On improving recovery performance in erasure code based geo-diverse storage clusters

Udaya Parampalli

In Collaboration With: Pablo Iganacio Serrano Caneleo, Lakshmi J Mohan, and Dr.Aaron Harwood
University of Melbourne, Australia

DRCN, March 2016, Paris

- Most cloud service providers are building geo-distributed network of data centers that have their data nodes spanning wide geographical areas.
- These process huge volumes of data that require data resiliency.
- Data centres now use erasure codes in place of default replication for providing fault-tolerance for archival type data.
- Erasure coded storage offers same or better resilience to data node failures as compared to replication at a reduced storage overhead.
- Handling node failures is difficult in coded systems since it consumes network traffic *(repair bandwidth)* involved in downloading the required data.
- This becomes more complex in a geo-diverse environment.

Repair problem with an example

Example storage system spanning three locations
coded using a (4,2) erasure code

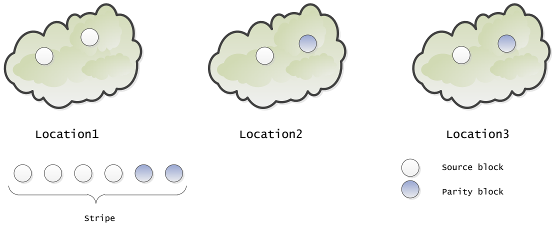


Figure: Before failure

A Node in Location2 fails and the repair process
consumes network bandwidth

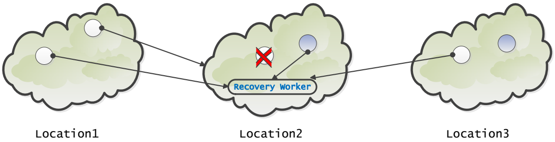


Figure: After failure of a node

Our contribution

- We present an XOR-based erasure coding technique
- Supplement it with topology awareness and parity duplication
- Compare with Facebook's RS codes and XORBAS local parity codes on a nation-wide cluster, which runs on Apache Hadoop
- While the storage requirement of the cluster increases, the idea results in decreased recovery time and recovery bandwidth, making it a better choice for large data centers

Storage using erasure codes

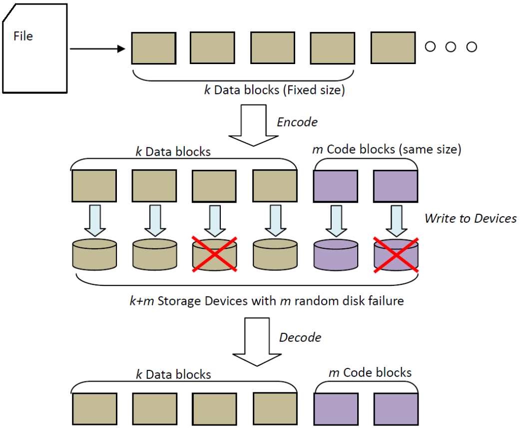


Figure: Nodes are distributed and connected by a network

Our contribution

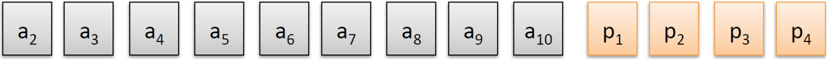


Figure: Facebook (10,4) RS code

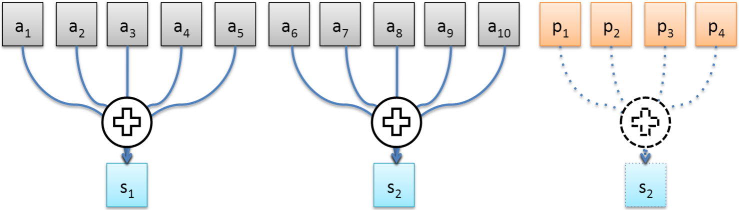
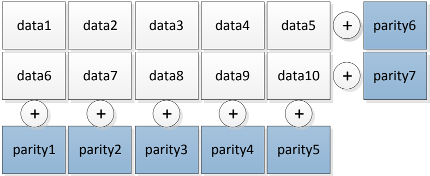


