NETWORK-BASED COMPUTING LABORATORY

Enhancing Checkpoint Performance with Staging IO & SSD

Xiangyong Ouyang

Sonya Marcarelli

Dhabaleswar K. Panda

Department of Computer Science & Engineering

The Ohio State University

OHIO STATE

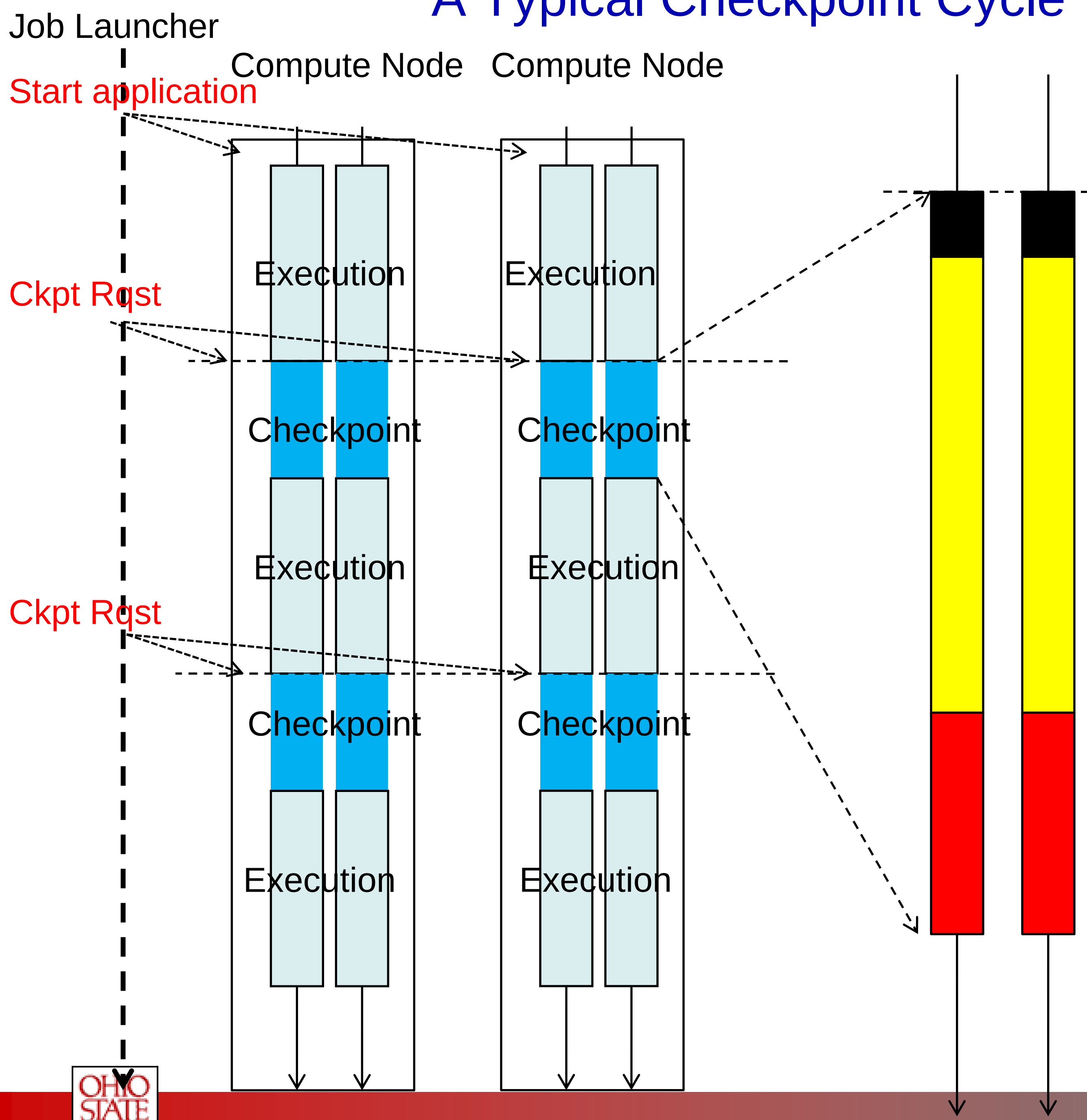
Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Design a High-Performance Parallel Storage for Checkpoint
- 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
- 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 storage system
- High performance storage devices (SSDs) are penetrating into HPC storage
  - High bandwidth, Random-accessibility, Power-efficiency
  - Can it help in a checkpoint storage system?

A Typical Checkpoint Cycle



Phase 1: Coordinate to reach a consistent global state

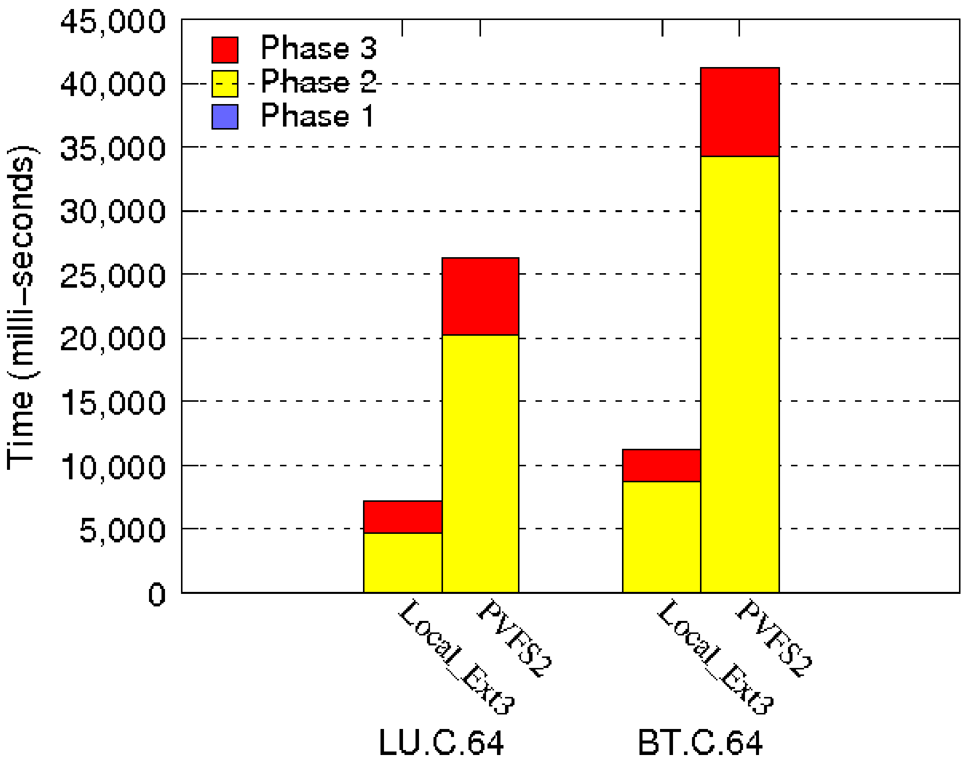
- Drain in-flight messages
- Tear down connections

Phase 2: Use the checkpoint library (BLCR) to checkpoint the individual processes

Phase 3: Re-establish connections between the processes, and continue execution

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
- Previous work on Write-Aggregation to improve ckpt to local file system (ICPP 09, HiPC 09)
- How to improve ckpt to parallel storage system?



