Application-Transparent Checkpoint/Restart for MPI Programs over InfiniBand

Qi Gao, Weikuan Yu, Wei Huang, Dhabaleswar K. Panda

Network-Based Computing Laboratory
Department of Computer Science & Engineering
The Ohio State University

Introduction

- Nowadays, clusters have been increasing in their sizes to achieve high performance.
- High Performance $\overset{?}{=}$ High Productivity
- Failure rate of the systems grows rapidly along with the system size
- System failures are becoming an important limiting factor of the productivity of large-scale clusters

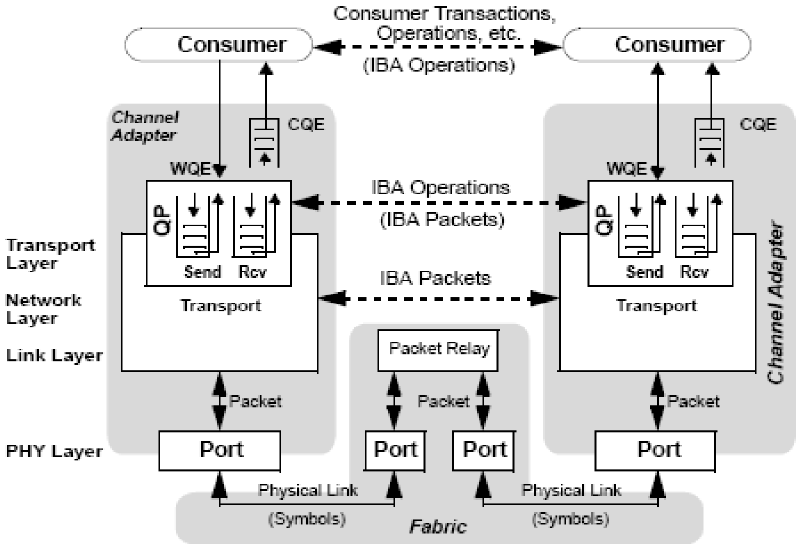
Motivation

- Most end applications are parallelized
  - Many are written in MPI.
  - More susceptible to failures.
  - Many research efforts, e.g. MPICH-V, LAM/MPI, FT-MPI, C^3, etc., for fault tolerance in MPI
- Newly deployed clusters are often equipped with high speed interconnect for high performance
  - InfiniBand: an open industrial standard for high speed interconnect.
    - Used by many large clusters in Top 500 list.
    - Clusters with tens of thousand cores are being deployed
- How to achieve fault tolerance for MPI on InfiniBand clusters to provide both high performance and robustness is an important issue

Outline

- Introduction & Motivation
- Background
  - InfiniBand
  - Checkpointing & rollback recovery
- Checkpoint/Restart for MPI over InfiniBand
- Evaluation framework
- Experimental results
- Conclusions and Future work

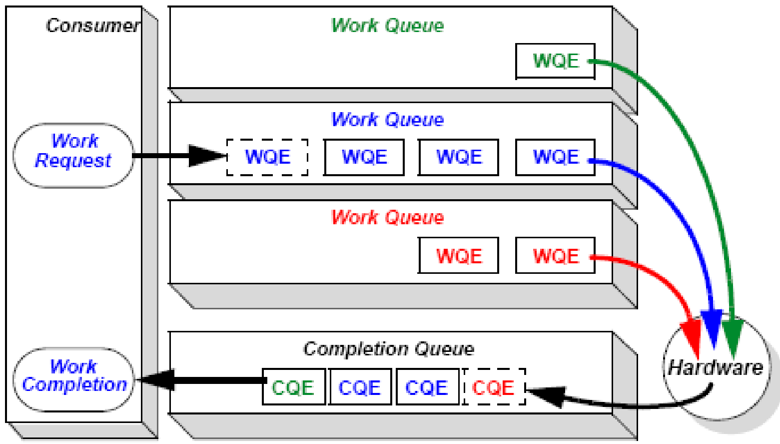
InfiniBand



InfiniBand Stack (Courtesy from IB Spec.)

- Native InfiniBand transport services
- Protocol off-loading to Channel Adapter (NIC)
- High performance RDMA operations

- Queue-based model
  - Queue Pairs (QP)
  - Completion Queues (CQ)
- OS-bypass
- Protection & Authorization
  - Protection Domain (PD)
  - Memory Regions (MR) and access keys



Queuing Model (Courtesy from IB Spec.)

