NETWORK-BASED COMPUTING LABORATORY

# Accelerating Checkpoint Operation by Node-Level Write Aggregation on Multicore Systems

Xiangyong Ouyang, Karthik Gopalakrishnan and Dhabaleswar K. (DK) Panda

Department of Computer Science & Engineering

The Ohio State University

OHIO STATE

# 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
  - 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
  - A low utilization of the raw throughput of the underlying file system

# Checkpointing a Parallel MPI Application

- Berkeley Lab's Checkpoint/Restart (BLCR) solution is used by many MPI implementations
  - MVAPICH2, OpenMPI, LAM/MPI
- Checkpointing a parallel MPI job includes 3 phases
  - 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

# Phase 2 of Checkpoint Restart

- 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
- File system performance depends on data I/O pattern
  - Writing one large chunk is more efficient than multiple writes of smaller size

# Problem Statement

- What's the checkpoint data writing pattern of a typical MPI application using BLCR?
- Can we optimize the data writing path to increase 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

# MVAPICH/MVAPICH2 Software

- High Performance MPI Library for InfiniBand and 10GE
  - MVAPICH (MPI-1) and MVAPICH2 (MPI-2)
  - Used by more than 975 organizations in 51 countries
  - More than 32,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 8th ranked 62,976-core cluster (Ranger) at TACC
  - Available with software stacks of many IB, 10GE and server vendors including Open Fabrics Enterprise Distribution (OFED)
  - http://mvapich.cse.ohio-state.edu/

# Initial Profiling

- MVAPICH2 Checkpoint/Restart framework
  - BLCR was 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
- 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

# Profiled Results

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

| | LU | BT | SP | CG |
|---|---|---|---|---|
| Time for one check-point(seconds) | 7.6 | 11.3 | 10.3 | 7.1 |
| Total data size(MB) per node | 184.0 | 320.0 | 316.0 | 163.2 |
| Number of VFS write per process | 975 | 1057 | 1367 | 820 |
| Total number of VFS writes per node | 7800 | 8456 | 10936 | 6560 |

# Sizes of File Write Operations

- The profiling revealed some characteristics of checkpoint writing
  - Most of file writes are associated with small data size
    - 60% of writes < 4KB, contribute 1.5% of total data, consume 0.2% of total write time
  - A few large writes
    - 0.8% of writes > 512KB, contribute 79% of all data, consume 35% of total write time
  - Some medium writes in between
    - 38% of all writes, contribute 20% of all data, consume 65 % of all time

# Checkpoint Writing Profile for LU.C.64

| | % of Writes | % of Data | % of Time |
|---|---|---|---|
| 0-64 | 50.86 | 0.04 | 0.17 |
| 64-256 | 0.61 | 0.00 | 0.00 |
| 256-1K | 0.25 | 0.01 | 0.00 |
| 1K-4K | 9.46 | 1.53 | 0.01 |
| 4K-16K | 36.49 | 11.36 | 44.66 |
| 16K-64K | 0.74 | 0.77 | 6.55 |
| 64K-256K | 0.49 | 3.79 | 11.80 |
| 256K-512K | 0.25 | 3.58 | 1.75 |
| 512K-1M | 0.61 | 17.72 | 14.72 |
| $>$ 1M | 0.25 | 61.21 | 20.35 |