[X. Ouyang, K. Gopalakrishnan, T. Gangadharappa and D. K. Panda, Fast Checkpointing by Write Aggregation with Dynamic Buffer and Interleaving on Multicore Architecture, HiPC '09]

[X. Ouyang, K. Gopalakrishnan and D. K. Panda, Accelerating Checkpoint Operation by Node-Level Write Aggregation on Multicore Systems, ICPP '09]

Problem Statement

- What's the typical checkpoint data writing pattern of an MPI application using BLCR?
- How to enhance checkpoint writing performance on Parallel Storage System?
  - Write-Aggregation and Staging I/O
- What are the potentials to apply SSDs into a checkpoint storage system?

Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Design a High-Performance Parallel Storage for Checkpoint
- Performance Evaluation
- Conclusions and Future Work

MVAPICH/MVAPICH2 Software

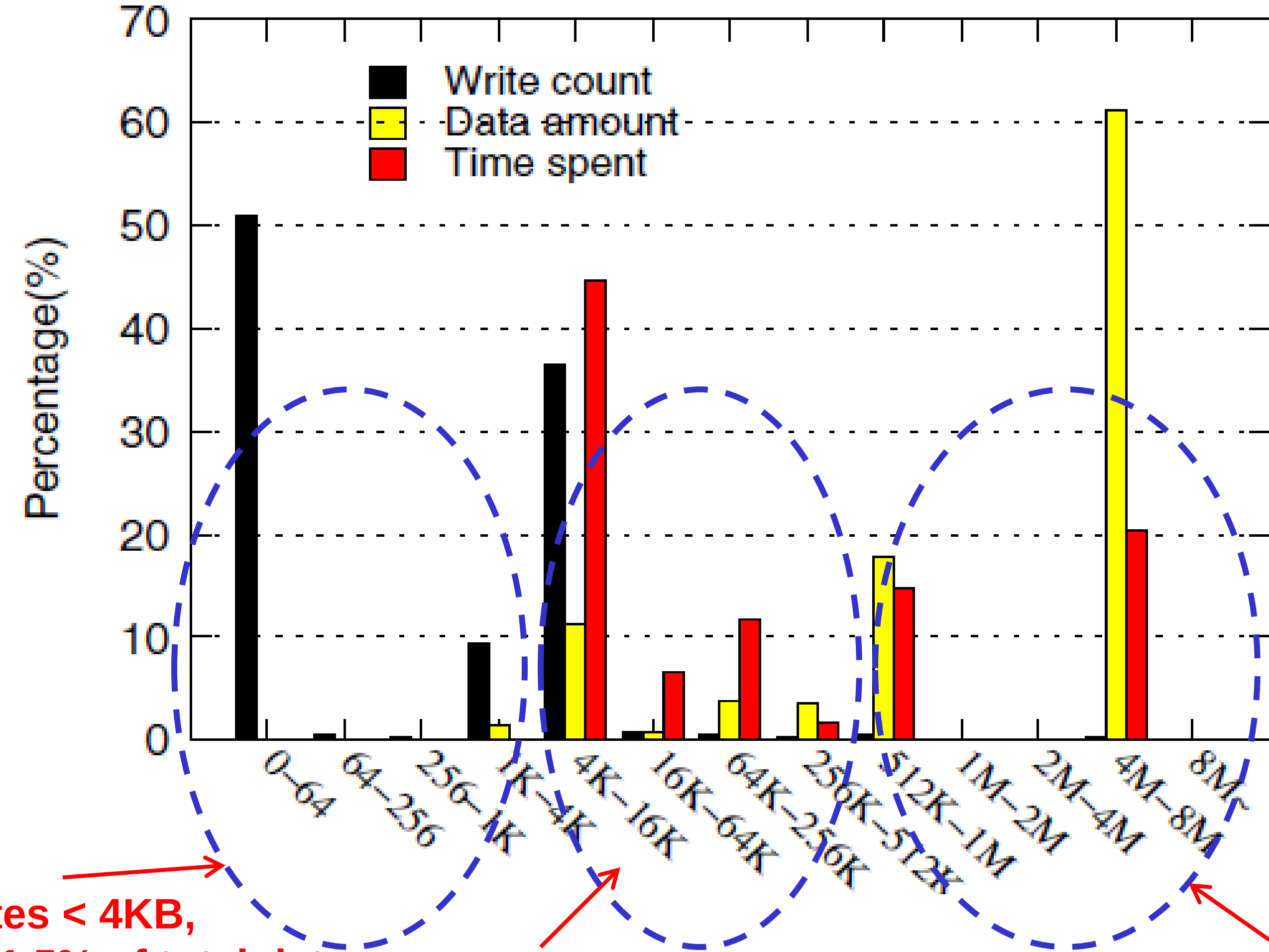
- High Performance MPI Library for InfiniBand, 10GigE/iWARP and RDMAoE
  - MVAPICH (MPI-1) and MVAPICH2 (MPI-2)
  - Used by more than 1,100 organizations in 56 countries
  - More than 39,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - Tianhe-1: 5th 71,680-cores in China (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)
  - Supports system-level Checkpoint/Restart with BLCR(Berkeley Lab's checkpoint/Restart Library)
  - http://mvapich.cse.ohio-state.edu/

Profiling Configuration

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

	LU.C.64	BT.C.64
Checkpoint file size (MB) per process	23.0	40.0
Checkpoint data per node (MB)	184.0	320.0
Total Checkpoint Data (MB)	1472	2560
VFS writes per process	975	1057
Total VFS writes per node	7800	8456

