Erasure Coding and Replication for Edge Devices

Sheshadri K R

Department of Computational and Data Sciences
Indian Institute of Science, Bangalore 560012 INDIA
Email: sheshadrik@iisc.ac.in

1. Problem Definition

The study investigates the problem of providing a reliable distributed data store for an edge device deployment that makes use of the collective storage space of the connected edge devices.

In doing so following questions are addressed :

- What is the impact of using erasure coding to store data reliably at the edge devices? Considering that the edge devices have less compute resources, how does storage and compute trade off?
- How does the erasure coding affect the read and write performance? How slow/fast does it get compared to full replication?
- What is the impact of failure of Edge devices and the costs of recovery(time) in such cases? How does it fare against recovery using replication?
- Examine Reed Solomon Coding techniques with varying values of N and K, investigate its impact on storage efficiency and read/write performance. *

The above mentioned questions are validated through building a data store to provide distributed file storage in the Edge devices, whcih supports both replication and erasure coding.

2. Novelty and Scalability

The proposed scheme is called as Partial Replication, Partial erasure coding (PRPE), in this technique, the client which wants to store a file in the data store, will specify an associated **storage space budget**.

Given this space budget and the file, our job is to find out what is the optimal way of utilizing the storage budget by using both erasure coding and replication. The given file is split into a number of smaller blocks of identical size. Depending on the available storage budget, a fraction of the total blocks will be erasure coded and the rest will be replicated.

The good thing about this technique is that, irrespective of what percentage of file is erasure coded or replicated, the reliability of the file remains the same as a whole. The file will be fault tolerant for same number of edge failures despite using a combination of both the techniques.

This study is carried out on Fog and edge devices, where Fog device acts as a master and edge devices are the storage layer, and are managed by the Fog device. Edge devices are moderately resourced compared to the Fog devices.

To put the storage, main memory and compute in perspective for edge devices, we can consider Raspberry Pi 3 as an edge device which comes with a modest 1.2Ghz quad core processor, with around 1GB of main memory and 32GBs of SD card storage. Compare this to a desktop computer, which has around 3.4Ghz 8-core processor with 8GB of main memory and 500GB of disk storage (we can call this as Fog).

As far as scalability is concerned, as and when the requests for storing more files come in, adding more edge devices will increase the collective storage space , hence, allowing us to store the files in the data store. The system naturally is expected to scale with more edge devices joining in, the only bottleneck being the amount of main-memory in the master node (Fog device).

Here, it is shown that the PRPE technique works well with 10 and 20 edge devices. Though these are in no way close to the massive real world deployments. It provides a study of the *"compute vs storage"* trade off, which should serve as good indicator which aids in taking educated decisions while building a data store.

3. Goals and Non Goals

Here I outline the minimum goals I would want to achieve.

- Design and develop a data store a central master node and a set of edge devices which are used as storage nodes.
- The reliability of the file as a whole should remain the same despite the techniques used underneath.
- Evaluate the time taken for doing reads and writes for files, by varying the allotted budget of storage.
- Examine the effect of varying the parameters of the Erasure coding (N and K for the Reed Solomon Coding technique) *
- Compare this technique with full replication and full erasure coding.
- Evaluate the recovery times for a single and double failures.

- Peform evaluation of reads and writes of conccurent clients. *
- Peform evaluation of reads and writes of different file sizes. *

The non-goals of this project are:

- Design a global distributed file system for Wide Area Network which is ready for a real world wide area deployment.
- Do not expect this to work for generic workload of traditional servers, or act as a replacement of any sort.
- Do not expect to be an equivalent of cloud computing environment for the Edge layer.

4. Key Contributions

The key contribution of this work is to provide a hybrid technique to store data reliably, it takes both Erasure Coding and Replication and provides best of both the techniques. For a given file, in the PRPE technique , using the given storage budget, I try to optimally store portions of file in replicated way and the rest in an erasure coded way.

