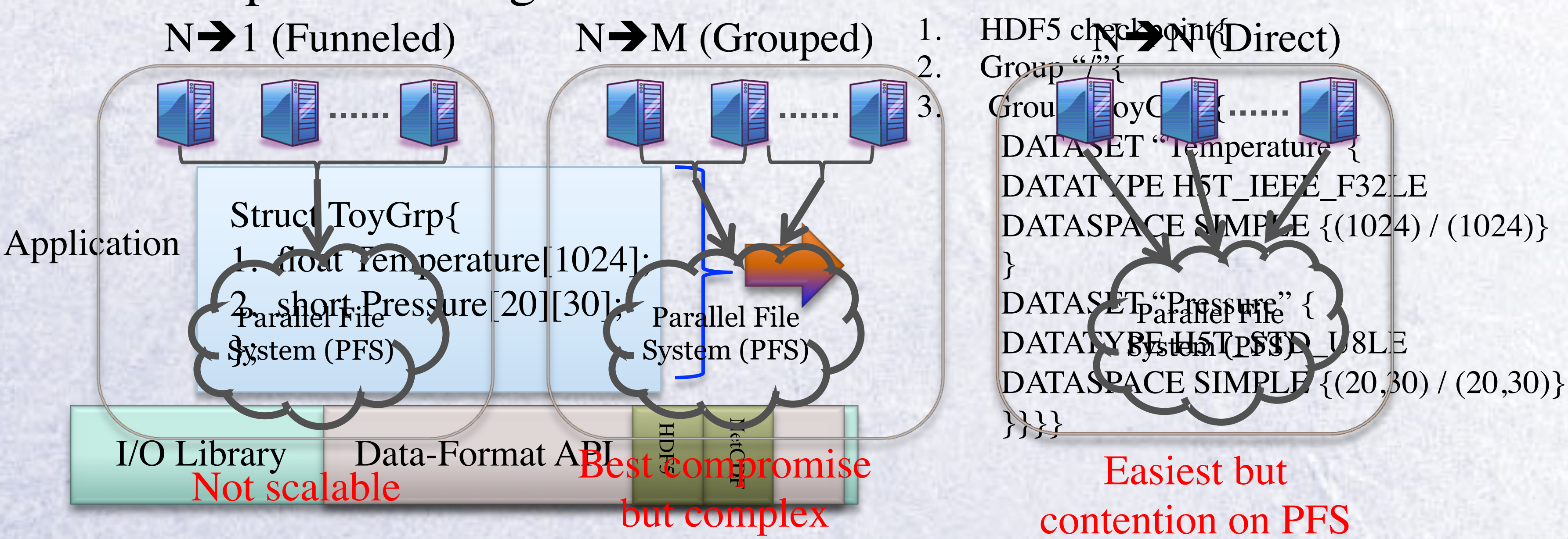
“mcrEngine”
a Scalable Checkpointing System using Data-Aware Aggregation and Compression

Tanzima Z. Islam, Saurabh Bagchi, Rudolf Eigenmann
School of ECE, Purdue University
West Lafayette, IN

Kathryn Mohror, Adam Moody, Bronis R. de Supinski
Lawrence Livermore National Laboratory
Livermore, CA

Background

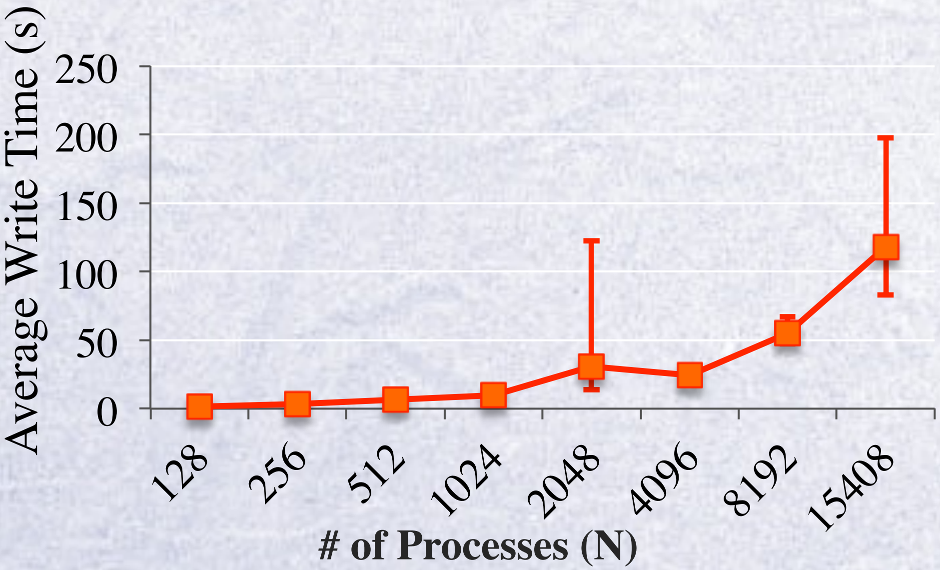
- Checkpoint-restart widely used
  - Projected MTBF at exascale 3-26 minutes
- MPI applications
  - Take globally coordinated checkpoints asynchronously
- Application-level checkpoint
- High-level data format for portability
  - HDF5, Adios, netCDF etc.
- Checkpoint writing

