CODED DATA REBALANCING

1 Data Skew and Data Rebalancing in Distributed Systems

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
Redistribute data across the available nodes to balance the distribution.

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

Introducing Coded Data Rebalancing
- Exploit data replication for Coded transmissions during rebalancing
- Preserve database structure post rebalancing.

2 Coded Data Rebalancing : Formal System Model

System Model: Initial database

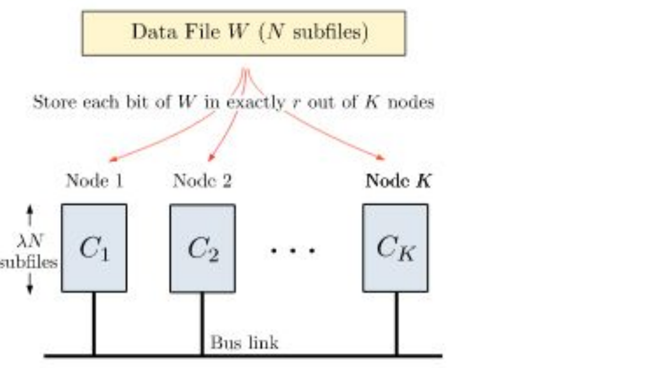
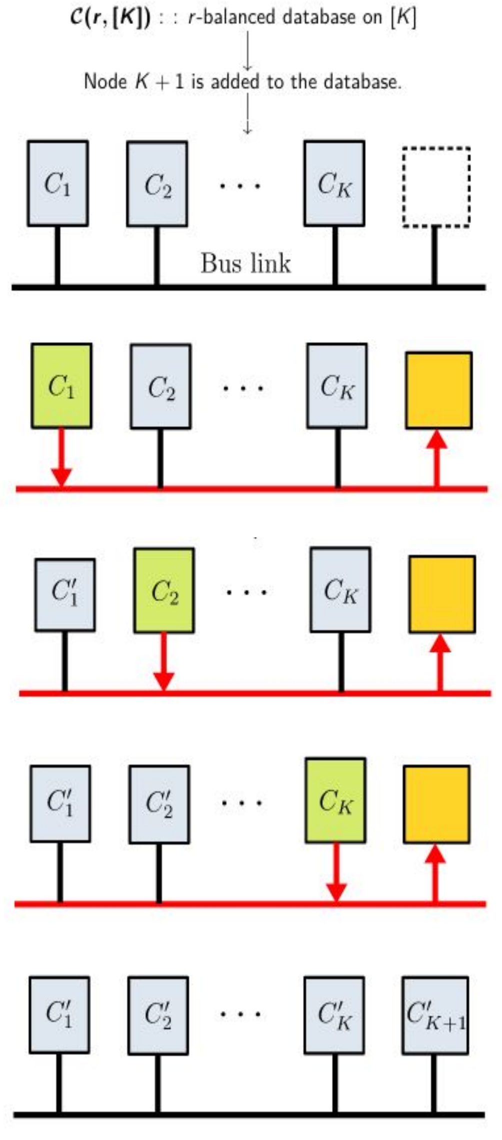


Figure: **An r-balanced distributed database $\mathcal{C}(r,[K])$, where $[K] = \{1,\ldots,K\}$.**

