

DOCTORAL THESIS

ESetStore: an erasure-coding based distributed storage system with fast data recovery

Liu, Chengjian

Date of Award:
2018

[Link to publication](#)

General rights

Copyright and intellectual property rights for the publications made accessible in HKBU Scholars are retained by the authors and/or other copyright owners. In addition to the restrictions prescribed by the Copyright Ordinance of Hong Kong, all users and readers must also observe the following terms of use:

- Users may download and print one copy of any publication from HKBU Scholars for the purpose of private study or research
- Users cannot further distribute the material or use it for any profit-making activity or commercial gain
- To share publications in HKBU Scholars with others, users are welcome to freely distribute the permanent URL assigned to the publication

Abstract

The past decade has witnessed the rapid growth of data in large-scale distributed storage systems. Triplication, a reliability mechanism with 3x storage overhead and adopted by large-scale distributed storage systems, introduces heavy storage cost as data amount in storage systems keep growing. Consequently, erasure codes have been introduced in many storage systems because they can provide a higher storage efficiency and fault tolerance than data replication. However, erasure coding has many performance degradation factors in both I/O and computation operations, resulting in great performance degradation in large-scale erasure-coded storage systems.

In this thesis, we investigate how to eliminate some key performance issues in I/O and computation operations for applying erasure coding in large-scale storage systems. We also propose a prototype named ESetStore to improve the recovery performance of erasure-coded storage systems. We introduce our studies as follows.

First, we study the encoding and decoding performance of the erasure coding, which can be a key bottleneck with the state-of-the-art disk I/O throughput and network bandwidth. We propose a graphics processing unit (GPU)-based implementation of erasure coding named G-CRS, which employs the Cauchy Reed-Solomon (CRS) code, to improve the encoding and decoding performance. To maximize the coding performance of G-CRS by fully utilizing the GPU computational power, we designed and implemented a set of optimization strategies. Our evaluation results demonstrated that G-CRS is 10 times faster than most of the other coding libraries.

Second, we investigate the performance degradation introduced by intensive I/O

operations in recovery for large-scale erasure-coded storage systems. To improve the recovery performance, we propose a data placement algorithm named ESet. We define a configurable parameter named overlapping factor for system administrators to easily achieve desirable recovery I/O parallelism. Our simulation results show that ESet can significantly improve the data recovery performance without violating the reliability requirement by distributing data and code blocks across different failure domains.

Third, we take a look at the performance of applying coding techniques to in-memory storage. A reliable in-memory cache for key-value stores named R-Memcached is designed and proposed. This work can be served as a prelude of applying erasure coding to in-memory metadata storage. R-Memcached exploits coding techniques to achieve reliability, and can tolerate up to two node failures. Our experimental results show that R-Memcached can maintain very good latency and throughput performance even during the period of node failures.

At last, we design and implement a prototype named ESetStore for erasure-coded storage systems. The ESetStore integrates our data placement algorithm ESet to bring fast data recovery for storage systems.

Keywords: Erasure coding, Storage System, ESet, R-Memcached, ESetStore

Table of Contents

Declaration	i
Abstract	ii
Acknowledgements	iv
Table of Contents	v
List of Tables	ix
List of Figures	x
Chapter 1 Introduction	1
1.1 Notation and Nomenclature	3
1.2 Erasure Coding	4
1.3 Recovery of Erasure-coded Storage Systems	6
1.4 In-memory storage	7
1.5 Thesis goals and contributions	8
1.6 Organization	9
Chapter 2 Background and Related Work	11
2.1 Erasure coding	11
2.2 GPU Computing	12
2.3 Recovery of Erasure-Coded Storage	14
2.4 Reliability of In-memory Storage	17

Chapter 3	G-CRS: GPU Accelerated Cauchy Reed-Solomon Coding	19
3.1	Introduction	20
3.2	Cauchy Reed-Solomon Coding	20
3.3	Design of G-CRS	22
3.3.1	Baseline Implementation	23
3.3.2	Optimization Strategies	26
3.4	Performance Model	35
3.4.1	Kernel Analysis	35
3.4.2	Dominant Factor Analysis	37
3.5	Pipelined G-CRS	39
3.6	Performance Evaluation	42
3.6.1	Throughput Under Different Workloads	43
3.6.2	Peak Raw Coding Performance	45
3.6.3	Optimization Analysis	47
3.6.4	Overall Performance	50
3.7	Summary	50
Chapter 4	ESet: Placing Data towards Efficient Recovery for Large-scale Erasure-Coded Storage Systems	52
4.1	Problem Definition	52
4.1.1	System Model	53
4.1.2	Problem Illustration	54
4.1.3	Problem Formulation	55
4.2	Reliability Analysis	55
4.2.1	Revisiting Failures	55
4.2.2	Our Solution: ESet	56
4.2.3	Reliability Constraint	60
4.3	Design of ESet	61
4.3.1	Grouping for Reliability	62
4.3.2	Generation of <i>ESets</i>	63

4.3.3	Recovery of a Failed Host	65
4.4	Performance Evaluation	66
4.4.1	Evaluation Overview	66
4.4.2	Recovery I/O Parallelism Analysis	67
4.4.3	Recovery Performance of Simulating A Year Failure	68
4.4.4	Recovery Performance of With Different λ Values	70
4.4.5	Recovery Performance of Burst Failures in An Hour	71
4.5	Summary	71

Chapter 5 R-Memcached: a Reliable In-Memory Cache for Big Key-Value Stores **72**

5.1	Background	73
5.1.1	Introduction to Memcached	73
5.1.2	Reliability Challenge	75
5.2	Design and Implementation of R-Memcached	75
5.2.1	System Architecture	76
5.2.2	RAIM Implementation	79
5.2.3	<i>Set</i> , <i>Get</i> and <i>Delete</i> in R-Memcached	80
5.2.4	Asynchronous Update and Degrade Read	82
5.3	Reliability Analysis	86
5.3.1	RAIM Set Reliability	87
5.3.2	Reliability of a R-Memcached Cluster	88
5.4	Performance Evaluation of R-Memcached	90
5.4.1	Testbed and performance baseline	91
5.4.2	Evaluation of RAIM-1	93
5.4.3	Evaluation of RAIM-5	94
5.4.4	Evaluation of RAIM-6	95
5.5	Summary	96

Chapter 6 ESetStore: Introducing Fast Data Recovery to the Erasure-	
 Coded Storage System	98
6.1 System Architecture of ESetStore	98
6.2 The Design and Implementation of ESetStore	100
6.2.1 ECMeta: the Metadata Service	100
6.2.2 Efficient Read and Write Operations	103
6.2.3 Fast Recovery with <i>ESet</i>	107
6.3 Evaluation	109
6.3.1 Experimental Setup	109
6.3.2 Read and Write Throughput	110
6.3.3 Recovery Performance	113
6.3.4 Recovery Performance with PPR	114
6.4 Summary	116
Chapter 7 Conclusions	117
7.1 Future Research Directions	118
Bibliography	120
Curriculum Vitae	132