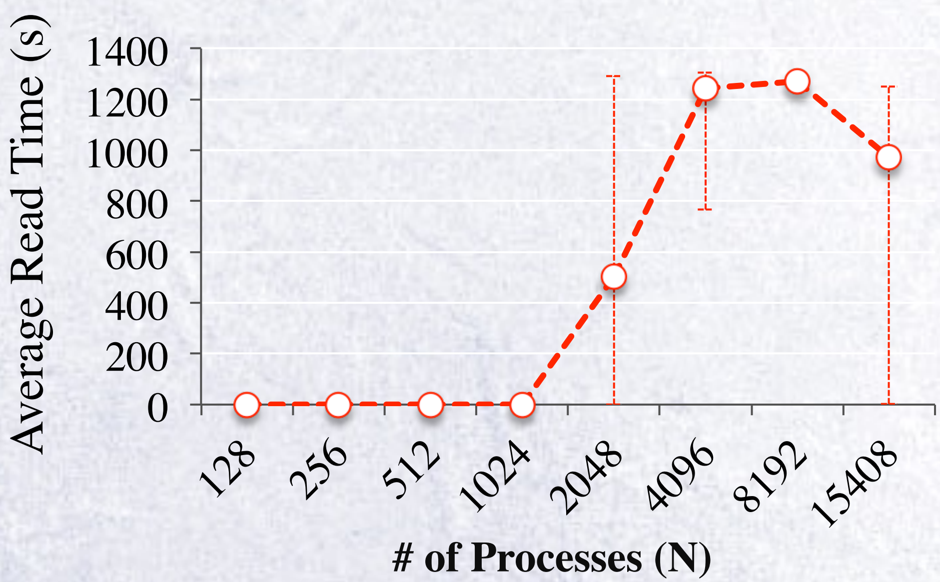


Impact of Load on PFS at Large Scale

- IOR
  - Direct (N→N): 78MB per process
- Observations:

(–) Large average write time ⇒ less frequent checkpointing
(–) Large average read time ⇒ poor application performance





What is the Problem?

- Today's checkpoint-restart systems will not scale
  - Increasing number of concurrent transfers
  - Increasing volume of checkpoint data

Our Contributions

- Data-aware aggregation
  - Reduces the number of concurrent transfers
  - Improves compressibility of checkpoints by using semantic information
- Data-aware compression
  - Reduces data almost 2x more than simply concatenating then compressing
- Design and develop mcrEngine
  - Grouped (N→M) checkpointing system
  - Improves checkpointing frequency
  - Improves application performance

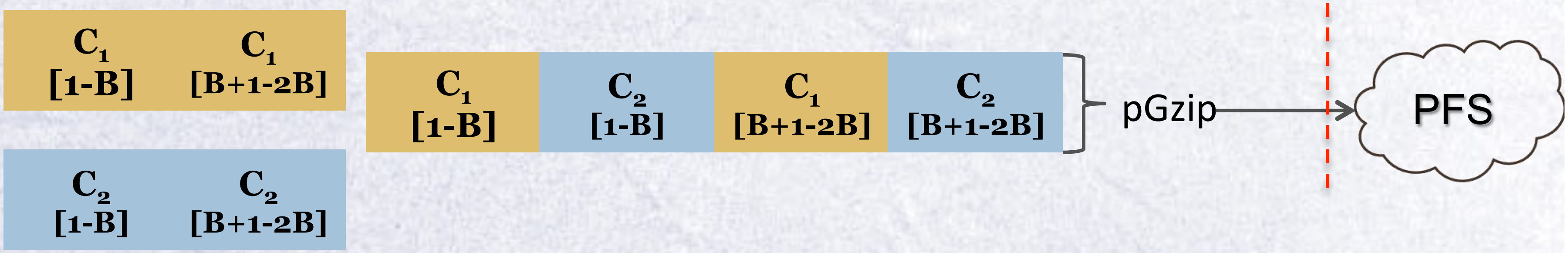
Overview

- Background
- Problem
- Data aggregation & compression
- Evaluation

Data-Agnostic Schemes

- Agnostic scheme – concatenate checkpoints

- Agnostic-block scheme – interleave fixed-size blocks



- Observations:
  - (+) Easy
  - (–) Low compression ratio

Identify Similar Variables ... Processes

P0

```
Group ToyGrp{
    float Temperature[1024];
    int Pressure[20][30];
};
```

P1

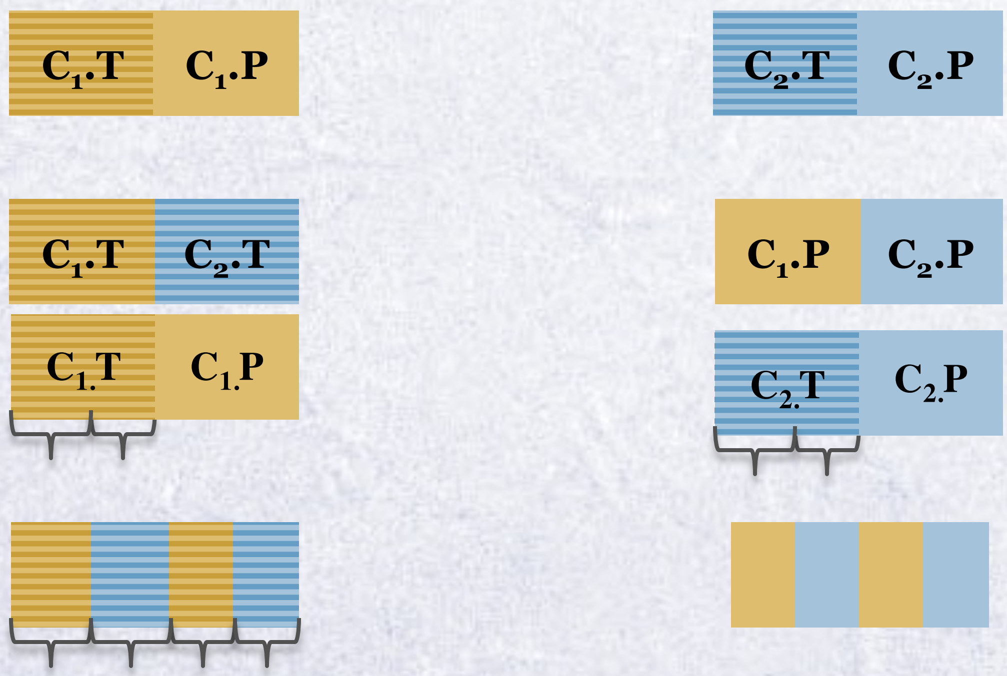
```
Group ToyGrp{
    float Temperature[100];
    int Pressure[10][50];
};
```

Meta-data:

