2020 OFA Virtual Workshop

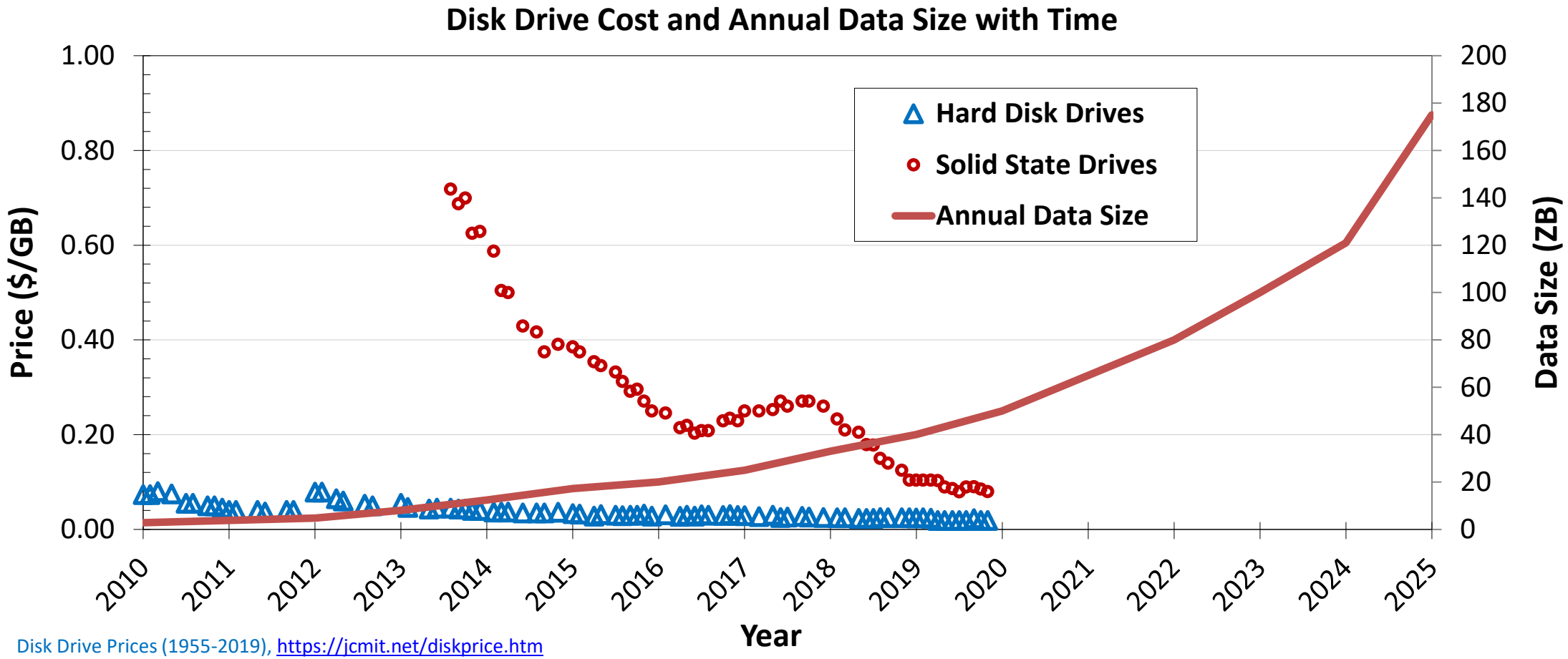
TRIEC: AN EFFICIENT ERASURE CODING NIC OFFLOAD PARADIGM BASED ON TRIPARTITE GRAPH MODEL

Xiaoyi Lu and Haiyang Shi

The Ohio State University

{lu.932, shi.876}@osu.edu

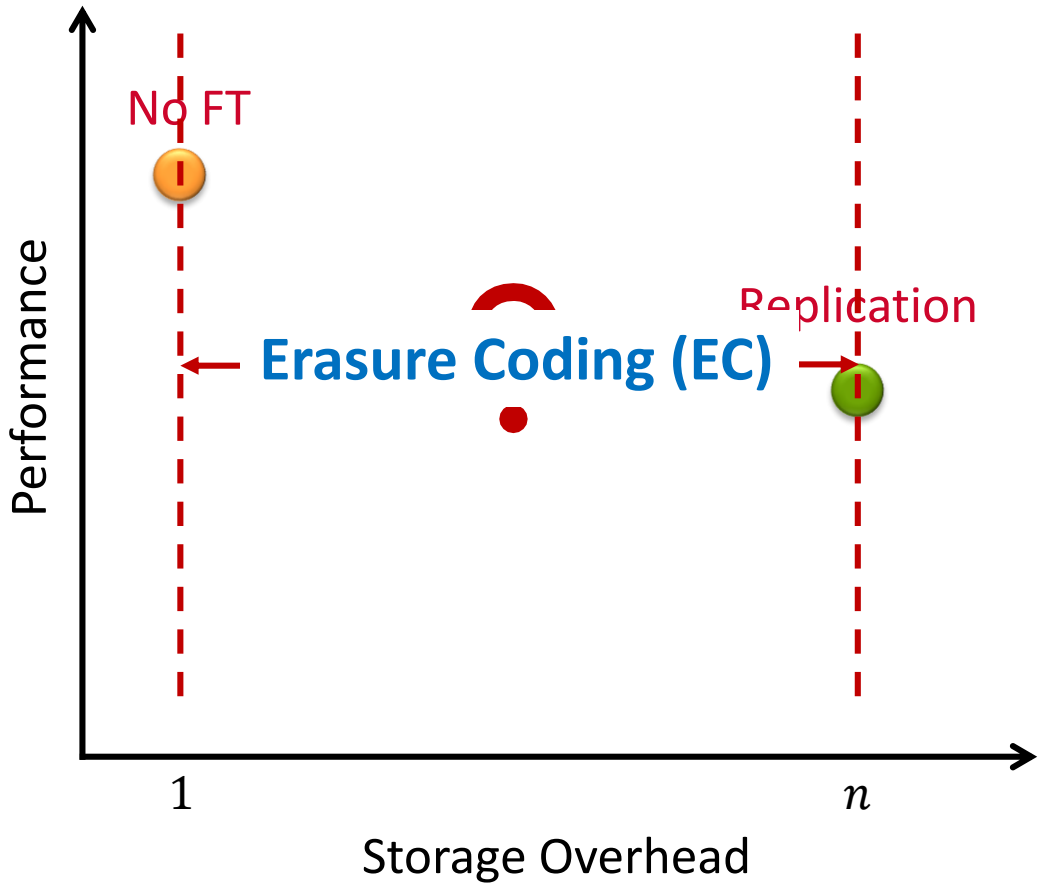
CHALLENGES OF DATA EXPLOSION



Disk Drive Prices (1955-2019), https://jcmit.net/diskprice.htm
Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018

