RAIDP: replication with intra-disk parity

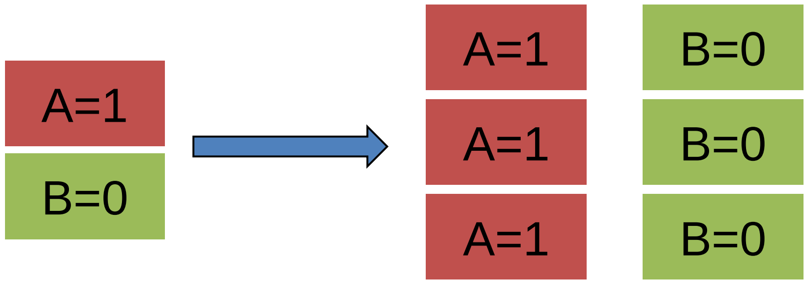
Eitan Rosenfeld, Aviad Zuck, Nadav Amit, Michael Factor, Dan Tsafrir

IBM, VMWare, Israel Institute of Technology

EuroSys'20

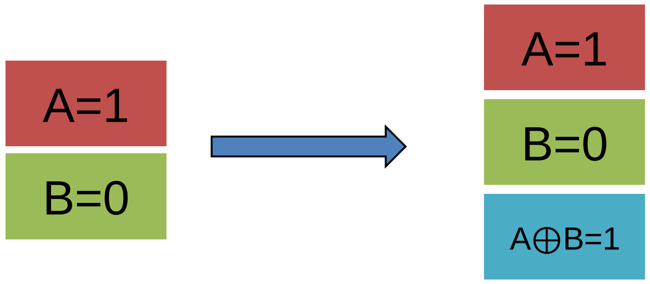
Problem: Disk Failure

- Failure of storage equipment in the data center is an inevitable event
  - So storage systems use **redundancy** when storing data
- Two forms of redundancy:
  - Replication



  - Waste storage space
  - Good performance

  - Erasure Code



  - Save storage space
  - Bad performance

Problem: Disk Failure

- **Current status:**
  - Replication is used for warm data only, most systems use 3-replica method
    - Very expensive! (Wastes storage, energy, network)
  - Erasure coding used for the rest data (cold data)
- **The main problem:**
  - **How to quickly recover from two simultaneous disk failures like 3-replica, but without restoring to a third replica for warm data?**

Contributions

- **RAIDP: A hybrid storage system use replication with intra-disk parity**
  - Achieves similar failure tolerance as 3-way replicated systems with 33% storage space saving
  - Better performance when writing new data, and small performance hit during updates
  - Uses considerably less network bandwidth for writes
- Real word application implementation and evaluation
  - 22% faster in HDFS writes, nearly same read performance with original HDFS
  - 14x faster recovers performance than RAID-6

Goals

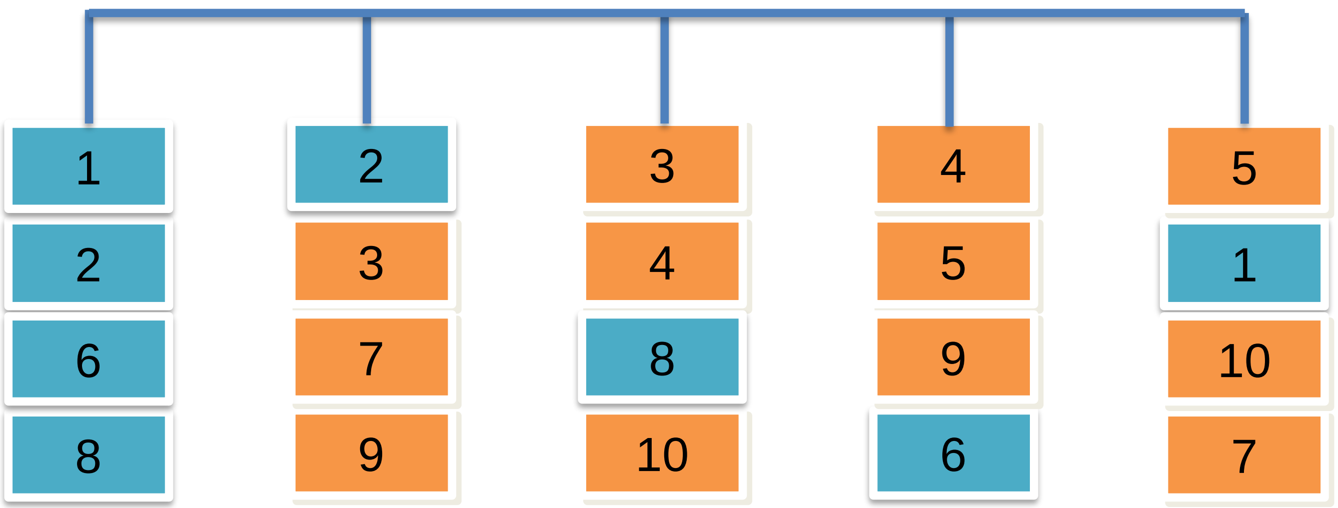
- Storage efficiency and data reliability
  - Saving storage space by reduce redundancy
  - Guarantee the data reliability of 3-replica method
- Low performance overhead
  - Maintaining 3-replica's read and write performance as much as possible
  - Guarantee efficiency during data recovery
- Lower cost
  - Reduce network bandwidth, storage devices, power consumption