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

Rebalancing Scheme for Node Addition $\mathcal{R}'_{\mathcal{C},\mathcal{C}'}$

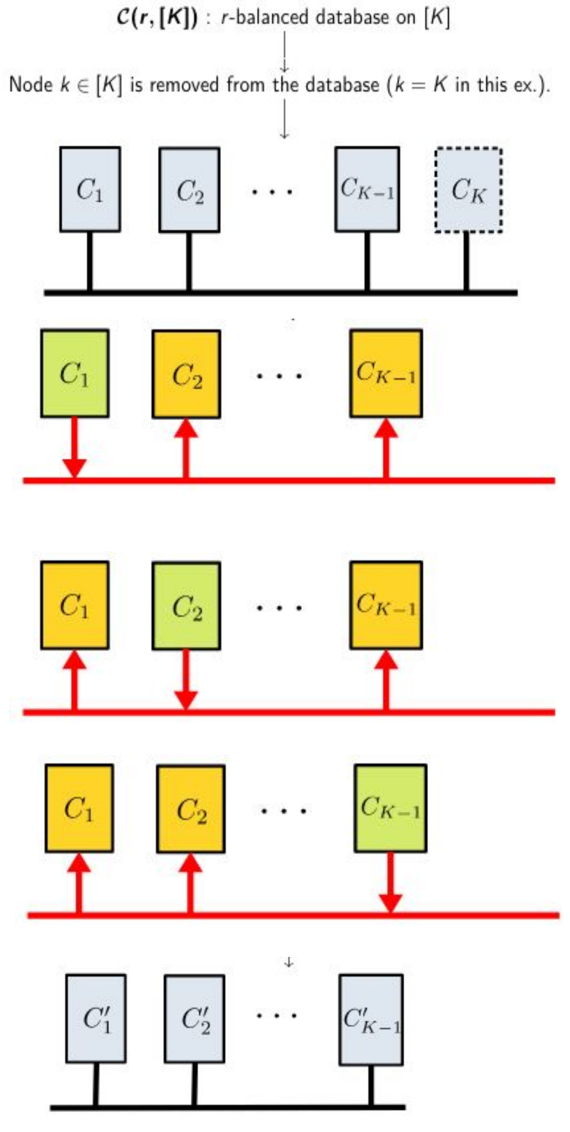
$\mathcal{C}(r,[K])$: r-balanced database on $[K]$



$\mathcal{C}'(r,[K+1])$: r-balanced database on $[K+1]$

Rebalancing Scheme for Node $k \in [K]$ Removal, $\mathcal{R}_{k,\mathcal{C},\mathcal{C}_k}$

$\mathcal{C}(r,[K])$: r-balanced database on $[K]$



$\mathcal{C}_k(r,[K]\backslash k)$: r-balanced database on $[K]\backslash k$

Communication Load of Rebalancing Scheme

Let l_i be the number of bits sent by node i.

Node addition:

$$L_{add}(\mathcal{R}'_{\mathcal{C},\mathcal{C}'}) \triangleq \frac{\sum_{i\in[K]} l_i}{\lambda_{add} N}, \quad \text{where } \lambda_{add} = \frac{r}{K+1}.$$

Node Removal:

$$L_{rem}(\mathcal{R}_{k,\mathcal{C},\mathcal{C}_k}) \triangleq \frac{\sum_{i\in[K]\backslash k} l_i}{\lambda N}, \quad \text{where } \lambda_{rem} = \frac{r}{K-1}.$$

$$L^*(r) = \inf_{\mathcal{C},\{\mathcal{C}_k:k\in[K]\},\mathcal{C}'} \ \inf_{\mathcal{R}_{k,\mathcal{C},\mathcal{C}_k},\mathcal{R}'_{\mathcal{C},\mathcal{C}'}} \ \max_{k\in[K]} L_{rem}(\mathcal{R}_{k,\mathcal{C},\mathcal{C}_k}) + L_{add}(\mathcal{R}'_{\mathcal{C},\mathcal{C}'})$$

Main Contribution

Converse

$$L^* \geq \frac{1}{r-1} + 1 \quad \text{(node removal + node addition)}.$$

Achievable Scheme
- Achieves optimal communication load (whereas $L_{uncoded} = 1 + 1$)
- Optimality for any sequence of node removals or additions
- Structural invariance: Maintains the structure of the database across node removal/addition.

