Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage

Athreya Chandramouli †, **Abhinav Vaishya** †, **Prasad Krishnan** †

† International Institute of Information Technology, Hyderabad (IIIT-H)

Information Theory Workshop, 2022
Mumbai, India.

Outline

1. Data Skew and Data Rebalancing in Distributed Systems
2. Coded Data Rebalancing : Formal System Model
3. Proposed Rebalancing Schemes for Cyclic Databases
4. Conclusions and Future Work

Table of Contents

1. Data Skew and Data Rebalancing in Distributed Systems
2. Coded Data Rebalancing : Formal System Model
3. Proposed Rebalancing Schemes for Cyclic Databases
4. Conclusions and Future Work

Distributed analytics engines comprise of

- Distributed File System to provide access to the distributed database across several nodes
- Distributed computing platform to enable parallel processing of data in the distributed database.

Data replication in the database provides

- Fault tolerance
- Availability
- Reduced latency

Data Skew in Distributed Databases

Data Skew

Non-uniform distribution of data across storage nodes

Can arise because of

- Node additions or removals
- Behaviour of client applications
- Behaviour of the file system

Leads to

- Load imbalance
- Stragglers
- Increase in task completion time

Remedy : *Data Rebalancing*

Data Rebalancing

Data Rebalancing

Redistribute data across the available nodes to **balance the distribution** and **maintain replication factor**

- Rebalancing may be needed at regular intervals
- Communication costs
- Reduction in performance during rebalancing.

Data Rebalancing

Data Rebalancing

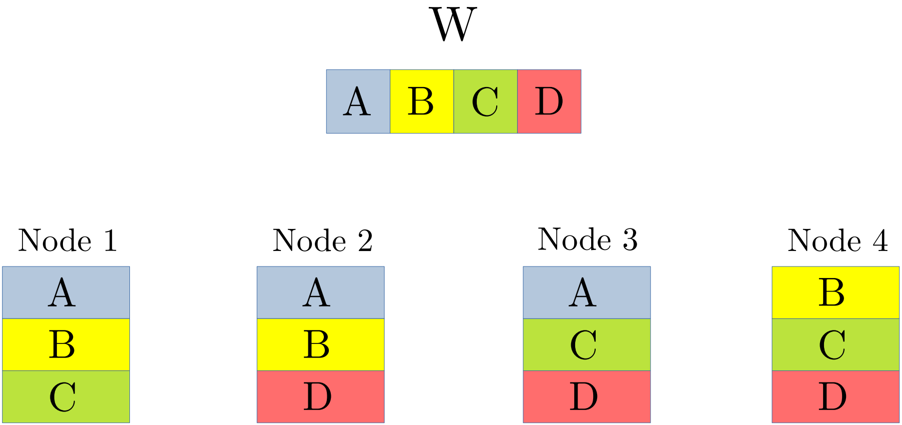
Redistribute data across the available nodes to **balance the distribution** and **maintain replication factor**

- Rebalancing may be needed at regular intervals
- Communication costs
- Reduction in performance during rebalancing.

Coded Data Rebalancing for node-removal and node-addition

- **Broadcast Coded transmissions** reduces rebalancing communication costs and time-to-rebalance.
- **Exploit data replication** for enabling coding opportunities.
- **Structural Invariance:** Preserve database structure (replication factor) post rebalancing.

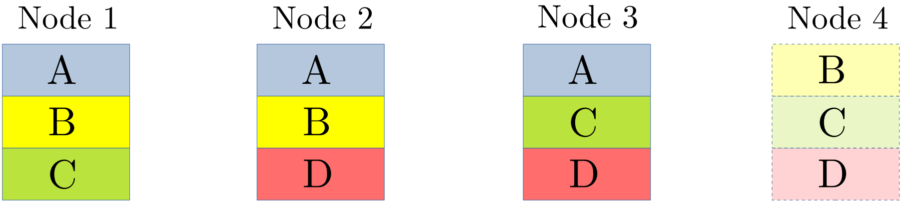
Example - Rebalancing after node removal



Replication factor $r = 3$

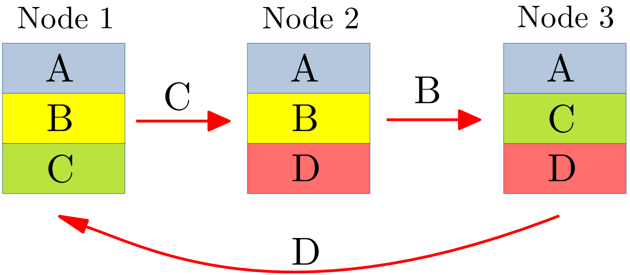
Example

Replication factor drops for B, C, D, after removal of Node 4.



