Efficient and Reliable Storage Solutions

Jin Li

Microsoft Corporation
Redmond (USA)

Outline

- How I went into the area of storage research
- Local Reconstruction Coding for Windows Azure Storage
- Primary Data Deduplication for Windows Server 2012

How I Went Into the Area of Storage System Research

How I Went Into the Area of Storage Research

- ~9-10 years ago, I was still actively engaged in scalable audio/image/video coding research & P2P communication research
- To market my technology, I attended Microsoft Technical Community Network (TCN)

What I Observed

- Storage Trend
  - Multimedia (audio/image/video) are dominant content in cloud storage
  - 60% growth in file based storage from 2001-2015 [IDC]
  - Rising total cost of ownership (TCO) of storage despite declining hard disk drive (HDD) cost
  - Customers show continued willingness to pay for storage solution rather than throw away data or storing data with loss of fidelity
- Two technologies in reducing TCO for data
  - Erasure coding for Windows Azure Storage
  - Primary data deduplication in Windows Server 2012

Local Reconstruction Coding for Windows Azure Storage

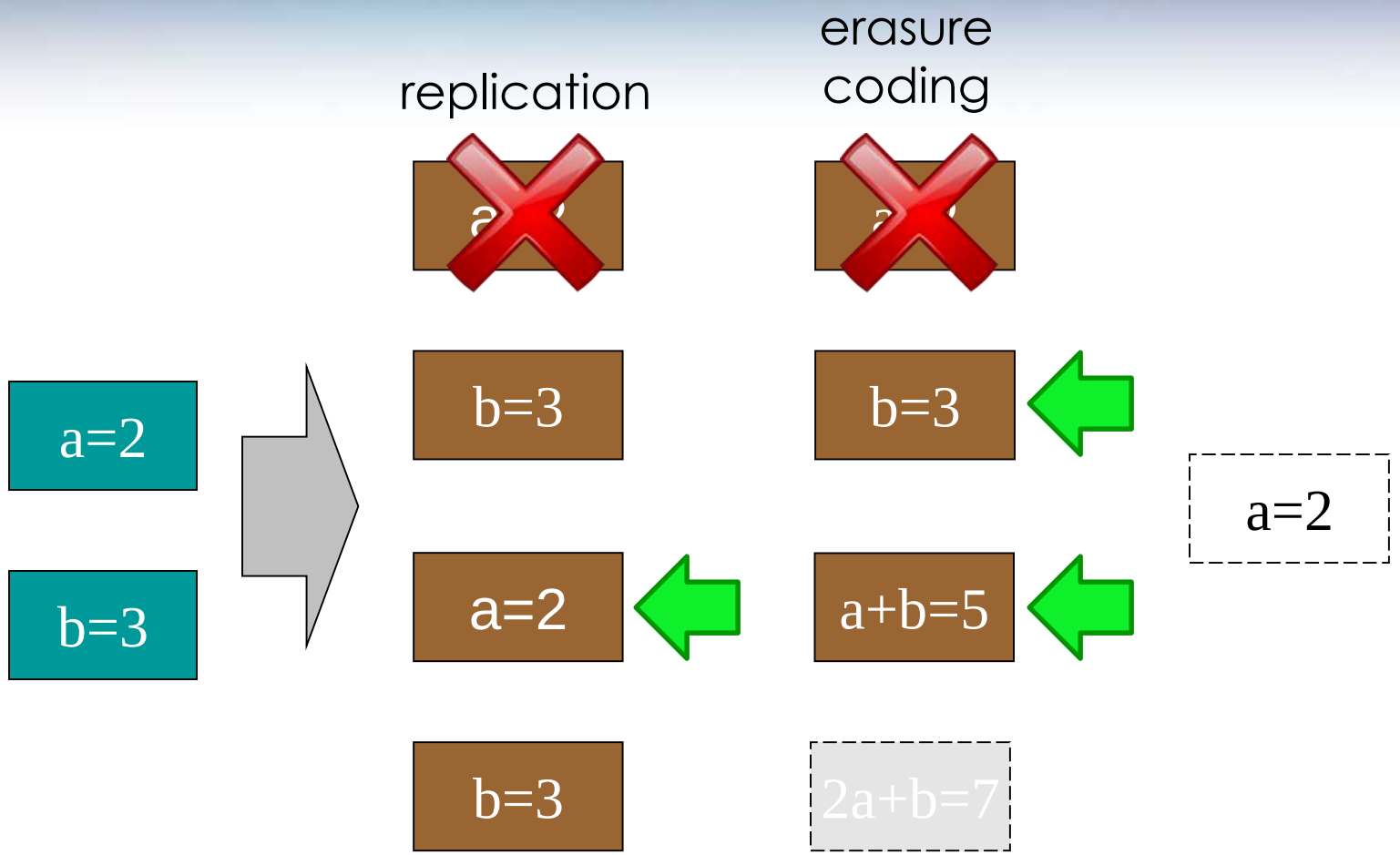
Massive Distributed Storage Systems in the Cloud

- Exabyte of data
  - Millions of disk drive,
  - Tens of thousands of servers
- Failures are norm rather than exception
- As the number of components increase, so does the probability of failure

$$MTTF_{First} = MTTF_{One} / n$$

- Redundancy is necessary to cope with failures
- Triple Replication
- Any better method

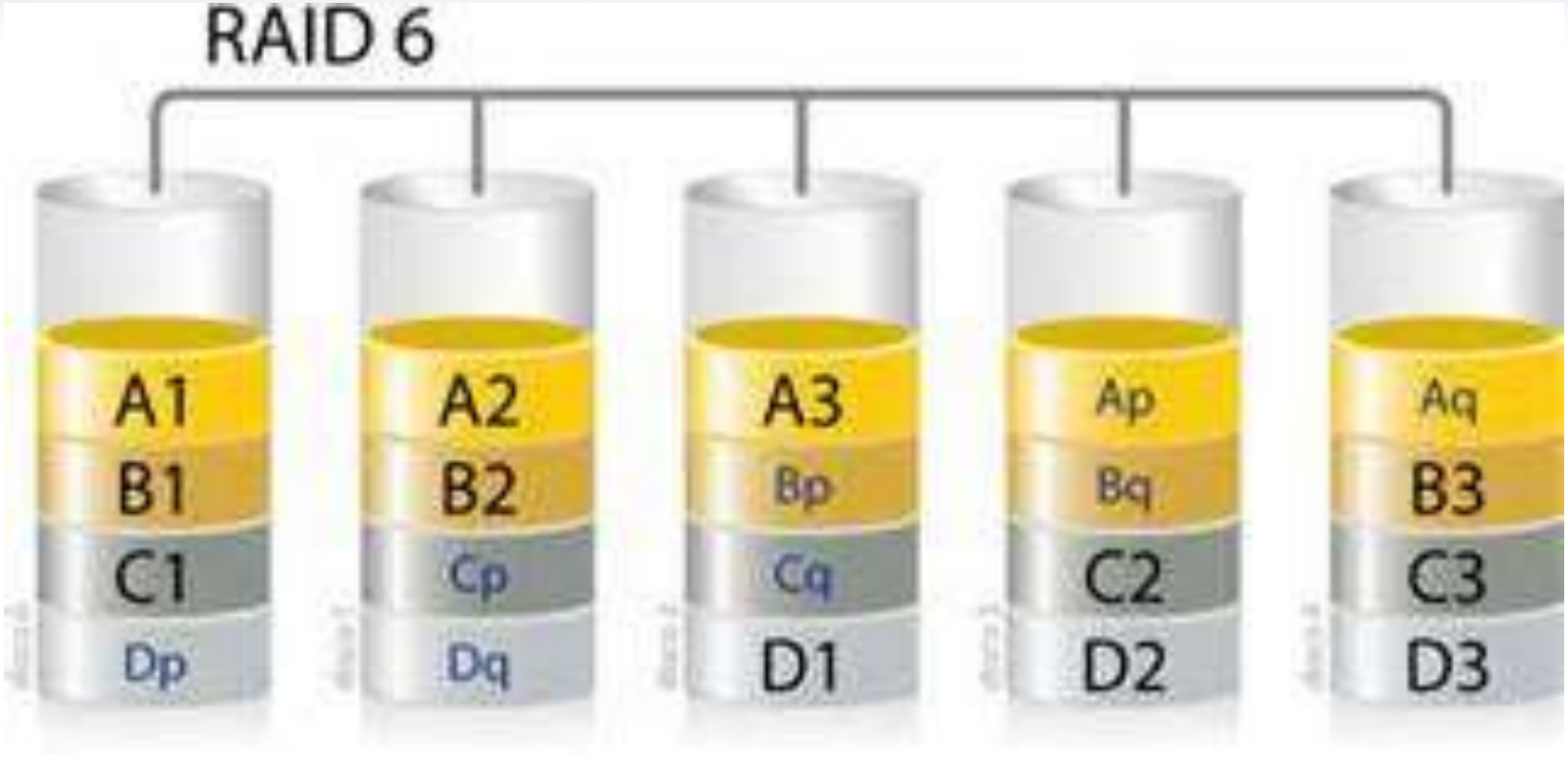
Replication vs. Erasure Coding



First Interaction with Engineers

- Reaction from engineer
  - RAID (or erasure coding) leads to an increase in seek (IOPs) operation, which reduces performance of cloud storage system
  - RAID has issue of data consistency
- Our answer:
  - Lazy erasure coding
  - High performance SSD storage solution (FlashStore/SkimpyStash/BW-Tree)