1. Name
2. Data-type
3. Class:
   -- Array, Atomic

Concatenating similar variables

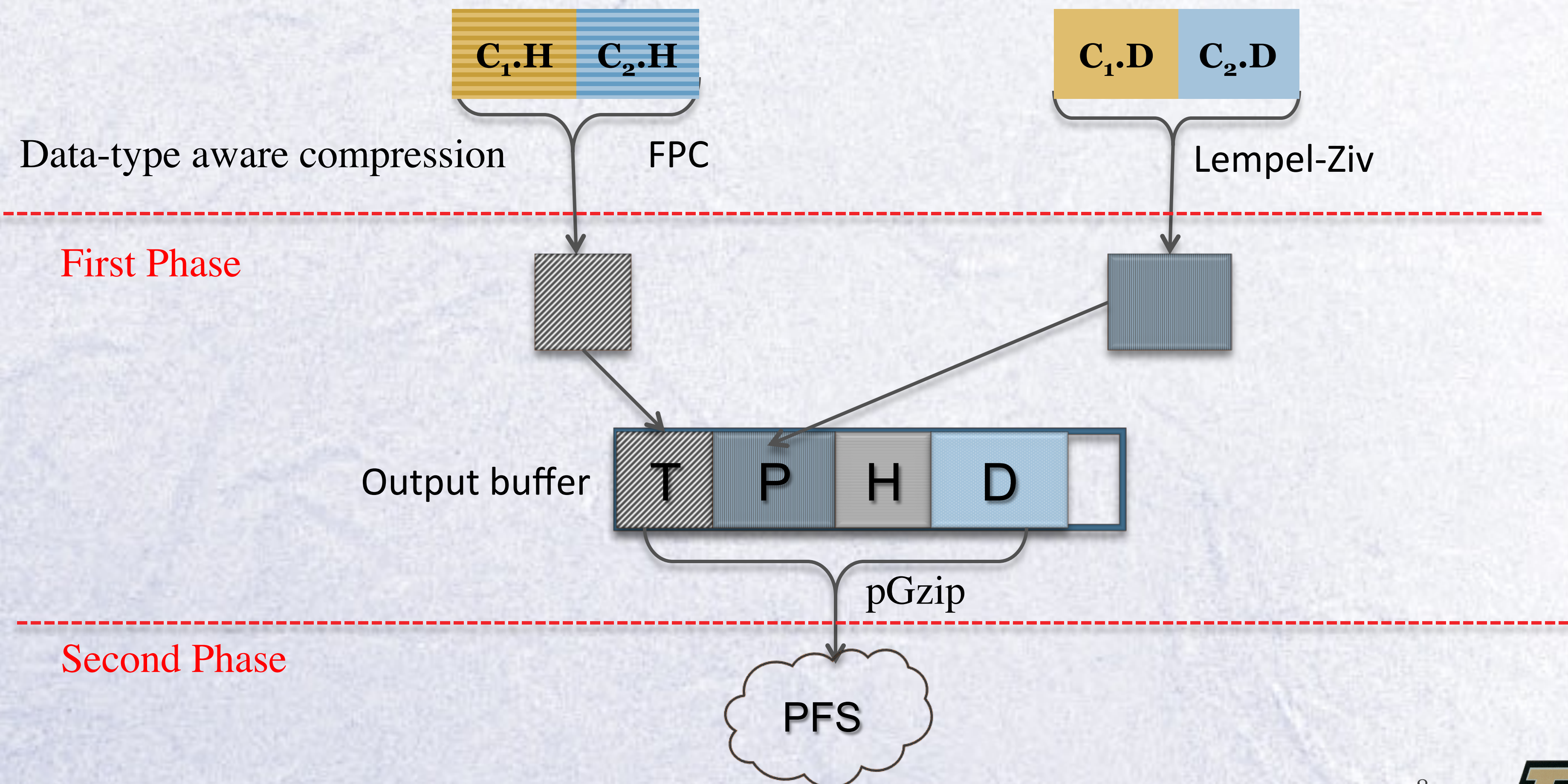
Interleaving similar variables



Next 'B' bytes / First 'B' bytes
of Temperature / of Pressure

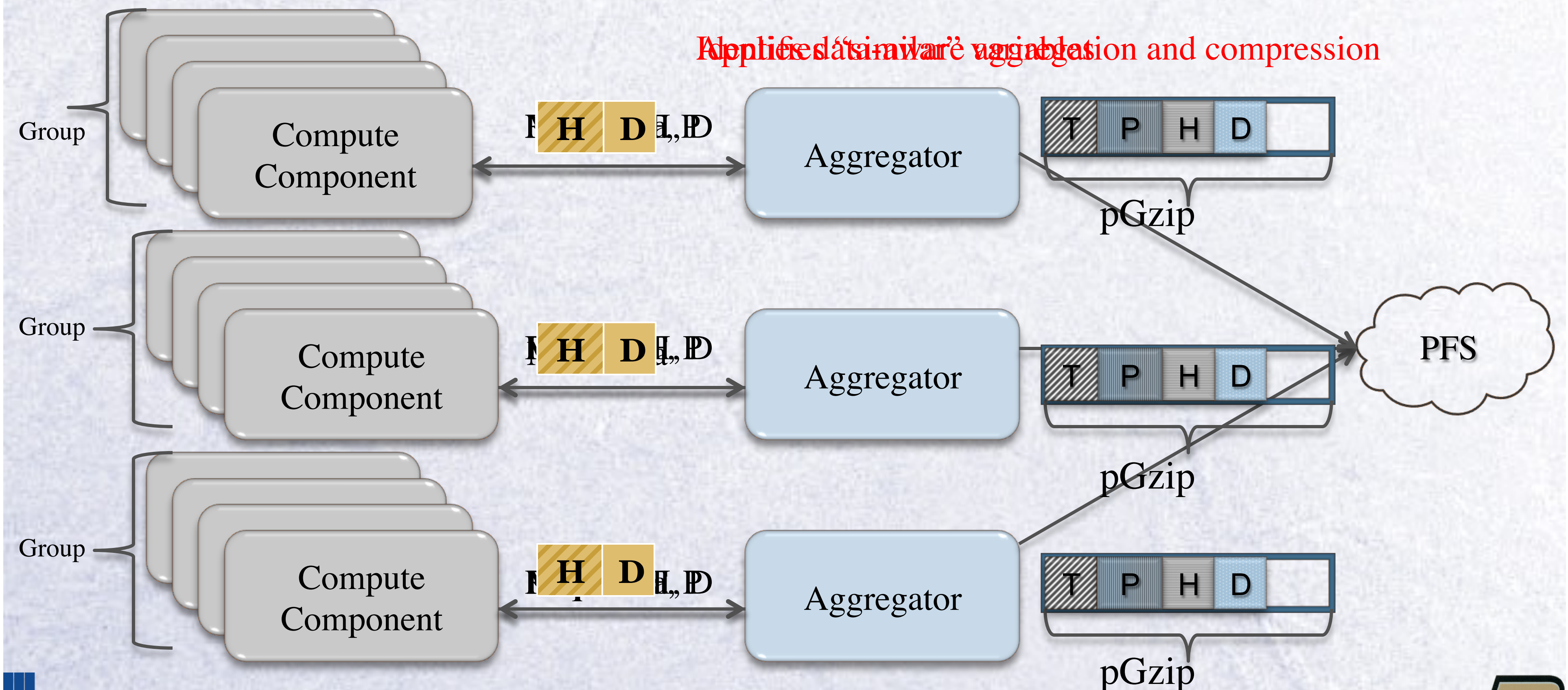
Data-Aware Aggregation & Compression

- Aware scheme – concatenate similar variables
- Aware-block scheme – interleave similar variables



How mcrEngine Works

- CNC : Compute node component
- ANC: Aggregator node component
- Rank-order groups, Grouped (N➔M) transfer



Overview

- Background
- Problem
- Data aggregation & compression
- Evaluation

Evaluation

- Applications
  - ALE3D – 4.8GB per checkpoint set
  - Cactus – 2.41GB per checkpoint set
