Delayed Parity Update for Bridging the Gap between Replication and Erasure Coding in Server-based Storage

Takayuki Fukatani[1,2], Hieu Hanh Le [1], Haruo Yokota[1]

[1]Tokyo Institute of Technology [2] Hitachi Ltd.

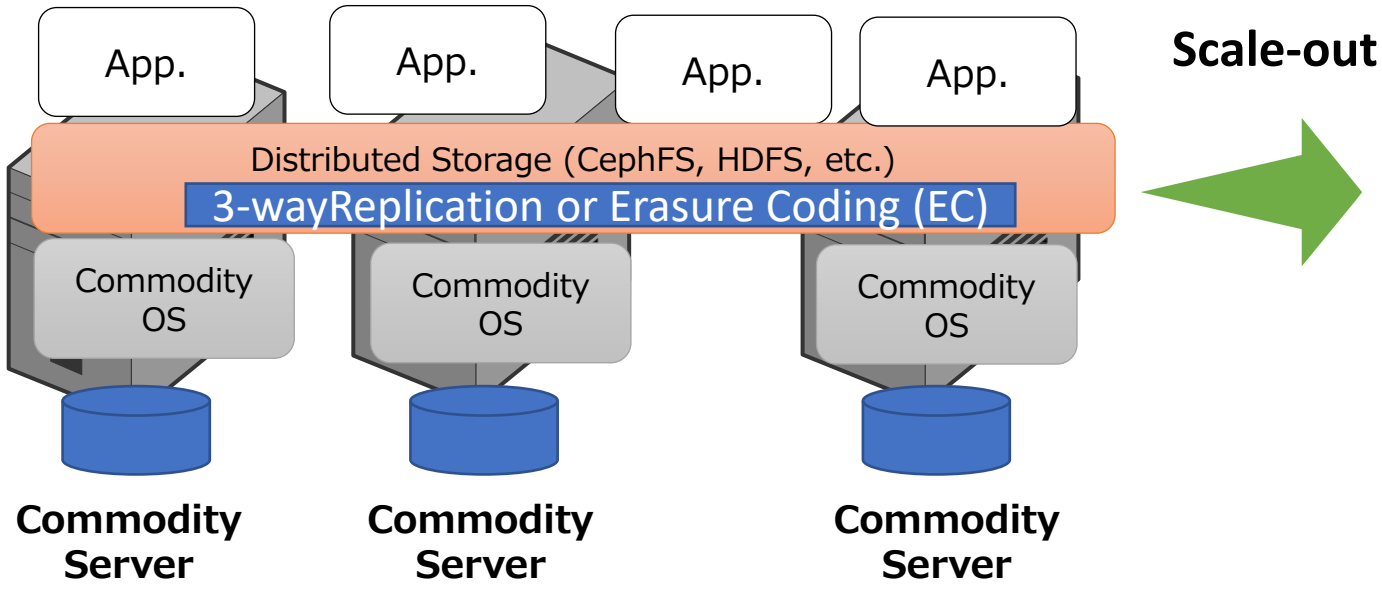
ADMS 2021

Backgrounds

- Demand for using inexpensive commodity server-based storage in a wide range of applications
- For legacy applications (DB, VDI, etc.), the low random performance of server-based storage becomes an issue
- Many studies have been focusing on improving the random performance of server-based storage

Server-based Storage

- A scale-out storage system consisting of many commodity servers
  - ➢ Distributes redundant data among servers for high-throughput and high reliability
  - ➢ 3-way Replication and Erasure Coding (EC) is used for data redundancy
- Inexpensive alternative for dedicated storage appliances



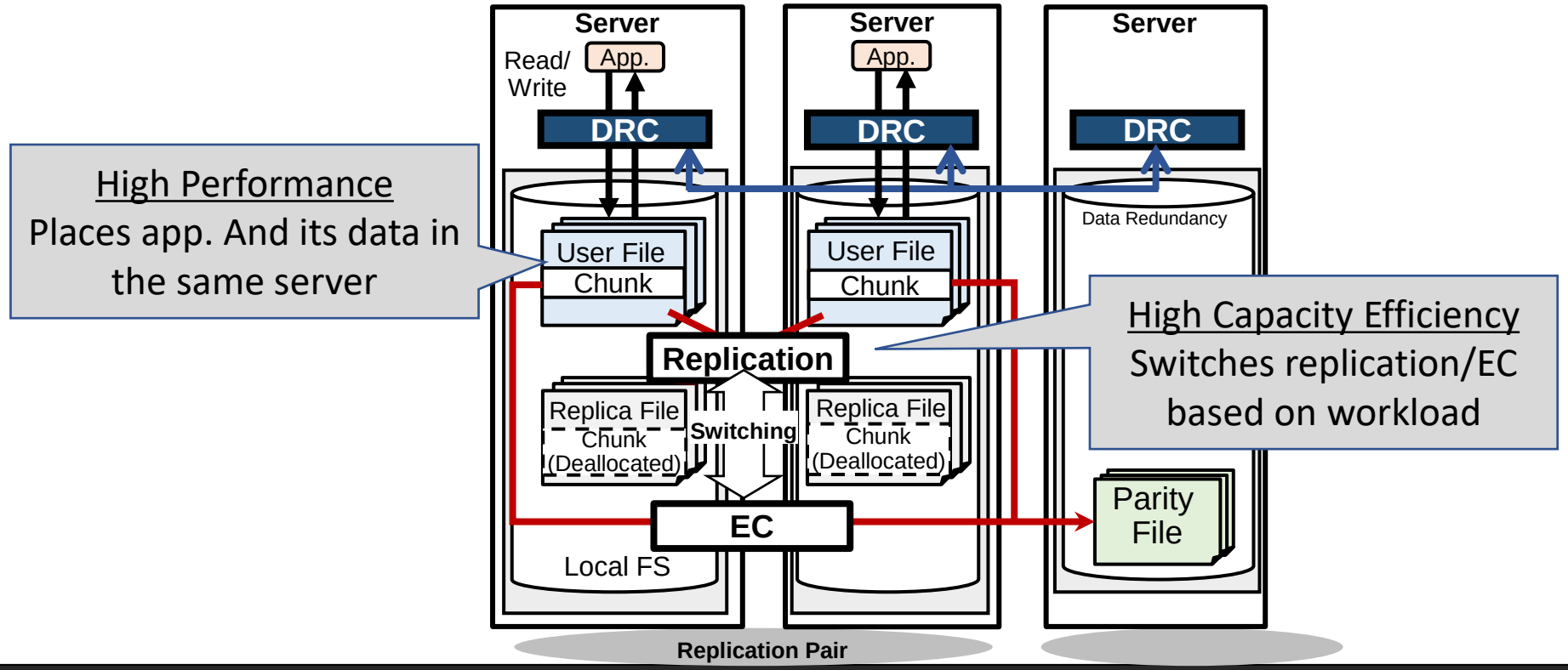
Applying Server-based Storage to Legacy Applications