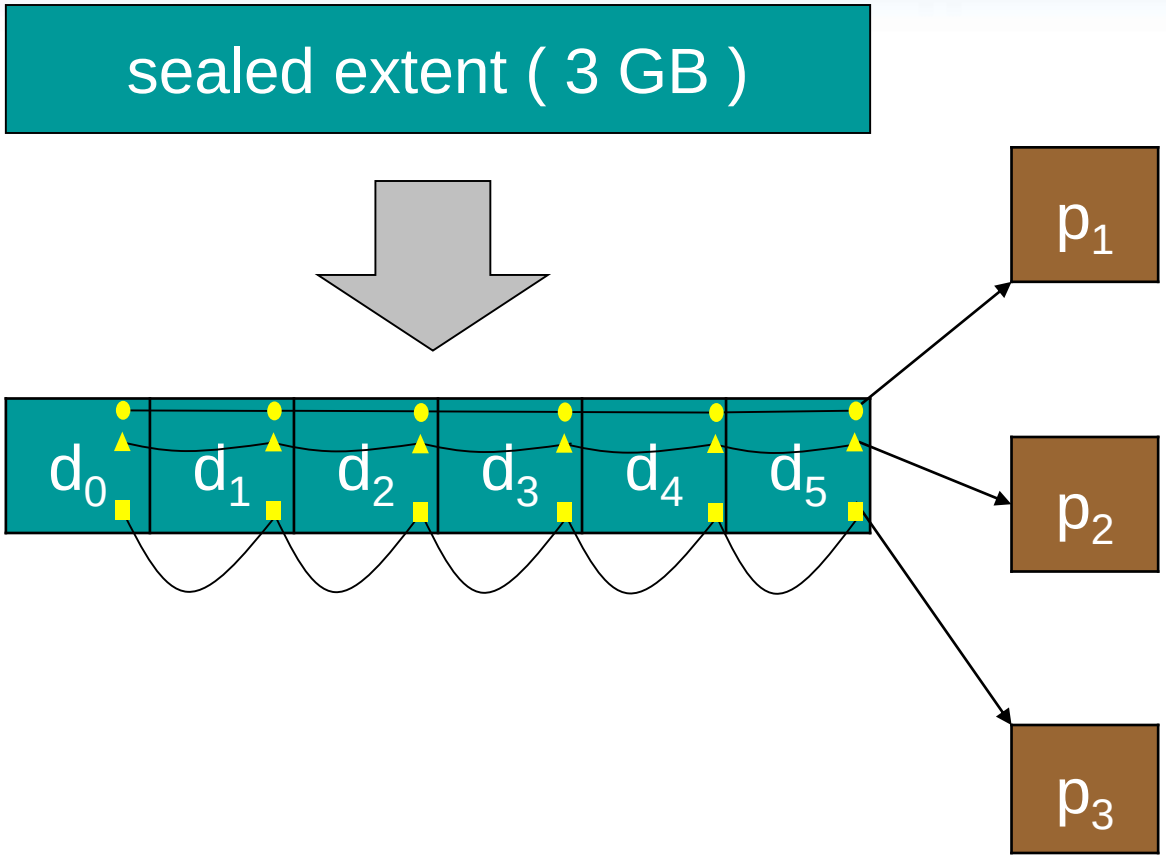
RAID: Online Erasure Coding



Lazy Erasure Coding in Windows Auzre

- Triple replication during extent creation
  - Leverage the fact that Windows Azure is an append-only distributed file system (read/write unit is a blob)
  - No additional seeks (IOPs)
  - Data consistency is easy to handle
- Extents sealed at ~3GB, and then lazily erasure encoded
  - Erasure coding applied at low priority, when the workload is light
  - Since erasure encoding applied on large files (3GB), there are few additional seek operation
  - Easy to handle data consistency, simply re-encode if erasure coding operation crashes
  - When erasure coding finishes, full replicas are deleted
  - Policies to choose between replication, erasure coding, or a mix

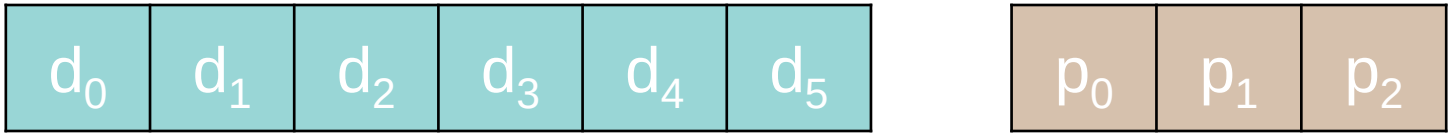
First Erasure Coding – Reed-Solomon 6+3



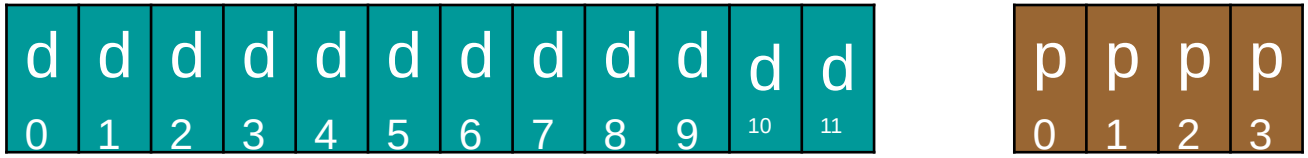
How to Further Reduce Storage Cost?

overhead

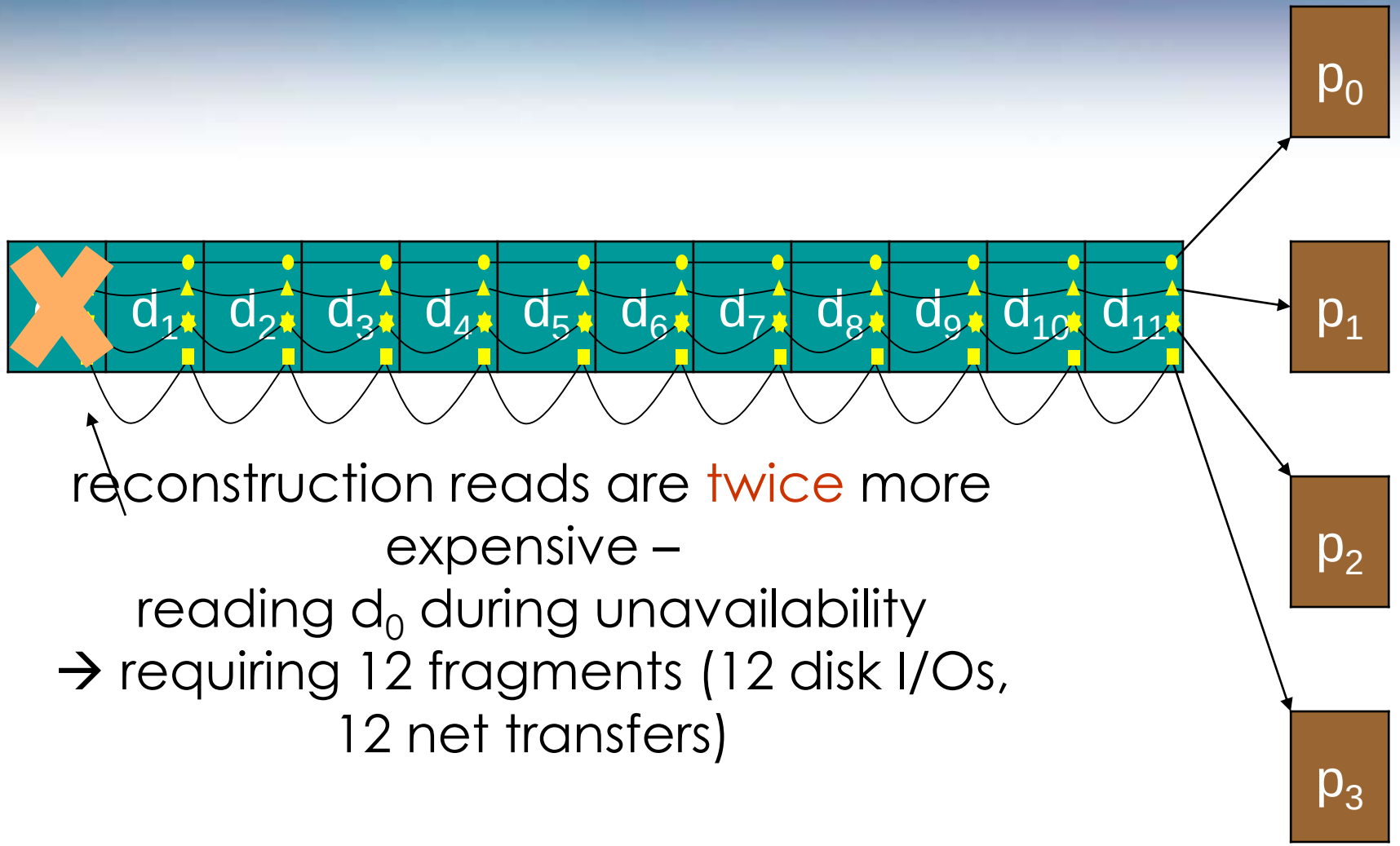
(6+3)/6 = 1.5x



(12+4)/12 = 1.33x



Challenge



reconstruction reads are twice more expensive –
reading d_0 during unavailability
→ requiring 12 fragments (12 disk I/Os, 12 net transfers)

Cloud EC – What to Optimize

- Conventional EC
  - all failures are equal → same reconstruction cost, regardless of failure #
- Cloud EC
  - Prob(1 failure) >> Prob(2 or more failures)
  - Transient unavailability >> permanent failures, with reconstruction read used for
    - Load balancing
      - avoid hot storage node → serve reads via reconstruction
    - Rolling upgrade
    - Node temporarily un-responsive
  - Optimize reconstruction read cost for 1 failure

