Group-based Coordinated Checkpointing for MPI: A Case Study on InfiniBand

Qi Gao, Wei Huang, Matthew J. Koop, and Dhabaleswar K. Panda

Network Based Computing Laboratory (NBCL) *

The Ohio State University

Outline

- Introduction, Background, and Motivation
- Main Idea and Design
- Experimental Platform
- Performance Results
- Conclusions

Introduction

- Fault tolerance becomes increasingly important for scientific applications

- When scaling up:
  - Mean Time Between Failure (MTBF) goes down
  - Cost of failure goes up

- How to achieve fault tolerance in large scale is a challenge.

Background: Checkpointing

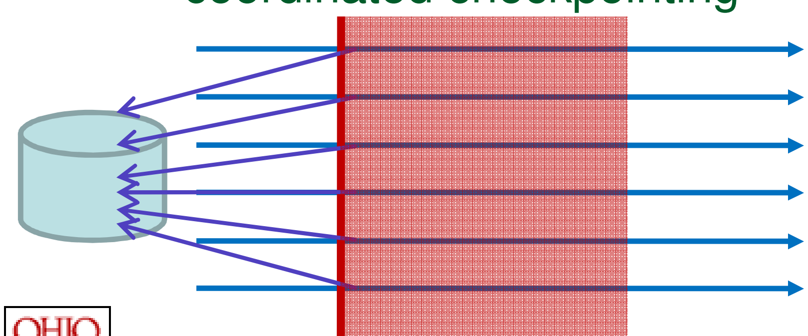
- Checkpointing and rollback recovery:
  - A commonly used method to achieve fault tolerance
  - Save intermediate execution state of the application
  - Upon failure, restart from previous saved state (checkpoint)
- Checkpointing MPI programs
  - Need to maintain global consistency among processes. Lost messages or orphan messages must be avoided.
  - Main categories of checkpointing protocols: Coodinated and Uncoordinated
- Cost of checkpointing
  - Dominating delay for checkpointing is storage access (over 95%)
  - In real world, large scale applications use shared central storage

Comparison between Checkpointing Protocols

Coordinated

- Use global coordination to guarantee consistency
- Processes save their states at relatively same time.
- Storage bottleneck when saving process states

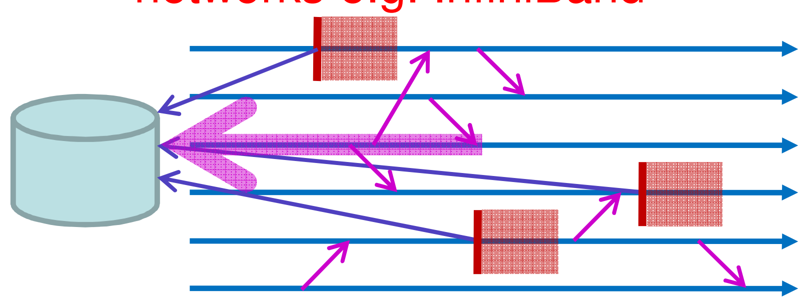
We choose to improve coordinated checkpointing