Main Idea

- Hybrid storage system for warm data with only two copies of each data object (Manage data at Superchunk level)
  - Superchunks' Distribution:
    - Each of N disks divide into N-1 Superchunks
    - Any two disks share at most 1 Superchunk's copy
    - No same Superchunk on same disk
  - Disk Add On (Lstor):
    - Fail separately from local disk
    - Store local Superchunks' parity

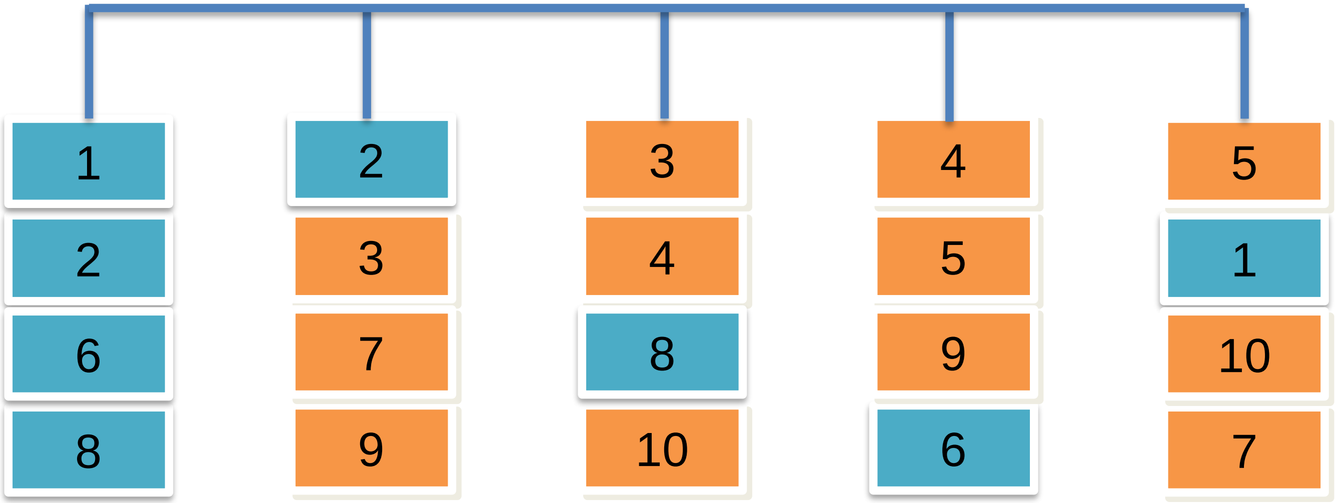
System Architecture

- **Each of the N disks is divided into N-1 superchunks**
  - **1-Mirroring**: Superchunks must be 2-replicated
  - **1-Sharing**: Any two disks share at most one superchunk
  - Handle 2 disk failure for most superchunks



