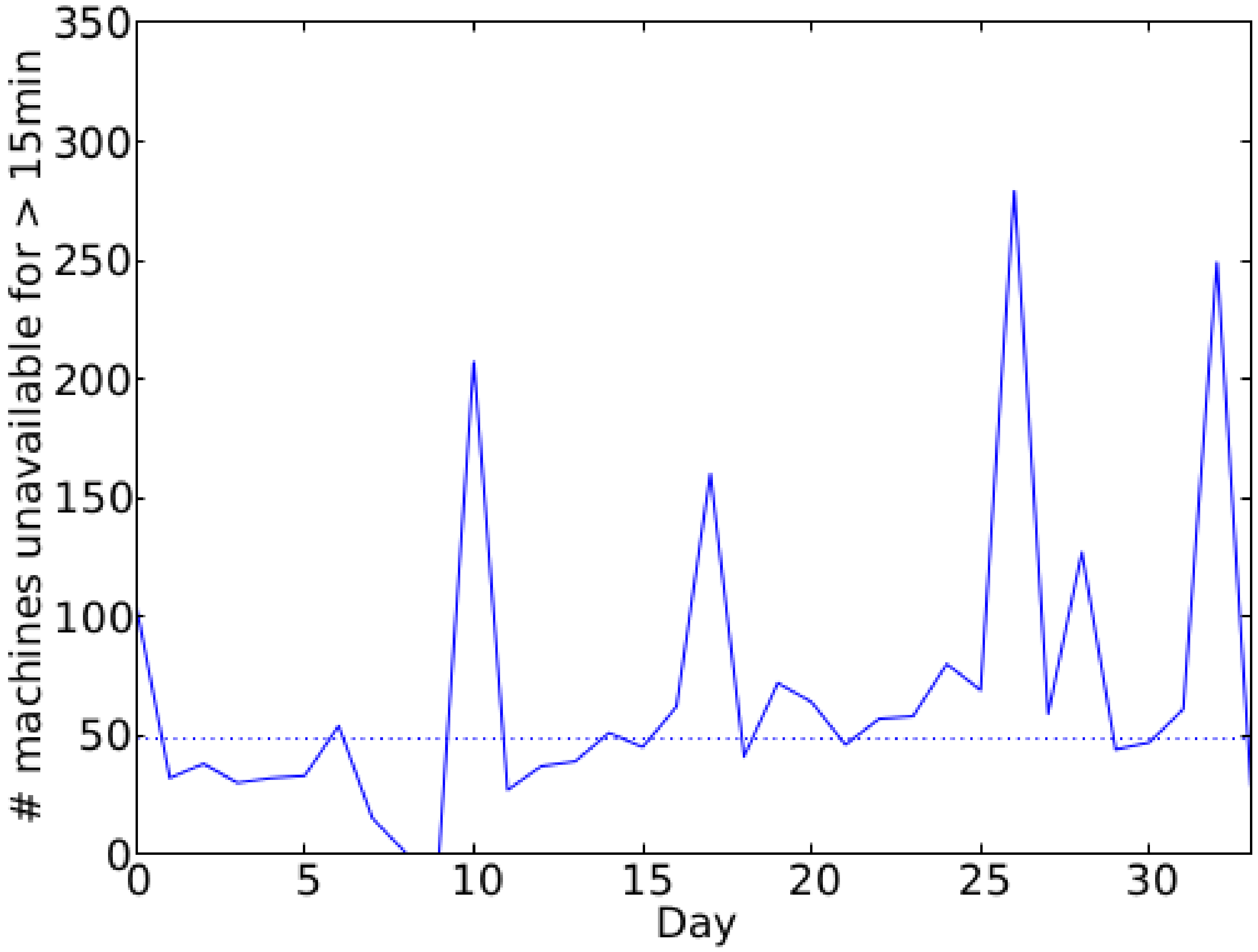
Practical erasure codes tradeoffs for scalable distributed storage systems

Cyril Guyot

Director, Software Solutions and Algorithms

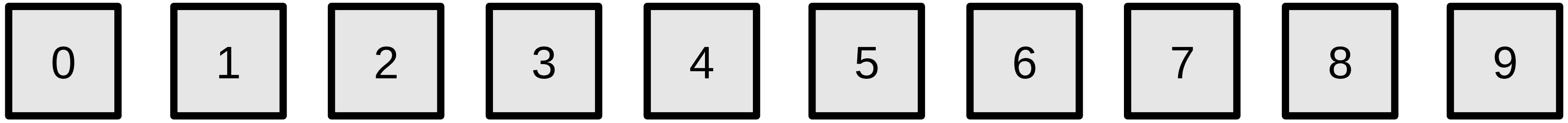
May 22nd, 2019

Data center failures + unavailability

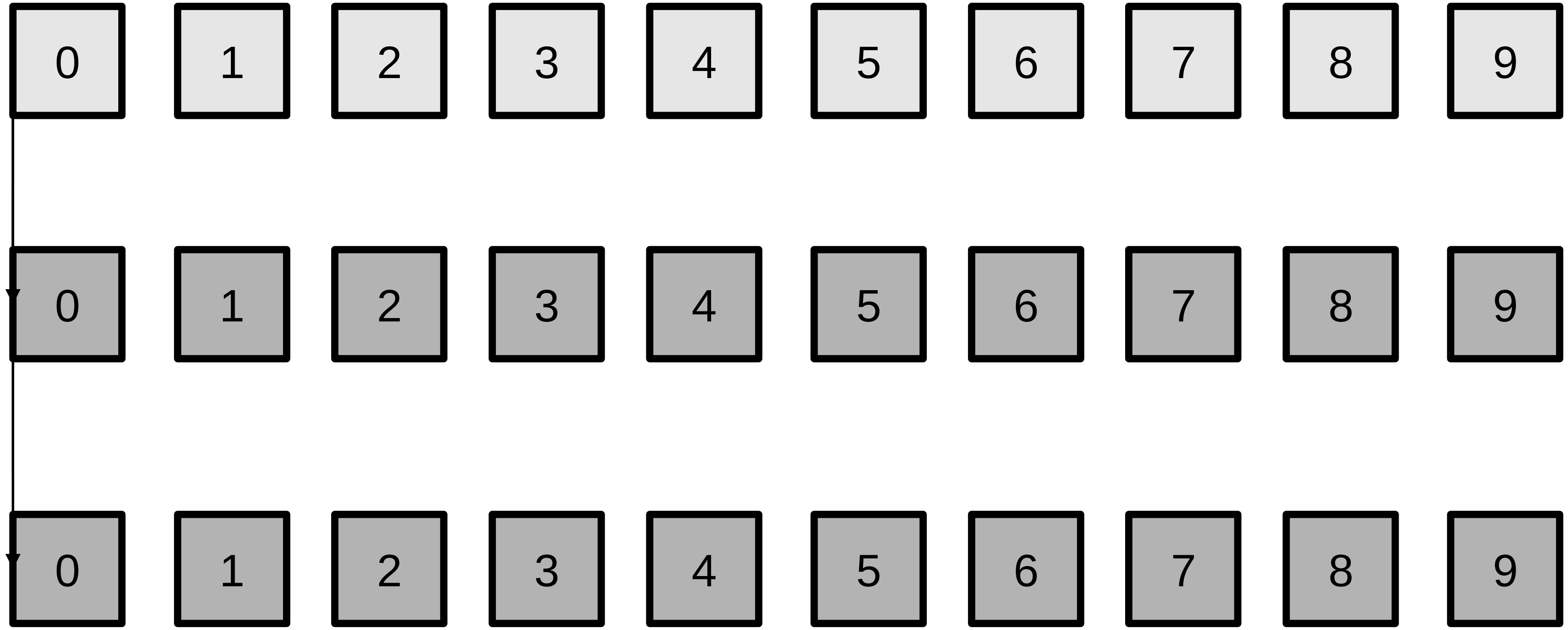


Source: Rashmi et. al., A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster

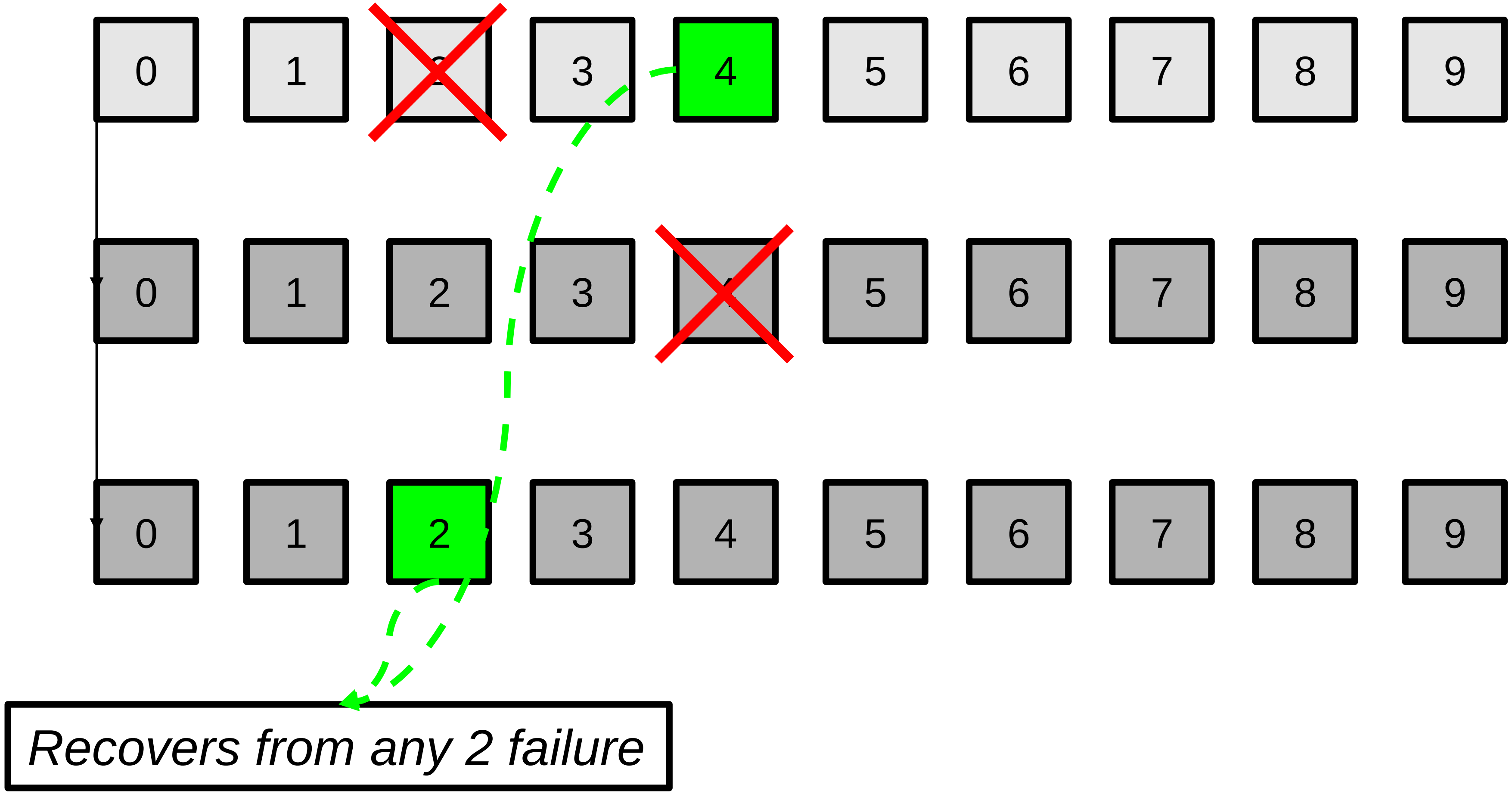
Simple solution: replication



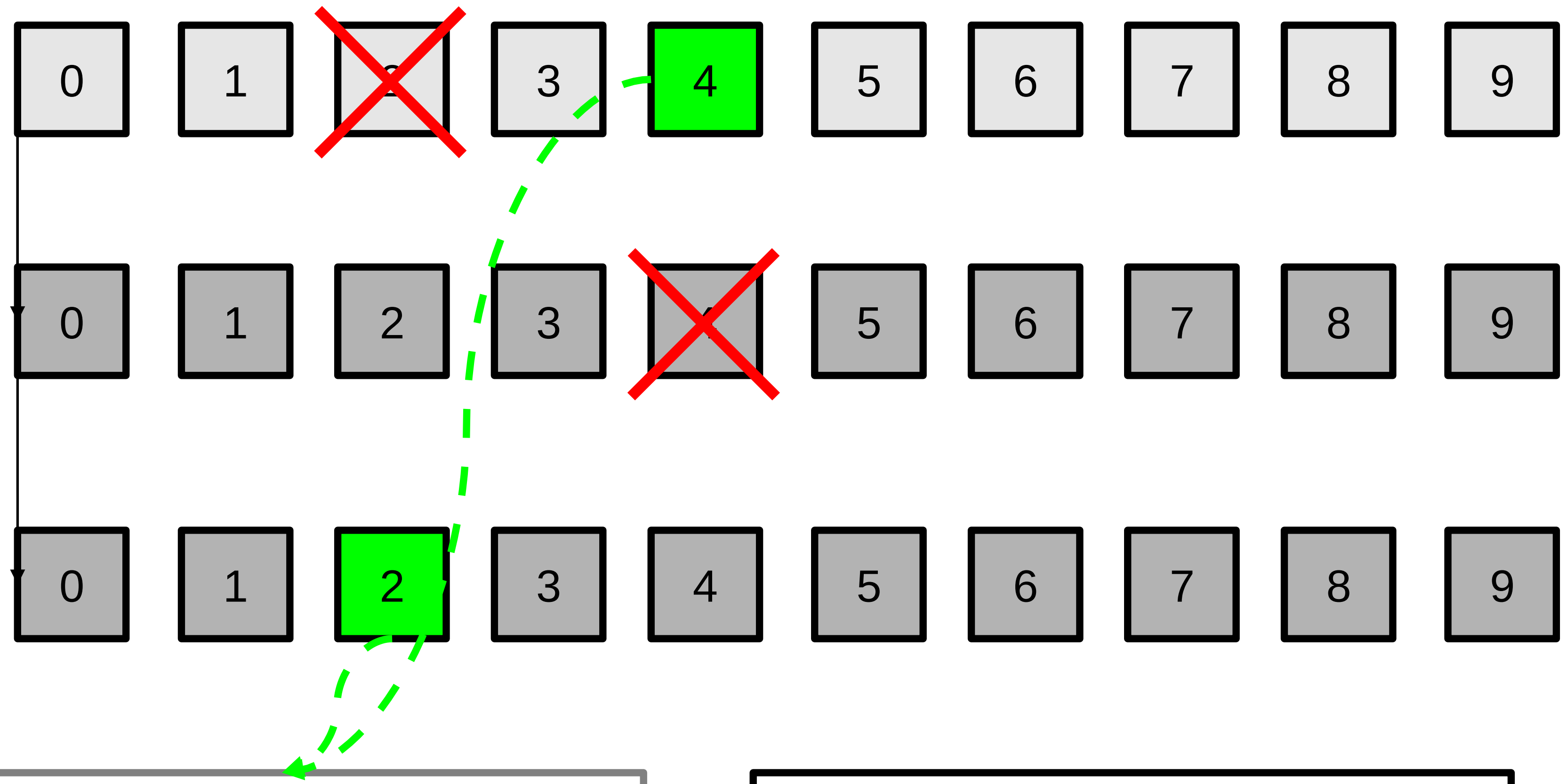
Replication



Replication



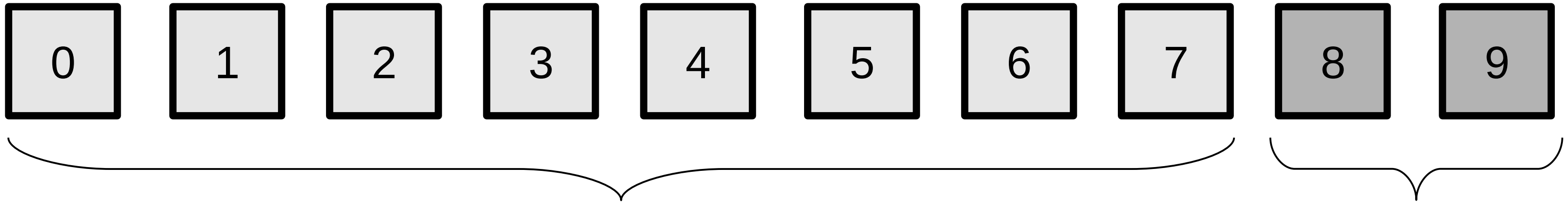
Replication



Recovers from any 2 failure

At the cost of 3X the storage!