  - Cosmology – 1.1GB per checkpoint set
  - Implosion – 13MB per checkpoint set
- Experimental test-bed
  - LLNL's Sierra: 261.3 TFLOP/s, Linux cluster
  - 23,328 cores, 1.3 Petabyte Lustre file system
- Compression algorithm
  - FPC [1] for double-precision float
  - Fpzip [2] for single-precision float
  - Lempel-Ziv for all other data-types
  - pGzip for general-purpose compression

Evaluation Metrics

- Effectiveness of data-aware compression
  - What is the benefit of multiple compression phases?
  - How does group size affect compression ratio?

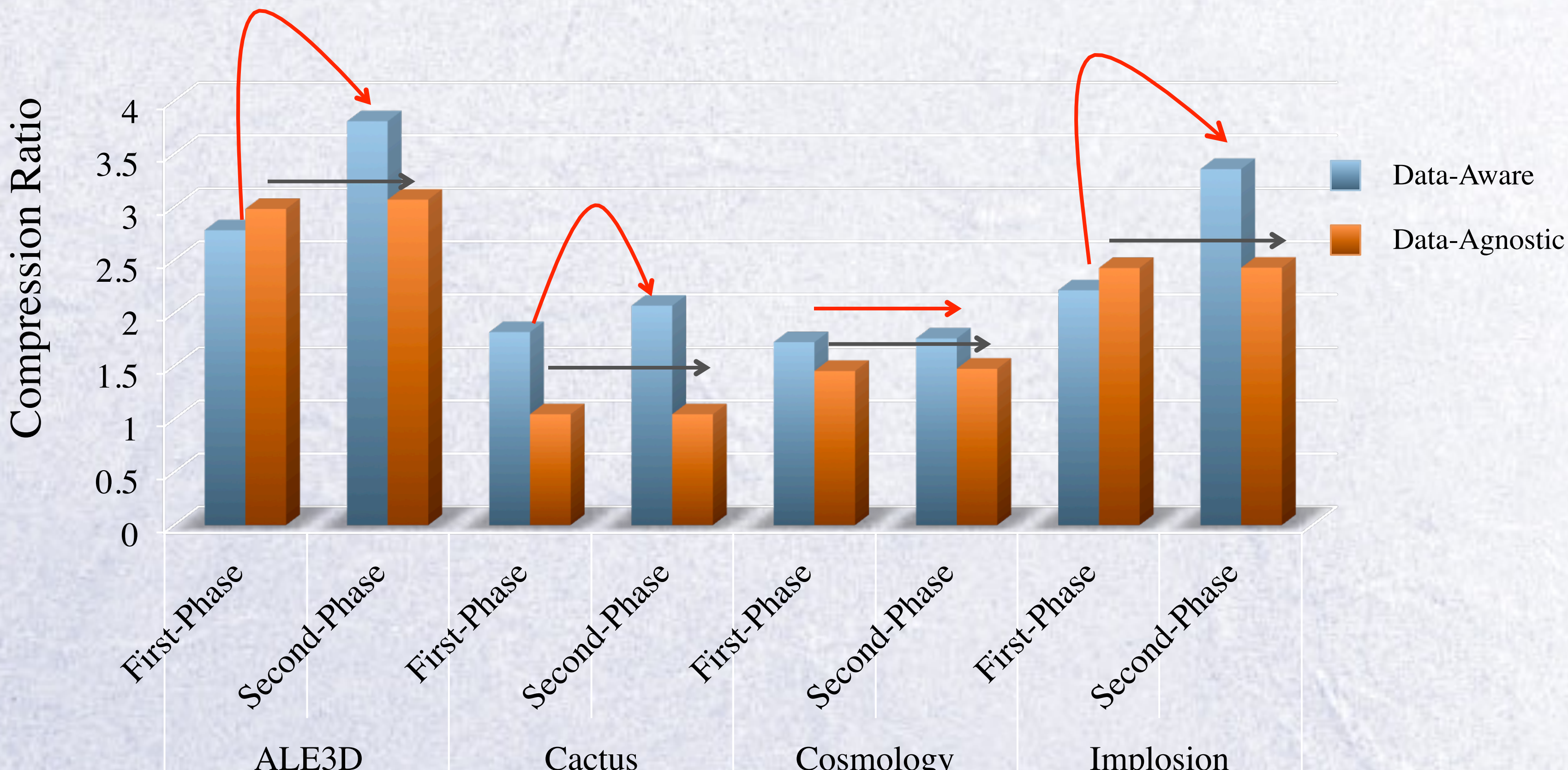
$$\text{Compression ratio} = \frac{\text{Uncompressed size}}{\text{Compressed size}}$$

