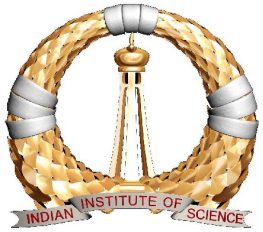
Evaluation of Codes with Inherent Double Replication for Hadoop



Indian Institute of Science



M. Nikhil Krishnan,

N. Prakash,

V. Lalitha,

Birenjith Sasidharan,

P. Vijay Kumar

Srinivasan Narayanamurthy,

Ranjit Kumar,

Siddhartha Nandi

Acknowledgement:

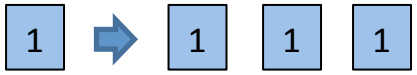
Many thanks go out to USENIX for the generous Travel Grant

that made this trip possible!!!

An Overview

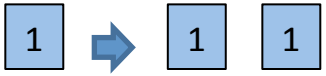
Double Versus Triple Replication

- Triple replication of data is common in Hadoop:



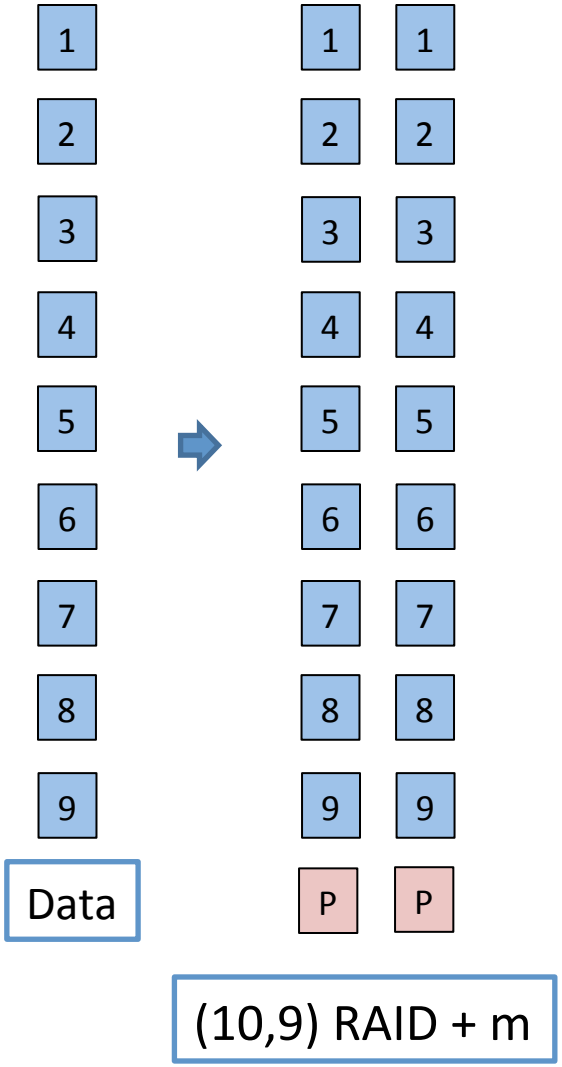
  - Data availability for MapReduce (MR) operations
  - Resiliency to node failure
  - Storage overhead (OH)

- Double replication in comparison:



  - Data availability for MR operations (moderate workloads)
  - Resiliency to node failure
    - Via additional parities (checksums)
    - Eg: RAID + mirroring (RAID + m)
  - Reduced OH

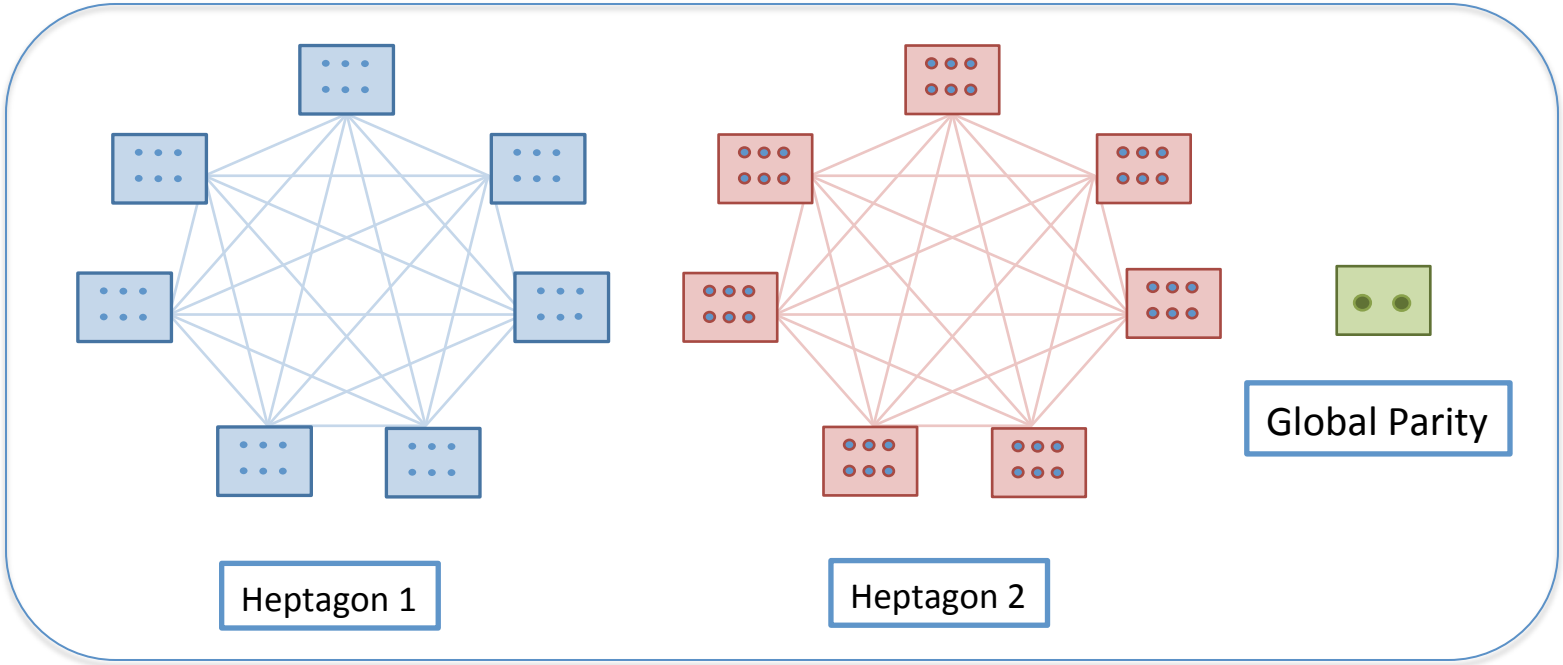
RAID + Mirroring (RAID+m)



- A simple example of double replication with added parity
- P is an XOR of the 9 data blocks
- The two parity blocks ensure adequate resiliency

The Heptagon-Local Code (HLC) :
An Alternate Code with Inherent Double Replication

HLC



Looks complicated, but is not as we shall see..

G. M. Kamath, N. Prakash, V. Lalitha, and P. Vijay Kumar, ``Codes with Local Regeneration and Erasure Correction'', to appear in *IEEE Trans. Inform. Theory*, 2014.