RETHINKING OF DISTRIBUTED STORAGE SYSTEMS

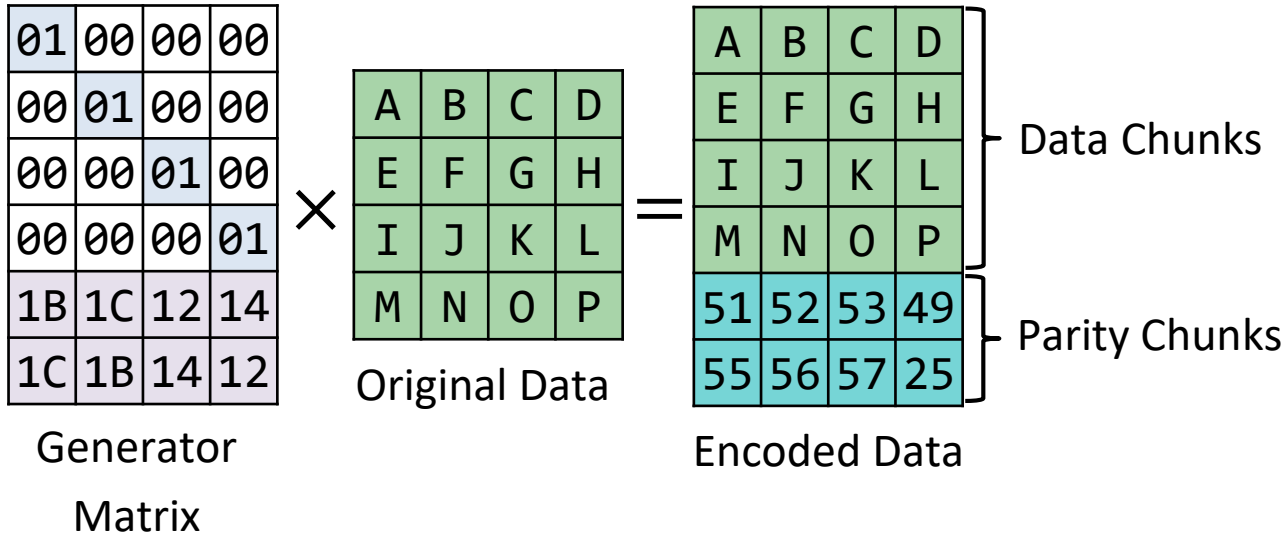
- **No Fault Tolerance**
  - ❖ High performance
  - ❖ Not reliable
- **N-way Replication**
  - Tolerate up to $n - 1$ node failures
  - Performance degradation
  - Large storage overhead



EC BASICS - REED-SOLOMON CODE

REED-SOLOMON (RS) CODE

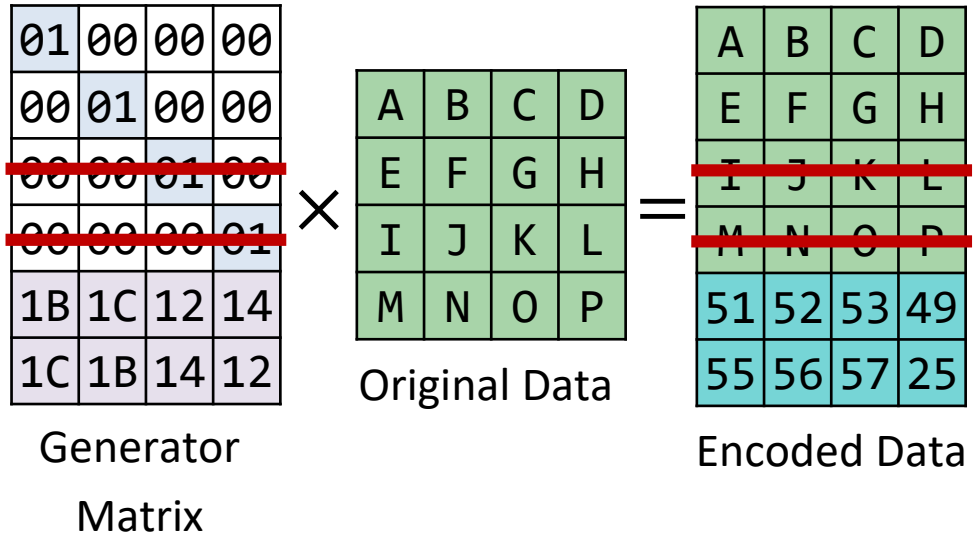
- $RS(k, m)$
  - Widely used (RAID, Microsoft Azure, HDFS 3.x)
  - Encodes on k data chunks to generate m parity chunks
  - Decodes on any k data/parity chunks to recover the original data
- *Storage Overhead of Common Codes*
  - RS(4,2) => 1.5x overhead
  - RS(6,3) => 1.5x overhead



Reed-Solomon EC for k=4 and m=2

REED-SOLOMON (RS) CODE

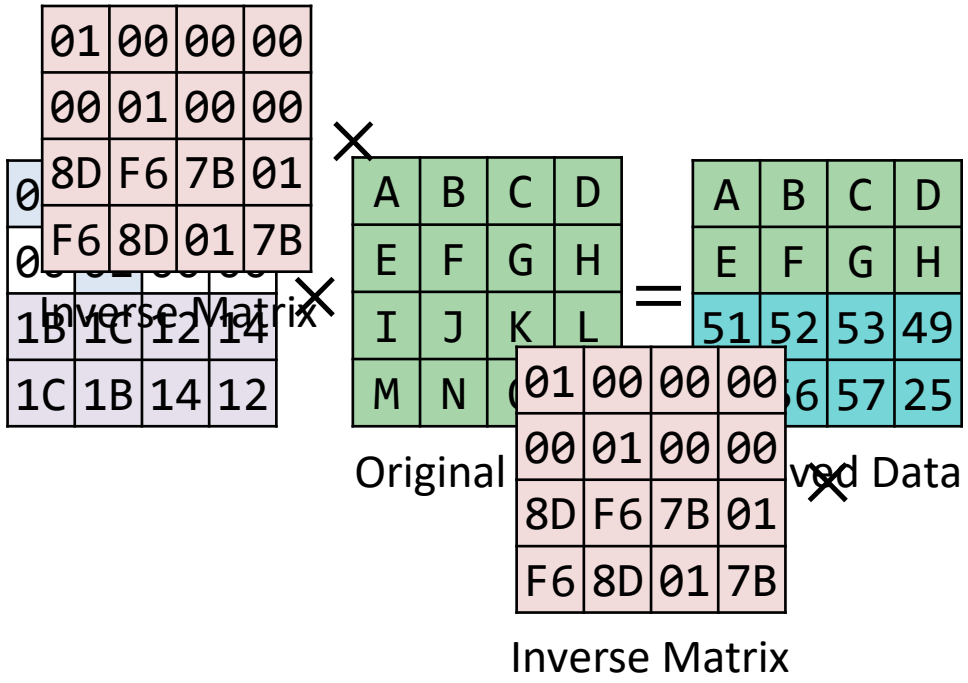
- $RS(k, m)$
  - Widely used (RAID, Microsoft Azure, HDFS 3.x)
  - Encodes on k data chunks to generate m parity chunks
  - Decodes on any k data/parity chunks to recover the original data
- *Storage Overhead of Common Codes*
  - RS(4,2) => 1.5x overhead vs. 3x in 3-way replication
  - RS(6,3) => 1.5x overhead vs. 4x in 4-way replication



Reed-Solomon EC for k=4 and m=2

REED-SOLOMON (RS) CODE

- $RS(k, m)$
  - Widely used (RAID, Microsoft Azure, HDFS 3.x)
  - Encodes on k data chunks to generate m parity chunks
  - Decodes on any k data/parity chunks to recover the original data
- *Storage Overhead of Common Codes*
  - RS(4,2) => 1.5x overhead vs. 3x in 3-way replication
  - RS(6,3) => 1.5x overhead vs. 4x in 4-way replication



Reed-Solomon EC for k=4 and m=2

EC: THE ALTERNATIVE RESILIENCE TECHNIQUE

Are there any good approaches to improve EC performance?