(n,k) systematic MDS erasure code



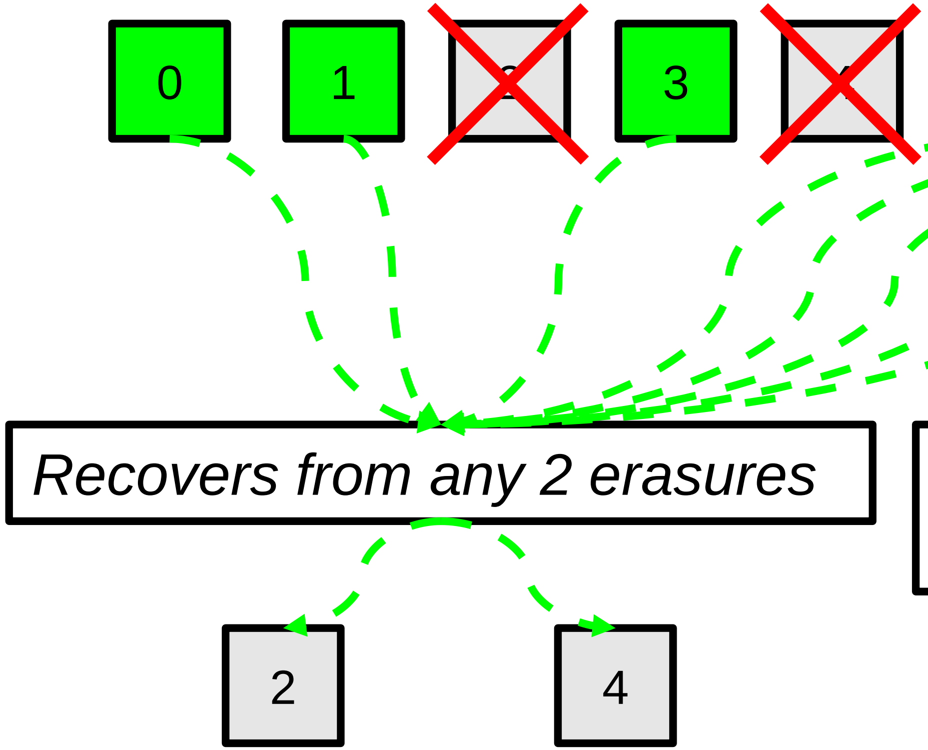
k systematic information nodes

n-k parity nodes

Recovers from any n-k erasures

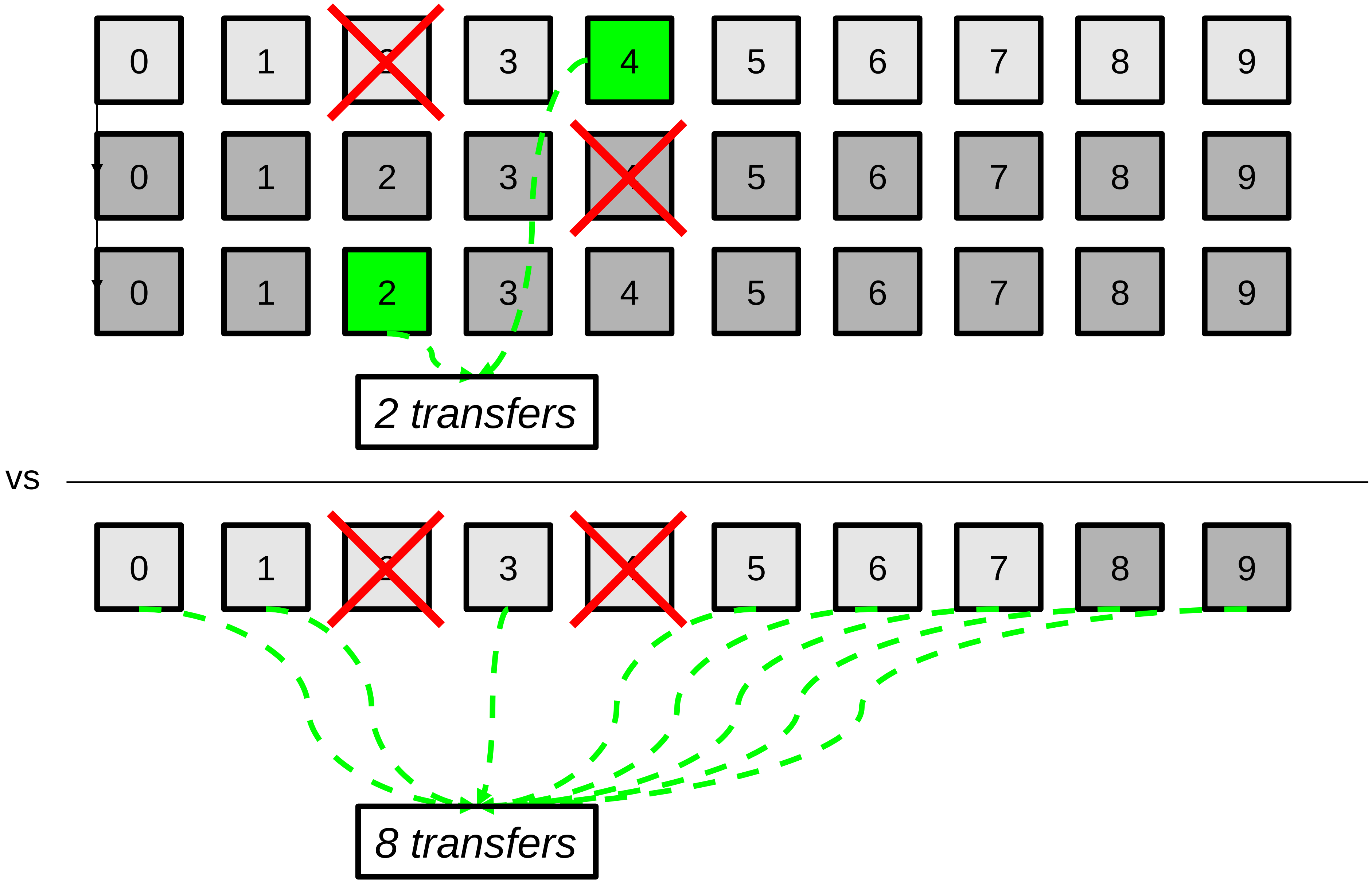
At the cost of n/k the storage!