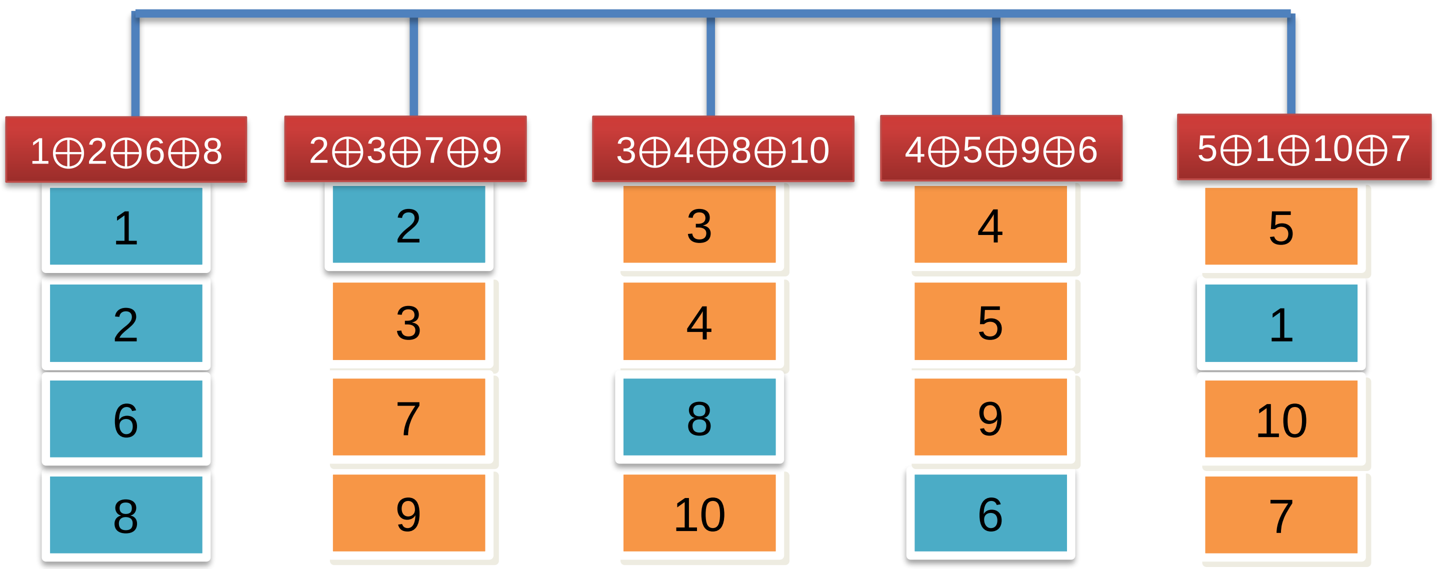
System Architecture

- **Each of the N disks is divided into N-1 superchunks**
  - **1-Mirroring**: Superchunks must be 2-replicated
  - **1-Sharing**: Any two disks share at most one superchunk
  - Handle 2 disk failure for most superchunks
  - How to handle 2 disk failure for all superchunks?



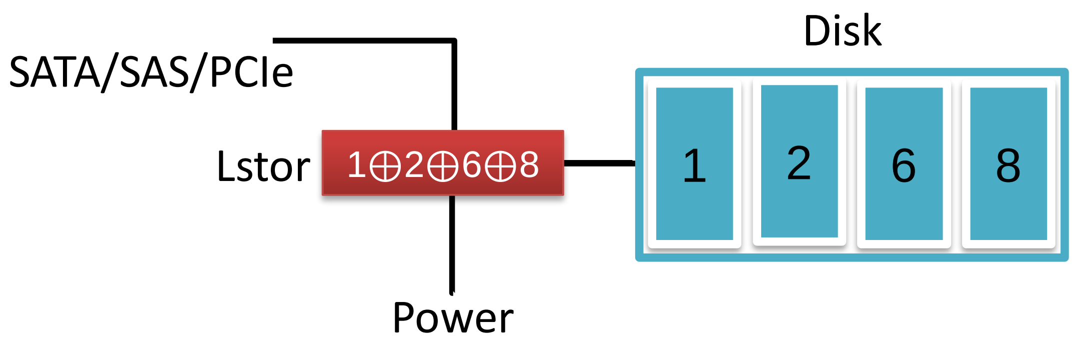
System Architecture

- **Each of the N disks is divided into N-1 superchunks**
  - **1-Mirroring**: Superchunks must be 2-replicated
  - **1-Sharing**: Any two disks share at most one superchunk
  - **Disk Add-On (Lstor):** Store parity of local disk's superchunks