This study should shed light on the trade off of compute versus storage, and show how read/write performances vary from a completely compute intesive scenario to completely data I/O scenario.

The study also examines the time to perform recovery of files using the proposed PRPE technique.

5. Assumptions

In order to achieve the goals we make the following assumptions.

- All edge devices can talk to each other.
- All edge devices can talk to the master node and vice versa.
- Master node will not fail and is hosted in one of the desktop grade computers.
- Edge devices have enough storage to host all the writes.
- All the files stored in master node are in a flat namespace.
- No appends or edits to the data.
- Edge Devices will fail.
- The edge devices all have same reliability and it is assumed to be 90

  Some of these assumptions maybe relaxed later, it will be stated explicitly if and when they are relaxed.

6. Motivation and Related Work

6.1. Motivation

IoT devices have seen a remarkable acceleration in deployment in the market in the recent years due to the practical relevance of their applications, cheaper compute costs and reduced pricing of devices.

The IoT devices which are deployed in the real-world act as sources of data (Eg: A video surveillance camera may periodically send small video streams, a temperature sensor in a laboratory might constantly stream its data). Performing fast analytics on these data are one of the key things to provide quality services in applications.

Cloud data centers are one of the solutions for reliably storing such data but, it comes with an associated storage monetery costs and data access latency.

So, as much as possible, we would like to have data close to where it is generated to gain on read latency and to avoid unwanted movement of data.

One of the major concerns of storing data at the edge is that the edge devices are low on reliability. They are often placed at outdoor environments, subjected to heat, cold and winds. The chances of devices failing are quite high.

The need for low latency and the inherent low reliability of edge device deployments provide an interesing challenge in building a reliable storage at the edge layer.

One of the most popular techniques used to store the data reliably is Replication, where reliability is achieved by storing multiple copies of data. This is a simple and elegant solution and offers reliable storage against failures, often, the data is replicated thrice to achieve the needed reliability. It is evident that the benefits come from an additional storage cost.

The alternative technique has been Erasure Coding which provide same reliability in half the storage space [1], but trades the storage savings with CPU compute. Nevertheless, it provides fault tolerance in the same level as that of Replication.

While both these techniques work quite well for commodity machines in the LAN data center, it is interesting to see how well these techniques work on the Edge devices.

The problem space is examining replication and erasure coding at the edge devices.

Replication offers the best read/write performance since they don't involve any compute during read and write, but, uses up 3x the space of the original block (considering the blocks are stored in an uncompressed manner). On the other hand, erasure coding offers same reliability in nearly 1.5x (for eg: RS(6,4) erasure coding technique) the storage but with an additional compute costs.

Since edge devices is modest on storage and compute, it is interesting to see how the proposed PRPE techniques work for storing files in Edge devices, which is what is the current study is about.

6.2. Related Work

HDFS is a distributed file system, used extensively in Facebook, Yahoo and other companies. HDFS guarantees reliability of the files it stores using replication. So as long as one copy of the file is present and accessible, the file is available. Using replication to provide reliability incurs a lot of storage cost, we essentially use up 3 times the storage space compared to a single file. To reduce this extra storage space and still provide the same reliability, it started using Erasure Coding [2].

A lot of work has been done in comparing the performance of Erasure Coding with Replication in Local Area Network, but there's a need for a comprehensive investigation of read/write performance characteristics in edge devices.

Similarly, [3] Hyrax:Cloud Computing on Mobile Devices using MapReduce has looked into the aspects of porting the HDFS on the Android Devices, however, this hasn't looked into the aspects of erasure coding.

7. Problem Definition and Approach

Given a system comprising of a set of edge devices which are managed by a master node, *the objective is to use the collective storage of the connected Edge devices to provide reliable distributed storage for a given file with an associated storage budget using replication and erasure coding.*

Given a file, and an associated storage budget, the following equation decides how many blocks are to replicated and how many are to be erasure coded. *

$$Nr = (K/N) * SB * totalBlocks \tag{1}$$

$$Ne = totalBlocks - Nr \tag{2}$$

Where,

- $Nr \rightarrow$ number of blocks that can be replicated
- $Ne \rightarrow$ number of blocks that needs to be replicated
- $SB \rightarrow$ storage budget given by the user
- $totalblocks \rightarrow$ total number of blocks for a given file.