Checkpointing & Rollback Recovery

- Checkpointing & rollback recovery is a commonly used method to achieve fault tolerance.
- *Which* checkpointing method is suitable for clusters with high speed interconnects like InfiniBand?
- Categories of checkpointing:

Coordinated

Pros:
- Easy to guarantee consistency

Cons:
- Coordination overhead
- All processes must rollback upon failure

Uncoordinated

Pros:
- No global coordination

Cons:
- *domino effect* or message logging overhead

Communication Induced

Pros:
- Guarantee consistency without global coordination

Cons:
- Requires per-message processing
- High overhead

Checkpointing & Rollback Recovery (Cont.)

- Implementation of checkpointing:

System Level

Pros:
- Can be transparent to user applications
- Checkpoints initiated independent to the progress of application

Cons:
- Need to handle consistency issue

Application Level

Pros:
- Content of checkpoints can be customized
- Portable checkpoint file

Cons:
- Applications' source code need to be rewritten according to checkpointing interface

Compiler Assisted

Pros:
- Application level checkpointing without source code modification

Cons:
- Requires special compiler techniques for consistency

- Our current approach: Coordinated, System-Level, Application Transparent Checkpointing

NETWORK-BASED COMPUTING LABORATORY

Outline

- Introduction & Motivation
- Background
- Checkpoint/Restart for MPI over InfiniBand
- Evaluation Framework
- Experimental Results
- Conclusions and Future Work

