Fast Checkpointing by Write Aggregation with Dynamic Buffer Interleaving

Xiangyong Ouyang
Karthik Gopalakrishnan
Tejus Gangadharappa
Dhabaleswar K. (DK) Panda

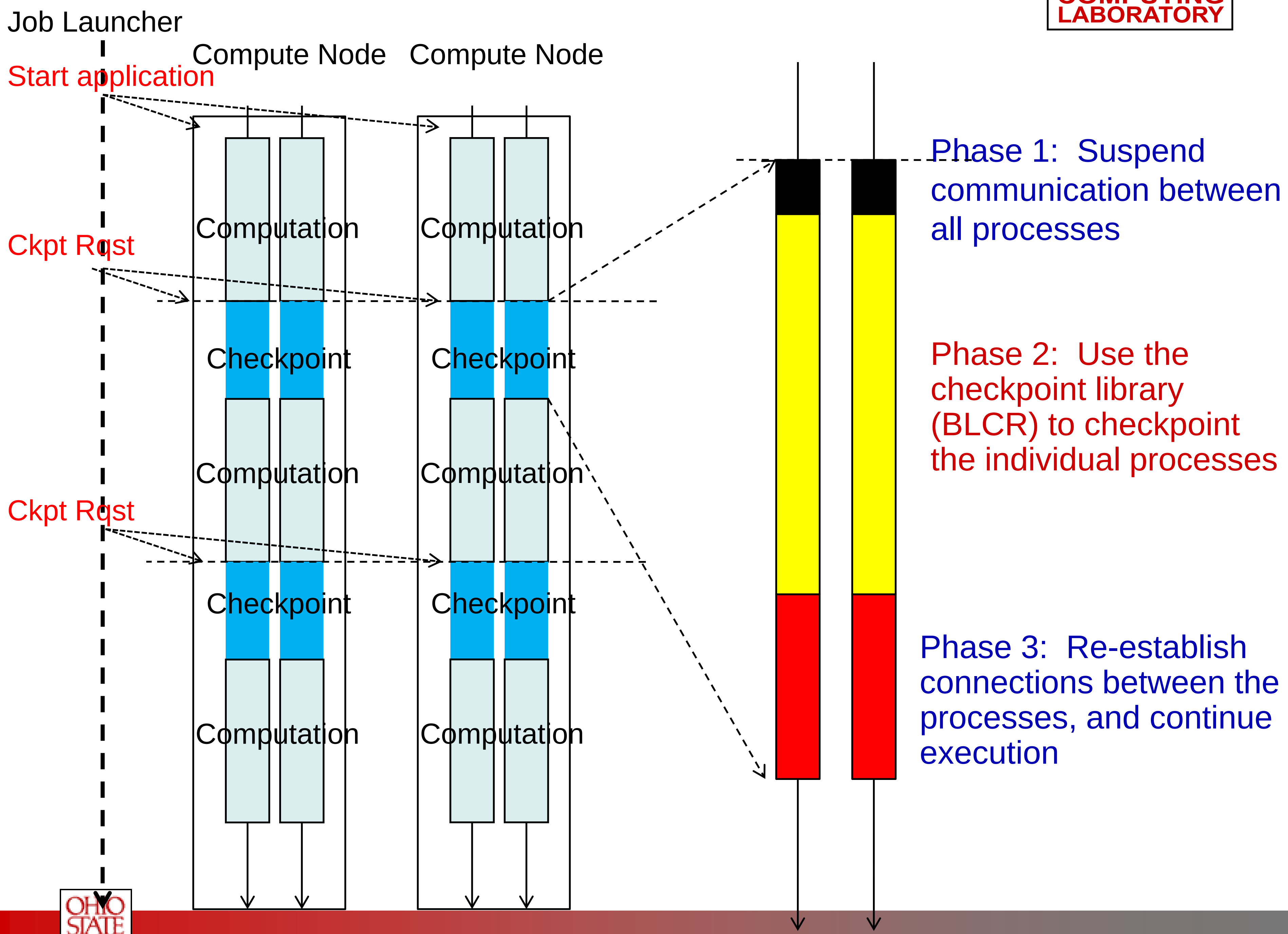
Department of Computer Science & Engineering
The Ohio State University

Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Write-Aggregation Design
- Performance Evaluation
- Conclusions and Future Work

Motivation

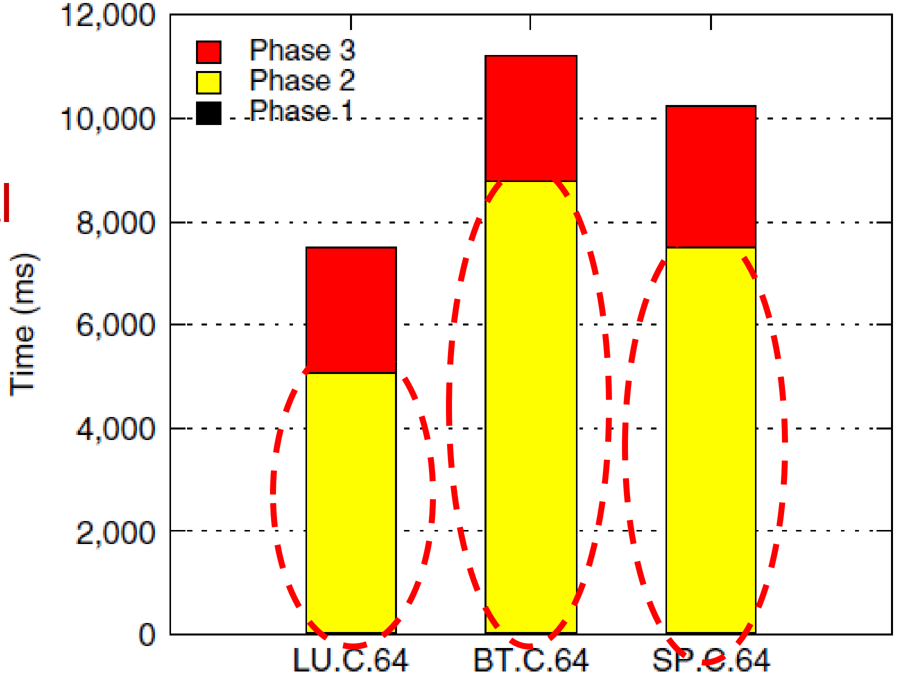
- Mean-time-between-failures (MTBF) is getting smaller as clusters continue to grow in size
  - Fault-Tolerance is becoming imperative in modern clusters
  - Checkpoint/Restart is becoming increasingly important
- Multi-core architectures are gaining momentum
  - Multiple processes on a same node checkpoint simultaneously
- Existing Checkpoint/Restart mechanisms don’t scale well with increasing job size
  - Multiple streams intersperse their concurrent writes to a shared storage media
  - A low utilization of the raw throughput of the underlying file system


Job Launcher
Compute Node
Compute Node
Start application
Ckpt Rqst
Ckpt Rqst
Computation
Computation
Checkpoint
Checkpoint
Computation
Computation
Checkpoint
Checkpoint
Computation
Computation
Phase 1: Suspend communication between all processes
Phase 2: Use the checkpoint library (BLCR) to checkpoint the individual processes
Phase 3: Re-establish connections between the processes, and continue execution

NETWORK-BASED COMPUTING LABORATORY

Phase 2 of Checkpointing

- Phase 2 involves writing a process' context and memory contents to a checkpoint file
- Usually this phase dominates the total time to do a checkpoint



OHIO STATE

Problem Statement

- What's the checkpoint data writing pattern of a typical MPI application using BLCR?
- Can we optimize the data writing path in a multicore architecture to improve the Checkpoint performance?
- What are the costs of the optimizations?

Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Write-Aggregation Design
- Performance Evaluation
- Conclusions and Future Work

NETWORK-BASED COMPUTING LABORATORY

MVAPICH/MVAPICH2 Software

- High Performance MPI Library for InfiniBand and 10GE
  - MVAPICH (MPI-1) and MVAPICH2 (MPI-2)
  - Used by more than 1000 organizations in 53 countries
  - More than 35,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - Tianhe-1: 5th 71,680-cores in China (in Nov. 2009)
    - Ranger: 9th 62,976-core at TACC (in Nov. 2009)
  - Available with software stacks of many IB, 10GE and server vendors including Open Fabrics Enterprise Distribution (OFED)
  - http://mvapich.cse.ohio-state.edu/

OHIO STATE

Profiling Configuration

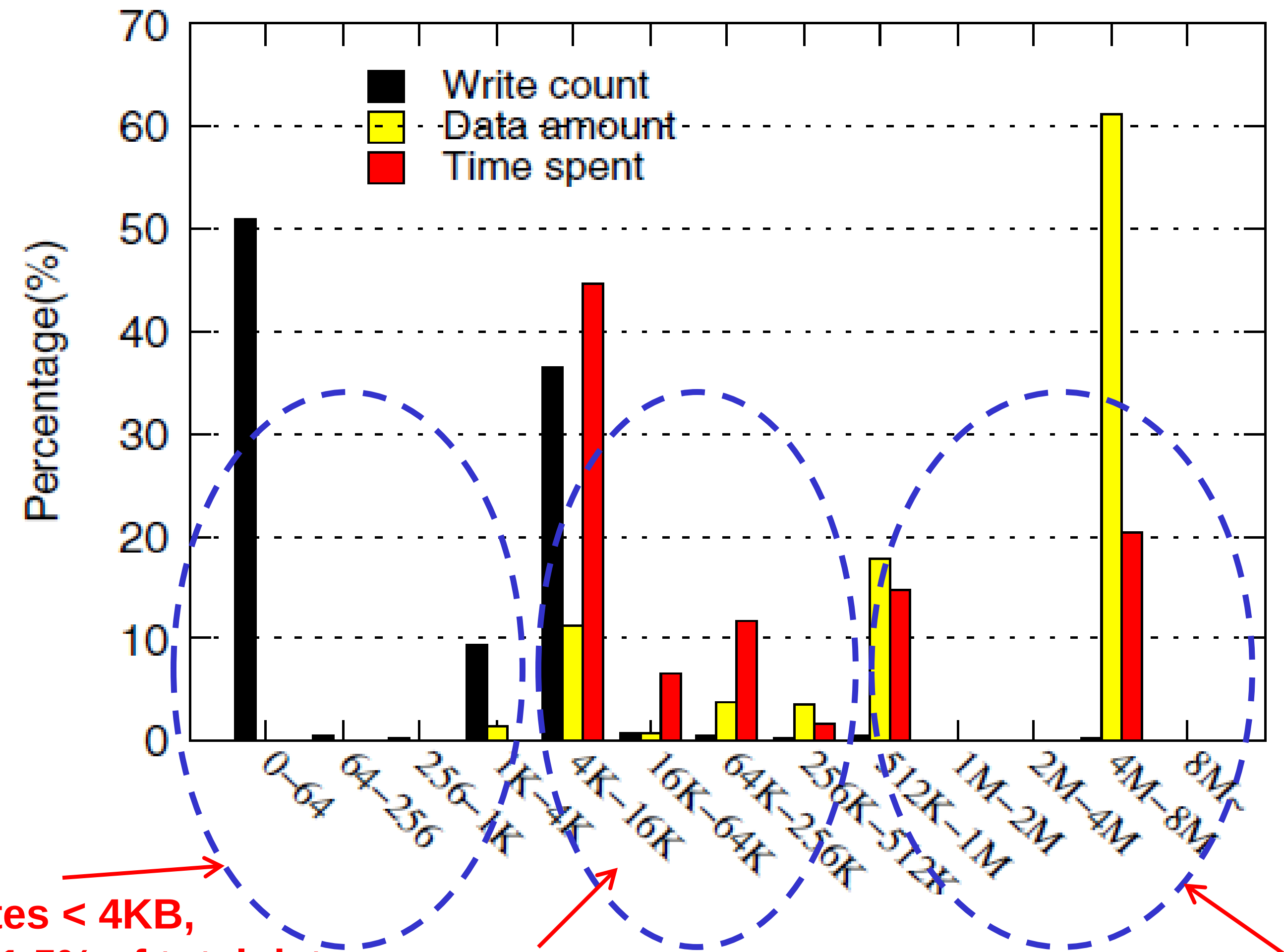
- NAS Parallel Benchmark suite version 3.2.1
  - Class C, 64 processes
  - Each process on one processor
  - Each process writes checkpoint data to a separate file on a local ext3 file system
- MVAPICH2 Checkpoint/Restart framework
  - BLCR 0.8.0 extended to provide profiling information
- Intel Clovertown cluster
  - Dual-socket Quad core Xeon processors, 2.33GHz
  - 8 processor per node, nodes connected by InfiniBand DDR
  - Linux 2.6.18

Basic Checkpoint Information

Checkpoint writing information (class C, 64 processes, 8 processes/node)

	LU.C.64	BT.C.64	SP.C.64
Checkpoint file size(MB) per process	23	40.0	39.5
Total data size(MB) per node	184	320.0	316.0
Number of VFS write per process	975	1057	1156
Total VFS write per node	7800	8456	9248

Checkpointing Profiling (1) (LU.C.64)

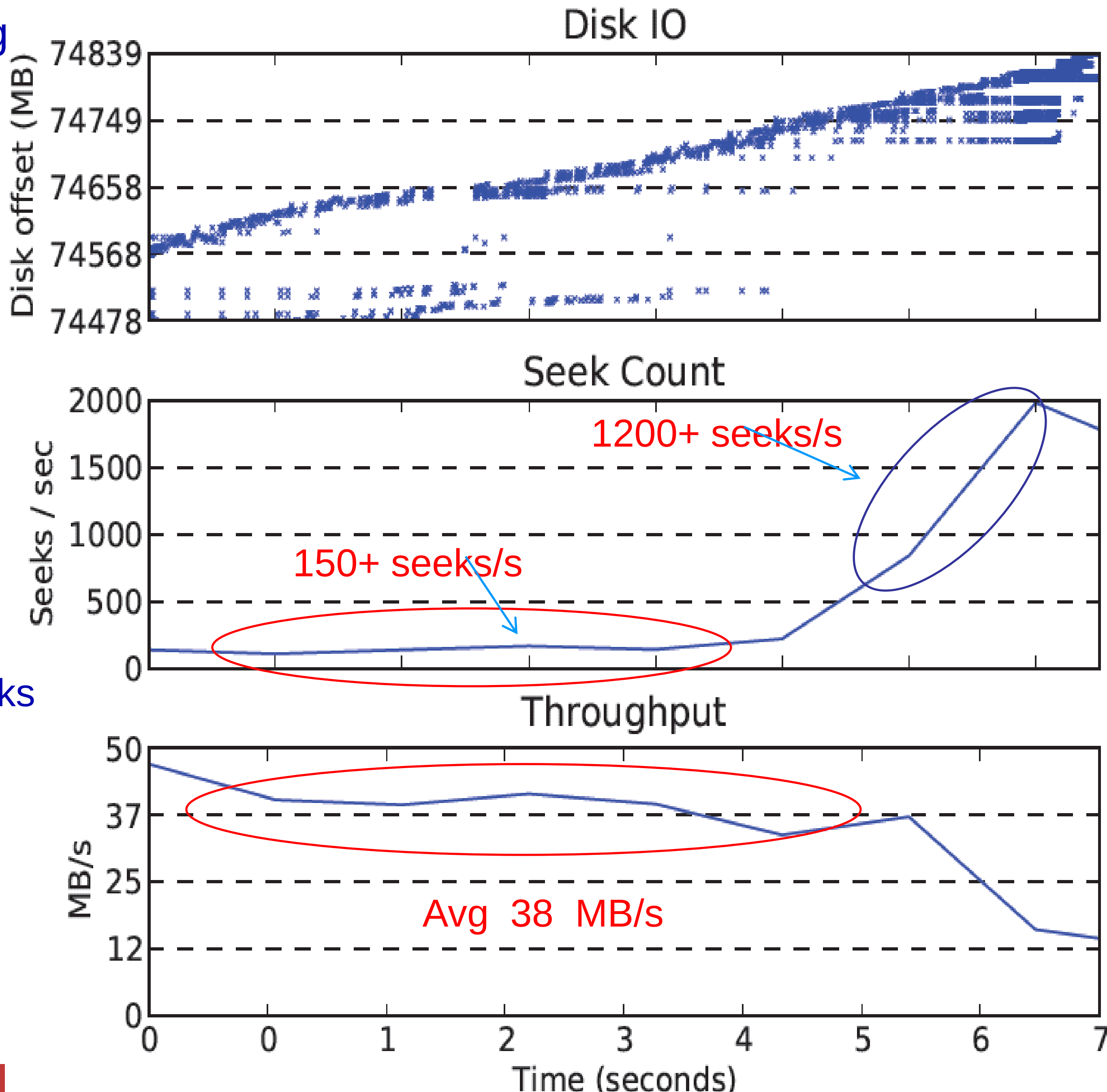


- 60% of writes < 4KB,
- contribute 1.5% of total data,
- consume 0.2% of total write time

