

RDMA-Based Job Migration Framework for MPI over InfiniBand

Xiangyong Ouyang

Sonya Marcarelli

Raghunath Rajachandrasekar

Dhabaleswar K. (DK) Panda

Department of Computer Science & Engineering

The Ohio State University

Outline

- **Introduction and Motivation**
- Job Migration Framework for MVAPICH2
- RDMA-based Process Migration
- Performance Evaluation
- Conclusions and Future Work

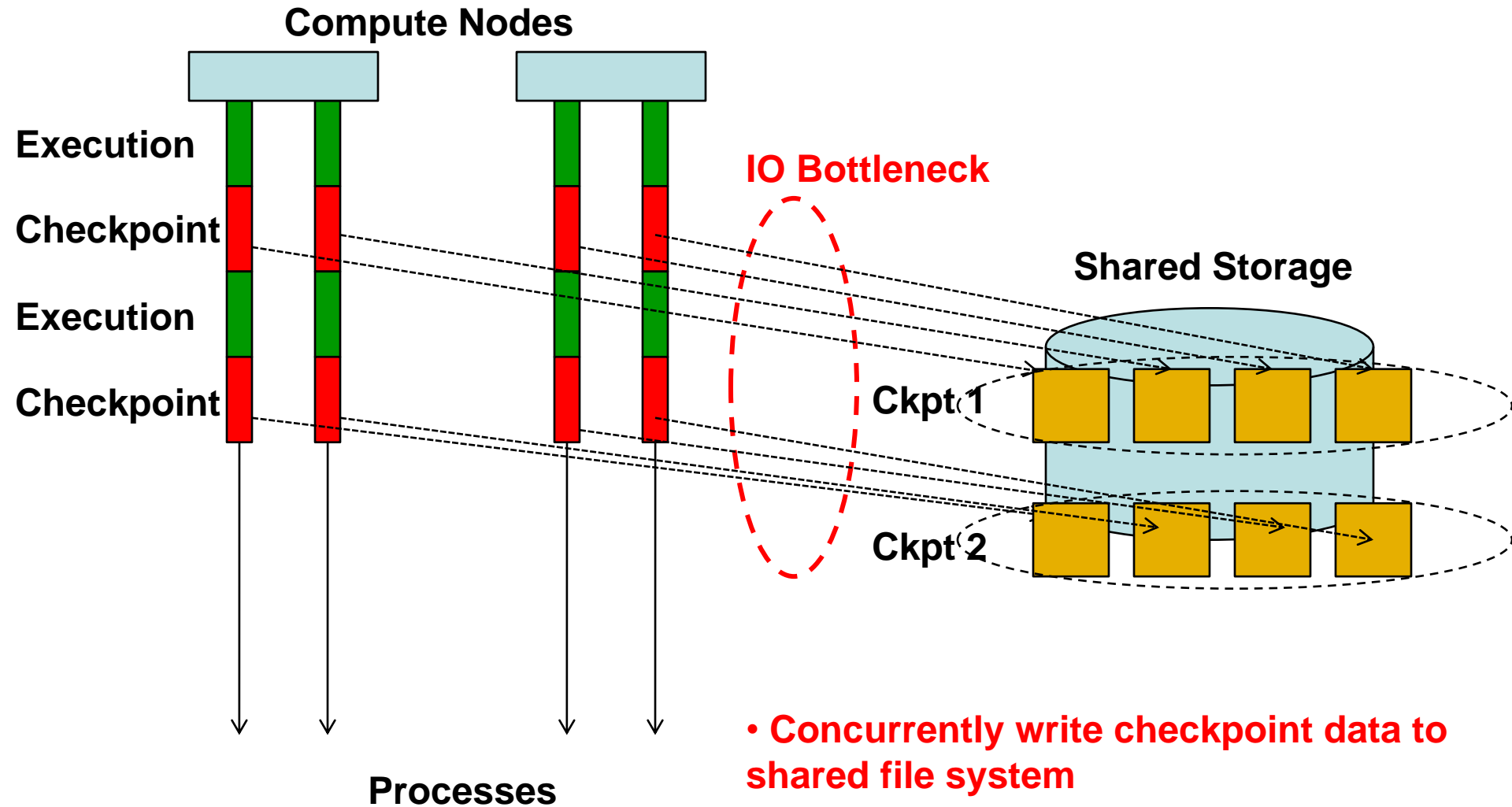
Motivation

- Multi-core architectures are gaining momentum
 - Multiple processes on a same node
- Computer clusters continue to grow larger
 - 100K cores
 - Heading towards Multi-PetaFlop and ExaFlop Era
 - Mean-time-between-failures (MTBF) is getting smaller
 - Fault-Tolerance becomes imperative
- Checkpoint/Restart – common approach to Fault Tolerance
 - Doesn't work well in large scale systems
- Current job schedulers
 - Insufficient support for process management under fragmentation
 - Can not dynamically relocate processes efficiently