Checkpointing Profiling(LU.C.64): to local ext3



- 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

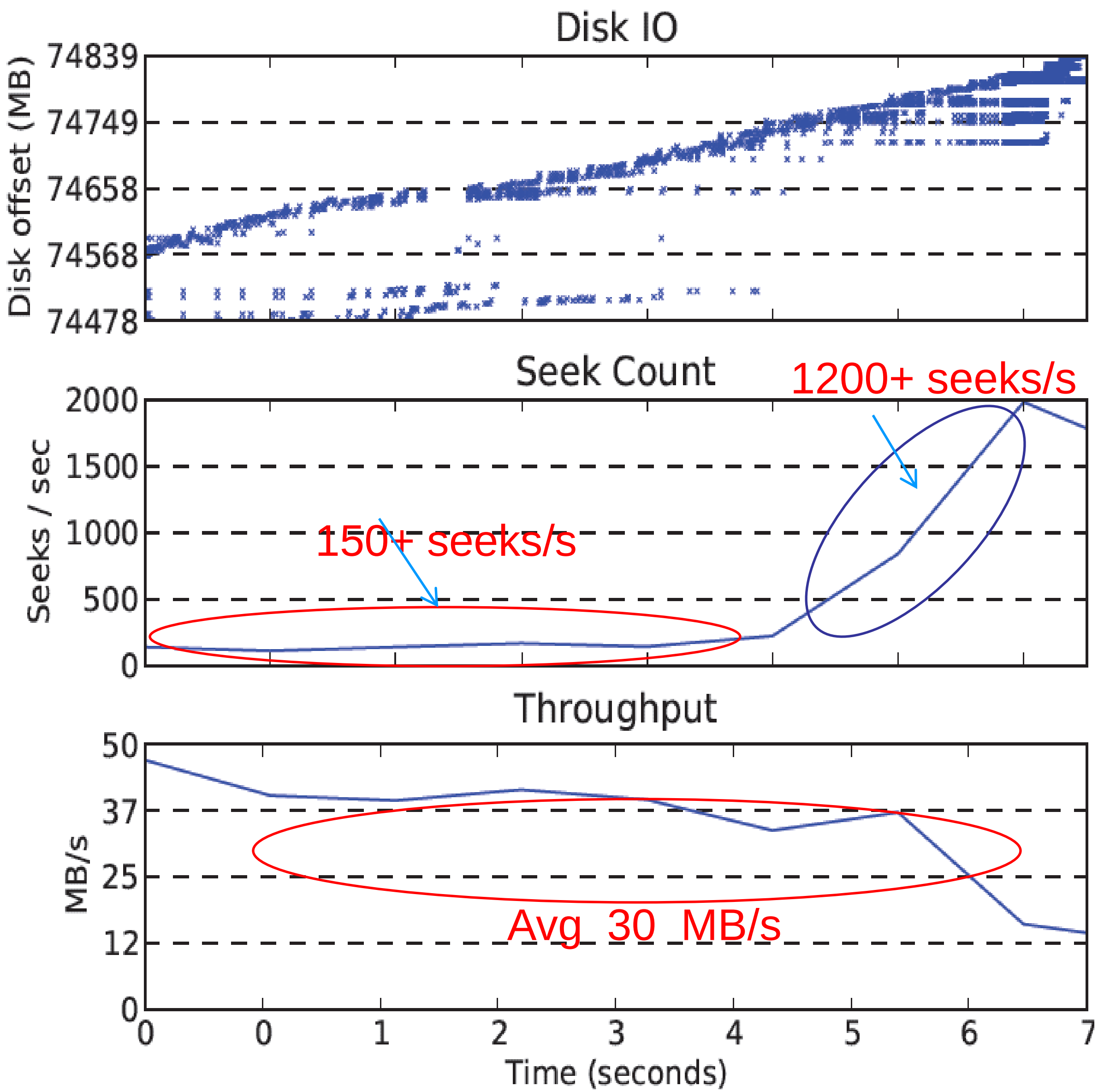
- 0.8% of writes > 512KB
- contribute 79% of all data
- consume 35% of total write time

Checkpointing Profiling (BT.C.64): to local ext3

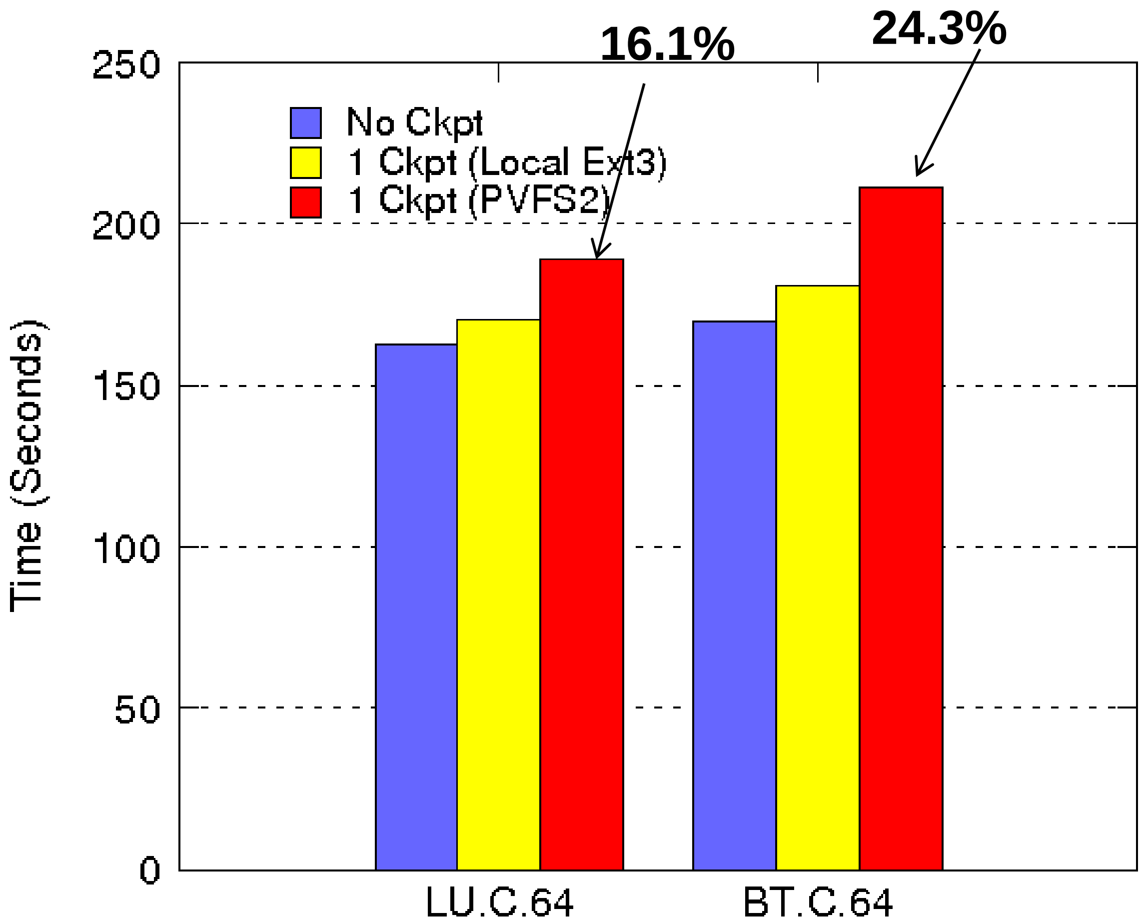
- Use "blktrace" to collect all block layer IO tracing

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

Disk raw bandwidth = 60MB/s



Checkpoint Overhead



Application execution time
w/o checkpoints (ext3 / PVFS 2.8.1)