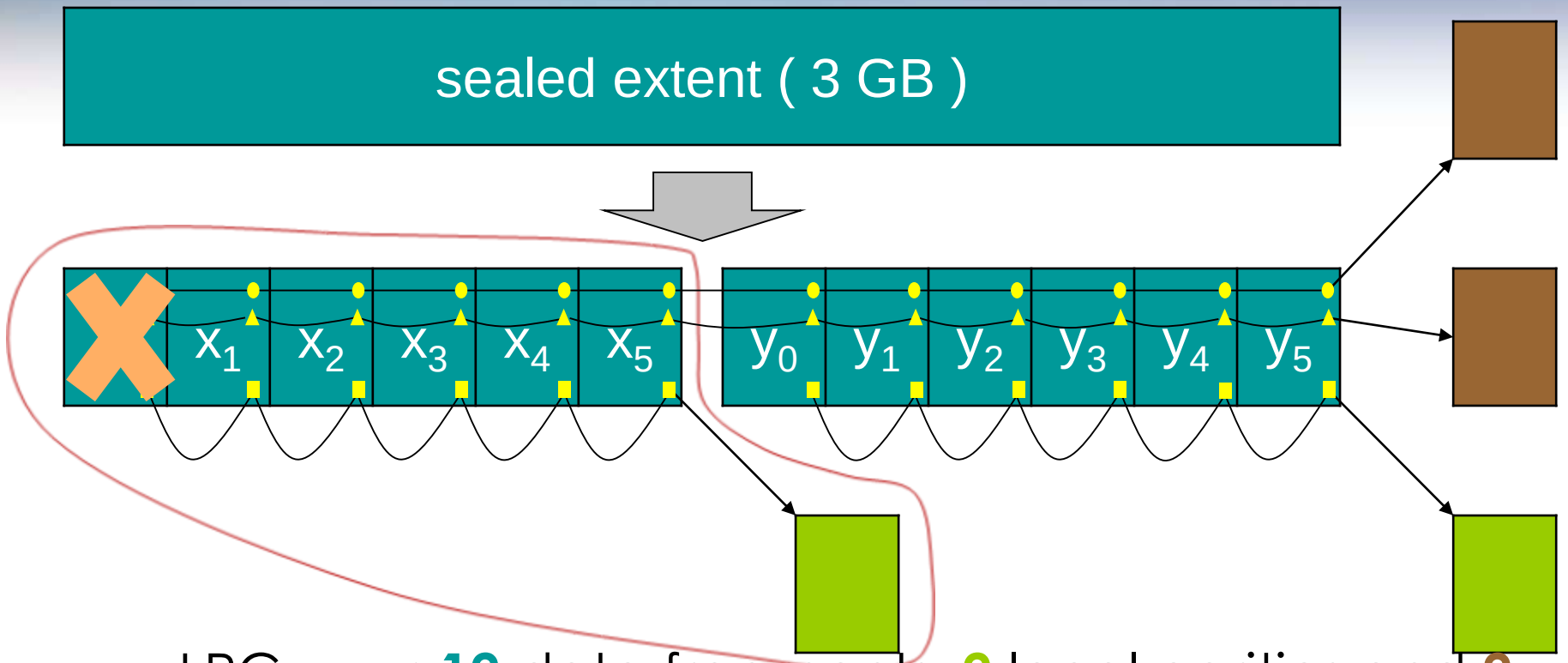
can we achieve1.33x overhead
while requiring only 6 fragments for reconstruction?

Cloud EC – What to Optimize

- Conventional EC
  - all failures are equal → same reconstruction cost, regardless of failure #
- Cloud EC
  - Prob(1 failure) >> Prob(2 or more failures)
  - Transient unavailability >> permanent failures, with reconstruction read used for
    - Load balancing
      - avoid hot storage node → serve reads via reconstruction
    - Rolling upgrade
    - Node temporarily un-responsive
  - Optimize reconstruction read cost for 1 failure

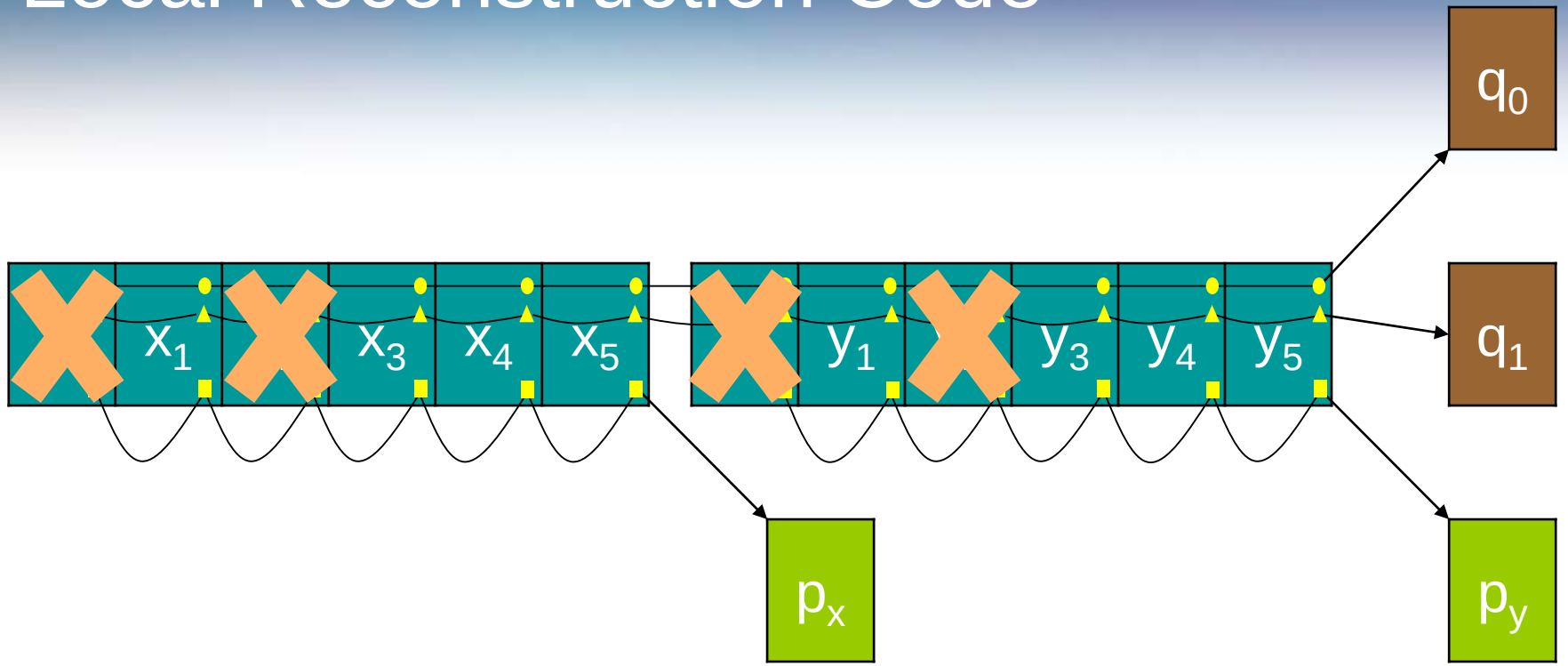
can we achieve1.33x overhead
while requiring only 6 fragments for reconstruction?

Local Reconstruction Code



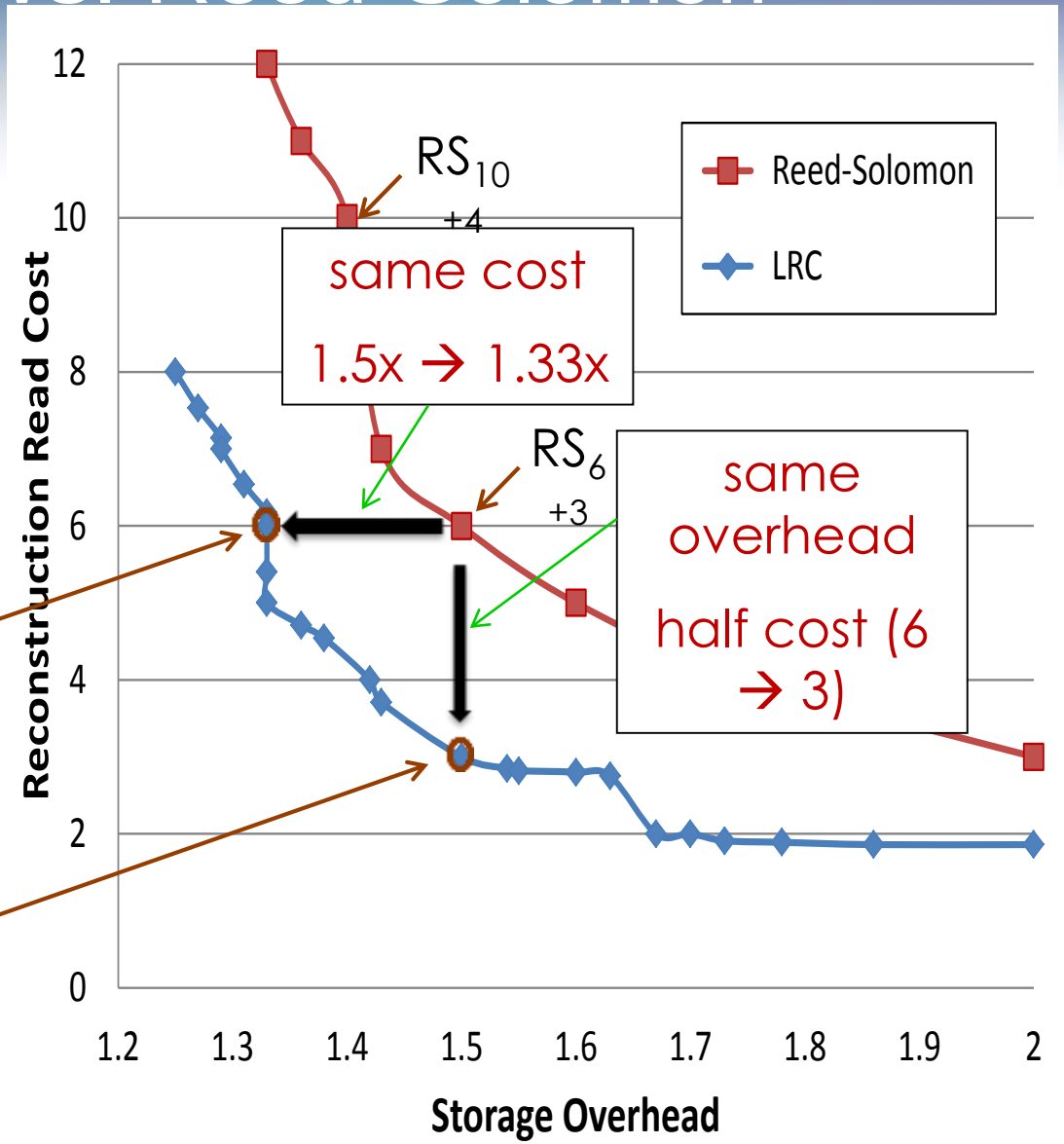
- LRC_{12+2+2}: **12** data fragments, **2** local parities and **2** global parities
  - storage overhead: (12 + 2 + 2) / 12 = 1.33x
- Local parity: reconstruction requires only 6 fragments

Local Reconstruction Code



- Recover arbitrary 3 failures
- Recover as many 4 failures as possible (maximally recoverable property, 86% of them)
- Reliability: $RS_{12+4} > LRC_{12+2+2} > RS_{6+3}$