Using this we can calculate the number of blocks that need to be stored using erasure coding, as shown in eqn 2.

Note: N and K are the standard parameters in the Reed Solomon encoding techniques. [4]

The erasure coding technique that I will be using throughout the study is the Reed-Solomon Erasure Coding, RS(N,K) which tolerates upto 2 nodes failure, where N is the total number of coded blocks, K is the number of data blocks

The figure 1 shows a high level architecture of the proposed system, it is a simple client server model, with one master node managing multiple edge nodes. The master node has an in-memory data structure which will help the client perform reads() and writes().

As we can see the 3 major modules are:

- Master Node
- Clients
- Edge Devices

The master node holds a mapping of files to blocks, and blocks to edge nodes locations to support replication, and some additional amount of information to perform decoding of erasure coding blocks.

The Client is the originator of reads and writes, and it interacts with master node to write and read data.

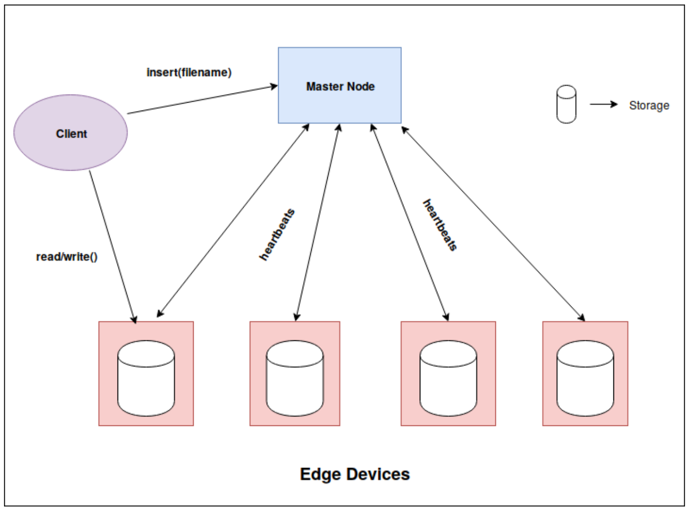

Figure 1: Master node and a set of Edge devices

The Edge devices are storage nodes, they will have a Erasure encoding/coding module , to perform encoding and writing of incoming block and decoder module to perform decoding of blocks to return a block to the client.

7.1. Proposed Approach and The APIs to implement

The initial bootstrapping happens as follows:

- The master node is deployed on a desktop grade system on a publicy known IP.
- A configuration file is distributed to all the edge devices so that they can join the master.
- The edge devices are deployed on containers which are configured to Edge device specifications, they connect to the central master node, and send frequent hearbeats and block reports.

After this the clients can issue read and write requests to the master server.

The APIs used to solve the problem is as follows:

7.2. Apis to implement

- **Master.open(filename)** : The *open()* api is called by the client on the master node, which wants to store a file in the system. It returns a sessionid, this is used as a key for subsequent communication of client with the master node.
- **Master.put(choice)** : The *put()* api is called by the client on the master node, which specifies the choice. the choice can be either replication or erasure coding. This api returns a list of block numbers and corresponding device locations for client to write to.
- **Master.write(data,blocknumber)** : This *write()* api is called by the client on the edge devices. This is to write the data on to the edge device directly by the client.

- **Master.getBlockNumbers(filename)** : The *getBlockNumbers()* api is called by the client on the master, this will return a set of block locations.
- **Master.get(blocknumber)** : The *get()* api is called by the client on the master,This is to read the data from the edge device directly by the client.
- **Master.close(sessionid)** : The *close()* api is called by the client on the master, This api is called after writing all blocks of data. The sessionid is used to identify the client and termination session beyond time outs. This api returns SUCCESS or FAILURE. If it is SUCCESS, the metadata is persisted at the master node.