- 38% of all writes
- contribute 20% of all data
- consume 65 % of all time

- 0.8% of writes > 512KB
- contribute 79% of all data
- consume 35% of total write time

Checkpointing Profiling (BT.C.64)

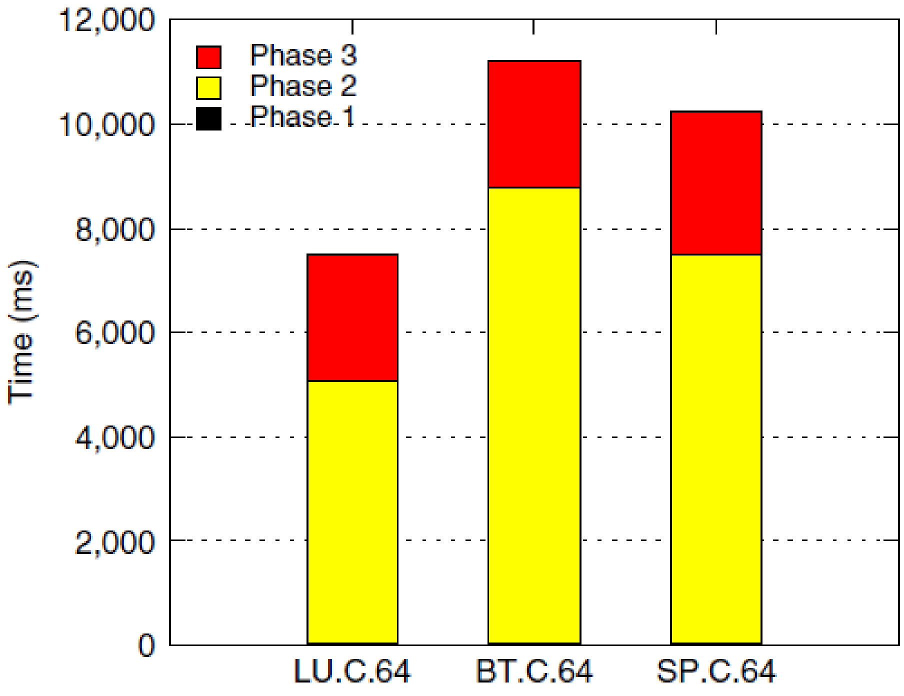


•Use "blktrace" to collect all block layer IO tracing

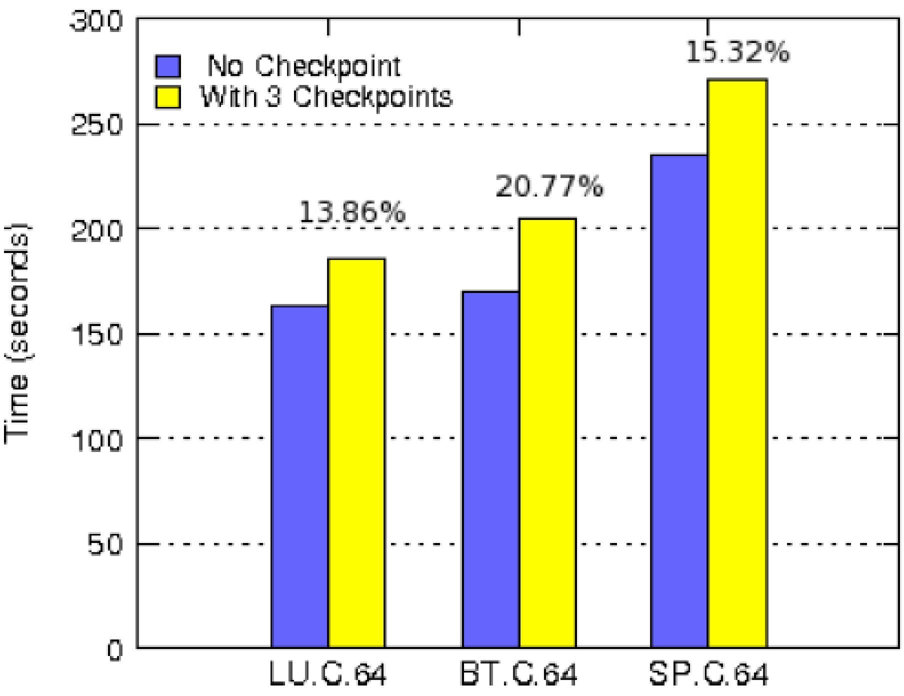
•Multiple write streams intersperse their concurrent writes to a shared storage media
➔ A lot of disk head seeks

Disk raw bandwidth
= 60MB/s

Checkpointing Profiling(3)