Figure: XORBAS local parity code

XORBAS: Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris S. Papailiopoulos, Alexandros G. Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. Xoring elephants: Novel erasure codes for big data. PVLDB, 6(5):325–336, 2013

- Belongs to the class of block array codes
- A simple code design which takes a single stripe of source blocks (amounting to 10 blocks)
- Rearranges the ten blocks into two rows of five blocks each
- Computes five vertical XOR parity blocks and two horizontal parity blocks, resulting in a total of seven parity blocks
- Very efficient in handling one node failures which involves the download of two of the surviving nodes



- Hadoop can be made location-aware by specifying the extending the idea of rack awareness
- Done via a script and making corresponding changes in the configuration files
- Without location awareness, the block replicas of the file are placed randomly, according to the default block placement policy
- Introducing location-awareness ensures that blocks placement happens exactly the way it is expected to as per the policy

Parity Replication

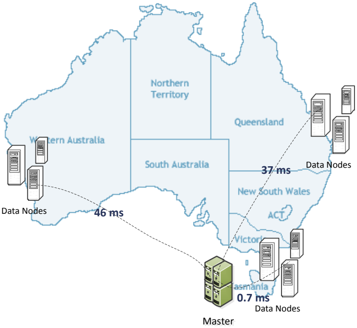
- Inspired by the default replication strategy relied on by Hadoop for storing hot-data
- More copies of parities increases the chances of locality in a geo-diverse cluster setting
- Store two replicas of parities with the aim of bringing down the time taken for reconstruction from node failure.

Code	**Replication**	**Storage Overhead**
Hadoop Reed Solomon	2	1.8x
XORBAS LRC	2	2.2x
MXOR	2	2.4x

Table: Impact of Double Replicating parities in various codes

NeCTAR cluster

- Spanning three locations across Australia on the NeCTAR (National e-Research Collaboration Tools and Resources) research cloud
- 15 data nodes per location, total of 45 nodes
- Master node is located at Tasmania zone
- Primary metric used for evaluating the recovery performance is the recovery time (the total of read time, decode time and waiting time), along with the bytes read



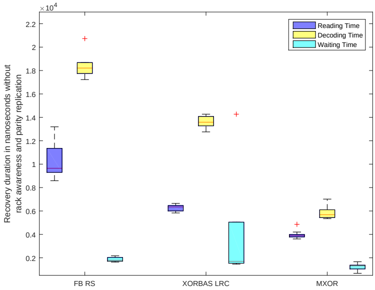
Impact of Parity replication in the test cluster

- For RAIDed source blocks, the replication is set to 1
- It is followed by the deletion of extra copies from the file system
- Metareplicated copies of the parity blocks are placed as per the default block placement policy of HDFS

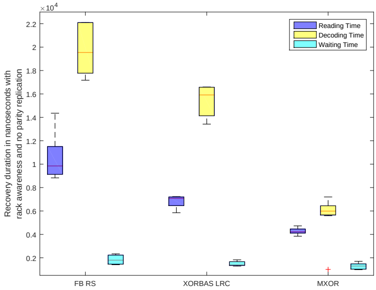
- With the parity block assignment as per the table, the recovery worker node assignment in Tasmania is the best case.
- Even if the recovery worker node is selected from the other two locations namely Queensland and Tasmania, there is still a 50% chance of the required parity block being present at the same location
- i.e. It results in locality of parity blocks, leading to faster recovery

Parity Block	**Tasmania**	**Queensland**	**Perth**
P1	✓	✓	
P2	✓		✓
P3	✓	✓	
P4	✓		✓
P5	✓	✓	
P6	✓		✓