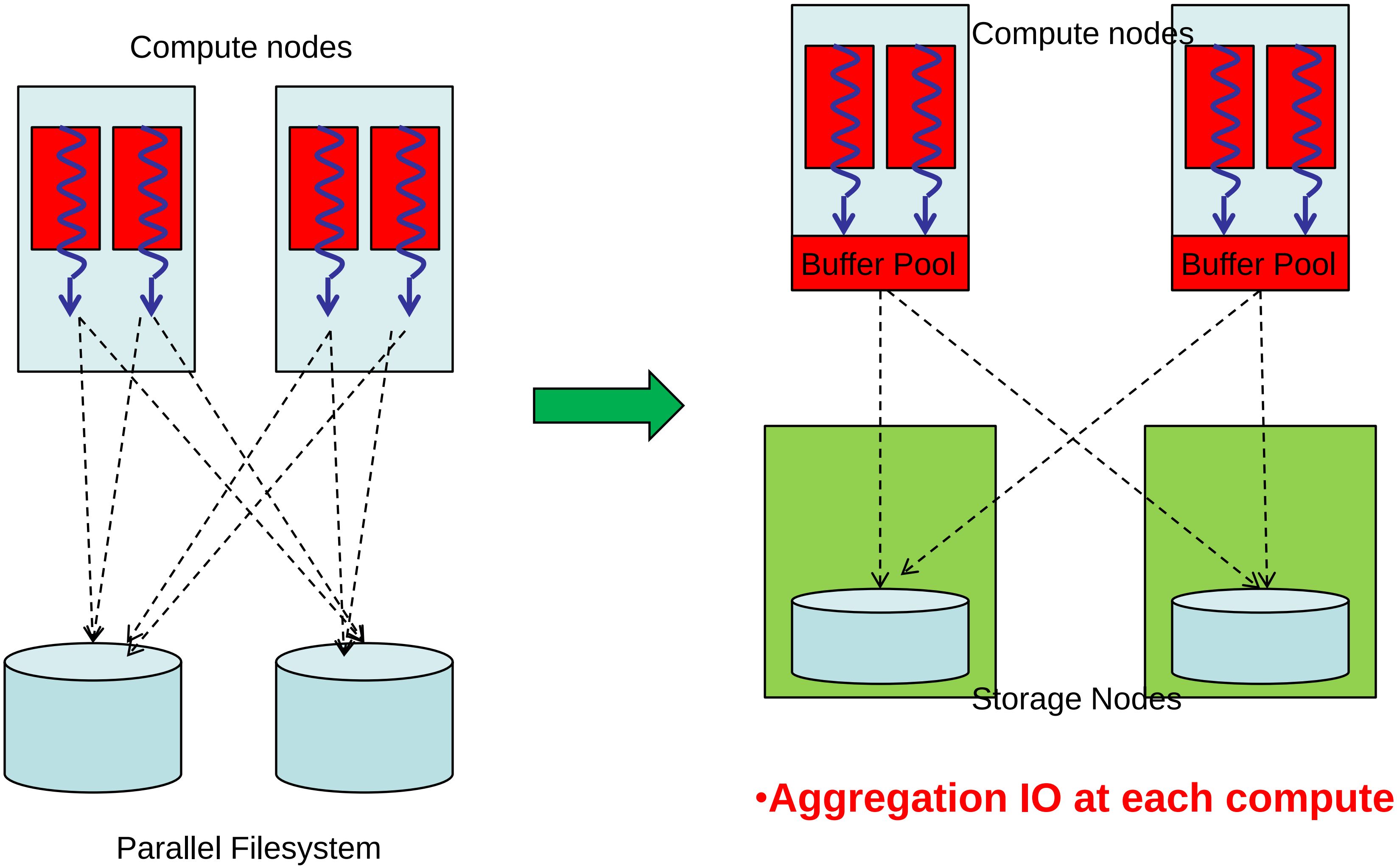
NETWORK-BASED COMPUTING LABORATORY

Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Design a High-Performance Parallel Storage for Checkpoint
- Performance Evaluation
- Conclusions and Future Work

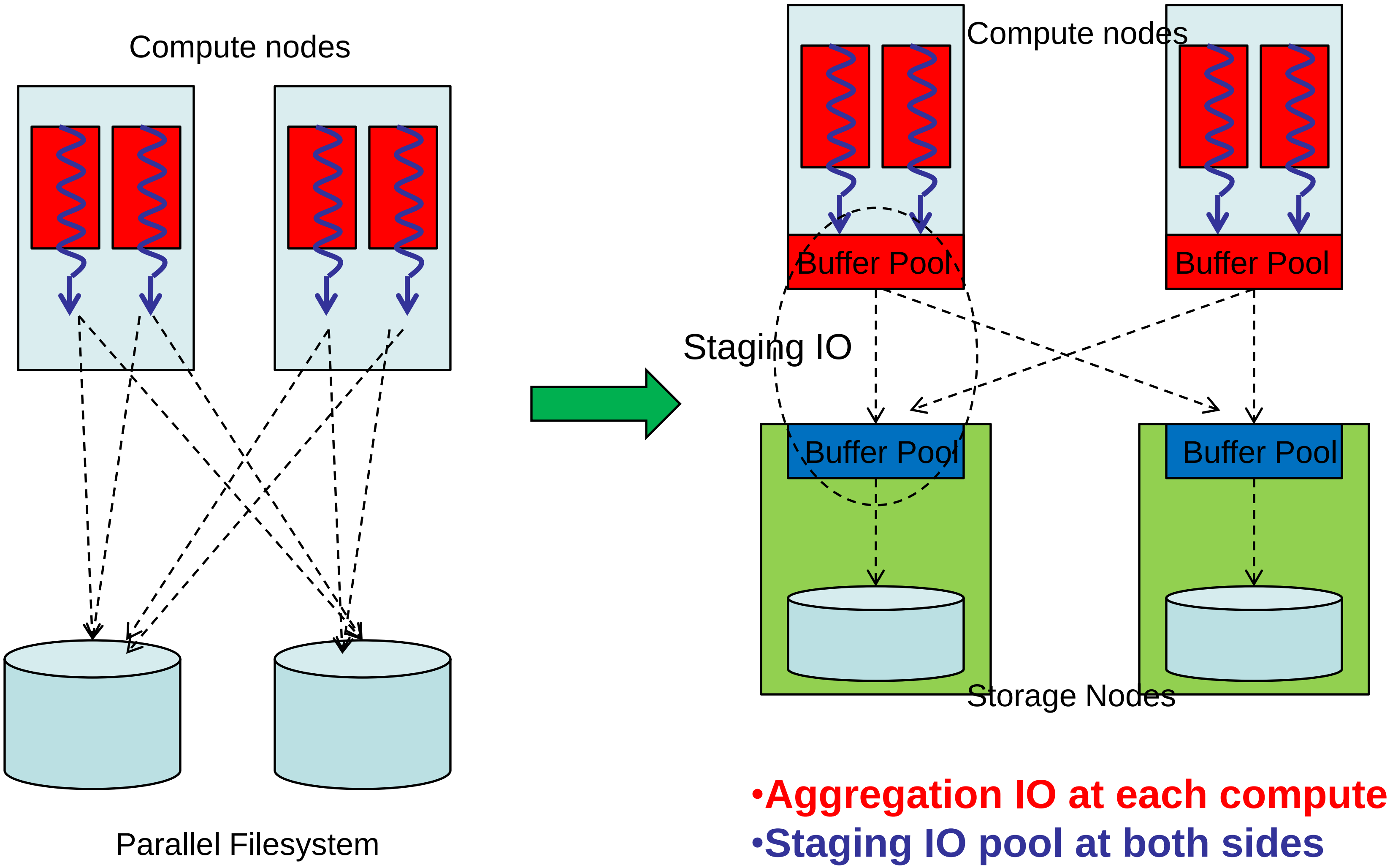
OHIO STATE

Basic Design Strategy (1)