Decomposition of Checkpoint Time



Application execution time with/without checkpoints

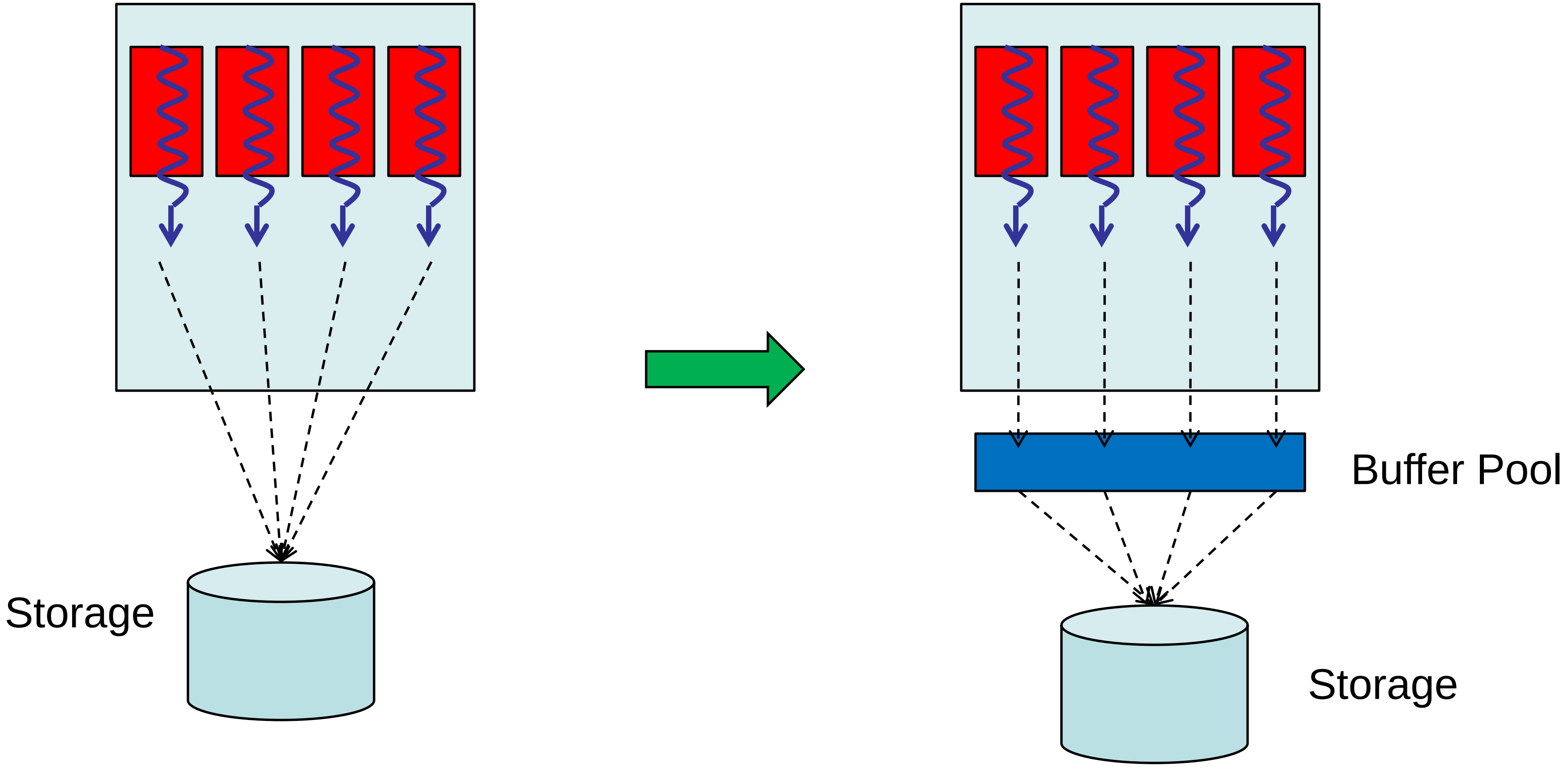
NETWORK-BASED COMPUTING LABORATORY

Outline

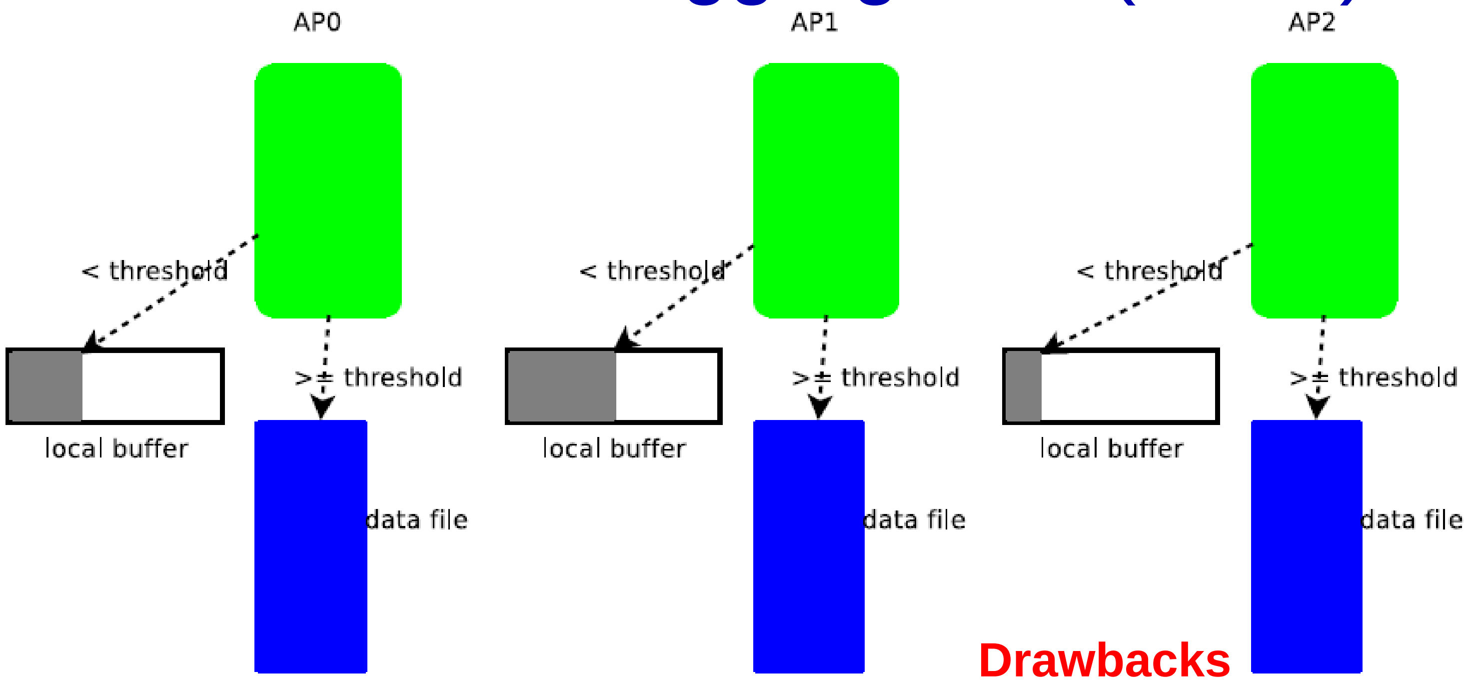
- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Write-Aggregation Design
- Performance Evaluation
- Conclusions and Future Work

OHIO STATE

Basic Design Strategy



Basic Write-Aggregation (WAG)



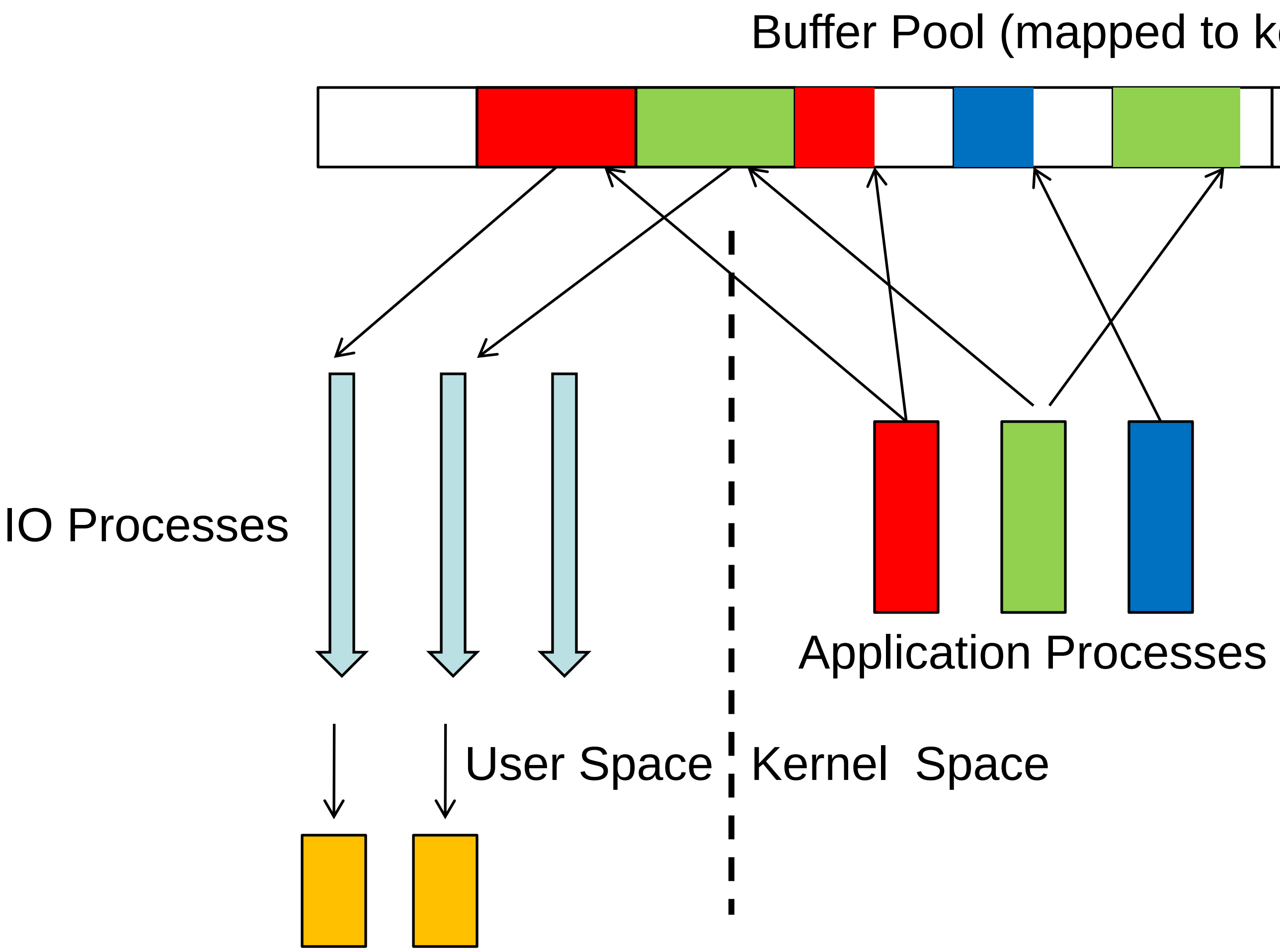
- Application process (AP) coalesces all writes < threshold to a process-specific buffer
- All writes >= threshold is directly written to data files
- At end of checkpoint, AP writes local buffer to a data file

Drawbacks

- Not optimal memory usage
  - A separate local buffer for each process
- Blocking mode File-writing (>=threshold)
  - AP waits for completion of large file writes

X. Ouyang, K. Gopalakrishnan and D. K. Panda, "Accelerating Checkpoint Operations by Node-Level Write Aggregation on Multicore Systems", ICPP '09, Sept. 2009

Write Aggregation with Dynamic Buffer Interleaving (WAG-DBI)