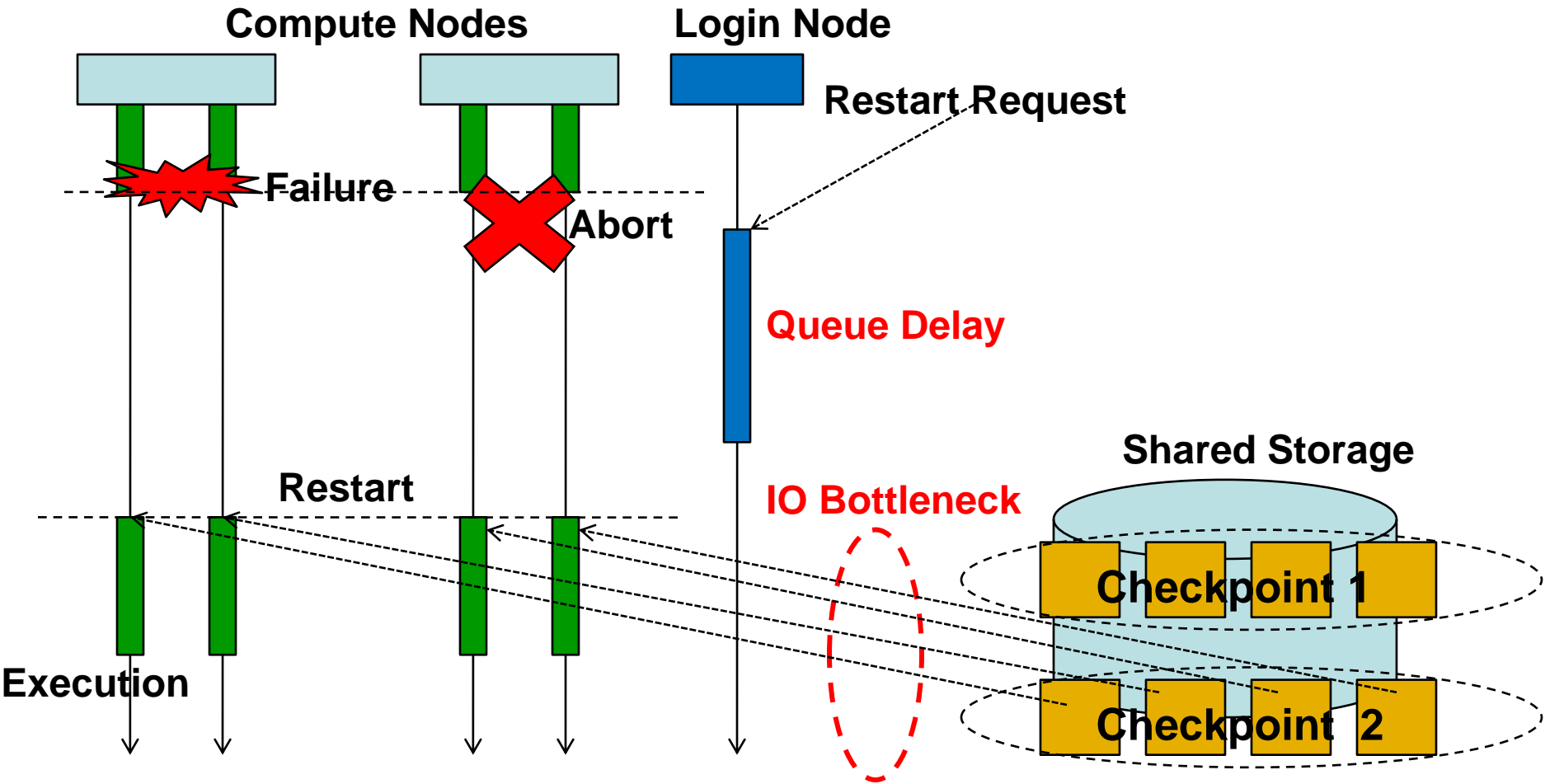
Problems with Checkpoint/Restart

- Coordinated Checkpoint/Restart
 - Checkpoint
 - Every process saves its snapshot to stable storage
 - Restart
 - After failure, restarts from the last checkpoint
- Problems
 - **During Checkpoint:** Concurrent writes to shared storage → Heavy IO bottleneck
 - **During Restart:** Resubmit the job → Long queue latency

