ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

MADSYS
THU•HPC

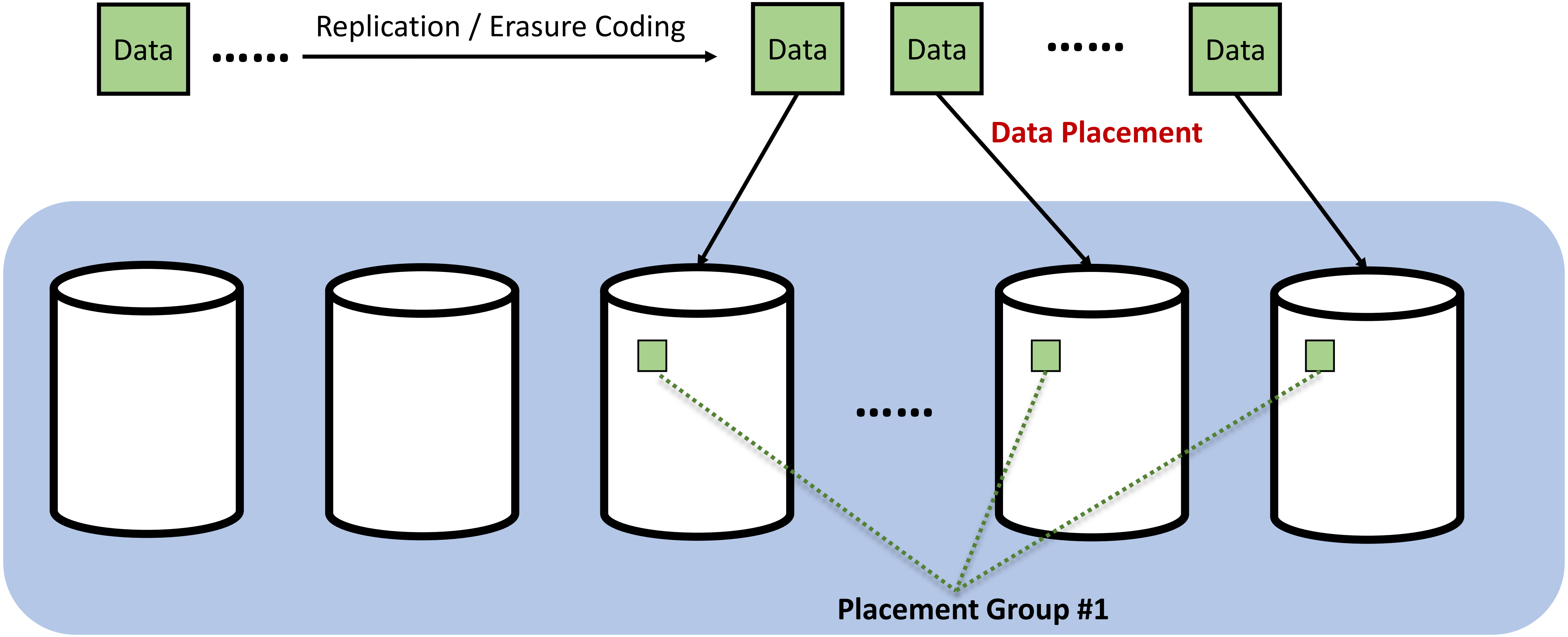
Explore Data Placement Algorithm for Balanced Recovery Load Distribution

Yingdi Shan[1,2], Kang Chen[2], Yongwei Wu[2]

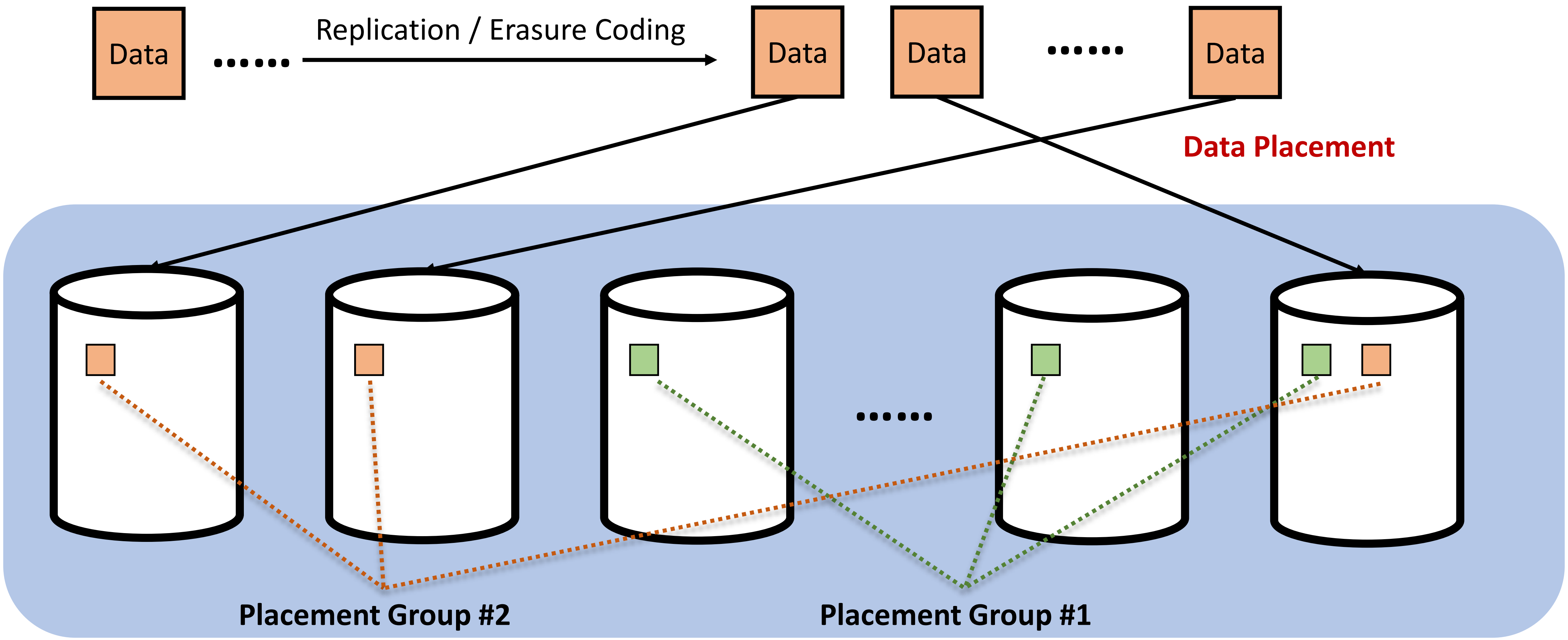
[1]Zhongguancun Laboratory [2]Tsinghua University

Background - Distributed Storage Systems

Distributed Storage Systems: HDFS, Ceph, GFS, RAMCloud, etc.