- VFS writes from a process are aggregated into a buffer-chunk
- Data from different processes are interleaved in the buffer pool

- IO processes write each buffer-chunk to a chunk file

Methodology of WAG-DBI

✓Aggregate many VFS writes into larger chunk writes
- •Reduce disk seeks, improve bandwidth utilization

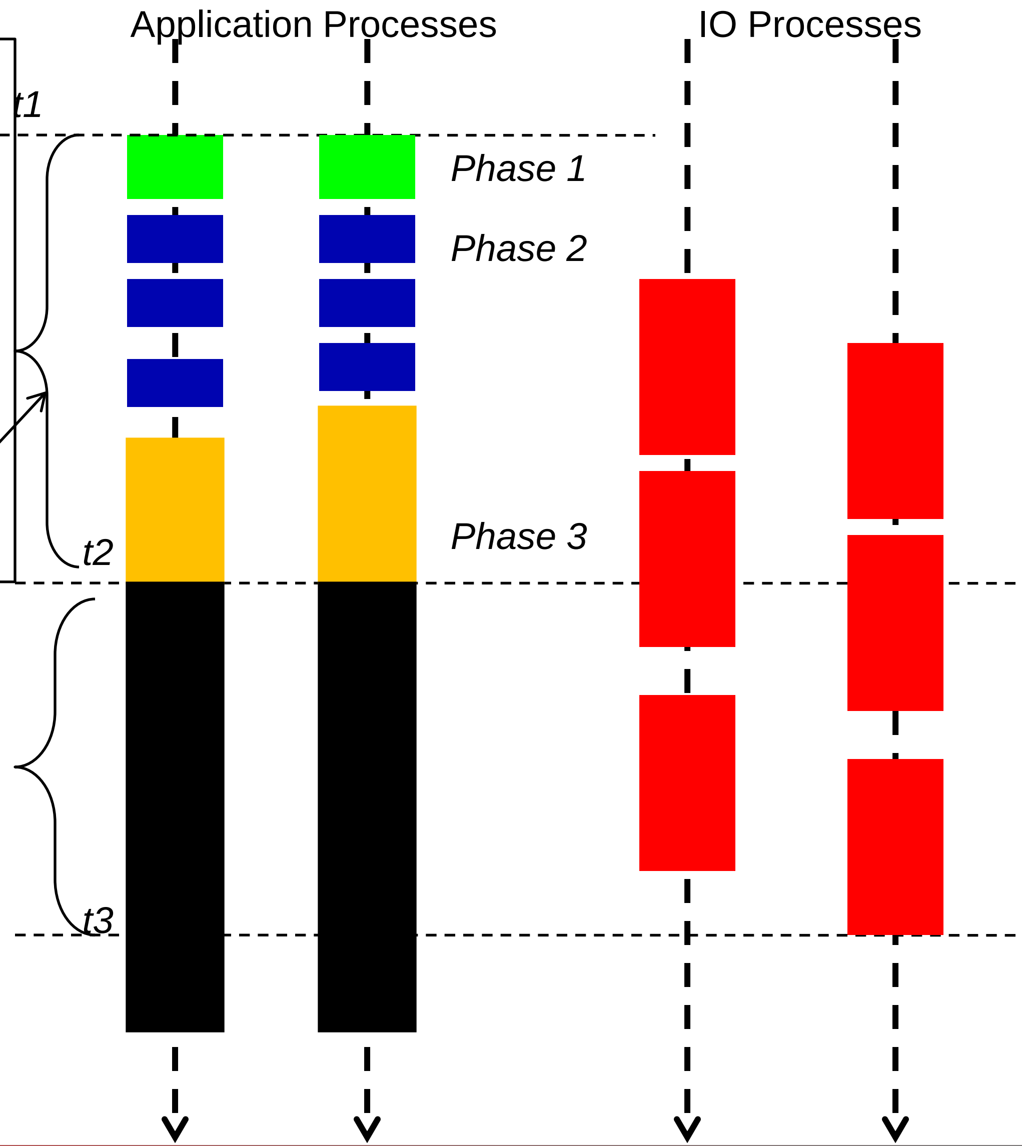
✓Overlaps application progress with slow checkpoint file writing
- •Improve application execution time

•Application-perceived checkpoint time = *t2 - t1*

•*(t3 - t2)* is overlapping between Application-Progress and File-Write

•Vulnerable to failures in this window

•Provide an interface to poll for File-Write completion



Restart for WAG-DBI

- The data file for each buffer-chunks has its filename encoding
  - (ckpt-id, Process-id, logical-offset)
- Reconstruct checkpoint files at restart
  - Find all files named as: ckptX-procY-offset1, ckptX-procY-offset2, …
  - Sort all filenames according to offset in ascending order
  - Concatenate all files into one large file

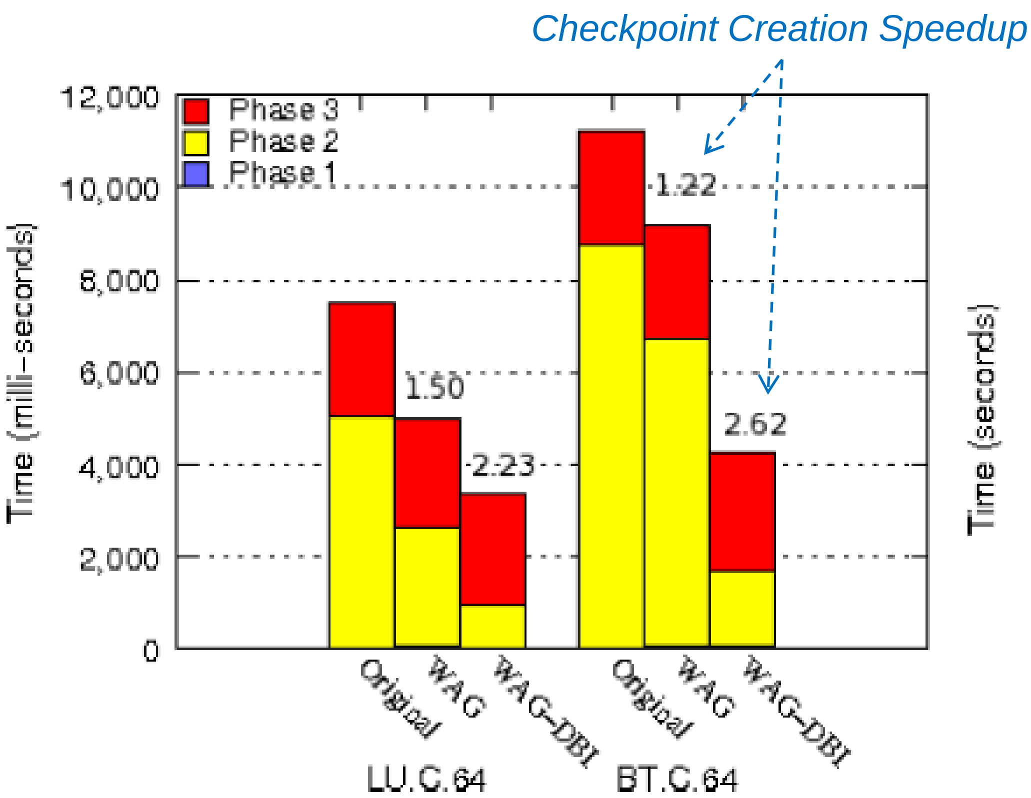
Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Write-Aggregation Design
- Performance Evaluation
- Conclusions and Future Work