3 Achievability

Achievable Scheme : Family of r-balanced Databases

- Divide the data W into $\frac{K!}{r!}$ subfiles.
- Index set of subfiles

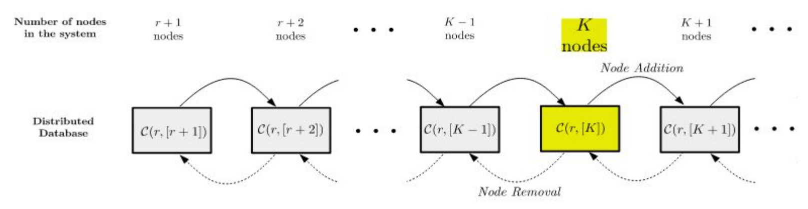
$$S([K], K-r) \triangleq \{\boldsymbol{i} = (i_1,\ldots,i_{K-r}) : \{i_1,\ldots,i_{K-r}\} \subseteq [K]\}$$

r-balanced Distributed Database
For each $\boldsymbol{i} = (i_1,\ldots,i_{K-r}) \in S([K],K-r)$

$$W_{\boldsymbol{i}} \text{ is stored in } \{j_1,\ldots,j_r\}, \quad \text{where } \{j_1,\ldots,j_r\} = [K]\backslash \boldsymbol{i}.$$

Easy to check this is r-balanced database. We denote this as $\mathcal{C}(r,[K])$

Illustration of Rebalancing and Structural Invariance



Achievable Scheme : Node Addition

- Starting database $\mathcal{C}(r,[K])$
- Subfile indices: $S([K], K-r)$

- Target Database $\mathcal{C}(r,[K+1])$ (keep r replication, balance node storage)
- Subfile indices $S([K+1], K+1-r)$

For each subfile $\boldsymbol{i} = (i_1,\ldots,i_{K-r}) \in S([K],K-r)$ (where $\{j_1,\ldots,j_r\} = [K]\backslash\boldsymbol{i}$)
- Split $W_{\boldsymbol{i}}$ into $K+1$ subfiles, re-indexed as

$$(j_1,i_1,\ldots,i_{K-r}),\ (j_2,i_1,\ldots,i_{K-r}),\ \ldots,\ (j_r,i_1,\ldots,i_{K-r})$$
$$(K+1,i_1,\ldots,i_{K-r}),\ (i_1,K+1,i_2,\ldots,i_{K-r}),\ \ldots,\ (i_1,\ldots,i_{K-r},K+1)$$

- Note that these new indices are in $S([K+1], K+1-r)$.
- Node j_l forwards $(j_l,i_1,\ldots,i_{K-r})$ to $K+1$, and then deletes it.

Example

$K=5, r=3$, with database $\mathcal{C}([5],3)$.
Node 6 is added.
- Subfile Splitting: $W_{[2\ 3]}$ is split into $K+1=6$ chunks at $\{1,4,5\}$ as

$$W_{[1\ 2\ 3]}, W_{[4\ 2\ 3]}, W_{[5\ 2\ 3]},$$
$$W_{[6\ 2\ 3]}, W_{[2\ 6\ 3]}, W_{[2\ 3\ 6]}$$

- Transmissions and Deletion:
  - Node 1 transfers $W_{[1\ 2\ 3]}$ to Node 6 and deletes it.
  - Node 4 transfers $W_{[4\ 2\ 3]}$ to Node 6 and deletes it.
  - Node 5 transfers $W_{[5\ 2\ 3]}$ to Node 6 and deletes it.

Same is done for each subfile.

Final database on 6 nodes: $\mathcal{C}([6],3)$.

Achievable Scheme : Node $k \in [K]$ Removal

- Starting database $\mathcal{C}(r,[K])$
- Subfile indices: $S([K], K-r)$

- Target Database $\mathcal{C}(r,\mathcal{K}')$, where $\mathcal{K}'$ denotes the survivor set $[K]\backslash k$.
- Rebalance to reinstate replication factor of subfiles in node k, and balance storage.
- Subfile indices $S' = S(\mathcal{K}', K-1-r)$

For subfile $\boldsymbol{i} \in S'$ (where $\{j_1,\ldots,j_r\} = \mathcal{K}'\backslash\boldsymbol{i}$)
- Consider the r subfile indices,

$$(j_1,\boldsymbol{i}),\ (j_2,\boldsymbol{i}),\ \ldots,\ (j_r,\boldsymbol{i})$$

- Note that these indices are in $S([K],K-r)$, representing existing subfiles that were stored in node k.
- These have replication factor reduced to $r-1$, must be reinstated to r.