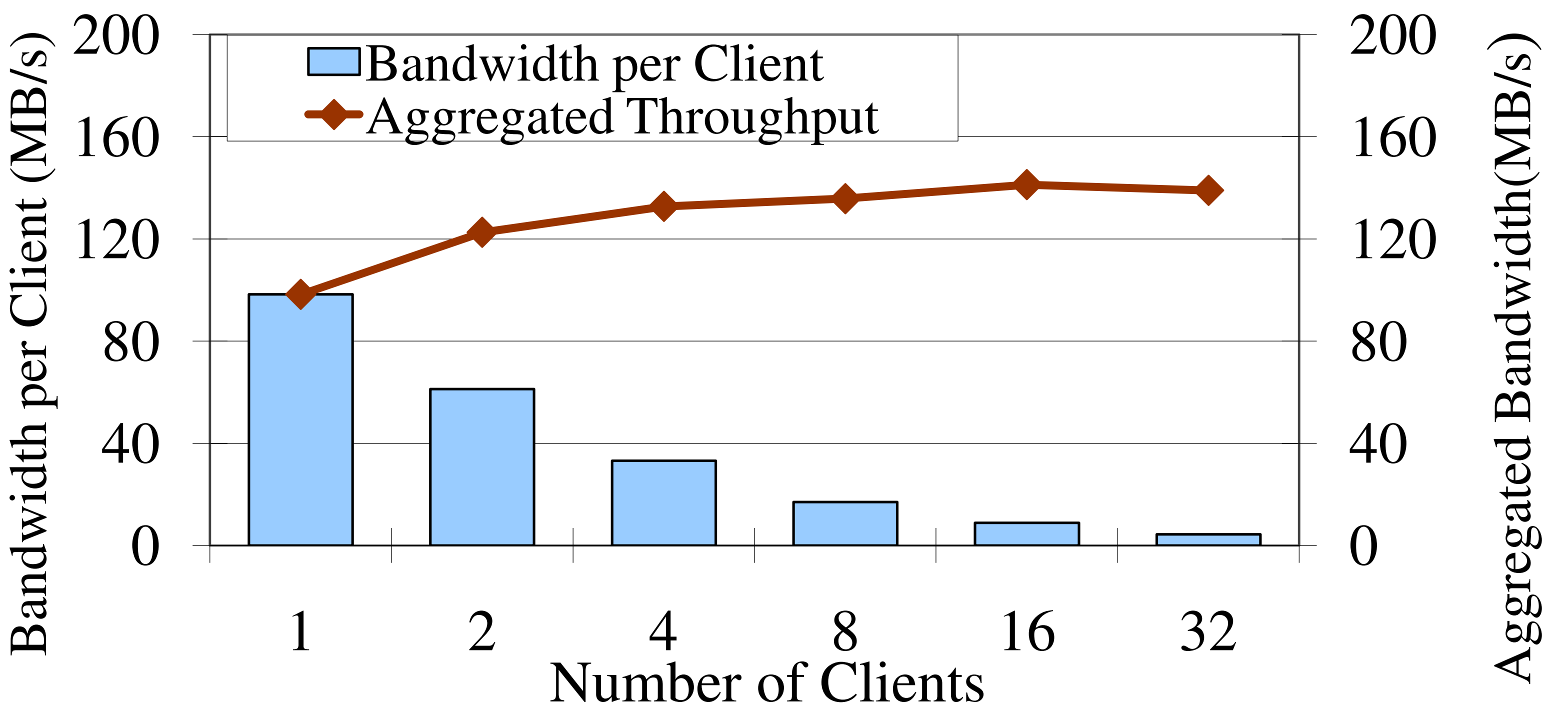
Uncoordinated

- Processes save their states mostly independently
- Use message logging to guarantee consistency
- Message logging incurs overhead in communication

Very expensive on high speed networks e.g. InfiniBand



Storage Bottleneck



32 Processes share 140MB/s aggregated bandwidth (4.38 MB/s per Proc)

- In real deployment of large clusters, the per process bandwidth to file system is even smaller than this.
  - Sandia Thunderbird cluster: 8960 CPUs with 6.0 GB/s storage bandwidth: (0.69 MB/s per Proc)

Summary of Motivation

- Scalability limitation of coordinated checkpointing
  - Large number of processes concurrently take checkpoint ➡ Less bandwidth per process ➡ Longer checkpointing delay
- Goals of this work
  - Combine the advantages of uncoordinated checkpointing to improve coordinated protocol.
  - Alleviate storage bottleneck to improve scalability in real-world scenario
  - Minimize failure-free overhead