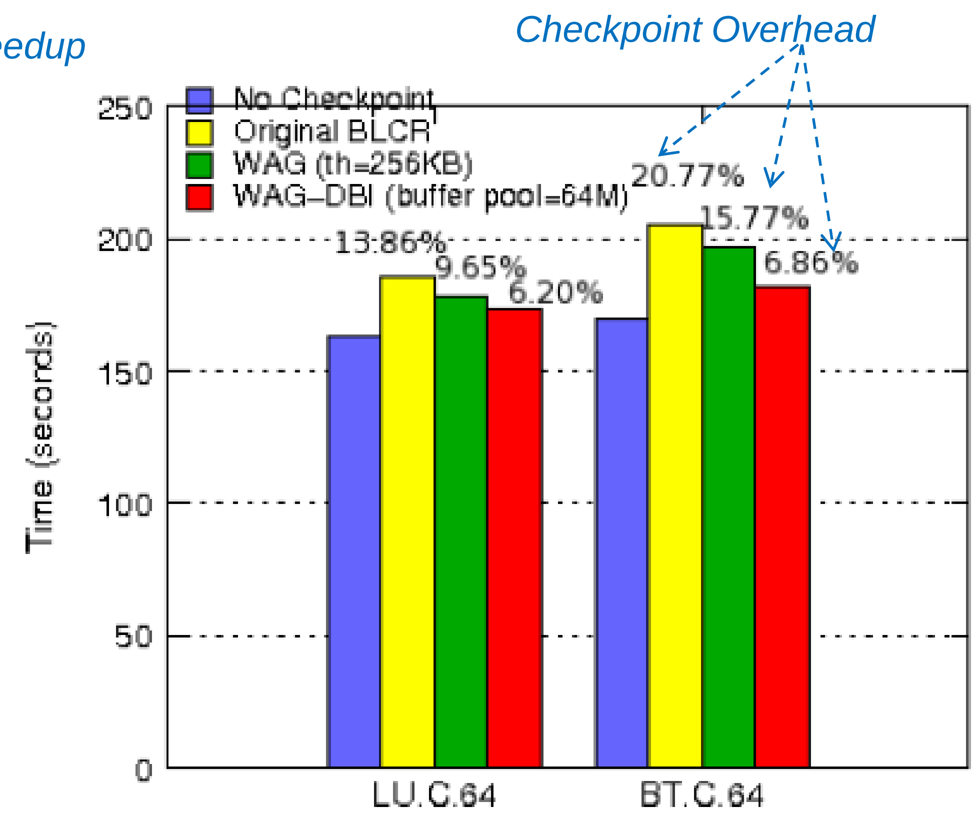
Experiments setup

- System setup
  - Intel Clovertown cluster
    - Dual-socket Quad core Xeon processors, 2.33GHz
    - 8 processor per node, nodes connected by InfiniBand
    - Linux 2.6.18
  - NAS parallel Benchmark suite version 3.2.1
    - LU/BT/CG, Class C, 64 processes
    - Each process on one processor
    - 8 nodes are used
    - Each process writes checkpoint data to a separate file on a local ext3 file system
  - MVAPICH2 Checkpoint/Restart framework, with BLCR 0.8.0 extended with WAG-DBI

Performance Comparison



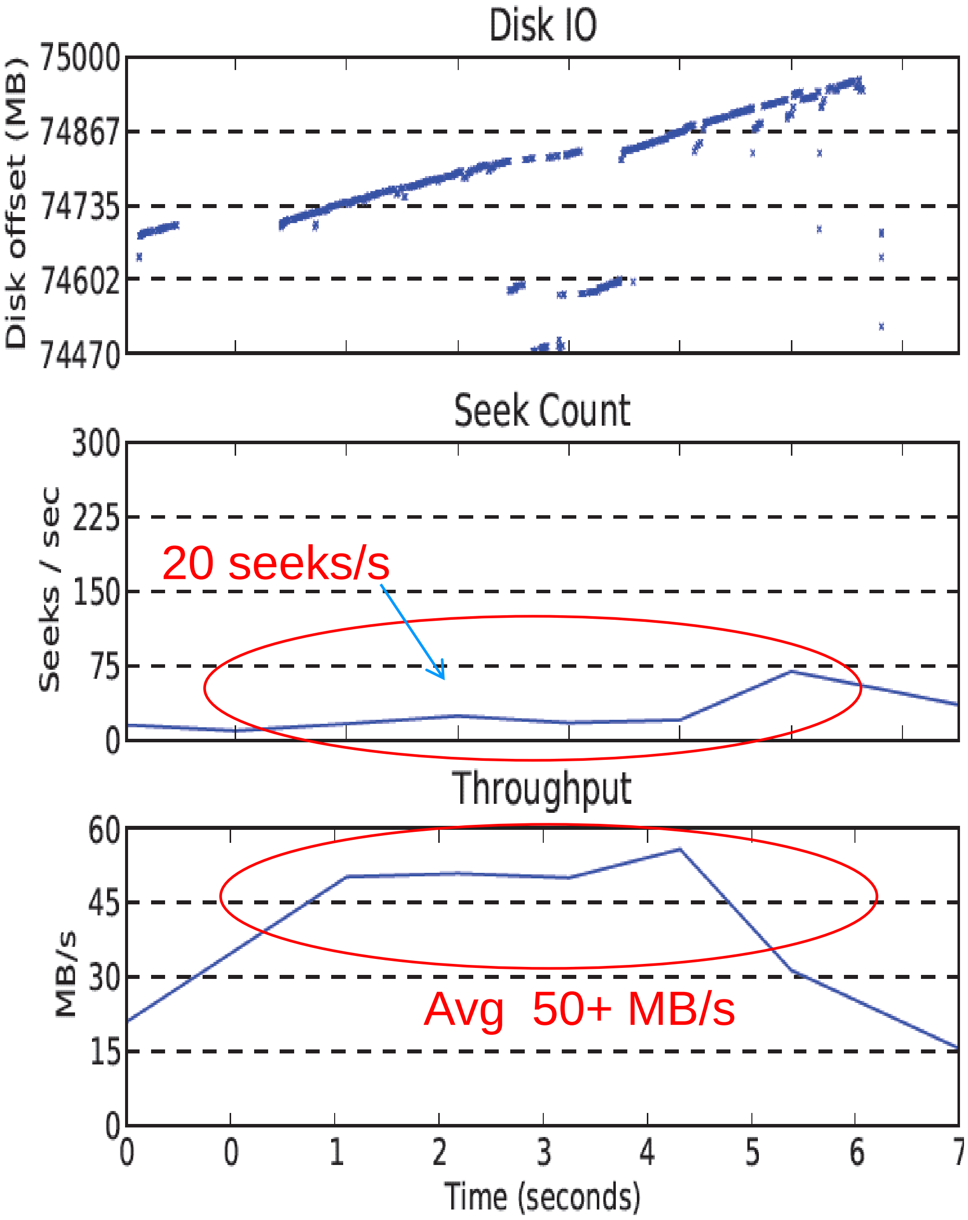
Checkpoint Time Decomposition



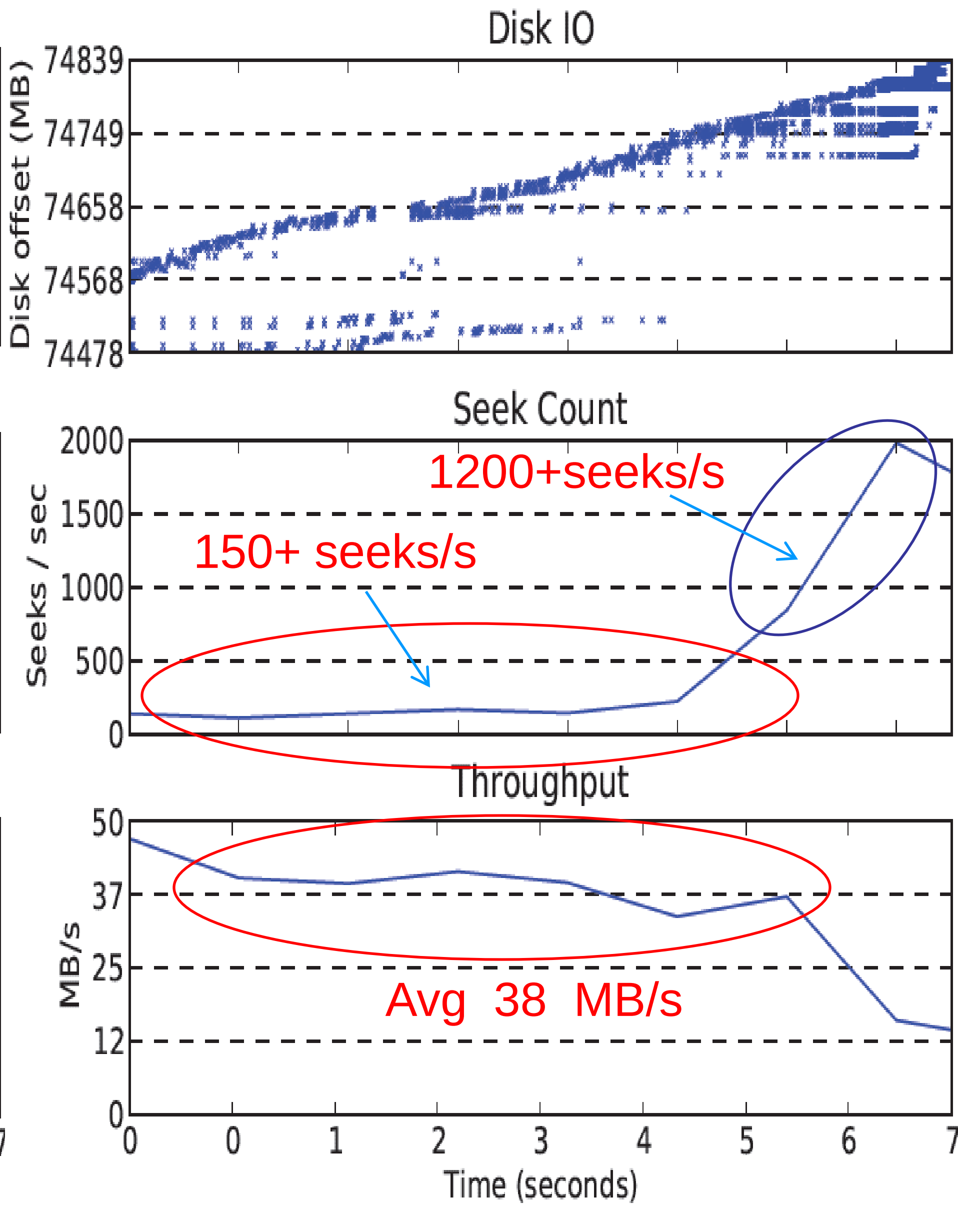
Application Execution Time with 3 Checkpoints