# Outline

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

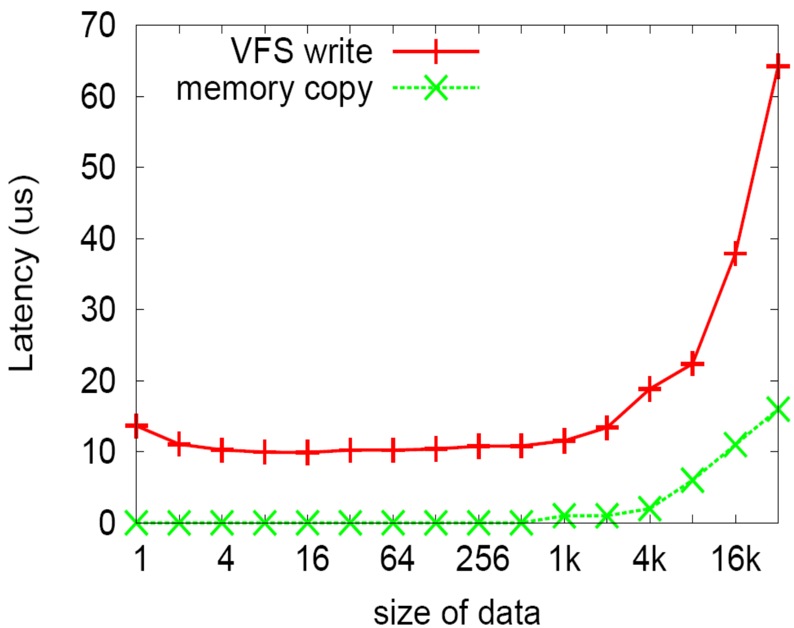
# Methodology

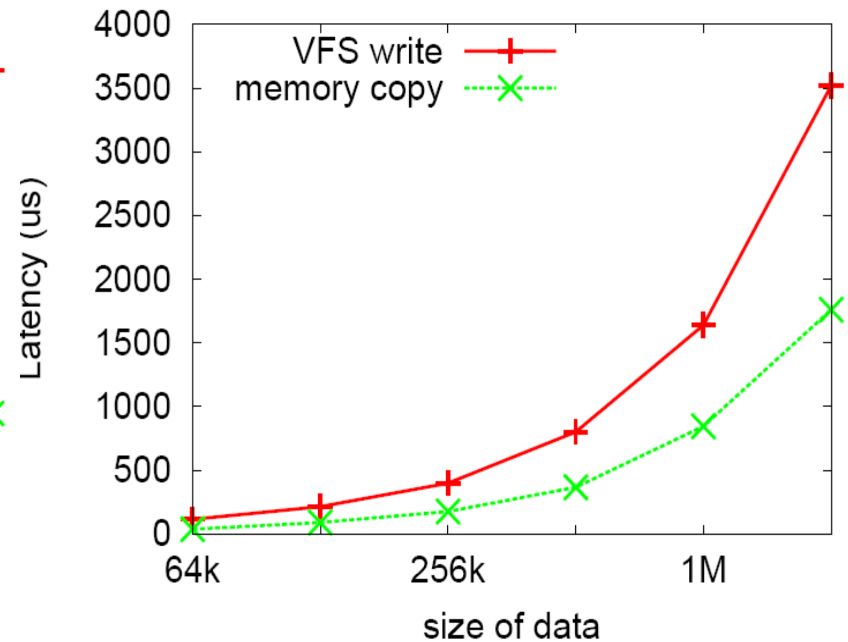
- Classify checkpoint writes into 3 categories
- Small writes
  - Frequent calls of vfs_write() with small size cause heavy overhead
  - Solution: Aggregate small writes in a local buffer
- Large writes
  - Memory copy cost becomes close to file write cost
  - Has to consider memory usage
  - Solution: Flush large writes directly to checkpoint files
- Medium writes
  - Depends on memory-copy cost vs. file write cost
  - Solution: Search a threshold
    - Size <= threshold: Aggregate in local buffer
    - Size > threshold: Flush directly to checkpoint files

# Memory-copy vs. File write

- Without aggregation, checkpoint data write overhead comes from
  - Vfs_write to move data to page cache
  - Move data from page cache to storage device
- With aggregation, checkpoint data write overhead comes from
  - Memory copy to local buffer
  - Vfs_write to move data from local buffer to page cache
  - Move data from page cache to storage device

# Memory-copy vs. File write Performance




- Memory-copy cost very low at small size
- Memory-copy cost becomes close to vfs_write at certain size
- A threshold should be determined by
    - Relative cost
    - Total Memory usage