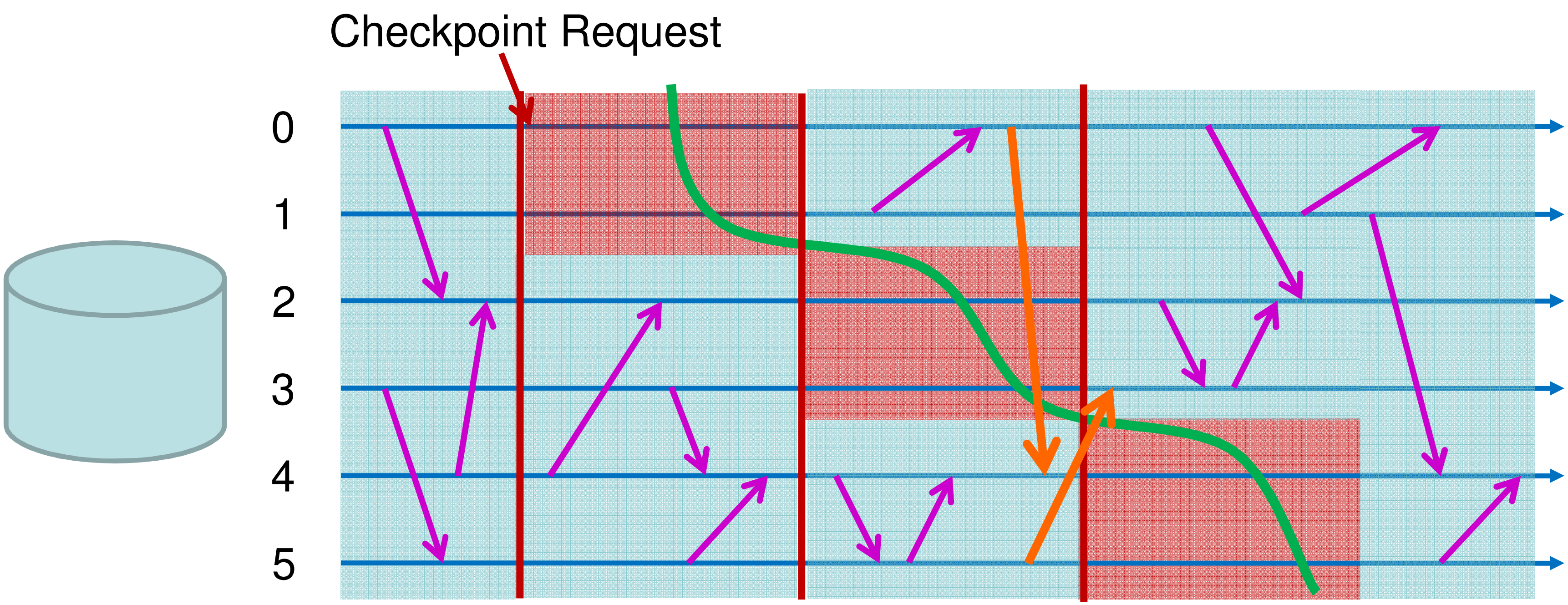
Outline

- Introduction, Background, and Motivation
- Main Idea and Design
- Experimental Platform
- Performance Results
- Conclusions

Main Idea

- Carefully *schedule* the MPI processes to take checkpoints *at slightly different time* to avoid storage bottleneck.
- Allow processes which are not currently taking checkpoints to *proceed with computation*.
- Maintain global consistency by a coordination protocol to *avoid message logging overhead*.

Design: Running Scenario



- Only a small group of processes save their states at same time, while other processes proceed computation
- Delay some messages to ensure global consistency

Detailed Design Issues

- Group formation
  – Statically or dynamically using heuristics
- Connection management
  – Disconnect/Reconnect to a specific set of processes
- Message and request buffering
  – Buffer the message content or the meta-info of the messages (MPI request)
- Asynchronous progress
  – Passive coordination when other groups are taking checkpoint

Outline

- Introduction, Background, and Motivation
- Main Idea and Design
- Experimental Platforms
- Performance Results
- Conclusions

Experimental Platform

- 32 Compute nodes
  – Intel 64-bit Xeon 3.6 GHz CPU, 2 GB memory
  – Mellanox MT25208 InfiniBand HCA
- 4 Storage nodes
  – AMD Operton 2.8 GHz CPU, 4 GB memory
  – Mellanox MT25208 InfiniBand HCA
  – PVFS2 on EXT3 using local SATA disks (File system performance is shown in previous graph)
- Software:
  – BLCR 0.5.0 to take checkpoints of individual processes.