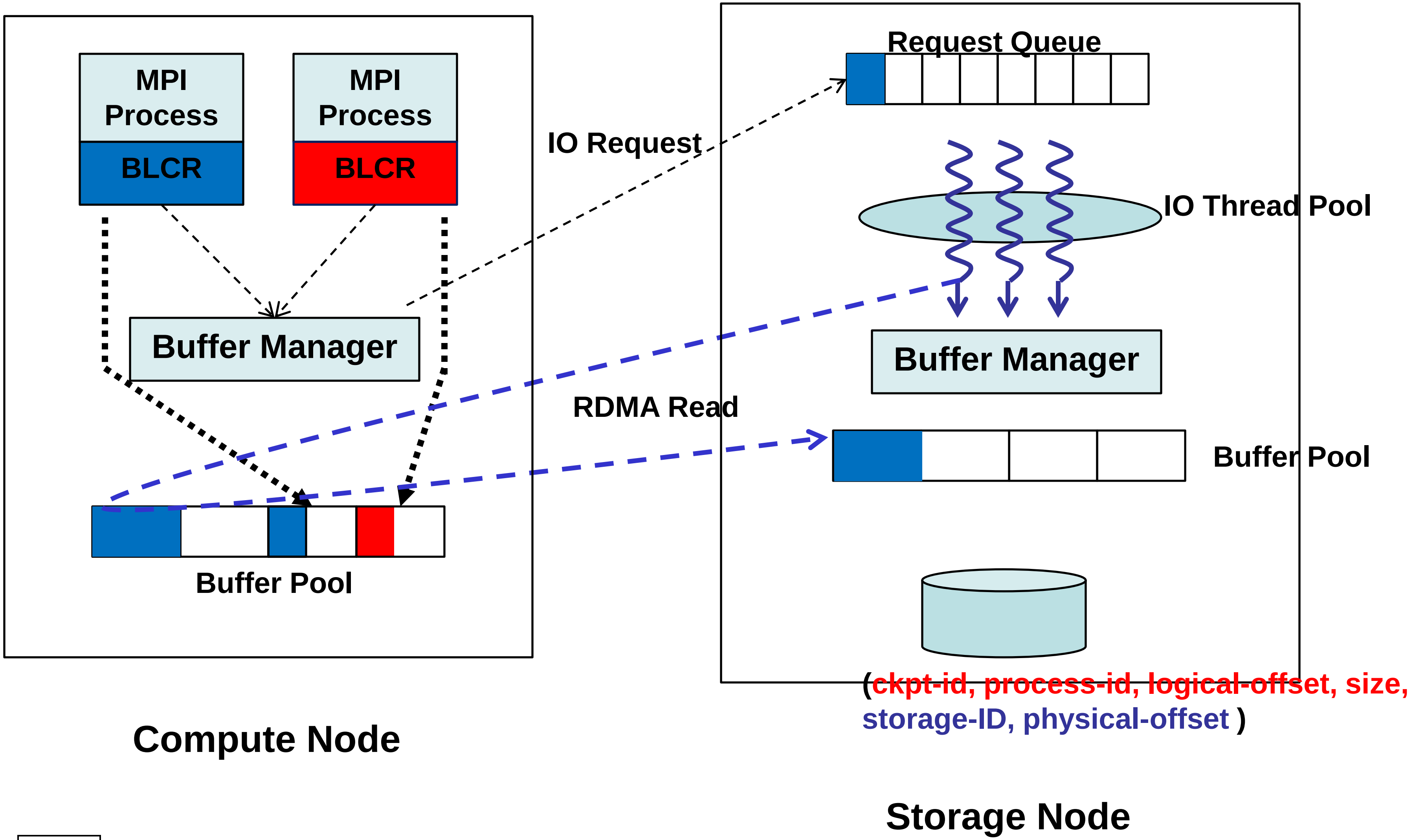
- **Aggregation IO at each compute node**

Basic Design Strategy (2)



- **Aggregation IO at each compute node**
- **Staging IO pool at both sides**
- **Applying SSD at storage nodes**

Enhance Checkpoint Writing with Staging IO



Outline

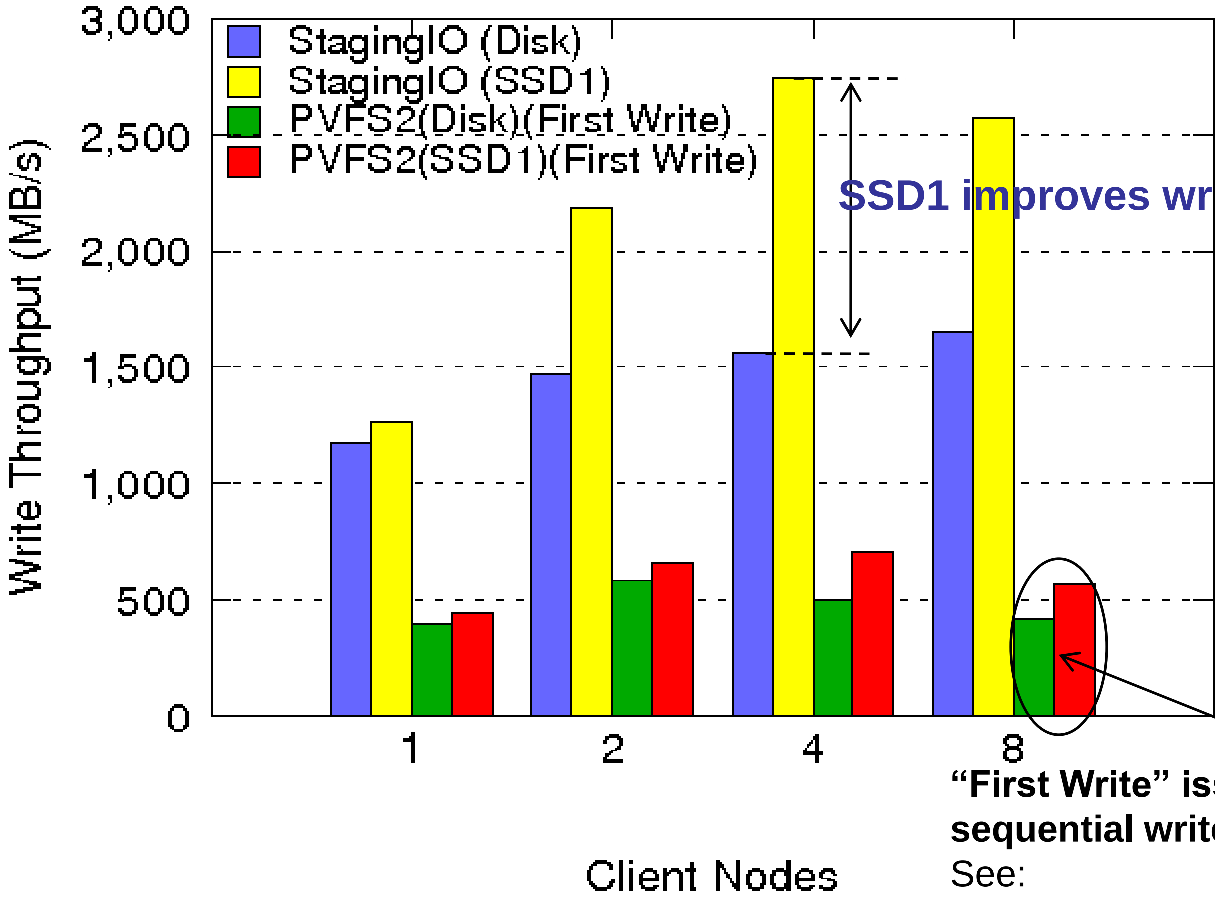
- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Design a High-Performance Parallel Storage for Checkpoint
- Performance Evaluation
- Conclusions and Future Work