- Performance of mcrEngine
  - Overhead of the checkpointing phase
  - Overhead of the restart phase

No Benefit with Data-Agnostic Double Compression

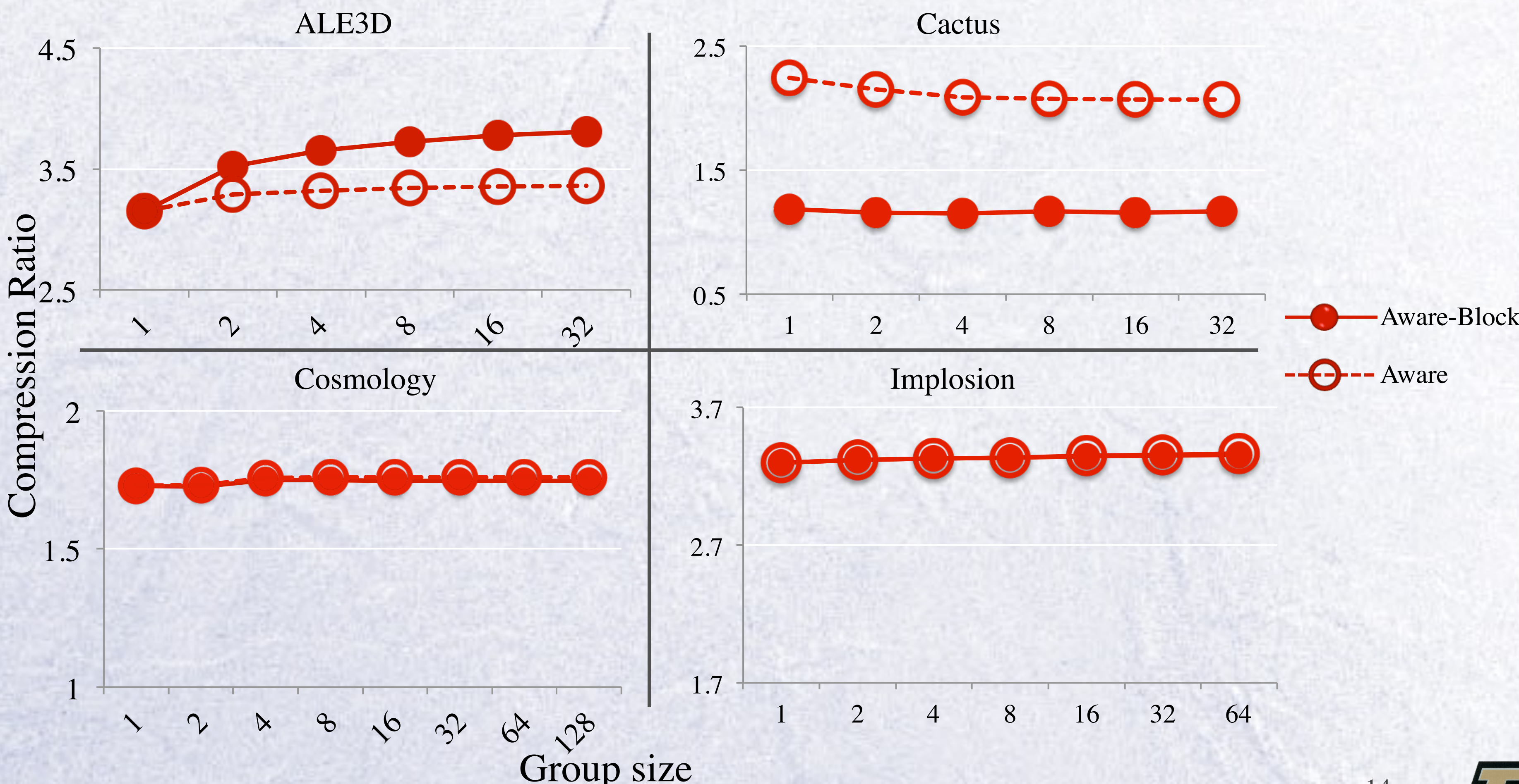
Multiple Phases of Data-Aware Compression are Beneficial

- Data-type aware compression improves compressibility
  - First phase changes underlying data format
- Data-agnostic double compression is not beneficial
  - Because, data-format is non-uniform and uncompressible