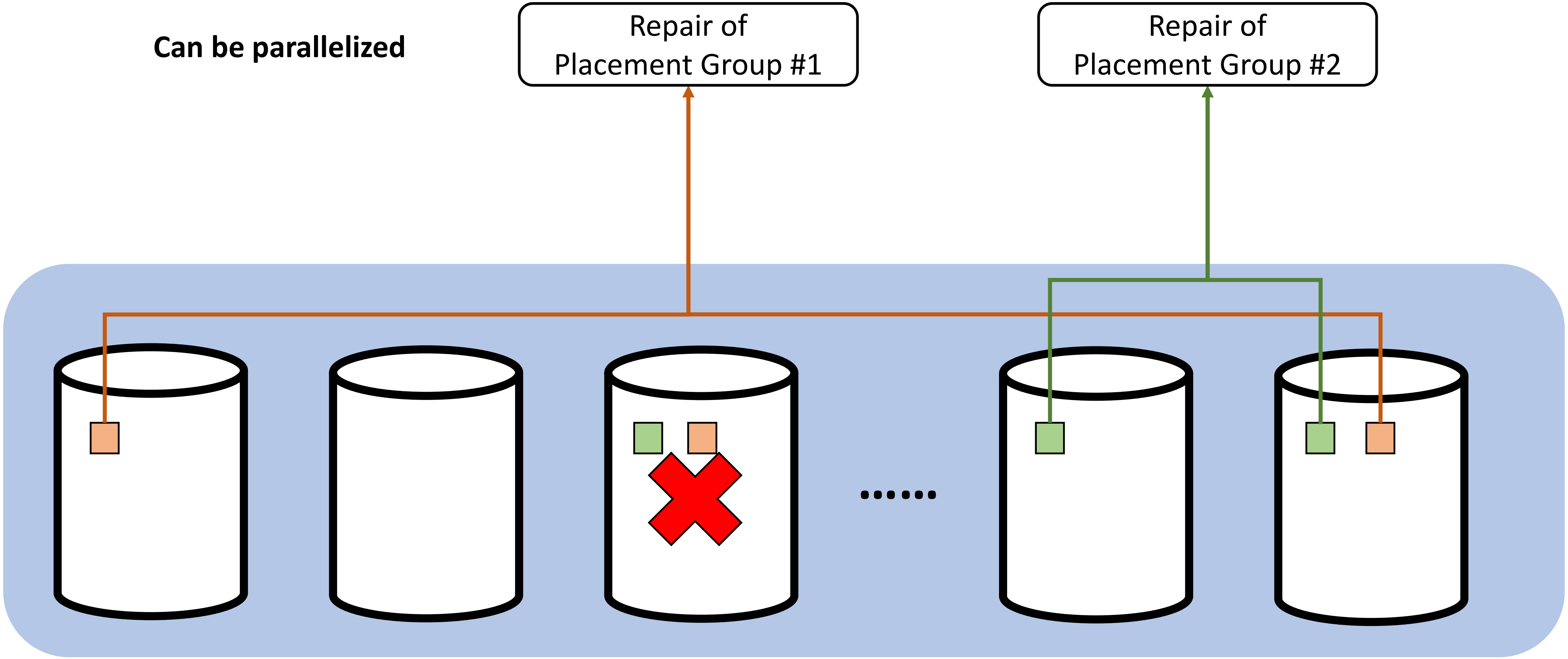


Background - Distributed Storage Systems

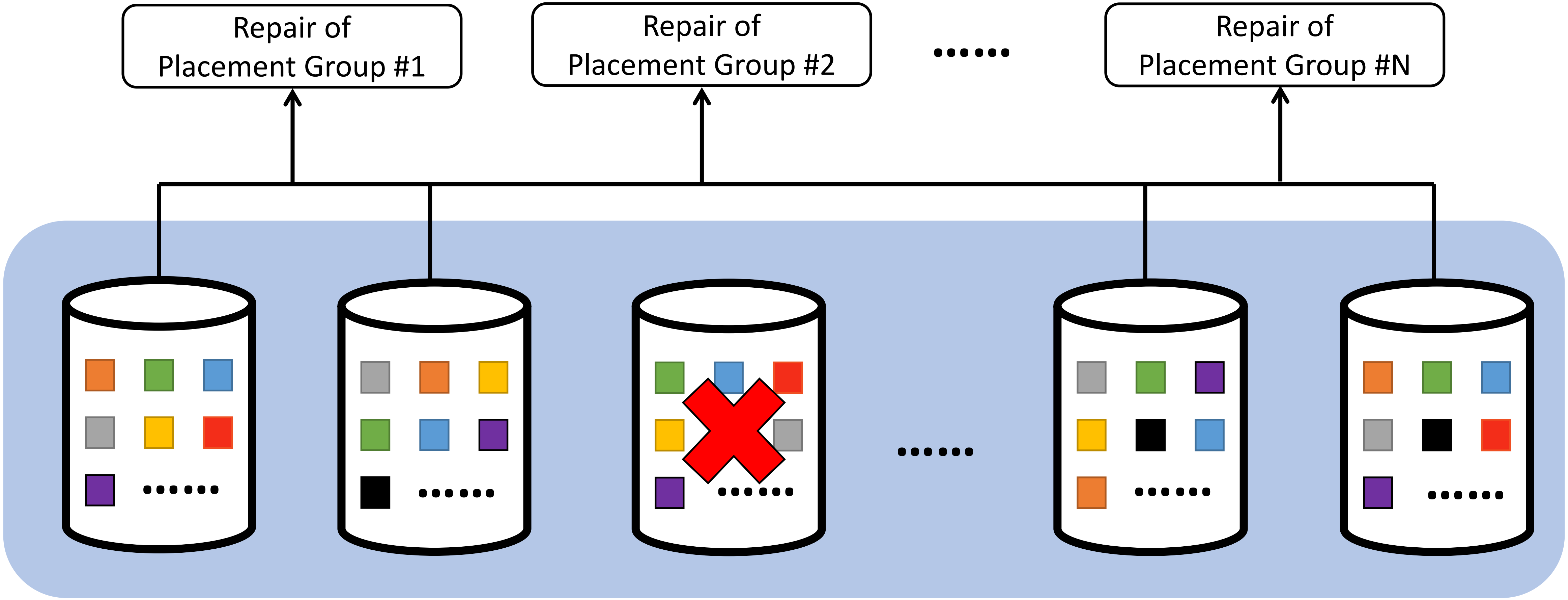


Background - Recovery

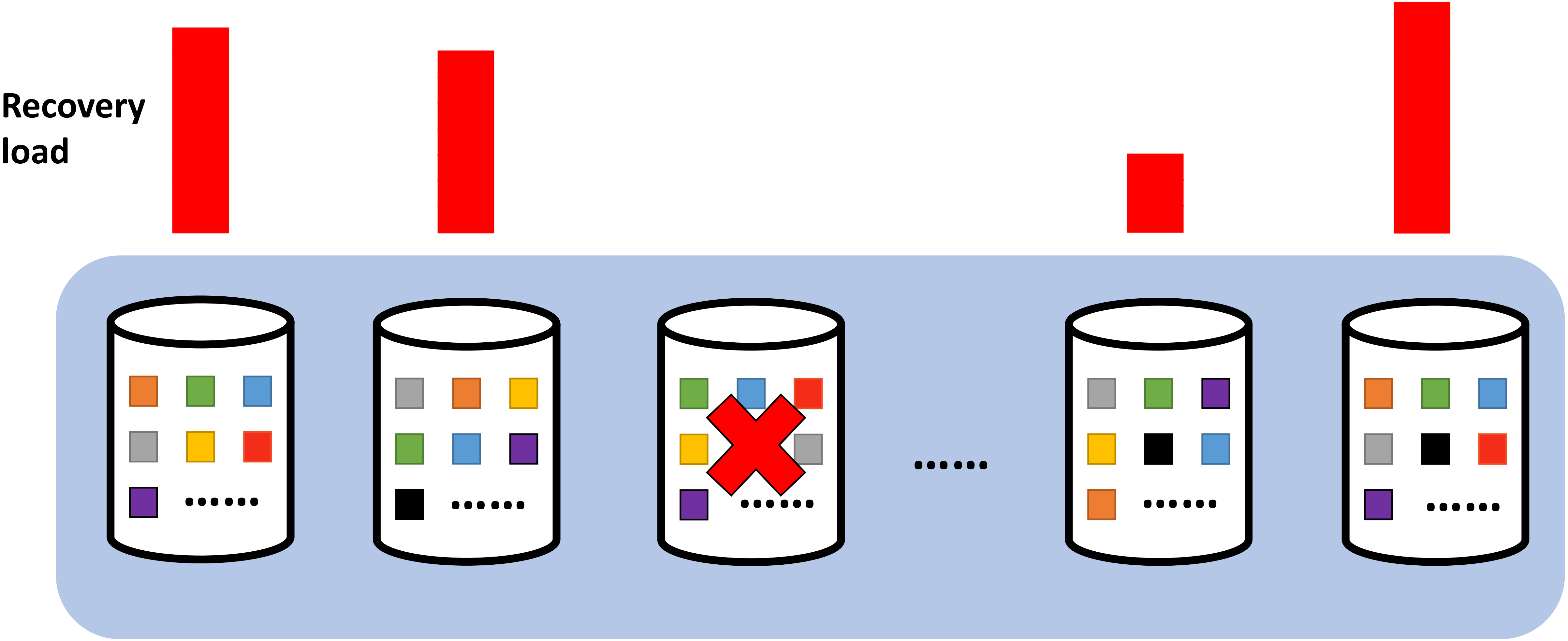
Can be parallelized



Parallelized Recovery


Repair of Placement Group #1
Repair of Placement Group #2
……
Repair of Placement Group #N
……

Recovery Load Imbalance



Recovery load can be imbalanced.

Fine-Grained Data Units

- A simple solution to load imbalance is by using **fine-grained** data units.

