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`
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 $A \bigoplus B = C$, If any one of them is missing, we can recover it by apply $\bigoplus$ to left ones.For example, if A is missing, $A = B \bigoplus C$. 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.

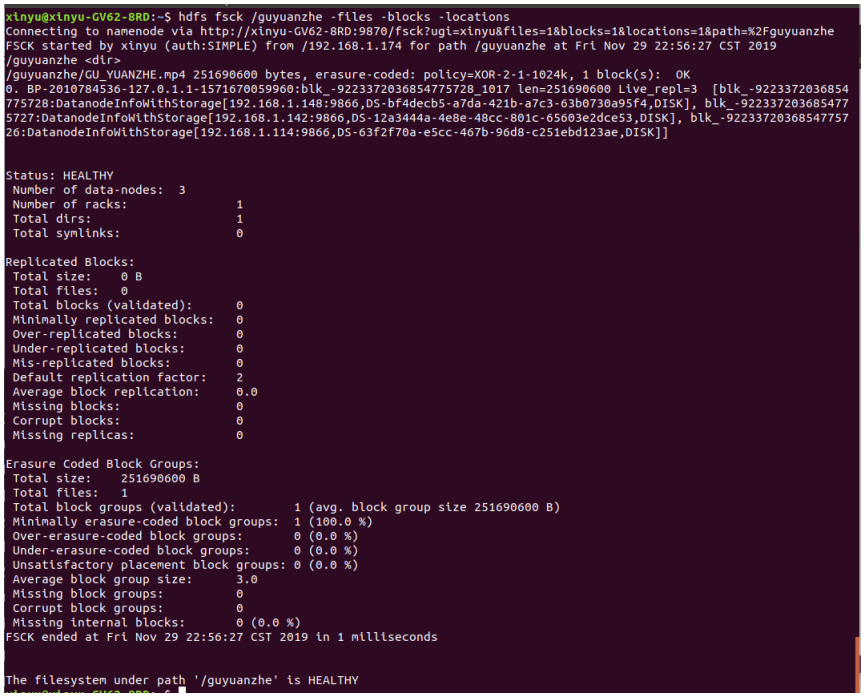


Figure 1: Save file GU_YUANZHE.mp4 with policy XOR

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.

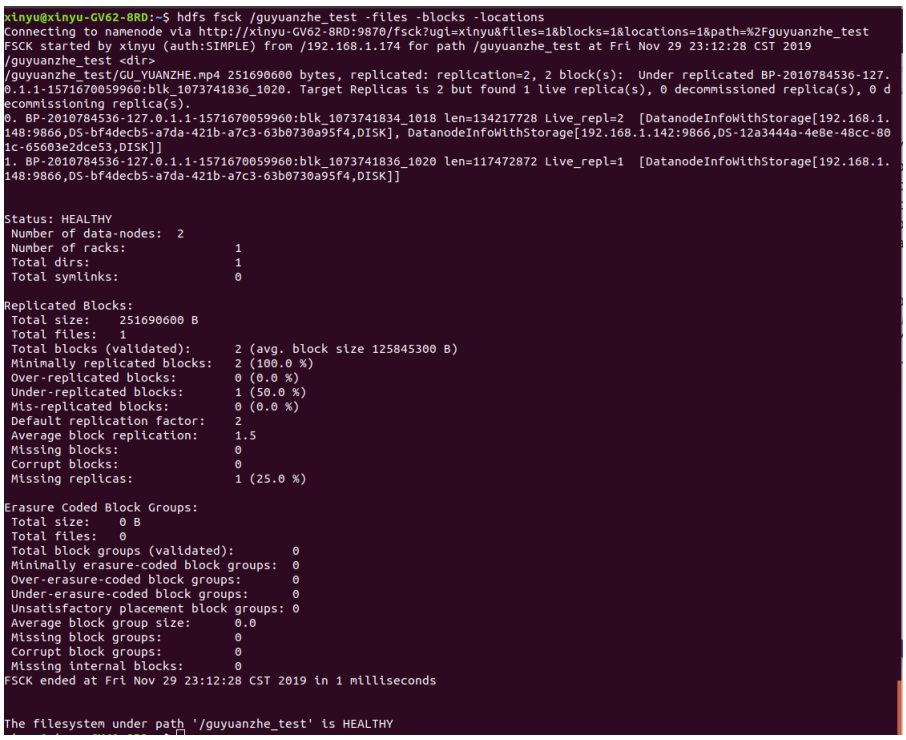
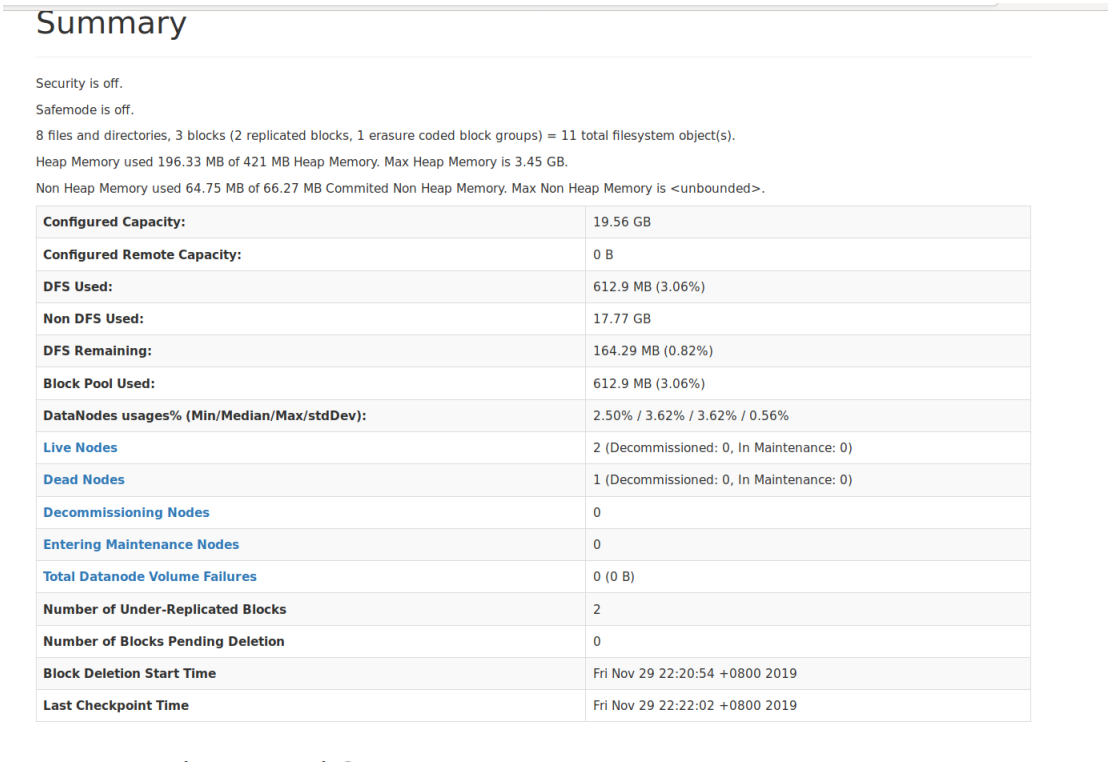


Figure 2: The distribution after disconnection



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 Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

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

Figure 3: Summary of hadoop system after disconnection