Low Complexity Cache-Aided Communication Schemes for Distributed Data Storage and Distributed Computing

Abhinav Vaishya

Master of Science
in
Computer Science and Engineering
by Research

Advisor: Dr. Prasad Krishnan
Signal Processing and Communication Research Center (SPCRC),
International Institute of Information Technology, Hyderabad

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Outline

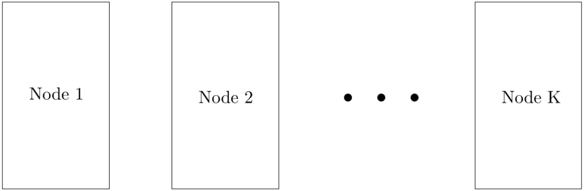
1. Introduction
2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
4. Conclusions and Future Work

Distributed Data Analytics Engines

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.

Distributed Data Storage System

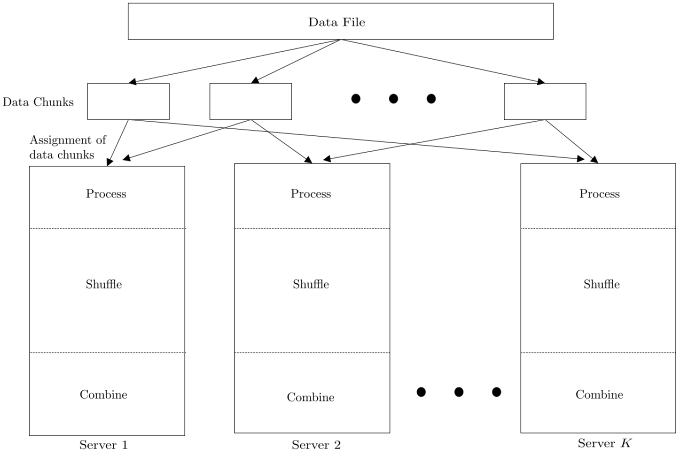


- Types:
  - Replication-based: Each chunk of data replicated at r nodes
  - Erasure Coded: Data encoded using a code and stored across the nodes
- Redundancies $\Rightarrow$ Improved fault tolerances and reduced risk of data loss.

Data Rebalancing in Distributed Data Storage System

- Technique to overcome the issue of data skew, i.e., non-uniform distribution of data.
- Involves transfer of high volumes of data (communication load) between the nodes.
- Coded Data Rebalancing: make use of coding opportunities to bring down the communication load.

Distributed Computing Framework



- Software framework used to process the data stored in a distributed file system in parallel.

Data Shuffling in Distributed Computing Framework

- MapReduce framework
- Three phases of computation: Map, Shuffle, Reduce.

Map Phase

- Generate intermediate values (IVAs) using the present data.

Data Shuffling in Distributed Computing Framework

- MapReduce framework
- Three phases of computation: Map, Shuffle, Reduce.

Map Phase

- Generate intermediate values (IVAs) using the present data.

Shuffle Phase

- Exchange of IVAs to fulfill the requirements.
- Involves movement of high volumes of data (communication load).
- Coded Distributed Computing: make use of coding opportunities to bring down the communication load.

Data Shuffling in Distributed Computing Framework

- MapReduce framework
- Three phases of computation: Map, Shuffle, Reduce.

Map Phase

- Generate intermediate values (IVAs) using the present data.

Shuffle Phase

- Exchange of IVAs to fulfill the requirements.
- Involves movement of high volumes of data (communication load).
- Coded Distributed Computing: make use of coding opportunities to bring down the communication load.

Reduce Phase

- Required outputs are produced.

Table of Contents

1. Introduction
2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
   - Data Skew and Data Rebalancing in Distributed Systems
   - Coded Data Rebalancing : Formal System Model
   - Proposed Rebalancing Schemes for Cyclic Databases
3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
4. Conclusions and Future Work