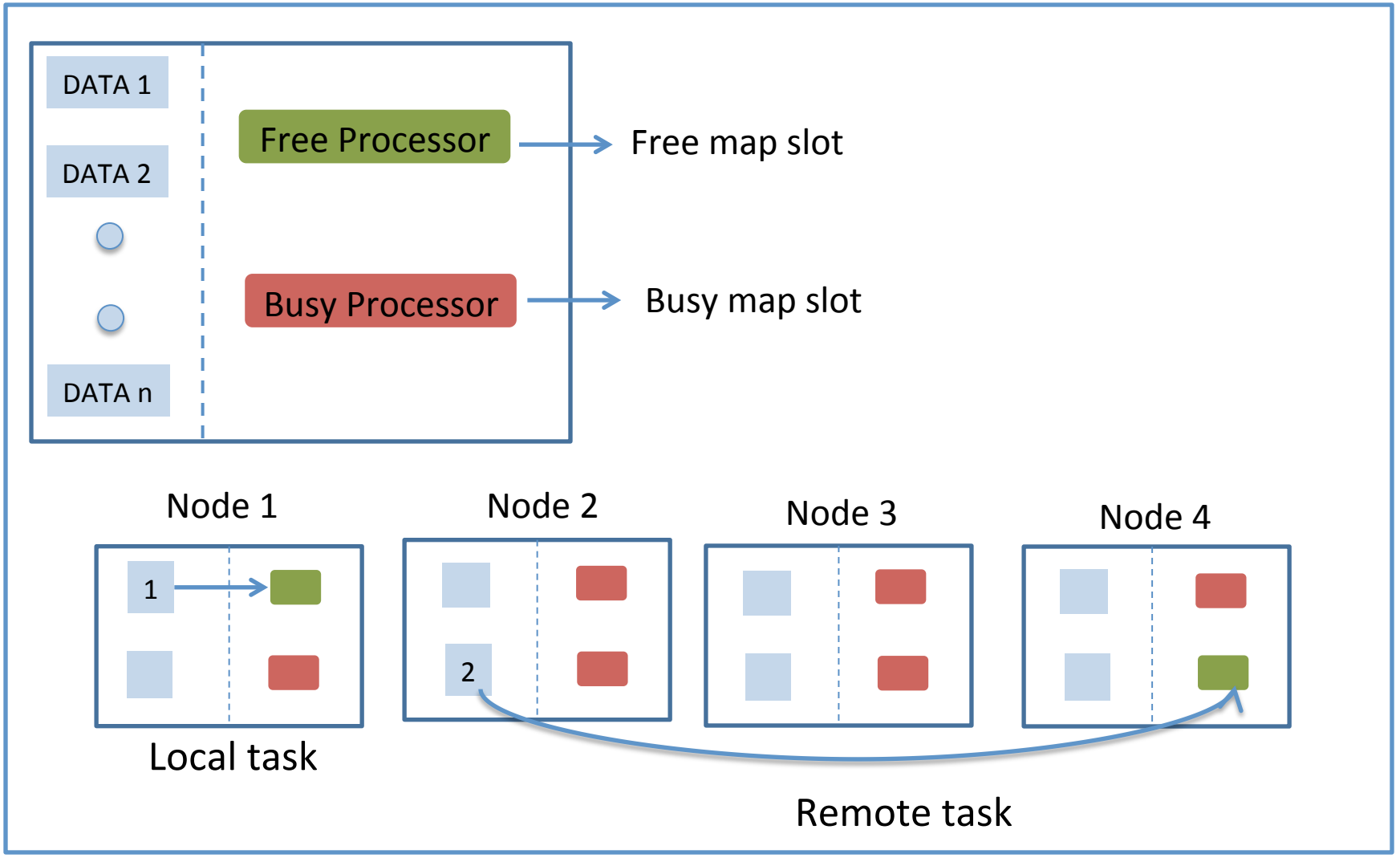
Performance Comparison:

Overhead (OH) and Resiliency (MTTDL)

Code	OH	MTTDL (in years)
2-REP	2	1.5E+05
3-REP	3	1.2E+09
RAID + m	2.22	2.0E+09
HLC	2.15	8.3E+09

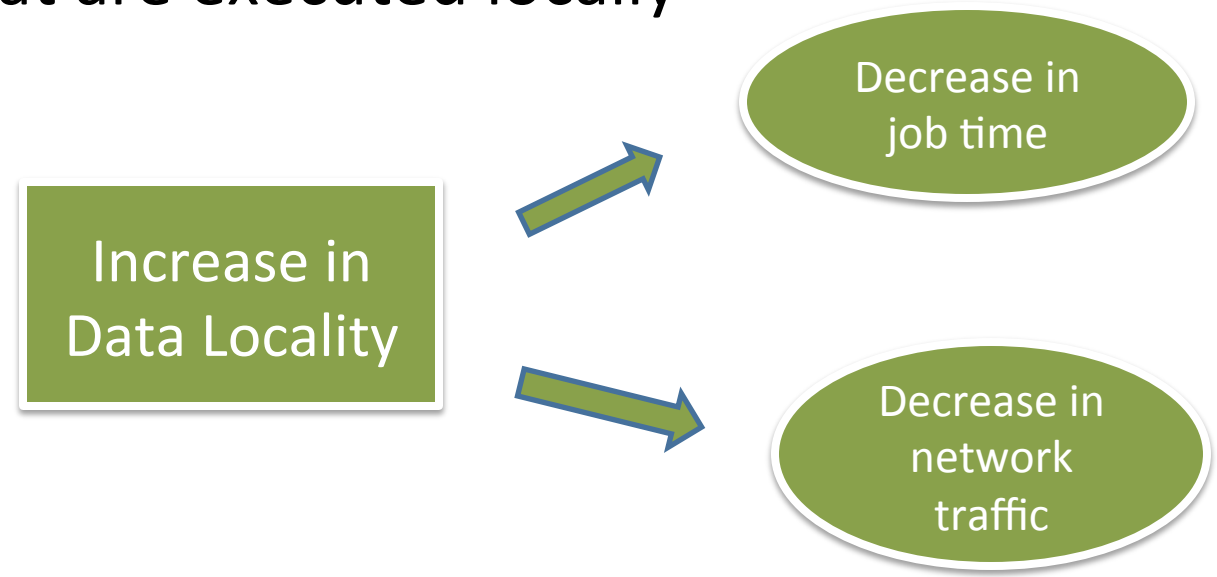
The HLC has reduced overhead for the desired resiliency but there is an issue relating to **Data Locality**….

Map Task Assignment: Local vs Remote Tasks



Importance of Data Locality for Map Tasks

- Data Locality → measured by percentage of tasks that are executed locally



J. Dean and S. Ghemawat, ``Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, 2008.

M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, ``Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in ACM Eurosys, 2010.

The Issue with Data Locality

- The HLC does not fare well in comparison with RAID+m in terms of locality for Map Task Assignment
- We will shortly see the reason for this
- One work around is to have a larger number of processors per node as we do here.

An Alternate Idea of Dealing with the Data Locality Issue

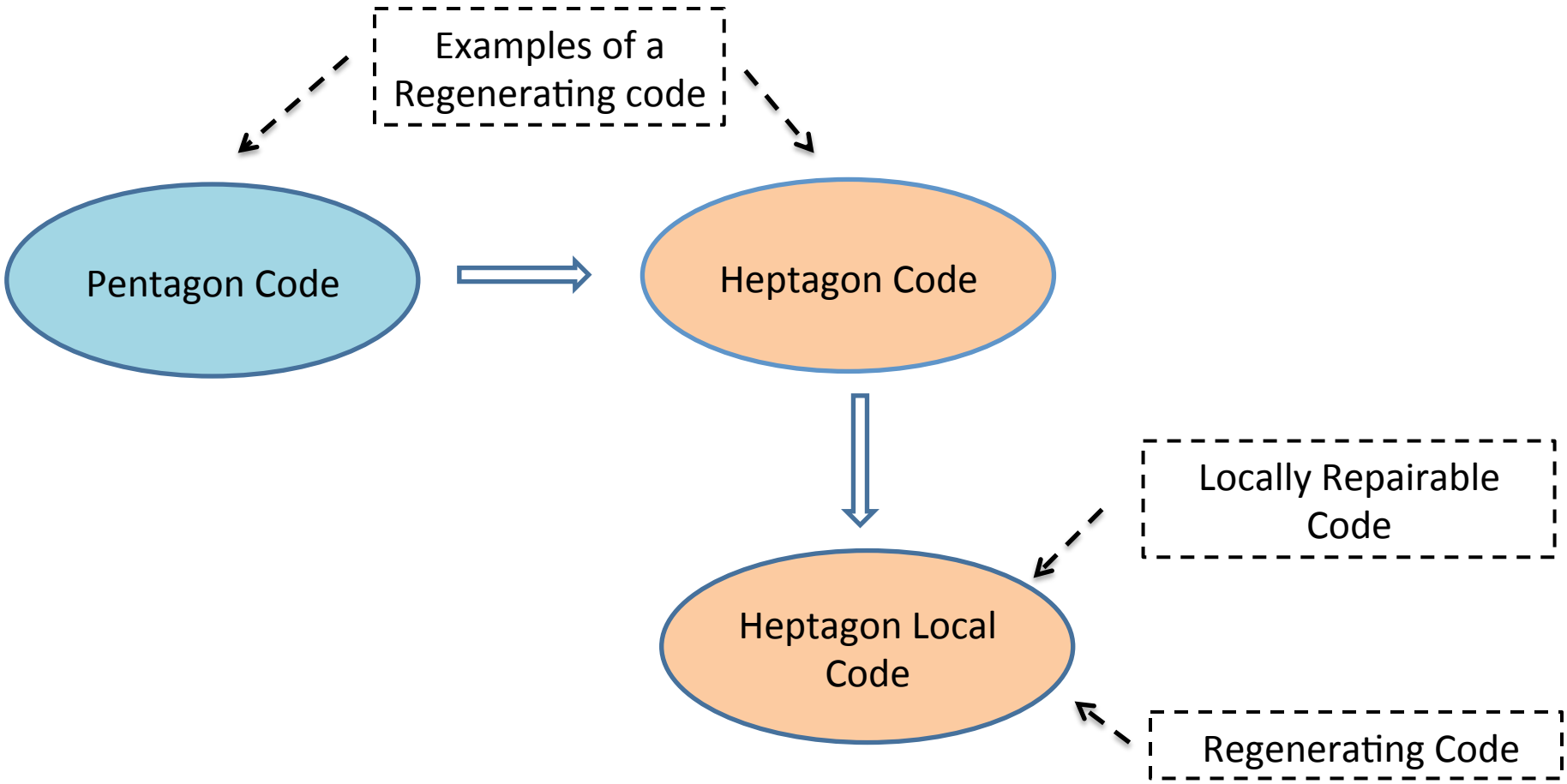
- Modify the HDFS to permit coding across files – this would eliminate this issue altogether (but not part of the present work)
- Other options ?