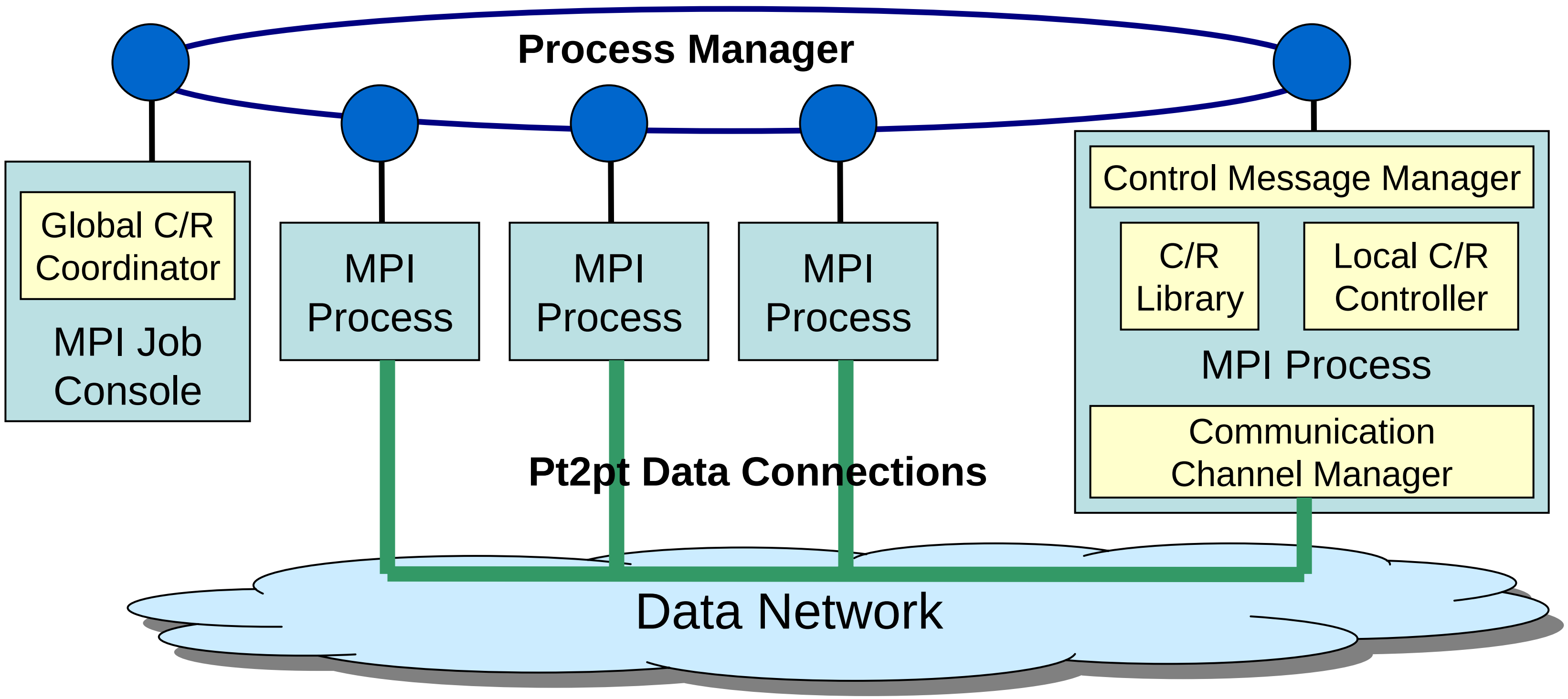
OHIO STATE

Overview

- Checkpoint/Restart for MPI programs over InfiniBand:
  - Using Berkeley Lab's Checkpoint/Restart (BLCR) for taking snapshots of individual processes on a single node.
  - Design coordination protocol to checkpoint and restart the entire MPI job;
  - Totally transparent to user applications;
  - Does not interfere critical path of data communication.

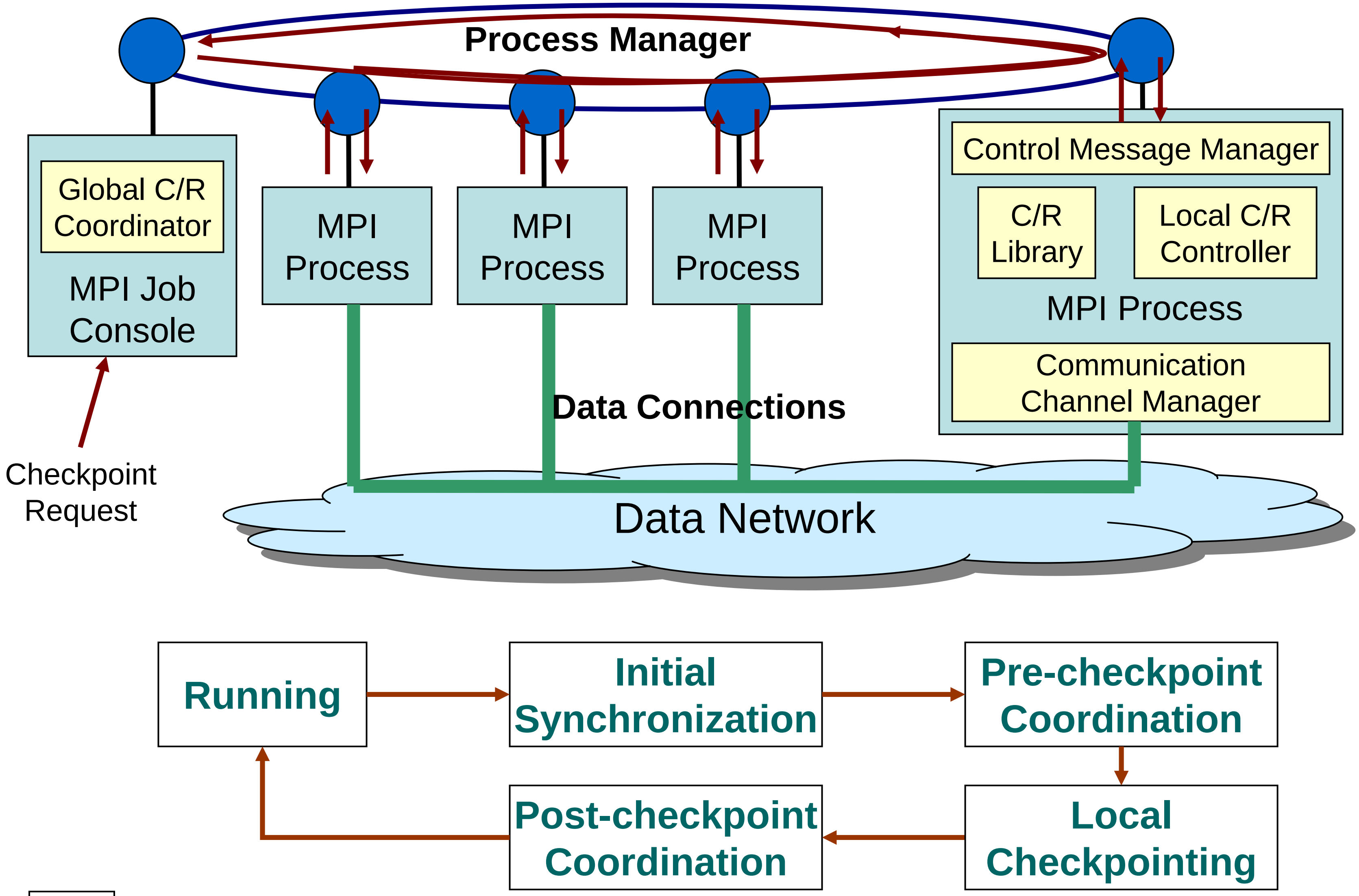
- Suspend/Reactivate the InfiniBand communication channel in MPI library upon checkpoint request.
  - Network connections on InfiniBand are disconnected
  - Channel consistency is maintained.
  - Transparent to upper layers of MPI library

