

SHANGHAI JIAOTONG UNIVERSITY

BIG DATA PROCESSING

---

## Erasure code

---

*Author:*

WANG Lei  
118260910044  
WU Tiance  
117260910057  
GU Yuanzhe  
118260910034  
LIU Qingmin  
118260910037

*Supervisor:*

WU CHENTAO

November 30, 2019

# 1 Problem

Erasure code is a technique which works by storing redundant pieces of information in a way that allows recovery from complete storage device failures.

Compared to mirroring technology which duplicates entire sets of data, erasure code can offer the same reliability at a lower cost. It adds redundant data to each record, so that the original data can be retrieved from a subset of the record. In this project, we compile hadoop system in ubuntu and realize one of the classic erasure code: XOR.

# 2 Method

First of all, we need to compile hadoop system in ubuntu. The whole process and necessary package is listed in [https://github.com/ThorraySJTU/Big-Data-Processing/blob/master/Compile\\_hadoop\\_source\\_code.md](https://github.com/ThorraySJTU/Big-Data-Processing/blob/master/Compile_hadoop_source_code.md). After that, we modify the files in erasure code. The algorithm is put in catalogue "rawcoder".

- "XORRawEncoder.java" contains how we encode data block into erasure code block.
- "XORRawDecoder.java" contains how we recover data from defective data(erasure code) blocks.
- "XORRawErasureCoderFactory.java" contains the initialization steps for encoder and decoder.

Then, "XORErasureEncoder.java" and "XORErasureDecoder.java" in "coder" will package it and do prepare step.

Next, add new algorithm in "ErasureCodeConstants.java" so that hadoop can recognize new algorithm.

For convenience, We modified files "XORRawEncoder" and "XORRawDecoder" and realized XOR erasure code.

The principle of XOR is: if we have  $A, B, C$  where  $A \oplus B = C$ , If any one of them is missing, we can recover it by apply  $\oplus$  to left ones. For example, if  $A$  is missing,  $A = B \oplus C$ . And if two block are missing, XOR is not able to recover the original blocks.

The storage efficiency of XOR is about 150%. Because for each two block, one erasure block will be generated.

### 3 Experiment and Result

We test our erasure code with three data nodes and one name node. The plan is that

- create a directory and set its policy to XOR
- put a file into it
- check the distribution of blocks of file
- close one of the data nodes.
- recheck the distribution of blocks of file
- download the file and check its integrity

Input command

**hdfs ec -enablePolicy -policy XOR**

**hdfs ec -setPolicy -path /guyuanzhe -policy XOR-2-1-1024k**

to set the policy of directory.

Then we put the file into this directory.

Using command

**hdfs fsck /guyuanzhe/GU\_YUANZHE.mp4 -files -blocks -locations**

to check the physical locations of the file

From Figure 1, We can view that the file is divided into three blocks with different ip addresses: 192.168.1.148 192.168.1.142 192.168.1.114, which correspond to three data nodes. The algorithm works well and successfully divides the file.

Then, we open one of the data node and input command

**hdfs -daemon stop datanode**

in one of the data nodes. Then, we copy the file and download it. The file is not changed. It seems that file is not influenced by disconnection.

For more details, in Figure 3, we can view that there is one dead node. And in Figure 2, we can view that the distribution of blocks has changed, the recovered data is saved in 192.168.1.142.

```

xinyu@xinyu-GV62-8RD:~$ hdfs fsck /guyuanzhe -files -blocks -locations
Connecting to namenode via http://xinyu-GV62-8RD:9870/fuse?ugi=xinyu&files=1&blocks=1&locations=1&path=%2Fguyuanzhe
FSCK started by xinyu (auth:SIMPLE) from /192.168.1.174 for path /guyuanzhe at Fri Nov 29 22:56:27 CST 2019
/guyuanzhe <dir>
/guyuanzhe/GU_YUANZHE.mp4 251690600 bytes, erasure-coded: policy=XOR-2-1-1024k, 1 block(s): OK
0. BP-2010784536-127.0.1.1-1571670059960:blk_-9223372036854775728_1017 len=251690600 Live_repl=3 [blk_-9223372036854
775728:DatanodeInfromWithStorage[192.168.1.148:9866,05-bf4dec65-a7da-421b-a7c3-63b0730a95f4,DISK], blk_-922337203685477
5727:DatanodeInfromWithStorage[192.168.1.142:9866,05-12a3444a-4e0e-48ce-801c-65603e2dce53,DISK], blk_-92233720368547757
26:DatanodeInfromWithStorage[192.168.1.114:9866,05-63f2f70a-e5cc-467b-96d8-c251ebd123ae,DISK]]

Status: HEALTHY
Number of data-nodes: 3
Number of racks: 1
Total dirs: 1
Total symlinks: 0

Replicated Blocks:
Total size: 0 B
Total files: 0
Total blocks (validated): 0
Minimally replicated blocks: 0
Over-replicated blocks: 0
Under-replicated blocks: 0
Mis-replicated blocks: 0
Default replication factor: 2
Average block replication: 0.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0

Erasure Coded Block Groups:
Total size: 251690600 B
Total files: 1
Total block groups (validated): 1 (avg. block group size 251690600 B)
Minimally erasure-coded block groups: 1 (100.0 %)
Over-erasure-coded block groups: 0 (0.0 %)
Under-erasure-coded block groups: 0 (0.0 %)
Unsatisfactory placement block groups: 0 (0.0 %)
Average block group size: 3.0
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0 (0.0 %)
FSCK ended at Fri Nov 29 22:56:27 CST 2019 in 1 milliseconds

The filesystem under path '/guyuanzhe' is HEALTHY