(10,2) MDS erasure code – say Reed Solomon

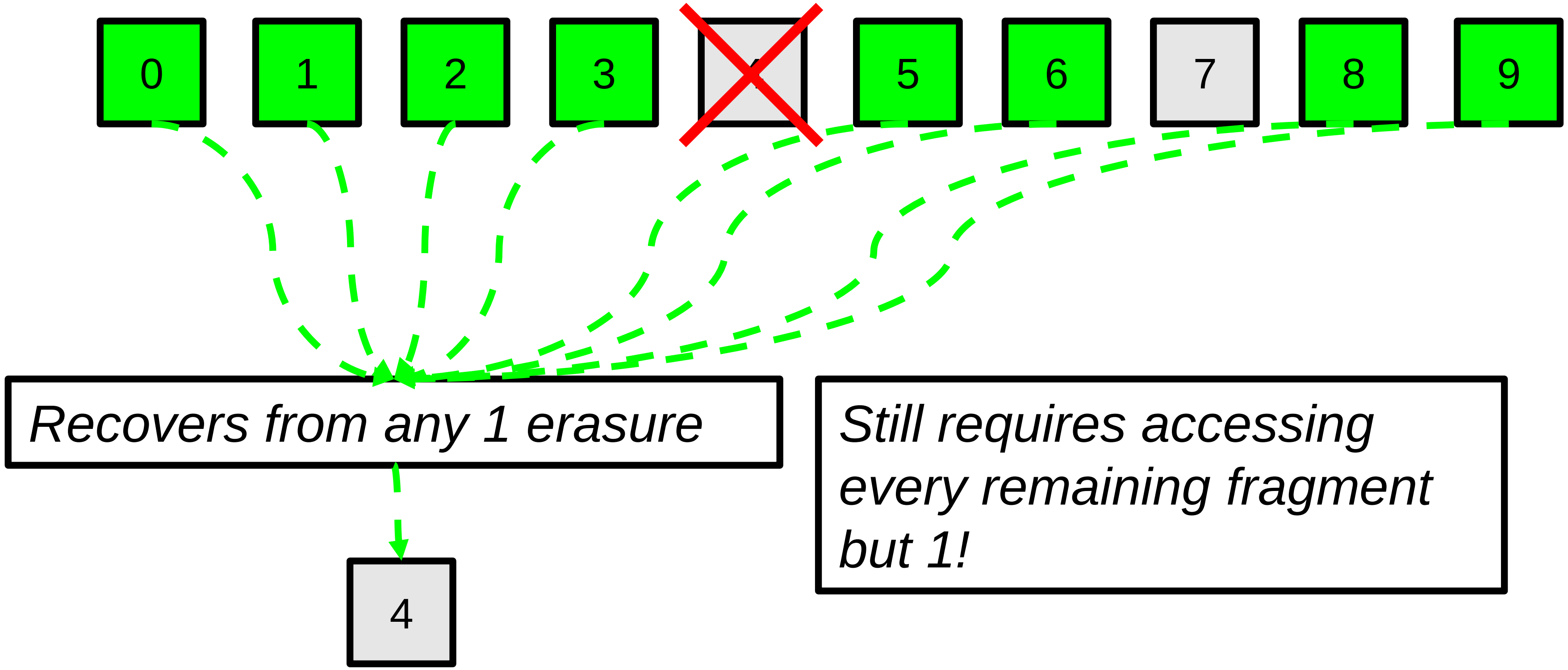


Requires accessing all the remaining fragments!

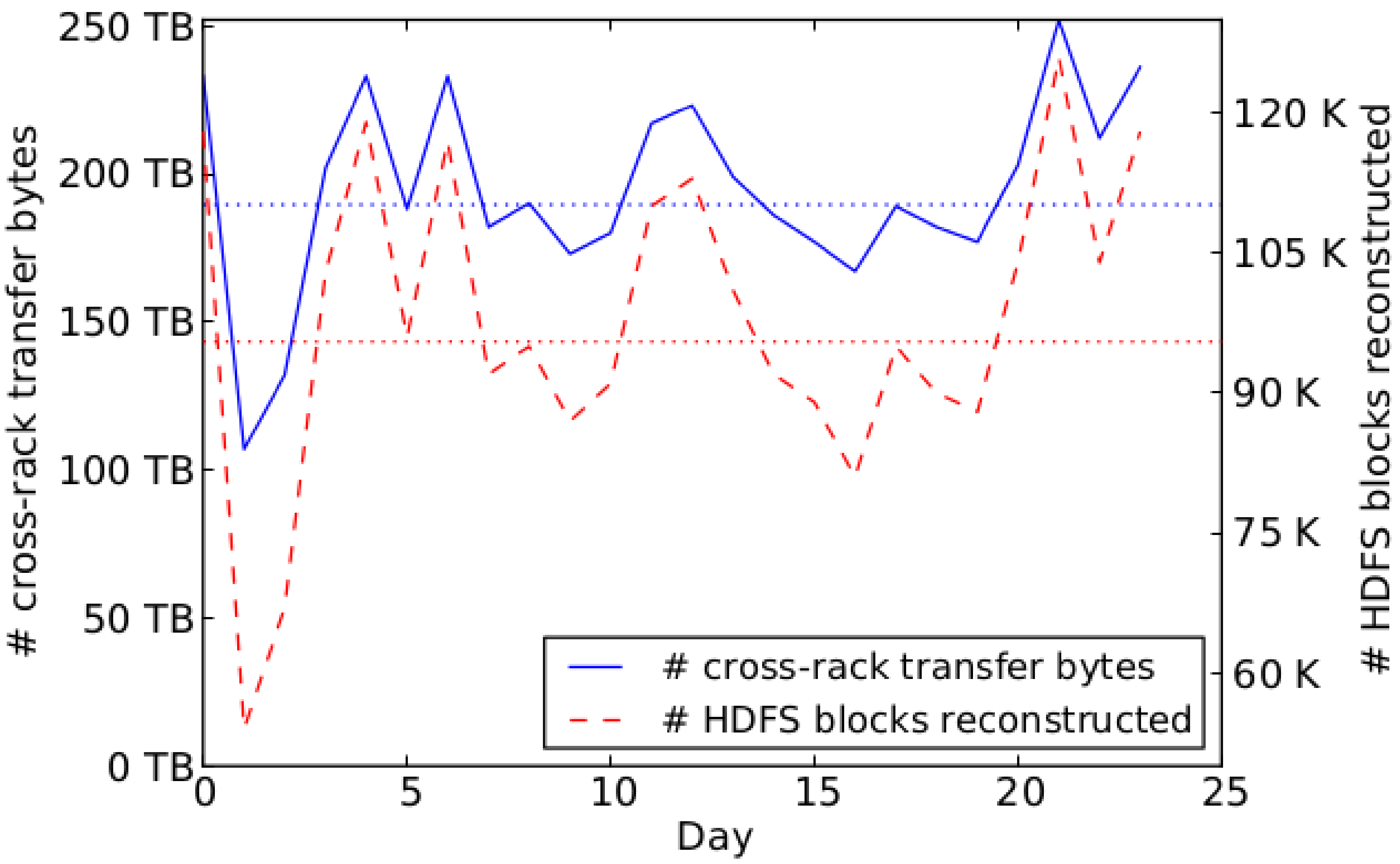
Recovery bandwidth



Recovery bandwidth



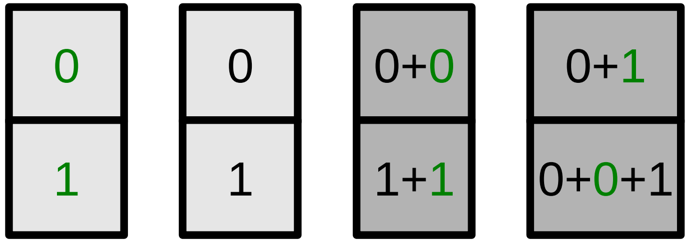
Network bandwidth cost



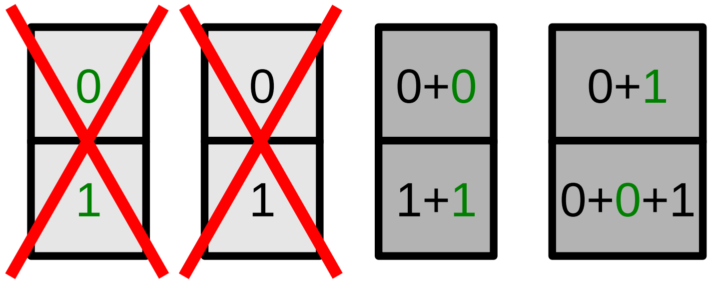
Source: Rashmi et. al., A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster