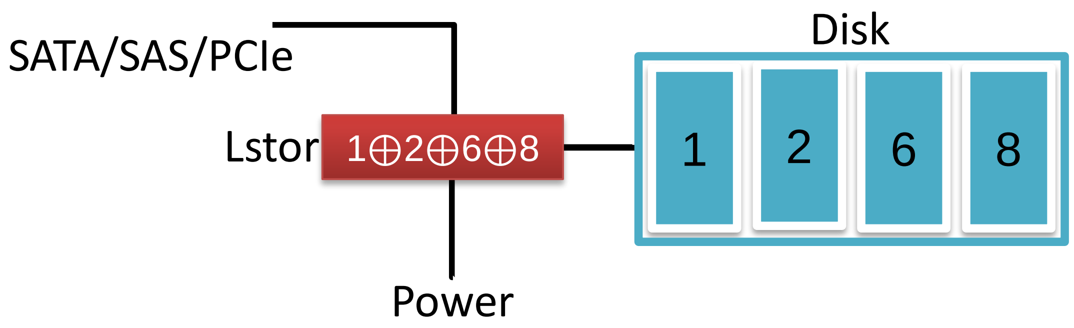
Lstor Design

- **Disk Add-on:**
  - Store parity of local disk's superchunks
  - Interposes all I/O to disk
  - Fails separately from the associated disk



Lstor Design

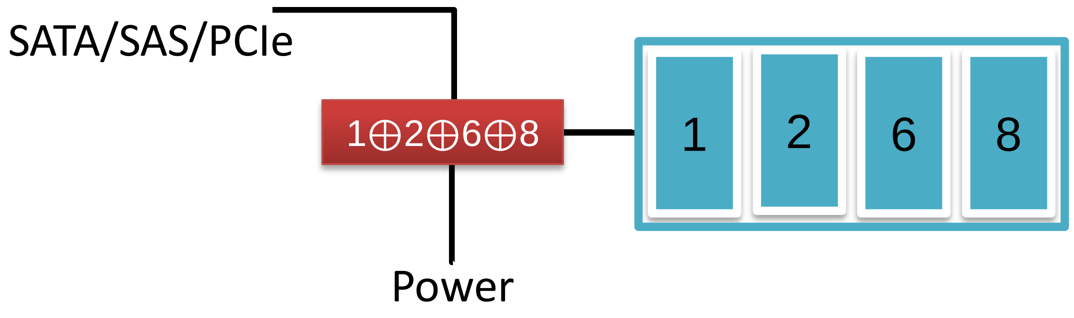
- **Disk Add-on:**
  - Store parity of local disk's superchunks
  - Interposes all I/O to disk
  - Fails separately from the associated disk



- **Goal:** Lstor need to be cheap, fast, and fail separately from disk
  - **Storage**: Enough to maintain parity (~$9)
  - **Processing**: Microcontroller for local machine independence (~$5)
  - **Power**: Several hundred Amps for 2–3 min from small supercapacitor to read data from the Lstor