```

Figure 1: Save file GU\_YUANZHE.mp4 with policy XOR

## 4 Future and Development

Because of the problems of compilation of hadoop system, to avoid floods of bugs, we only realize XOR. If we are capable to solve the problem of hadoop compilation, we may try to implement LRC, an erasure code based on RS. For encoding part, we need to add local parity  $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$  into RS. And for decoding part, if local parity is missing, we can recover it in the way of XOR. And if global parity is missing, we can reuse the recovery method of RS.

```

xinyu@xinyu-GV62-8RD:~$ hdfs fsck /guyuanzhe_test -files -blocks -locations
Connecting to namenode via http://xinyu-GV62-8RD:9870/fsck?ugi=xinyu&files=1&blocks=1&locations=1&path=%2Fguyuanzhe_test
FSCK started by xinyu (auth:SIMPLE) from /192.168.1.174 for path /guyuanzhe_test at Fri Nov 29 23:12:28 CST 2019
/guyuanzhe_test <dir>
/guyuanzhe_test/GU_YUANZHE.mp4 251690600 bytes, replicated: replication=2, 2 block(s): Under replicated BP-2010784536-127.
0.1.1-1571670059960:blk_1073741836_1020. Target Replicas is 2 but found 1 live replica(s), 0 decommissioned replica(s), 0 d
ecommissioning replica(s).
0. BP-2010784536-127.0.1.1-1571670059960:blk_1073741834_1018 len=134217728 Live_repl=2 [DatanodeInfoWithStorage[192.168.1.
148:9866,05-bf4dec65-a7da-421b-a7c3-63b0730a95f4,DISK], DatanodeInfoWithStorage[192.168.1.142:9866,05-12a3444a-4e8e-48cc-80
1c-65003e2dce53,DISK]]
1. BP-2010784536-127.0.1.1-1571670059960:blk_1073741836_1020 len=117472872 Live_repl=1 [DatanodeInfoWithStorage[192.168.1.
148:9866,05-bf4dec65-a7da-421b-a7c3-63b0730a95f4,DISK]]

Status: HEALTHY
Number of data-nodes: 2
Number of racks: 1
Total dirs: 1
Total symlinks: 0

Replicated Blocks:
Total size: 251690600 B
Total files: 1
Total blocks (validated): 2 (avg. block size 125845300 B)
Minimally replicated blocks: 2 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 1 (50.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 2
Average block replication: 1.5
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 1 (25.0 %)

Erasure Coded Block Groups:
Total size: 0 B
Total files: 0
Total block groups (validated): 0
Minimally erasure-coded block groups: 0
Over-erasure-coded block groups: 0
Under-erasure-coded block groups: 0
Unsatisfactory placement block groups: 0
Average block group size: 0.0
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0
FSCK ended at Fri Nov 29 23:12:28 CST 2019 in 1 milliseconds

The filesystem under path '/guyuanzhe_test' is HEALTHY

```

Figure 2: The distribution after disconnection

## Summary

Security is off.

Safemode is off.

8 files and directories, 3 blocks (2 replicated blocks, 1 erasure coded block groups) = 11 total filesystem object(s).

Heap Memory used 196.33 MB of 421 MB Heap Memory. Max Heap Memory is 3.45 GB.

Non Heap Memory used 64.75 MB of 66.27 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

<b>Configured Capacity:</b>	19.56 GB
<b>Configured Remote Capacity:</b>	0 B
<b>DFS Used:</b>	612.9 MB (3.06%)
<b>Non DFS Used:</b>	17.77 GB
<b>DFS Remaining:</b>	164.29 MB (0.82%)
<b>Block Pool Used:</b>	612.9 MB (3.06%)
<b>DataNodes usages% (Min/Median/Max/stdDev):</b>	2.50% / 3.62% / 3.62% / 0.56%
<b>Live Nodes</b>	2 (Decommissioned: 0, In Maintenance: 0)
<b>Dead Nodes</b>	1 (Decommissioned: 0, In Maintenance: 0)
<b>Decommissioning Nodes</b>	0
<b>Entering Maintenance Nodes</b>	0
<b>Total Datanode Volume Failures</b>	0 (0 B)
<b>Number of Under-Replicated Blocks</b>	2
<b>Number of Blocks Pending Deletion</b>	0
<b>Block Deletion Start Time</b>	Fri Nov 29 22:20:54 +0800 2019
<b>Last Checkpoint Time</b>	Fri Nov 29 22:22:02 +0800 2019

Figure 3: Summary of hadoop system after disconnection