Table: Placement of Double Replicated parities in XORBAS



(a) With no topology awareness and no parity replication



(b) With only topology awareness and no parity replication

Figure: Recovery performance

- In the absence of topology, the efficiency of XORBAS is affected by the distance between the nodes

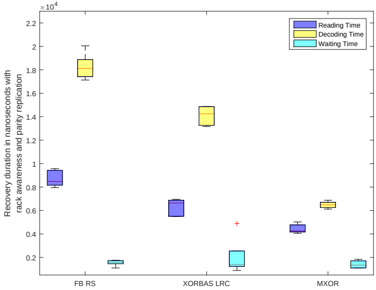


Figure: With both topology awareness and parity replication

- MXOR codes perform faster recovery as compared to FB RS codes and XORBAS LRC codes
- XORBAS LRC in our set-up is observed to perform confirming the claimed performance with a decrease in repair times by 25% as compared to FB RS.

Average Recovery times

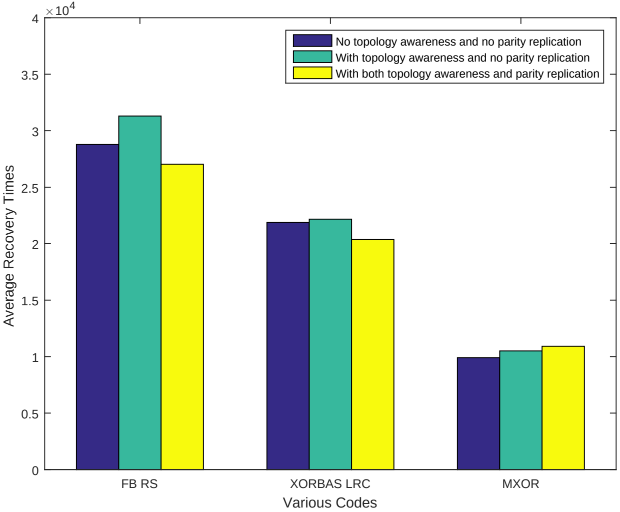


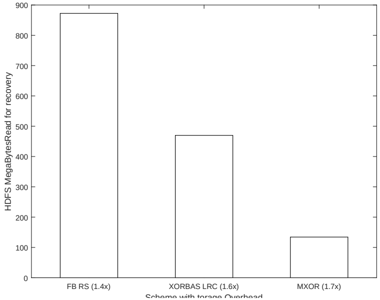
Figure: Average recovery times of codes under different settings

Code	Recovery Time with replication	Recovery Time without replication	Improvement
FB RS	27,039	28,778	6.04%
XORBAS LRC	20,375	21,887	6.91%
MXOR	9,904	10,920	-10.25%

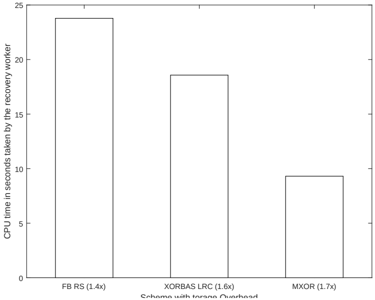
Table: Improvement in recovery performance with parity replication

- MXOR codes do not gain benefit with metareplication
- It requires downloading only one parity block for recovery, hence having extra copies of parities becomes very less relevant
- Also, Hadoop involves internal computation for choosing parity blocks from the available replicas, which is not beneficial for MXOR codes

Improvement with parity replication



(a) Storage Overhead vs. MegaBytes Read



(b) Storage Overhead vs. CPU time

Figure: Bytes read and CPU times Vs. Storage overhead

New Developments

Two recently considered codes have sprungup specificaly to address exact node repairs in DSS:

- Regenerative Codes: focus on the need to minimize the amount of data download needed for node repair.
- Codes with Locality: focus on need to minimize the number of nodes from which data is accessed for node repair.

How many blocks do you need to download to repair the failed node?

