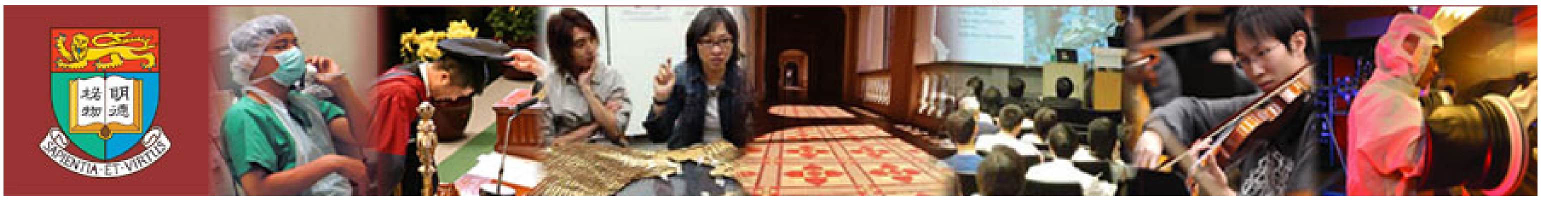

Flexible Download Time Analysis of Coded Storage Systems

Qiqi Shuai Victor O.K. Li
Electrical and Electronic Engineering Department, The University of Hong Kong
{qqshuai, vli}@eee.hku.hk

Categories and Subject Descriptors D.4.3 [*File Systems Management*]: Access methods

Keywords Systematic code, MDS property, Flexible read, Download latency

1. Introduction and Motivation

Download time is a key performance metric in distributed storage systems since it greatly impacts user experience, especially for latency-sensitive applications such as Google Search and so on. Recently, plenty of research has pointed out that coding can reduce download time. Till now, almost all previous studies analyze download time when a user requires all the information in a codeword. However, in practical storage systems such as the Windows Azure Storage System (WAS), only when files reach a certain size (e.g., 1GB), will it be a candidate for erasure coding [1]. That is, in practice, files stored in a codeword are usually very large and users' requests may only desire part of these files. Therefore, it is significant to analyze the latency performance when users only request a subset of the erasure-coded content.

2. Model Description

We focus on a systematic (n, k) MDS-coded storage system, in which each codeword consists of k data nodes and $n - k$ parity nodes and any k out of the n nodes can reconstruct all the information in the codeword.

In this work, we propose a method called flexible read which not only takes advantage of both direct and k-access reads [2] but also reduces latency flexibly according to users' required size of files from a codeword. With flexible reads, when a read request which desires d out of the k data nodes arrives, it is sent to all the n nodes. The request is complete if the d desired data nodes finish their services or any k out of the n nodes finish their services and the unfinished read tasks are cancelled immediately.

SYSTOR '16, June 06-08, 2016, Haifa, Israel
ACM 978-1-4503-4381-7/16/06.
http://dx.doi.org/10.1145/2928275.2933278

3. Performance Evaluation

We evaluate the latency performance of flexible reads with real service time traces from Amazon S3. These traces are for reading files of 1MB in size from an S3 bucket, located in the Northern California. We take the download latency over 1 million sample paths for each experiment.

We first compare the average latency of direct, k-access and flexible read methods. We observe that flexible reads can always achieve the best latency performance, and when $d = k$, flexible read is equivalent to k-access read. It is also noted that the average latency of direct reads is much higher than that of the others, especially when d is big. This is because a direct read needs all the d required nodes to finish their services while k-access and flexible reads enjoy the diversity benefit of coding, which results in decreased average latency.

Then we present simulation results demonstrating the fundamental tradeoffs between storage cost, required number of nodes d, average latency and latency volatility with flexible reads. Fixing $n = 10$, as k increases, the storage cost decreases while the average latency increases. As k increases, the possible number of sets of k out of the n nodes decreases, thus losing the diversity benefit of coding and leading to higher latency. When d increases, the average latency also increases since bigger d results in higher latency for the d required nodes and causes higher average latency for flexible reads. Besides, we also observe that, when $n = k$ the latency volatility increases as d increases while latency volatility decreases as d increases when $k < n$.

These tradeoffs can be used to meet the latency constraints on content download in distributed storage systems.

References

[1] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin, et al. Erasure coding in Windows Azure storage. In *USENIX ATC*, pages 15–26, 2012.

[2] Q. Shuai and V. O. Li. Delay performance of direct reads in distributed storage systems with coding. In *the 17th International Conference on High Performance Computing and Communications (HPCC)*, pages 184–189. IEEE, 2015.