- The use of server-based storage is expanded from **cloud applications** to **legacy applications**
  - Cloud applications
    - ✓ Data analysis, archiving, and etc.
    - ✓ Sequential access is dominant
  - Legacy applications
    - ✓ DB, VDI , and etc.
    - ✓ Random access is dominant

- Issues
  - Network-based distributed storage is disadvantageous for random-intensive workload in terms of performance
  - Trade-off relationship between capacity efficiency and performance in redundancy methods
    - ✓ Replication: High random write performance, but low capacity efficiency
    - ✓ EC: High capacity efficiency, but low random write performance/

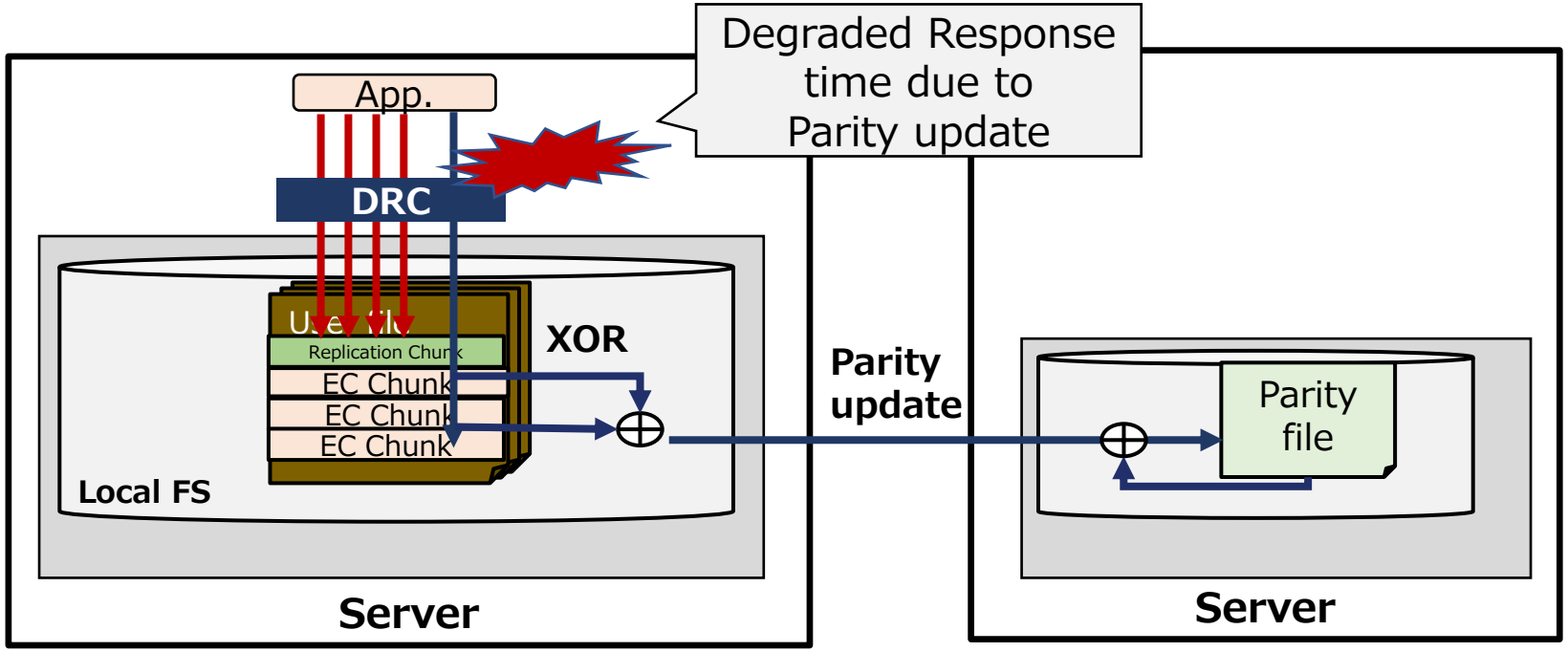
Dynamic Redundancy Control (DRC) [1]

- DRC achieves high performance while maintaining high capacity efficiency
  - Places an application and its data in the same server for high random performance
    - ➢ Makes the local user data redundant between servers.
  - Dynamically switches replication/EC among servers based on workload for high capacity efficiency



Challenge in DRC

- DRC has an issue with degraded random write response time for EC data
  - Write to EC data entails parity update and degrades response time from replication
  - Even though frequency of EC data is low, unstable data access causes usability degradation of entire system



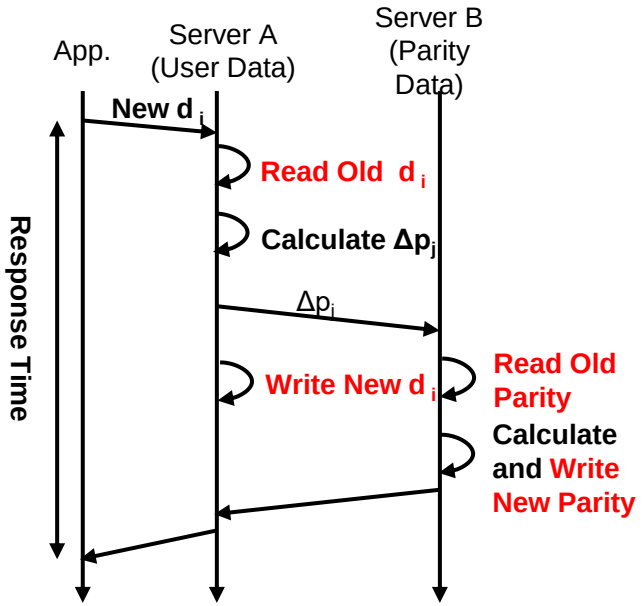
Conventional Approaches:
EC Write Optimizations

- Parity Logging [2] and Speculative Parity Write(SPW) [3] are proposed to improve random write performance

