Erasure Coding in Windows Azure Storage

Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin

Microsoft Corporation

USENIX ATC, Boston, MA, June 2012
Outline

• Introduction to Windows Azure Storage (WAS)

• Conventional Erasure Coding in WAS

• Local Reconstruction Coding in WAS
North America Region
- West U.S. Sub-region
- N. Central U.S. Sub-region
- S. Central U.S. Sub-region
- East U.S. Sub-region

Europe Region
- N. Europe Sub-region
- W. Europe Sub-region

Asia Pacific Region
- E. Asia Sub-region
- S.E. Asia Sub-region
Windows Azure Storage

• Abstractions
  • **Blobs** – File store in the cloud
  • **CDN** – High performance file delivery through proximity caching
  • **Drives** – Durable NTFS volumes for Windows Azure applications
  • **Tables** – Massively scalable NoSQL storage
  • **Queues** – Reliable storage and delivery of messages

• Easy client access
  • Easy to use REST APIs and Client Libraries
  • Existing NTFS APIs for Windows Azure Drives
Massive Distributed Storage Systems in the Cloud

- Failures are norm rather than exception
- As the number of components increase, so does the probability of failure
  \[ MTTF_{First} = \frac{MTTF_{One}}{n} \]
- Redundancy is necessary to cope with failures
- Replication vs. Erasure Coding?
Replication vs. Erasure Coding

Replication

a=2
b=3

Erasure Coding

b=3
a=2
b=3
2a+b=7
a+b=5

a=2
WAS Stream Layer

- Append-Only Distributed File System
- Provides replication inside a stamp
- Streams are very large files
  - Has file system like namespace
  - Ordered list of pointers to extents

- Extents
  - Unit of replication
  - Sequence of blocks
  - Size target (3GB), unsealed/sealed
Replication and Erasure Coding

- Extents triple replicated
  - when first created
  - and while being appended
- Extents sealed at around 3GB
  - Erasure coded in the background
  - When erasure coding finishes, full replicas are deleted
  - Policies to choose between replication, erasure coding, or a mix
Conventional Erasure Coding – Reed-Solomon 6+3

sealed extent (3 GB)

\[ d_0, d_1, d_2, d_3, d_4, d_5 \]

\[ p_1, p_2, p_3 \]
Designing For Erasure Coding - 1

- Arithmetic for Erasure Coding
  - Direct use of Galois Field operations is costly
  - Use bit-matrix and XOR transformations
- IO scheduling
  - Reconstruction/recovery/on-demand traffic need to be prioritized and throttled carefully
- Data consistency
  - Checksum handoff and verification between all levels
  - Scrub periodically
Designing For Erasure Coding - 2

- Efficient/fast on-demand reads
  - Reconstructing larger blocks for reuse
- Replica Placement for reliability
  - Each replica or fragment for an extent placed in independent fault domains
  - Replicas/fragments are placed across upgrade domains to keep high availability during rolling upgrades
Space Savings with RS 6+3 (over 3-replication)
How to Further Reduce Storage Cost?

\[(6+3)/6 = 1.5x\]

sealed extent (3 GB)

overhead

\[d_0 \quad d_1 \quad d_2 \quad d_3 \quad d_4 \quad d_5\]

\[p_0 \quad p_1 \quad p_2\]
How to Further Reduce Storage Cost?

sealed extent (3 GB)

overhead

\[
\frac{(6+3)}{6} = 1.5x
\]

\[
\frac{(12+4)}{12} = 1.33x
\]
reconstruction read is expensive – reading $d_0$ during unavailability
→ requiring 12 fragments (12 disk I/Os, 12 net transfers)
Reconstruction Read – When?

• Load balancing
  • avoid hot storage node → serve reads via reconstruction

• Rolling upgrade

• Transient unavailability and permanent failures

Can we achieve 1.33x overhead while requiring only 6 fragments for reconstruction?
Opportunity

- Conventional EC
  - all failures are equal $\rightarrow$ same reconstruction cost, regardless of failure #
- Cloud storage
  - Prob(1 failure) $\gg$ Prob(2 or more failures)

optimize erasure coding for cloud storage
$\rightarrow$ making single failure reconstruction most efficient
• LRC\(_{12+2+2}\): 12 data fragments, 2 local parities and 2 global parities
  • storage overhead: (12 + 2 + 2) / 12 = 1.33x
• Local parity: reconstruction requires only 6 fragments
One More Thing – Ensuring Reliability in LRC

- $\text{LRC}_{12+2+2}$ needs to recover
  - arbitrary 3 failures
  - as many 4 failures as possible
Recover 3 Failures – Local Recovery

recover $y_1$ from $p_y$ (group $y$)
Recover 3 Failures – Global Recovery

recover $y_1$ from $p_y$ (group $y$)
recover $x_0$ and $x_2$ from $q_0$ and $q_1$
Recover 4 Failures – More Challenging

how to recover the 4 failures and all similar cases?
(see paper)
Properties of LRC

• Achieving recovery limit
  • $LRC_{12+2+2}$: arbitrary 3 failures and 86% of 4 failures
  • reliability: $RS_{12+4} > LRC_{12+2+2} > RS_{6+3}$

• Requiring minimum storage overhead, given
  • reconstruction cost
  • fault tolerance
    • separate paper to appear in IEEE Trans. on Information Theory
Cost & Performance Analysis

• Vary LRC parameters → trade-off points in 3D space
  • storage overhead
  • reconstruction cost
  • reliability (MTTDL)

• Reliability is a hard requirement
  • set MTTDL_{3-replication} as target
  • reduce trade-off space to 2D
LRC vs. Reed-Solomon

- **RS**
- **LRC**

**RS**
- **RS_{10+4}**
- **RS_{6+3}**

**LRC**
- **LRC (12+2+2)**
- **LRC (12+4+2)**

- **RS_{10+4}**: HDFS-RAID at Facebook
- **RS_{6+3}**: GFS II (Colossus) at Google

**Reconstruction Read Cost**

**Storage Overhead**

- same cost
- 1.5x → 1.33x
- same overhead
- half cost (6 → 3)

- **same cost**
- **1.5x → 1.33x**
- **same overhead**
- **half cost (6 → 3)**
Choice of Windows Azure Storage

RS (6 + 3)
reconstruction cost = 6

RS (14 + 4)
reconstruction cost = 14

LRC (14 + 2 + 2)
reconstruction cost = 7

14% savings
Summary

• Erasure coding enables significant storage cost savings in Windows Azure Storage with higher reliability than 3-replication.

• LRC achieves additional 14% savings without compromising performance.

• Windows Azure Storage Team Blog