Experiences Building Network-Coding-Based Distributed Storage Systems

Patrick P. C. Lee
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
Email: pclee@cse.cuhk.edu.hk

I. INTRODUCTION

Large-scale distributed storage systems are prone to node failures. To provide fault tolerance, data is often encoded to maintain data redundancy over multiple storage nodes. If a node fails, it can be repaired by downloading data from surviving nodes and regenerating the lost data in a new node. Network coding has recently been proposed (e.g., see [2]) to generate data redundancy. It is shown that network coding can minimize the amount of data being transferred for repair, while maintaining the same fault tolerance as in conventional erasure coding schemes. Its idea is to have storage nodes first encode their stored data and then send the encoded data for regeneration.

On the other hand, the topic of network coding in storage systems is mostly investigated in theoretical studies. Its performance in real deployment remains an open issue. This motivates us to study the practicality of deploying network coding in real-world distributed storage systems.

We highlight two of our implementation projects of network-coding-based storage systems at the Chinese University of Hong Kong, namely NCCloud [3] and CORE [5]. Both of them target different storage applications, while building on network coding to enable high availability and efficient recovery of storage systems.

II. NCCLOUD

We first consider an application of network coding in multiple-cloud storage. By striping data across multiple cloud storage vendors, we mitigate the single-point-of-failure and vendor lock-in problems as seen in single-cloud storage. We design and implement NCCloud [1], [3], a proxy system that realizes the benefits of network coding over multiple cloud storage vendors. NCCloud targets long-term archival storage applications, in which data is rarely accessed but needs to be persistently stored for a long period of time.

NCCloud builds on an implementable design of a coding scheme called functional minimum storage regenerating (FMSR) codes. Our FMSR code implementation maintains double-fault tolerance and has the same storage overhead as in the conventional RAID-6 scheme, but incurs less repair traffic when recovering a single-cloud failure. The reduction of repair traffic is up to 50%. One key feature of FMSR codes is that they eliminate the need of performing encoding operations in storage nodes, while preserving the benefits of network coding in minimizing the amount of repair traffic. The trade-off is that unlike most erasure coding schemes that are systematic (i.e., original data chunks are kept), FMSR codes are non-systematic and store only linearly combined code chunks. Thus, FMSR codes introduce additional computational decoding overhead to recover the original data, yet they will be suited to long-term archival storage applications where data is rarely accessed. The correctness of FMSR codes is also theoretically proven [4].

We conduct testbed experiments on NCCloud and validate the practicality of FMSR codes. We also conduct cost analysis on FMSR codes. We show that while cloud failures are rare, the monetary benefits brought by FMSR codes in unexpected repair events can be significant. Detailed results are found in the papers [1], [3].

III. CORE

We next consider an application of network coding on clustered or distributed storage systems, where data can be frequently read. We design and implement CORE [5], a system that augments existing optimal regenerating codes to support a general number of failures including single and concurrent failures. CORE targets the write-once-read-many (WORM) model, in which data permits unlimited accesses, but cannot be modified once written.

The motivation of CORE is that node failures are often correlated and co-occurring in large-scale storage systems in practice. Existing regenerating codes often focus on optimizing single failure recovery. CORE’s goal is to augment minimum storage regenerating (MSR) codes to support optimal recovery for both single and concurrent failures. Its idea is to construct a linear equation for reconstructing the lost data for each of the failed nodes, using the existing single failure recovery mechanism of MSR codes. By solving the system of linear equations for all failed nodes, we reconstruct the lost data for all failed nodes. We also deal with the technical details if the system of linear equations cannot return a unique solution. We show that in over 98% of cases, CORE minimizes the amount of repair traffic (i.e., achieving the optimal point) for a general number of failures. Note that CORE minimizes the existing coding construction of MSR codes. Thus, it can build on systematic MSR codes (e.g., Interference Alignment codes [8] and Product-Matrix codes [6]), which keep original data blocks in storage.
We implement CORE on Hadoop Distributed File System [7] and conduct testbed experiments to take into account different factors including network bandwidth, disk I/Os, and encoding/decoding overhead. Our experience is that we can mitigate the encoding/decoding overhead through extensive multi-threading. Thus, minimizing the amount of repair traffic being transferred plays a key role in improving the overall recovery performance. Detailed results are found in the paper [5] and its technical report.

REFERENCES


