Distributed Dependable Systems

Assignment 2 - Erasure Coding

Benzi Matteo
matteo.benzi97@gmail.com

Firpo Tiziano
tiziano@firpo.me

April 2021

1 Introduction

The goal of this assignment was to study and analyze a backup application that use an erasure coding approach and different servers to keep safe the data as long as possible. In the specific we build a simulation such that a node split a file into N blocks, those blocks are scores between N servers using 2 different approach: 1 block per server and N block per server. The original file is safe until there is at least K blocks among the servers and the node.

2 Background

The simulation want to simulate the Polynomial Oversampling technique to split a file F into N different blocks, shareable between servers/memory devices, so that F is resistant up to M failure (i.e. you need K=N-M blocks to restore F).

The Polynomial Oversampling is an erasure coding technique which use a polynomial function f to encode a message $c_0...c_{n-1}$ into $f(x_0)...f(x_{n-1})$, so that it's possible to reconstruct the entire message knowing only K point of f $(x_i; y_i)$. **The simulator does not implement it and simulate only his behaviour.**

3 Summary

Initially the servers start without any block and the node have to upload his blocks to them.

The node can have no more than 1 block in download and 1 in upload. If one of those fails it loses all the progress and it has to restart the stream again.

The simulator takes multiple variables, if not specified the default values are:

variable	value
N	10
K	8
NODE_LIFETIME	30 days
NODE_UPTIME	8 days
NODE_DOWNTIME	16 days
DATA_SIZE	100 Gb
UPLOAD_SPEED	0.5 Mb
DOWNLOAD_SPEED	2 Mb
SERVER_LIFETIME	365 days
SERVER_UPTIME	30 days
SERVER_DOWNTIME	2 days
MAXT	100 years

4 Single block per server (default values)

In this modality each server can store only 1 block, so the node have to spreads his N blocks into N servers.

Executing the simulator with the default shows an interesting aspect:

Average	51.0137 years
Variance	±49.97 years
Min	0.002 years
Max	100 years

Table 1: Simulation executed 100 times

The huge variance shows that the "game overs" happen in the initial phase or at MAXT (100 years). The game over distribution confirms this behaviour:

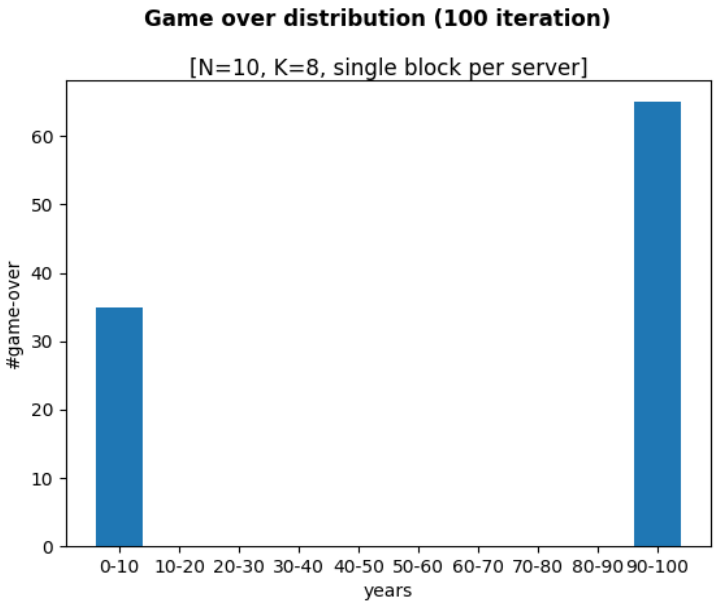


Figure 1: Distribution of resistance of the file using the default parameters (the data used to plot this graphic is not the same used for the previous table

As we can see the file is in real danger only in the first 10 years, after that it seems to be *safe* for long time. This is caused by the fact that in the initial phase only the node have all the blocks, and it could fails before the servers receive their block.

Another aspect is that the upload of the blocks is slow (UPLOAD_SPEED = 0.5) and has a huge "failure rate" due to the high value of NODE_UPTIME (i.e. the average time spent online).

This cause that often a block has to be uploaded again, losing a lot of time and increasing the probability to not have enough block uploaded to the server when the node fails.