- Recovery load can be more balanced through **randomization** by distributing a sufficient number of data units across various nodes.

- However, fine-grained data units can increase **the probability of data loss**. (e.g. many combinations of simultaneous failures can cause data loss vs. only specific combinations can cause data loss)

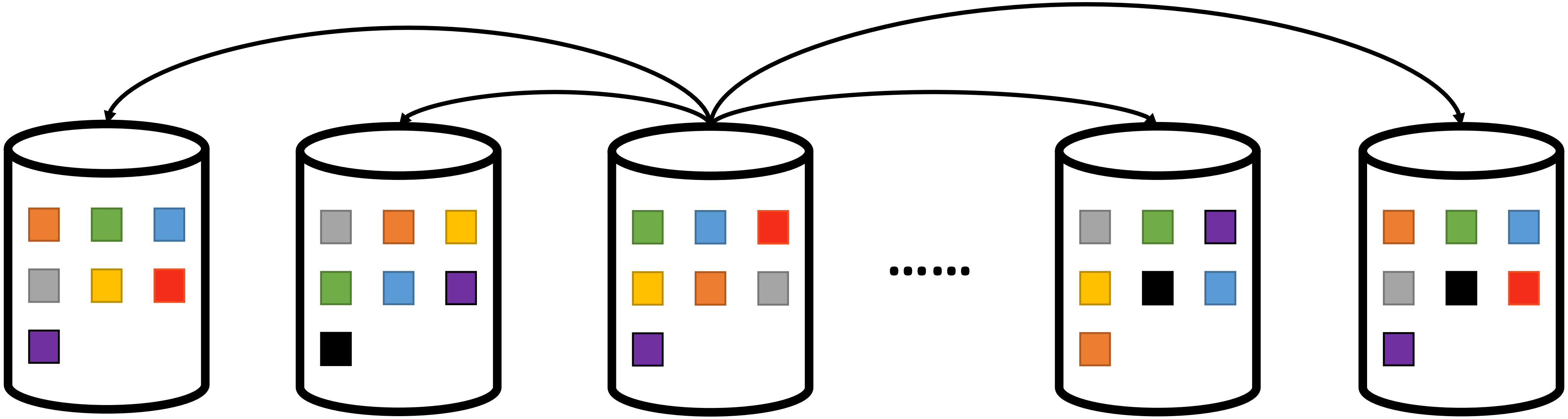
Data Placement Algorithm for Recovery Load Balance

Recovery load distribution depends on data placement.