Flow of EC Write

Naïve EC

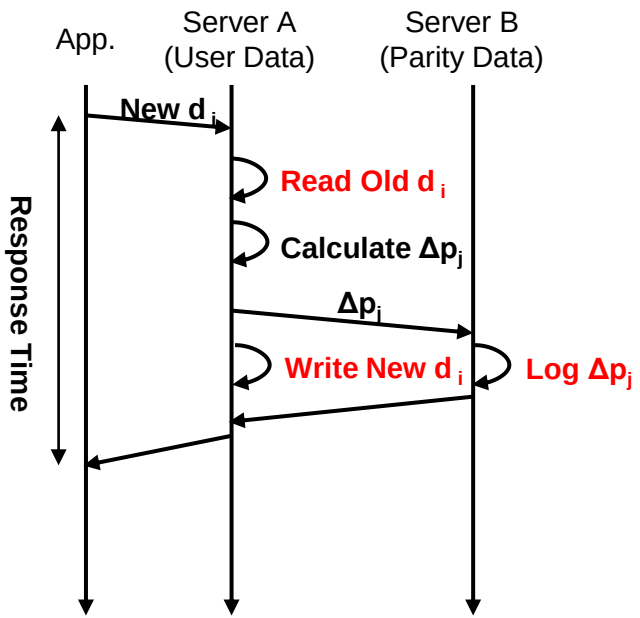
Calculate the new parity from the old/new data and the old parity



2 Reads/ 2 Writes

Parity Logging

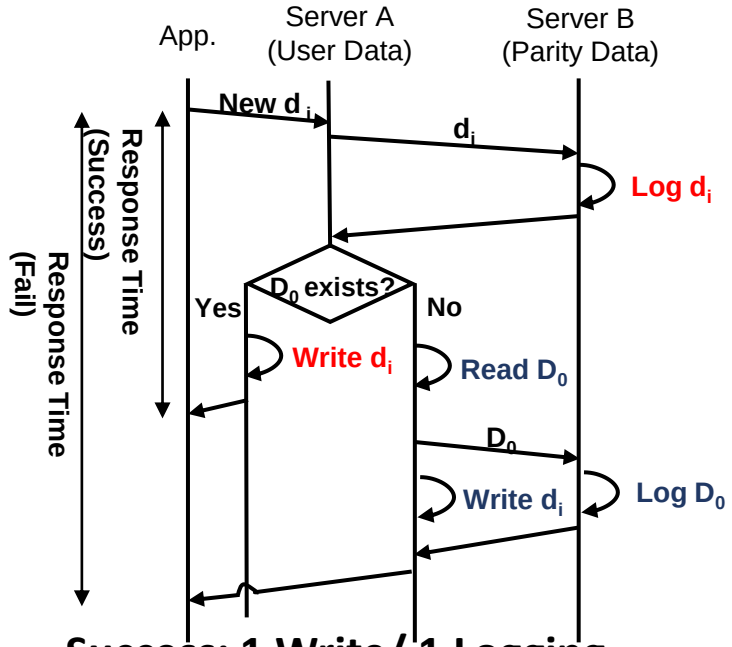
Parity delta logging to avoid reading the old parity



1 Read/ 1 Write/ 1 Logging

Speculative Parity Write(SPW)

Data logging to avoid reading the old data



Success: 1 Write/ 1 Logging

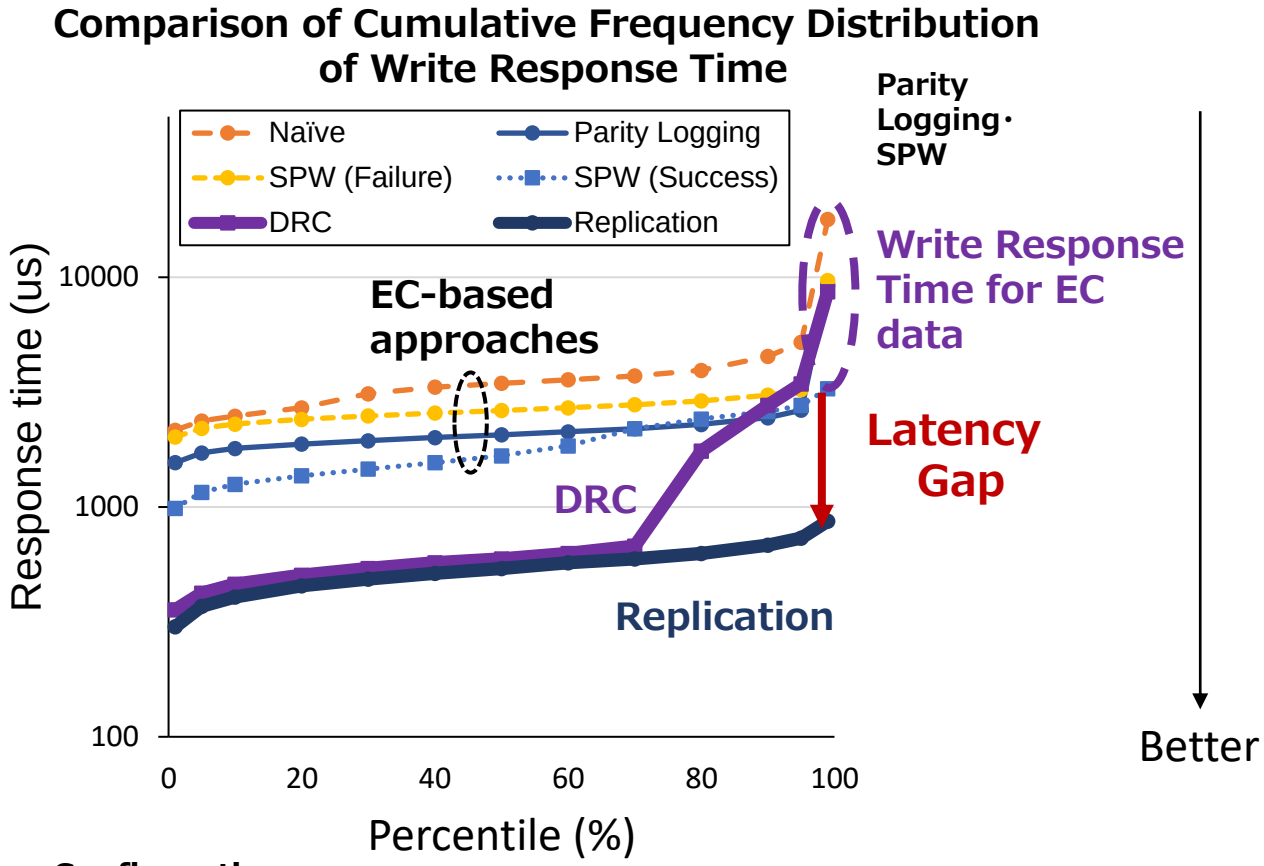
Fail: 1 Read/ 2 Writes/ 2 Logging

[2] Jeremy C. W, et al., "Parity logging with reserved space: towards efficient updates and recovery in erasure-coded clustered storage," FAST'14

[3] Y. Chang, et al., "PBS: An efficient erasure-coded block storage system based on speculative partial writes," ACM TOS, 2020