MVPAICH Project

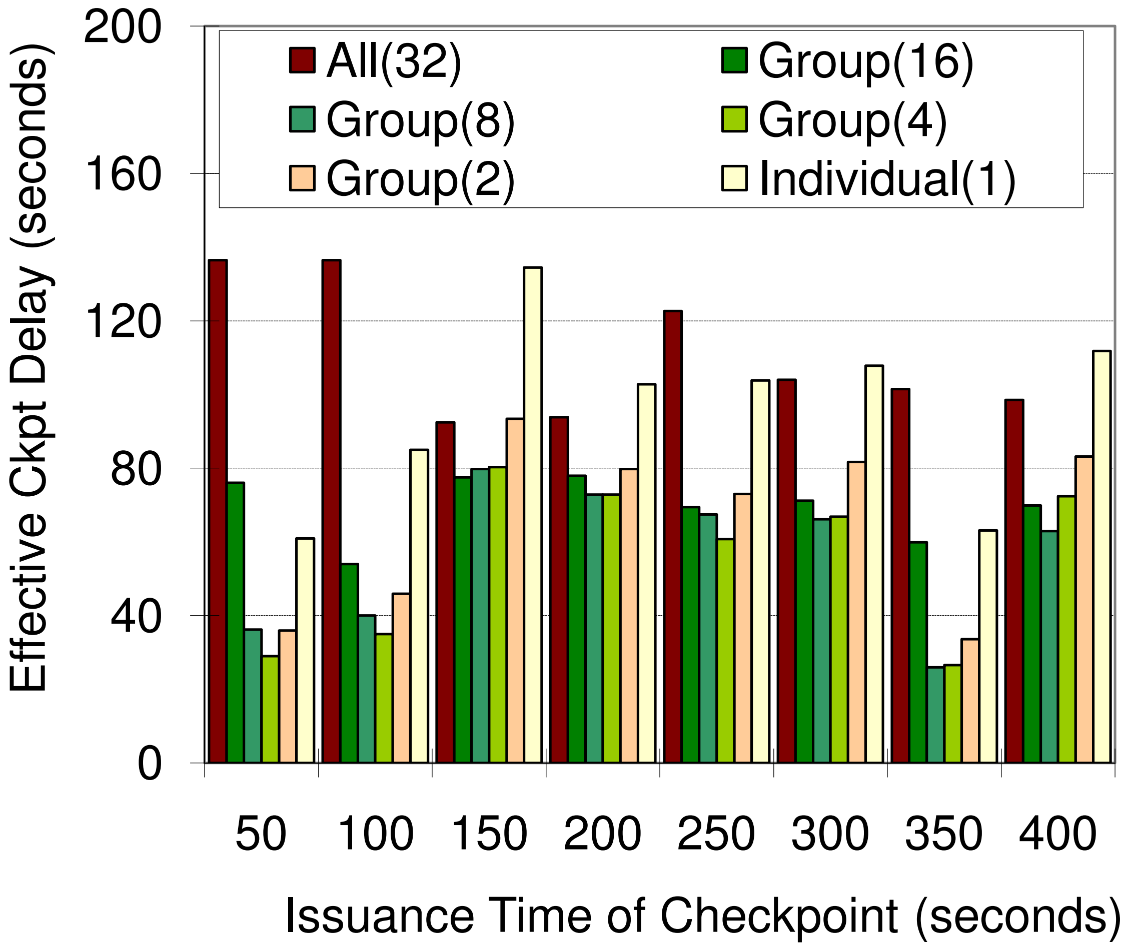
- MVAPICH2
  - High Performance MPI-1/MPI-2 implementation over InfiniBand
  - Has powered many supercomputers in TOP500 supercomputing rankings
  - Currently being used by more than 545 organizations (academia and industry worldwide)
  - http://mvapich.cse.ohio-state.edu/
- MVAPICH2-0.9.8 is currently integrated with coordinated checkpointing.

Q. Gao, W. Yu, W. Huang, and D. K. Panda. Application-Transparent Checkpoint/Restart for MPI Programs over InfiniBand. In proc of *ICPP 06*

Outline

- Introduction, Background, and Motivation
- Main Idea and Design
- Experimental Platforms
- Performance Results
- Conclusions