LRC vs. Reed-Solomon

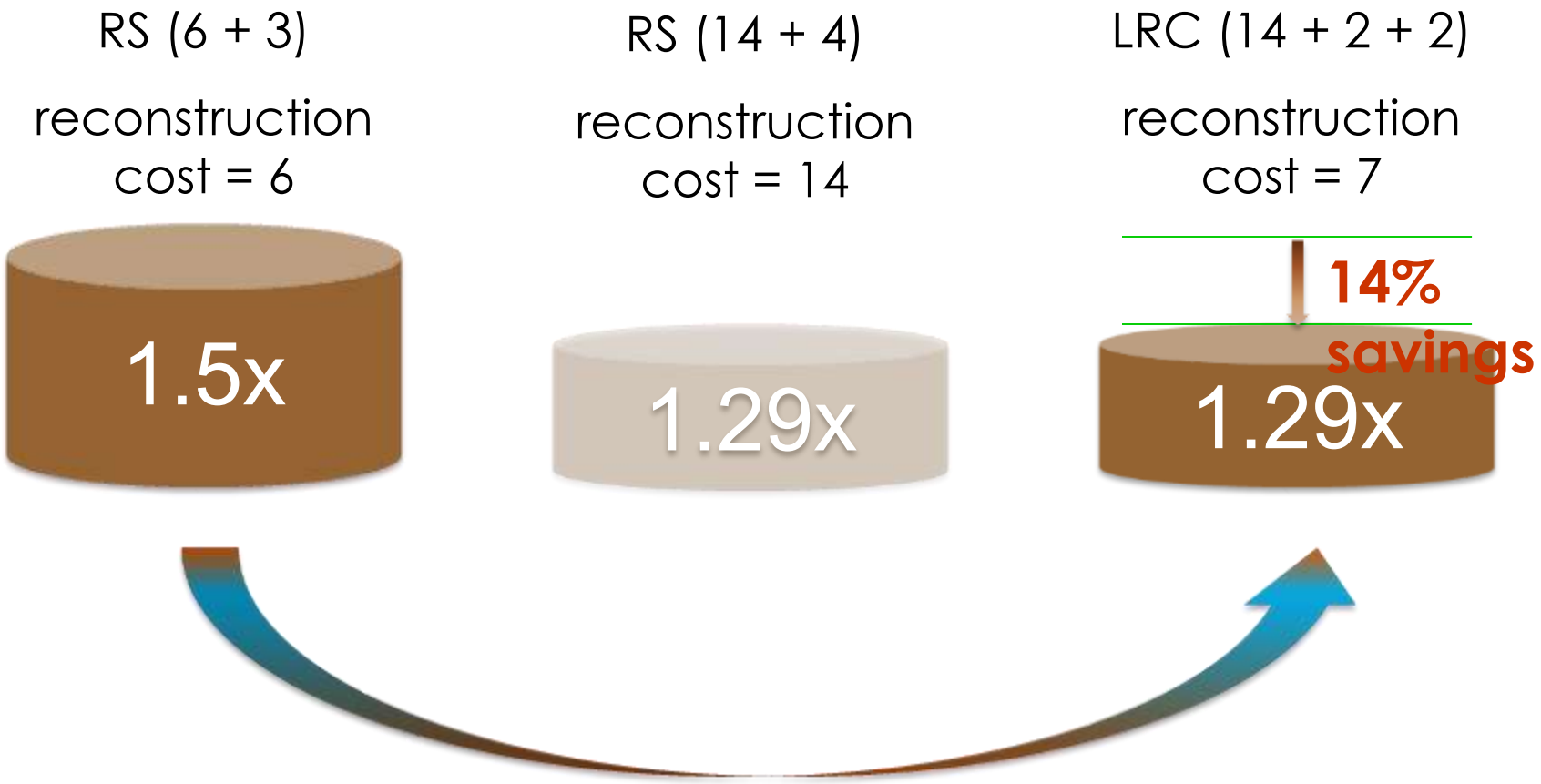


LRC (12+2+2)

LRC (12+4+2)

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

Choice of Windows Azure Storage



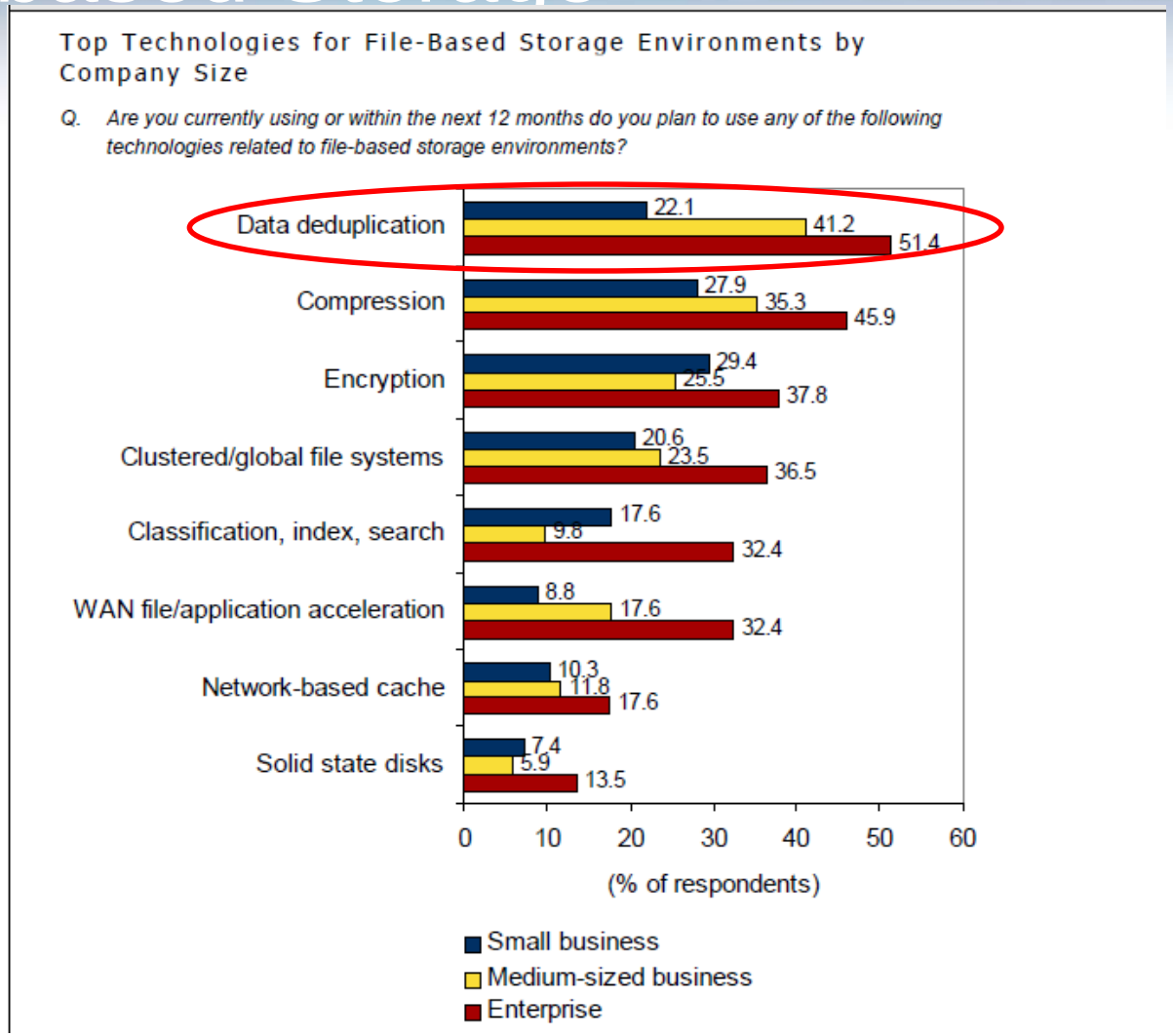
Community Feedback

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
  - http://blogs.msdn.com/b/windowsazurestorage/

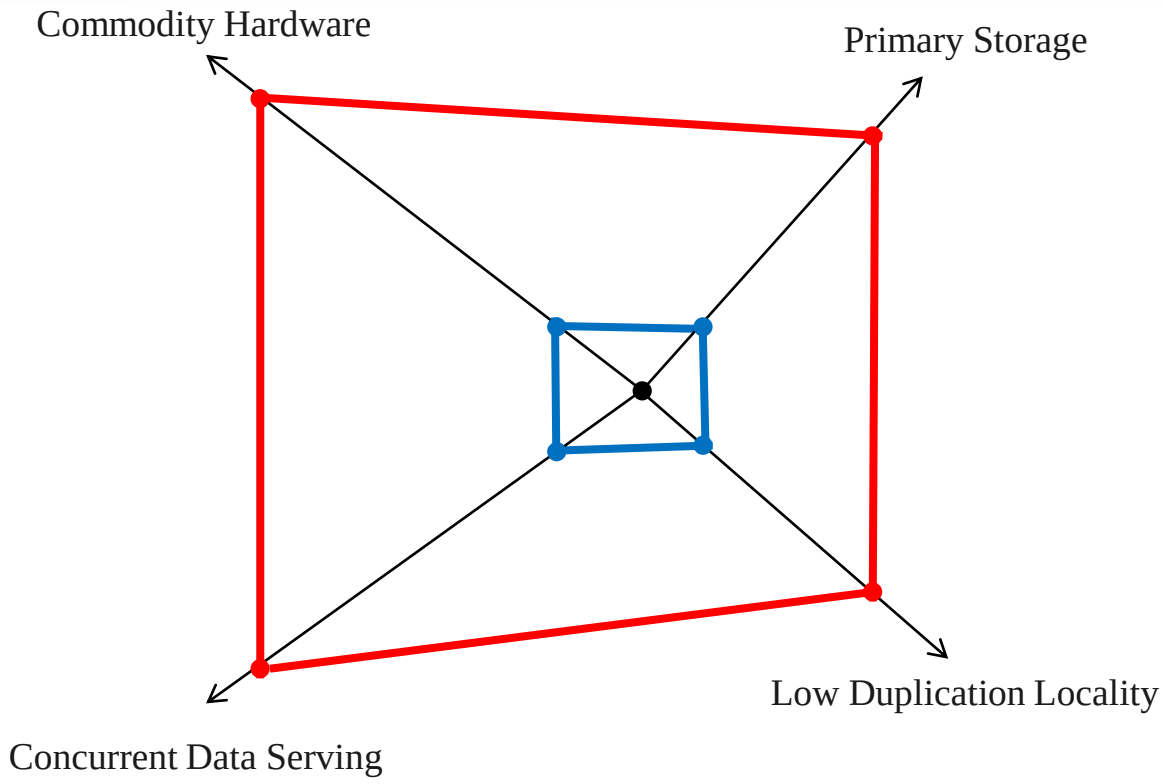
Primary Data Deduplication for Windows Server 2012