……..future work!!!

And now for the details….

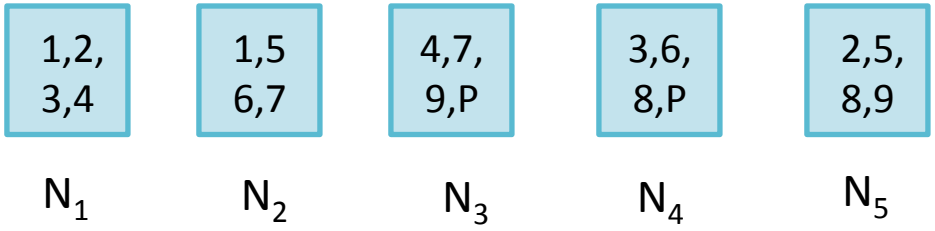
Explaining the Origins of the Heptagon-Local Code (HLC)

Evolution of the Heptagon-Local Code



Regenerating Codes and Locally Repairable Codes are two classes of codes designed with distributed storage in mind

The Pentagon Code



- An example of a code with inherent double replication
- 9 data blocks encoded into 20 blocks and stored in 5 nodes
- Each node stores 4 blocks

Building the Pentagon Code



- Input: 9 data blocks

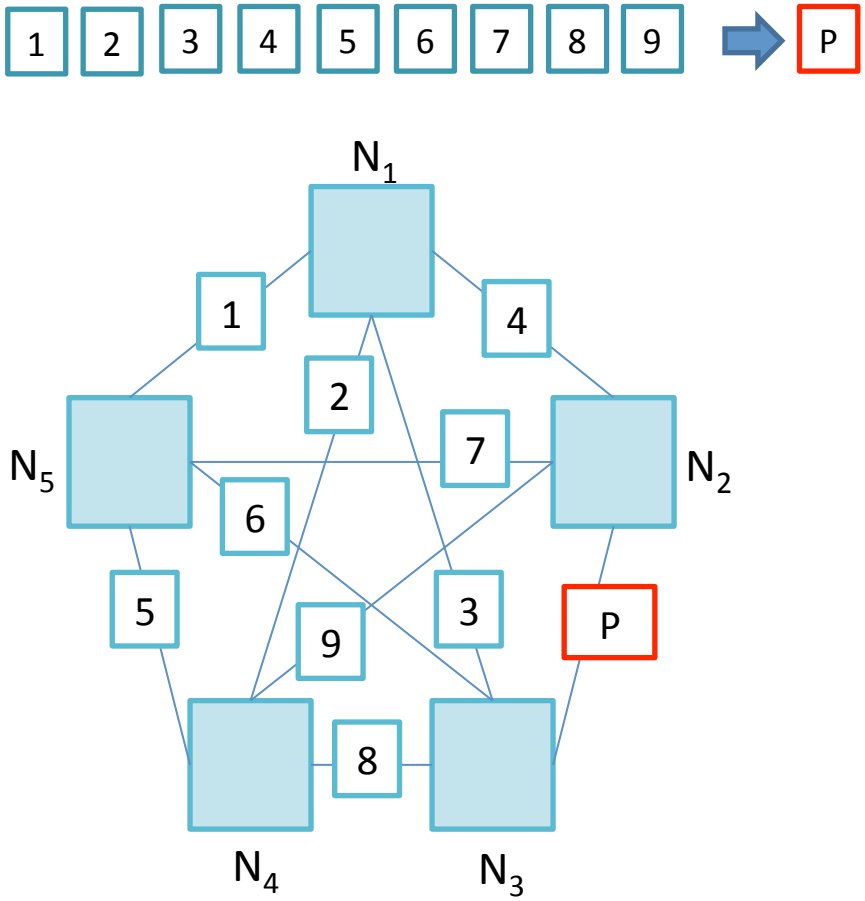
N. B. Shah, K. V. Rashmi, P. Vijay Kumar and K. Ramchandran, ``Distributed Storage Codes with Repair-by-Transfer and Non-achievability of Interior Points on the Storage-Bandwidth Tradeoff, '' *IEEE Trans. Inform. Theory*, Mar. 2012.

Step 1: Adding a Single Checksum



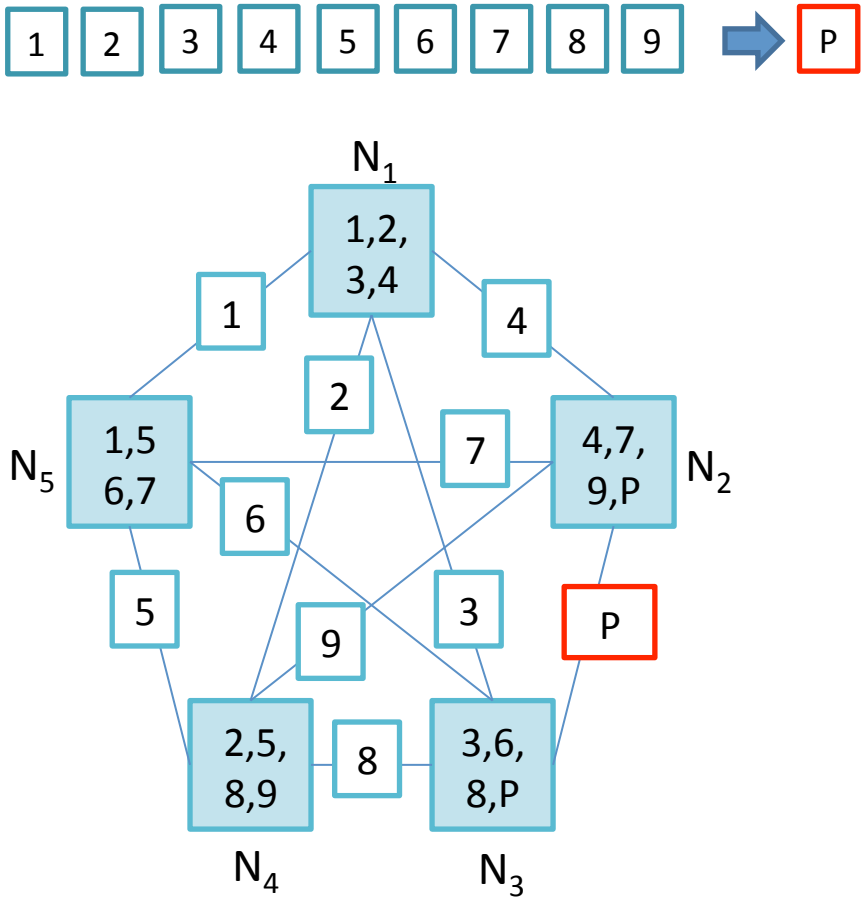
- Input: 9 data blocks
- An additional XOR parity is added