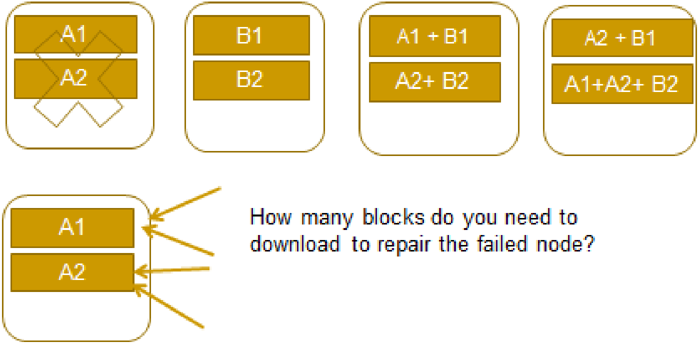


Figure: Example

2 nodes are enough recreate;3 blocks are sufficient

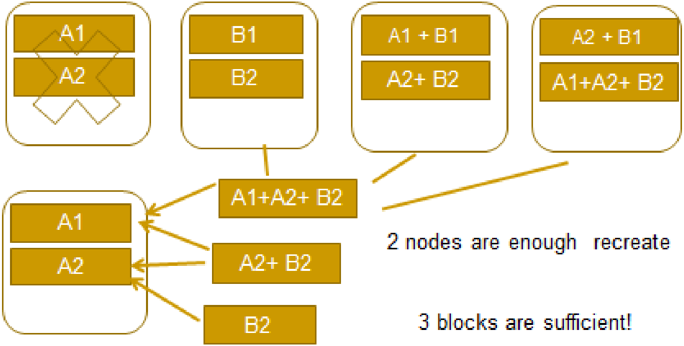


Figure: Example

Repairing the last node

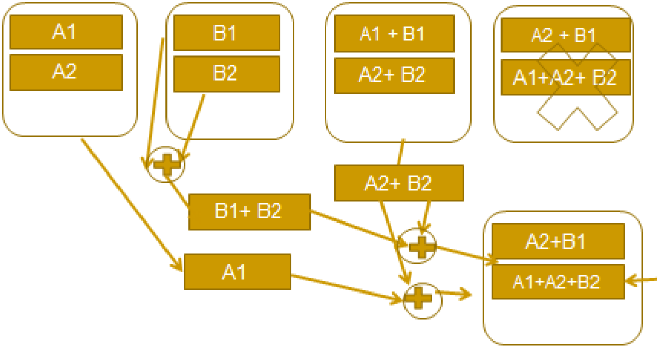
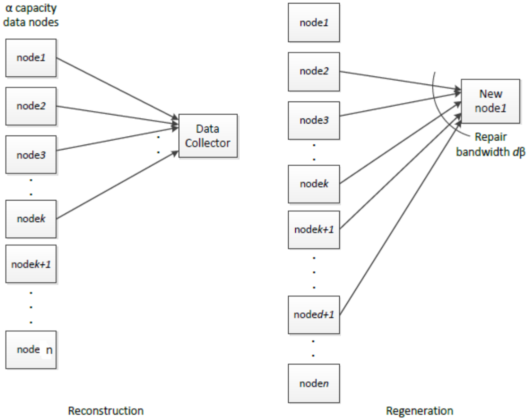


Figure: Example

Model: Codes with regeneration

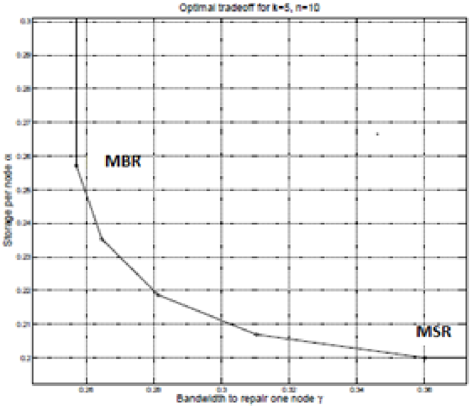
$$\mathcal{B} \leq \sum_{i=0}^{k-1} min\{\alpha, (d-i)\beta\}. \quad (1)$$

A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, Network Coding for Distributed Storage Systems,âœ IEEE Trans. Inform. Th., Sep. 2010.