Replication-based Distributed Data Storage Systems

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

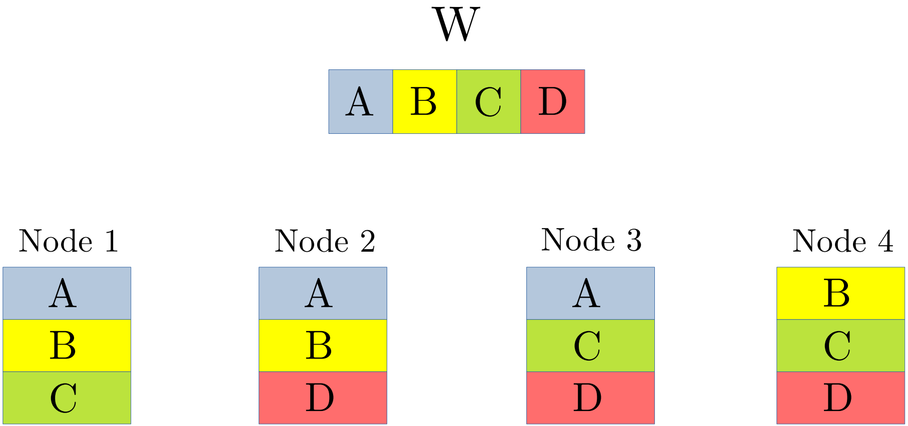
Redistribute data across the available nodes to **balance the distribution** and **maintain replication factor**

- 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.

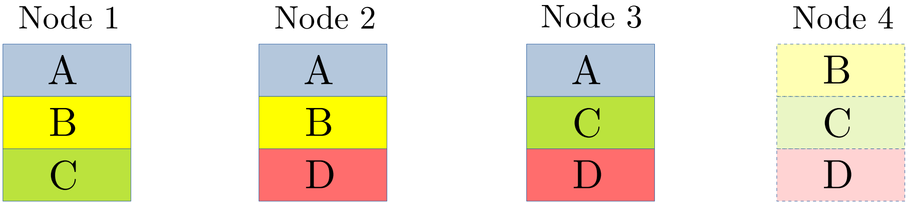
Example - Rebalancing after node removal



Replication factor $r = 3$

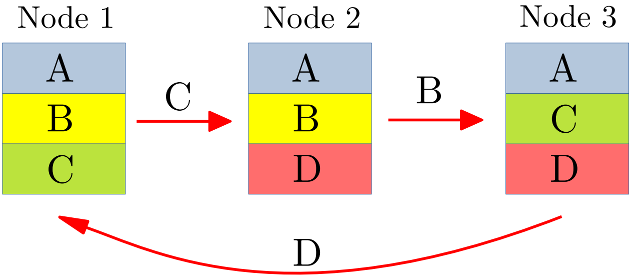
Example

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



Example - Uncoded Rebalancing Scheme

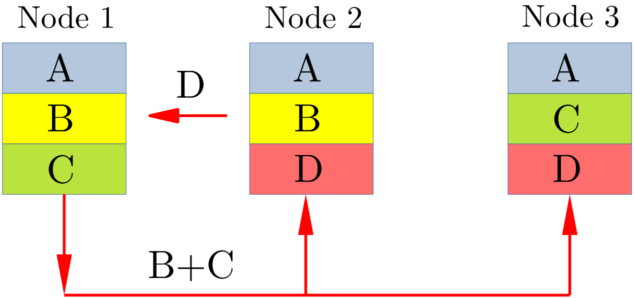
Uncoded rebalancing to restore replication factor



Requires 3 transmissions

Example - A Coded Rebalancing Scheme

Coded rebalancing over broadcast to restore replication factor



Requires 2 transmissions

Example - Final Database

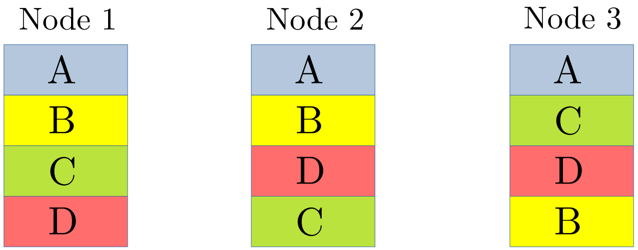


Table of Contents

1. Introduction
2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
   - Data Skew and Data Rebalancing in Distributed Systems
   - Coded Data Rebalancing : Formal System Model
   - Proposed Rebalancing Schemes for Cyclic Databases
3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
4. Conclusions and Future Work

System Model: Initial database

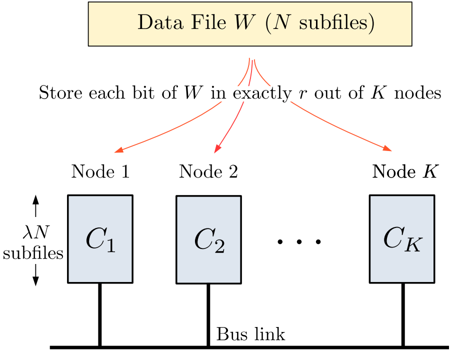


Figure: **An r-balanced distributed database $\mathcal{C}(r, [K])$, where $[K] = \{1, \ldots, K\}$.**

- 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, Krishnan et al (2020) [1]

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, Krishnan et al (2020) [1]

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

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

Optimality

Optimal communication load for node removal and node addition scenarios.