High Performance Linpack



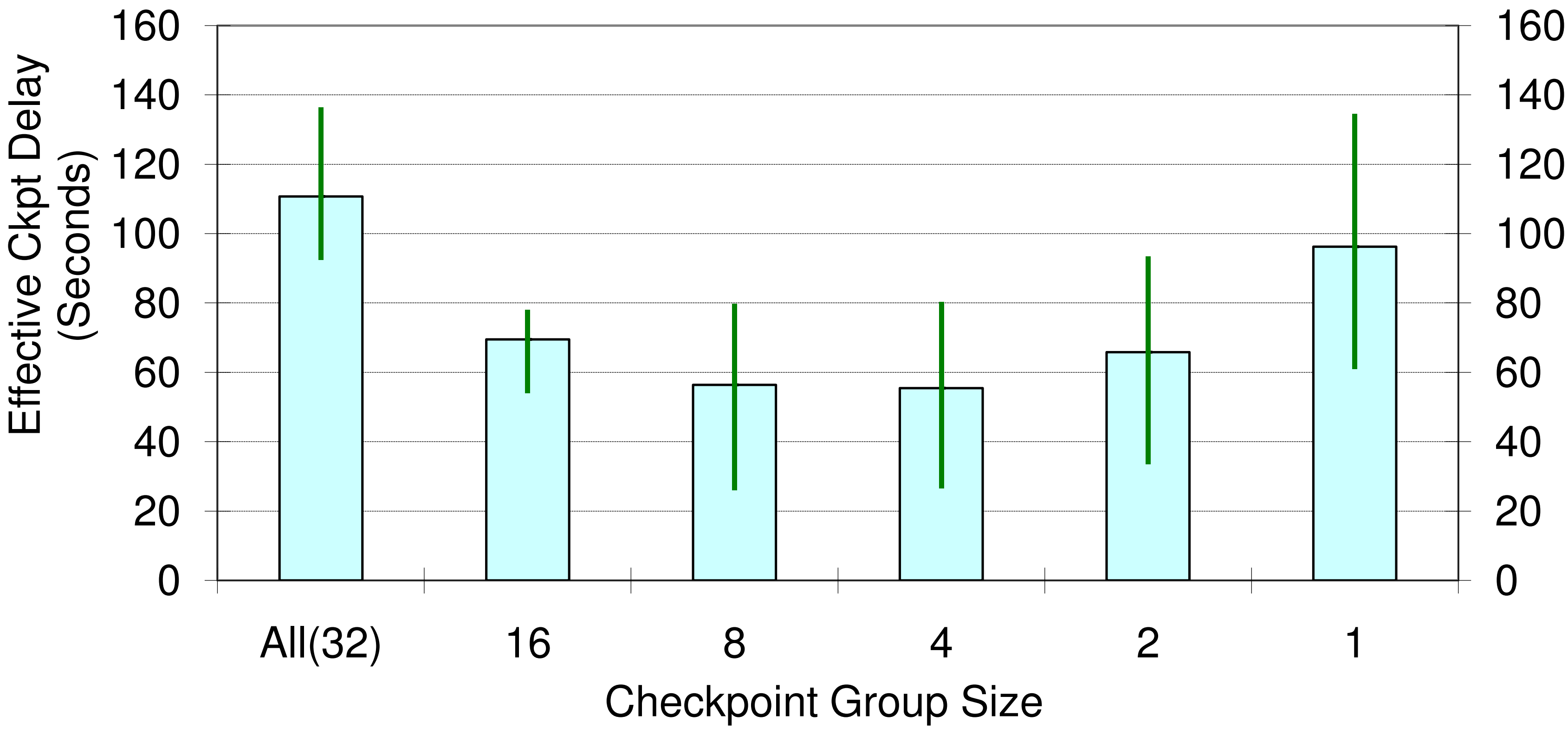
HPL: Solving dense linear system

Configuration:
32 processes, (8 X 4)
Group size is four larger block size.