Preliminary Examination

- We evaluated the random write response time
  - ➢ Replication
  - ➢ Naïve EC, Parity logging, SPW
  - ➢ DRC (Replication + Naïve EC)
    - ✓ For DRC, we encodes 20% of data
    - ✓ 20% writes are for EC data
- Large gap in 99 percentile response time between replication and EC
  - ✓ Replication: sub milli-second
  - ✓ EC methods: over 10 milli-seconds



Configuration:
HW: 3 servers (Intel (R) Xeon E 5620 2.40 GHz, 4 core x 2, 12GB, SATA SSD x 2, 10 GbE NIC), Benchmark-tool: fio random access

The latency gap between replication and conventional EC based-approaches needs to be complimented

Motivation and Approaches

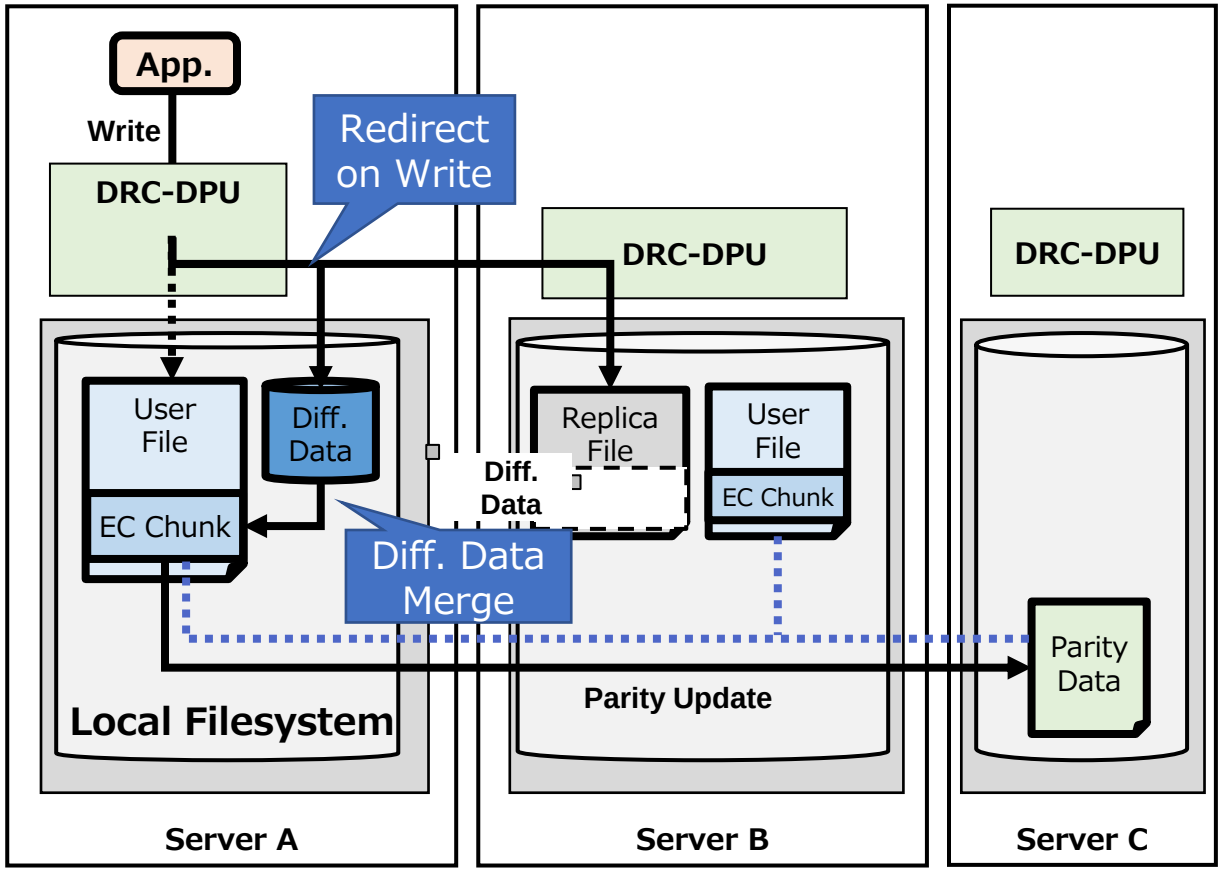
- Motivation

  Improve write response time degradation of EC data in DRC and achieve stable response time in server-based storage

- Approaches
  - Redirecting EC writes to replicated differential data
  - Consolidating differential data to the same EC data while merging them in the background

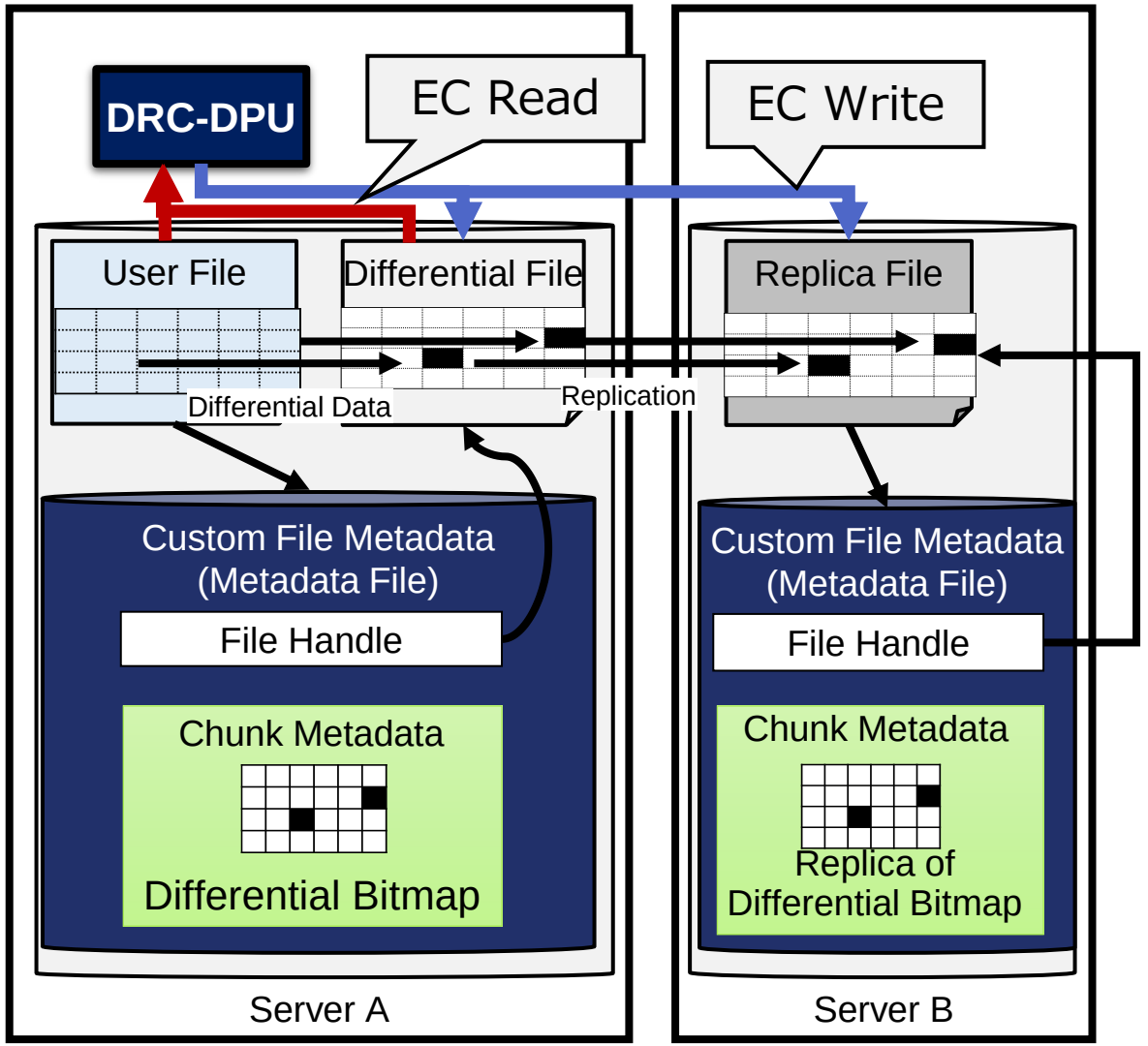
Dynamic Redundancy Control with Delayed Parity Update (DRC-DPU)