Checkpoint/Restart (C/R) Framework

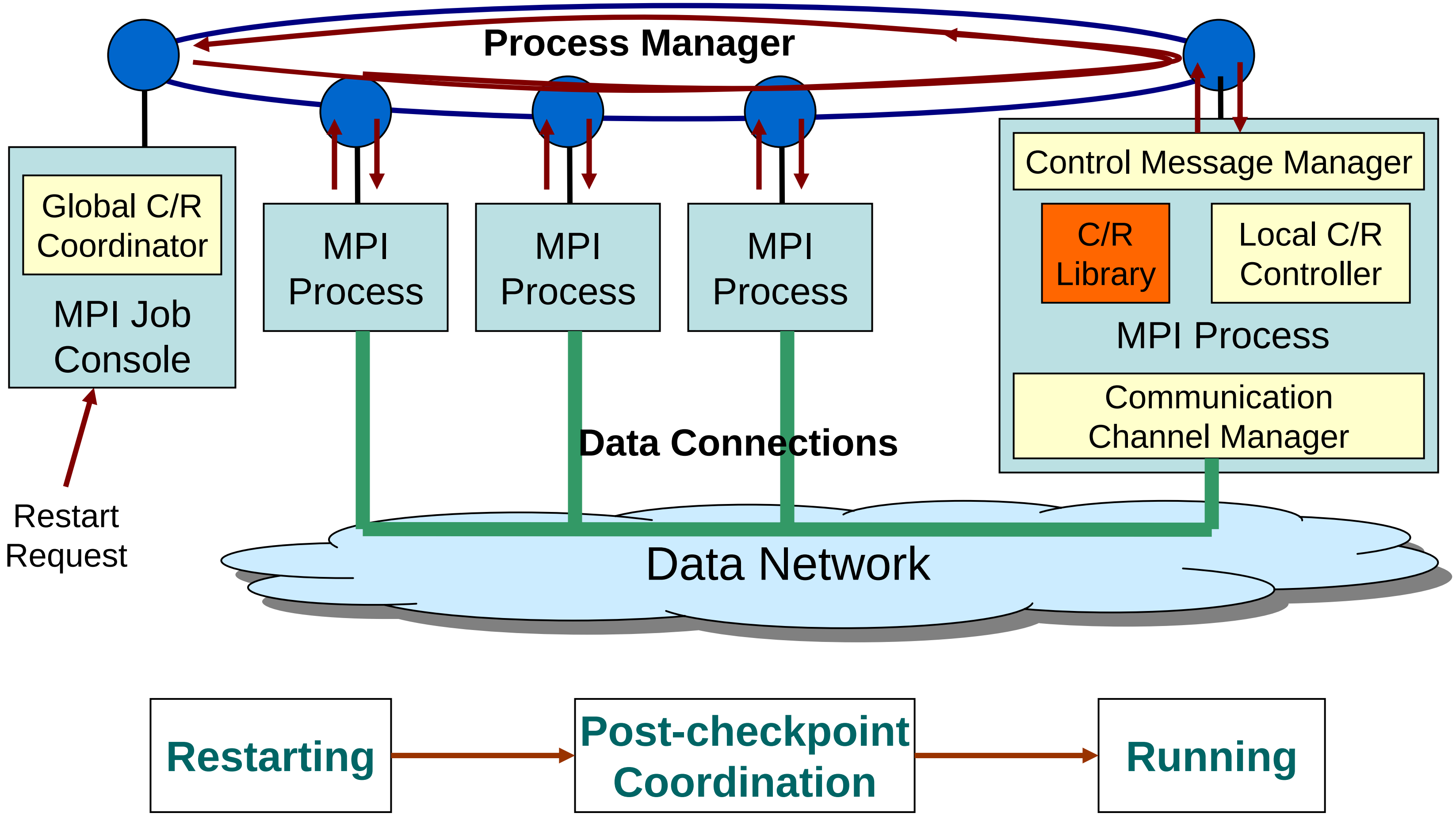


- In our current implementation:
  - Process Manager: Multi-Purpose Daemon (MPD), developed in ANL, extended with C/R messaging support
  - C/R Library: Berkeley Lab’s Checkpoint/Restart (BLCR)