- *WAG: aggregation threshold = 256 KB*
  - *LU.C.64 uses 78.2 MB memory, BT.C.64 uses 81.2 MB memory*
- *WAG-DBI: buffer pool = 64 MB*

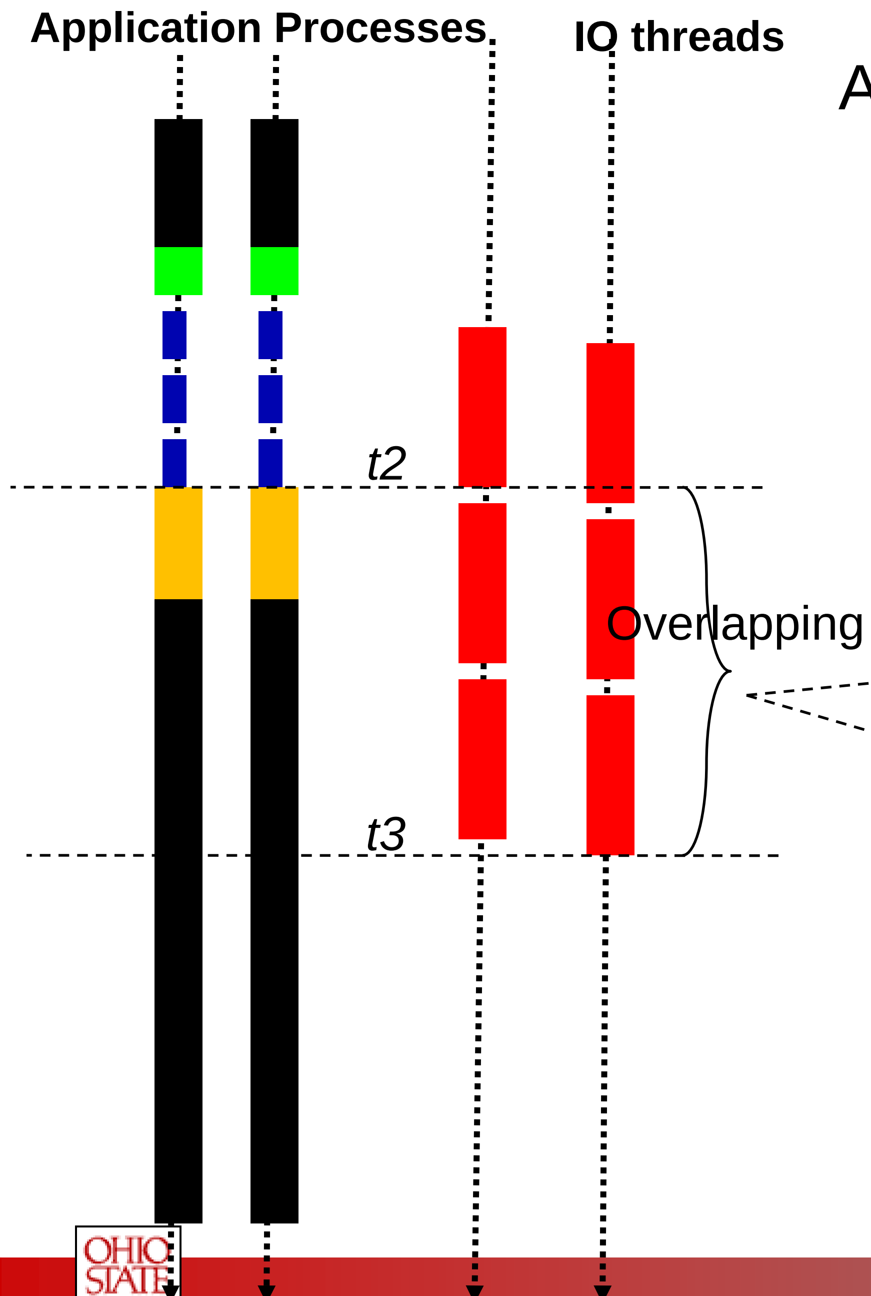
WAG-DBI, buffer=64MB, BT.C.64

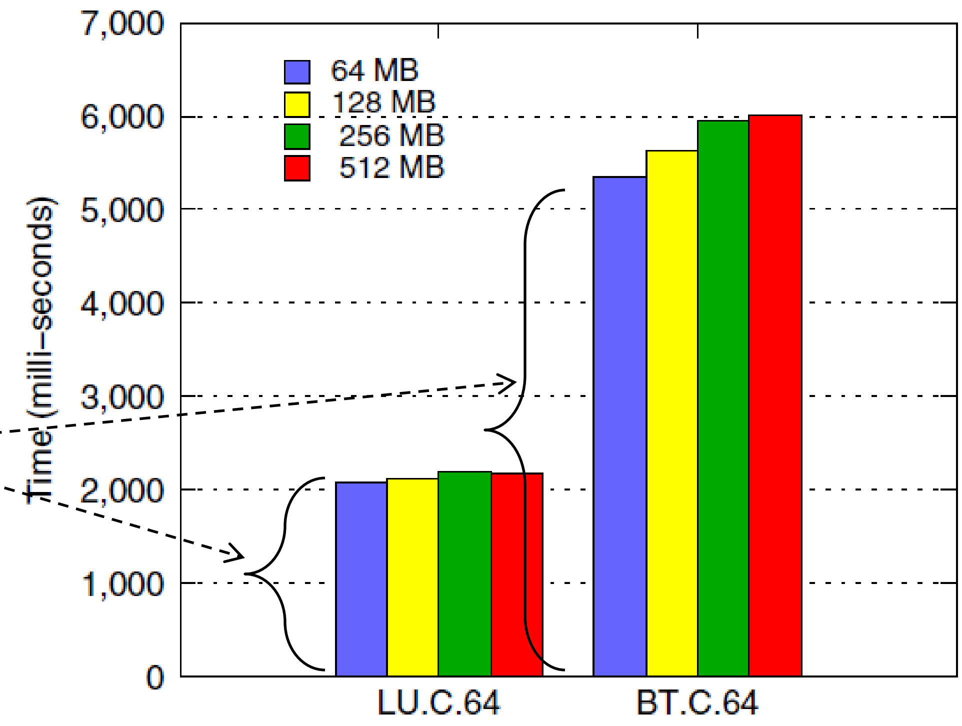


Original BLCR, BT.C.64



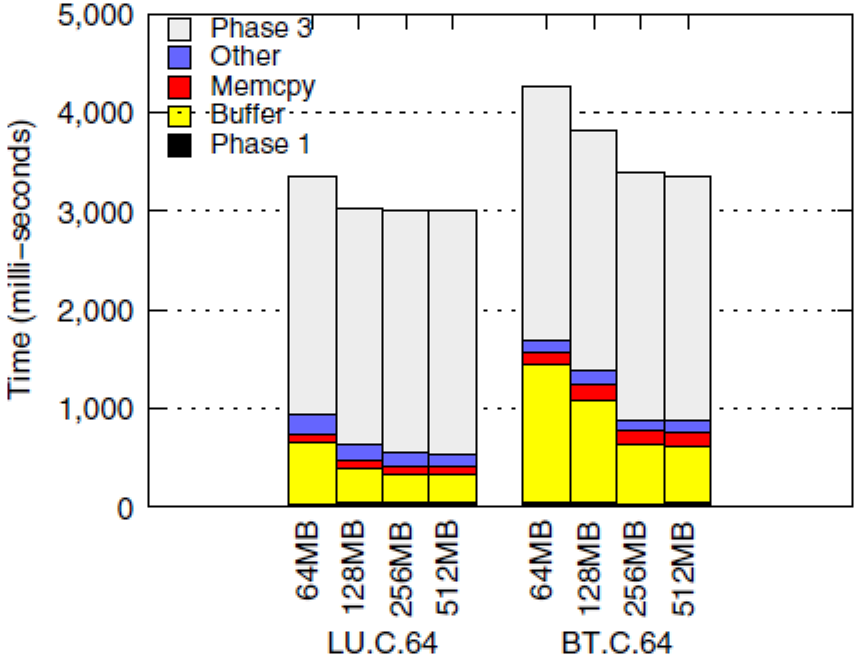
WAG-DBI: Overlapping between Application-progress and File-Write