Current C/R solution: Checkpoint



Current C/R solution: Restart



- Long queue latency
- Heavy IO overhead to load checkpoint files concurrently

Dynamic Process Relocation

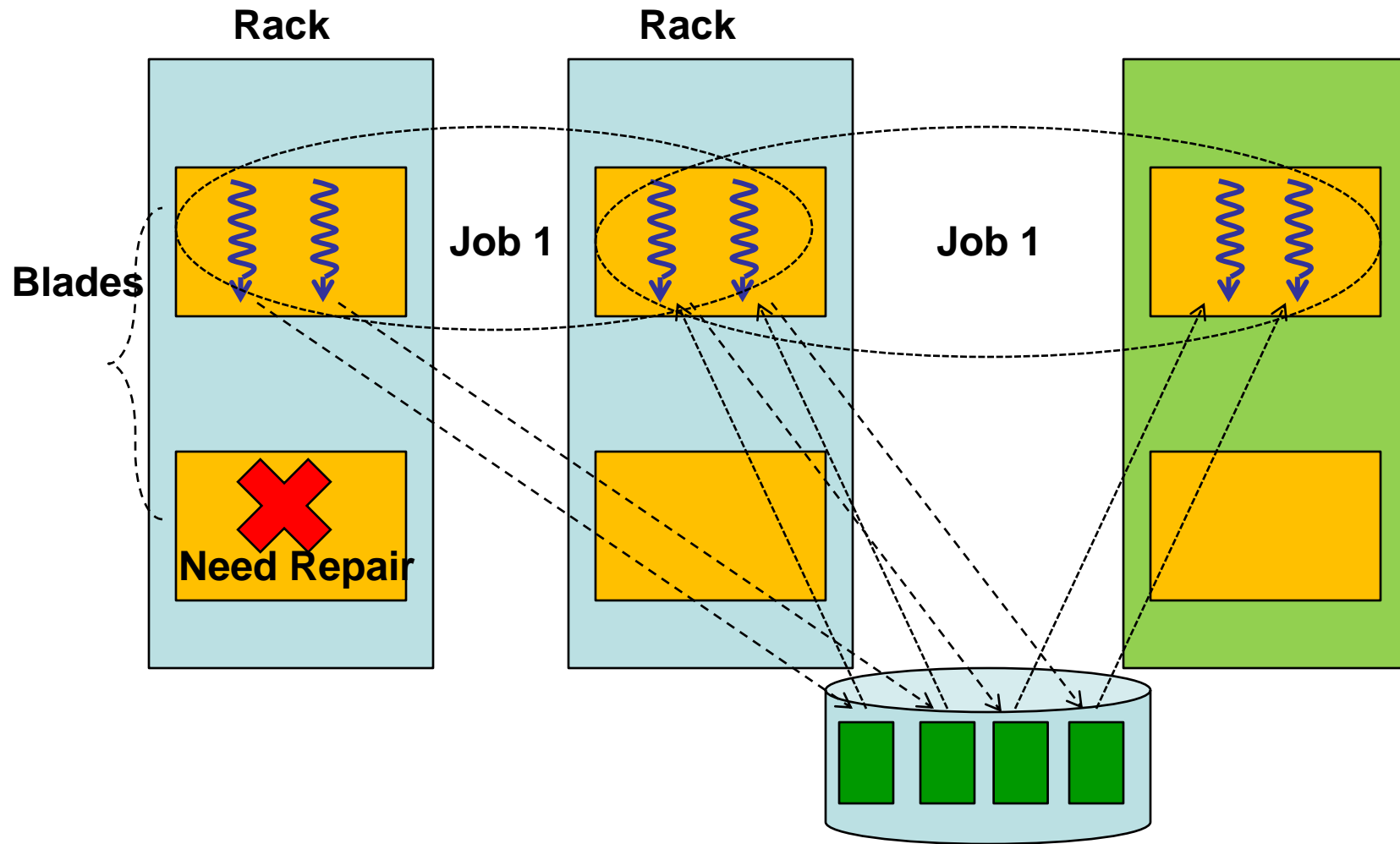
- Dynamic Process Relocation is needed under failure or maintenance
 - Replace a node not used by a job
 - Upgrade some nodes (blades or racks) not used by a job
 - Remap processes to other nodes/blades/racks in the system
- Health Monitoring Tools like IPMI provide advance warning about failures
- Failure predication models are also being designed

Dynamic Process Relocation

- To minimize fragmentation
- Typical job scheduler on large supercomputer centers
 - Jobs get scattered across nodes after sometime
 - A job may not be running on contiguous allocation
- Can we do dynamic process relocation while a job is running
 - To bring the processes into a contiguous allocation

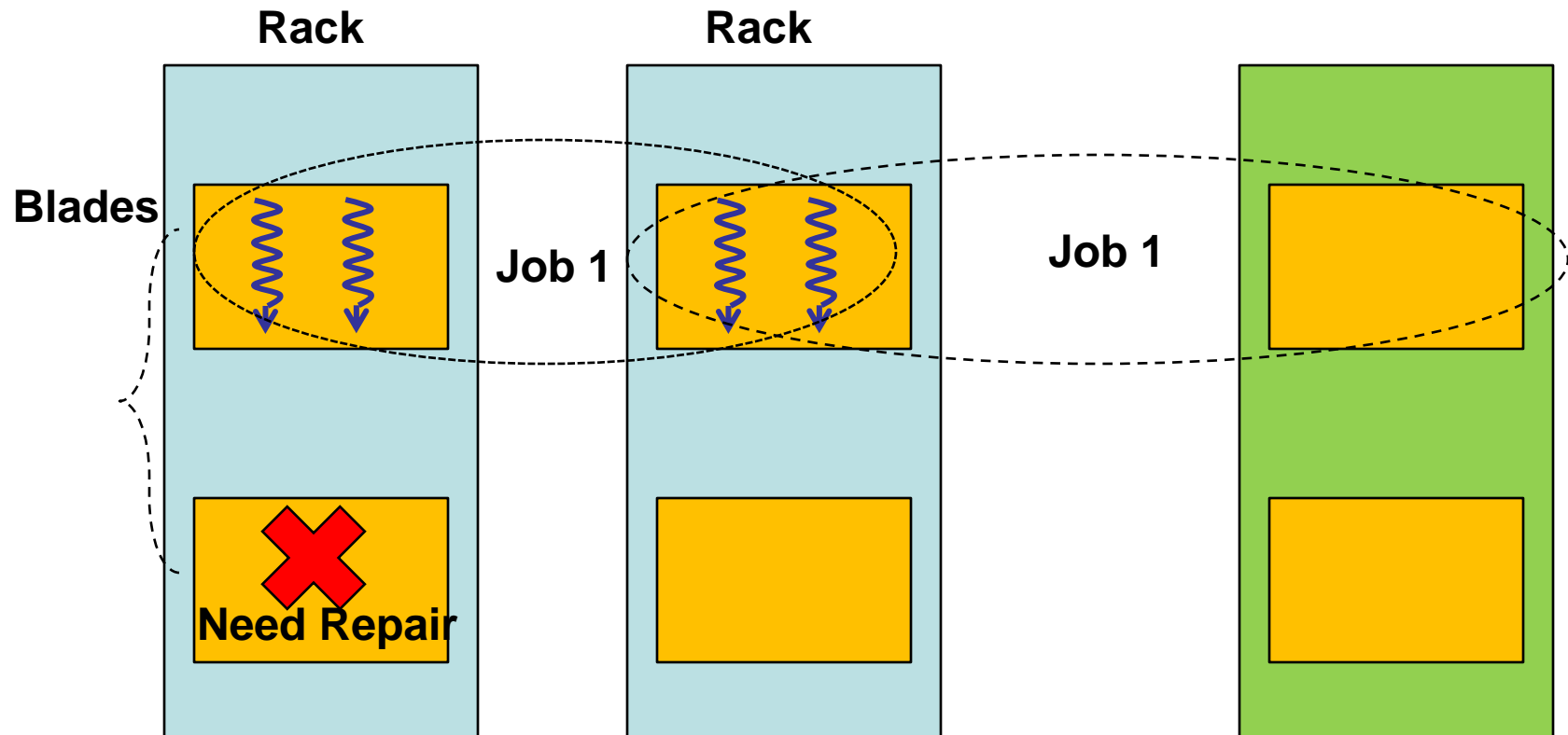
Need for Dynamic Process Relocation

Current Solution:



Need for Dynamic Process Relocation

Efficient Solution:



Key Observations

- When a failure is going to happen
 - Only failing node needs checkpoint
 - Other healthy nodes can pause their processes (not checkpoint)
 - Don't need to save images to shared storage
 - Migrate processes from faulty nodes to spare nodes
 - Don't need to resubmit the job
 - Resume all processes after migration

Problem Statement

- How to reduce the overhead to handle node failure/maintenance/migration?
 - How to design an efficient job migration framework within a MPI library?
- How to take advantages of advanced network (such as InfiniBand) to improve migration performance?
 - How to exploit RDMA to migrate processes
- What are the costs and benefits of the proposed solutions?

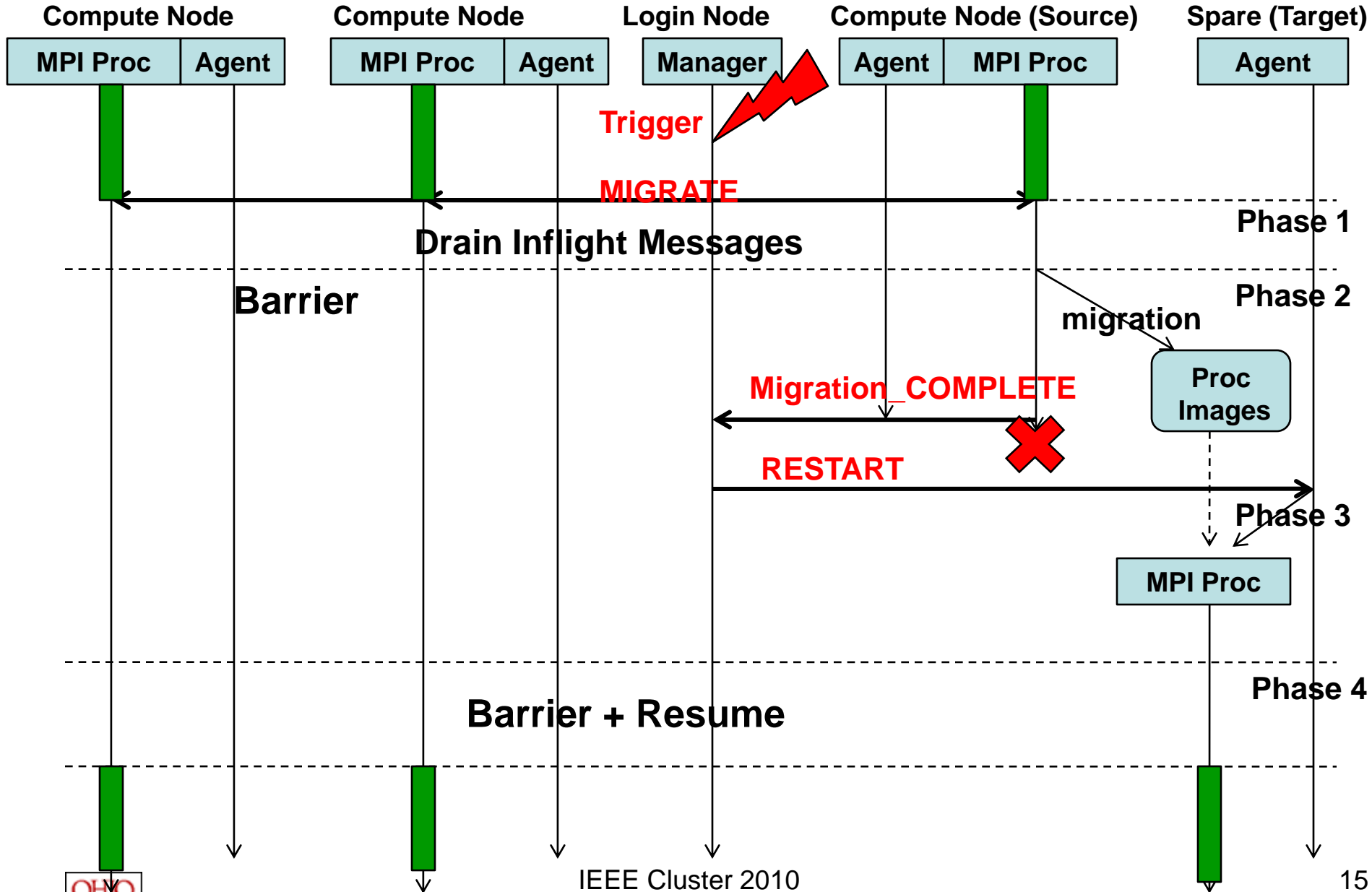
Outline

- Introduction and Motivation
- Job Migration Framework for MVAPICH2
- RDMA-based Process Migration
- Performance Evaluation
- Conclusions and Future Work