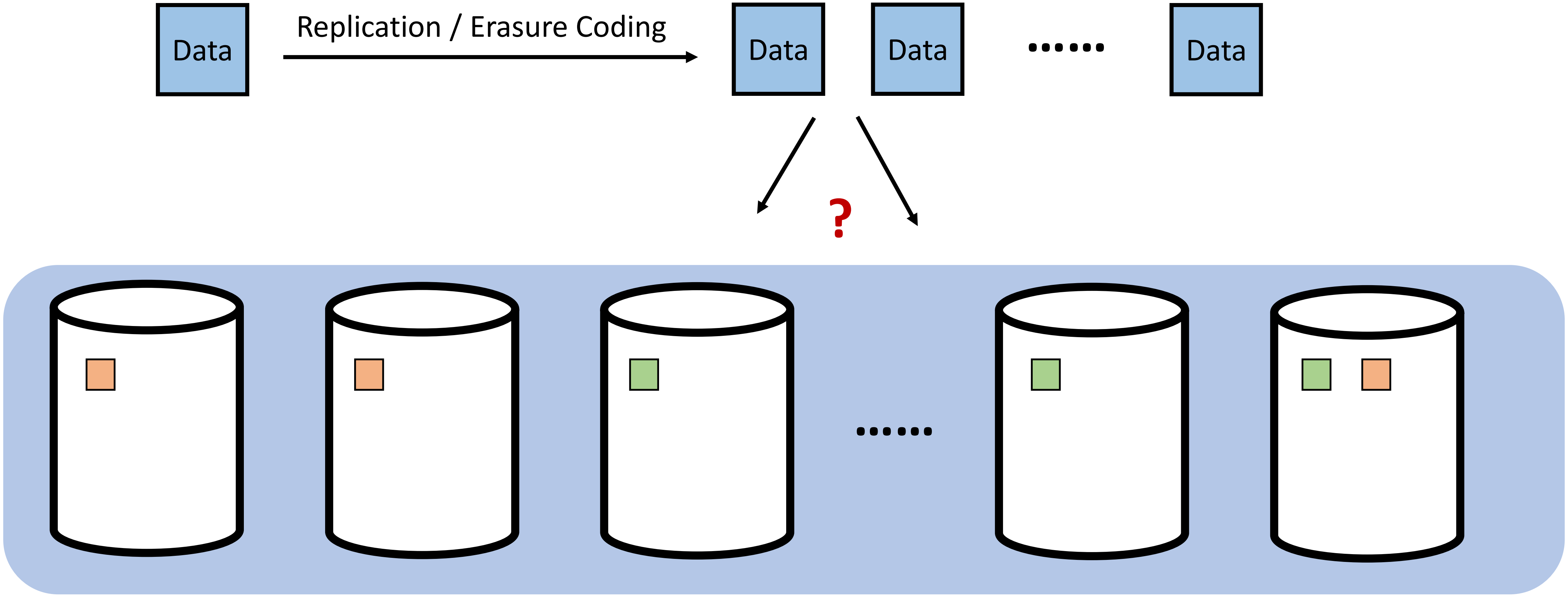
How can we design a data placement algorithm to balance recovery load?

Recovery Load Graph

Let each node denote a disk/a server, let $E_{i,j}$ denote the load of reading data from node *j* when node *i* fails, we can then build a **recovery load graph**.



Optimal Recovery Load Distribution Problem

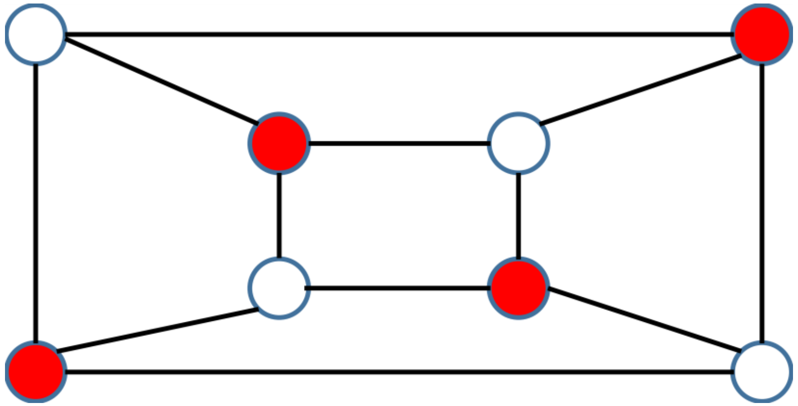


Given current data placement, how should we integrate a new placement group so that the maximal edge weight in recovery load graph is minimized?

NP-Hardness of the Problem

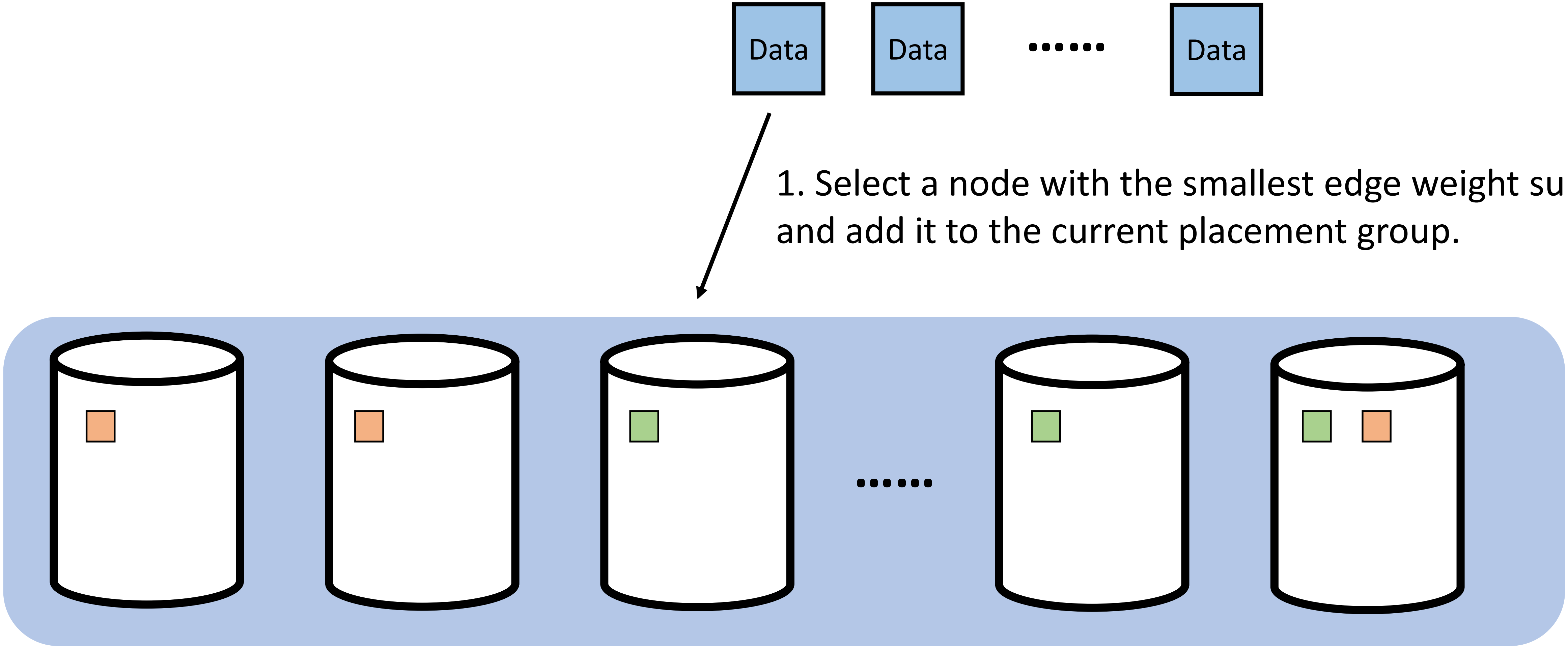
Maximum Independent Set Problem:

Verify if a graph has a non-adjacent vertex subset of size no less than n.