A regenerating code – Butterfly code

EnGad-Mateescu-Blagojevic-Guyot-Bandic ISIT’13 and PamiesJuarez et al. FAST’16

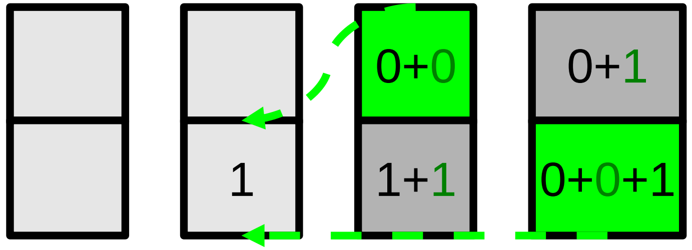


Butterfly code, k=2



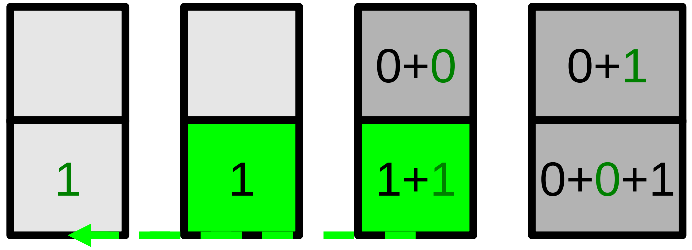
Recovers from any 2 failures

Butterfly code, k=2



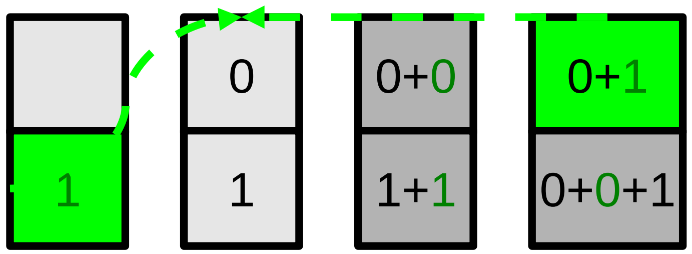
Recovers from any 2 failures

Butterfly code, k=2



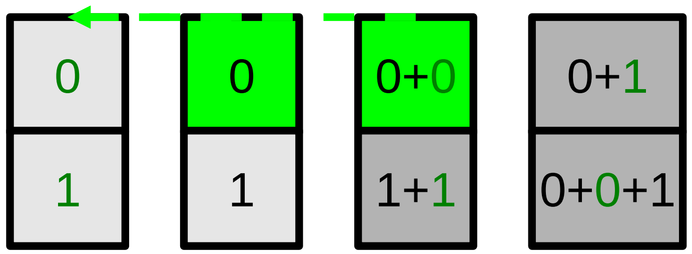
Recovers from any 2 failures

Butterfly code, k=2



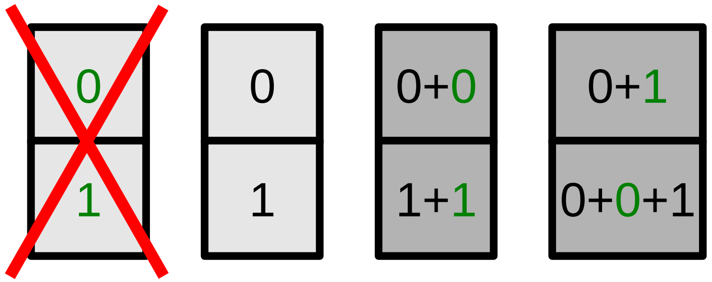
Recovers from any 2 failures

Butterfly code, k=2



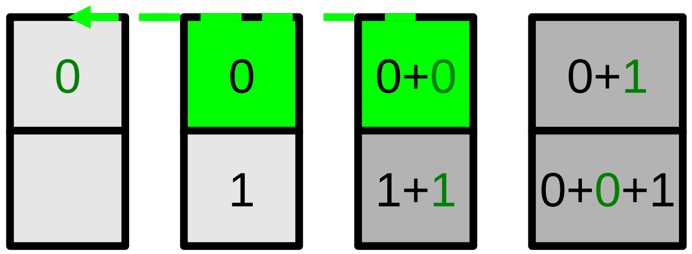
Recovers from any 2 failures

Butterfly code, k=2



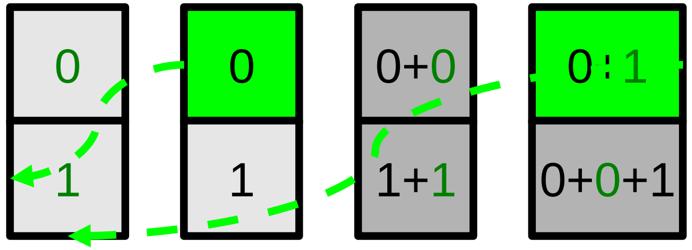
Recovers from any 1 erasure among systematic nodes

Butterfly code, k=2



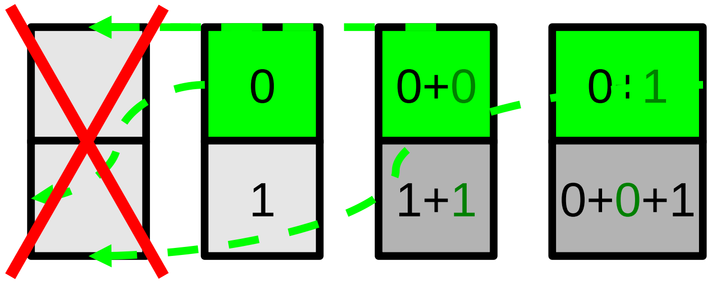
Recovers from any 1 erasure among systematic nodes

Butterfly code, k=2



Recovers from any 1 erasure among systematic nodes

Butterfly code, k=2



Recovers from any 1 erasure among systematic nodes

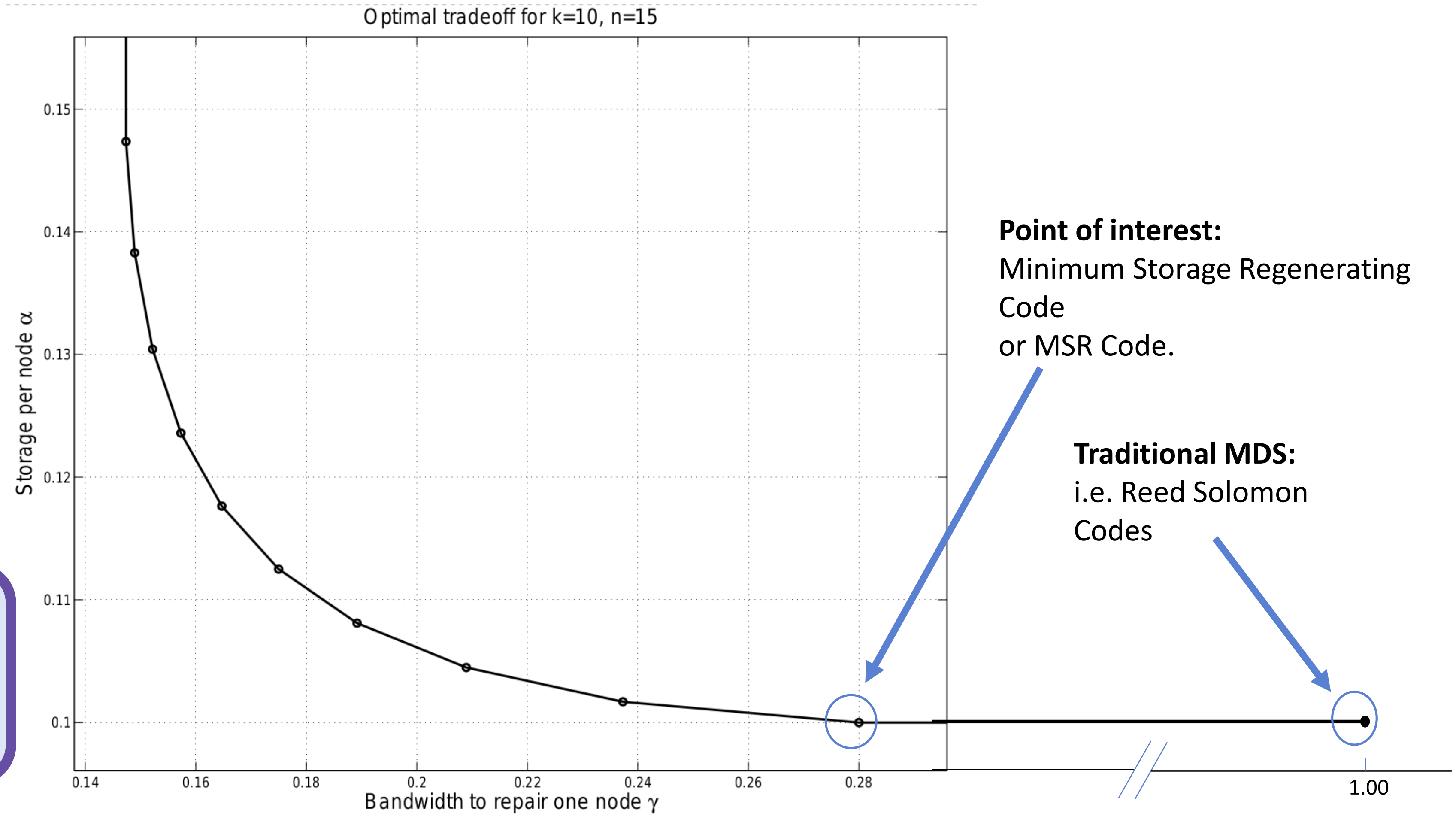
Using only ½ of the remaining data!