7.3. Feasibility

The CPU speed , main memory and storage of the edge devices are sufficient to run the encoder/decoder module on the edge devices. In terms of specifications of the devices it is feasible to run the data node server on it.

7.4. Hardware and Software Requirements

To conduct these evaluations we need a set of 10 containers each to run the edge devices server, and a single node to run the master node.

The hardware requirements for the master node is. (this may change later, in such circumstances it'll be duly reported)

System Environment of Master Node(Indicative):

- **CPU :** Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
- **Total Installed RAM:** 8GB
- **CPU Architecture :** 64-bit
- **Number of Processing Cores :** 8
- **Java Version:** 1.8.0201
- **Message Exchange format:** JSON/Google Protobuf (TBD)
- **OS :** Linux Based Distribution

System Environment of Edge Devices (Indicative):

- **CPU :** Quad Core Intel Processor running at 1.2 GHz
- **Installed RAM:** 1 GB
- **CPU Architecture :** 64-bit
- **Number of Processing Cores :** 4
- **Java Version:** 1.8.0201
- **Message Exchange format:** JSON/Google Protobuf (TBD)
- **OS :** Linux Based Distribution
- **Storage :** 32GB/64GB

8. Experimental Design

8.1. Dataset for experiment

Random text files of size 10MB are generated. These files are used for put() experiments , to store files in the datastore. Random files of size 1MB and 5MB will also be generated and put(), get() and recovery evaluations will be performed on the same too. *

8.2. Experiments

I am planning to run the following experiments on 2 edge device deployments. The first deployment has 10 edge devices and a single master. The second deployment has 20 edge devices with a single master.

The experiments that will be carried are as stated below:

- Perform single client put() and get(), 100 put() of file size 10MB.
- Perform concurrent put() and get(), where multiple clients will simultaeneously perform put() and get() *
- Perform single client put() and get() experiments for different familes of Erasure Coding (i.e by varying values of N and K) *
- Measure the read/write latency of putting a file in the data store with full replication/full erasure coding.
- Measure the read/write latency of putting a file in the data store with repliation-erasure coding method conduct the experiments for the following set of values for storage budget 1.5x, 1.7x, 1.9x, 2.1x, 2.3x , 2.5x, 3.0x
- Plot the graph of how the read/write latency goes from 1.5x to 3x.
- Kill upto 2 nodes and recover all the blocks and measure the recovery times for various storage budgets.

9. Mid Term deliverables

- Build a simple client-server system to support Partial-Replication Partial-Erasure Coding(PRPE).
- Given a file and a storage budget, store it using the PRPE technique and compare the read/write performance against fully replicated system to fully erasure coded system.
- Compare the recovery times for single failure for PRPE, with fully replicated and fully erasure coded.

10. Finals Deliverables

- Examine the performance in the PRPE technique , for different families of Erasure Coding (by varying N and K) *
- Recovery for upto 2 node failures. *
- A study on read/write performances , where multiple clients can conccurently perform put() and get() *
- Examine the read/write performances for different file sizes.*

11. Strech Goals

Parallize the encoding and decoding components of Erasure coding, to improve the performance of read() and writes().

References

[1] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster." in *HotStorage*, 2013.

[2] M. Xia, M. Saxena, M. Blaum, and D. Pease, “A tale of two erasure codes in hdfs.” in *FAST*, 2015, pp. 213–226.

[3] E. E. Marinelli, “Hyrax: cloud computing on mobile devices using mapreduce,” Carnegie-mellon univ Pittsburgh PA school of computer science, Tech. Rep., 2009.

[4] J. S. Plank, “Erasure codes for storage systems: A brief primer,” *Login: The USENIX Magzine*, vol. 38, no. 6, pp. 44–50, 2013.