Write(Encoding)

Read with Erasures (Decoding)

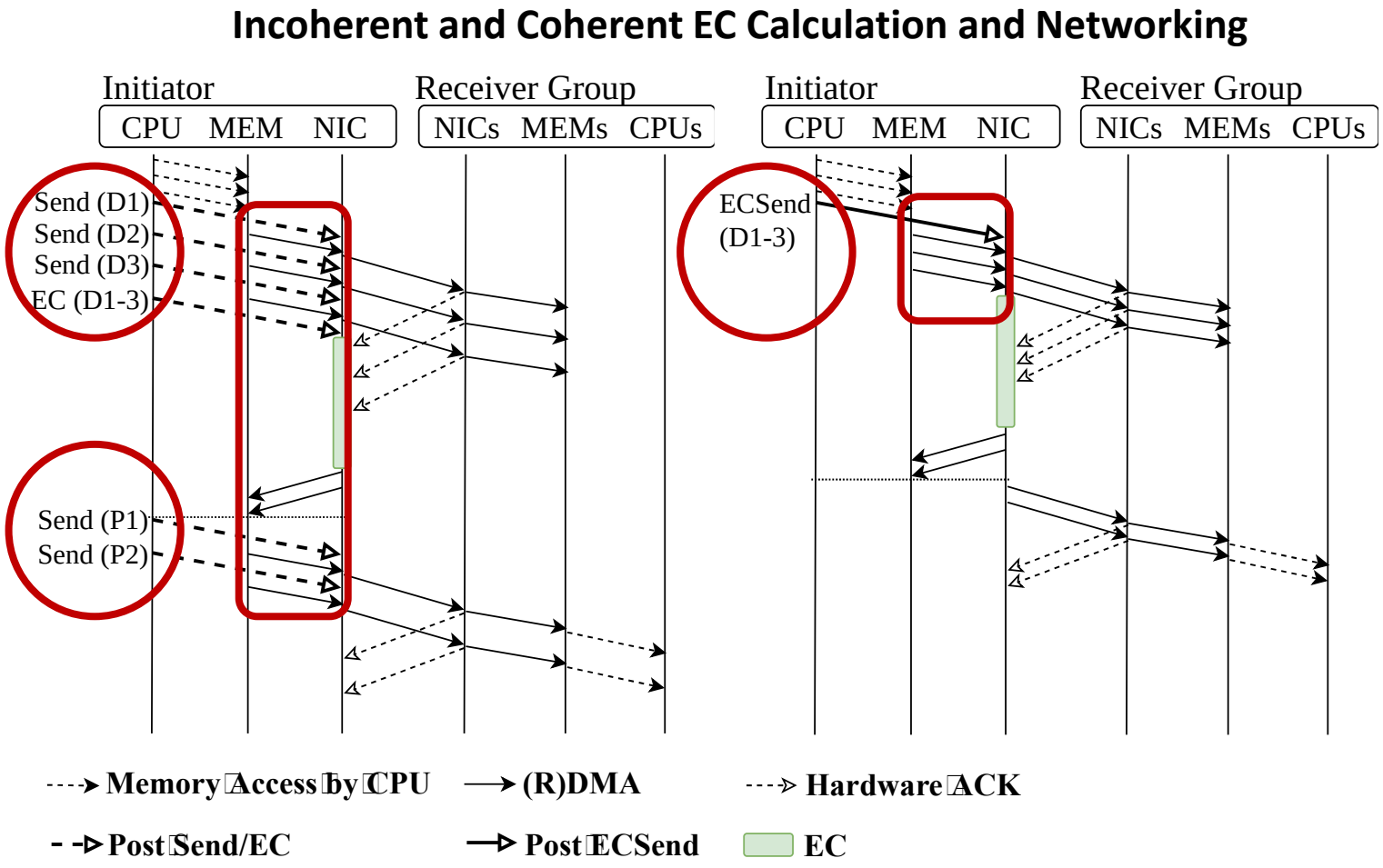
- Exploiting EC will introduce both computation overhead and communication overhead

EC OFFLOAD ON SMARTNICS

EC OFFLOAD ON SMARTNICS

- **Promising Capability on Modern SmartNICs**
  - Mellanox InfiniBand ConnectX-4 and later
- **About 100 supercomputers in latest TOP500 are equipped with Mellanox InfiniBand NICs**

OVERVIEW EC CAPABILITIES ON MODERN SMARTNICS



- **Coherent EC Calculation and Networking**
  - Less CPU involvement
  - Less DMA operations

RETHINKING TRADITIONAL EC PARADIGM ON SMARTNICS

- Encoding Procedure
  - ❖Collect k data chunks
  - ❖Post encode-and-send

- One set of nodes (e.g., clients) perform EC encoding calculations
- The other set of nodes (e.g., data nodes) receive encoded chunks