Example - Uncoded Rebalancing Scheme

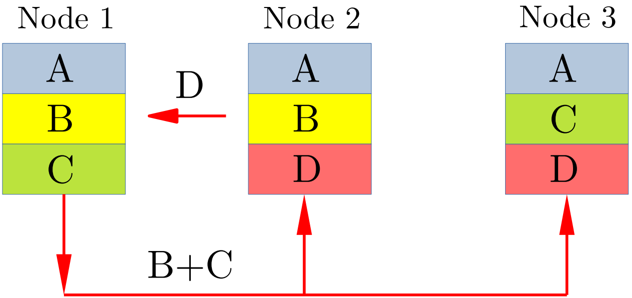
Uncoded rebalancing to restore replication factor



Requires 3 transmissions

Example - A Coded Rebalancing Scheme

Coded rebalancing over broadcast to restore replication factor



Requires 2 transmissions

Example - Final Database

Node 1	Node 2	Node 3
A	A	A
B	B	C
C	D	D
D	C	B

Table of Contents

1. Data Skew and Data Rebalancing in Distributed Systems
2. Coded Data Rebalancing : Formal System Model
3. Proposed Rebalancing Schemes for Cyclic Databases
4. Conclusions and Future Work

System Model: Initial database

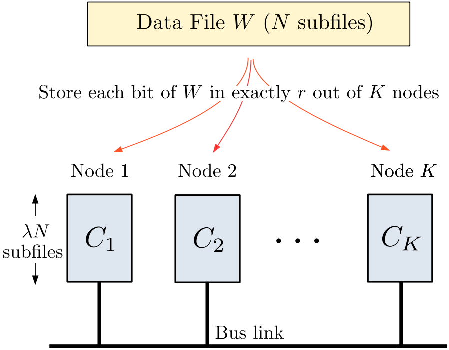


Figure: **An r-balanced distributed database $\mathcal{C}(r, [K])$, where $[K] = \{1, \ldots, K\}$.**

- r : *Replication factor*
- 'Balanced': each node stores $\frac{r}{K}$ fraction of the data.

Node Removal and Rebalancing

- Suppose node K is removed from the system.
- Let T be the size of a segment (subfile).

Rebalancing Process

- Broadcast coded transmissions between the surviving $K-1$ nodes.
- Let X_i be the transmission from node i.

Communication Load

$$L_{rem}(r) = \frac{\text{Number of bits transmitted}}{\text{Size of a segment}} = \frac{\sum_{i=1}^{K-1} |X_i|}{T}$$

Previous Results

Main Result

For balanced distributed databases on K nodes with replication factor $r \geq 2$, there exists a rebalancing scheme for node removal

$$L_{rem}(r) = \frac{\frac{Nr}{K}}{r-1}, \text{ where N is the number of segments of a file}$$

Previous Results

Main Result

For balanced distributed databases on K nodes with replication factor $r \geq 2$, there exists a rebalancing scheme for node removal

$$L_{rem}(r) = \frac{\frac{Nr}{K}}{r-1}, \text{ where N is the number of segments of a file}$$

Optimality

Optimal communication load for node removal and node addition scenarios. [1]

Previous Results

Main Result

For balanced distributed databases on K nodes with replication factor $r \geq 2$, there exists a rebalancing scheme for node removal

$$L_{rem}(r) = \frac{\frac{Nr}{K}}{r-1}, \text{ where N is the number of segments of a file}$$

Optimality

Optimal communication load for node removal and node addition scenarios. [1]

Major Issue

File Size NT must be at least exponential in K.

[1] P. Krishnan, V. Lalitha, and L. Natarajan, "Coded data rebalancing: Fundamental limits and constructions", ISIT 2020.