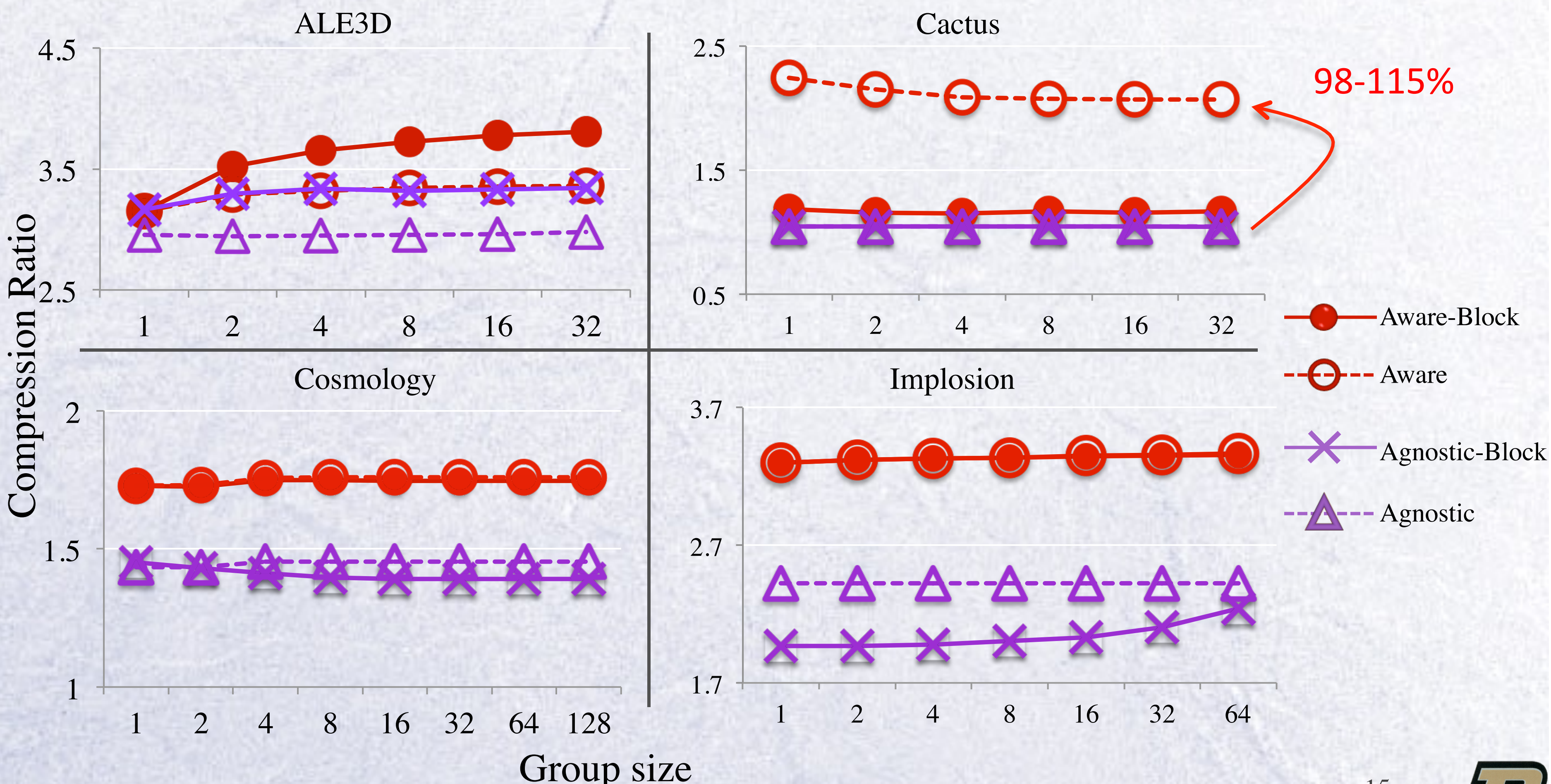
Impact of Group Size on Compression Ratio

- Different merging schemes better for different applications
- Larger group size beneficial for certain applications
  - ALE3D: Improvement of 8% from group size 2 to 32



Data-Aware Technique Always Wins over Data-Agnostic

- Data-aware technique always yields better compression ratio than Data-Agnostic technique

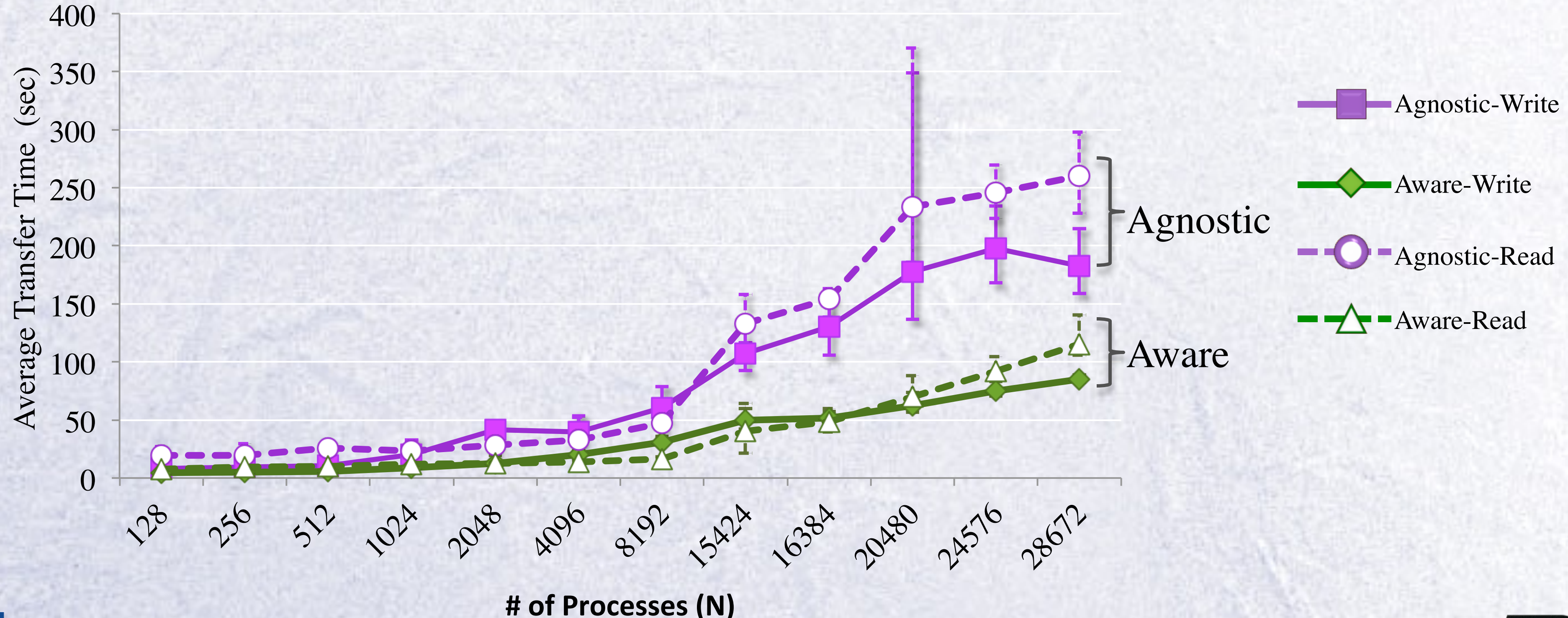


Summary of Effectiveness Study

- Data-aware compression always wins
  - Reduces gigabytes of data for Cactus
- Larger group sizes may improve compression ratio
- Different merging schemes for different applications
- Compression ratio follows course of simulation
  - Details in our paper