Up to 78% reduction in effective ckpt delay

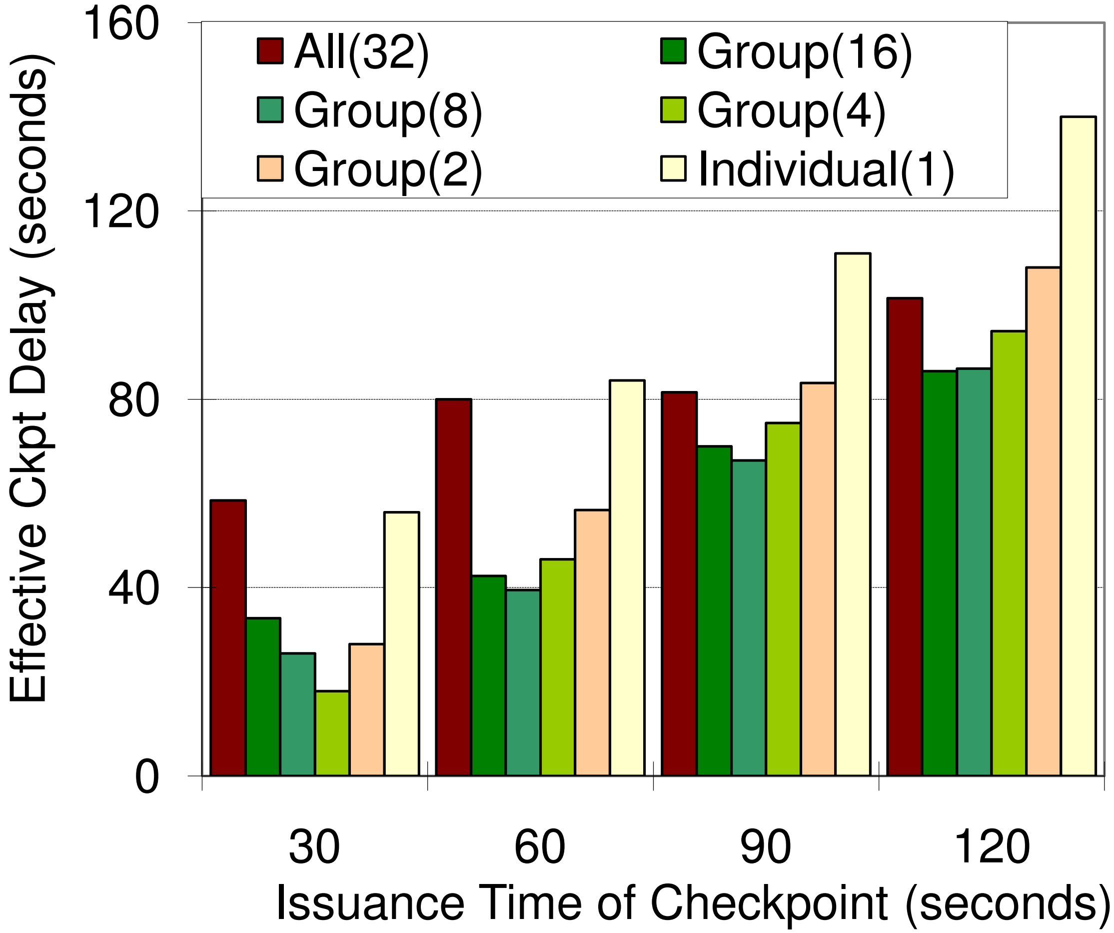
Note: process has different sizes of memory footprint at different time points

High Performance Linpack



Average reduction in delay for group-size 2, 4, 8, 16 are 37%, 46%, 46%, 35%, respectively