# Write-Aggregation Scheme

- Each node has one IO process (IOP), many application processes (AP)
- Each AP has a local buffer (for small writes aggregation)
- A large buffer shared by all APs (for medium writes aggregation)

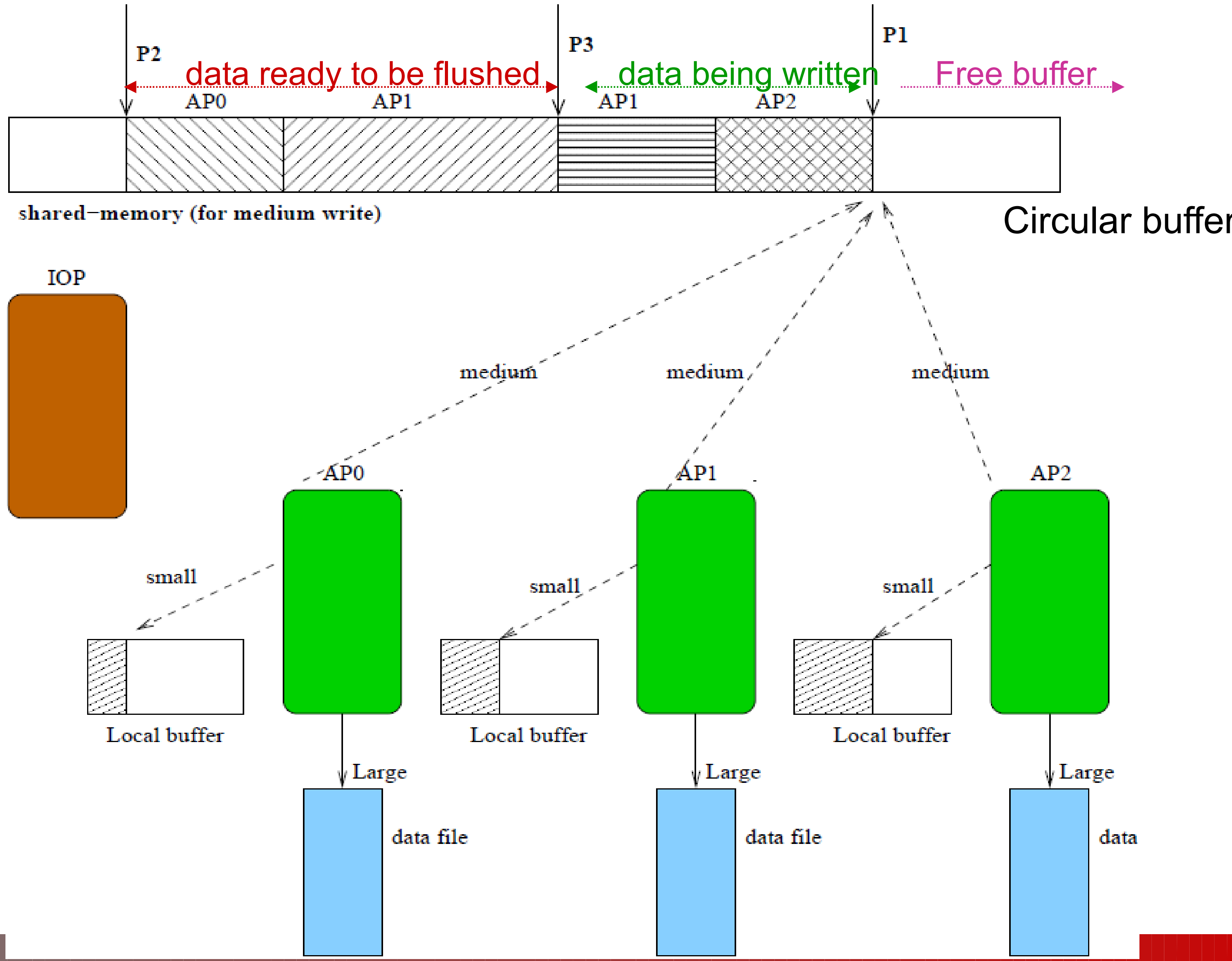
# Write-Aggregation Scheme

- Small writes (< 512B)
  - AP puts it to local buffer
- Medium writes (< threshold )
  - AP grabs a free chunk from shared buffer, copy to the chunk
- All writes >= threshold
  - AP directly flushes it to checkpoint file