Achievable Scheme : Node $k \in [K]$ Removal (Step 1 - Grouping and Transmissions)

For each $\boldsymbol{i} \in S'$ (where $\{j_1,\ldots,j_r\} = \mathcal{K}'\backslash\boldsymbol{i}$)
- Subfile Grouping: Consider the r subfiles indices,

$$(j_1,\boldsymbol{i}),\ (j_2,\boldsymbol{i}),\ \ldots,\ (j_r,\boldsymbol{i})$$

- Only subfile $W_{(j_l,\boldsymbol{i})}$ is unavailable at j_l, available at $\{j_1,\ldots,j_r\}\backslash j_l$.
- Subfile Exchange via Coded Transmissions: Each j_l does one coded transmission of size $\frac{1}{r-1}$ of that of a subfile

$$\bigoplus_{m\in[r]\backslash l} W^l_{(j_m,\boldsymbol{i})}, \text{(for example, } W^1_{(j_2,\boldsymbol{i})} \oplus \ldots \oplus W^1_{(j_r,\boldsymbol{i})} \text{by node } j_1)$$

where $W^l_{(j_m,\boldsymbol{i})}$ represents one chunk of $(r-1)$ chunks of subfile $W_{(j_m,\boldsymbol{i})}$.
- After these r transmissions, $(j_l,\boldsymbol{i})$ is available at node j_l also.

Achievable Scheme : Node $k \in [K]$ Removal (Step 2 - Recombining)

For each $\boldsymbol{i} \in S'$ (where $\{j_1,\ldots,j_r\} = \mathcal{K}'\backslash\boldsymbol{i}$)
- Each node $\{j_1,\ldots,j_r\}$ has following subfiles

$$(j_1,i_1,\ldots,i_{K-r-1}),\ (j_2,i_1,\ldots,i_{K-r-1}),\ \ldots,\ (j_r,i_1,\ldots,i_{K-r-1})$$
$$(k,i_1,\ldots,i_{K-r-1}),\ (i_1,k,i_2,\ldots,i_{K-r-1}),\ \ldots,\ (i_1,\ldots,i_{K-r-1},k)$$

- Combine the above K subfiles and create a new (larger) subfile with index $\boldsymbol{i} \in S'$.

Example

$K=5, r=3$, with database $\mathcal{C}([5],3)$.
Node 5 is removed.
- Subfile Grouping: Form a group with $\boldsymbol{i} = [3] \in S([4],1)$

$$W_{[1\ 3]}, W_{[2\ 3]}, W_{[4\ 3]}$$

- Subfile Exchange through Coded transmissions
  - Node 1 transmits $W^1_{[2\ 3]} \oplus W^1_{[4\ 3]}$.
  - Node 2 broadcasts $W^2_{[1\ 3]} \oplus W^2_{[4\ 3]}$.
  - Node 4 broadcasts $W^4_{[1\ 3]} \oplus W^4_{[2\ 3]}$.
- Combine the following at nodes $\{1,2,4\}$, and relabel as $W_{[3]}$

$$W_{[1\ 3]}, W_{[2\ 3]}, W_{[4\ 3]},$$
$$W_{[5\ 3]}, W_{[3\ 5]}$$

Same is done for each $\boldsymbol{i} \in S([4],1)$.
Final database on 4 nodes: $\mathcal{C}([4],3)$.

Dr. Prasad Krishnan
https://arxiv.org/abs/2001.04939
Research Center Name: SPCRC