- Redirect on Write
  - ➢Replicate and store write data for EC data to differential data area
  - ➢Improve response time by responding before completion of asynchronous parity updates
- Differential Data Merge
  - ➢Asynchronously update parity data of EC data
  - ➢Reduce EC overhead by consolidating parity updates to the same EC data



Redirect on Write (1/1)
Data Processing

- Uses differential data to process read and write for EC data
  - ➢ EC Write:
    - ✓ Redirects EC writes to replicated **differential files** for smaller response time
  - ➢ EC Read
    - ✓ Reads from original EC chunk and differential file and merges them
- Manages differential pages with differential bitmap
  - ➢ Uses **custom file metadata** which is extensive File Metadata using **metadata file** [4]



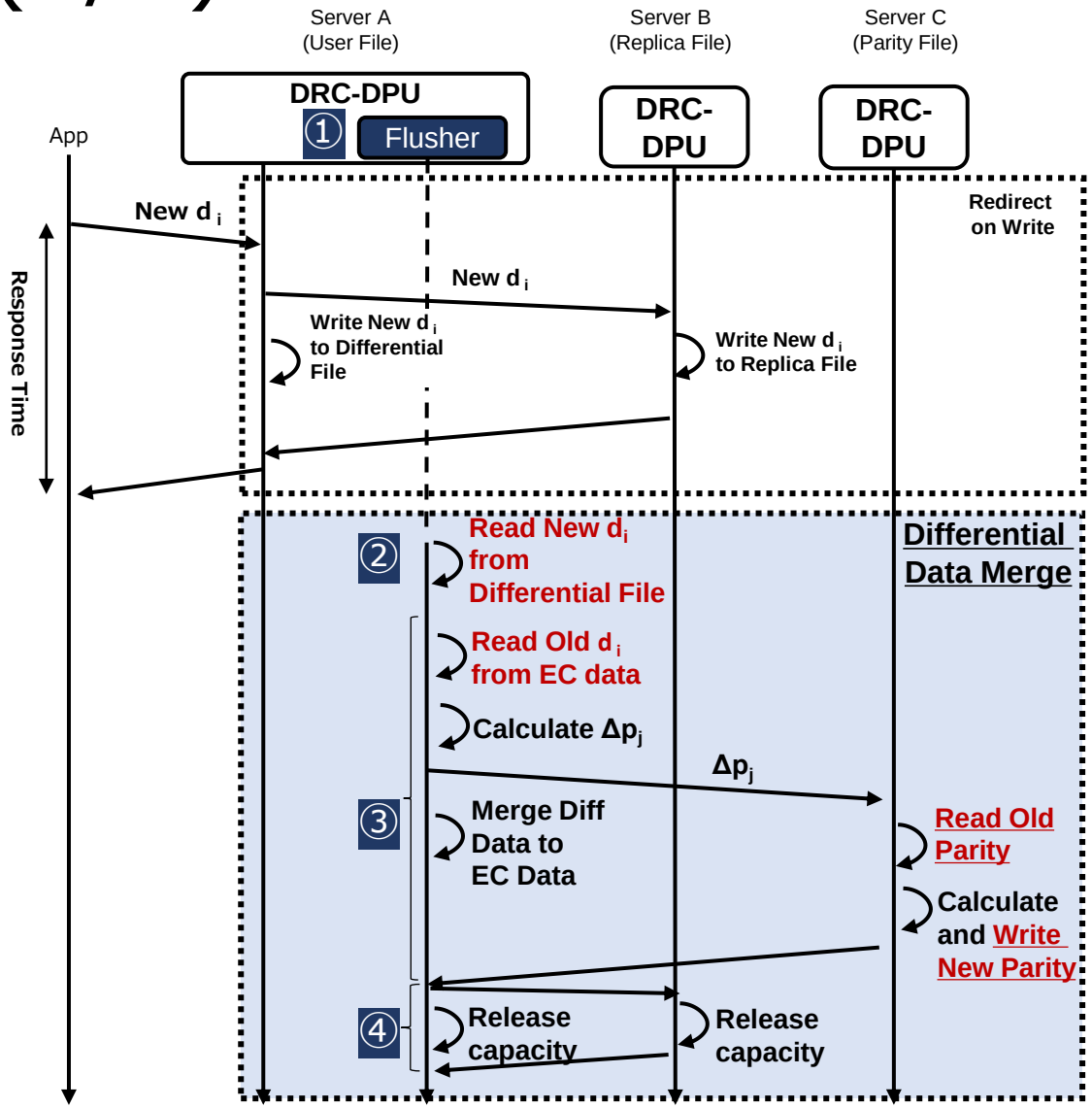
Redirect on Write (2/2):
Read-Modify-Write on Differential Pages

- DRC-DPU uses Read-modify-write processing for small writes less than differential page size

- Read old EC data from disks and merge the old data and new data
  - Smaller page size decreases the frequency of RMW while increasing metadata capacity
  - We adopt 4 KB for page size based on evaluations (Discussed later)