- We name it as Bipartite Graph Based EC (BiEC) Paradigm

N_y^x Node y in layer x

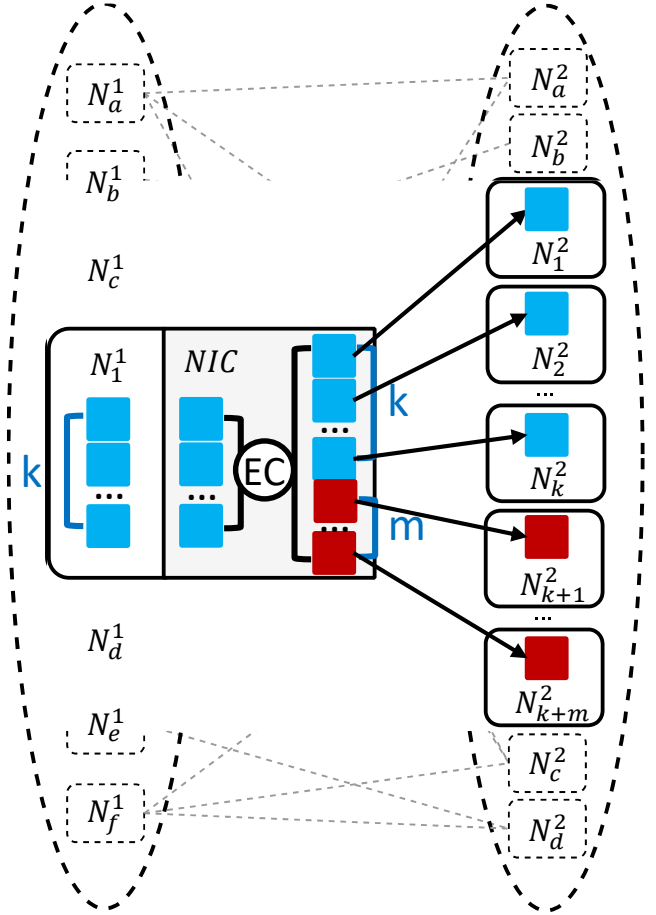
⊗ Exclusive OR

EC Erasure Coder

Data Chunk

Parity Chunk

Intermediate Chunk

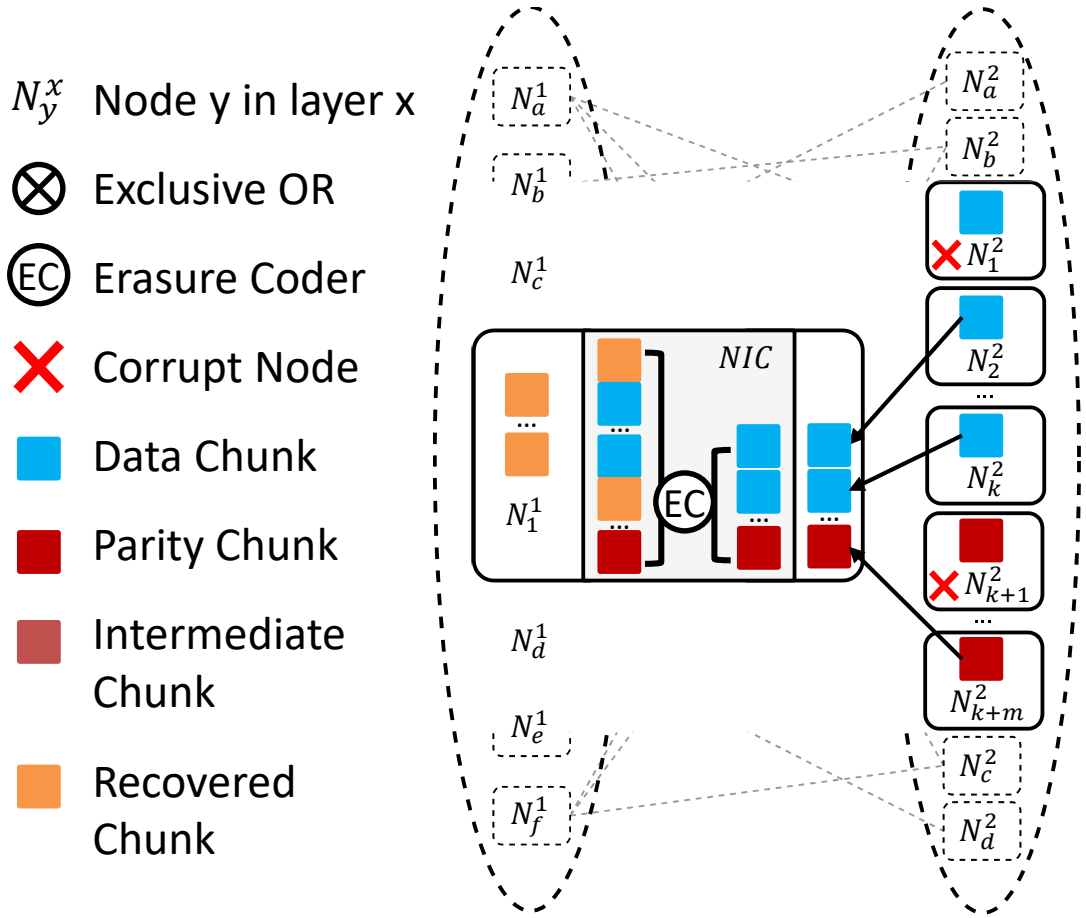


RETHINKING TRADITIONAL EC PARADIGM ON SMARTNICS

- Decoding Procedure
  - ❖ Fetch k survived chunks
  - ❖ Decode to reconstruct corrupt chunks

- One set of nodes (e.g., data nodes) send requested chunks
- The other set of nodes (e.g., clients) receive and decode to reconstruct

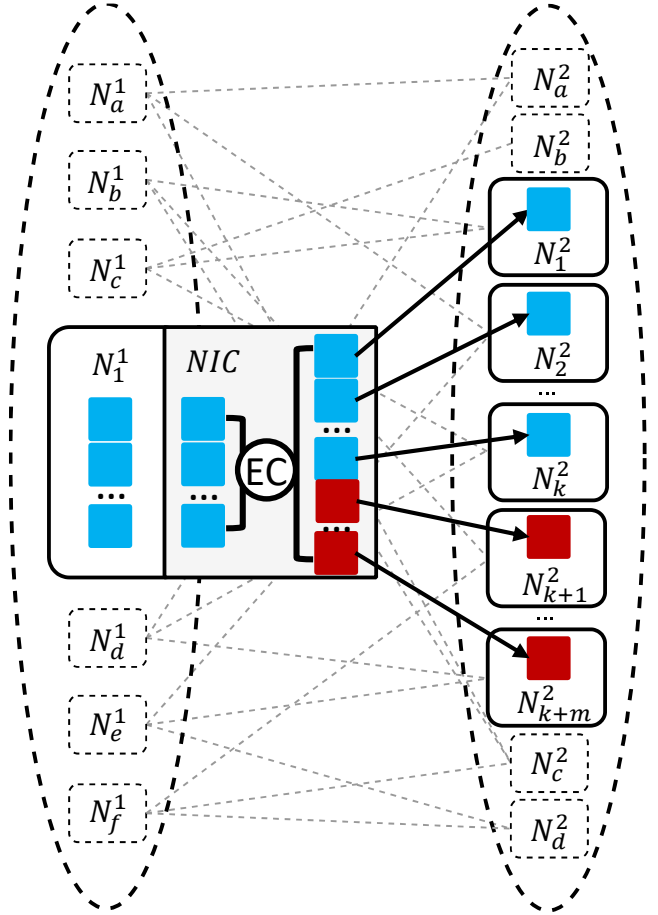
- Follows Bipartite Graph Based EC (BiEC) Paradigm



LIMITATIONS AND CHALLENGES

LIMITATIONS OF BIEC PARADIGM

- **Limitation 1: Lack of parallelism**
- **Limitation 2: Treat a NIC as a power processor, not fully exploit networked resources**



LIMITATIONS OF EXISTING APIS FOR EC ON SMARTNICS

- **Encode and Send Primitive**
  - Offload encoding operation and data transmissions simultaneously

- **Limitation 3: No receive and decode primitive support**

CHALLENGES

1. **Can we propose a better EC paradigm, which is able to exploit the networked resources in an optimized approach?**

2. **If such an EC paradigm exists, how can we achieve better overlapping?**

3. **Can such a new EC paradigm be applied uniformly to both encoding and decoding procedures?**

4. **Are there any additional challenges to co-design applications with the proposed EC paradigm?**

TRIEC IDEAS

TRIPARTITE GRAPH BASED EC PARADIGM (TRIEC)

Key Observation: An EC computation can be decomposed into sub EC calculations, which are possible to be performed on a set of nodes in parallel

- **Bipartite EC graph transforms to Tripartite EC graph**
- **We name it as Tripartite Graph Based EC Paradigm**

TRIEC FOR ENCODING

- **Encoding Procedure**
  - Client (e.g., N_1^1) sends each data chunk
  - Data node in layer-2 (e.g., N_1^2) posts encode-and-send
  - Data node in layer-3 (e.g., N_1^3) sums up k intermediate chunks to generate a parity chunk