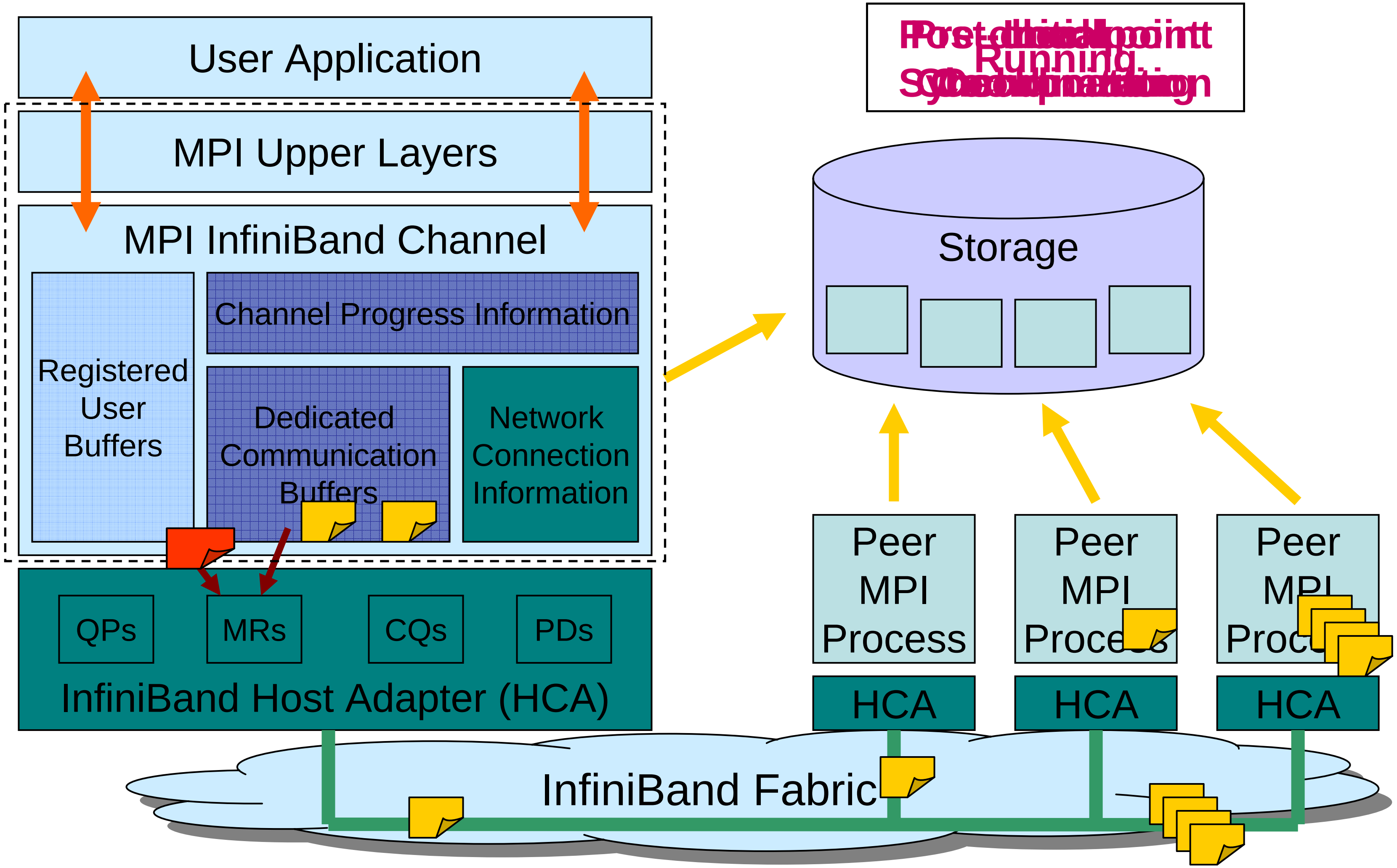
Global View: Procedure of Checkpointing



Global View: Procedure of Restarting



Local View: InfiniBand Channel in MPI



Outline

- Introduction & Motivation
- Background
- Checkpoint/Restart for MPI over InfiniBand
- Evaluation Framework
- Experimental Results
- Conclusions and Future Work