- IOP periodically flushes data in shared buffer to a data file

- Experiment indicates 64KB to be a good threshold for current generation platforms

# Write-Aggregation Design



# Restart

- Each write is encapsulated into a chunk

| | | | |
|---|---|---|---|
| **Process Rank** | **Data size** | **Original Offset** | **Data** |

- At restart,
  - Unpack data from the data files
  - Rebuild checkpoint file for each AP
  - AP calls BLCR library to restart
- Restarts are infrequent, thus slight overhead is OK

# 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 Write-Aggregation Design

# Time Cost Decomposition into 3 Phases

| | Phase 1 | Phase 2 | Phase 3 | Improvement in phase 2 (%) |
|---|---|---|---|---|
| LU-orig | 33 | 5418 | 2150 | |
| LU th=16K | 59 | 4612 | 2169 | 14.88 |
| LU th=64K | 67 | 4389 | 2132 | 18.99 |
| LU th=256K | 74 | 3474 | 2047 | 35.88 |
| LU th=512K | 64 | 3081 | 2115 | 43.13 |
| BT-orig | 34 | 9136 | 2141 | |
| BT th=16K | 34 | 8142 | 2034 | 10.88 |
| BT th=64K | 48 | 7725 | 2159 | 15.44 |
| BT th=256K | 48 | 7084 | 2137 | 22.46 |
| BT th=512K | 34 | 5463 | 2142 | 40.20 |
| CG-orig | 40 | 4987 | 2103 | |
| CG th=16K | 42 | 4344 | 2073 | 12.89 |
| CG th=64K | 43 | 4055 | 2026 | 18.69 |
| CG th=256K | 44 | 3178 | 2124 | 36.27 |
| CG th=512K | 45 | 2959 | 2168 | 40.67 |