N_y^x Node y in layer x

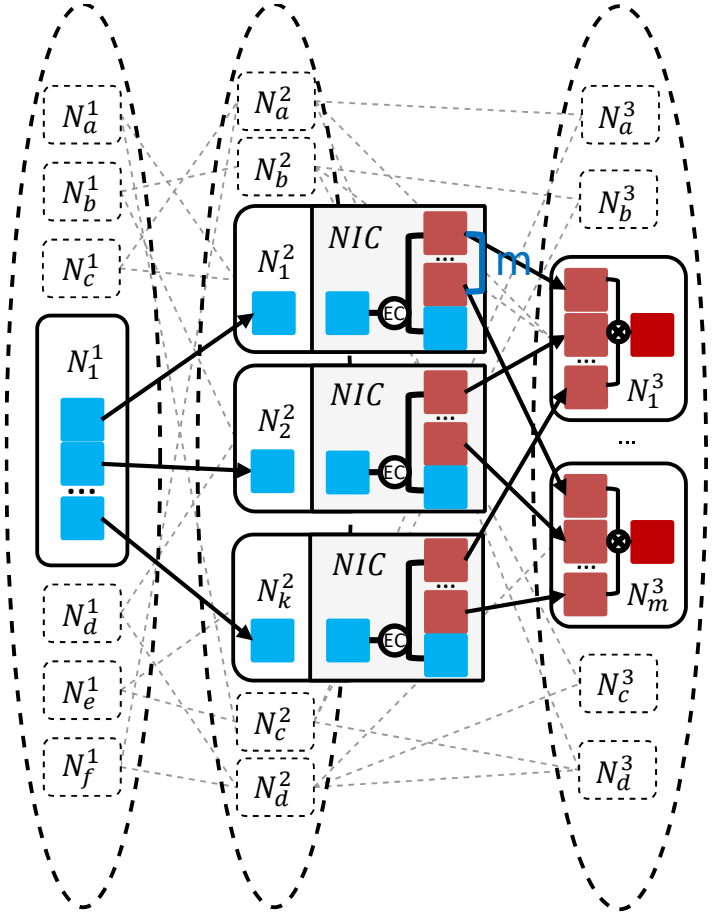
⊗ Exclusive OR

EC Erasure Coder

Data Chunk

Parity Chunk

Intermediate Chunk



TRIEC FOR DECODING

- **Decoding Procedure**
  - Data node in layer-3 (e.g., N_1^3) decodes to generate p intermediate chunks
  - Client in layer-1 (e.g., N_1^1) gets recovered chunks