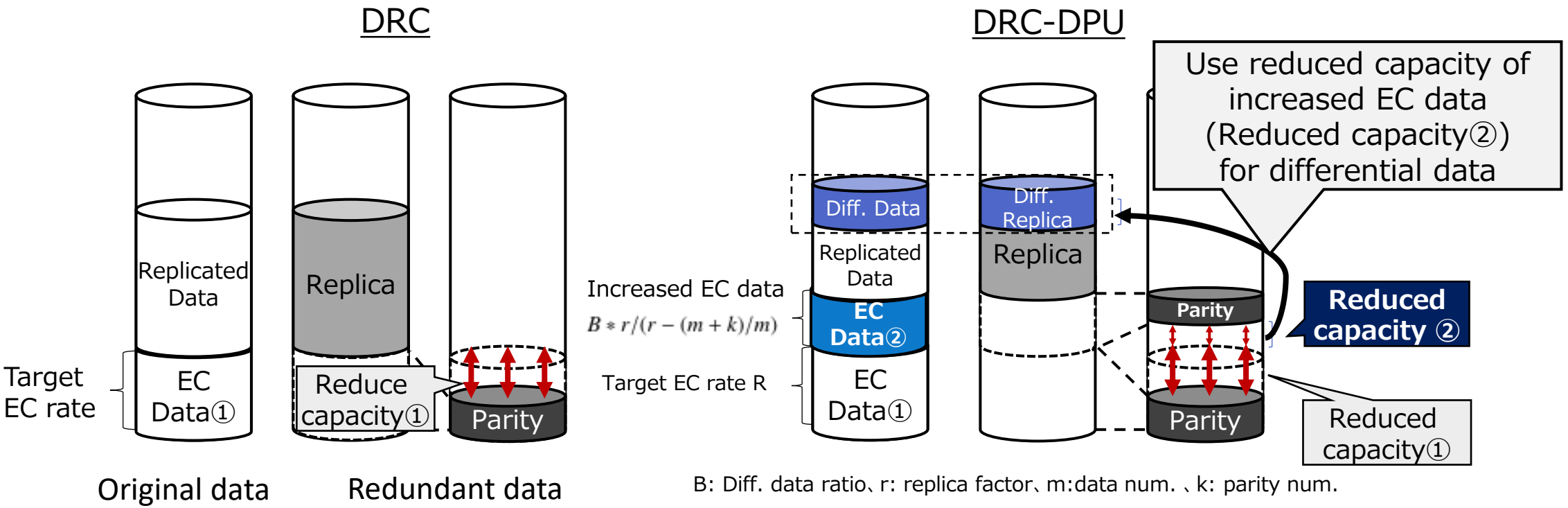
Differential Data Merge (1/2)

- DRC-DPU performs Differential data merge to merge the differential data into EC data
  - ① Periodically starts an internal thread called Flusher in the background
  - ② Flusher reads the differential pages in the differential file
  - ③ DRC-DPU performs parity update in the same way as a normal EC write
  - ④ Flusher then releases the capacity of the differential data
- Asynchronizing parity updates enables small write response time



Differential Data Merge (2/2)

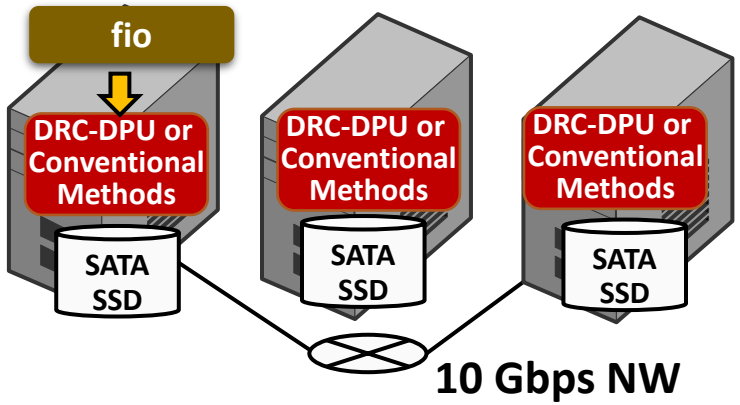
- DRC-DPU reserves a certain amount of capacity for storing differential data.
  - ➢ Uses the LRU list to delay merging recently updated differential pages
  - ➢ Reduces the number of parity updates when the same differential page is updated multiple times.
- To maintain capacity efficiency, DRC-DPU increases the amount of EC data and reserves capacity for differential pages



B: Diff. data ratio、r: replica factor、m:data num. 、k: parity num.

Experiments: Synthesized Workload

- Workloads
  - fio 8KB Random write response time
  - fio 2KB Random write response time with RMW
- Targets
  - Replication
  - Naïve EC
  - Parity Logging
  - SPW (Success/Failure)
  - DRC (Replication 80%、 EC(2D1P) 20%)
  - DRC-DPU (Replication 80%、 EC (2D1P) 20%)