Step 2: Label the Edges of the Graph with these 10 Blocks



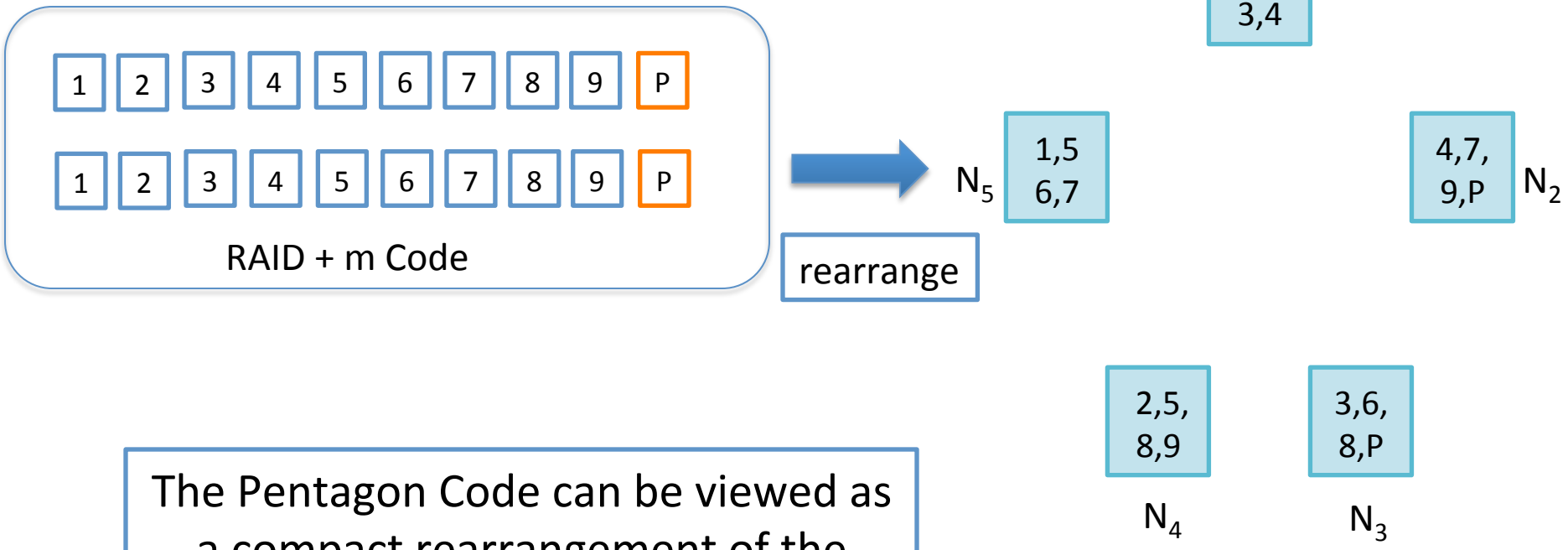
- Input: 9 data blocks
- An additional XOR parity is added
- The resultant 10 symbols are placed on the edges of a fully-connected pentagon

Final Step: Using Edge Labels to Populate the Nodes



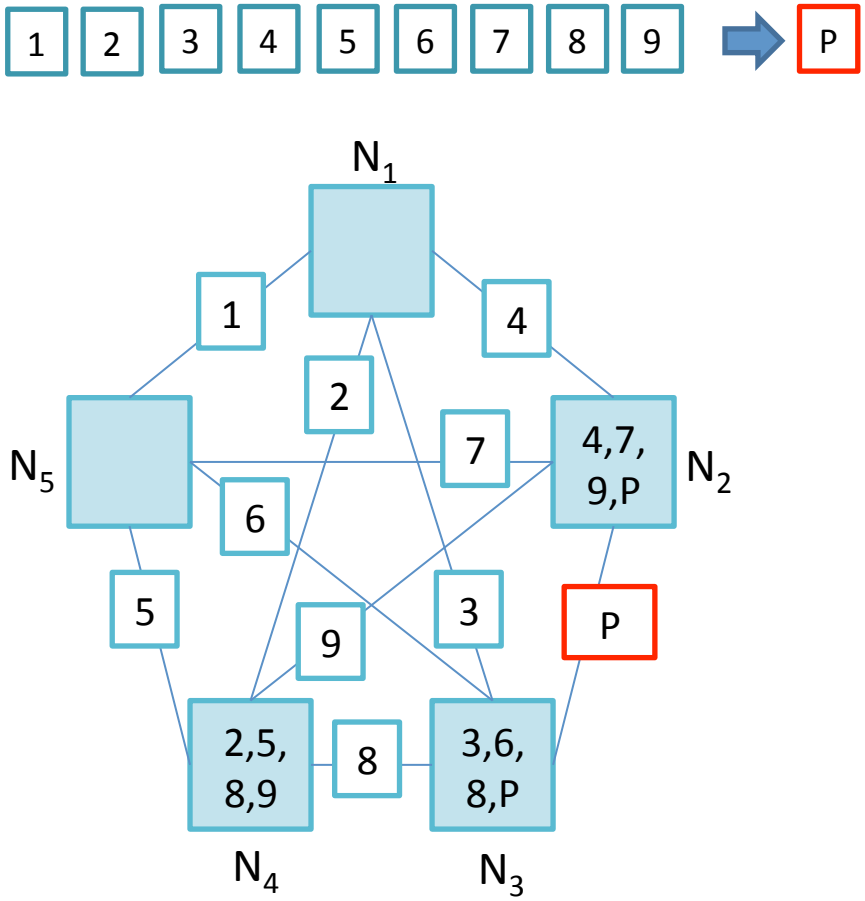
- Input: 9 data blocks
- An additional XOR parity is added
- The resultant 10 symbols are placed on the edges of a fully-connected pentagon
- Each node stores the data appearing on its incoming edges and we are done!

An Alternate View: the Pentagon Code as a Rearrangement of RAID+m



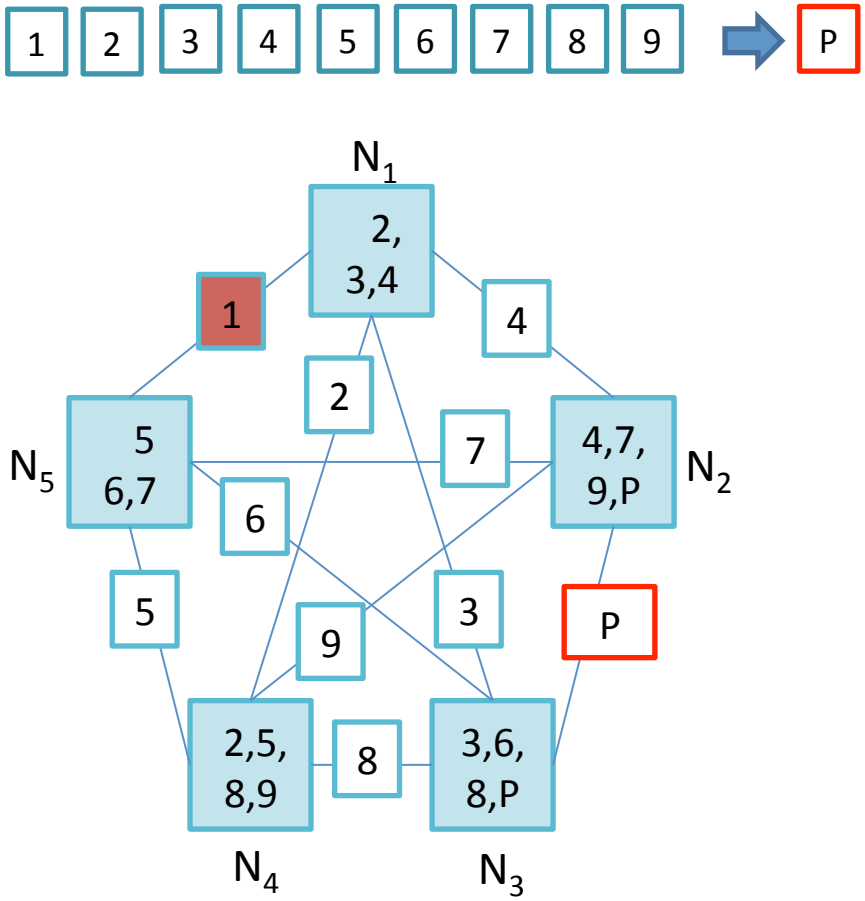
The Pentagon Code can be viewed as a compact rearrangement of the block of the (10,9) RAID+m code