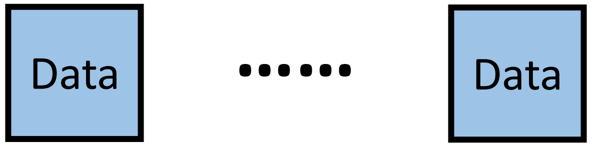
- **Maximum Independent Set Problem** can be reduced to optimal recovery load distribution problem in polynomial time.
- The optimal recovery load distribution problem is thus **NP-hard**.
- Detailed proof is in the paper.

Our Algorithm – Greedy Data Placement

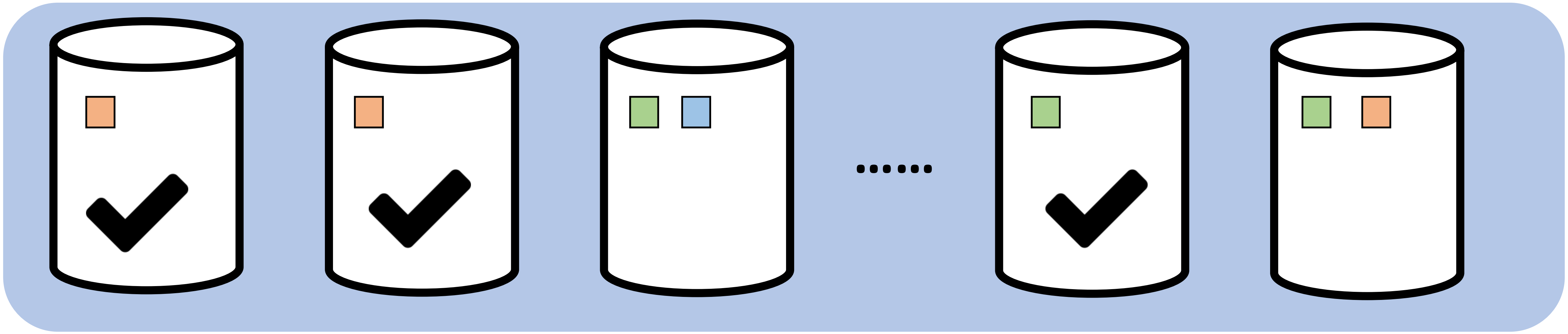


1. Select a node with the smallest edge weight sum and add it to the current placement group.

Our Algorithm – Greedy Data Placement