Previous Results

Main Result, Krishnan et al (2020) [1]

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

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

Optimality

Optimal communication load for node removal and node addition scenarios.

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," 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 2020, pp. 640-645, doi: 10.1109/ISIT44484.2020.9174482.

Cyclic Databases : Family of r-balanced Databases

Overcoming the large file-size requirement

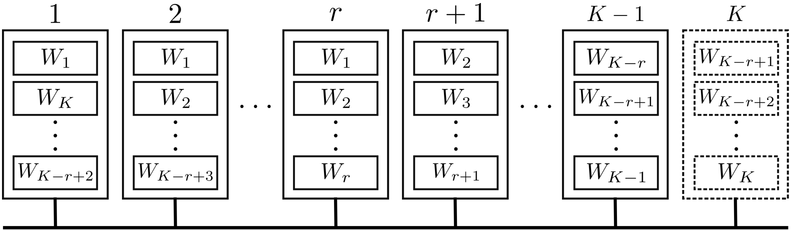


Figure: r-balanced cyclic database on nodes $[K]$

- 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

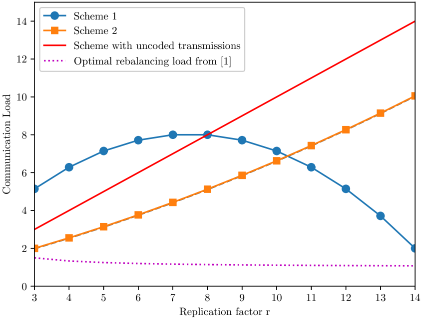


Figure: K=15, varying r

[1] P. Krishnan, V. Lalitha and L. Natarajan, "Coded Data Rebalancing: Fundamental Limits and Constructions," 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 2020, pp. 640-645, doi: 10.1109/ISIT44484.2020.9174482.

Table of Contents

1. Introduction

2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
   - Data Skew and Data Rebalancing in Distributed Systems
   - Coded Data Rebalancing : Formal System Model
   - Proposed Rebalancing Schemes for Cyclic Databases

3. A New Low Complexity Distributed Computing Scheme via Subspace Designs

4. Conclusions and Future Work

Initial and Final balanced databases

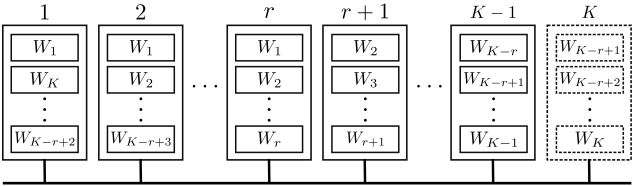


Figure: r-balanced cyclic database on nodes $[K]$

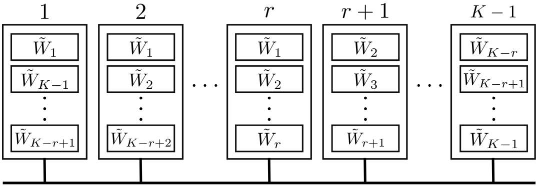


Figure: Target r-balanced cyclic database on nodes $[K-1]$

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.
- Making $|\tilde{S}_j \cap S_i|$ large reduces $|\tilde{S}_j \setminus S_i|$, which further reduces the movement of subsegments during rebalancing.
- 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 $W_i^{\tilde{S}_j \setminus S_i}$.

Splitting

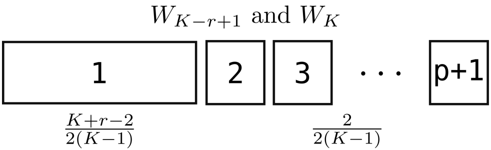


Figure: Splitting of the corner segments when $K - r$ is even. Here, $p = \lfloor \frac{K-r}{2} \rfloor$.

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

Splitting

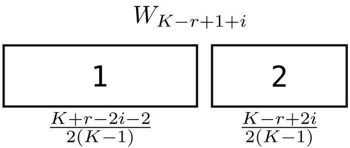


Figure: Splitting of the middle segments.