Resiliency of the Pentagon Code



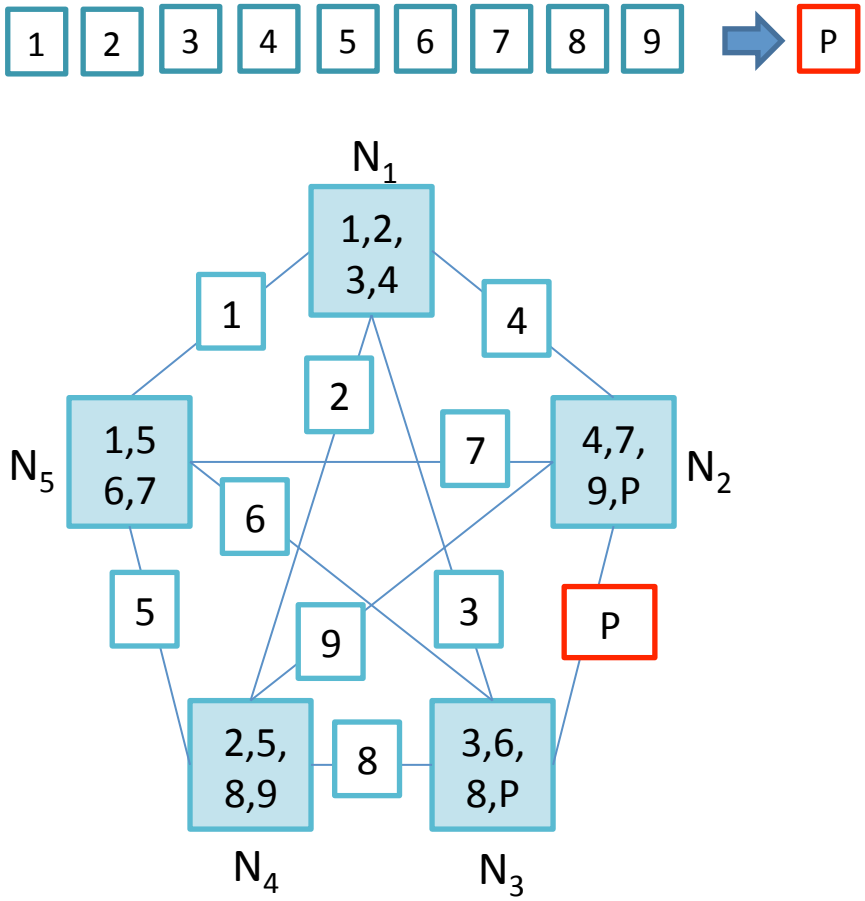
- Can tolerate 2 (out of 5) node failures

Resiliency of the Pentagon Code



- Can tolerate 2 (out of 5) node failures
- All blocks except with the exception of one block 1 can be copied from the other 3 nodes

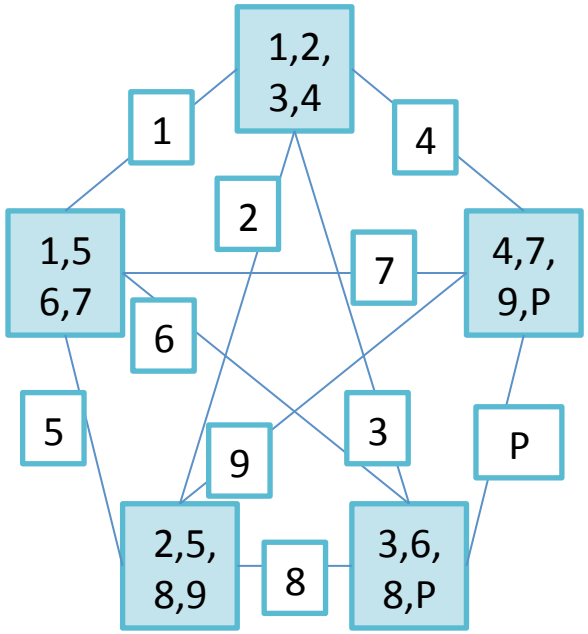
Resiliency of the Pentagon Code



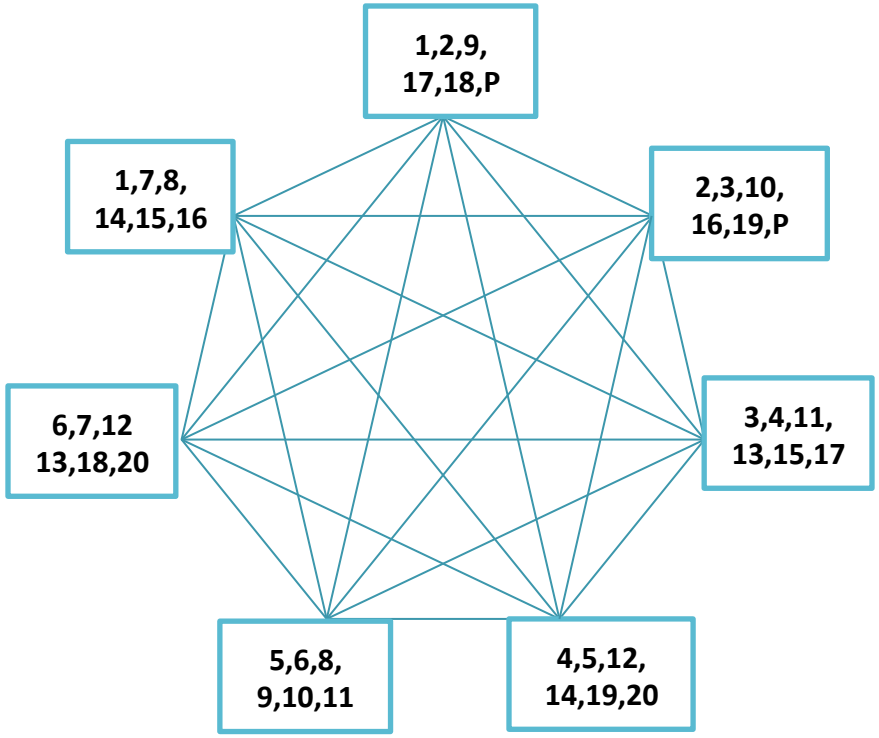
- Can tolerate 2 (out of 5) node failures
- All blocks except with the exception of one block 1 can be copied from the other 3 nodes
- Block shared between the two failed nodes, recovered via parity P