Lstor Design

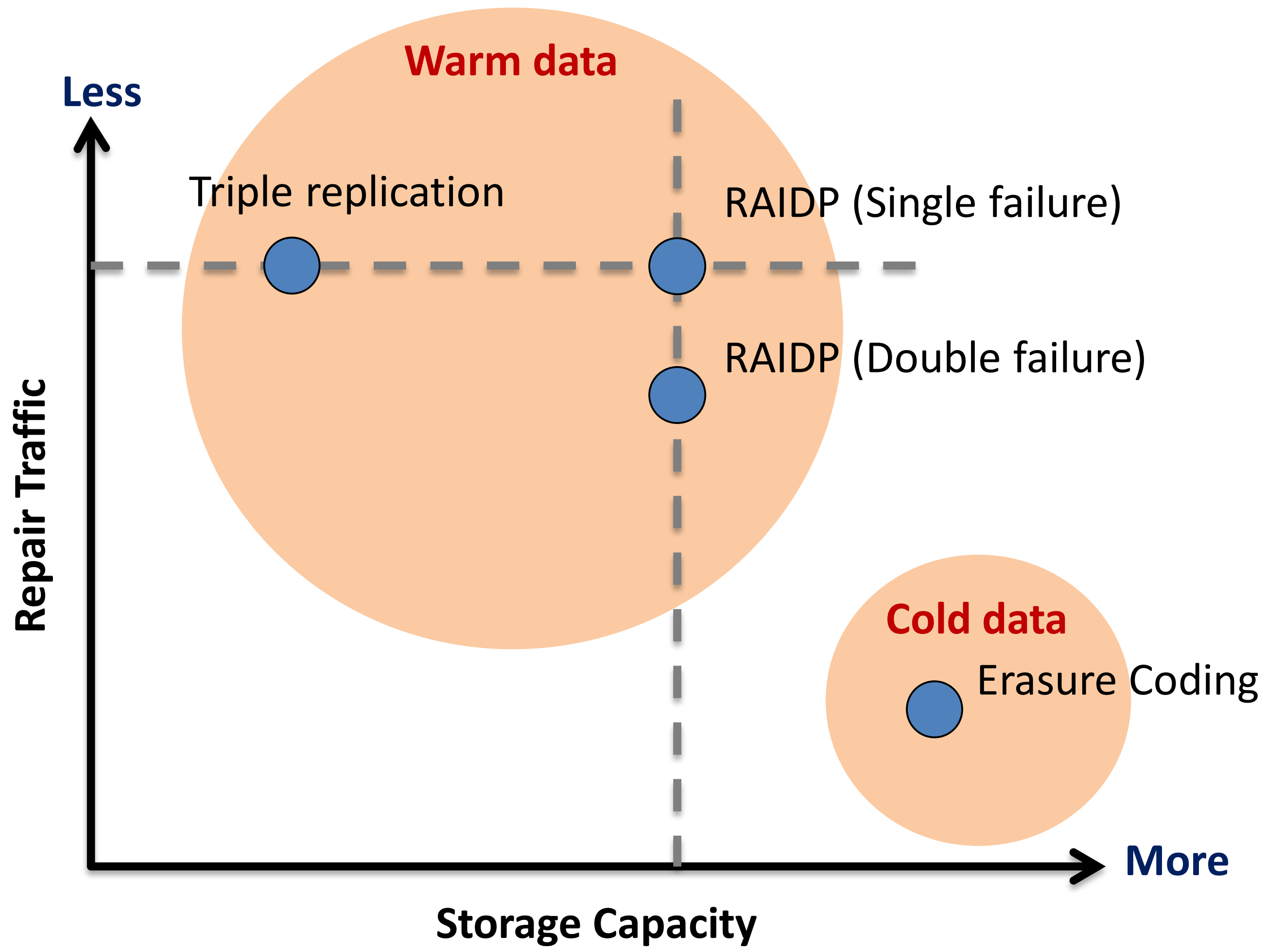
- **Disk Add-on:**
  - Store parity of local disk's superchunks
  - Interposes all I/O to disk
  - Fails separately from the associated disk



- **Goal:** Lstor need to be cheap, fast, and fail separately from disk
  - **Storage**: Enough to maintain parity (~$9)
  - **Processing**: Microcontroller for local machine independence (~$5)
  - **Power**: Several hundred Amps for 2–3 min from small supercapacitor to read data from the Lstor

Commodity 2.5" 4TB disk for storing an additional replica costs $100:
66% more than a conservative estimate of the cost of two Lstors