Primary Data Deduplication for File-based Storage



Data deduplication: #1 Technology feature when choosing a storage solution

Deduplication Problem Space



- Backup Data dedup
- Primary Data dedup

Key Challenge

Because Windows Server needs to run well on commodity hardware and

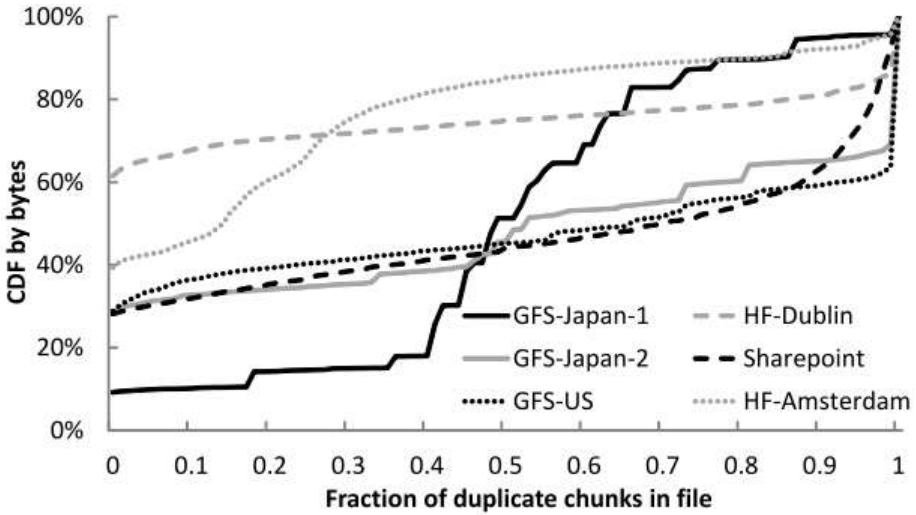
because deduplication has to be friendly to the primary data serving workload,

this introduces unique challenges in terms of balancing

server resource usage,
deduplication throughput, and
deduplication space savings.

Data Driven Design Philosophy

Dataset	Dedup Space Savings		
	File Level	Chunk Level	Gap
HF-Amsterdam	1.9%	15.2%	8x
HF-Dublin	6.7%	16.8%	2.5x
HF-Japan	4.0%	19.6%	4.9x
GFS-Japan-1	2.6%	41.1%	15.8x
GFS-Japan-2	13.7%	39.1%	2.9x
GFS-US	15.9%	36.7%	2.3x
Sharepoint	3.1%	43.8%	14.1x
VL	0.0%	92.0%	∞



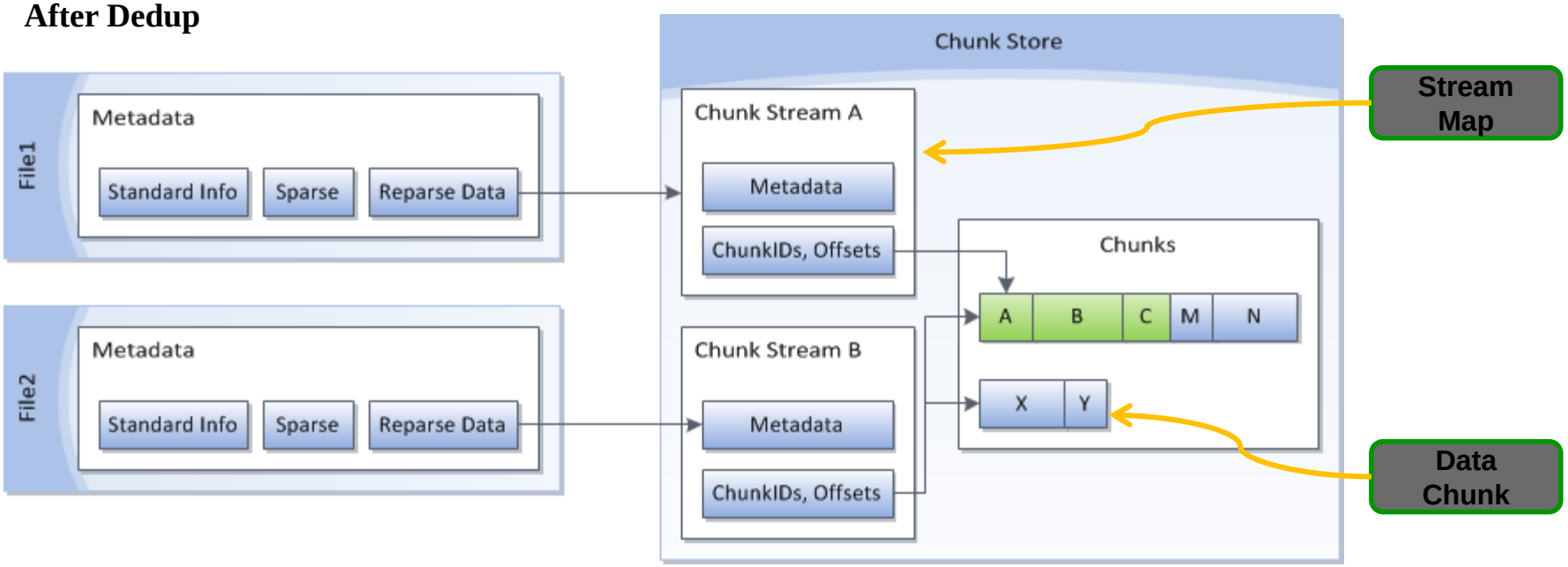
sub-file dedup vs. whole-file dedup

- Numbers obtained using average chunk size of ~64KB

Key Design Decisions – Post Processing Dedup

Before Dedup

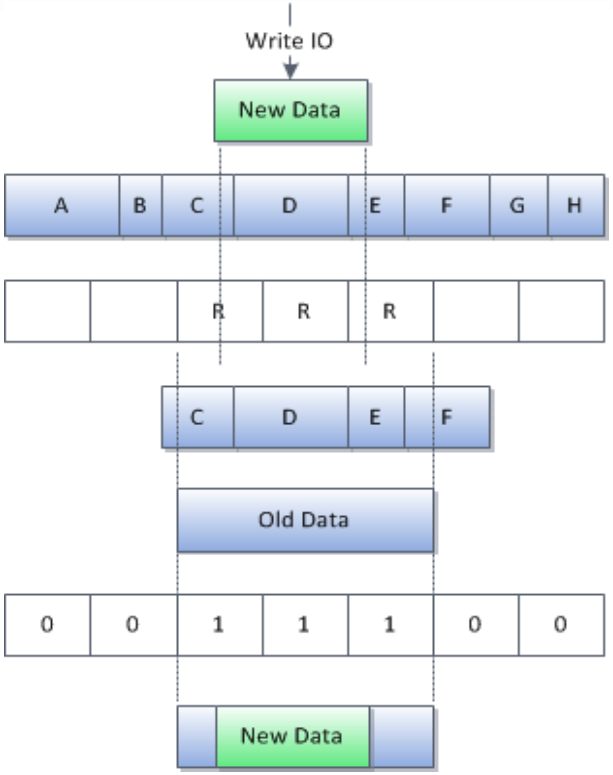
After Dedup



Write path to Optimized File

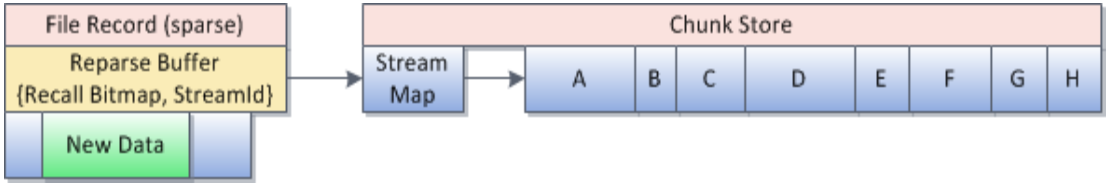
① Pre write File Layout

② Write flow

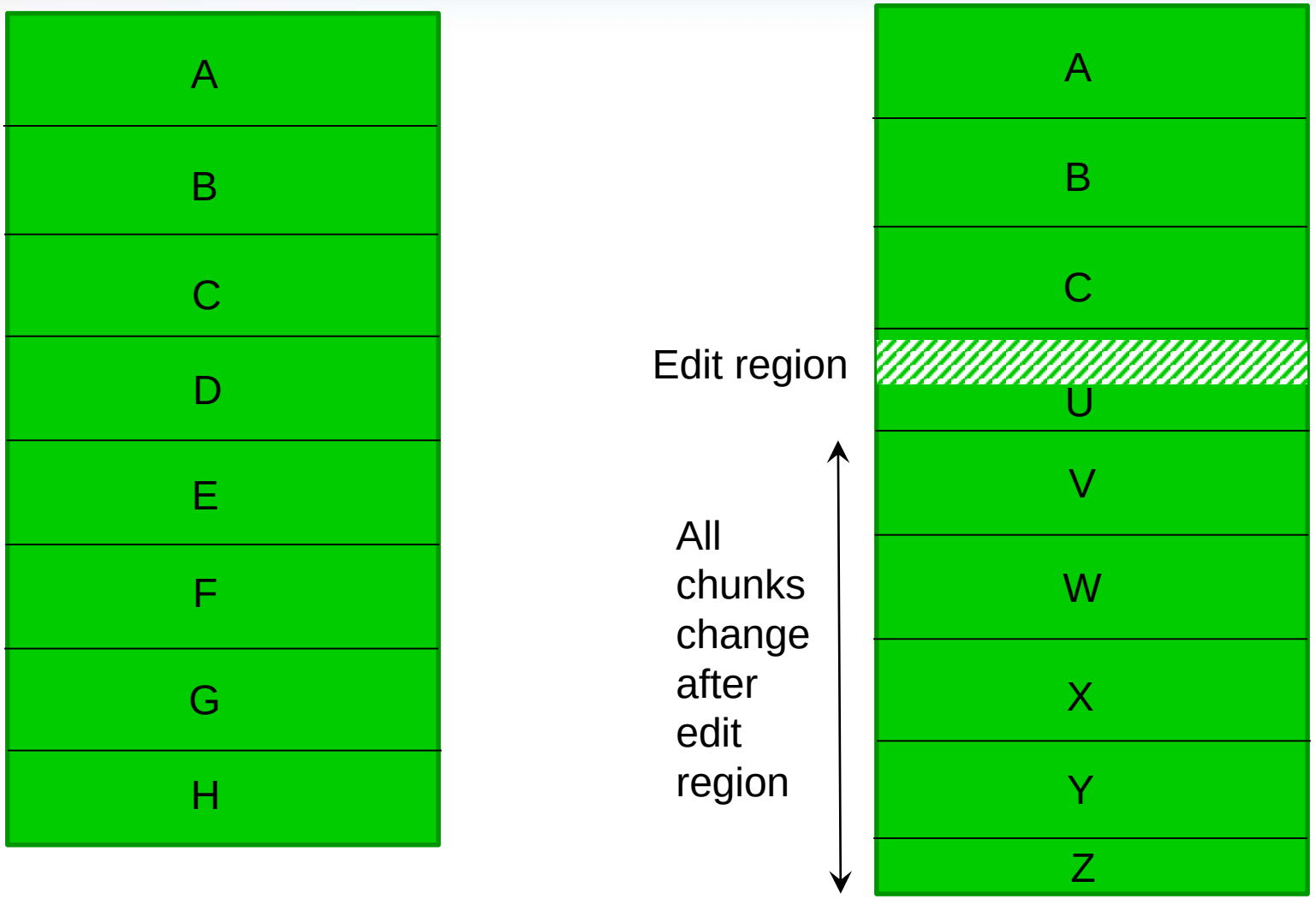


- Reduce latency for small writes to large files (e.g. OS image patching scenario)
- Recall granularity grows with file size
- Incremental Dedup will later optimize the new data
- GC cleans up unreferenced chunks (chunk “D” in example)