N_y^x Node y in layer x

$\otimes$ Exclusive OR

TriEC can be uniformly applied to both encoding and decoding

Parity Chunk

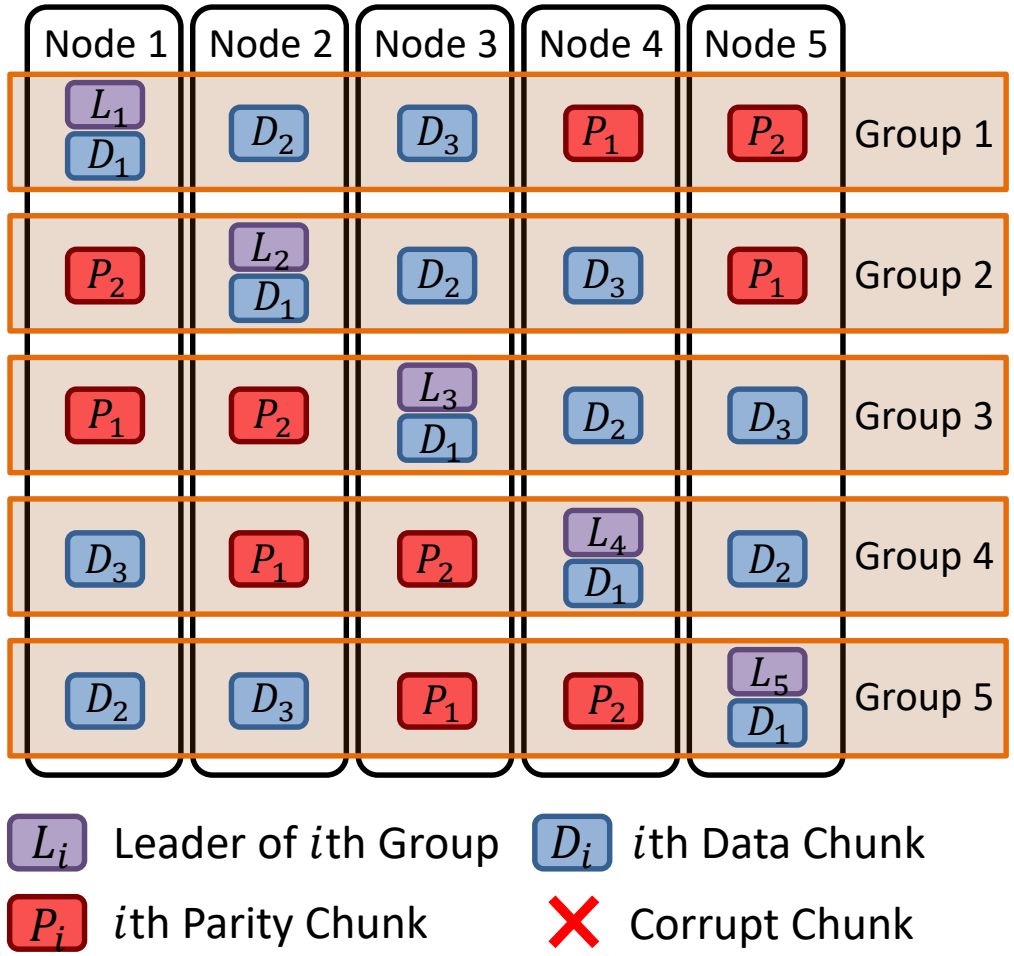
Intermediate Chunk

Recovered Chunk

TRIEC-CACHE

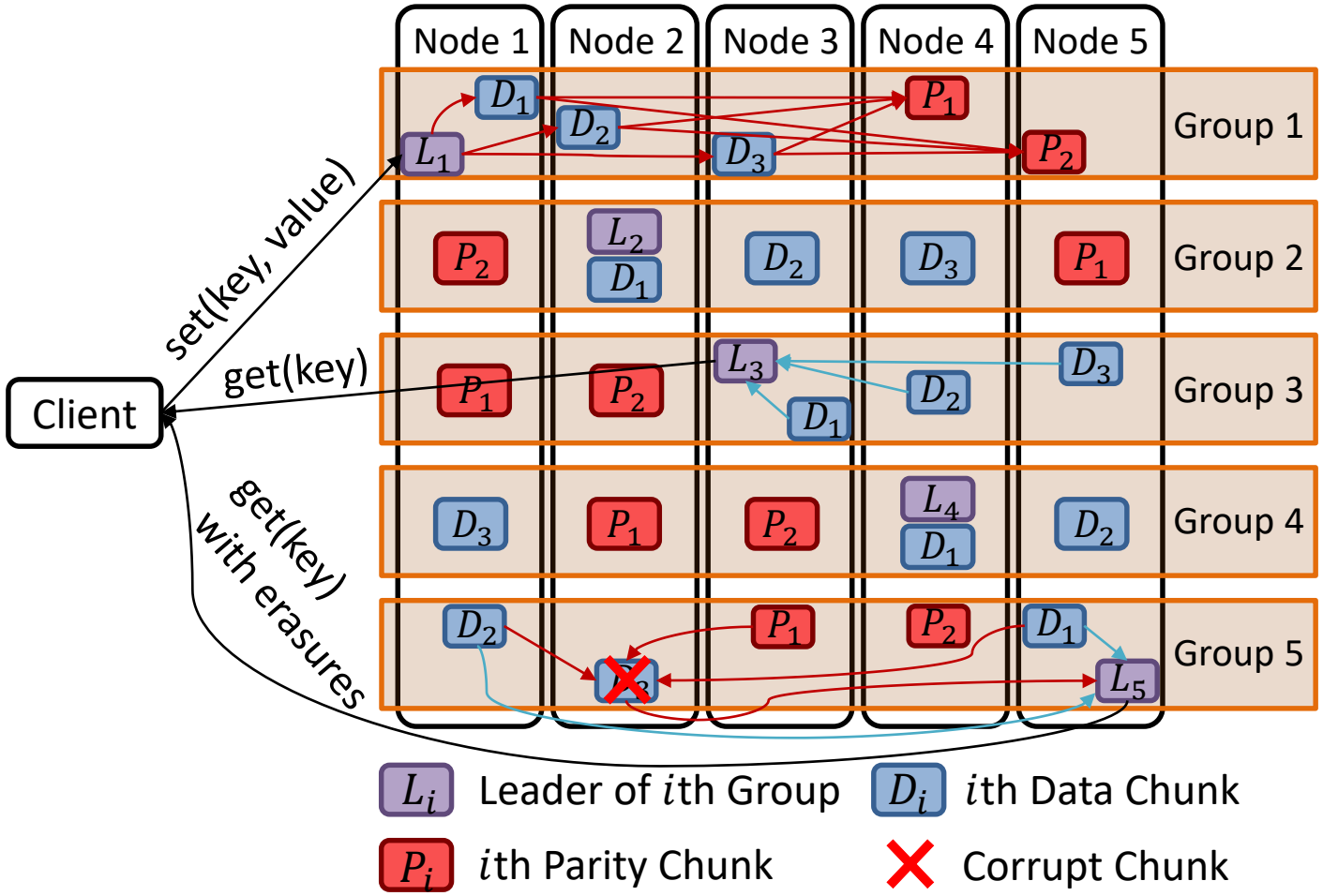
ARCHITECTURE OVERVIEW OF TRIEC-CACHE

- **Based on *memcached* (v1.5.12)**
- **Interleaved Architecture**
  - EC Group: Leader, data processes, and parity processes
  - Interleave EC groups into the cluster such that data processes and parity processes are evenly distributed
  - Balance workload and resource utilizations

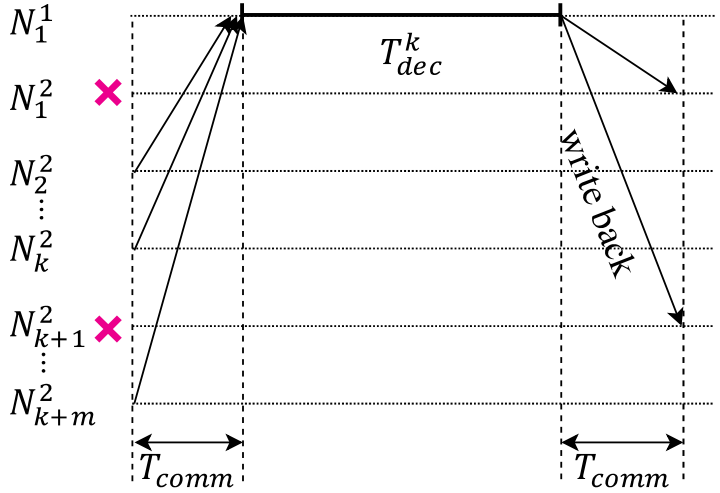


ARCHITECTURE OVERVIEW OF TRIEC-CACHE

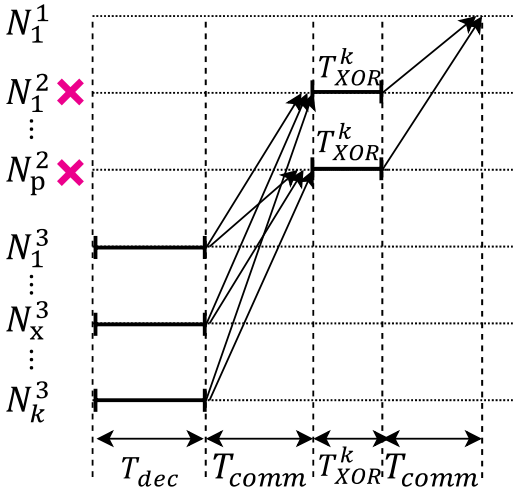
- **set(key, value)**
  - Implemented with encode-and-send primitive
- **get(key)**
  - No EC operations
- **get(key) with erasures**
  - Involve data recoveries
  - Implemented with recv-and-decode primitive



RECOVERY MECHANISM OF BIEC AND TRIEC



Out-of-Band Recovery for BiEC

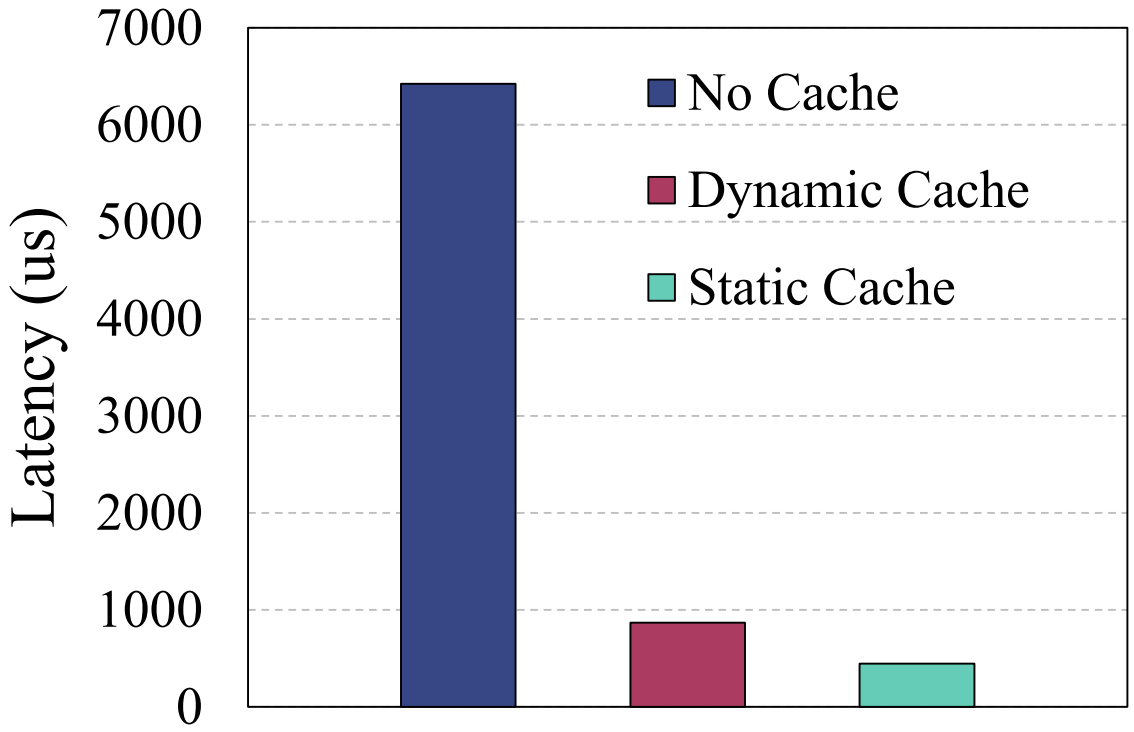


In-Band Recovery for TriEC

- Out-of-Band Recovery
  - ❖Write back
  - ❖Recover in the background
- In-Band Recovery
  - ❖No extra computations or communications