Storage Method Analysis



- Saving storage space
- Reduce repair traffic

Implementation

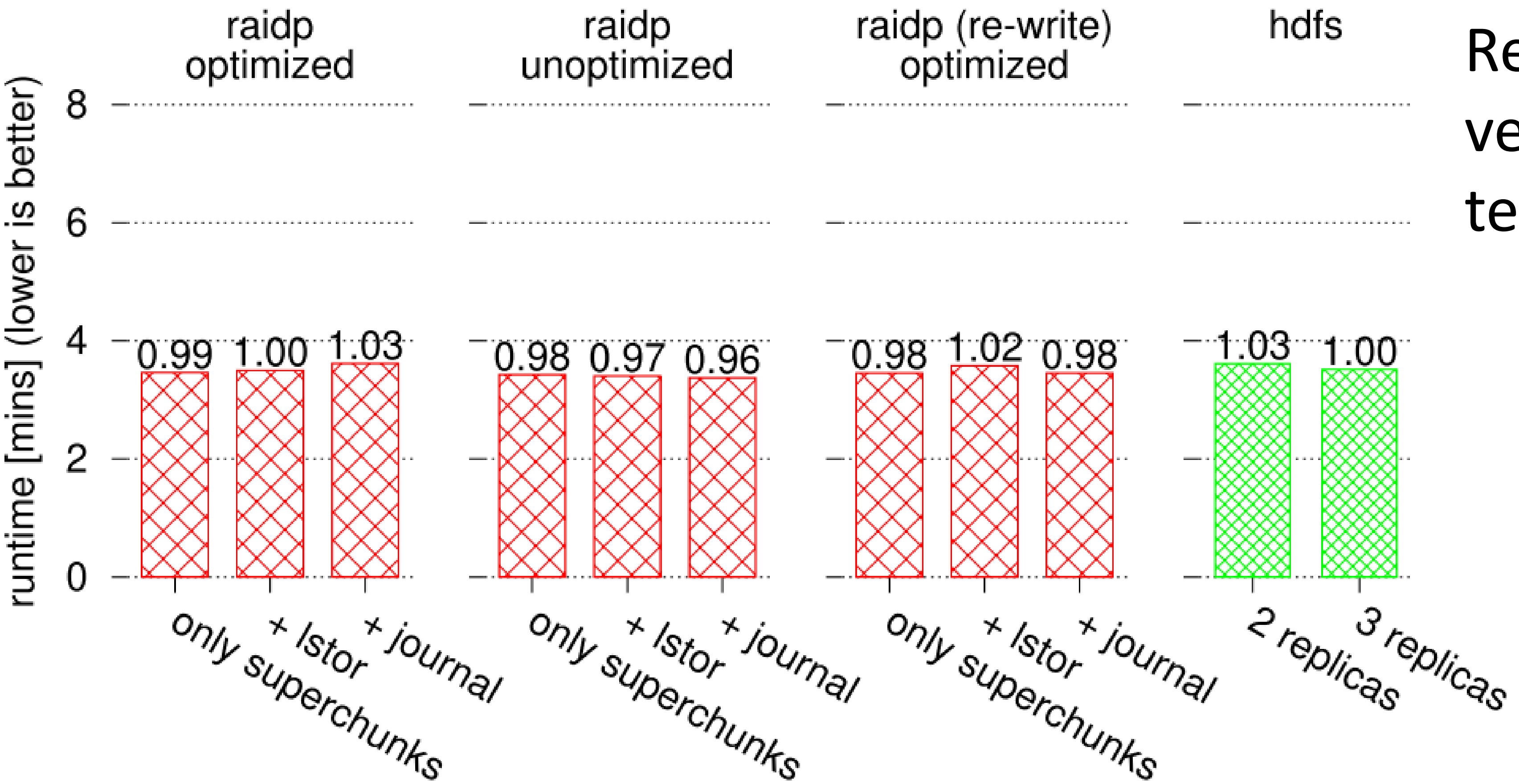
- **RAIDP implemented in in Hadoop 1.0.4**
  - Append-only
  - Updates-in-place
- **3K LOC extension to HDFS**
  - Pre-allocated block files to simulate superchunks
  - Lstors simulated in memory
  - Added crash consistency and several optimizations
- **Optimization**
  - Process based on 64MB block rather than 64KB packet in HDFS
  - locking mechanism when a block file has fully accumulated (reduce syncing & seeking)

Evaluation

- **Compare objects & Benchmarks**
  - HDFS with 3-replica and HDFS with 2-replica
  - Hadoop standard benchmarks
  - TeraSort and Wordcount benchmarks via Intel's HiBench suite
- **Testbed**
  - 16-node cluster, with Intel Xeon E3 CPU, 16GB RAM, 7200RPM HDD, and 10GbE
  - 6GB superchunks, ~800GB cluster capacity
- **Target**
  - **The impact of read and write performance?**
  - **The efficiency and overhead during failure recovery?**