Bandwidth and Storage Trade-off

$$\mathcal{B} \leq \sum_{i=0}^{k-1} min\{\alpha, (d-i)\beta\}. \qquad (2)$$



Current Implementations

- Facebook and Microsoft have working implementations.
- XORBAS: Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris S. Papailiopoulos, Alexandros G. Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. Xoring elephants: Novel erasure codes for big data. PVLDB, 6(5):325–336, 2013
- Zigzag, Long MDS, Hadamard Design Based Constructions.
- Extensive research is available:
  Jie Li, Xiaohu Tang, and Udaya Parampalli. A framework of constructions of minimum storage regenerating codes with the optimal update/access property for distributed storage systems based on invariant subspace technique, IEEE Transactions on Information Theory, Vol. 61, Issue 4, pp 1920-1932, 2015.

Conclusions and Open problems

- An assessment of three popular codes along with two simple ideas of managing location awareness information and maintaining additional copies of parities was presented
- The results of our study have revealed new facets of erasure codes when implemented on Hadoop in a geo-distributed environment
- Erasure codes, in particular, are not a silver-bullet solution for providing reliability
- The experimental results confirm that topology awareness and metareplication improve the recovery performance to some extent

- A better design that takes into account the block placement policy of both source and parity blocks to suit a geo-diverse cluster is expected to increase the recovery performance
- This is an open problem for future research

About MXOR Implementation

- MXOR code was implemented in Java using standard libraries only. Because the code is a XOR-based code there was no need of special libraries or classes to perform GF arithmetic.

```
@Override
public void encode(int[] message, int[] parity) {
  assert(message.length == stripeSize && parity.length == paritySize);

  int cols = paritySize;
  // init the code values
  for (int i = 0; i < cols; i++)
    parity[i] = message[i];
  // xor the rest properly
  for (int i = cols; i < stripeSize; i++)
    parity[i % cols] ^= message[i];
}
```

- Compilation was made using the standard compilation process available in Hadoop-0.20 and its components. This process utilises *ANT* to organise dependencies and build the jar files.
  - `ant package -Ddist.dir = $HADOOP_HOME/build`

Topology changes and Rack Awarness

- To perform the rack awareness we utilised the default method to specify a particular topology in Hadoop. This method is based on a script and a text file where the pair {*machine*, *IP*} is declared.

```
127.0.0.1          rack-01
localhost          rack-02
10.0.0.1           rack-01
```

- The script was written in *bash*, based on several community scripts. It works reading the file, indicating to Hadoop how to build the topology in a hierarchical order.

```
<property>
    <name>topology.script.file.name</name>
    <value>/usr/local/hadoop/conf/rack_topology.sh</value>
    <description>This is the script that Hadoop will use to identify the
   network layout proposed by the network admin. Note that must be
   executed receiving just 1 IP as input parameter.</description>
</property>
```

Parity replication

- Parity replication was achieved through changes in the config files only. A better block placement policy could improve this step, because now it's Hadoop who decides where in the cluster the copies must be placed.

```
<property>
  <name>raid.config.file</name>
  <value>/media/HDFS/app/hadoop/conf/raid.xml</value>
  <description>This is needed by the RaidNode </description>
</property>
```

XORBAS codes:Lessons learnt

- XORBAS is a class of novel codes, based on locally repairable codes, that was built upon the Reed-Solomon codes in HDFS-RAID module.
- XORBAS results in $2x$ decrease in the repair traffic because of the idea of locality.
- Our experimental cluster had its nodes spread across five different locations around Australia.
- The source blocks and parity blocks were placed according to the default block placement policy of Hadoop.
- This is precisely why inspite of having local parities, a decrease of just 5% in the repair time was noticed.