
Degraded Read Latency

![Graph showing Degraded Read Latency against Storage Overhead for different systems: Azure Storage (LRC(12,2,2)), Facebook HDFS-RAID (RS(10,4)), Single Coded Systems (RS(6,3)), and Google Colossus.](image)
Degraded Read Latency

![Degraded Read Latency Diagram](image)

- **HACFS-PC**
- **Single Coded Systems**
- **LRC(12,2,2)**
- **RS(10,4)**
- **RS(6,3)**
Degraded Read Latency

- HACFS-PC
- Single Coded Systems
- Google Colossus: 1.5x

Graph shows the relationship between Degraded Read Latency and Storage Overhead for various systems.
Degraded Read Latency

![Graph showing Degraded Read Latency vs Storage Overhead with markers for different systems: Single Coded Systems (HACFS-PC, HACFS-LRC), LRC(12,2,2), CC1, CC2, CC3, CC4, FB, RS(10,4), RS(6,3).]
Degraded Read Latency

CC1
86% accesses to hot data
94.2% of the data is cold
Reconstruction Time

Cold data in CC3: 75.5%
Reconstruction Time

Cold data in CC1: 94.2%

Cold data in CC3: 75.5%

HACFS-PC
HACFS-LRC
RS(10,4)
RS(6,3)
LRC(12,2,2)

Single Coded Systems

Storage Overhead
## System Comparisons

<table>
<thead>
<tr>
<th></th>
<th>HACFS using Product Codes</th>
<th>HACFS using LRCs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Colossus FS</td>
<td>FB HDFS</td>
</tr>
<tr>
<td>Degraded Read Latency</td>
<td>25.2%</td>
<td>46.1%</td>
</tr>
<tr>
<td>Reconstruction Time</td>
<td>14.3%</td>
<td>43.7%</td>
</tr>
<tr>
<td>Storage Overhead</td>
<td>2.3%</td>
<td>-4.7%</td>
</tr>
</tbody>
</table>
## System Comparisons

<table>
<thead>
<tr>
<th></th>
<th>HACFS using Product Codes</th>
<th>HACFS using LRCs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Colossus FS</td>
<td>FB HDFS</td>
</tr>
<tr>
<td>Degraded Read Latency</td>
<td>25.2%</td>
<td>46.1%</td>
</tr>
<tr>
<td>Reconstruction Time</td>
<td>14.3%</td>
<td>43.7%</td>
</tr>
<tr>
<td>Storage Overhead</td>
<td>2.3%</td>
<td>-4.7%</td>
</tr>
</tbody>
</table>
Conclusions

![Graph showing recovery cost vs. storage overhead for different systems: Facebook HDFS, Microsoft Azure, Google ColossusFS, and HACFS. The graph illustrates the performance of these systems in terms of recovery cost and storage overhead.](image-url)
Thanks Q/A

FAST’15

A Tale of Two Erasure Codes in HDFS

Mingyuan Xia, Mohit Saxena
Mario Blaum, David Pease

IBM Research Almaden & McGill University