Extending the Pentagon to a Heptagon

- Construction extends to a polygon of any size



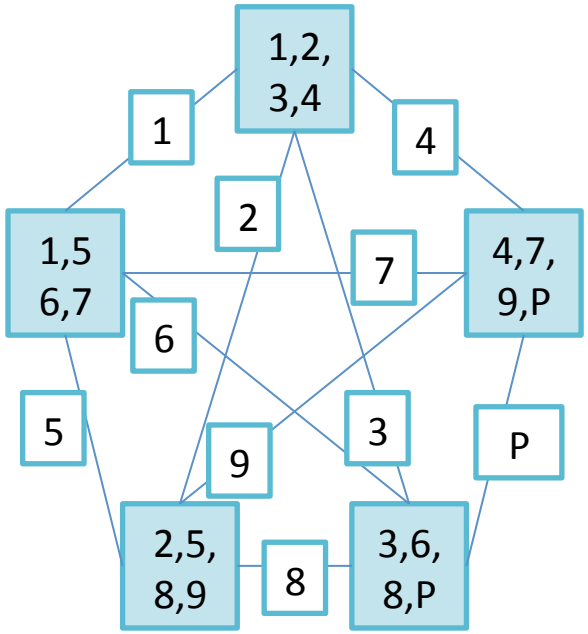
Pentagon : (10, 9) RAID + m



Heptagon : (21, 20) RAID + m

Extending the Pentagon to a Heptagon

- Construction extends to a polygon of any size



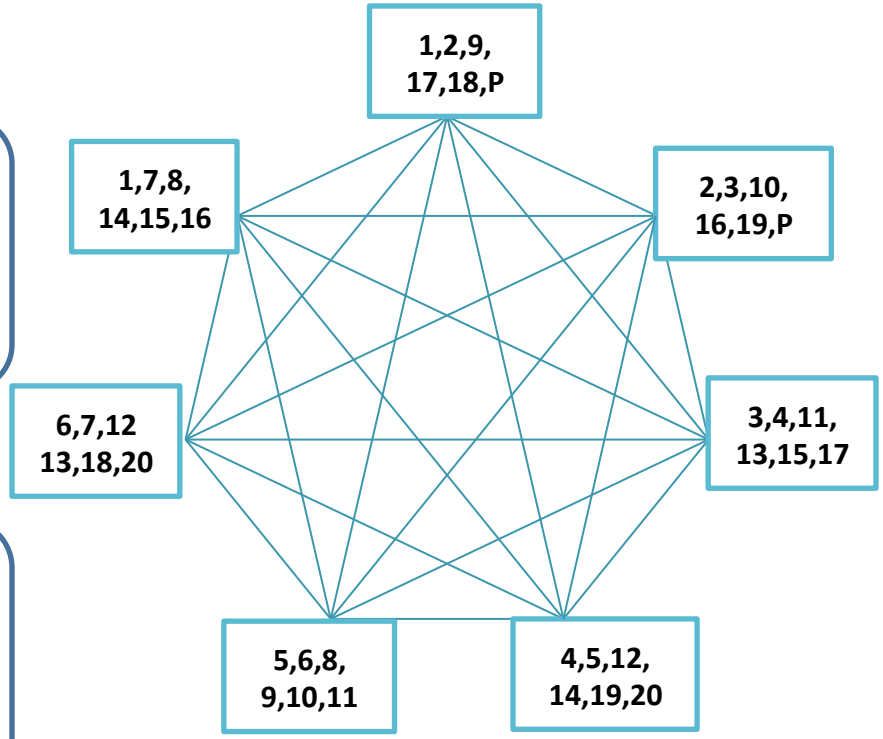
Pentagon : (10, 9) RAID + m

OH decreases

$$\frac{42}{20} < \frac{20}{9}$$

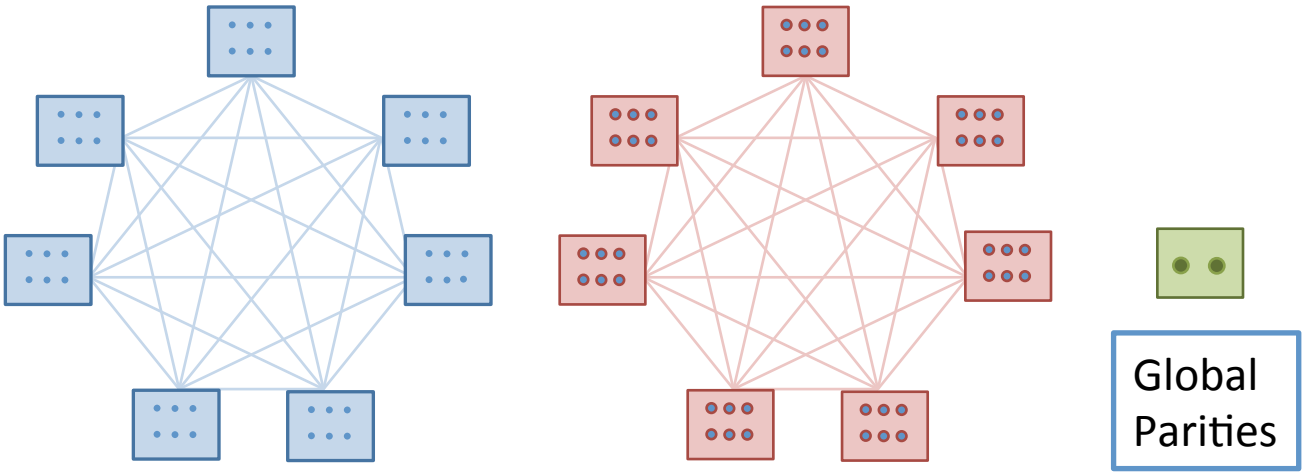
But resiliency
Is reduced

$$\frac{2}{7} < \frac{2}{5}$$



Heptagon : (21, 20) RAID + m

Finally: The Heptagon-Local Code (HLC)



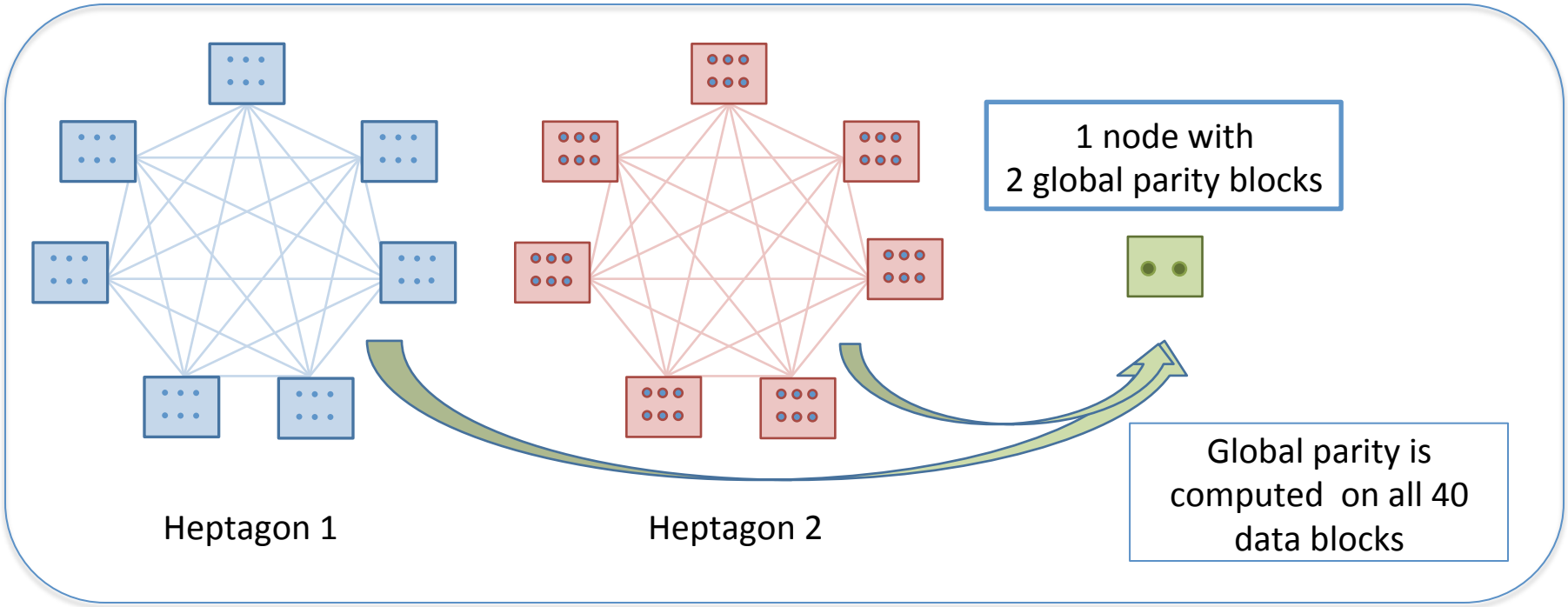
Heptagon 1

Heptagon 2

Global parities offer a simple way to increase the resiliency of the Heptagon Code, at a slight increase in OH

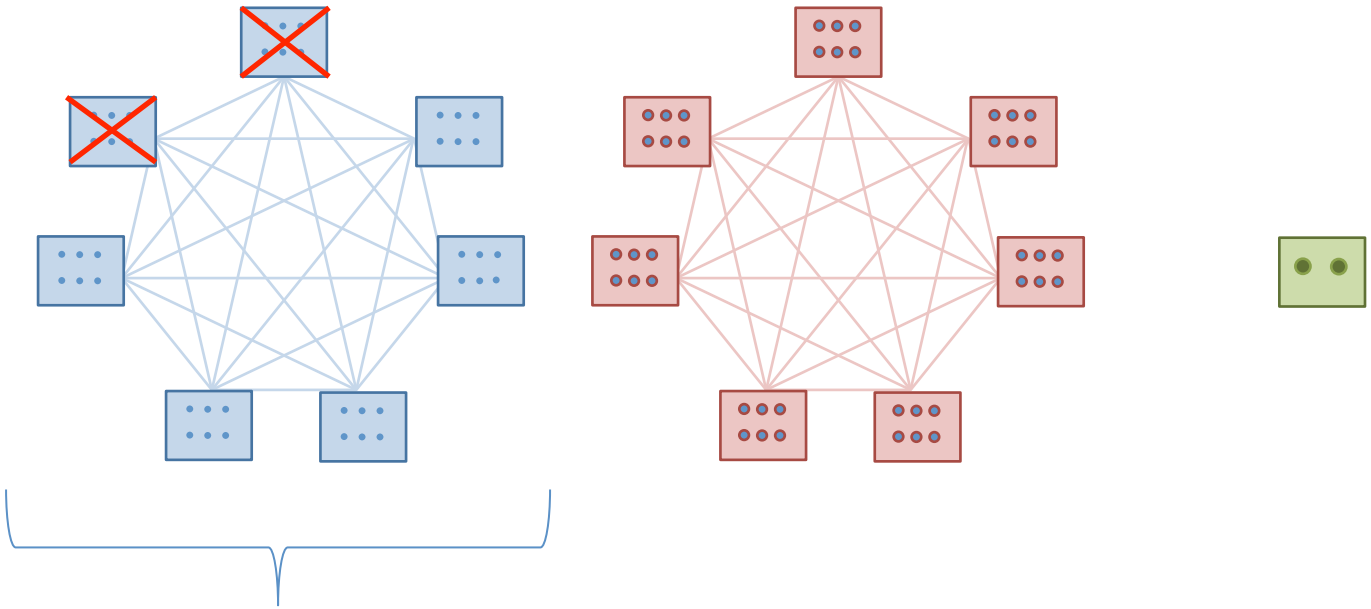
G. M. Kamath, N. Prakash, V. Lalitha, and P. Vijay Kumar, ``Codes with Local Regeneration and Erasure Correction'', accepted for publication in *IEEE Trans. Inform. Theory*, 2014.