- 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
$W_3^{\{1\}}$	$\textcircled{s}$	—	*	*	*	*	$\circledast$
$W_4^{\{2\}}$	*	s	—	*	*	*	*
$W_5^{\{3\}}$	*	*	$\textcircled{s}$	—	*	*	$\circledast$
$W_6^{\{4\}}$	*	*	*	s	—	*	*
$W_7^{\{5\}}$	*	*	*	*	$\textcircled{s}$	—	$\circledast$
$W_4^{\{3\}}$	*	—	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 $\tilde{W}_j$.

Conclusion

- Rebalancing algorithm for cyclic databases
- Cubic file size requirement
- Communication Load strictly lower than the uncoded scheme
- Two schemes $\rightarrow$ two parameter regimes
- Similar techniques but one does better than the other in one regime and vice versa

Table of Contents

1. Introduction
2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
   - System Model for Coded MapReduce
   - Binary Matrices and Distributed Computing
   - Subspace Designs
   - Proposed Distributed Computing Scheme
   - Numerical Comparisons
4. Conclusions and Future Work

System Parameters

- K Servers: indexed by the set $\mathcal{K}$.
- File: divided into F subfiles, for $F \geq K$.
- Parameter F: known as the file complexity.
- F subfiles: indexed by the set $\mathcal{F}$.
- Computation Load r: the average number of nodes that map each subfile.
- Denote the set of subfiles assigned to node k ($k \in \mathcal{K}$) as $\mathcal{M}_k \subseteq \mathcal{F}$.
- Goal: Compute Q output functions on a file using K distributed computing nodes (servers).

Computing Functions

- The Q output functions are denoted as $\phi_1, \ldots, \phi_Q$. Each ϕ_q maps all the input files to a fixed length binary stream $u_q = \phi_q(\{\forall f \in \mathcal{F}\})$.
- The map function $g_{q,f}, \forall q \in [Q], \forall f \in \mathcal{F}$ maps the input subfile $f \in \mathcal{F}$ into Q length-T intermediate values (IVAs), denoted as $\{v_{1,f}, \ldots, v_{Q,f}\}$. Each $v_{q,f} \triangleq g_{q,f}(f),\ q \in [Q],\ f \in \mathcal{F}$ is an IVA corresponding to the subfile f and the q^{th} map function.
- The reduce function $h_q, q \in [Q]$ maps the IVAs $v_{q,f} : \forall f \in \mathcal{F}$ into the output value u_q. Thus, $u_q = \phi_q(\{\forall f \in \mathcal{F}\}) = h_q(\{v_{q,f} : \forall f \in \mathcal{F}\}) = h_q(\{g_{q,f}(f) : \forall f \in \mathcal{F}\})$.

Workflow

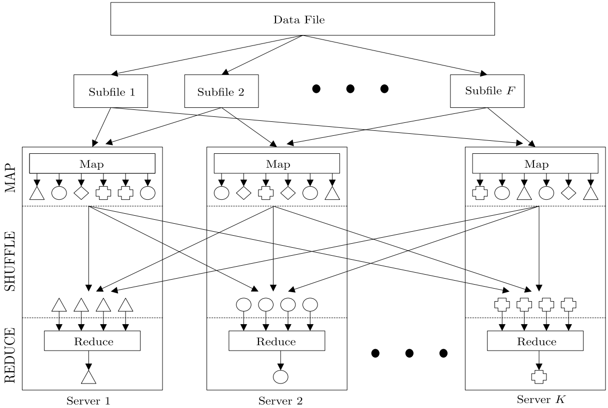


Figure: Workflow of a generic MapReduce framework on K servers.

Workflow

Map Phase

- Each server k uses the map functions to compute the IVAs of the subfiles in $\mathcal{M}_k$.
- Server k will have $\{v_{q,f} : \forall q \in [Q], \forall f \in \mathcal{M}_k\}$ after the map phase.

Workflow

Map Phase

- Each server k uses the map functions to compute the IVAs of the subfiles in $\mathcal{M}_k$.
- Server k will have $\{v_{q,f} : \forall q \in [Q], \forall f \in \mathcal{M}_k\}$ after the map phase.

Shuffle Phase

- Let $\mathcal{W}_k$ denote the indices of the functions to be reduced at server $k \in \mathcal{K}$.
- Server k requires $\{v_{q,f} : \forall q \in \mathcal{W}_k, \forall f \notin \mathcal{M}_k\}$.
- Servers send broadcast transmissions in order to fulfill the requirements of all the servers.

Workflow

Map Phase

- Each server k uses the map functions to compute the IVAs of the subfiles in $\mathcal{M}_k$.
- Server k will have $\{v_{q,f} : \forall q \in [Q], \forall f \in \mathcal{M}_k\}$ after the map phase.