Cyclic Databases : Family of r-balanced Databases

Overcoming the large file-size requirement

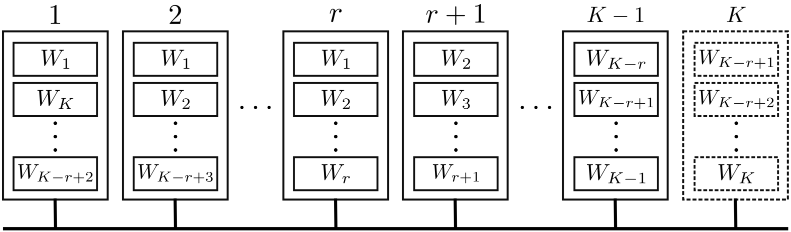


Figure: r-balanced cyclic database on nodes $[K]$

- The file W is divided into K segments, $W_1, W_2, \ldots, W_K$.
- Each $W_i, i \in [K]$ is stored in r consecutive nodes starting from i in a wrap-around fashion.

Main Contributions

Cyclic balanced databases

Rebalancing schemes for Cyclic balanced databases

Main Contributions

Cyclic balanced databases

Rebalancing schemes for Cyclic balanced databases

File Size NT

$$NT = O(K^3)$$

Main Contributions

Cyclic balanced databases

Rebalancing schemes for Cyclic balanced databases

File Size NT

$$NT = O(K^3)$$

Communication Load

- The communication load for the node removal case is strictly lower than that of the uncoded scheme.
- Optimal load for the node addition case.

Main Theorem

For an r-balanced cyclic database having K nodes and $r \in \{3, \ldots, K-1\}$, rebalancing schemes exist which achieve the following communication load

$$L_{\text{rem}}(r) = \frac{K-r}{(K-1)} + \min\left(L_1(r), L_2(r)\right)$$

where, $L_1(r) = \frac{(K-r)(2r-1)}{(K-1)}$ and $L_2(r) = \frac{1}{2(K-1)}\left(K(r-1) + \lceil \frac{r^2-2r}{2} \rceil\right)$.

Comparisons with other schemes

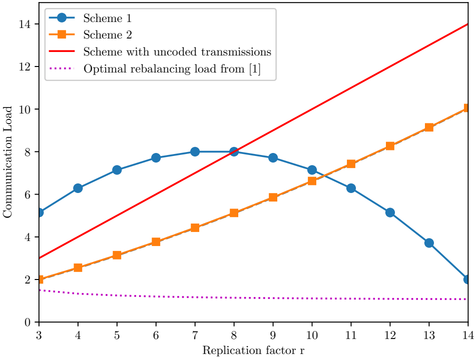


Figure: K=15, varying r

[1] P. Krishnan, V. Lalitha, and L. Natarajan, "Coded data rebalancing: Fundamental limits and constructions", ISIT 2020.

Table of Contents

1 Data Skew and Data Rebalancing in Distributed Systems

2 Coded Data Rebalancing : Formal System Model

3 Proposed Rebalancing Schemes for Cyclic Databases

4 Conclusions and Future Work

Initial and Final balanced databases

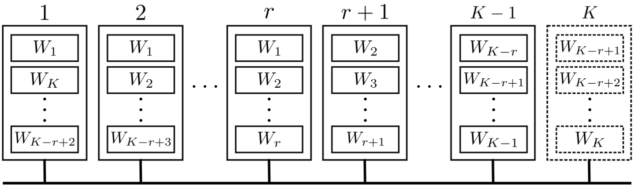


Figure: r-balanced cyclic database on nodes $[K]$

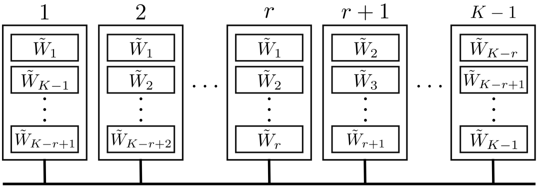


Figure: Target r-balanced cyclic database on nodes $[K-1]$

Example

- $K = 8, r = 6$.
- Divide W into 8 segments, indexed by W_i, $i \in [8]$.
- W_1 is stored in nodes $\{1, 2, \ldots, 6\}$, W_2 in nodes $\{2, 3, \ldots, 7\}$, and so on.
- Node 8 which has segments $\{W_3, W_4, \ldots, W_8\}$ is removed.