OPTIMIZATIONS WITH MLNX OFED

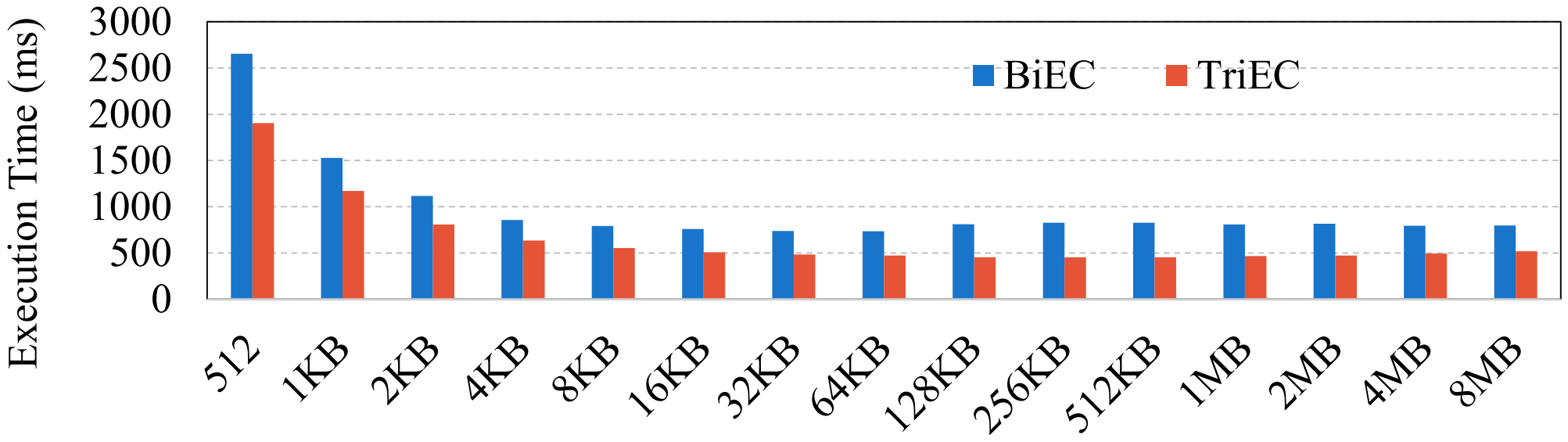
- **Avoidance of Sending Unrequested Chunks**
  - Nullify the work requests to avoid sending out unrequested chunks
- **EC Calculator Cache**
  - Initializing EC calculators is very expensive with Mellanox's EC offload APIs
  - Static EC calculator cache vs. dynamic EC calculator cache



Performance Impact of Calculator Cache (OSU RI2 Cluster)

EVALUATION

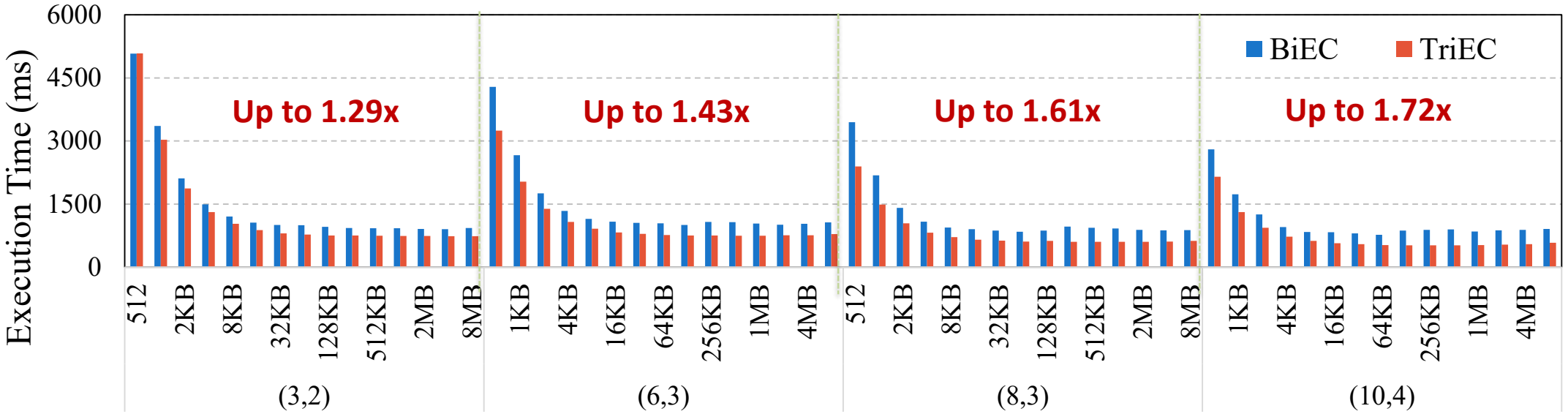
MICROBENCHMARK – TRIEC ENCODING



Encoding Performance Comparisons for RS(12,4) with Diverse Chunk Sizes (OSU RI2 Cluster)

- **TriEC reduces the encoding time with RS(12,4) by 23.5% - 45.1%**

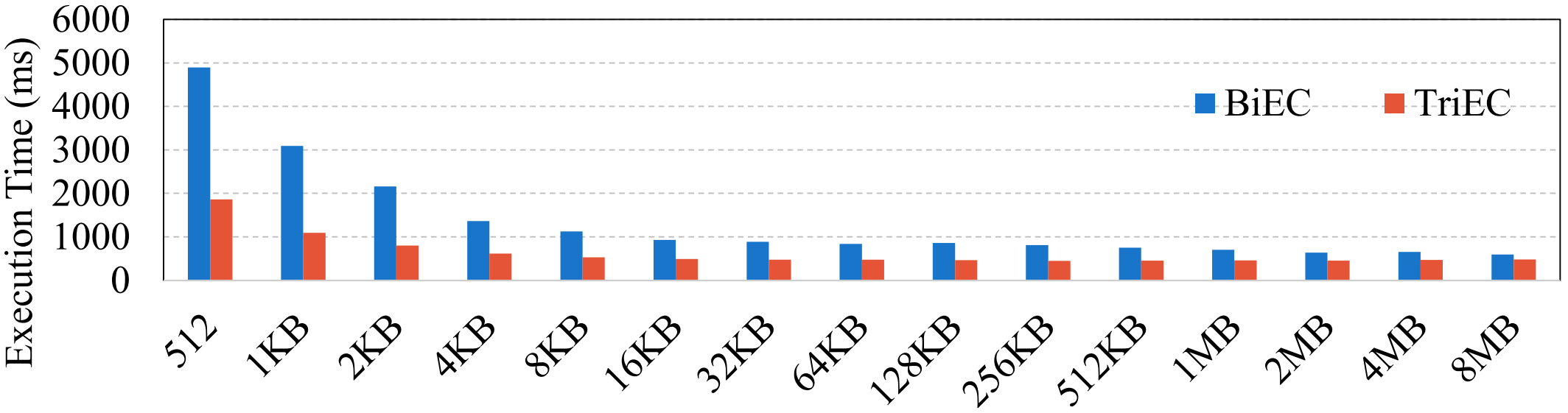
MICROBENCHMARK – TRIEC ENCODING



Encoding Performance Comparisons with Varied Configurations and Chunk Sizes (OSU RI2 Cluster)