Experiments setup

- System setup
  - Intel Clovertown cluster
    - Dual-socket Quad core Xeon processors, 2.33GHz
    - nodes connected by InfiniBand DDR
    - Linux 2.6.18
  - NAS parallel Benchmark suite version 3.2.1
    - LU/BT Class C, 64 processes, 8 processes/node
    - 8 nodes are used
  - MVAPICH2 Checkpoint/Restart framework,
    - BLCR 0.8.0 extended with IO Aggregation
  - Storage Devices

	Write BW(MB/s)	Read BW(MB/s)
Hard Drive (250GB)	55	64
SSD1 (64GB)	179	202
SSD2 (80GB)	600	700

Aggregated Write Bandwidth

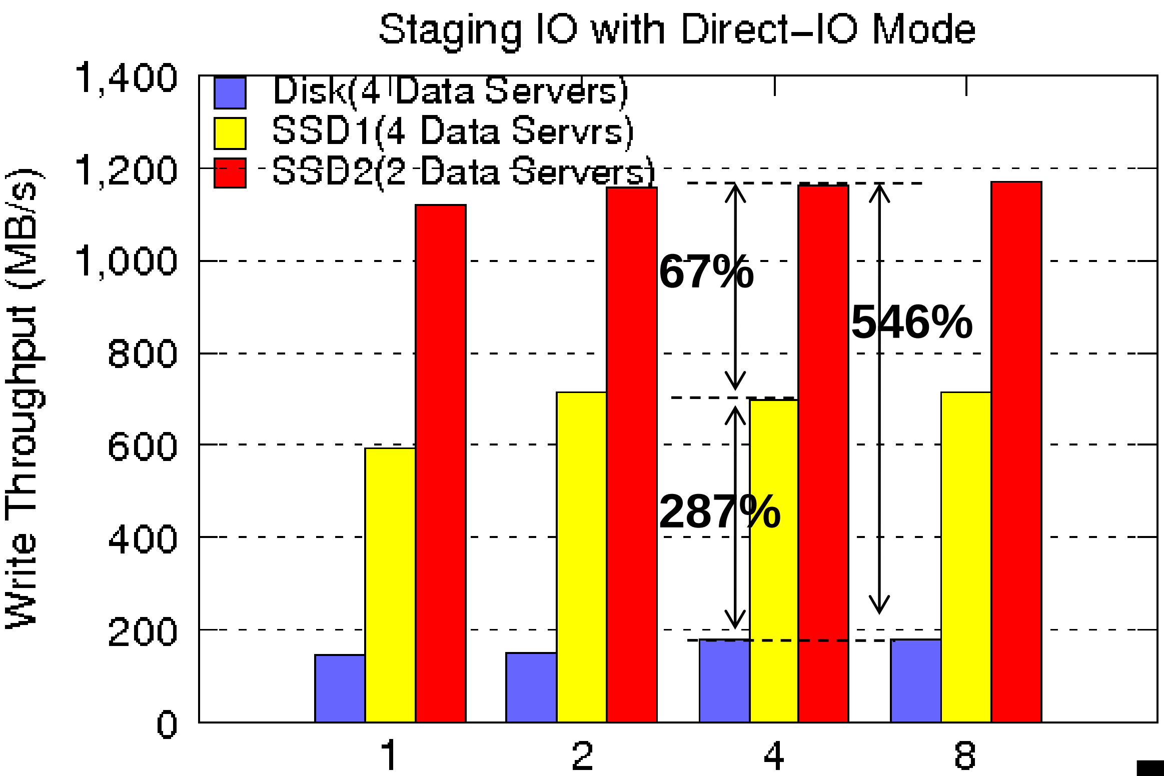


"First Write" issue: file metadata updates affects sequential write.

See: http://www.bowulf-underground.org/pipermail/pvfs2-users/2009-April/002770.htmle

- **Staging IO:**
  4 Storage Nodes, Buffer-pool=64MB, chunk=4MB
- **PVFS 2.8.1**
  4 DS, stripe=1MB, bmi_mod=IB

Aggregated Write Bandwidth (Direct-IO)