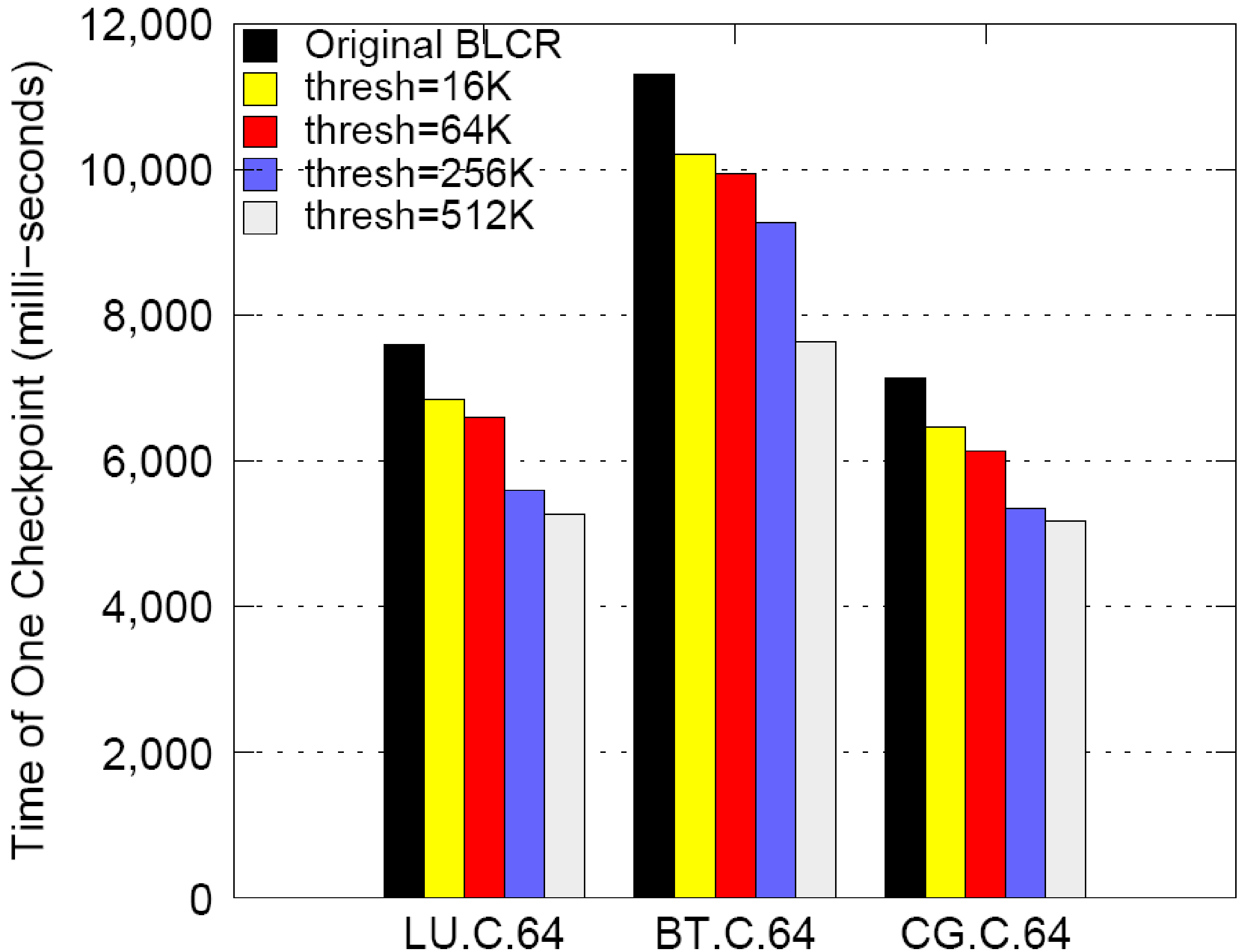
- Phase 1: Suspend communication
- Phase 2: Checkpoint individual process
- Phase 3: Re-establish connections

(Time in milli-seconds)

# Overall Checkpoint Time with Write-Aggregation

At Threshold=16K,64K, 256K,512K, reductions of checkpoint time are:

- LU.C.64: 10.0%, 13.3%, 26.4%, 30.8%
- BT.C.64: 9.7%, 12.2%, 18.0%, 32.5%
- CG.C.64: 9.4%, 14.1%, 25.0%, 27.5%



# Memory Usage at Different Threshold

Memory Usage in MB

| | 16 KB | 64 KB | 256 KB | 512 KB |
|---|---|---|---|---|
| LU.C.64 | 42.6 | 50.0 | 78.2 | 81.1 |
| BT.C.64 | 33.6 | 44.8 | 81.2 | 160.5 |
| CG.C.64 | 39.2 | 48.8 | 64.8 | 76.0 |

# 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

# Outline

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

# Conclusions

- Write-Aggregation can improve Checkpoint efficiency in multi-core systems
  - Significantly reduces the cost of checkpoint write
- Improvement depends on varied threshold values
  - Larger threshold yields better improvements, but requires extra amount of memory usage

# Future Work

- Larger scale test on different multi-core platforms
  - Study the effectiveness of Write-Aggregation on platforms with 16/24-cores
  - Search the optimal threshold values at given buffer size, with different memory bandwidth
- Inter-node Write Aggregation
- Usage of emerging Solid State Drive (SSD) to accelerate Checkpoint-Restart

# Thank you !

MVAPICH

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

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

Network-Based Computing Laboratory