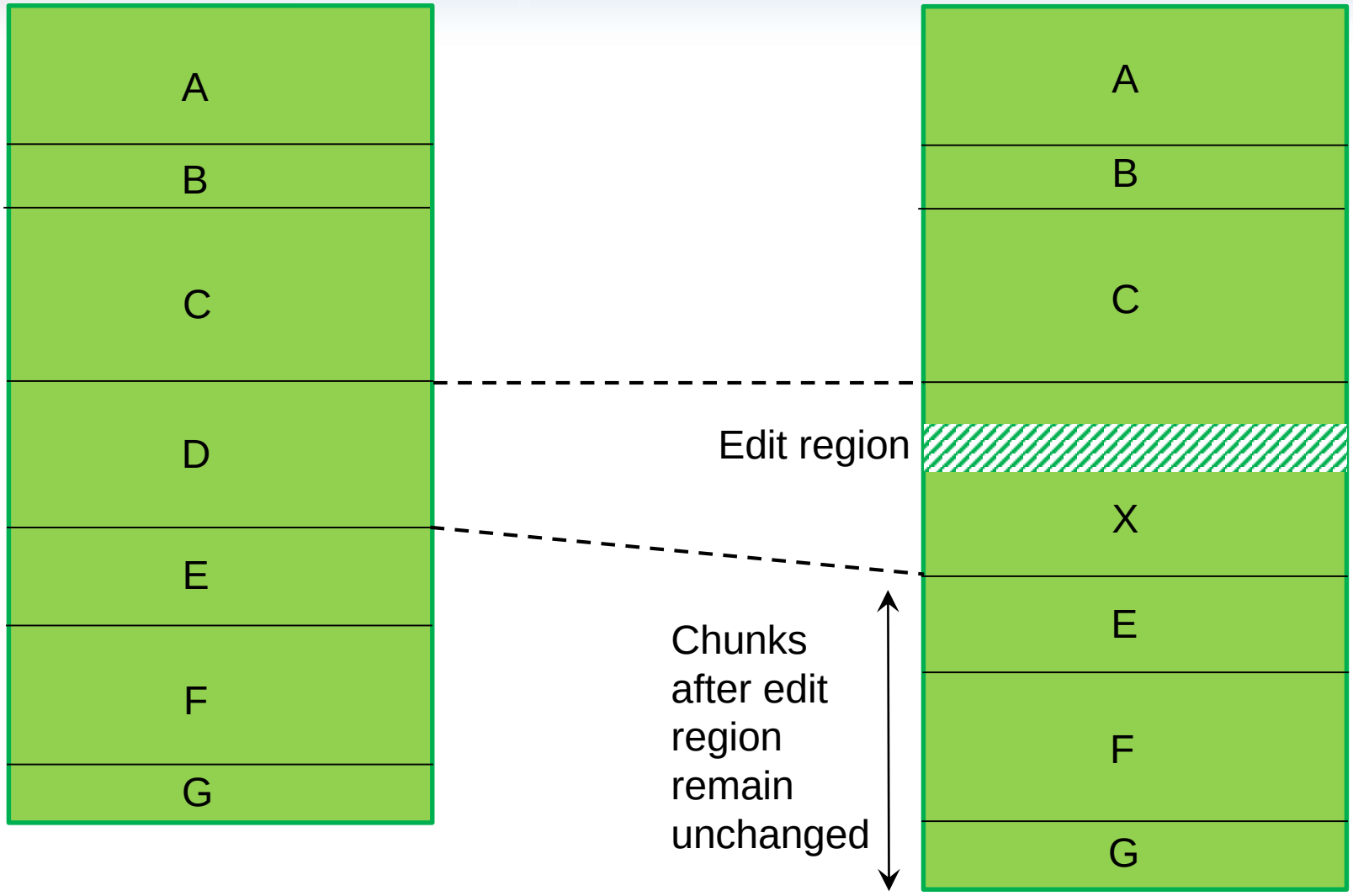
③ Post Write File Layout



Chunking Strategies: Fixed Size

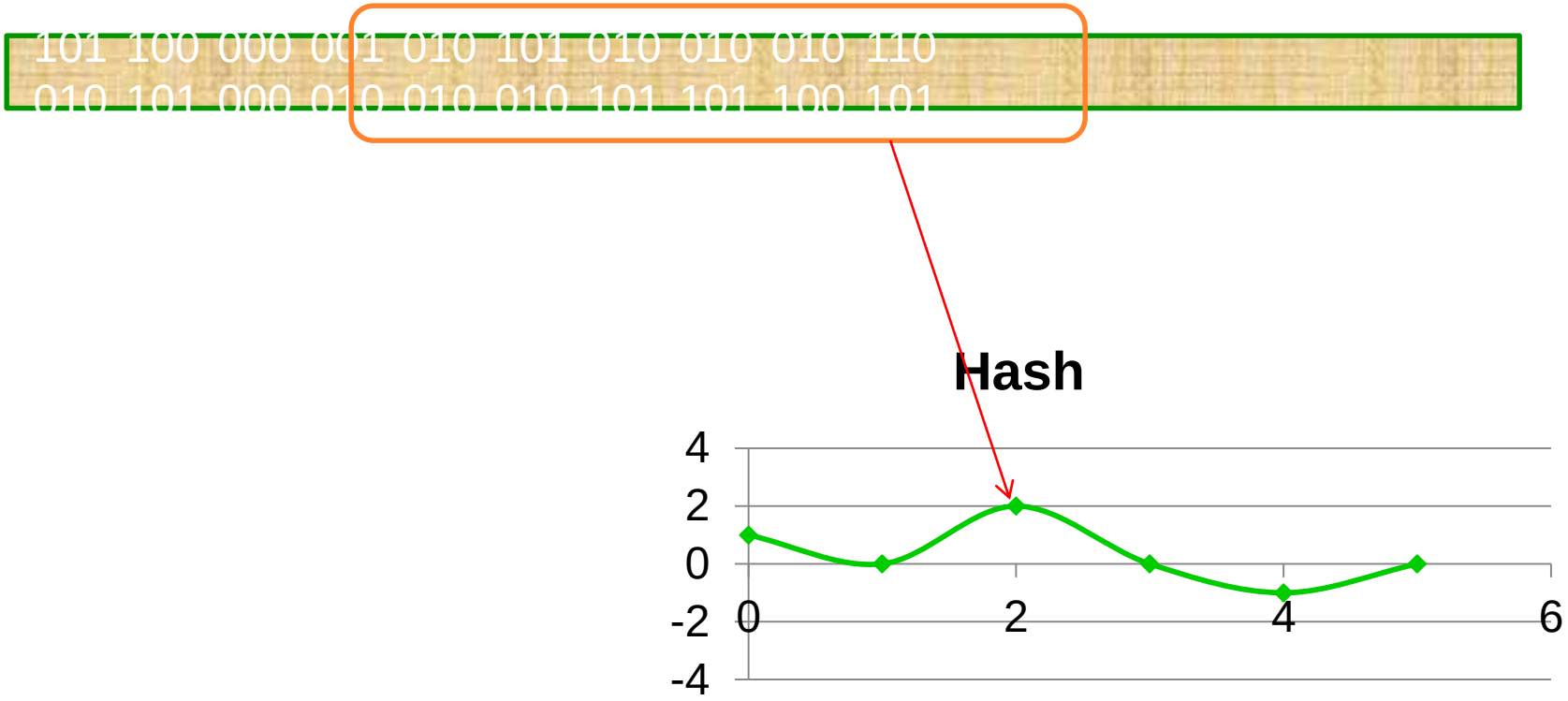


Chunking Strategies: Content Dependent



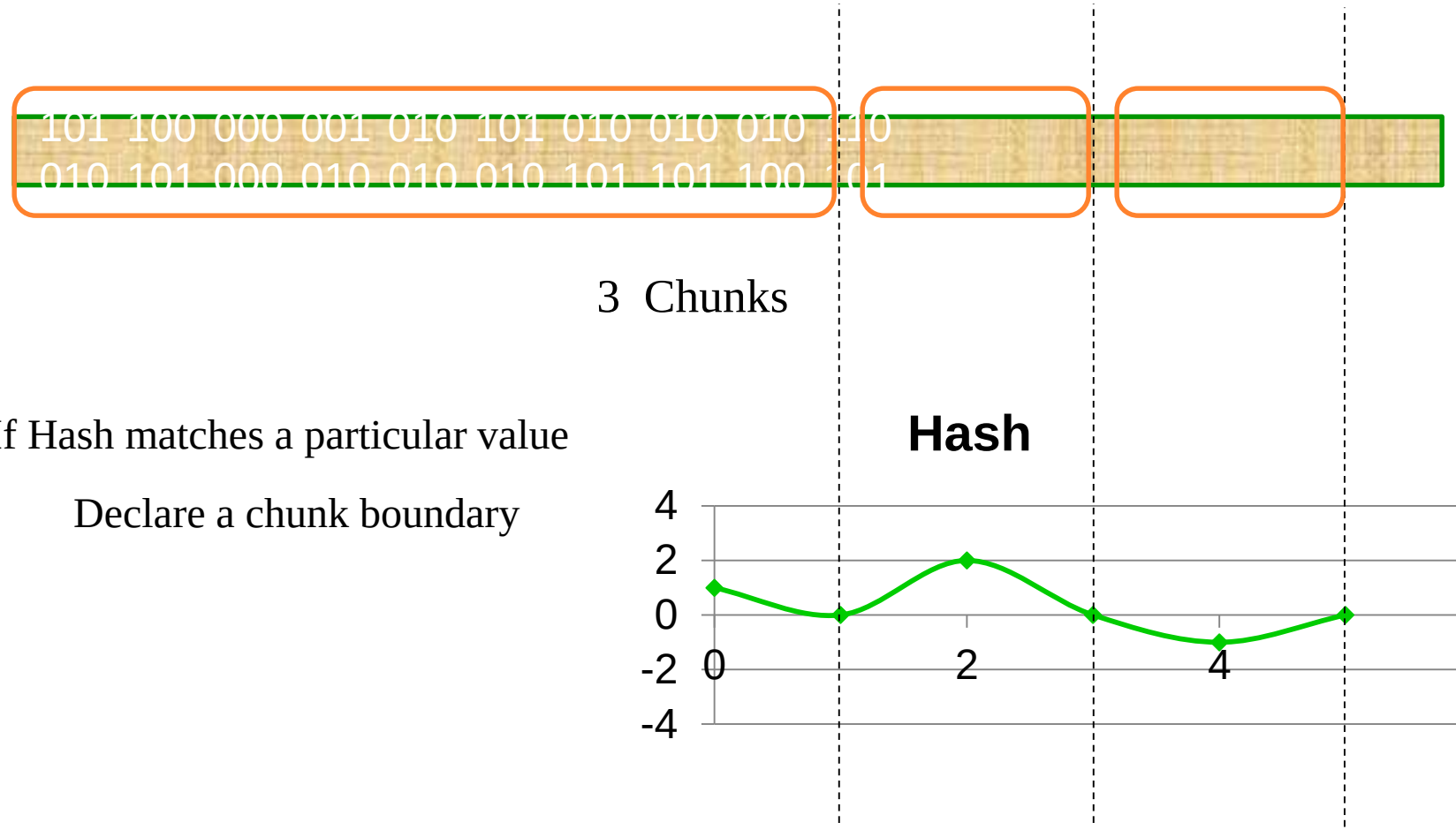
Content based Chunking

- Calculate fingerprint hash for each sliding window (16 byte)



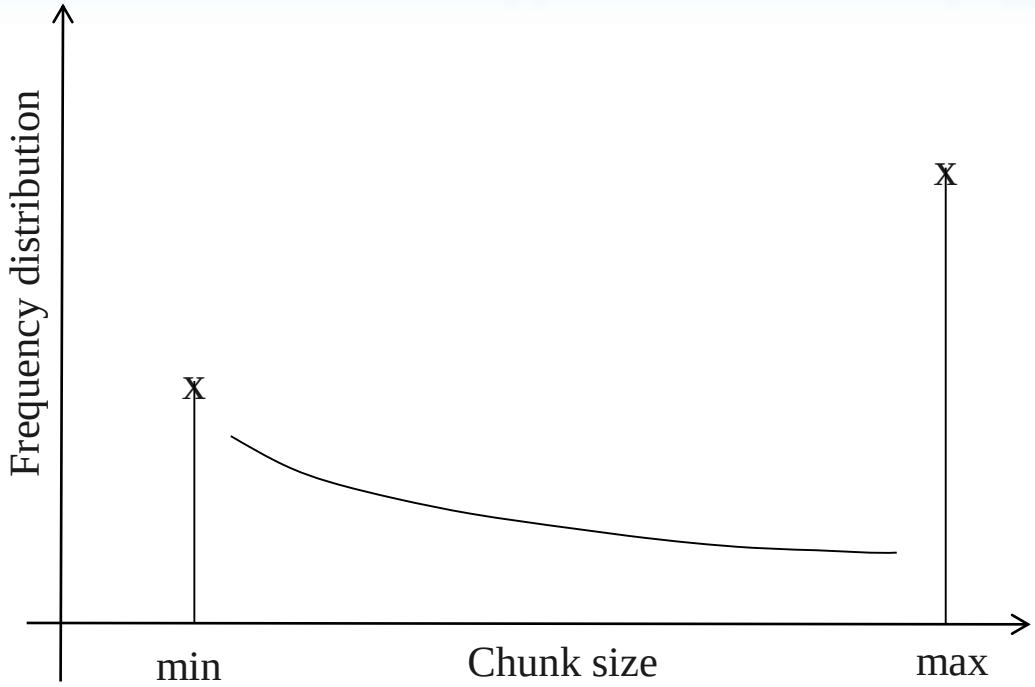
Content based Chunking

- Calculate fingerprint hash for each sliding window (16 byte)



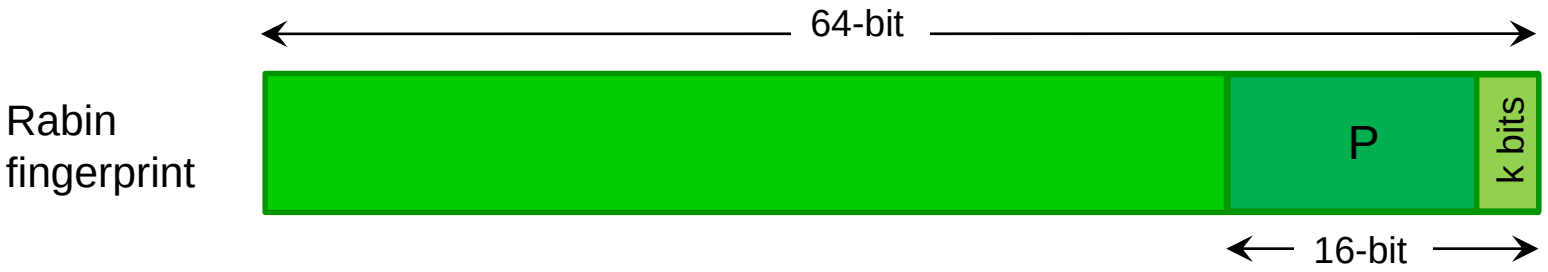
Basic version of fingerprint based chunking

- Skewed chunk size distribution