MVAPICH/MVAPICH2 Software

- MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RDMA over Converged Enhanced Ethernet (RoCE)
 - MVAPICH (MPI-1) and MVAPICH2 (MPI-2)
 - Used by more than 1,255 organizations worldwide (in 59 countries)
 - Empowering many TOP500 clusters (6th, 7th, 11th, ..)
 - Available with software stacks of many IB, 10GE/iWARP and RoCE, and server vendors including Open Fabrics Enterprise Distribution (OFED)
 - Available with Redhat and SuSE Distributions
 - <http://mvapich.cse.ohio-state.edu/>
- Has support for Checkpoint-Restart for the last several years
 - Already used by many organizations

Job Migration Framework for MVAPICH2



MVAPICH2 Job Migration

- Phase 1: Job Stall
 - Suspend communication, drain in-flight messages
- Phase 2: Job Migration
 - Processes on healthy nodes enter a barrier
 - Source node(s) take checkpoint
 - Transfer checkpoint(s) to target
- Phase 3: Restart
 - Target node(s) restart processes
- Phase 4: Resume
 - All processes synchronize at barrier
 - Rebuild connection and resume

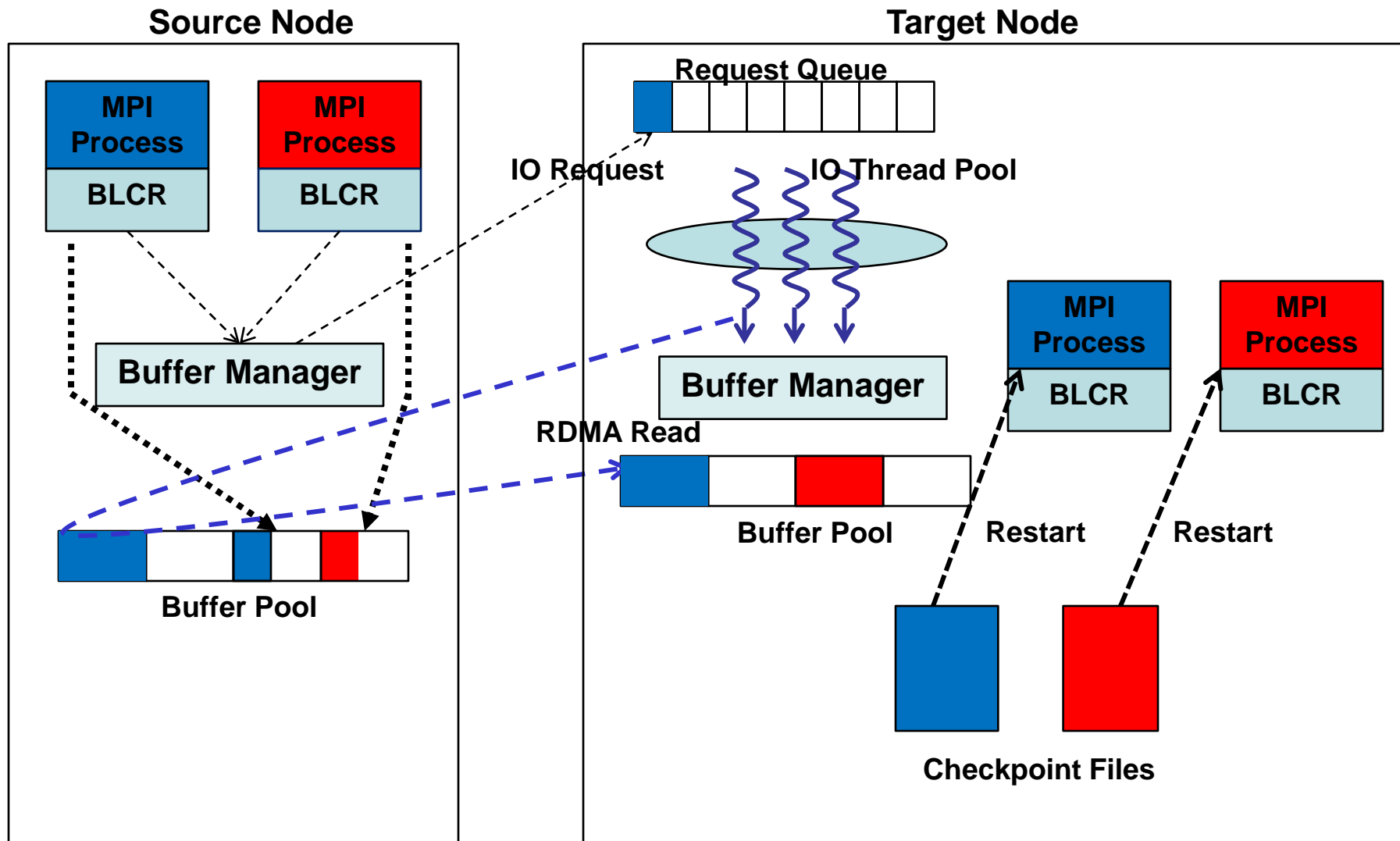
Challenges

- How to migrate processes
 - Source node: Use BLCR to checkpoint the processes
 - How to move process images to target
 - Save to shared filesystem? (IO bottleneck)
 - Solution
 - Utilize InfiniBand memory-based communication (RDMA)
 - Transfer image data on-the-fly to target node
 - Save image data as checkpoint files on target local file system

Outline

- Introduction and Motivation
- Job Migration Framework for MVAPICH2
- **RDMA-based Process Migration**
- Performance Evaluation
- Conclusions and Future Work