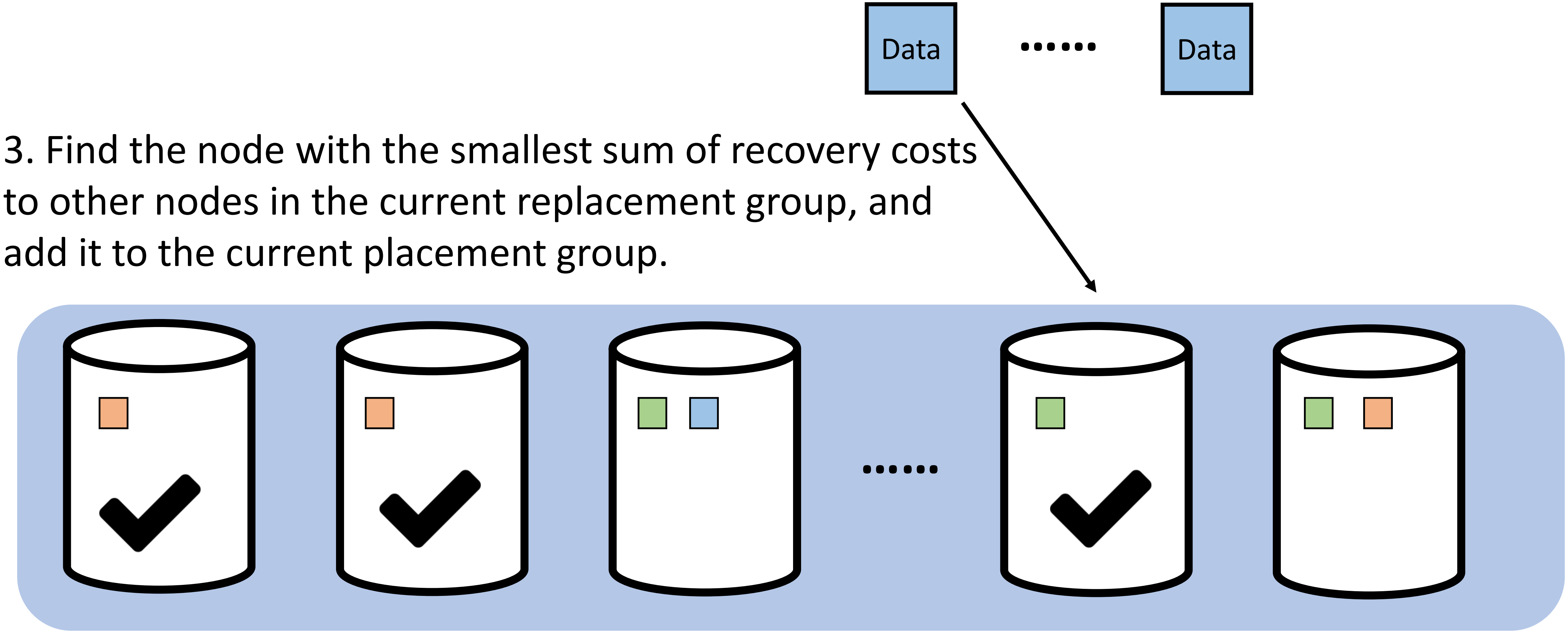


2. Find nodes whose amount of data do not exceed the threshold.



Our Algorithm – Greedy Data Placement

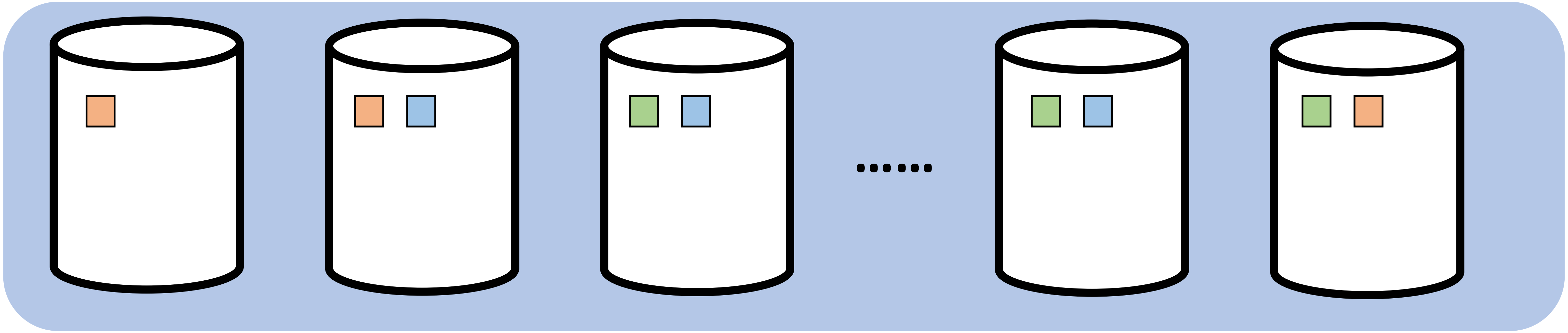
3. Find the node with the smallest sum of recovery costs to other nodes in the current replacement group, and add it to the current placement group.



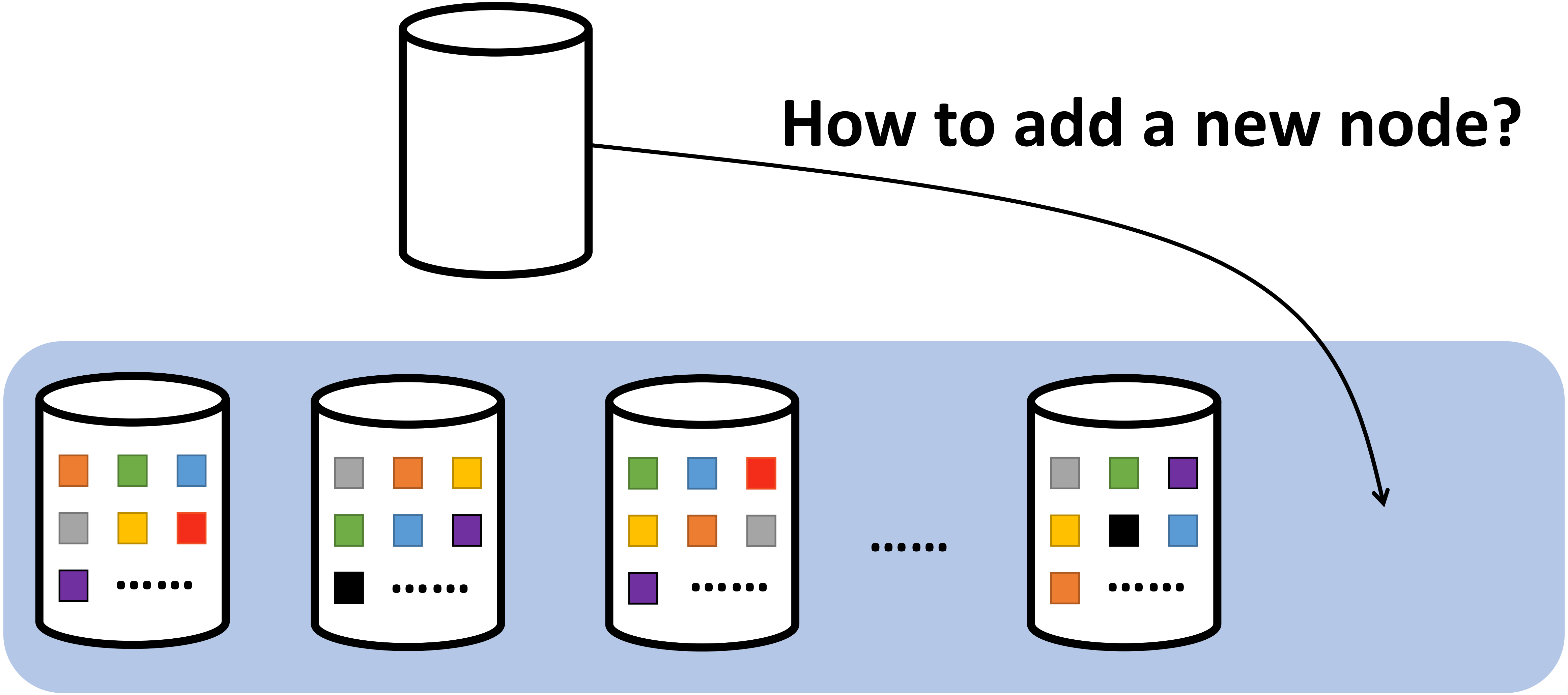
The node with the **smallest sum of recovery costs** to other nodes in the current replacement group