- **Evaluate TriEC with all widely-used EC configurations**

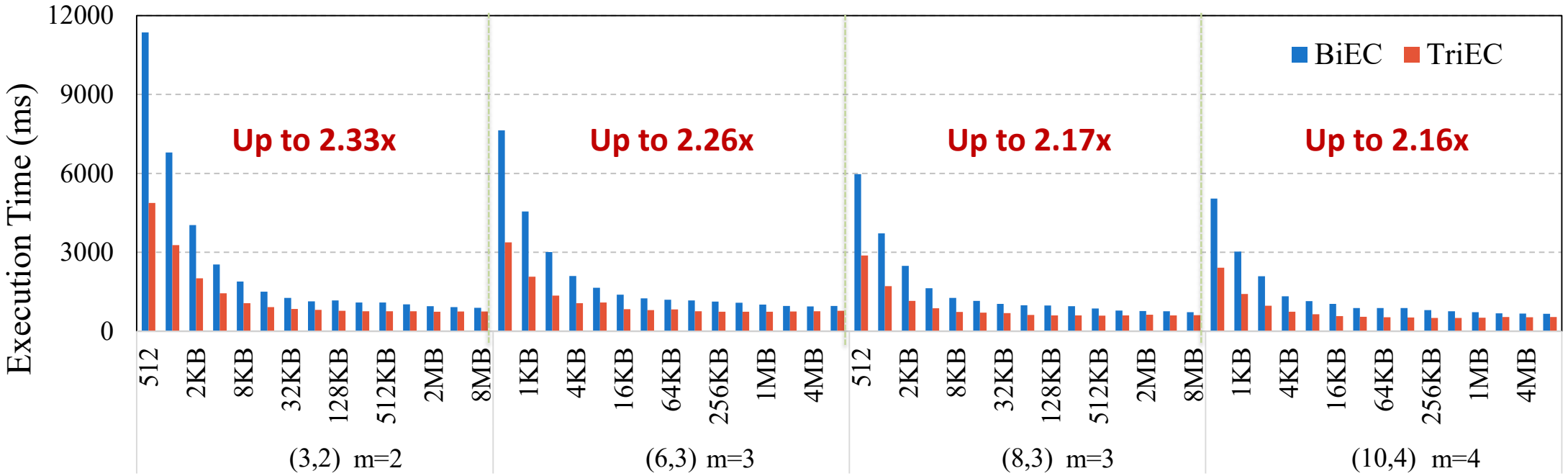
MICROBENCHMARK – TRIEC DECODING



Performance Comparisons for Recovering Four Data Chunks for RS(12,4) with Diverse Chunk Sizes (OSU RI2 Cluster)

- **TriEC reduces the decoding time with RS(12,4) by 18.6% - 64.6%**

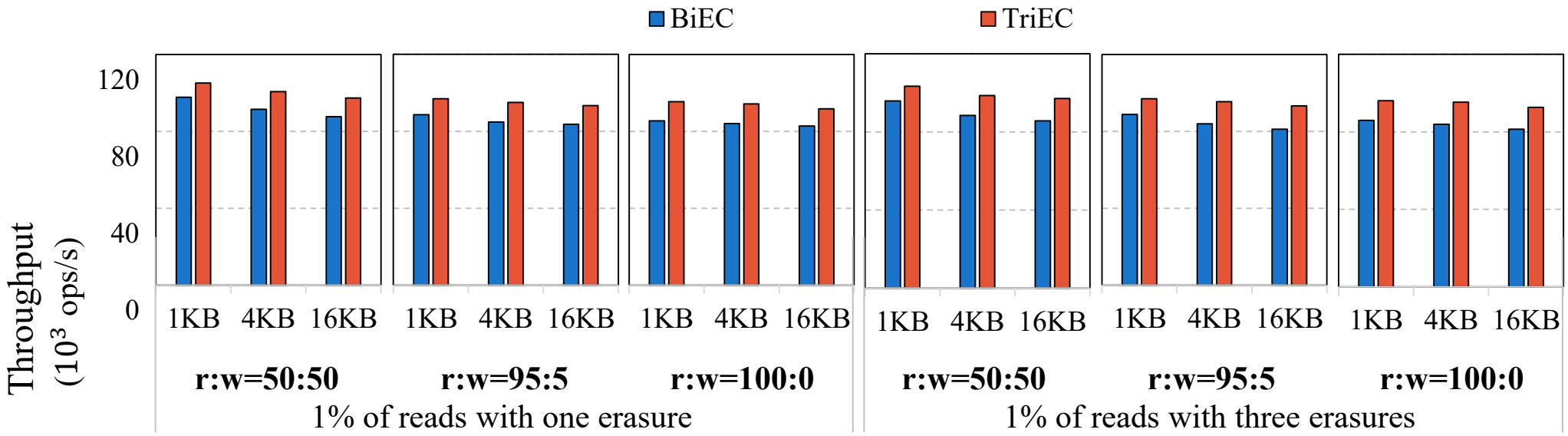
MICROBENCHMARK – TRIEC DECODING



Performance Comparisons for Recovering m Data Chunks with Varied Configurations and Chunk Sizes (OSU RI2 Cluster)

- **Evaluate TriEC with all widely-used EC configurations**

THROUGHPUT OF TRIEC-CACHE



Throughput Comparisons for RS(6, 3) (OSC Pitzer Cluster)

- **TriEC speeds up the overall throughput performance by up to 13.3% for equal-shares, 14.8% for read-mostly, and 13.9% for read-only workloads.**

CONCLUSION

CONCLUSION AND FUTURE WORK

- A new EC NIC offload paradigm based on tripartite graph model (i.e., TriEC)
- A new receive-and-decode primitive on top of Mellanox EC NIC offload APIs
- Co-design a TriEC-based key value store based on Memcached
- TriEC reduces the average write latency by up to 23.2% and the recovery time by up to 37.8% even with only 1% failure occurrences
- In the future, we plan to apply TriEC to other storage systems and hardware platforms

Haiyang Shi and Xiaoyi Lu. TriEC: Tripartite Graph Based Erasure Coding NIC Offload. In Proceedings of the 32nd International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2019. (Best Student Paper Finalist)

ACKNOWLEDGEMENT

- We would like to thank Ohio Supercomputer Center (OSC) for providing the cluster access.
- We would also like to thank Mellanox for donating SmartNICs.
- We also thank the anonymous reviewers for their precious feedback to this work.
- This work is supported in part by National Science Foundation grant #CCF-1822987.

2020 OFA Virtual Workshop

THANK YOU

Xiaoyi Lu, Research Assistant Professor

The Ohio State University