Regenerating Codes

Theoretically Appealing, Practically Difficult

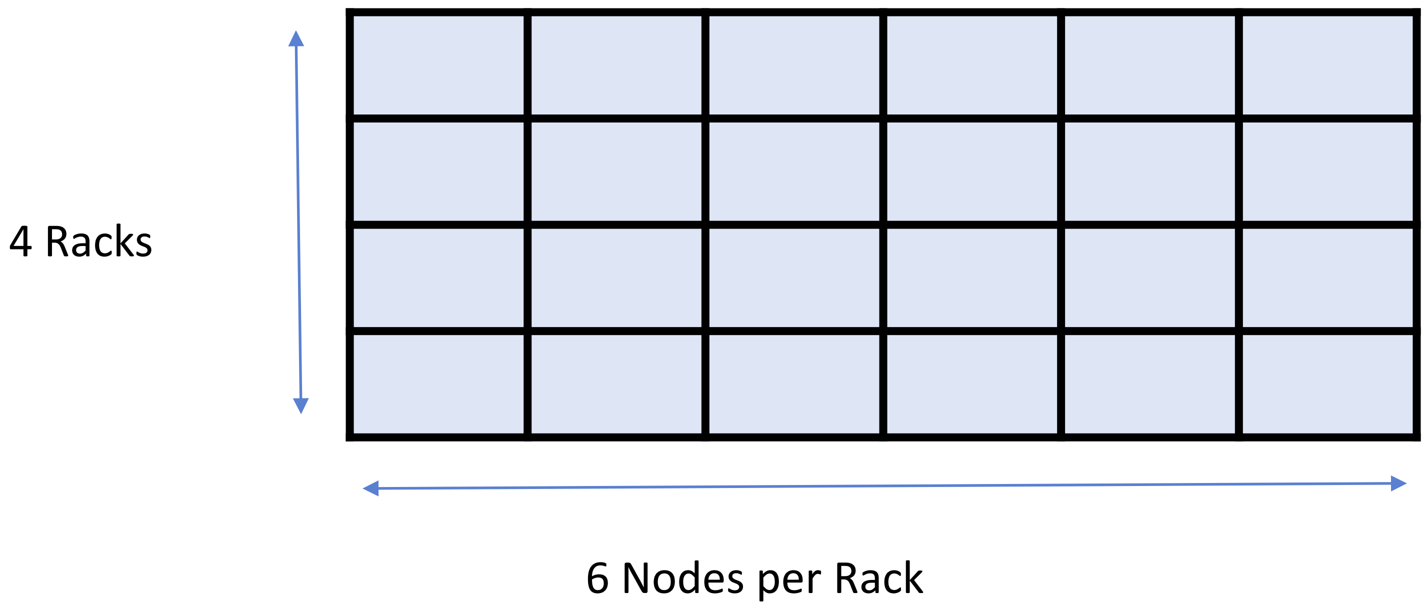
- Trade-off between:
  - Code Rate (storage overhead)
  - Repair Traffic (amount data read)

- Problems with MSR Codes:
  - Either requires exponentially growing finite fields, *(which makes them computationally inefficient)*,
  - or the number of chunks grows exponentially *(challenging its deployment in real systems).*



What about failure locality?

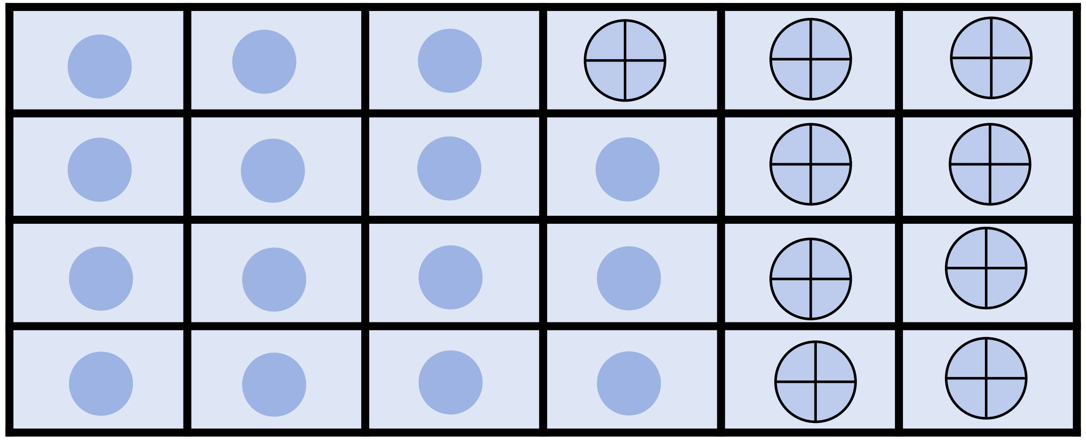
- Nodes fail independently with probability p.
- Racks fail independently with probability q.
- For some choices of p and q some 7-failure patterns are more likely than some 5-failure patterns.



Failure locality: Alternative 1

Ignore it…

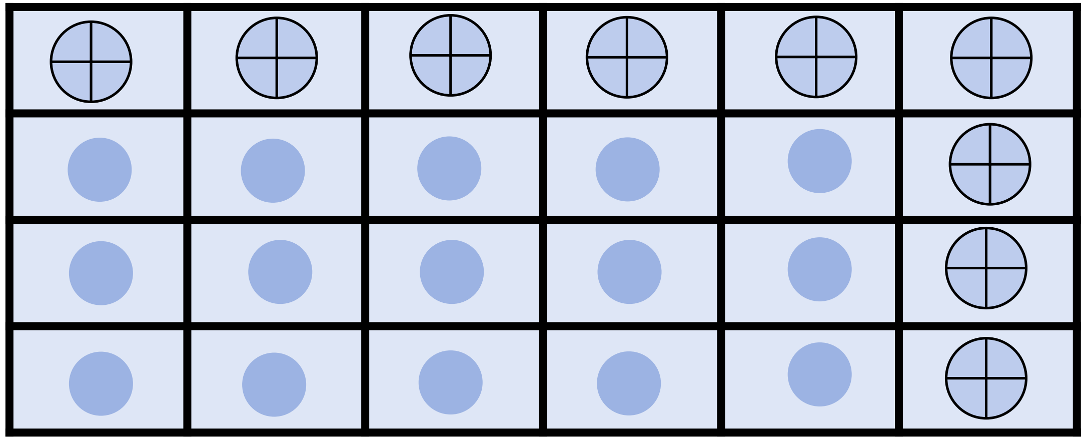
- Want to tolerate 1 rack + 3 more failures (9 total).
- Use RS code
  - Corrects any 9 failures



