Reproducible Evaluation of Multi-level Erasure Coding

I. Introduction

Storage systems require data redundancy strategies such as replication and erasure coding (EC) to protect data from drive failures. As the scale, complexity, layering and disk capacity of distributed mass-capacity storage continue their unabated growth, both the frequency of drive failures and the time to rebuild a failed drive are growing. Therefore, a better data protection approach is needed.

Erasure coding is a commonly used approach to protect data against disk failures. In k+p EC, a stripe of data is divided into k small chunks, p parity chunks are generated, and then all these k+p chunks are placed in different disk locations. In this way, if one disk fails, the lost chunks can be reconstructed from other chunks in surviving disks.

Existing massive storage systems often adopt Single-level Erasure Coding (SLEC), either at the network-level or at the local-level. The local-level SLEC usually performs erasure coding locally using in-enclosure controllers, and all the chunks of a stripe are placed in different disks in the same enclosure. On the other hand, network-level SLEC performs erasure coding in a top-level server, and every chunk from the stripe is placed on a different disk in a separate rack. Both SLEC methods have their own pros and cons. For local-level erasure coding, although the repair can always be done locally without introducing repair network traffic (which is usually constrained in modern data centers), the main drawback is that it cannot tolerate enclosure/rack failures. On the other hand, the network-level SLEC is able to recover data from enclosure/rack failures, but the tradeoff is that the disk repair always introduces repair network traffic which can interfere with user network traffic on the foreground.

Due to the limitations of the SLEC approaches, Multi-level Erasure Coding (MLEC) starts to be favored by modern large-scale data centers such as in national laboratories like LANL. In MLEC, a stripe of data is first divided into large chunks by the top-level server, and each large chunk is distributed to an enclosure in a separate rack, where the large chunk is further divided into small chunks to compute local parities.

As a hybrid of the two SLEC approaches, MLEC gains the advantages from both sides, as it can repair most disk failures locally without interfering with foreground user network traffic, and meanwhile is able to tolerate enclosure/rack failures. Despite its increasing popularity, there have not been many systematic studies to analyze and evaluate MLEC. Therefore, many questions remain to be answered. For example, what are the possible design considerations/configurations of MLEC and what are their advantages and disadvantages in performance and durability? How is MLEC compared to SLEC, in terms of durability and erasure coding overheads such as encoding computation overheads and repair network traffic?

Since it's costly to deploy and evaluate different configurations of MLEC in a real data center, we propose to build a platform to reproducibly evaluate the performance and durability of MLEC and SLEC with different configurations such as chunk placement and coding parameter selections. We aim to make the evaluation reproducible such that other system designers can reproduce our evaluation results and meanwhile test their configurations according to their systems' requirements and constraints.

II. Project Goals

In this project, we propose to build a platform to reproducibly evaluate the performance and durability of MLEC for large-scale storage systems under different design configurations. We plan to do the evaluation through simulation, since it's costly to modify the configurations of a real large-scale system to see the effects.

The expected deliverables of this project will be:

- An MLEC simulator that can reproducibly simulate different configurations of the MLEC system, e.g. coding parameter selection, chunk placement scheme, repair method choice, etc.
- An analysis of the performance and durability tradeoffs between different MLEC design choices based on the evaluation results from the simulation
- Reproduced SLEC evaluation results using existing SLEC simulators
- A comparison between MLEC and SLEC on performance and durability tradeoffs
- Well-written documents and detailed guides on how to reproduce the evaluation results

We hope our simulator and evaluation results can provide designers of large-scale storage systems with valuable insights on choosing the most appropriate erasure coding configuration per their needs.

III. Implementation plan

I plan to implement the MLEC based on existing SLEC simulators such as SOL-Sim [1]. In order to do that, I will study the SOL-Sim to understand its implementation. I will read current papers about different erasure coding methods, especially understanding the implementation of MLEC. I will work on these tasks during my spring school quarter to get enough preparation and warm-up for the project.

During the summer, I will work full time with my mentors to carry out the design. I will schedule weekly meetings with my mentors to update my progress and problems and get feedback and help from them.

I would use my skills in C and python to build the simulator. I plan to implement the simulator using the Monte Carlo simulation method. The simulator should simulate the disk failures based on distributions or real traces, execute the repair behaviors of certain MLEC repair methods, and figure out the system status (i.e. is there any data loss) after each disk failure event. The simulation should be run for a large number of iterations in order to accurately estimate the durability of the system and report the average performance overheads of the repairs (such as repair time, repair network traffic, etc.)

IV. Project Timeline

We plan on working on this project over the summer (June - August.)
I plan to work 35 hours per week on it, for at least 10 weeks.
The planned timeline is as follows:

Time	Plan	Duration
6.12-6.18	Study and execute the existing SLEC simulator to reproduce its evaluation results.	1 week
6.19-7.19	Implement the basic logic of the MLEC simulator. Implement the simplest repair method and chunk placement of MLEC.	3 weeks
7.17-8.13	Implement other design choices of MLEC, e.g. different chunk placement schemes and repair methods.	4 weeks
8.14-8.20	Evaluate and analyze the performance and durability of different MLEC design choices and between MLEC and SLEC based on simulation results	1 week
8.21-8.27	Write the documents and guides for reproducing the evaluation results. Write the report.	1 week

V. Biographical Information

I am currently a second year undergraduate student at the University of Chicago, majoring in Computer Science and Mathematics. I am very interested in storage systems, and have taken classes about operating systems and network securities. I am familiar with basic concepts about storage systems, fluent in C and python, and also know probability theories. I've also had experiences from my Operating system class in building and revising a kernel.

I am dedicated to this project because I have strong interests in storage systems . Moreover, as someone who is new to system research, I want to gain more exposure to the field and the research process. I believe that I can bring in my experience in system designs to implement the simulators.

Below is my contact information:

Name: Zhiyan “Alex” Wang

Email: zhiyanw@uchicago.edu

Current affiliation: University of Chicago

[1] Ke, Huan, et al. "Extreme protection against data loss with single-overlap declustered parity." 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2020.