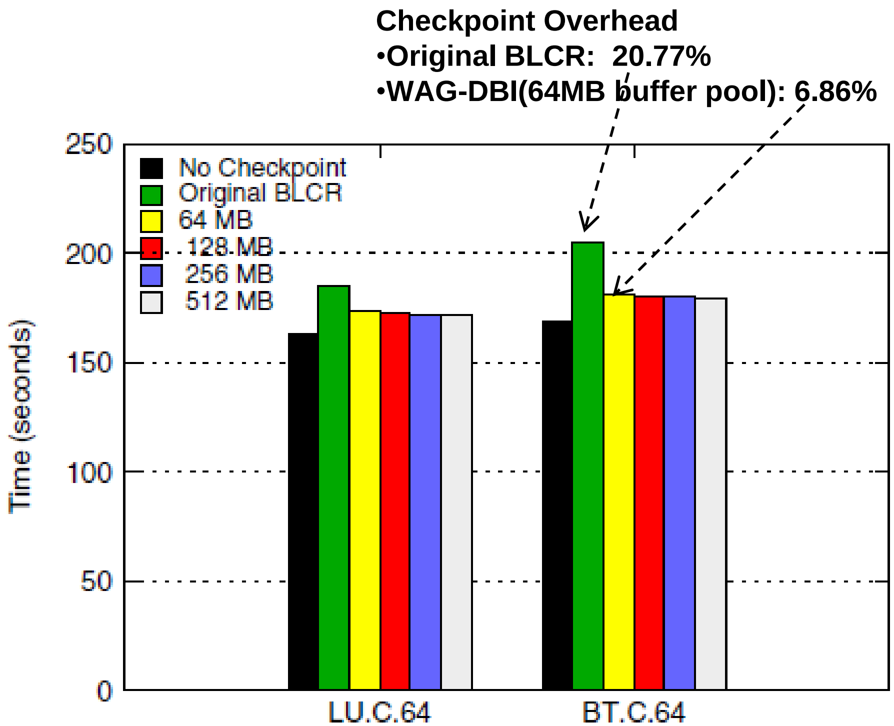


Overlapping between application-progress and file-write (t2 ~ t3)

WAG-DBI Performance



Decomposition of Checkpoint Time

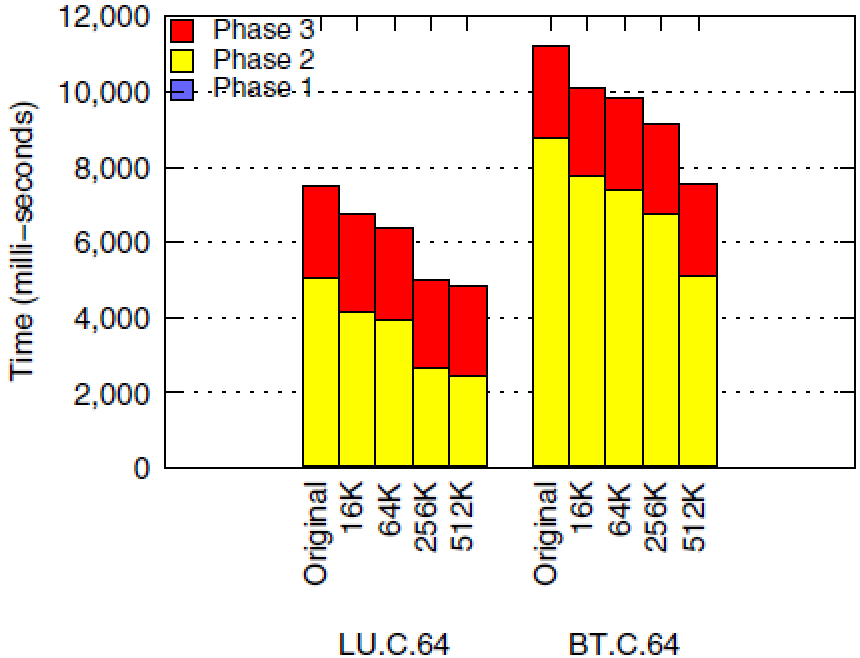


Application Execution Time at Different Buffer Pool Sizes

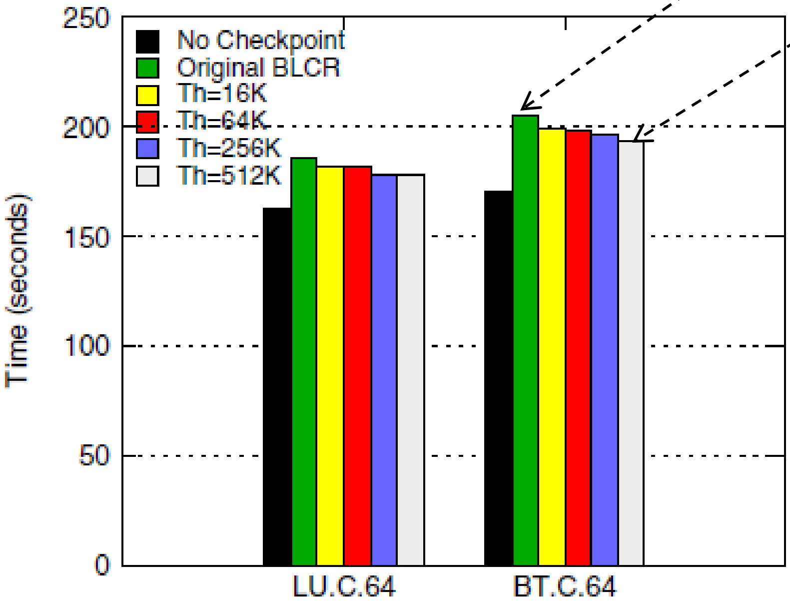
Phase 2 time =
(buffer allocation) +
(memory copy) +
(synchronization, etc)

WAG Performance

Checkpoint Overhead
- **Original BLCR: 20.77%**
- **WAG(threshold=512KB): 9.21%**



Decomposition of checkpoint time



Application Execution Time at different threshold values

Memory Usage per Node(MB) at different threshold values

	16 KB	64 KB	256 KB	512 KB
LU.C.64	42.6	50.0	78.2	80
BT.C.64	33.6	44.8	81.2	160.5

Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Write-Aggregation Design
- Performance Evaluation
- Conclusions and Future Work

Conclusions

- WAG-DBI can improve Checkpoint efficiency in multi-core systems
  - Write Aggregation improves write bandwidth
  - Reduce application execution time by overlapping application progress with checkpoint file write
- WAG-DBI outperforms WAG with less memory usage
  - Aggregate all checkpoint data into a buffer pool
  - Overlap application progress with file IO

Software Distribution

- Current MVAPICH2 1.4 supports basic Checkpoint-Restart
  - Downloadable from http://mvapich.cse.ohio-state.edu/
- The proposed aggregation design will be available in MVAPICH2 1.5

Future Work

- Include the WAG-DBI into a stackable filesystem
- Save checkpoint data to remote data servers
- Usage of emerging Solid State Drive (SSD) to accelerate Checkpoint-Restart

Thank you!

MVAPICH

http://mvapich.cse.ohio-state.edu

{ouyangx, gopalakk, gangadha, panda}
@cse.ohio-state.edu

Network-Based Computing Laboratory