RDMA-based Process Migration



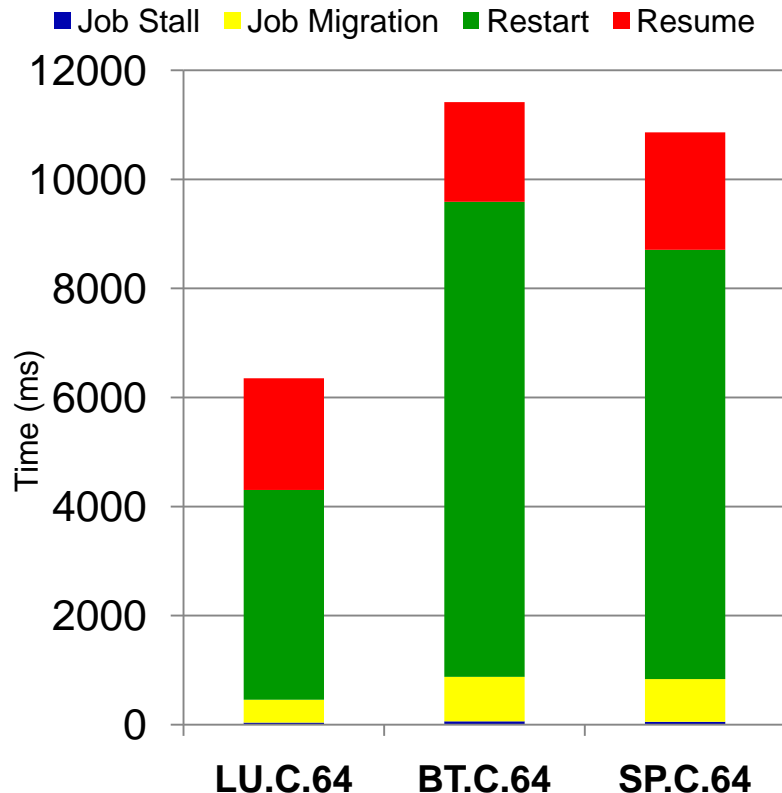
Outline

- Introduction and Motivation
- Job Migration Framework for MVAPICH2
- RDMA-based Process Migration
- **Performance Evaluation**
- Conclusions and Future Work

Experimental setup

- System setup
 - Intel Clovertown cluster
 - Dual-socket Quad core Xeon processors, 2.33GHz
 - Nodes are connected by InfiniBand DDR
 - RedHat 5.4
- NAS parallel Benchmark suite version 3.2.1
 - LU/BT/SP Class C, 64 processes, 8 processes/node
 - 8 compute nodes used
- MVAPICH2 with Job Migration Framework
 - BLCR 0.8.0 extended with IO Aggregation

Migration overhead

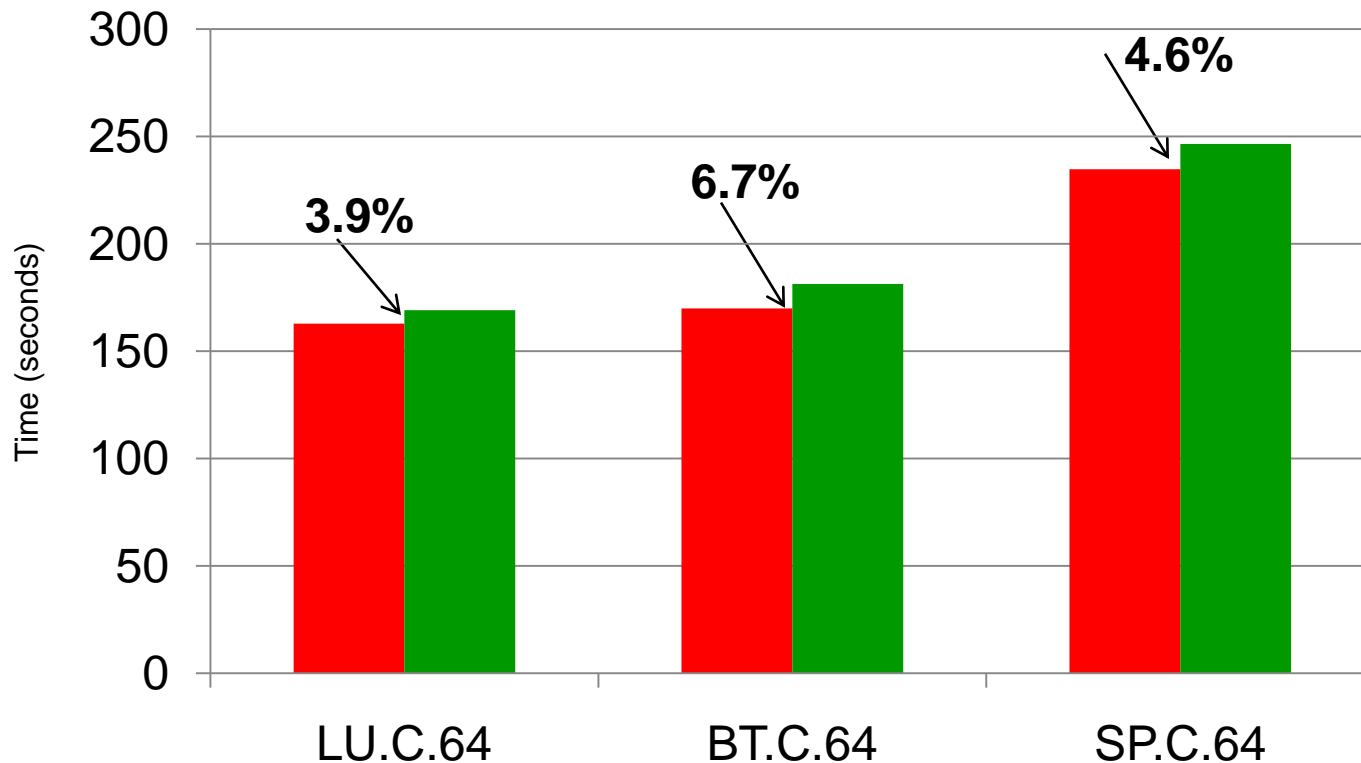


- Phase 1: Very swift
- Phase 2: RDMA-based transfer efficiently moves process image
- Phase 3: Restart processes from disk files
- Can improve this cost