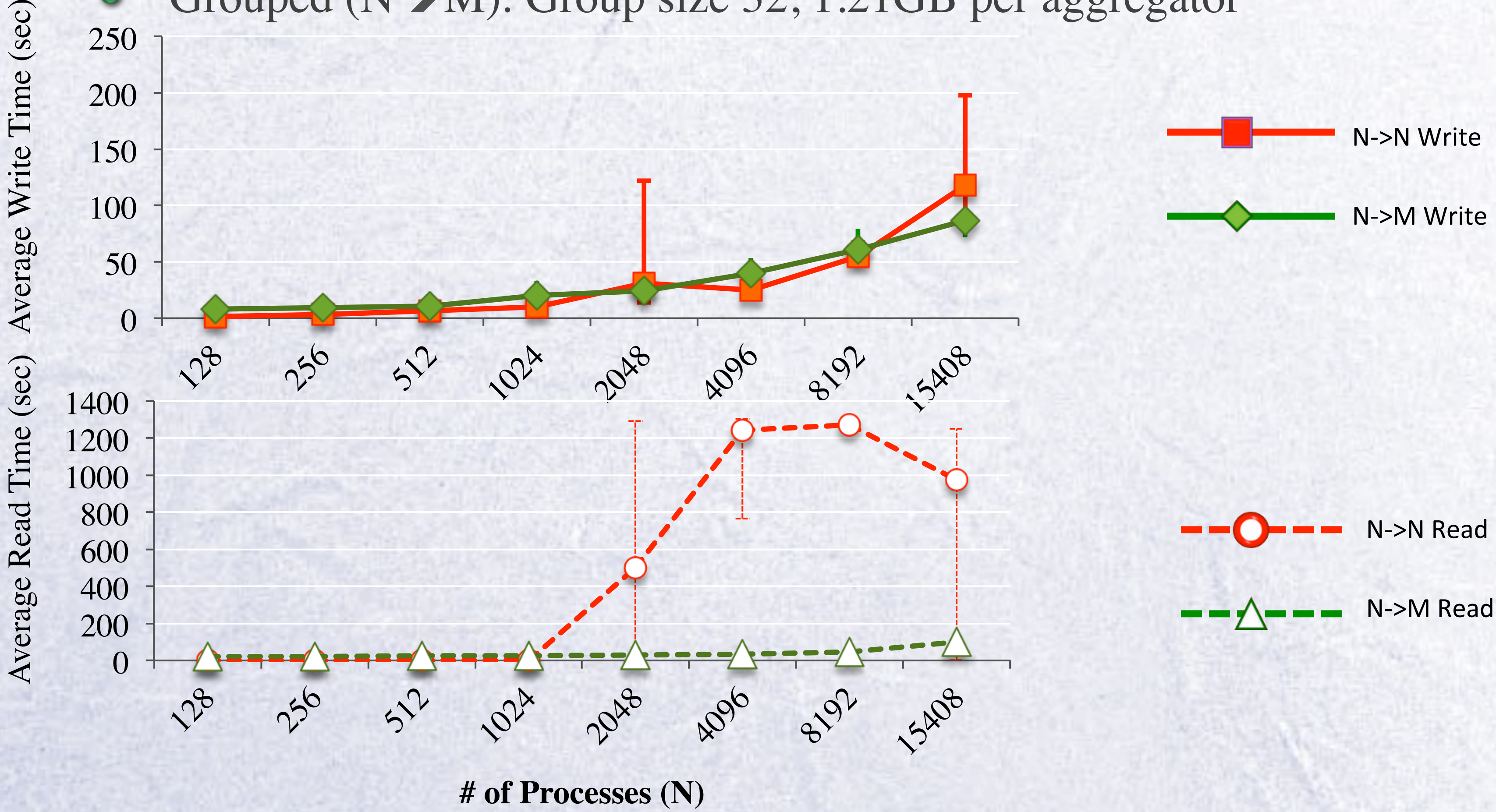
Impact of Data-Aware Compression on Latency

- IOR with Grouped(N→M) transfer, groups of 32 processes
  - Data-aware: 1.2GB, data-agnostic: 2.4GB
- Data-aware compression improves I/O performance at large scale
  - Improvement during write 43% - 70%
  - Improvement during read 48% - 70%



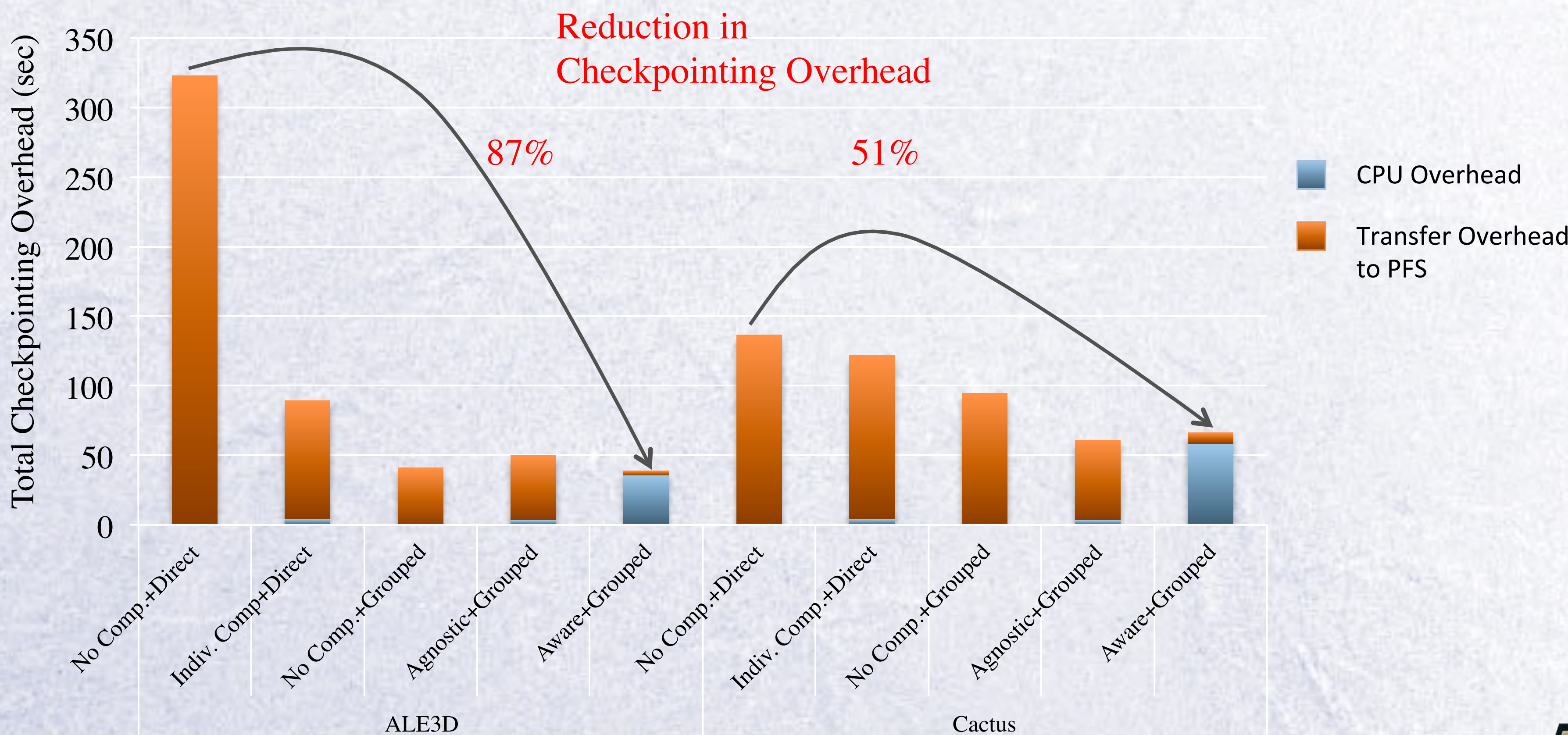
Impact of Aggregation & Compression on Latency