Shuffle Phase

- Let $\mathcal{W}_k$ denote the indices of the functions to be reduced at server $k \in \mathcal{K}$.
- Server k requires $\{v_{q,f} : \forall q \in \mathcal{W}_k, \forall f \notin \mathcal{M}_k\}$.
- Servers send broadcast transmissions in order to fulfill the requirements of all the servers.

Reduce Phase

- Each server k computes $h_q(\{v_{q,f} : \forall f \in \mathcal{F}\})$ for each $q \in \mathcal{W}_k$.
- This results in computing the value of the $\phi_q : \forall q \in \mathcal{W}_k$ on the input file.

Communication Load

Communication Load

Let T be the size of each IVA in bits. The communication load, denoted by L, $0 \leq L \leq 1$, is defined as the (normalized) total number of bits communicated by the K computing nodes during the Shuffle phase and can be calculated using the following.

$$L \triangleq \frac{\text{Total number of bits transmitted in shuffle phase}}{QFT}.$$

Communication Load

Communication Load

Let T be the size of each IVA in bits. The communication load, denoted by L, $0 \leq L \leq 1$, is defined as the (normalized) total number of bits communicated by the K computing nodes during the Shuffle phase and can be calculated using the following.

$$L \triangleq \frac{\text{Total number of bits transmitted in shuffle phase}}{QFT}.$$

Relationship with r

- As r increases, L decreases and vice versa
- Reason: Coding opportunities increase as r increases

Previous Work

Uncoded Scheme

- Total IVAs needed across K nodes $= QFT$.
- Available IVAs after Map Phase $= rF \cdot \frac{Q}{K} = \frac{rQF}{K}$.

$$L_{\text{uncoded}} = \frac{(QFT - \frac{rQFT}{K})}{QFT} = 1 - \frac{r}{K}.$$

Previous Work

Uncoded Scheme

- Total IVAs needed across K nodes $= QFT$.
- Available IVAs after Map Phase $= rF \cdot \frac{Q}{K} = \frac{rQF}{K}$.

$$L_{\text{uncoded}} = \frac{(QFT - \frac{rQFT}{K})}{QFT} = 1 - \frac{r}{K}.$$

Optimal Scheme, Li et al (2018) [2]

- Careful mapping of the subfiles at r distinct nodes to enable maximal coding opportunities.

$$L^* = \frac{1}{r}(1 - \frac{r}{K}).$$

- Advantage: Multiplicative gain equal to r
- Drawback: File complexity F required to be exponential in K.

[2] S. Li, M. A. Maddah-Ali, Q. Yu and A. S. Avestimehr, "A Fundamental Tradeoff Between Computation and Communication in Distributed Computing," in IEEE Transactions on Information Theory, vol. 64, no. 1, pp. 109-128, Jan. 2018, doi: 10.1109/TIT.2017.2756959.

Table of Contents

1 Introduction

2 Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage

3 A New Low Complexity Distributed Computing Scheme via Subspace Designs
- System Model for Coded MapReduce
- Binary Matrices and Distributed Computing
- Subspace Designs
- Proposed Distributed Computing Scheme
- Numerical Comparisons

4 Conclusions and Future Work

Binary Computing Matrix

Binary Computing Matrix, Agrawal et al (2020) [3]

$$C = \begin{array}{c} \\ 1 \\ \\ K \end{array} \begin{array}{c} \begin{array}{ccc} 1 & & F \end{array} \\ \begin{pmatrix} 0 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 0 \end{pmatrix} \end{array}.$$

- Server $k \in \mathcal{K}$ maps subfile $f : \forall f \in \mathcal{F}$ if $C(k, f) = 0$ and does not map it if $C(k, f) = 1$.
- The number of 0s in any column is constant and is equal to r (computation load)

[3] S. Agrawal and P. Krishnan, "Low Complexity Distributed Computing via Binary Matrices with Extension to Stragglers," 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 2020, pp. 162-167, doi: 10.1109/ISIT44484.2020.9174080.

Previous Results

Main Result

Consider a computing matrix C of size $K \times F$ with a non-overlapping identity submatrix cover $\mathfrak{C} = \{C_1, C_2, .., C_S\}$ where the size of each identity submatrix is $g \geq 2$. Then, there exists a distributed computing scheme with K nodes, attaining computation load r and communication load $L = \frac{2}{g}\left(1 - \frac{r}{K}\right)$, with file complexity F.