  - Small chunk size => increase in chunk metadata in the system
  - Large chunks => reduced dedup savings, benefit of caching
- Forced chunk boundaries
  - Forced boundary at max chunk size is content independent, hence may reduce



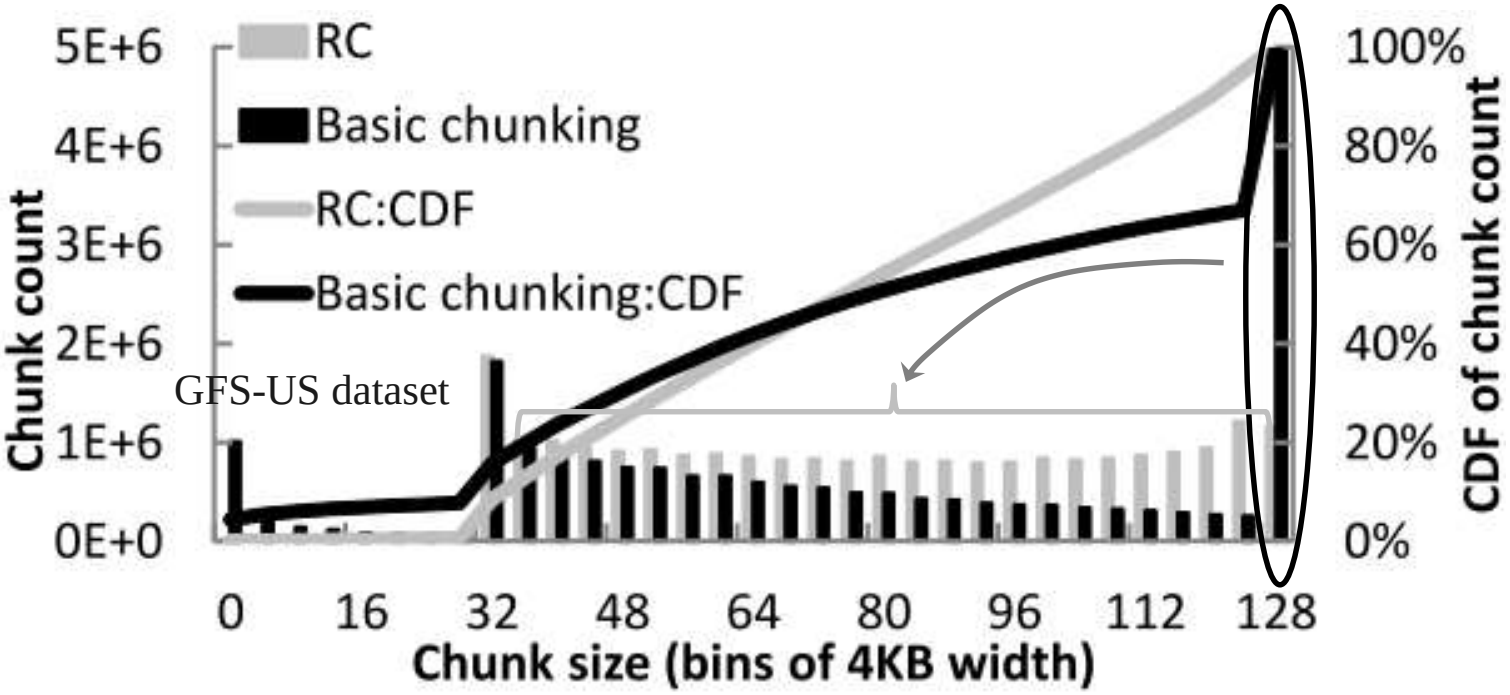
Regression Chunking Algorithm

- Basic idea
  - When max chunk size is reached, relax match condition to some prefix (suffix) of bit pattern P
  - Match |P| - i bits of P, with decreasing priority for i=0,1, …, k



- Reduces probability of forced boundary at max size
  - $2x10^{-3}$ for k=1, 10^{-14} for k=4