- Used IOR
  - Direct (N➔N): 87MB per process
  - Grouped (N➔M): Group size 32, 1.21GB per aggregator



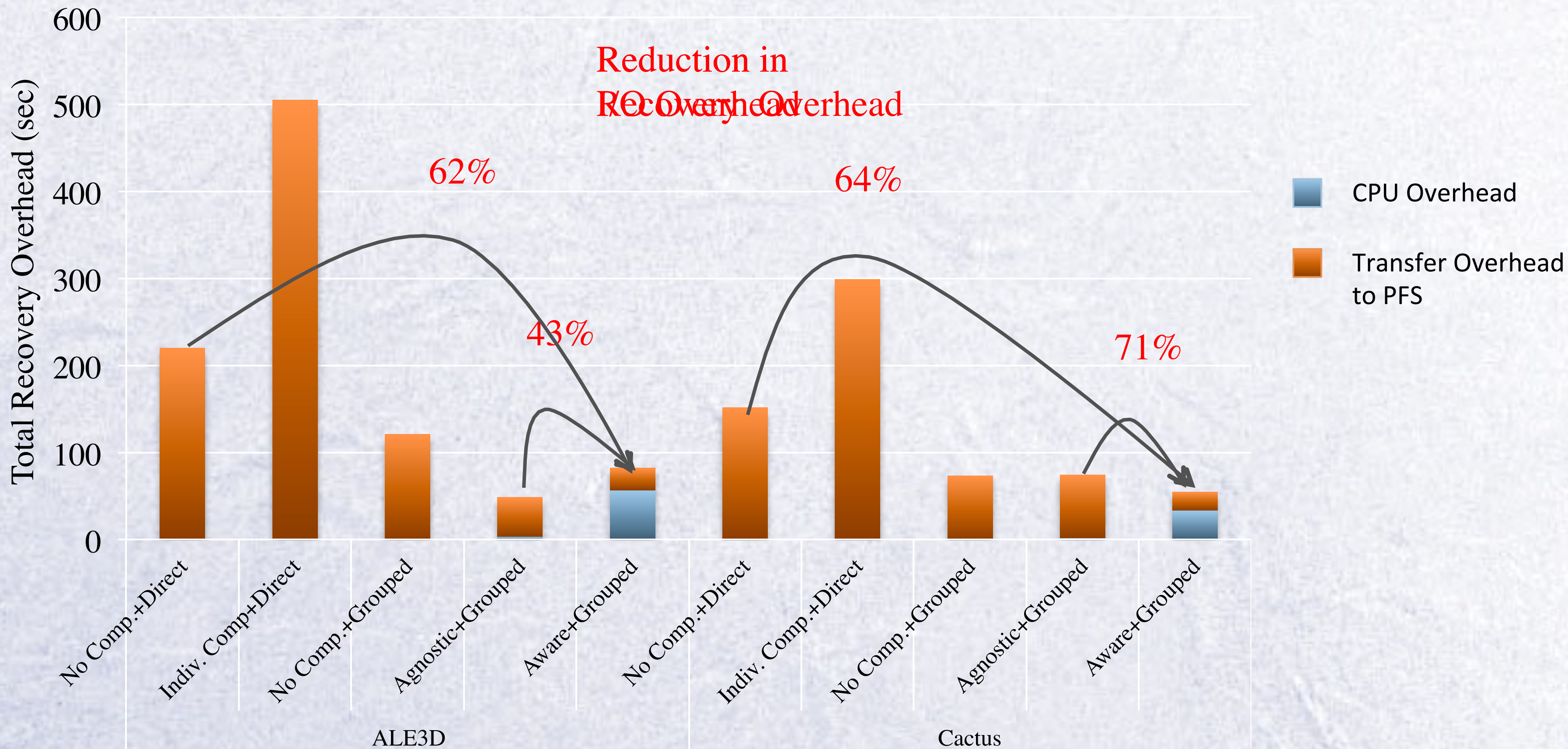
End-to-End Checkpointing Overhead

- 15,408 processes
  - Group size of 32 for N➔M schemes
  - Each process takes a checkpoint
- Converts network bound operation into CPU bound one



End-to-End Restart Overhead

- Reduced overall restart overhead
- Reduced network load and transfer time



Conclusion

- Developed data-aware checkpoint compression technique
  - Relative improvement in compression ratio up to 115%
- Investigated different merging techniques
  - Demonstrated effectiveness using real-world applications
- Designed and developed mcrEngine
  - Reduces recovery overhead by more than 62%
  - Reduces checkpointing overhead by up to 87%
  - Improves scalability of checkpoint-restart systems

Contact Information

- Tanzima Islam (tislam@purdue.edu)
- Website: web.ics.purdue.edu/~tislam