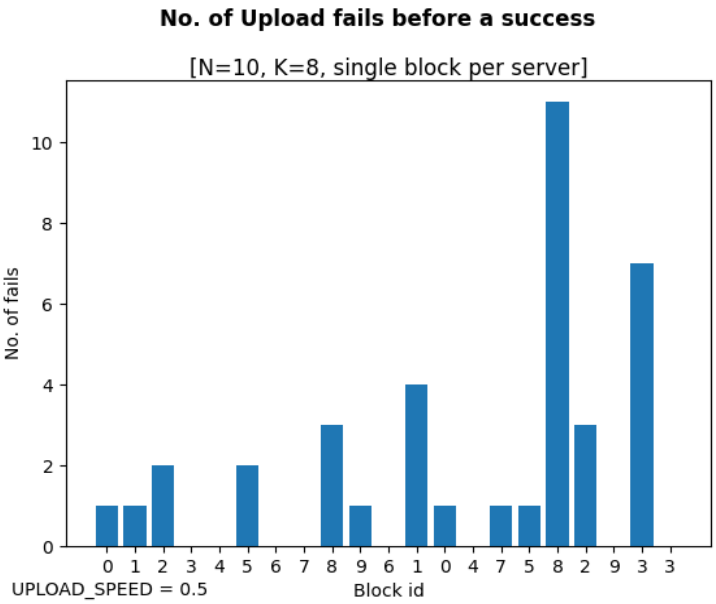


Figure 2: Number of upload failed before a success, when the number of failure is 0 means that the block has been correctly uploaded first try

5 Single block per server (analysis)

Using the default values is pointed out that the main problem is in the initial phase and it's caused by the high "failure rate" of the uploads. To see if decreasing this rate the situation gets better we tried to variate the DATA_SIZE (which means also increasing the connection speed):

Figure 3

It's clear that increasing the DATA_SIZE causes the average "game over time" to decrease, so a small time spent to transfer the blocks is a key to obtain a much more robust backup system.

Changing the number of blocks points out that there is an interval in which there is a better safeness for the data. This interval seems to be between 10 and 15, and the fact that growing too much the number of blocks could be derived by the fact that the node have to upload at least K blocks, and for consistency also K growth.

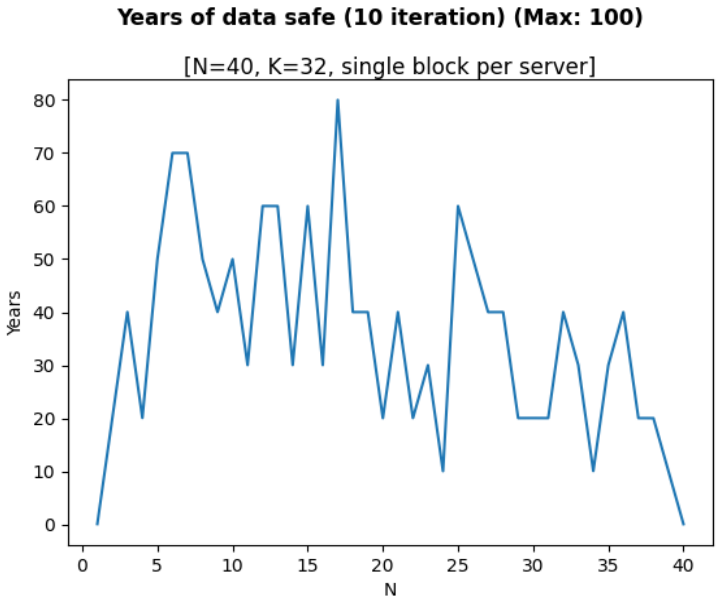


Figure 4: The value of K is calculated as 80% of N

Instead changing the mean lifetime of the node or the servers points out that the most important one is the NODE_LIFETIME. Increasing a bit the lifetime of the node drastically increase the safeness of the file, and decreasing the average lifetime of the servers didn't change a lot the result.

Figure 5: Mean lifetime in days, both graphic were computed using the default value of the other lifetime variable

By default the NODE_LIFETIME value is 30 days and the server one is an year, but if the values were inverted the result would be much better:

Average	88.005 years
Variance	±32.483 years
Min	0.012 years
Max	100 years

Table 2: Simulation executed 100 times

6 Multiple blocks per server (default values)

In this modality each server can store all the N blocks. Since the first uploading phase is critical, the node first try to upload all the blocks to at least one server, otherwise the result were too critical since it fails more or less every time in the first 30 days.