Heptagon-Local Code



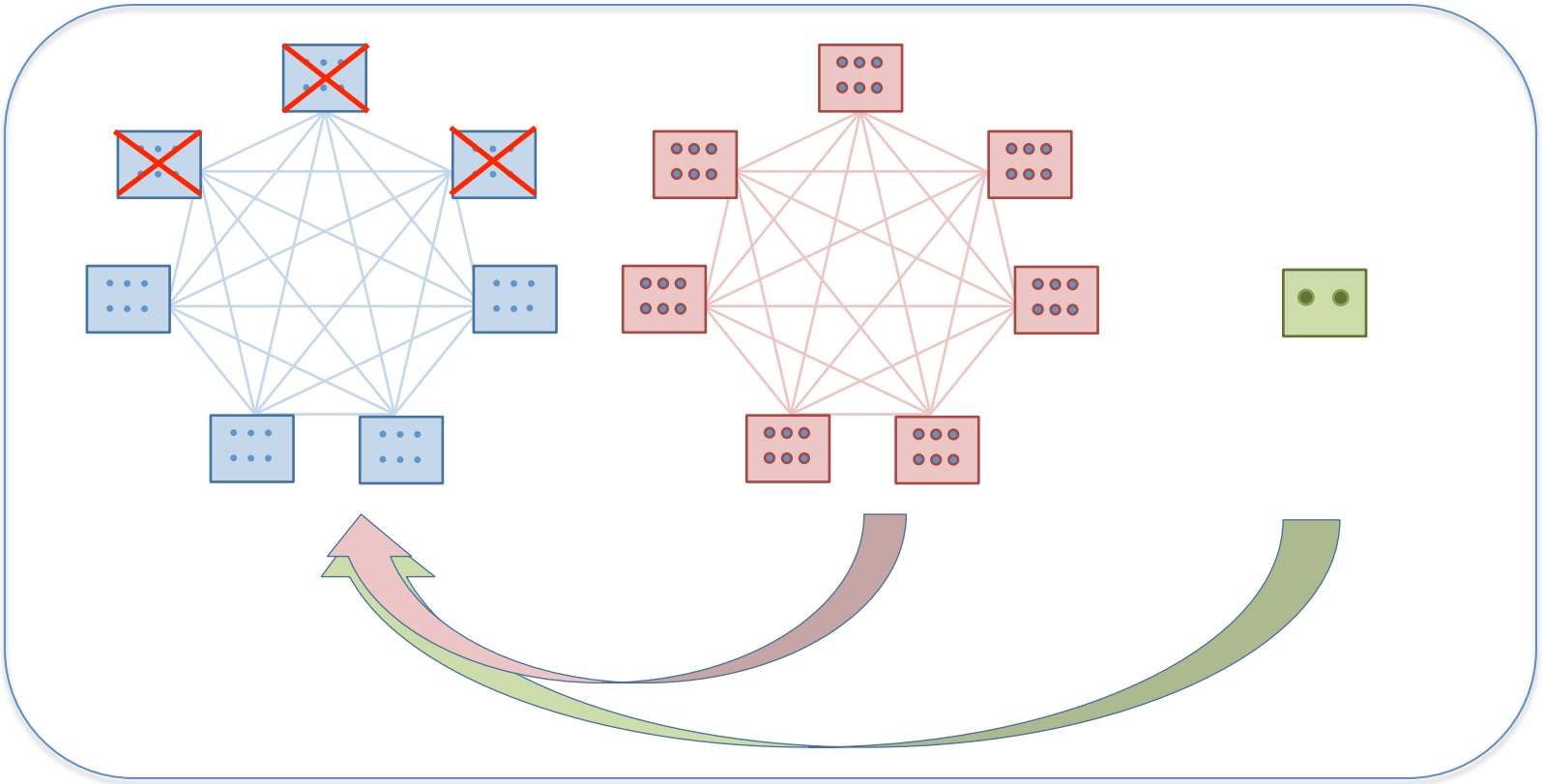
- Overall : 40 data blocks → 86 encoded blocks
- Stored in 15 nodes
- Can tolerate any 3 (out of 15) node failures

Heptagon-Local Code: Recovering from Two Node Erasures



2 node failures within a Heptagon
recovered locally (as in the Pentagon)

Heptagon-Local Code: Recovering from Three Node Erasures



3 node failures within a Heptagon recovered using the two parities and with the help of the second Heptagon

Overhead vs Resiliency

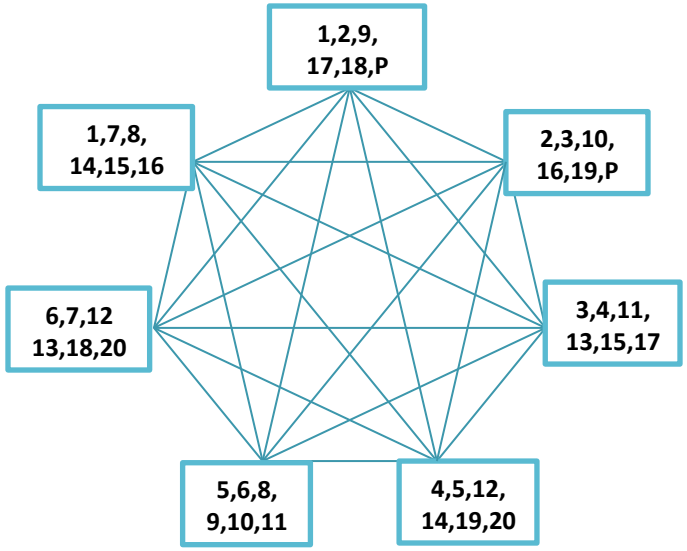
Code	Overhead	MTTDL (in years)
2-REP	2	1.46E+05
3-REP	3	1.20E+09
Pentagon	2.22	1.05E+08
Heptagon	2.1	2.68E+07
(10, 9) RAID + m	2.22	2.03E+09
(12,11) RAID + m	2.18	6.50E+08
Heptagon-Local	2.15	8.34E+09

- MTTDL calculated for a 25 node system using standard models
- The heptagon-local code is a better choice than RAID + m

Q. Xin, E. L. Miller, T. Schwarz, D. D. Long, S. A. Brandt, and W. Litwin, ``Reliability mechanisms for very large storage systems," in *IEEE Massive Storage Systems and Technology (MSST)*, 2003.