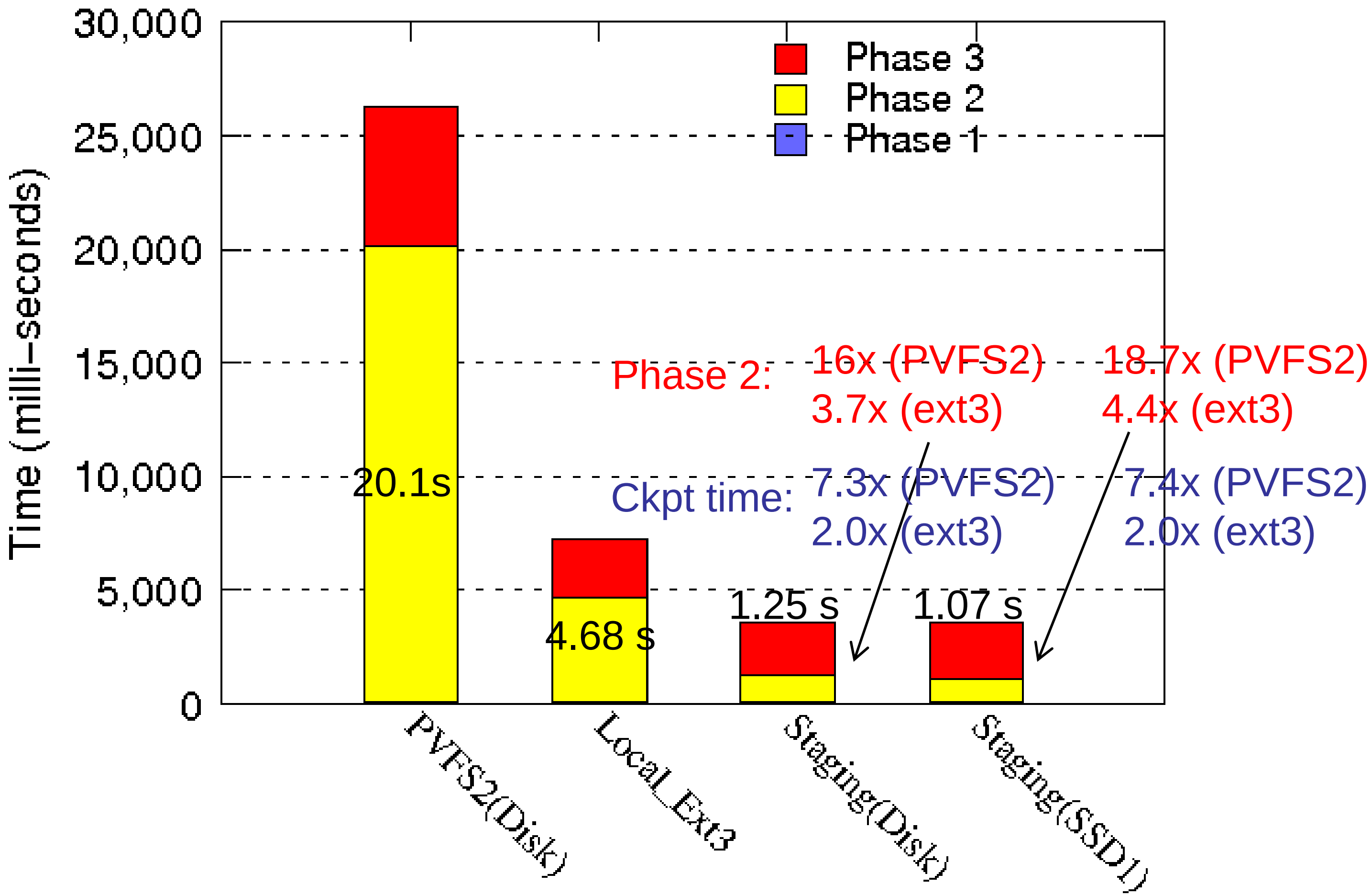


SSD2: 97% of raw bw
SSD1: 97% of raw bw

	Write BW(MB/s)	Read BW(MB/s)
Hard Drive (250GB)	**55**	64
SSD1 (64GB)	**179**	202
SSD2 (80GB)	**600**	700

Disk: 4 storage nodes
SSD1: 4 storage nodes
SSD2: 2 storage nodes
- **Buffer-pool=64MB, chunk=4MB**

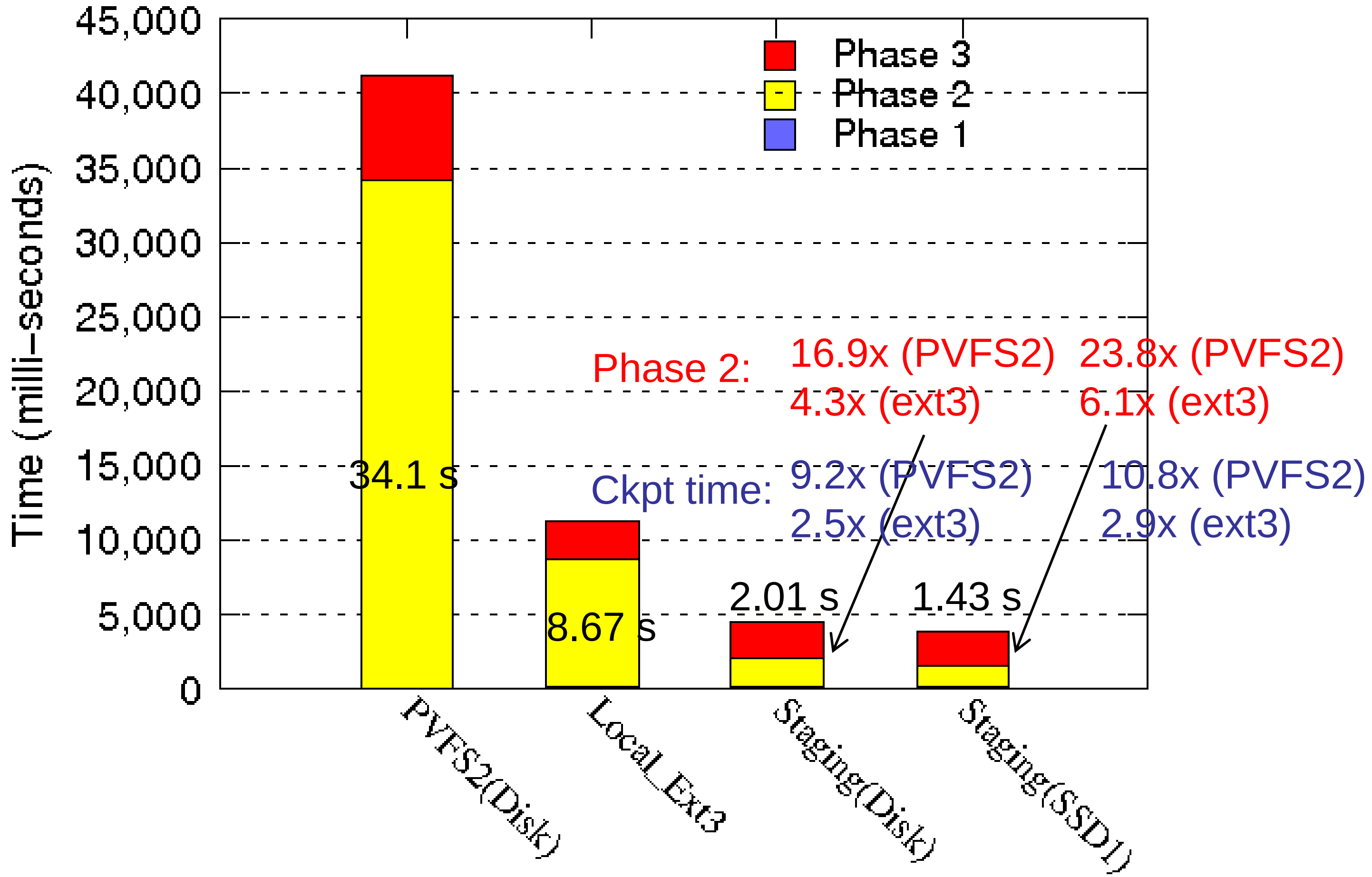
Checkpoint Time: LU.C.64 (8 client nodes)