Evaluation

➢ **RAIDP and HDFS read performance under different configurations.**

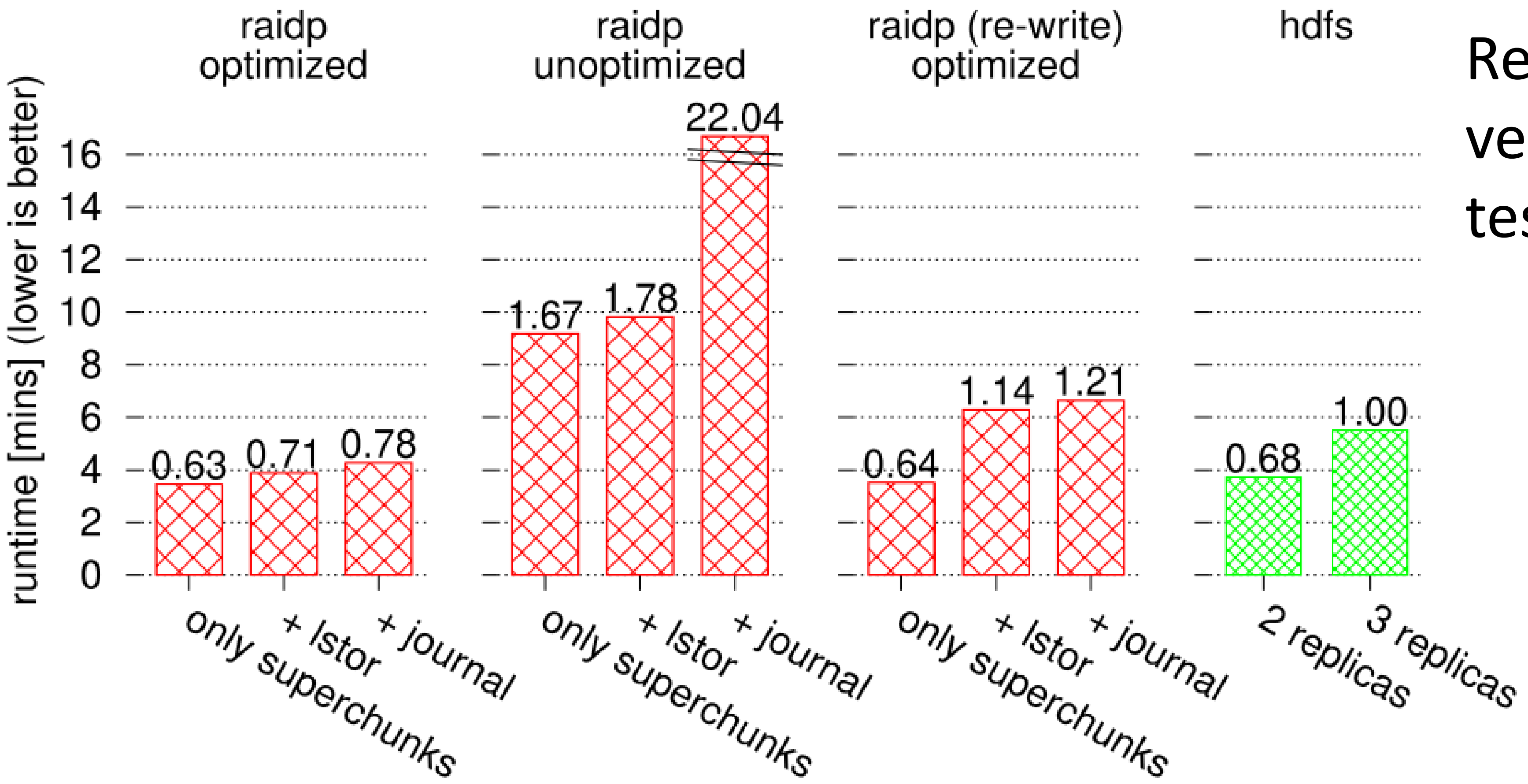


Re-write is version update test

➢ Almost same read performance

Evaluation

➢ **RAIDP and HDFS write performance under different configurations.**

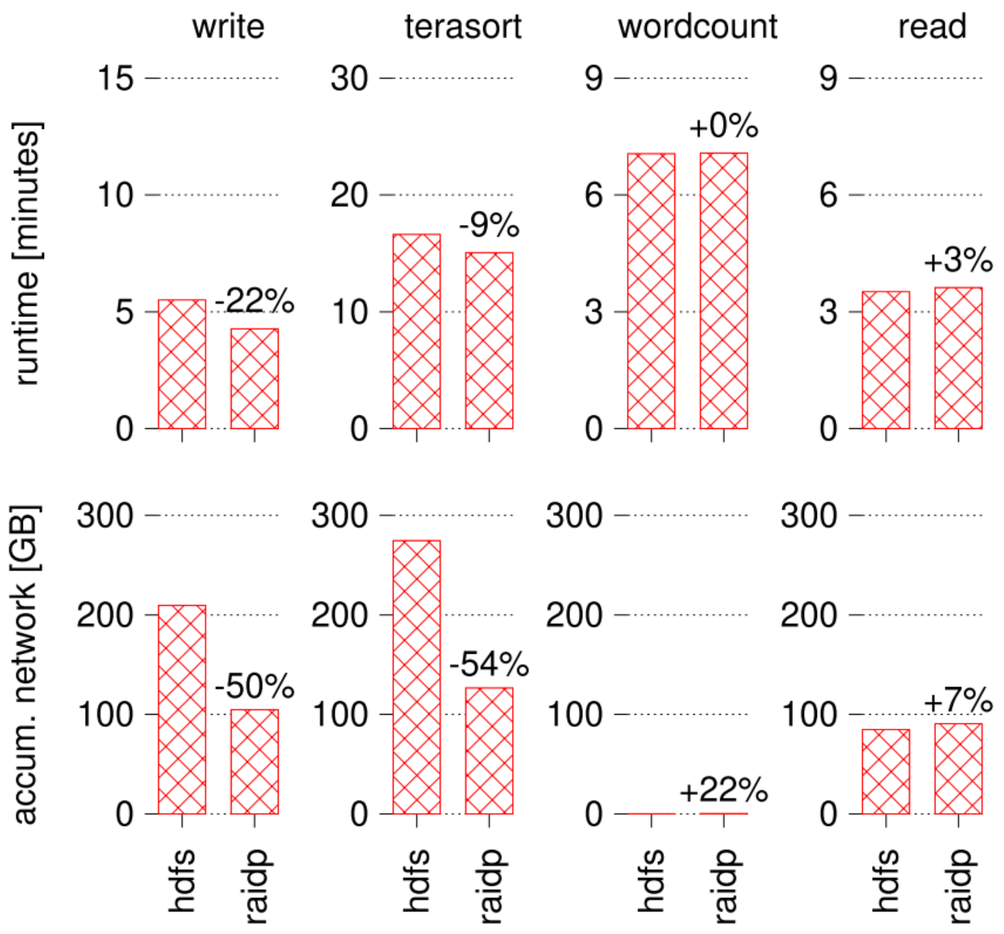


Re-write is version update test

➢ Improved write performance

Evaluation

- **RAIDP versus HDFS-3 performance**



- **Terasort**: sorts 100GB data (balanced read & write).
- **Wordcount**: computes string frequency over a 100GB input (most I/O is read)

Evaluation

- **Recovery performance**

System Type	Chunk Size	10GbE	1GbE
RAIDP Byte range lock	4MB	125(sec)	827(sec)
	64MB	160(sec)	848(sec)
RAIDP Superchunk lock	4MB	187(sec)	850(sec)
	64MB	211(sec)	852(sec)
RAID-6	4MB	1823(sec)	12300(sec)
	64MB	2227(sec)	13146(sec)

- RAID-6 is cross network
- Chunk size is HDFS chunk size

RAIDP recovers 14x faster

Conclusion

- **RAIDP achieves similar failure tolerance as 3-way replicated systems**
  - Better performance when writing new data
  - Small performance hit during updates
  - Requires 33% less storage
  - Uses considerably less network bandwidth for writes
  - Recovery is much more efficient than EC
- **Opens the way for storage vendors and cloud providers to use 2 (instead of 3, or more) replicas**

Q&A