Intuition for Rebalancing Algorithm

To keep the communication load small,

- Move bits as minimally as possible.
- Maximize use of coding opportunity (encode many subsegments together in each transmission).

Overview of Rebalancing Algorithm

Our rebalancing algorithm involves three phases:

Splitting

The segments which were present in the removed node are split into subsegments.

Overview of Rebalancing Algorithm

Our rebalancing algorithm involves three phases:

Splitting

The segments which were present in the removed node are split into subsegments.

Transmission

Coded (and some uncoded) subsegments are transmitted.

Overview of Rebalancing Algorithm

Our rebalancing algorithm involves three phases:

Splitting

The segments which were present in the removed node are split into subsegments.

Transmission

Coded (and some uncoded) subsegments are transmitted.

Merging

Decoded subsegments are merged with existing segments.

Splitting: Intuition

Notations

- $\tilde{W}_j$: j^{th}segment in the target database
- S_i: set of nodes containing i^{th}segment in the initial database
- $\tilde{S}_j$: set of nodes containing j^{th}segment in the target database

Splitting: Intuition

Notations

- $\tilde{W}_j$: j^{th}segment in the target database
- S_i: set of nodes containing i^{th}segment in the initial database
- $\tilde{S}_j$: set of nodes containing j^{th}segment in the target database

Intuition

- We seek to split W_i into subsegments and merge these into those $\tilde{W}_j : j \in [K-1]$ such that $|\tilde{S}_j \cap S_i|$ is as large as possible.
- The subsegment of segment W_i which is to be merged into $\tilde{W}_j$, and thus to be placed in the nodes $\tilde{S}_j \setminus S_i$, as $W_i^{\tilde{S}_j \setminus S_i}$.
- Making $|\tilde{S}_j \cap S_i|$ large reduces $|\tilde{S}_j \setminus S_i|$, which further reduces the movement of subsegments during rebalancing.

Splitting

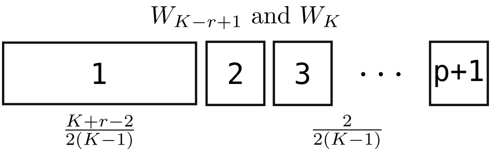


Figure: Splitting of the corner segments when $K - r$ is even. Here, $p = \lfloor \frac{K-r}{2} \rfloor$.

- The first subsegment, i.e., the largest subsegment, will be transmitted via coded transmissions.
- Uncoded transmissions for all the other smaller subsegments.

Splitting

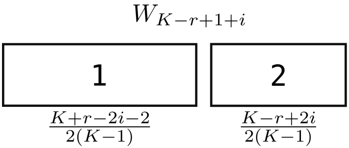


Figure: Splitting of the middle segments.

- Two subsegments in total.
- Coded transmissions for both.

Transmission: Main Idea

XOR-coded Transmissions

Due to cyclicity, groups of nodes separated by $K - r$ indices provide Coding Opportunity $\Rightarrow$ XOR-based schemes

Transmission: Main Idea

XOR-coded Transmissions

Due to cyclicity, groups of nodes separated by $K - r$ indices provide Coding Opportunity $\Rightarrow$ XOR-based schemes

Uncoded Transmissions

Subsegments which won't be a part of any XOR-coded transmission will be broadcast separately to the nodes where they are required.

Example

- $K = 8, r = 6$
- Node 8 has segments $\{W_3, W_4, W_5, W_6, W_7, W_8\}$
- Splitting:
  - W_3: $W_3^{\{1\}}(large), W_3^{\{2\}}(small)$
  - $W_4 : W_4^{\{2\}}, W_4^{\{3\}}$
  - $W_5 : W_5^{\{3\}}, W_5^{\{4\}}$
  - $W_6 : W_6^{\{4\}}, W_6^{\{5\}}$
  - $W_7 : W_7^{\{5\}}, W_7^{\{6\}}$
  - W_8: $W_8^{\{6\}}(large), W_8^{\{7\}}(small)$
- Transmission: The superscript $\{1\}$ in $W_3^{\{1\}}$ means that this subsegment will be transmitted to node 1.
- Merging: $W_3^{\{1\}}$ will be merged with $\tilde{W}_3$ as $\tilde{S}_3 \setminus S_3 = \{1\}$ $(S_3 = \{3, \ldots, 8\}, \tilde{S}_3 = \{3, \ldots, 7, 1\})$.