Failure locality: Alternative 2

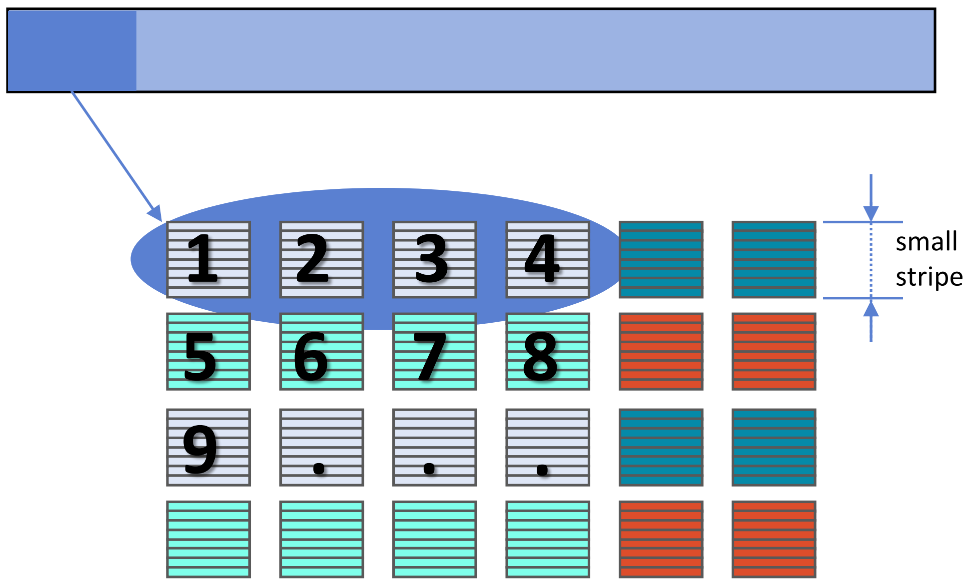
Take advantage of it…

- Want to tolerate 1 rack + 3 more failures (9 total).
- Use LRCs!
- Other benefit: low-overhead degraded reads!



Streaming vs. Buffered Encoding

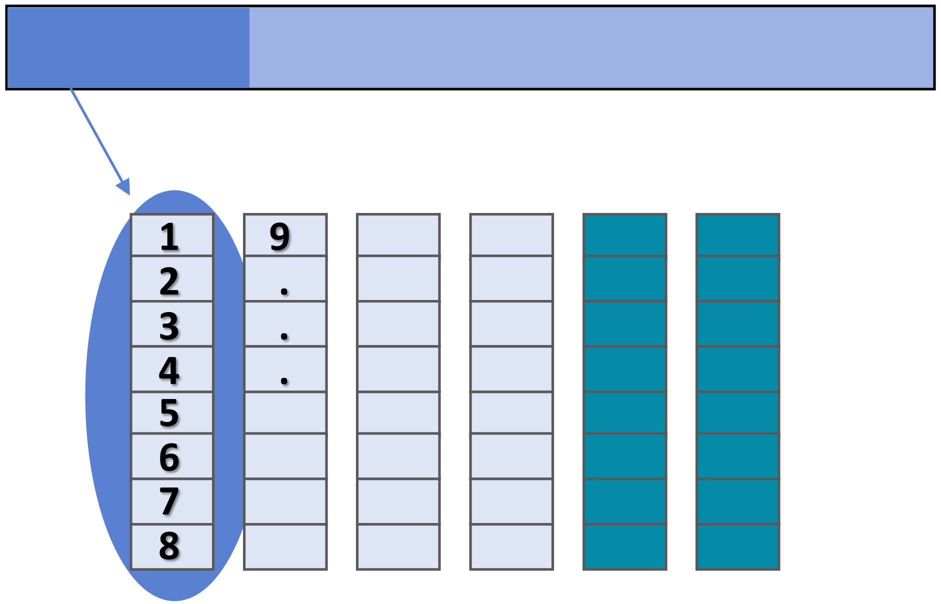
Streaming Encoding (Ceph):



- Input is split into small, chunks, and each of them encoded and stored individually.
- Streaming in/out, low time to first byte.
- PROBLEMS: Small chunks.

- CEPH:
  - default 4MB stripes

Buffered Encoding (HDFS):



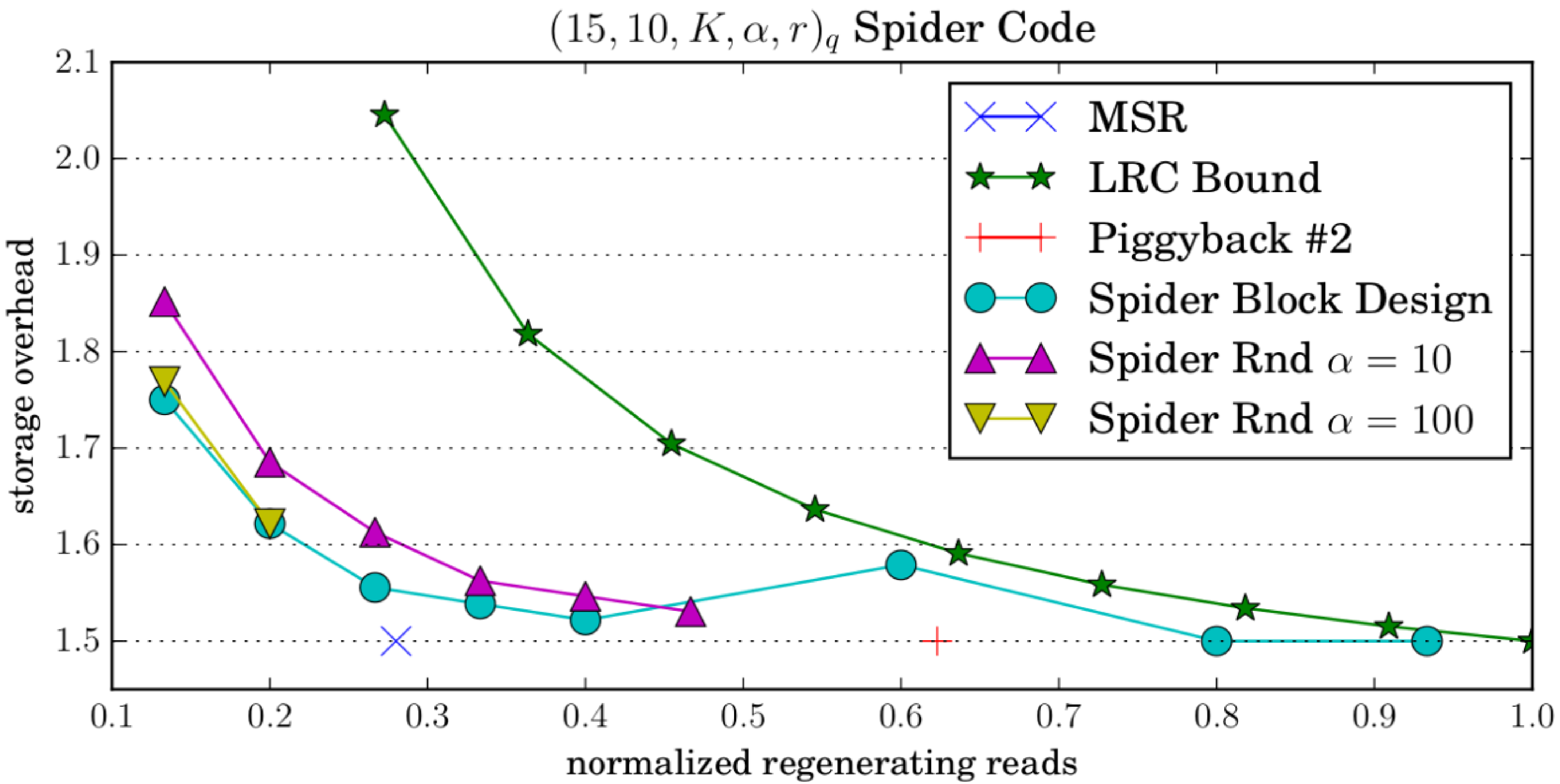
- The entire object needs to be buffered before encoding and storing it.
- PROBLEMS: Large memory, long latency.