Corollary

For any positive integers K and $r \in [K]$, there exists a $(K, \binom{K}{r}, r)$-computing matrix, from which we get a distributed computing scheme on K nodes with computation load r and communication load $L = \frac{2}{r+1}\left(1 - \frac{r}{K}\right)$, with file complexity $F = \binom{K}{r}$. Further, this load $L < 2L^*(r)$, where $L^*(r)$ is the optimal rate for a given computation load r.

Example - Scheme via Binary Matrix

Consider a set system $(\mathcal{K}, \mathcal{F})$ given by

$$\mathcal{K} = \{1,2,3,4,5,6,7\}$$
$$\mathcal{F} = \{127,145,136,467,256,357,234\}.$$

The incidence matrix C for this set system is

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 127 & 145 & 136 & 467 & 256 & 357 & 234 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

We can see that the above matrix is a $(7,7,4)$-computing matrix.

Example

	127	145	136	467	256	357	234
1	1	1	1	0	0	0	0
2	1	0	0	0	1	0	1
3	0	0	1	0	0	1	1
4	0	1	0	1	0	0	1
5	0	1	0	0	1	1	0
6	0	0	1	1	1	0	0
7	1	0	0	1	0	1	0

$$C = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Figure: The identity submatrices (using 7 different shapes), each of size 3, of the above matrix form an identity submatrix cover, which consists of 7 non-overlapping identity submatrices.

Example

Consider the identity submatrix denoted as C_1, where

$$C_1 = \begin{array}{c} \\ 1 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 145 & 256 & 357 \end{array} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right] \end{array}.$$

- Servers: row indices $\{1, 6, 7\}$.
- Subfiles: column indices $\{145, 256, 357\}$.
- C_1 corresponds to one round of transmission.
- One round of transmission has one coded (by server 1) and one uncoded transmission (by server 6).

Example

$$C_1 = \begin{array}{c} \\ 1 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 145 & 256 & 357 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}.$$

- Let $Q = 14$ (i.e., $\beta = 2$) and $\mathcal{W}_1 = \{1, 8\}$, $\mathcal{W}_6 = \{6, 13\}$, $\mathcal{W}_7 = \{7, 14\}$.
- Missing IVAs at:
  - Server 1: $\{v_{1,145},\ v_{8,145}\}$
  - Server 6: $\{v_{6,256},\ v_{13,256}\}$
  - Server 7: $\{v_{7,357},\ v_{14,357}\}$
- Coded transmission sent by server 1: $\{v_{7,357} \oplus v_{6,256},\ v_{14,357} \oplus v_{13,256}\} \rightarrow$ decoded at server 6 and server 7 as required.
- Uncoded transmission by server 6: $\{v_{1,145},\ v_{8,145}\} \rightarrow$ received by server 1.

Straggler Scenario

- Straggler: nodes that are slower than the other nodes.
- Full Straggler:
  - Nodes that are unable to complete any map tasks completely.
  - Considered as failed nodes.
  - For $K - \kappa \in [0 : g-2]$ full stragglers, $L(\kappa) = \frac{2}{g}\left(\frac{K}{\kappa} - \frac{r}{\kappa}\right)$.
- Partial Straggler:
  - Nodes that are slower than the other nodes by some factor.
  - Not considered as failed nodes.
  - For $K - \kappa' \in [0 : g-2]$ partial stragglers, $L(\kappa') = \frac{2}{g}\left(1 - \frac{r}{K}\right)$.
- Optimal Scheme, Yan et al (2020) [4]

$$L^*(\kappa) = \left(1 - \frac{r}{K}\right) \sum_{i=r+\kappa-K}^{min\{r,\kappa-1\}} \frac{1}{i} \frac{\binom{r}{i}\binom{K-r-1}{\kappa-i-1}}{\binom{K-1}{\kappa-1}}, K - \kappa \leq r - 1$$

[4] Q. Yan, M. Wigger, S. Yang and X. Tang, "A Fundamental Storage-Communication Tradeoff for Distributed Computing With Straggling Nodes," in IEEE Transactions on Communications, vol. 68, no. 12, pp. 7311-7327, Dec. 2020, doi: 10.1109/TCOMM.2020.3020549.