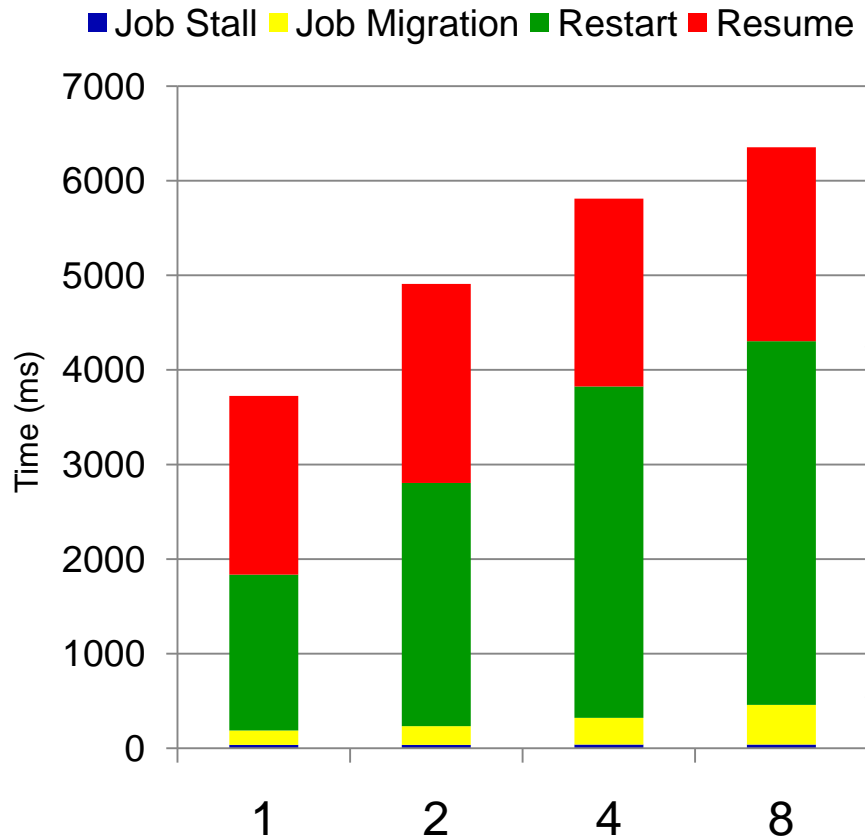
- Buffer pool = 10MB, chunk-size=1MB
- Not sensitive to buffer pool size

Impact on Total execution time with migration

■ No Migration ■ 1 Migration



Migration Design Scalability with Multi-core Architecture



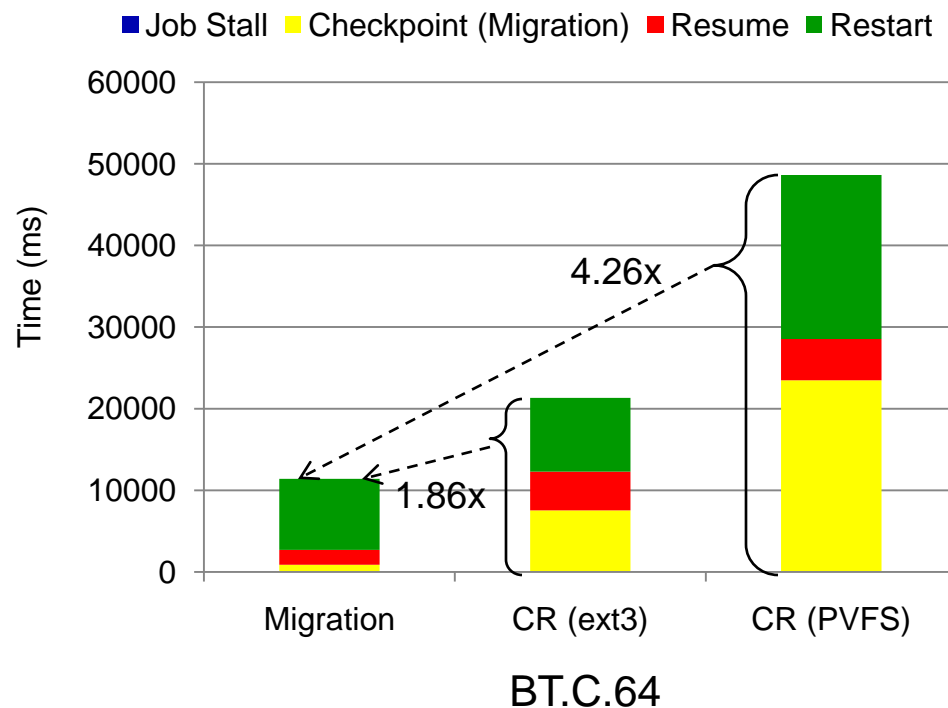
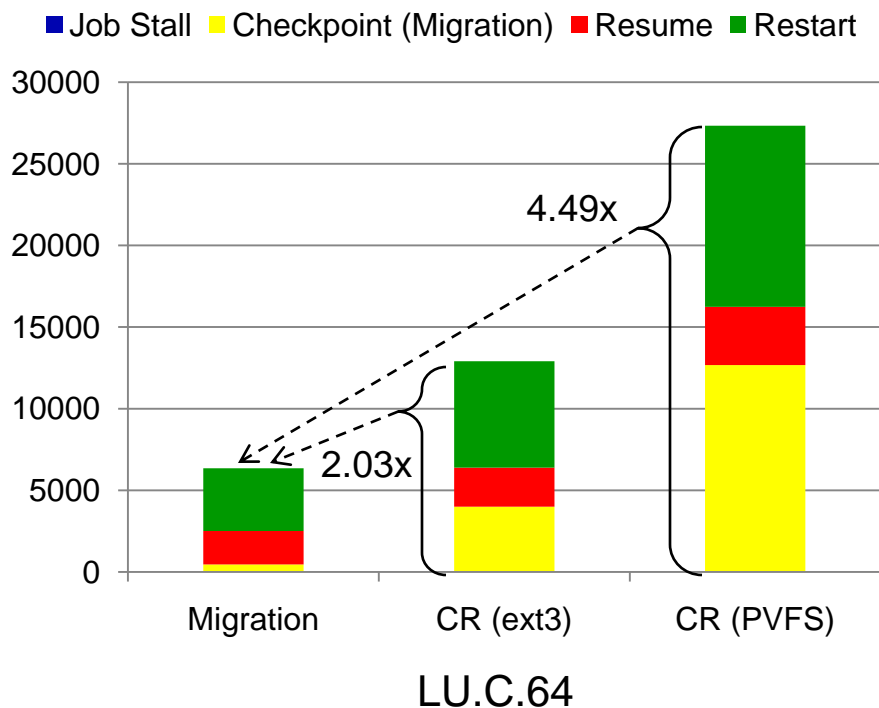
- Application LU.C, on 8 compute nodes