Regression Chunking: Uniform Chunk Size Distribution



GFS-US dataset

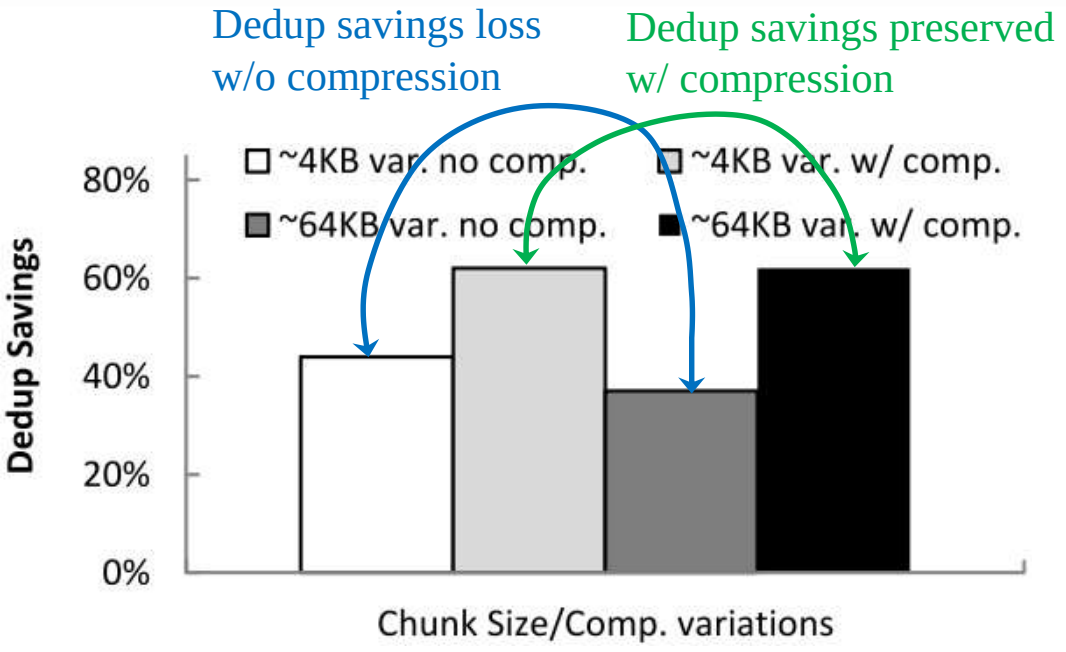
Regression Chunking: Dedup Savings Improvement

Dataset	Dedup Space Savings		
	Basic Chunking	Regression Chunking (RC)	RC Benefit
Audio-Video	2.98%	2.98%	0%
PDF	9.96%	12.70%	27.5%
Office-2007	35.82%	36.65%	2.3%
VHD	48.64%	51.39%	5.65%
GFS-US	36.14%	37.2%	2.9%

GFS-US dataset

Average Chunk Size

- Compression compensates for savings decrease with higher chunk size
  - Compression is more efficient on larger chunks
- Use larger chunk size of ~80KB (32-128KB)
  - Without sacrificing dedup savings
  - Reduce chunk metadata in the system



GFS-US dataset

The Chunk Indexing Problem

- Chunk metadata too big to fit in RAM
  - 48 bytes per chunk: 32-byte SHA hash + 16-byte location
  - 20TB of unique data, average 64KB chunk size => 15GB of RAM
  - Not cost effective to keep all of this huge index in RAM
- Index accesses will go to secondary storage
  - But, no locality in key space for chunk hash lookups
  - Will impact deduplication throughput

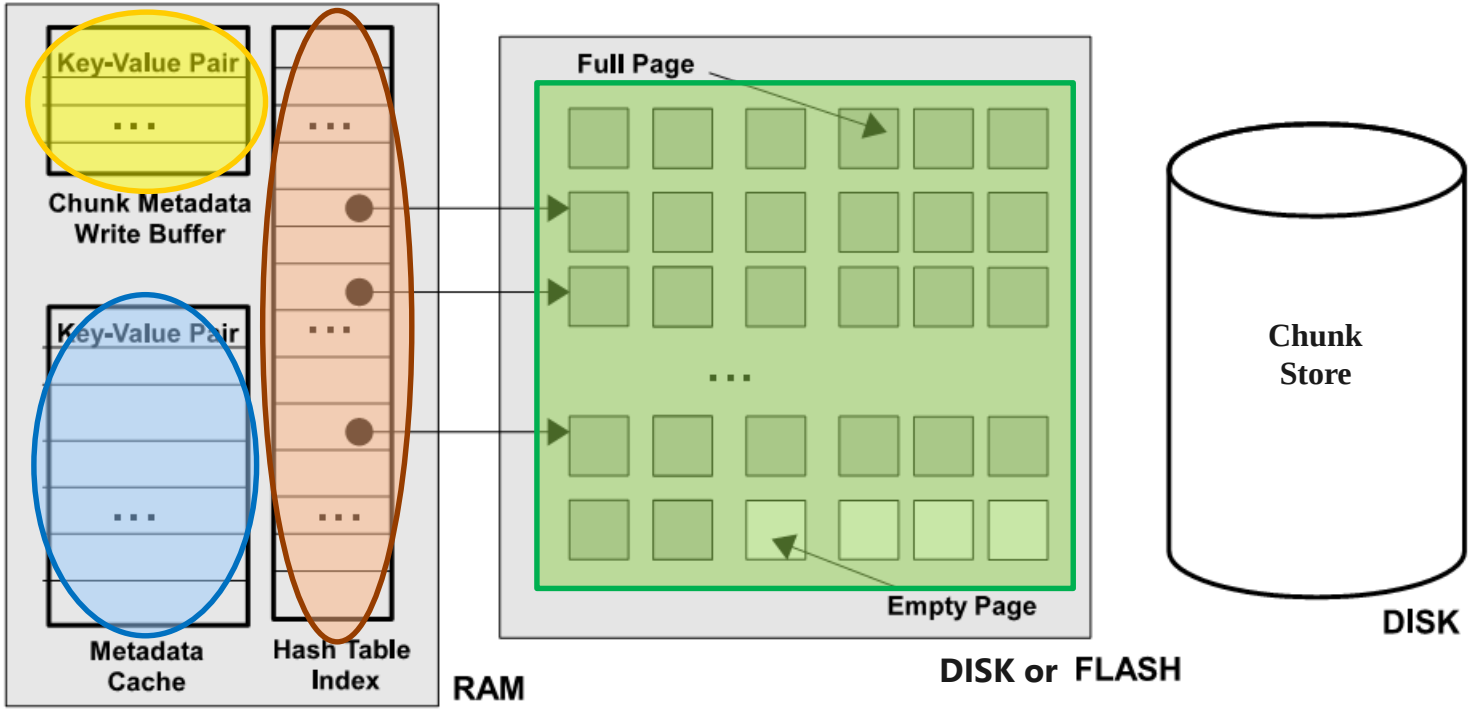
Deduplication index resource usage scaling with data size

- RAM frugal chunk hash index
- Explore deduplication locality
- Data partitioning and reconciliation

Chunk Index Architecture

RAM write buffer for chunk mappings in currently open session

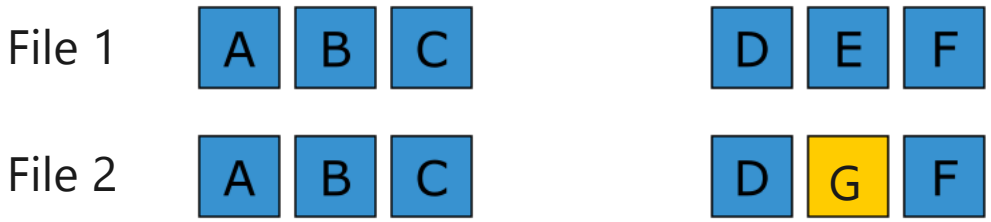
Chunk metadata organized in log-structured manner



Prefetch cache for chunk metadata in RAM for sequential predictability of chunk lookups

Chunk metadata indexed in RAM using a specialized space efficient hash table

Exploit Deduplication Locality



- Locality in sequence of index lookups
  - Can be exploited by prefetch cache in RAM
  - Prefetch chunk mappings for next several chunks in *same I/O*

Performance Evaluation – Resource Usage

- RAM frugality
  - Index memory usage reduction of 24x vs. baseline
- Low CPU utilization
  - 30-40% per core
  - Enough room available for serving primary workload in multi-core modern file servers
- Low disk usage
  - Median disk queue depth is zero in all cases
  - At 75-th percentile, increase by 2-3; impact of index lookups going to disk and/or reconciliation

	Regular Index (Baseline)	Optimized index	Regular index w/ partitions	Optimized index w/ partitions
Partitioning factor	1	1	3	3
Index entry size (bytes)	48	6	48	6
Index memory usage	931MB	116MB	310MB	39MB
Single core utilization	31.2%	35.2%	36.8%	40.8%

Other Technologies

- Read performance optimization
- Crash consistency
- Detect, contain, repair, report chunkdata and metadata corruptions

Dedup among top Windows Server 2012 features being talked about

Why Windows 8 server is a game-changer

… game-changers like deduplication …

Top 10: New Features in Windows Server 8

Windows Server 8: built for the cloud, built for virtualization

Windows Server 8 includes storage deduplication. Microsoft demonstrations of the technology reduced the disk footprint of a VDI server by some 96 percent. Microsoft's approach is also different from that of a virtualization system like VMware, insofar as it's not restricted to virtualization workloads. File servers too can benefit from the technology. The result should be both more robust and more effective …

Customer quotes

"We achieved in our setup a deduplication reduction of 43%, without compression. So the 641 GB became 363 GB ! **More than twice better dedupe ratio than Netapp !!"**

(a customer in Netherlands)

The results can only be called Jaw Dropping! These two deduplication features complement one another incredibly well and return results that have to be seen to be believed. The process takes an initial file size of 250 GB , and when both deduplication methods are applied, the ending size on disk is an astounding 7GB! **The first time I saw it I thought there must be some mistake! Not so!** It is the most compelling disk deduplication demonstration I have ever seen!

[…] **the savings rate is 90%.** I have included data for all servers but it only shows GB saved. IMHO, dedup completely rocks. The technology is responsible for around 62TB of saved disk space in atlas today. There was some degree of trepidation going into Blue and using a v1 technology, **but that has been proved to be unfounded as we have had 0 issues to date**.

Summary

- Primary data deduplication in Windows Server 2012
  - Extending problem frontier in multiple dimensions w.r.t. backup dedup
- Design decisions driven by large scale data analysis
  - 7TB of data across 15 globally distributed servers in Microsoft
- Primary workload friendly
  - Scale deduplication processing resource usage with data size
  - Data chunking and compression
  - Chunk indexing
  - On-disk structures optimized for read/modify of deduplicated data
- Robust and reliable on commodity hardware
  - Detect, contain, repair, report user and metadata corruptions