- HDFS:
  - Defaults 64 MB blocks, initially replicated.
  - Batch process encodes k blocks together to form a codeword.

Tradeoffs

	Storage Overhead	Fast Repairs	Encoding Throughput	Static Reliability	Read Performance	Code Complexity
rate, $K/(n\alpha)$	✓			✗		
number of columns, n			✗	✓		
distance, d	✗		✗	✓		
field size, q			✗	✓		✗
regenerating		✓	✗			✗
locality, r	✗	✓		✗	✓	
generator sparsity			✓	✗		
systematic			✓		✓	✓

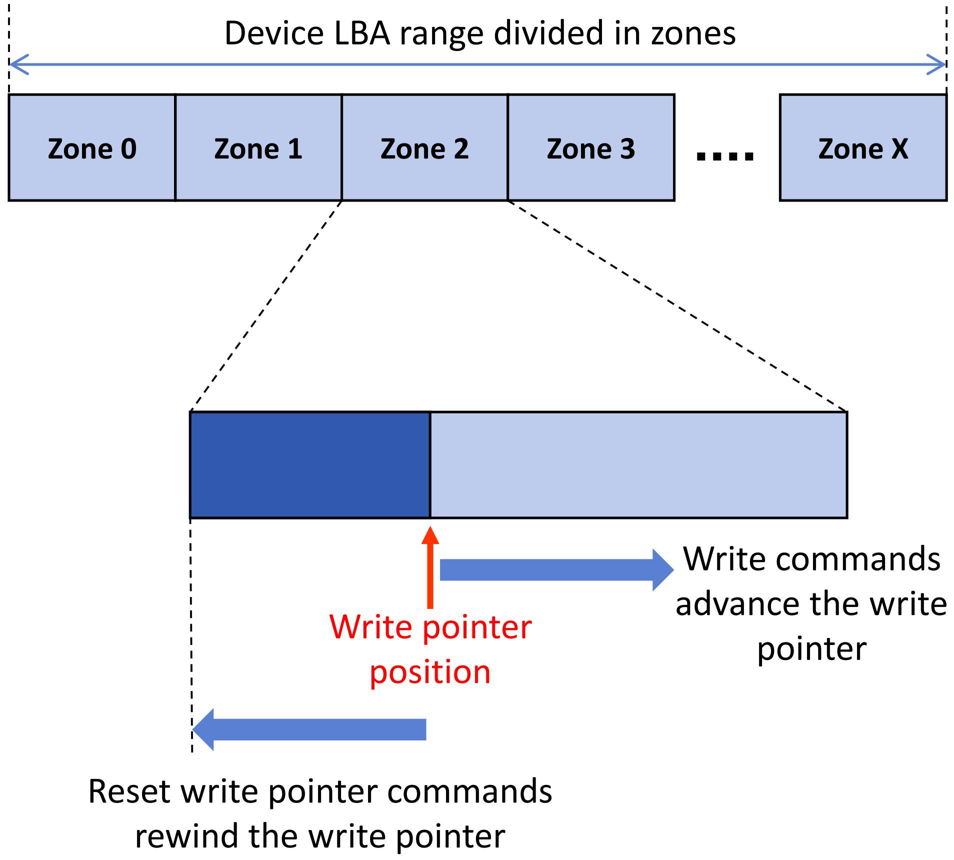
Tradeoffs: Spider codes [Pamies et. al 2016]



What are Zoned Block Devices?

The new paradigm in storage

- The storage device LBA range is divided into Zones.
- Writes within a zone must be sequential.
- Each Zone has a write pointer that keeps track of the position for the next write.
- Data in a Zone cannot be overwritten. The Zone must first be erased before it can be rewritten sequentially.

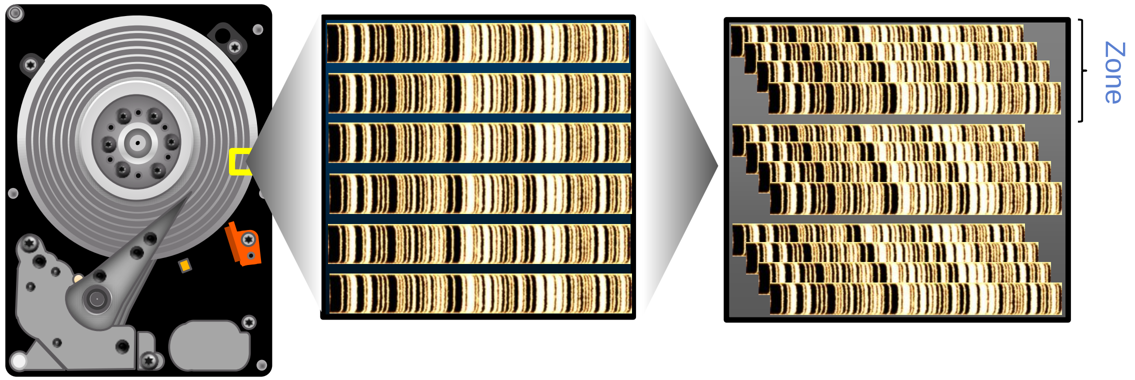


Why Zone Storage Technologies

Addressing the needs of large-scale data infrastructure

- Hard Drives (Shingled Magnetic Recording)

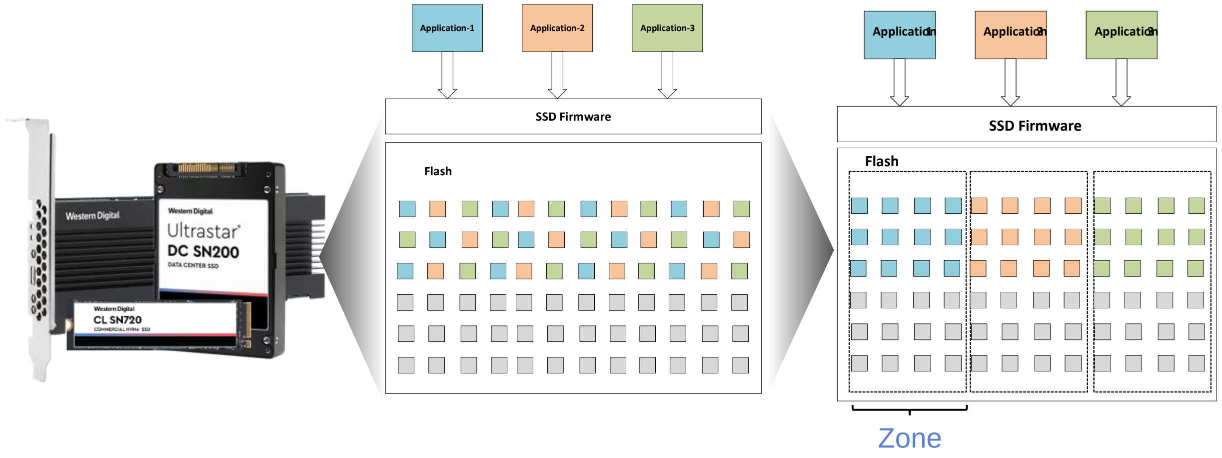
Conventional **Zoned**



  – SMR technology enables areal density growth and increased HDD Capacities

- Solid State Drives (Zoned Namespaces)

Conventional **Zoned**



  – Zoned Namespaces (ZNS)
    - Reduce SSD Write Amplification -> Increase usage
    - Reduce SSD Over-Provisioning -> Increase capacity
    - Reduce SSD DRAM needs → Reduce the cost
    - Improve at scale QoS → Reduce latency outliers

New constraints bring new tradeoffs!

- Streaming encoding vs. **buffered encoding**
- Minimal update vs. **whole codeword update**
- In-place updates vs. **multi-version coding**

- …and more to be explored
- Interested? Contact us!
  - Cyril.Guyot@wdc.com

Thank you!

Questions?