The distribution using the default values are more or less the same as the single block per server. This is caused for the same reason: the node fails before upload K blocks to the servers, and this doesn't depends on how many blocks the servers can store.

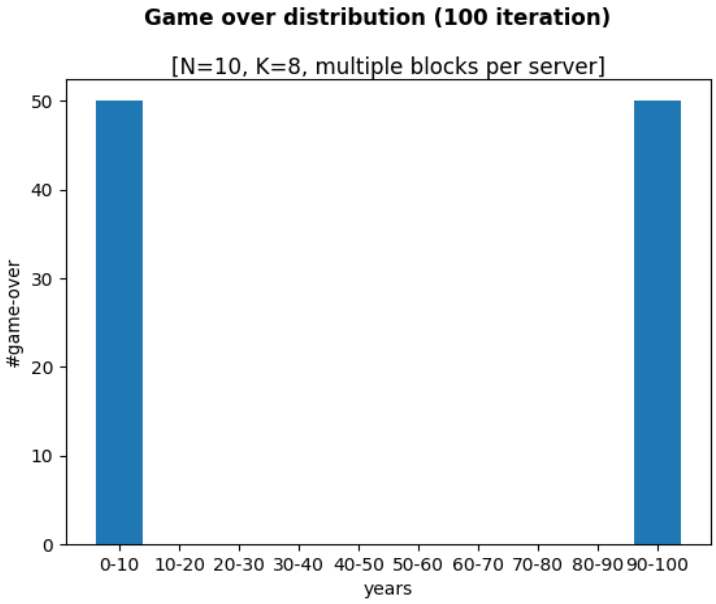


Figure 6: Distribution of resistance of the file using the default parameters

To be more precise, this implementation have a higher probability to fail in the first phase because the node uploads all the blocks to one server per time, so even if had uploaded all the blocks, if the node and the first server fails the file is lost. In the previous modality if this happen the file would be still safe.

This fact could be avoided if instead uploading all to one server per time, the node spreads all blocks to as many server as possible.

To see some difference between the previous modality, we decided to try to invert the lifetimes values of node and servers, to have NODE_LIFETIME = 365 days and SERVER_LIFETIME = 30 days. The result is a visible increment of the safeness of the file:

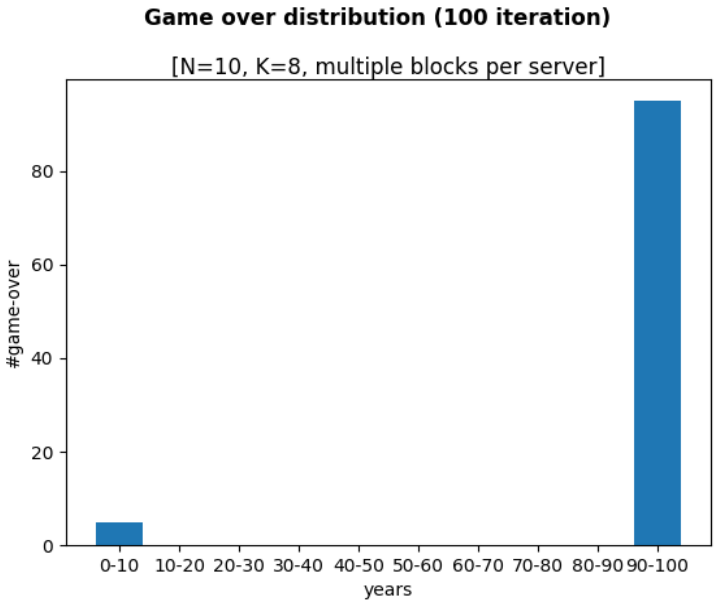
Average	96.004 years
Variance	±19.581 years
Min	0.0341 years
Max	100 years