Our Algorithm – Greedy Data Placement

4. Repeat 2 and 3 until we find the placement group.

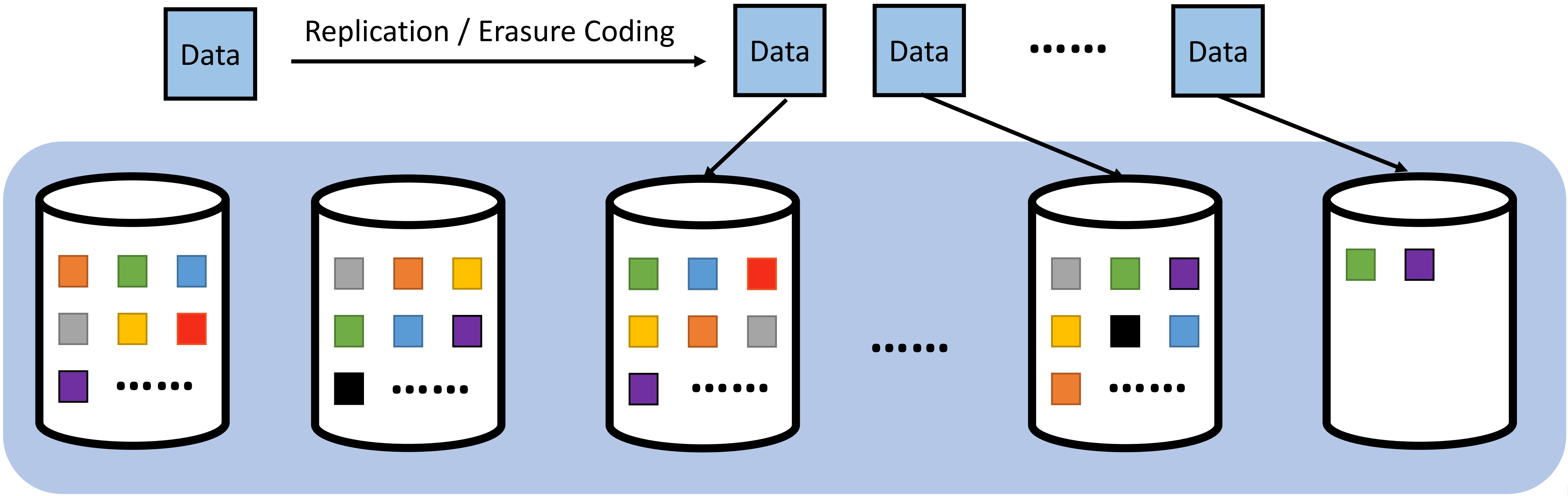


System Expansion Support

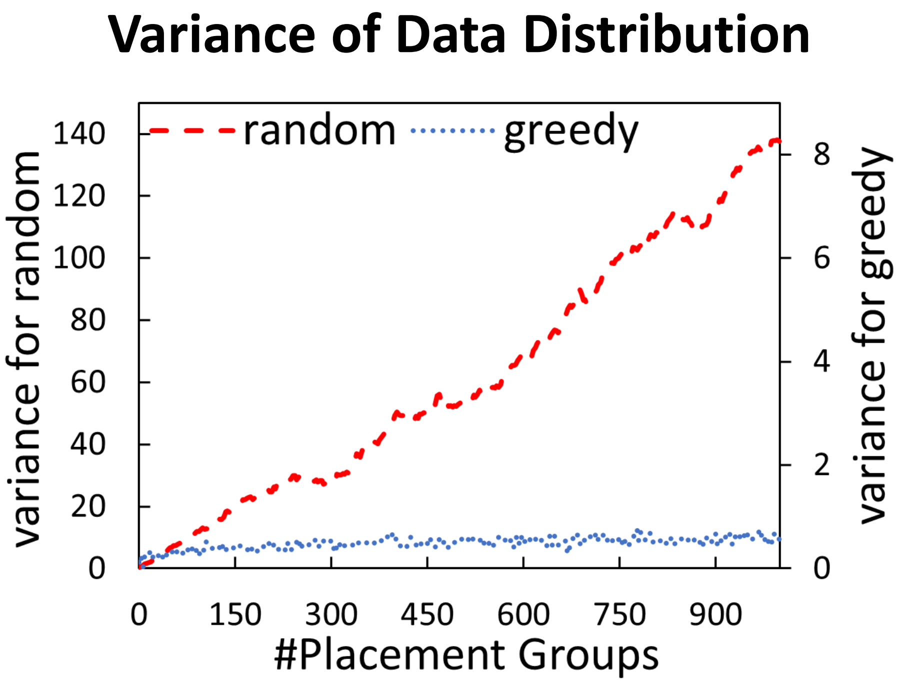

How to add a new node?

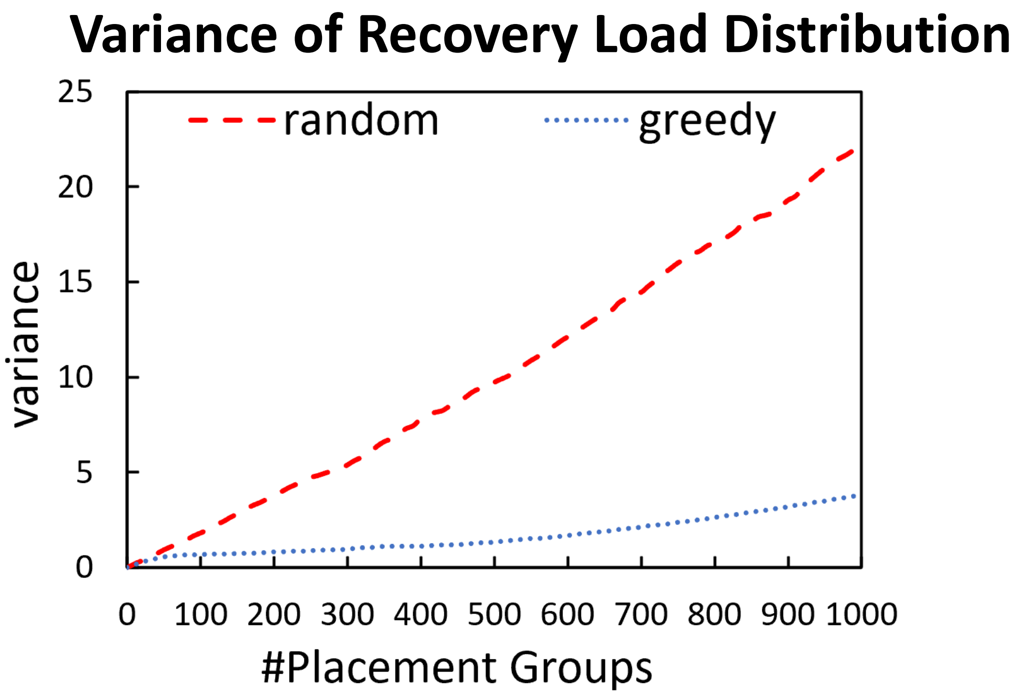
System Expansion Support

- No need to migrate data.
- Put new data at a **controlled speed** on new nodes.
- The system will become balanced over time, as new data is more possible to be placed on new nodes.