Table of Contents

1 Introduction

2 Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage

3 A New Low Complexity Distributed Computing Scheme via Subspace Designs
- System Model for Coded MapReduce
- Binary Matrices and Distributed Computing
- Subspace Designs
- Proposed Distributed Computing Scheme
- Numerical Comparisons

4 Conclusions and Future Work

Combinatorial Designs

Design $(\mathcal{X}, \mathcal{A})$

A design is a pair $(\mathcal{X}, \mathcal{A})$ with the following properties:

- $\mathcal{X}$ is a set of elements called points.
- $\mathcal{A}$ is a collection (i.e., multiset) of nonempty subsets of $\mathcal{X}$ called blocks.

Combinatorial Designs

Design $(\mathcal{X}, \mathcal{A})$

A design is a pair $(\mathcal{X}, \mathcal{A})$ with the following properties:

- $\mathcal{X}$ is a set of elements called points.
- $\mathcal{A}$ is a collection (i.e., multiset) of nonempty subsets of $\mathcal{X}$ called blocks.

***t*-designs**

For $v, k, \lambda, t \in \mathbb{Z}^+$ such that $v > k \geq t$. A t-(v, k, λ)-design (or simply t-design) is a design $(\mathcal{X}, \mathcal{A})$ with the following properties:

- $|\mathcal{X}| = v$.
- Each block contains exactly k points.
- Every set of t distinct points is contained in exactly λ blocks.

Subspace Designs

Subspace Designs

Let $\mathcal{V}$ be a vector space over the finite field $\mathbb{F}_q$ of dimension v. For $v, k, \lambda, t \in \mathbb{Z}^{0+}$ such that $t \leq k \leq v$, a pair $\mathcal{D} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{A}$ is a collection of k-dimensional subspaces (blocks) of $\mathcal{V}$, is called a t-$(v, k, \lambda)_q$-subspace design on $\mathcal{V}$ if each t-dim subspace of $\mathcal{V}$ is contained in exactly λ blocks.

Note: A subspace design is also referred to as a q-analog of an equivalent t-design.

Table of Contents

1. Introduction
2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
   - System Model for Coded MapReduce
   - Binary Matrices and Distributed Computing
   - Subspace Designs
   - Proposed Distributed Computing Scheme
   - Numerical Comparisons
4. Conclusions and Future Work

Some Definitions

- Let $(\mathcal{V}, \mathcal{A})$ denote a t-$(v, k, 1)_q$-subspace design.
- Let $\mathcal{A} = \{B_1, \ldots, B_b\}$ be the set of blocks.
- $T \triangleq$ set of all 1-dim subspaces of $\mathcal{V}$.
- $H \triangleq$ set of all t-dim subspaces of $\mathcal{V}$.
- $R \triangleq$ set of all $(t-1)$-dim subspaces of $\mathcal{V}$.
- $\begin{bmatrix} v \\ k \end{bmatrix}_q \triangleq$ the number of subspaces of dimension k in any v dimensional vector space over $\mathbb{F}_q$, the finite field with q elements.

Binary Matrix Construction

Binary Matrix C

- Rows: indexed by set R
- Columns: indexed by $\{(y, B) : y \in T, y \subset B, B \in \mathcal{A}\}$.
- Number of Rows $= \begin{bmatrix} v \\ t-1 \end{bmatrix}_q$, Number of Columns $= b \begin{bmatrix} k \\ 1 \end{bmatrix}_q = \dfrac{\begin{bmatrix} v \\ t \end{bmatrix}_q \begin{bmatrix} k \\ 1 \end{bmatrix}_q}{\begin{bmatrix} k \\ t \end{bmatrix}_q}$
- For some $D \in R$, the matrix $C = (C(D, (y, B)))$ is defined by the rule,

$$C(D, (y, B)) = \begin{cases} 1, & \text{if } D \bigoplus y \in H, D \bigoplus y \subset B \\ 0, & \text{otherwise.} \end{cases}$$

- Matrix C is a constant column weight matrix (required for distributed computing).
- Claim: Matrix C as defined above leads to a coded distributed computing scheme.

Proof format

- Design a method to pick a submatrix of C.
- Show that the submatrix is an identity submatrix as follows:
  - Square matrix
  - Row and column weight equal to 1.
- Show that the submatrices don't overlap.
- Show that all the 1's in C are covered $\Rightarrow$ identity submatrix cover.