OSU MPI over InfiniBand

- Open Source High Performance Implementations
  - MPI-1 (MVAPICH)
  - MPI-2 (MVAPICH2)
- Has enabled a large number of production IB clusters all over the world to take advantage of InfiniBand
  - Largest being Sandia Thunderbird Cluster (4512 nodes with 9024 processors)
- Have been directly downloaded and used by more than 390 organizations worldwide (in 30 countries)
  - Time tested and stable code base with novel features
- Available in software stack distributions of many vendors
- Available in the OpenFabrics(OpenIB) Gen2 stack and OFED
- More details at

  http://nowlab.cse.ohio-state.edu/projects/mpi-iba/

Evaluation Framework

- Implementation based on MVAPICH2 version 0.9.0
- Will be released with newer version of MVAPICH2 soon
- Test-bed:
  - InfiniBand Cluster with 12 nodes, dual Intel Xeon 3.4 GHz CPUs, 2 GB memory, Redhat Linux AS 4 with kernel version 2.6.11;
  - Ext3 file system on top of local SATA disks
  - Mellanox InfiniHost MT23108 HCA adapters
- Experiments:
  - Analysis of overhead for taking one checkpoint and restart
    - NAS Parallel Benchmarks
  - Performance impact to applications when checkpointing periodically
    - NAS Parallel Benchmarks
    - HPL Benchmark
    - GROMACS

Outline

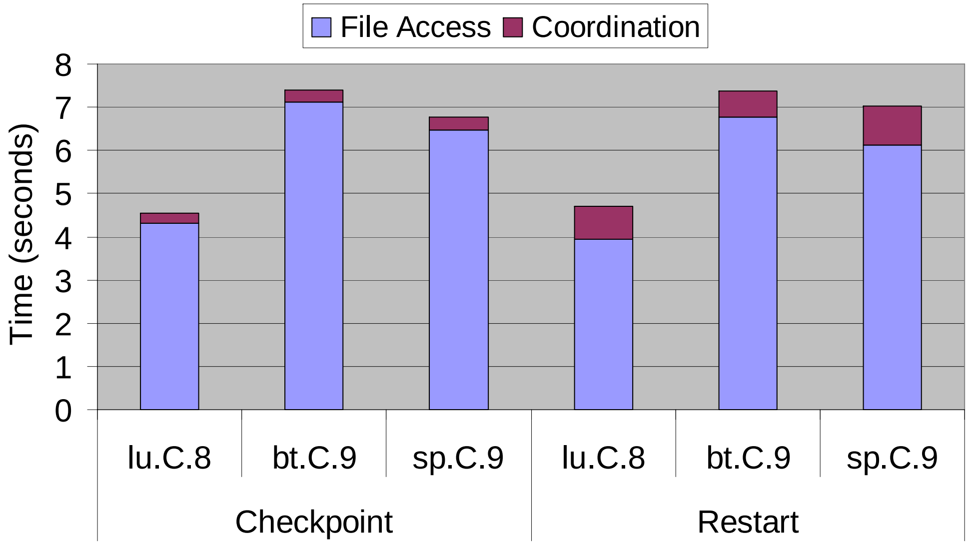
- Introduction & Motivation
- Background
- Checkpoint/Restart for MPI over InfiniBand
- Evaluation Framework
- Experimental Results
- Conclusions and Future Work

Checkpoint/Restart Overhead

- Storage overhead
  - Checkpoint size is same as the memory used by process:

Benchmark	LU.C.8	BT.C.9	SP.C.9
Checkpoint size per process	126MB	213MB	193MB

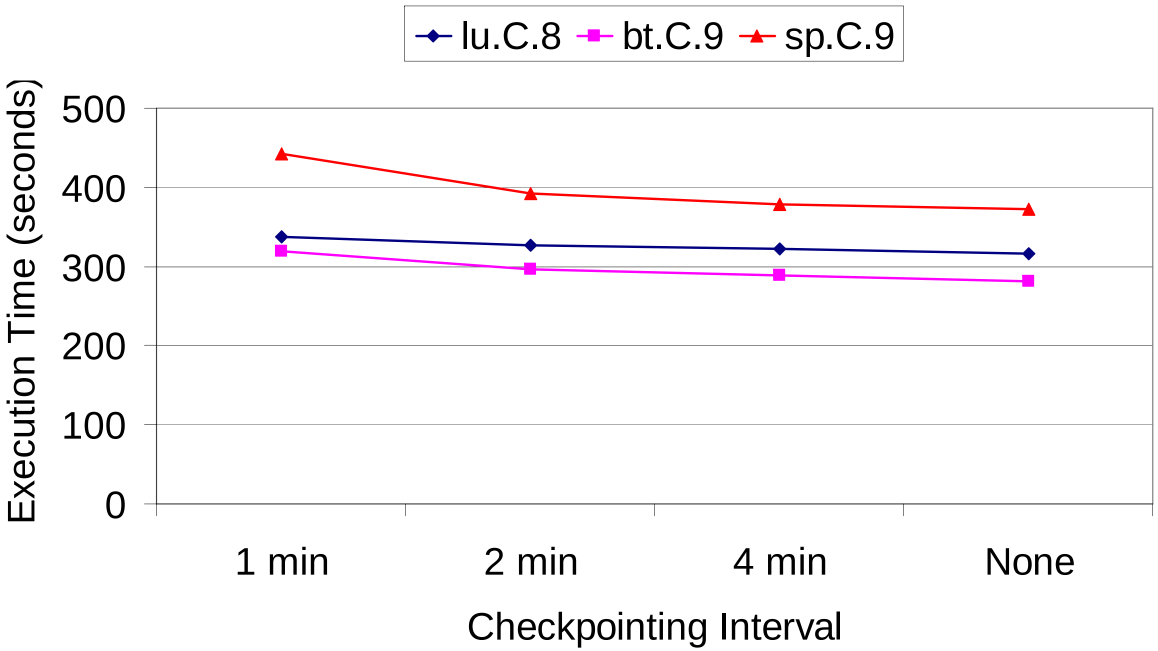
- Time for checkpointing



- Delay from issuance of checkpoint/restart request to program resumes execution
- Sync checkpoint file to local disk before program continues

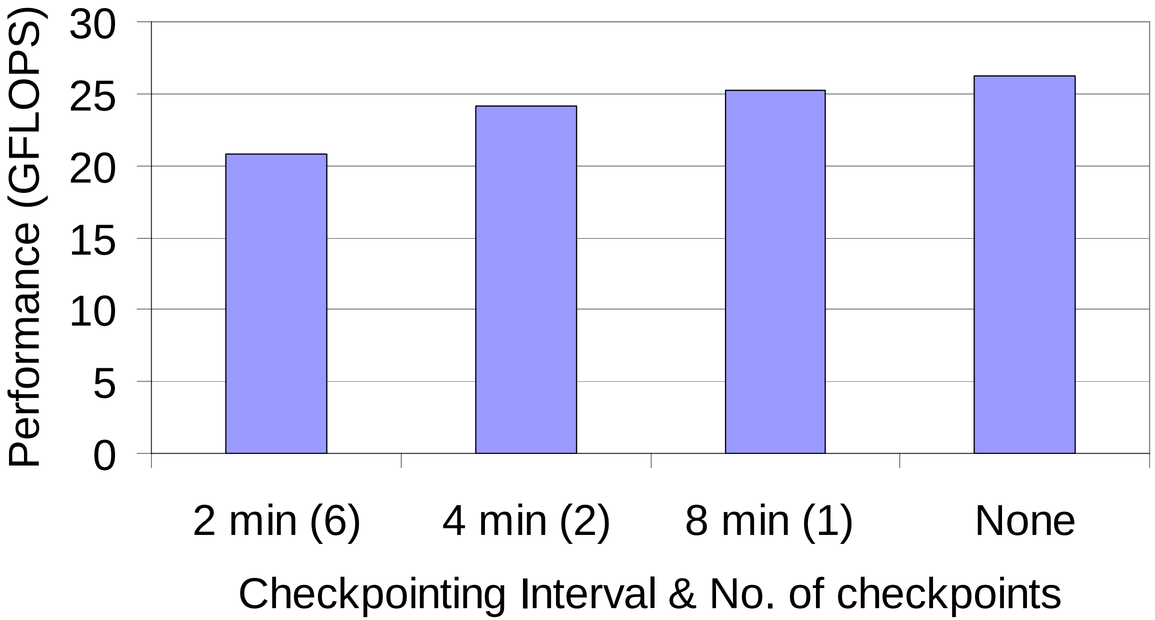
File accessing time is the dominating factor of checkpoint/restart overhead

Performance Impact to Applications – NAS Benchmarks



- NAS benchmarks, LU, BT, SP, Class C, for 8~9 processes
- For each checkpoint, the execution time increases for about 2-3%