- **PVFS2**
  - **4 DS, stripe=1MB, bmi_mod=IB**
- **Staging IO:**
  - **4 Storage Nodes, Buffer-pool=64MB, chunk=4MB**

	LU.C.64
Total Checkpoint Data (MB)	1472
VFS writes per node	7800

Checkpoint Time: BT.C.64 (8 client nodes)



- **PVFS2**
  - **4 DS, stripe=1MB, bmi_mod=IB**
- **Staging IO:**
  - **4 Storage Nodes, Buffer-pool=64MB, chunk=4MB**

	BT.C.64
Total Checkpoint Data (MB)	2560
VFS writes per node	8456

Outline

- Motivation and Introduction
- Checkpoint Profiling and Analysis
- Design a High-Performance Parallel Storage for Checkpoint
- Performance Evaluation
- Conclusions and Future Work

Conclusions

- Staging IO significantly improves Checkpoint Writing performance to parallel storage system
  - IO Aggregation improves write bandwidth at client-side
  - Staging IO reduces contentions at storage nodes
- SSD can boost aggregated IO throughput in parallel storage systems

Future Work

- Staging IO for Read
- Integrate the IO Aggregation and Staging IO into a stackable filesystem
- Apply Staging IO to Process-Migration design

Software Distribution

- Current MVAPICH2 1.4 supports basic Checkpoint-Restart
  - Downloadable from http://mvapich.cse.ohio-state.edu/

- The proposed Staging IO design will be available in upcoming MVAPICH2 releases

Thank you!

MVAPICH

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

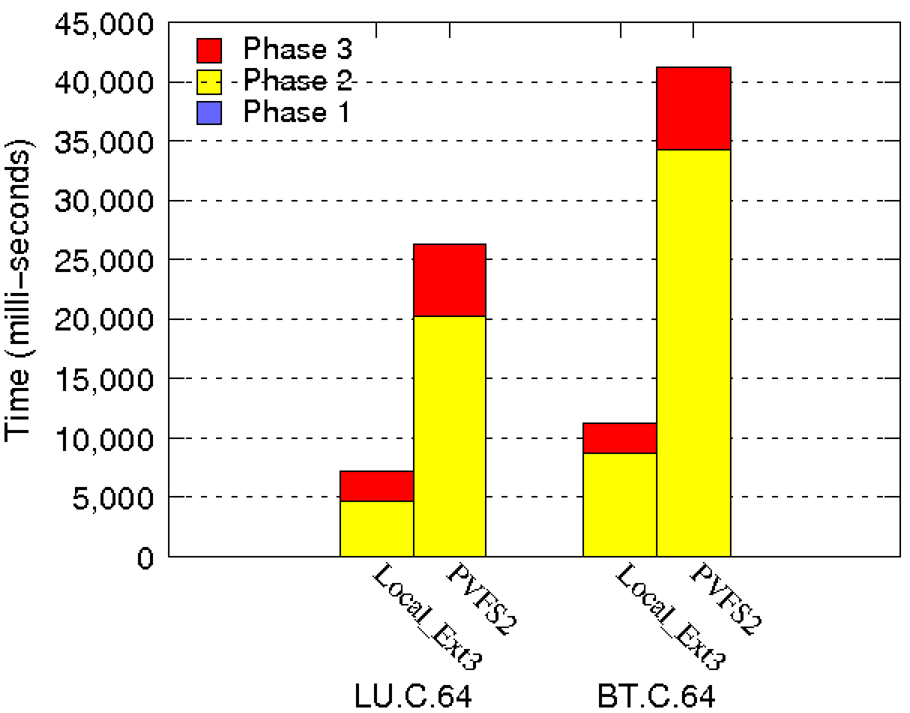
{ouyangx, smarcare, panda}
@cse.ohio-state.edu

Network-Based Computing Laboratory

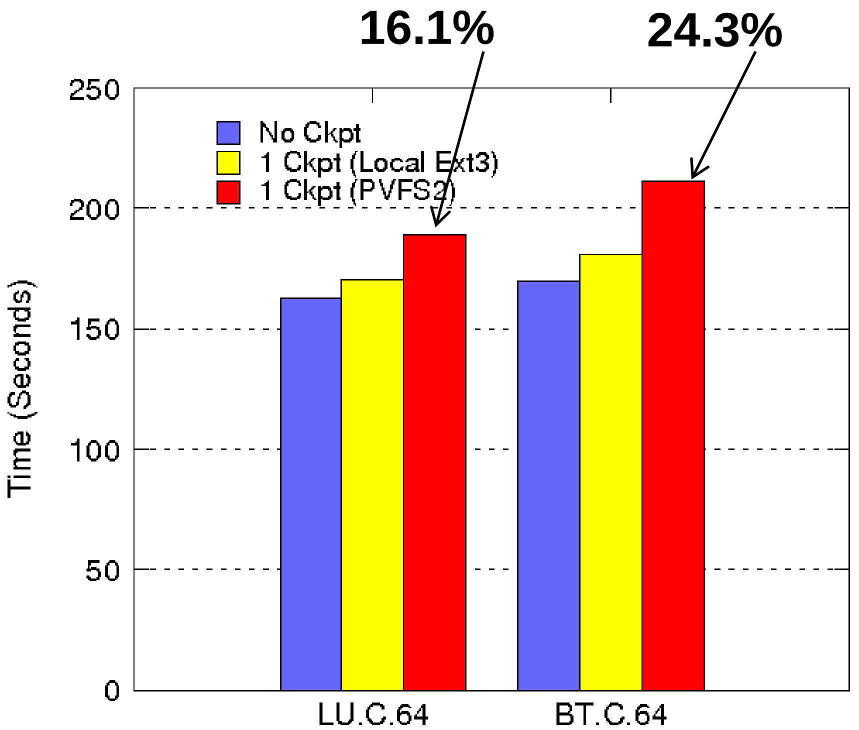
Reconstruct Checkpoint Files

- The storage node maintains metadata for each buffer-chunk
  - (ckpt-id, Process-id, logical-offset, size, storage-node-ID, physical-offset)
- Compute node reconstructs checkpoint files during restart
  - Collect metadata from all Storage Nodes
  - Request data-chunks from storage nodes
    - Given (Storage-Node-ID, Physical-offset, size)
  - Concatenate all chunks belonging to a process into one file
    - All chunks with same (ckpt-id, process-id)

Checkpoint Overhead



Decomposition of Checkpoint Time



Application execution time w/o checkpoints