- 8/16/32/64 processes, 1/2/4/8 processes per node

- Phase 3: Restore processes context from checkpoint files,

- Cost in proportion to task scale / Image size

Comparison with Checkpoint/Restart (CR)

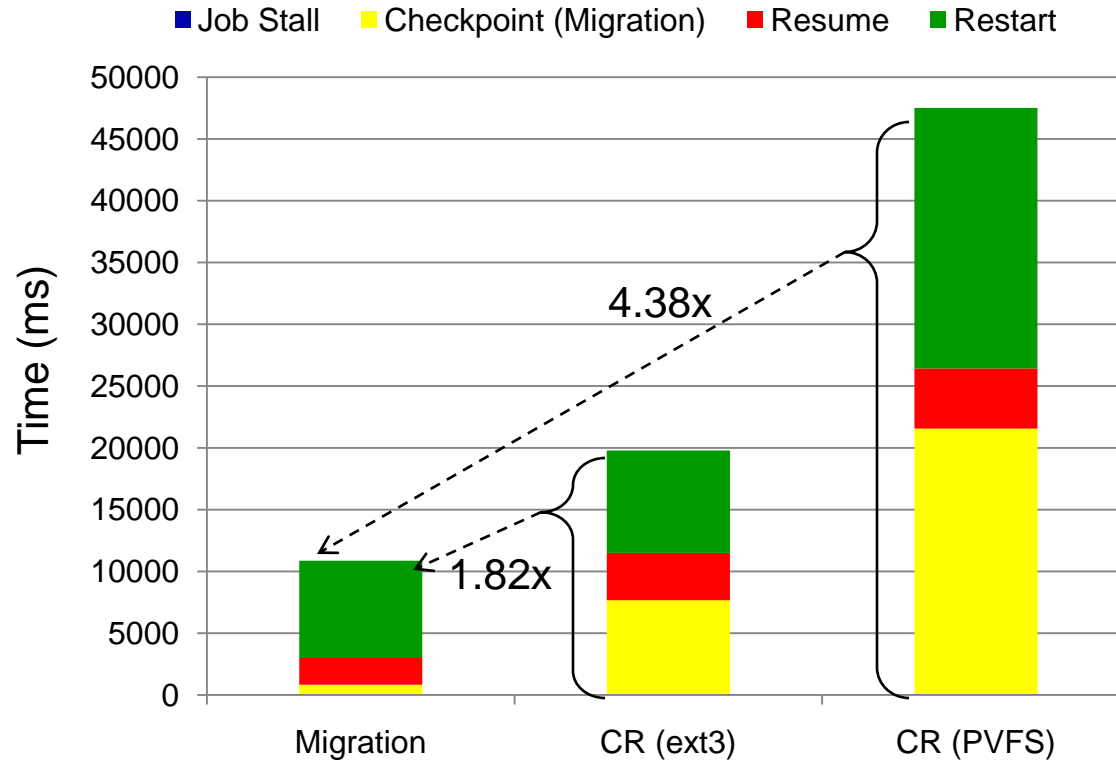


Data moved(MB)

| | Job Migration | CR |
|---------|---------------|--------|
| LU.C.64 | 170.4 | 1363.2 |
| BT.C.64 | 308.8 | 2470.4 |
| SP.C.64 | 303.2 | 2425.6 |

- **Ext3:** save checkpoint files at local disk
- **PVFS:** save checkpoint files to shared PVFS 2.8.1 file system

Comparison with Checkpoint/Restart (CR)



SP.C.64

Outline

- Introduction and Motivation
- Job Migration Framework for MVAPICH2
- RDMA-based Process Migration
- Performance Evaluation
- Conclusions and Future Work

Conclusion

- Job Migration is an effective approach to achieve fault tolerance
 - Less susceptible to IO bottleneck
 - Less overhead
 - Relies on a failure detection/prediction model
- Job Migration can also help in avoiding fragmentation
- RDMA-based process migration significantly reduces migration overhead
- Will be available in upcoming MVAPICH2 releases

Future Work

- Enhance Job Migration
 - Reduce the cost of restart at spare node
 - In-memory restart form partial data
- Design hybrid Checkpoint-Restart + Migration framework to handle failures for very large-scale systems

Thank you!



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

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

Network-Based Computing Laboratory