Parameters

- Number of servers $K = \begin{bmatrix} v \\ t-1 \end{bmatrix}_q$
- File Complexity $F = \dfrac{\begin{bmatrix} v \\ t \end{bmatrix}_q \begin{bmatrix} k \\ 1 \end{bmatrix}_q}{\begin{bmatrix} k \\ t \end{bmatrix}_q}$
- Computation Load $r = \begin{bmatrix} v \\ t-1 \end{bmatrix}_q - \begin{bmatrix} k-1 \\ t-1 \end{bmatrix}_q q^{t-1}$
- Communication Load for non/partial straggler case $= \dfrac{2\begin{bmatrix} k-1 \\ t-1 \end{bmatrix}_q^2 q^{t-1}}{\begin{bmatrix} v \\ t-1 \end{bmatrix}_q \begin{bmatrix} v-1 \\ t-1 \end{bmatrix}_q}$
- Communication Load for $K-\kappa$ full straggler case $= \dfrac{2\begin{bmatrix} k-1 \\ t-1 \end{bmatrix}_q^2 q^{t-1}}{\kappa \begin{bmatrix} v-1 \\ t-1 \end{bmatrix}_q}$

Table of Contents

1. Introduction

2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage

3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
   - System Model for Coded MapReduce
   - Binary Matrices and Distributed Computing
   - Subspace Designs
   - Proposed Distributed Computing Scheme
   - Numerical Comparisons

4. Conclusions and Future Work

Numerical Comparisons

$t-(v,k,\lambda)_q$	K	F	r	L for non/partial straggler case	L for $K-\kappa=1$	L for $K-\kappa=2$
$2-(3,2,1)_2$	7	21	5	0.19	0.22	0.27
$4-(5,4,1)_2$	155	465	147	0.00688	0.00693	0.00697
$3-(4,3,1)_3$	130	520	121	0.01065	0.01073	0.01082
$4-(5,4,1)_3$	1210	4840	1183	0.0011157	0.0011166	0.0011175

Table: Numerical comparisons of communication loads of our schemes in non/partial straggler case and straggler case.

Numerical Comparisons

K	r	F	F in [4]	κ	Load in this work	Optimal Load in [4]
13	10	52	286	13	0.115	0.023
13	10	52	286	11	0.136	0.028
130	121	520	2.2×10^{13}	130	0.0106	0.00057
130	121	520	2.2×10^{13}	128	0.0108	0.00058
1210	1183	4840	1.18×10^{55}	1210	0.001115	1.887×10^{-5}
1210	1183	4840	1.18×10^{55}	1208	0.001117	1.889×10^{-5}

Table: Numerical comparisons between the scheme from Yan et al (2020) [4] and subspace designs based computing schemes presented in this work.

[4] Q. Yan, M. Wigger, S. Yang and X. Tang, "A Fundamental Storage-Communication Tradeoff for Distributed Computing With Straggling Nodes," in IEEE Transactions on Communications, vol. 68, no. 12, pp. 7311-7327, Dec. 2020, doi: 10.1109/TCOMM.2020.3020549.

Table of Contents

1. Introduction
2. Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage
3. A New Low Complexity Distributed Computing Scheme via Subspace Designs
4. Conclusions and Future Work

Conclusions and Future work

Conclusions

- Coded Data Rebalancing
  - Framework for Coded Rebalancing for handling data skew in cyclic databases with an improved file-size requirement.
  - Communication Load strictly lesser than the uncoded scheme.
- Distributed Computing
  - A low complexity distributed computing scheme via subspace designs.
  - Marginal increase in the communication load as compared to the optimal scheme.

Conclusions and Future Work

Future work

- Coded Data Rebalancing
  - Multiple simultaneous node removals or additions in case of cyclic databases.
  - Constructing good converse arguments in the cyclic database setting.
- Distributed Computing
  - Primarily useful for the large local storage scenario $\Rightarrow$ constructing schemes for lower local storage.
  - Considering wider classes of subspace designs (i.e., $\lambda > 1$).

Related Publications

Conferences

Athreya Chandramouli*, Abhinav Vaishya*, Prasad Krishnan. "Coded data rebalancing for distributed data storage systems with cyclic storage." In 2022 IEEE Information Theory Workshop (ITW), pp. 618-623. IEEE, 2022.

Journals (Under Review)

Shailja Agrawal, K V Sushena Sree, Prasad Krishnan, Abhinav Vaishya, Srikar Kale. "Cache-Aided Communication Schemes via Combinatorial Designs and their q-analogs." arXiv preprint arXiv:2302.03452 (2023). [Submitted to IEEE Journal on Selected Areas in Information Theory (JSAIT), 2023.]

Preprints

Athreya Chandramouli*, Abhinav Vaishya*, and Prasad Krishnan. "Coded Data Rebalancing for Distributed Data Storage Systems with Cyclic Storage." arXiv preprint arXiv:2205.06257 (2022).

Thank You!