Example

Nodes / Subsegments	1	2	3	4	5	6	7
$W_3^{\{1\}}$	s	—	*	*	*	*	*
$W_4^{\{2\}}$	*	s	—	*	*	*	*
$W_5^{\{3\}}$	*	*	s	—	*	*	*
$W_6^{\{4\}}$	*	*	*	s	—	*	*
$W_7^{\{5\}}$	*	*	*	*	s	—	*
$W_4^{\{3\}}$	*	—	s	*	*	*	*
$W_5^{\{4\}}$	*	*	—	s	*	*	*
$W_6^{\{5\}}$	*	*	*	—	s	*	*
$W_7^{\{6\}}$	*	*	*	*	—	s	*
$W_8^{\{7\}}$	*	*	*	*	*	—	s
$W_3^{\{2\}}$	—	s	*	*	*	*	*
$W_8^{\{6\}}$	*	*	*	*	*	s	—

Example

Nodes / Subsegments	1	2	3	4	5	6	7
$W_3^{\{1\}}$	(s)	—	*	*	*	*	(*)
$W_4^{\{2\}}$	*	s	—	*	*	*	*
$W_5^{\{3\}}$	*	*	(s)	—	*	*	(*)
$W_6^{\{4\}}$	*	*	*	s	—	*	*
$W_7^{\{5\}}$	*	*	*	*	(s)	—	(*)
$W_4^{\{3\}}$	*	—	s	*	*	*	*
$W_5^{\{4\}}$	*	*	—	s	*	*	*
$W_6^{\{5\}}$	*	*	*	—	s	*	*
$W_7^{\{6\}}$	*	*	*	*	—	s	*
$W_8^{\{7\}}$	*	*	*	*	*	—	s
$W_3^{\{2\}}$	—	s	*	*	*	*	*
$W_8^{\{6\}}$	*	*	*	*	*	s	—

Example

Nodes / Subsegments	1	2	3	4	5	6	7
$W_3^{\{1\}}$	S	—	*	*	*	*	*
$W_4^{\{2\}}$	*	S	—	*	*	*	*
$W_5^{\{3\}}$	*	*	S	—	*	*	*
$W_6^{\{4\}}$	*	*	*	S	—	*	*
$W_7^{\{5\}}$	*	*	*	*	S	—	*
$W_4^{\{3\}}$	*	—	S	*	*	*	*
$W_5^{\{4\}}$	*	*	—	S	*	*	*
$W_6^{\{5\}}$	*	*	*	—	S	*	*
$W_7^{\{6\}}$	*	*	*	*	—	S	*
$W_8^{\{7\}}$	*	*	*	*	*	—	S
$W_3^{\{2\}}$	—	S	*	*	*	*	*
$W_8^{\{6\}}$	*	*	*	*	*	S	—

Merging and Relabelling

- All the subsegments $W_i^{\tilde{S}_j \setminus S_i}$ for all possible $i \in [K-r+1, K]$, will be merged into $\tilde{W}_j$, as $|\tilde{S}_j \setminus S_i|$ is the minimum set difference possible.
- For $j \in [1, K-r]$, W_j will also be merged into $\tilde{W}_j$.

Table of Contents

1 Data Skew and Data Rebalancing in Distributed Systems

2 Coded Data Rebalancing : Formal System Model

3 Proposed Rebalancing Schemes for Cyclic Databases

4 Conclusions and Future Work

Conclusions and Future work

Conclusions

- Framework for Coded Rebalancing for handling data skew in cyclic databases with an improved file-size requirement.
- Rebalancing Schemes for node removal and addition.

Future work

- Multiple simultaneous node removals or additions in case of cyclic databases
- Constructing good converse arguments in the cyclic database setting.

https://arxiv.org/abs/2205.06257

Thank You!