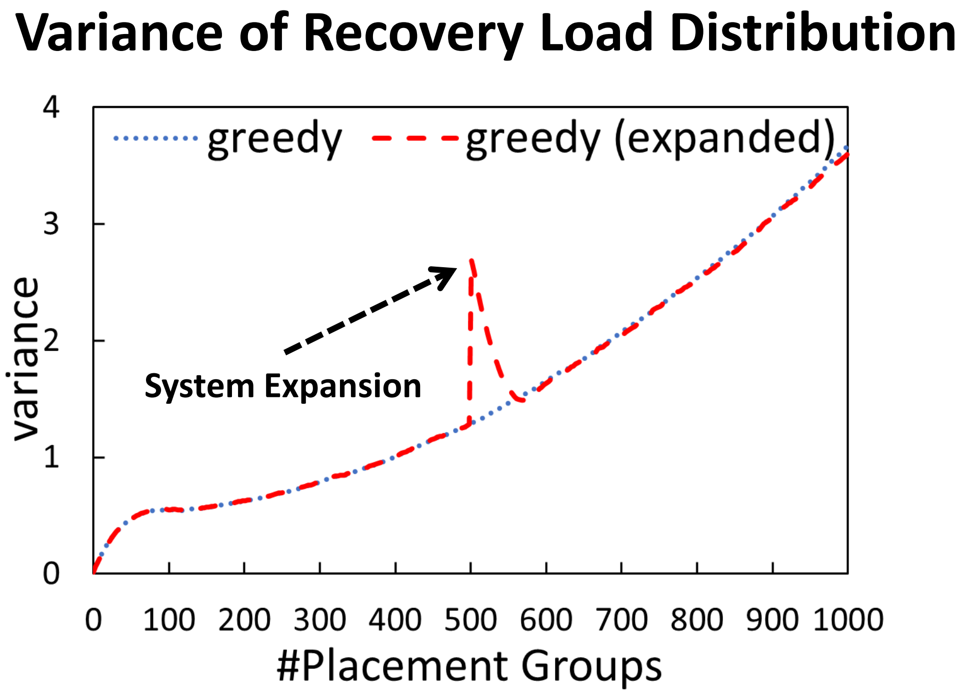
Evaluation – Micro benchmark



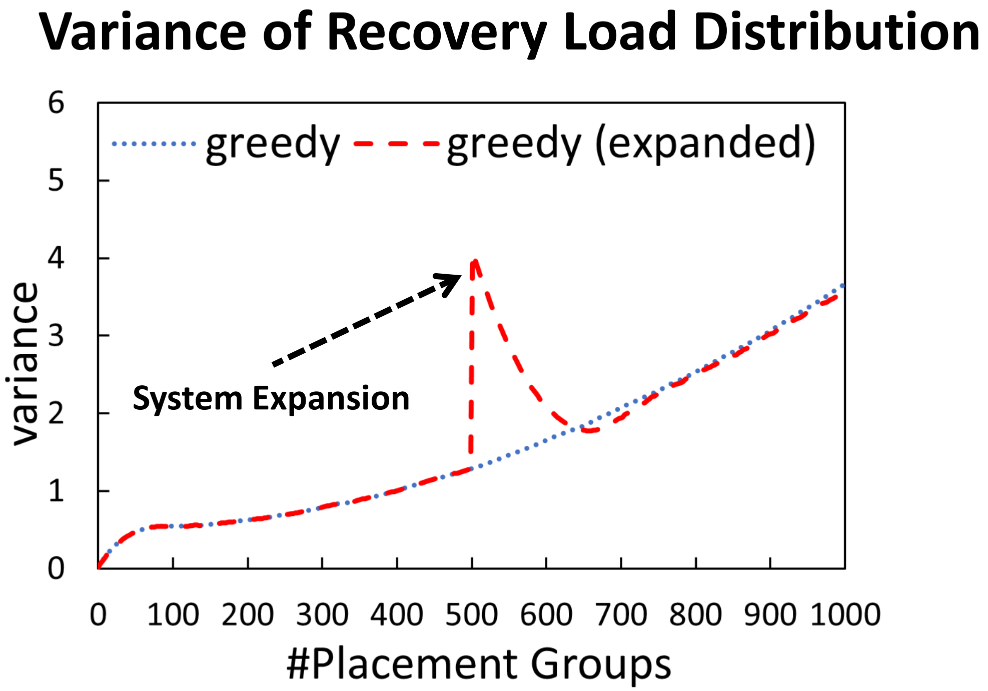


Greedy data placement can provide more balanced data distribution and recovery load distribution.

Evaluation – System Expansion



Add a node

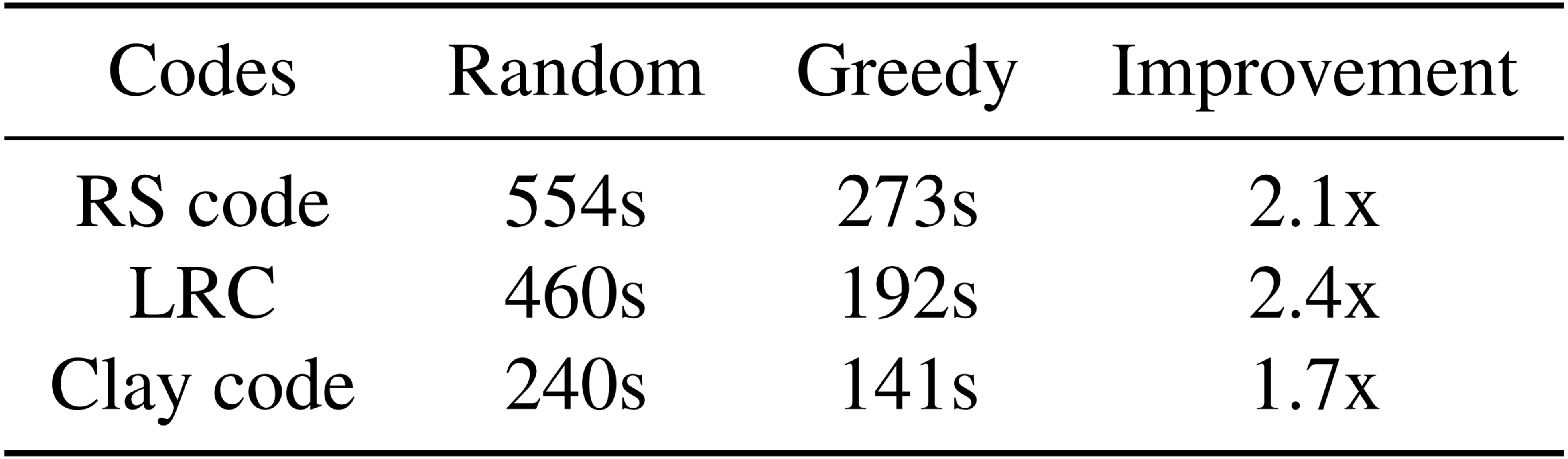


Add two nodes

Recovery load can re-establish its balance after system expansion when new disks are populated with data.

Evaluation – Overall Performance

Average Total Recovery Time



Codes	Random	Greedy	Improvement
RS code	554s	273s	2.1x
LRC	460s	192s	2.4x
Clay code	240s	141s	1.7x

Greedy data placement can reduce recovery time significantly.

Conclusion

- Recovery load can be imbalanced for parallelized recovery.

- Optimal Recovery Load Distribution Problem is an **NP-hard** problem.

- **Greedy** data placement algorithm can be used to improve recovery load balance.

- Better algorithms remain to be explored.

Thanks!

shanyingdi@zgclab.edu.cn