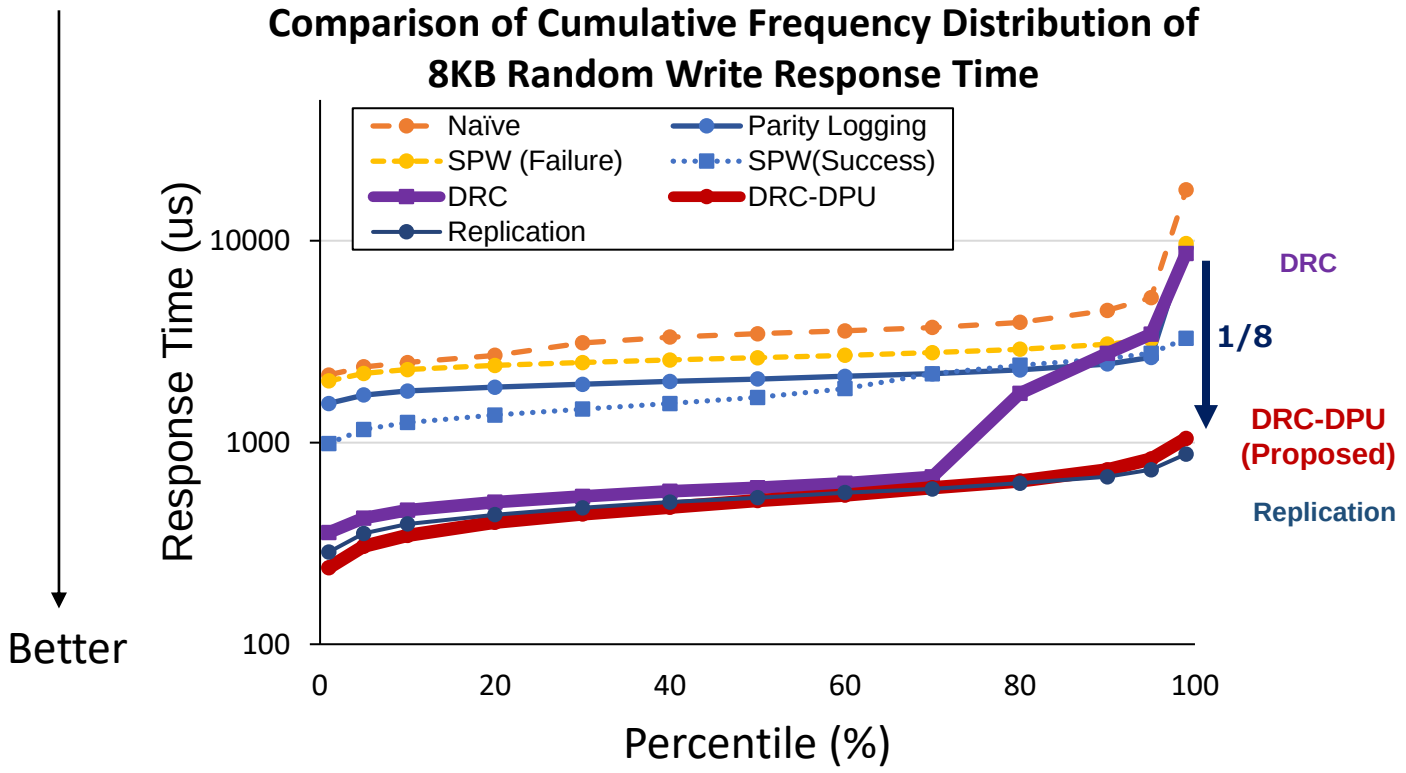


Items	Description
Server	HP Proliant DL 160 G6 x 3
CPU / Server	Intel (R) Xeon E 5620 2.40 GHz, 4 core x 2
RAM /Server	12GB
NIC / Servre	HP NC550SFP 10 GbE Server Adapter
Disks / Server	SATA SSD x 2

Experiments: Synthesized Workload

Random Write Performance

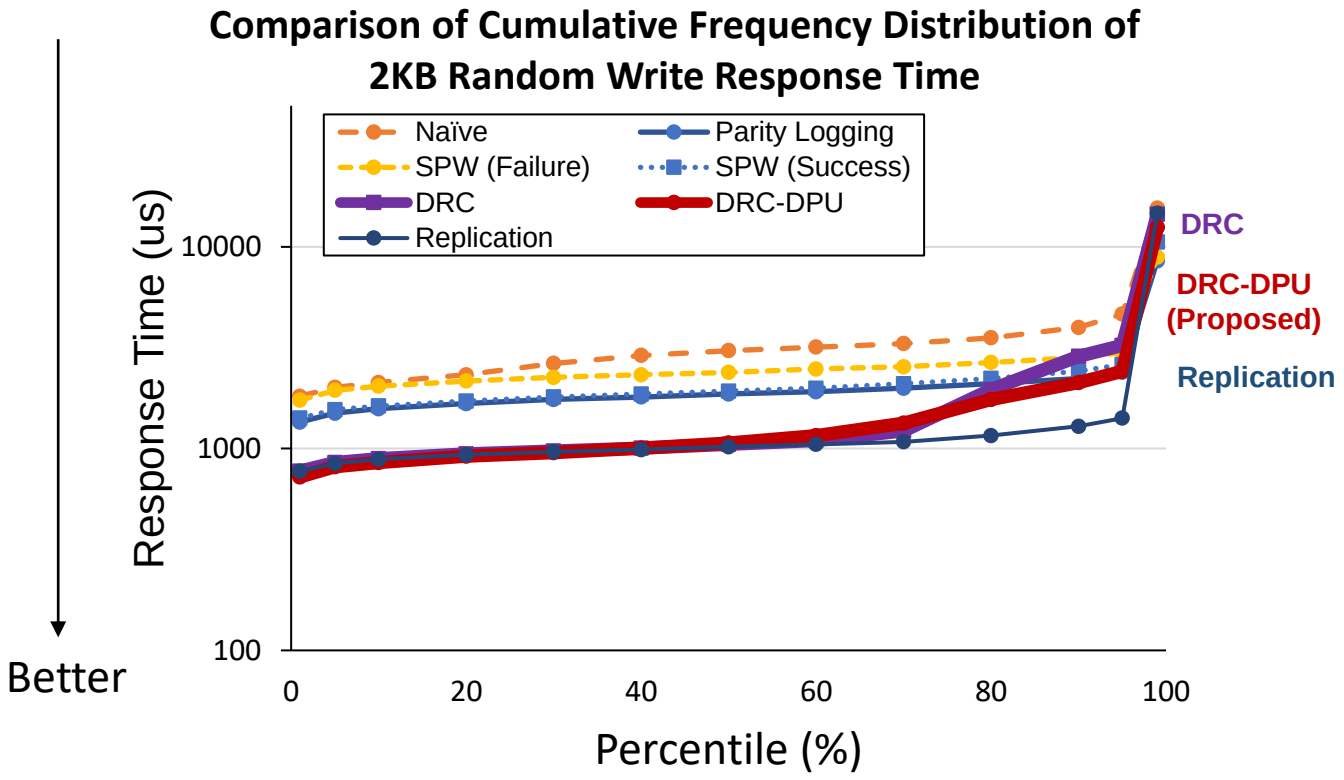
- DRC-DPU achieves a similar response time with replication
  - 1/8 of 99 percentile response time of DRC
  - Almost zero response time degradation in EC data write



Experiments: Synthesized Workload

Random Write Performance with RMW

- Although RMW causes write response time degradation in EC data write, DRC-DPU achieves better performance than DRC



Experiments: Real-world Workloads

- Evaluation items
  - Parity update reduction against amount of differential pages
  - Metadata capacity overhead against differential page sizes
- Methods
  - Examined the number of parity updates and metadata overhead with different differential page amounts/sizes by using Microsoft Research (MSR) traces [5]
  - Used the first six days in the MSR traces to encode low-access frequency chunks into EC and simulated the remaining one day