Table 3: Simulation executed 100 times

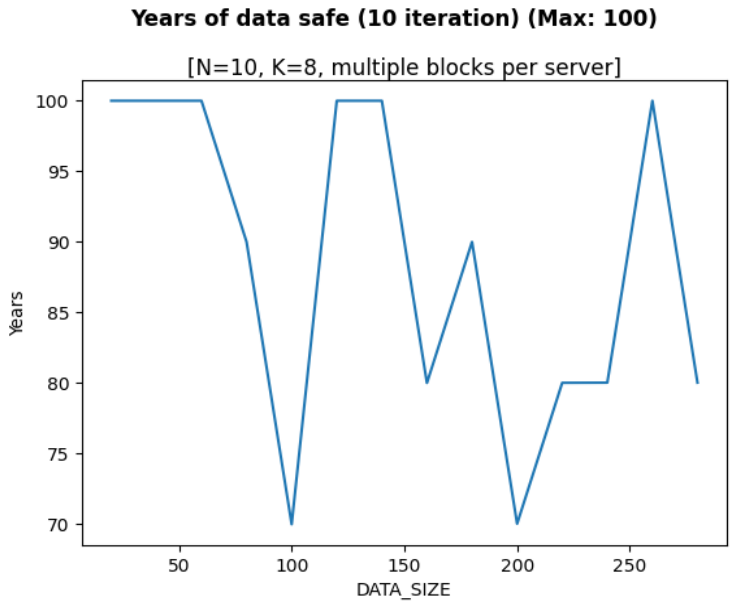
Also here sometimes fails in the first few days, but it's a quite strong improvement having the average very near to 100 and a variance that pass from ~32 to ~19.

7 Multiple blocks per server (analysis)

The distribution after the swap of the lifetimes clearly shows that the only thing that can put in crisis this backup system we found since now is the startup. This change only improve the result and seems that having N unstable servers but a single strong node don't create any other problem or weakness.

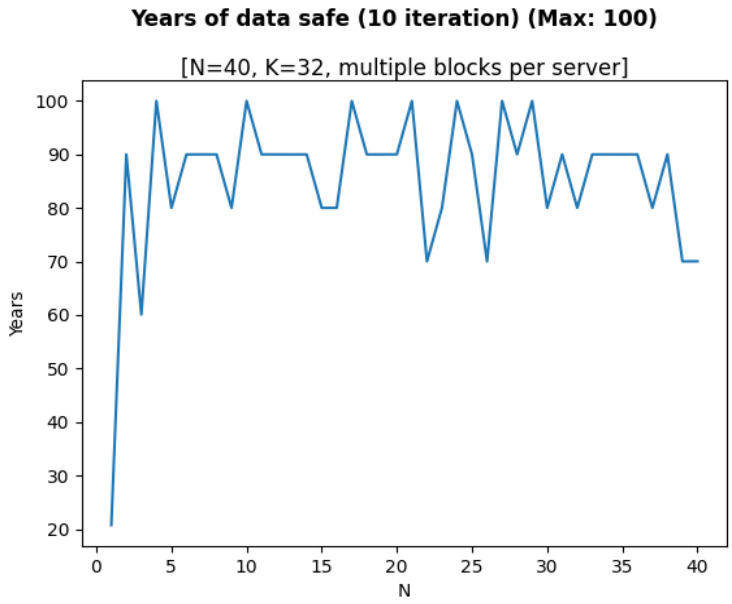


With this modality there are a much higher number of upload since the node needs to send N*N block to the servers at the beginning, and every time a server fails it has to upload again all the N blocks. So we decided to increasing the connection speed (or reducing the size of the blocks) to see if that affects somehow the result:



The size of the connection speed (or the dimension of the blocks) doesn't affect much the mean. This probably because after the initial phase even if the uploads or downloads fail many times, the file is still safe in other servers.

In this modality, also changing the number of blocks doesn't affect much either the results. Excepts for very low values of N, the average of years never falls down below 70 years.



8 Conclusion

In conclusion, the modality chosen for made the backup of a file could be not so important if some property is unlucky and in combo, for example a low node lifetime and a low connection speed put the 2 modalities on the same level because the majority of the fail happen in the first phase of uploading.

Anyway, store multiple time the same block in different server, as expected, on average increase the safeness of the file even in some "difficult" situation (like very low connection speed).

The strong problem pointed out is the initial phase, in which the node have to upload all his block to the servers, and if it fails the file is completely lost. Here a speedy connection, or a way to avoid the completely fail of an upload/download, could solve this problem. Another possibility is to increase the node lifetime. Even with a low server lifetime this ensure a much higher safeness for the file.