Returning to the Data Locality Issue

Data Locality for the Heptagon Code



- Data Locality affected by the concentration of data blocks in the Heptagon Code
- Say 2 map slots / node and MR job on 1 Heptagon-Coded file
  - No. of local tasks ≤ (7 X 2) = 14
  - No. of remote tasks ≥ (20 – 14) = 6
- Data Locality : RAID + m > Heptagon-Local

Evaluation of MR Performance in Hadoop

Experimental Set-up

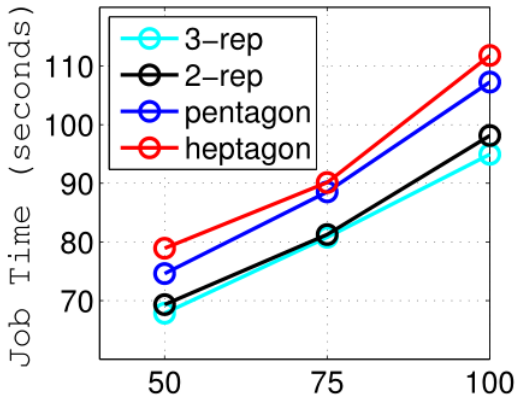
- Implemented the Pentagon and Heptagon-Local Codes on top of HDFS-RAID
- HDFS-RAID : Facebook's open-source implementation of Reed-Solomon codes over HDFS

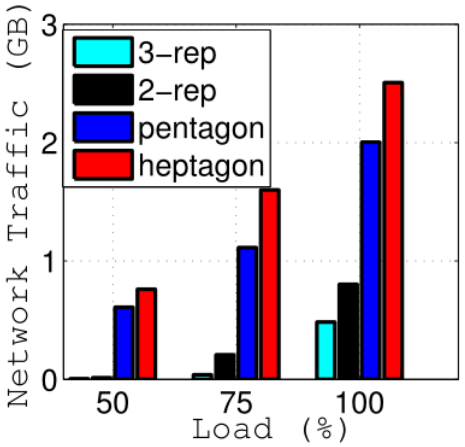
	SET-UP A	SET-UP B
Processors (Map Slots) Per Node	2	4
Nodes	25	10
Platforms	Dual-Core Laptops	Server-Class Machines
Codes Tested	Pentagon, Heptagon-Local*	Pentagon

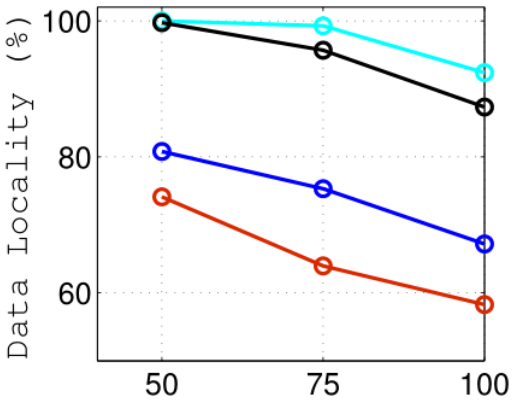
* The Heptagon code was used in the experiments, but both Heptagon and Heptagon-Local have the same Map-Reduce performance

MR Performance (Batch Jobs)

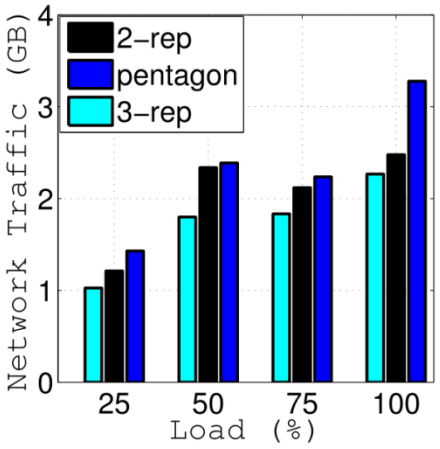
Set-up 1 : 2 map slots / node

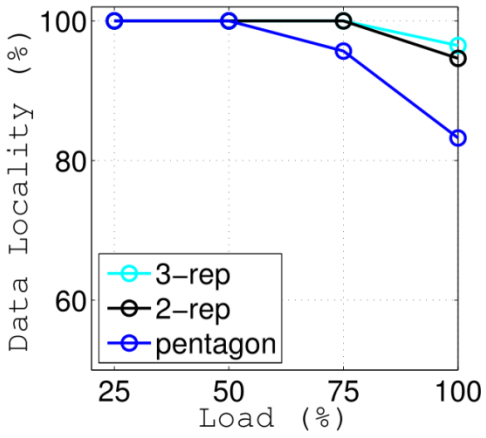






Set-up 2 : 4 map slots / node





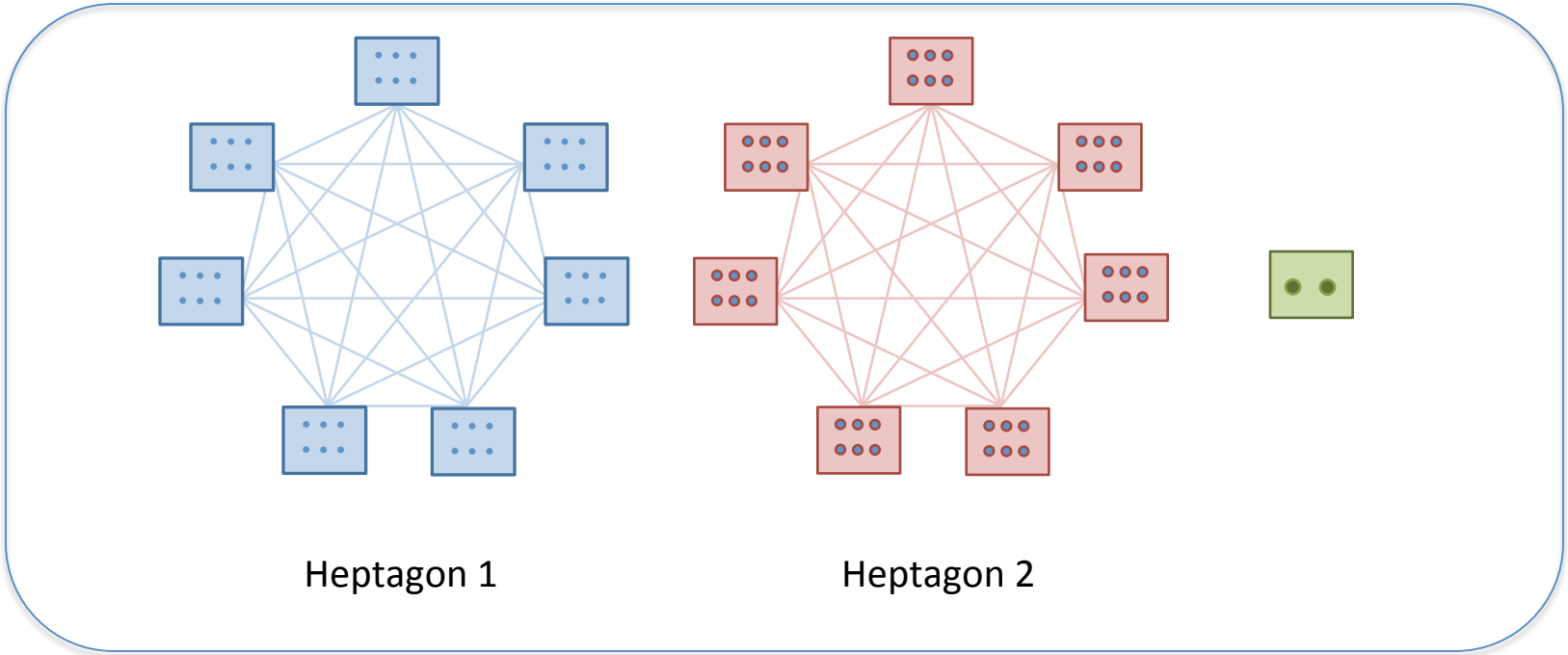
- As expected, substantial loss in performance with 2 map slots
- With 4 map slots, pentagon code performs well, even at load of 75 %

Summary

- We discussed a coding scheme – the Heptagon-Local Code, having inherent double replication for data blocks
- The Heptagon-Local Code enjoys good overhead and resiliency, when compared with other schemes such as RAID +m
- For good data locality, a larger number of processor cores / node is needed

Future Work : Restoring Data Locality

- Modify the HDFS to permit erasure coding across files which would eliminate the issue of data concentration



Related Work : Erasure Codes in Distributed Storage

Implementation of locally repairable codes, focus on overhead vs repair-cost

(Huang et al, Sathiamoorthy et al)

Regenerating codes for cloud based systems, focus on decreasing repair bandwidth

(Hu et al, Runhui Li et al)

A new erasure coding scheme having better single-node repair cost than a Reed-Solomon code

(Rashmi et al)

A new class of codes (rotated Reed-Solomon codes) for improved degraded read performance

(Khan et al)

These codes possess a single copy of data block and in the Hadoop context, are perhaps most suited for cold-data storage.

Thanks !