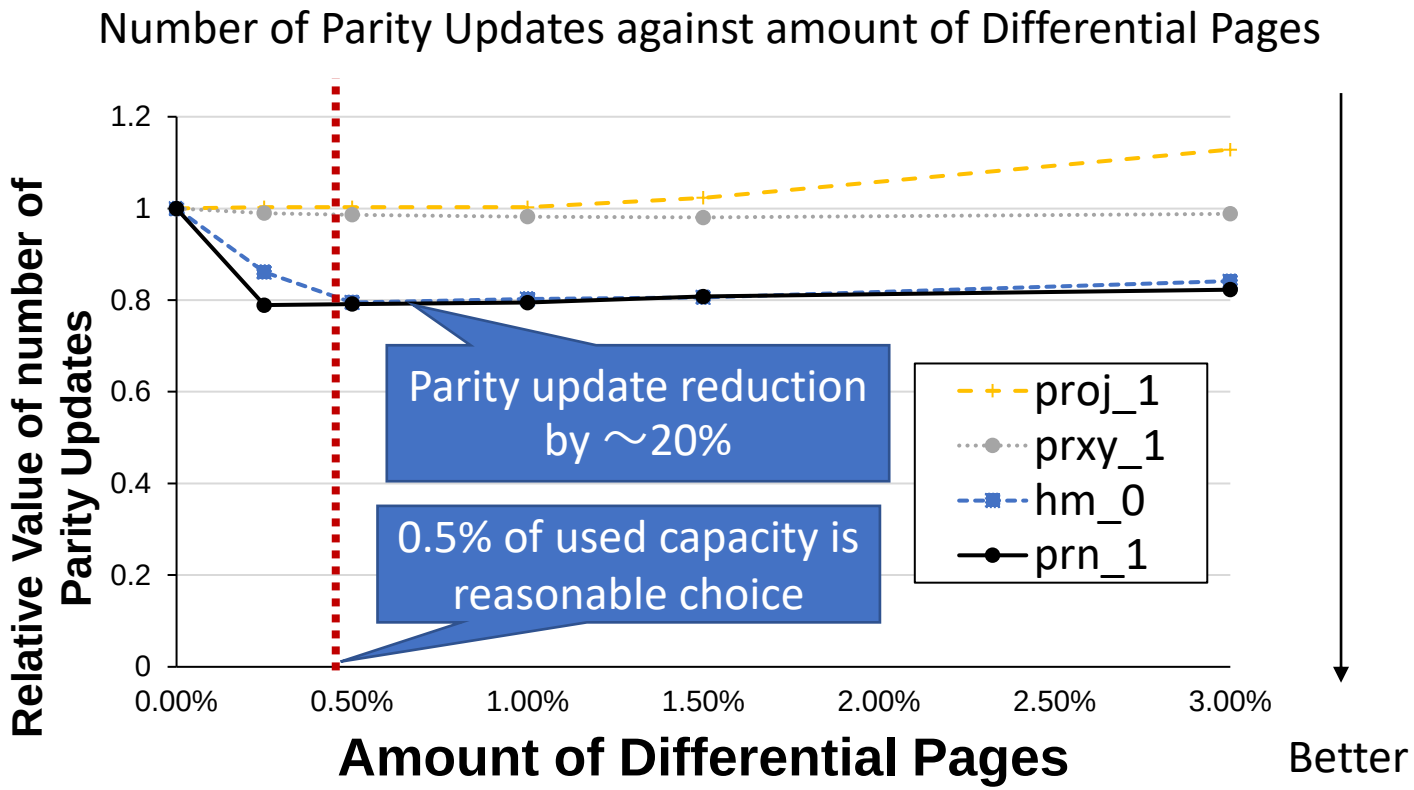
Name	Description	Avg. Write Size	Avg. Updates/ Chunk	4KB Alignments
hm_0	HW monitoring server	7.79 KB	88	Yes
prn_1	Printer server	14.8 KB	14	No
proj__1	Project file server	13.7 KB	772	Yes
prxy_1	Firewall/Proxy server	9.44 KB	12	Yes

[5] Dushyanth Narayanan,et al. . Write offloading: Practical power management for enterprise storage. ACM Transactions on Storage (TOS) 4, 3 (2008), 1–23.

Experiments: Real-world workloads

Parity Update Reduction

- Reduced the number of parity updates by up to 20% in 2 out of 4 traces
  - ➢ Differential merge achieves considerable parity update reduction for some workloads



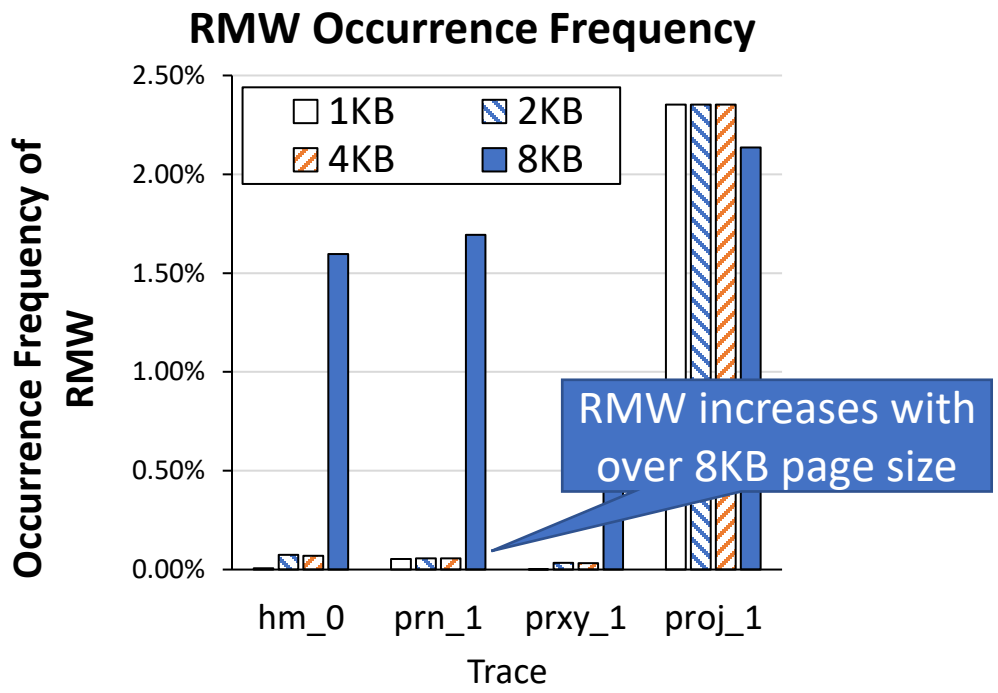
Experiments: Real-world workloads

Metadata Capacity Consumption

- The metadata consumption capacity is ~0.1%
- 4KB page sizes is a reasonable trade-off point between capacity and performance
  - ➢ Over 8KB page size dramatically increases RMWs

Metadata capacity consumption (2D1P、 Page size in parentheses)

#	Items	Metadata Capacity Ratio
1	DRC	0.030%
2	DRC-DPU (1KB)	0.097%
3	DRC-DPU (2KB)	0.061%
4	DRC-DPU (4KB)	0.042%
5	DRC-DPU (8KB)	0.033%



Conclusion

- We propose DRC-DPU to improve the write response time of EC data which is the issue of DRC

- DRC-DPU achieves similar response time with replication
  - 1/8 of 99 percentile response time of DRC

- DRC-DPU also achieves less parity update overhead and less metadata consumption
  - Up to 20% parity update reduction
  - Metadata overhead is less than 0.1%

Thank you !