Performance Impact to Applications – HPL Benchmark

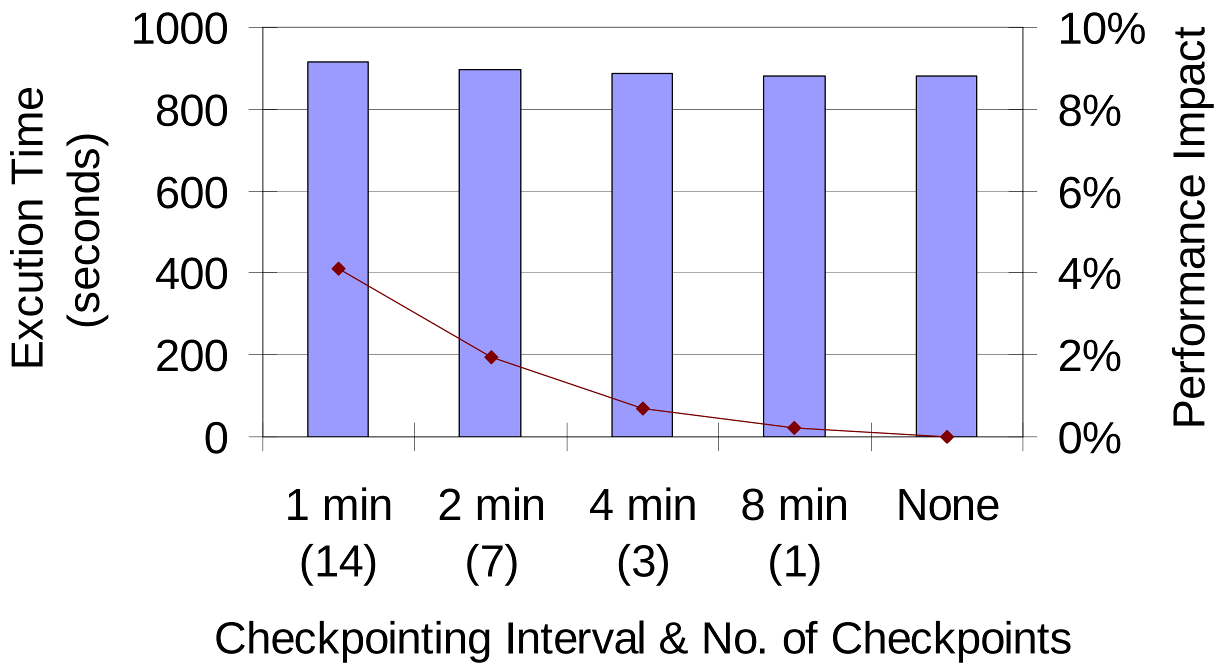


- HPL benchmarks, 8 processes.
- Performs same as original MVAPICH2 when taking no checkpoints
- For each checkpoint, the performance degradation is about 4%.

Benchmarks V.S. Target Applications

- Benchmarks
  - Seconds, minutes (checkpoint in a few minutes)
  - Load all data into memory at beginning
  - The ratio of (memory usage / running time) is high
- Target applications: long running applications
  - Days, weeks, months (checkpoint hourly, daily, or weekly)
  - Computation intensive or load data into memory gradually
  - The ratio of (memory usage / running time) is low
- Benchmarks reflects almost the worst case scenario
  - Checkpointing overhead largely depends on checkpoint file size (process memory usage)
  - Relative overhead is very sensitive to the ratio.

Performance Impact to Applications – GROMACS



- GROMACS
  - Molecular dynamics for biochemical analysis.
  - DPPC dataset running on 10 processes.
- Small memory usage with relatively longer running time.
- For each checkpoint, the execution time increases less than 0.3%.

NETWORK-BASED COMPUTING LABORATORY

Outline

- Introduction & Motivation
- Background
- Checkpoint/Restart for MPI over InfiniBand
- Evaluation Framework
- Experimental Results
- Conclusions and Future Work

OHIO STATE

Conclusions and Future Work

- Design & implement a framework to checkpoint and restart for MPI programs over InfiniBand
- Totally transparent to MPI applications
- Evaluations based on NAS, HPL, and GROMACS show that the overhead for checkpointing is not significant

- Future work:
  - Reduce the checkpointing overhead
  - Design a more sophisticated framework for fault tolerance in MPI
  - Integrate into MVAPICH2 release

Acknowledgements

Our research is supported by the following organizations

- Current Funding supported by

- Current Equipment supported by

Web Pointers

{gaoq, yuw, huanwei, panda}@cse.ohio-state.edu

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

MVAPICH Web Page
http://nowlab.cse.ohio-state.edu/projects/mpi-iba/