Parallel Version of MotifMiner

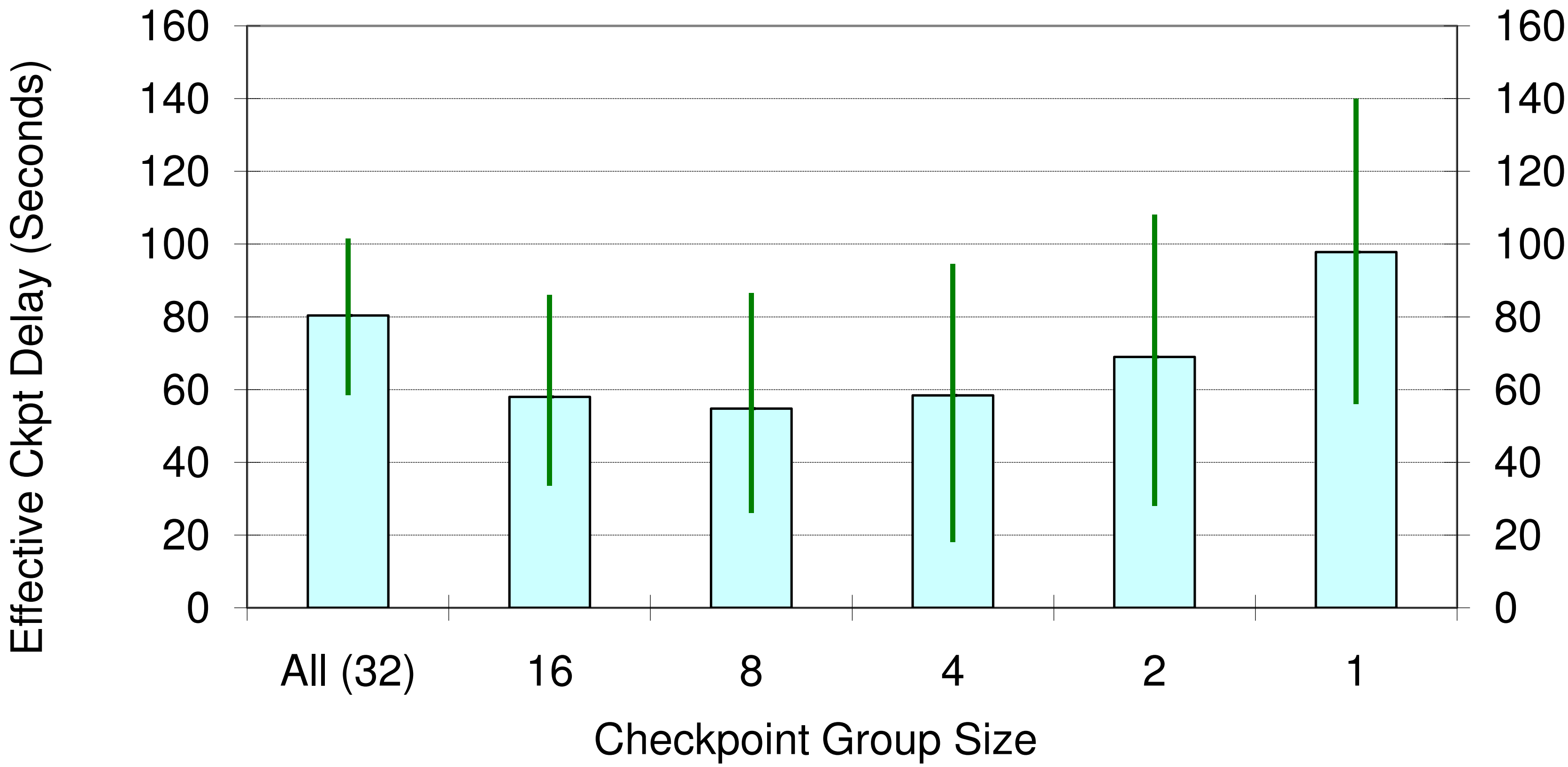


MotifMiner: A data mining toolkit that can mine for structural motifs in a wide area of biomolecular datasets.

Chao Wang and Srinivasan Parthasarathy. "Parallel Algorithms for Mining Frequent Structural Motifs in Scientific Data". In proc of ICS'04

Up to 70% reduction in effective ckpt delay

Parallel Version of MotifMiner



Average reduction in delay for group-size 2, 4, 8, 16 are 14%, 27%, 32%, 28%, respectively

Outline

- Introduction, Background, and Motivation
- Main Idea and Design
- Experimental Platforms
- Performance Results
- Conclusions

Conclusions

- We analyze the scalability limitation of coordinated checkpointing caused by storage bottleneck.
- We present a design of group-based checkpointing to address the scalability limitation.
- We implement the design based on MVAPICH2 and evaluated it using settings similar to production clusters.
- Experimental results show that effective checkpoint delay can be reduced significantly by group-based checkpointing, up to 78% for HPL and 70% for MotifMiner

Acknowledgements

Our research is supported by the following organizations

- Current Funding support by

Office of Science U.S. DEPARTMENT OF ENERGY
NSF
Mellanox Technologies
CISCO SYSTEMS EMPOWERING THE INTERNET GENERATION
intel
Linux Networx proven
NetApp
Sun microsystems

- Current Equipment support by

Mellanox Technologies
intel
AMD
Sun microsystems
APPRO HPC Cluster Solutions
AMMASSO
IBM
Microway
PathScale
SilverStorm

Web Pointers

NBCL home page

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

MVAPICH home page

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

Backup Slides

Level to Implement Checkpointing

- Application level V.S. system level

Application level:
• Application programmers save/restore running states, and handle consistency
• Application specific
• Can only save states at certain points.

System level:
•System provide interfaces to save/restore running states, and automatically handle consistency
• Application independent
• Can save states in any given point.

- Compiler assisted application level checkpointing: application gives hints and library performs checkpoint

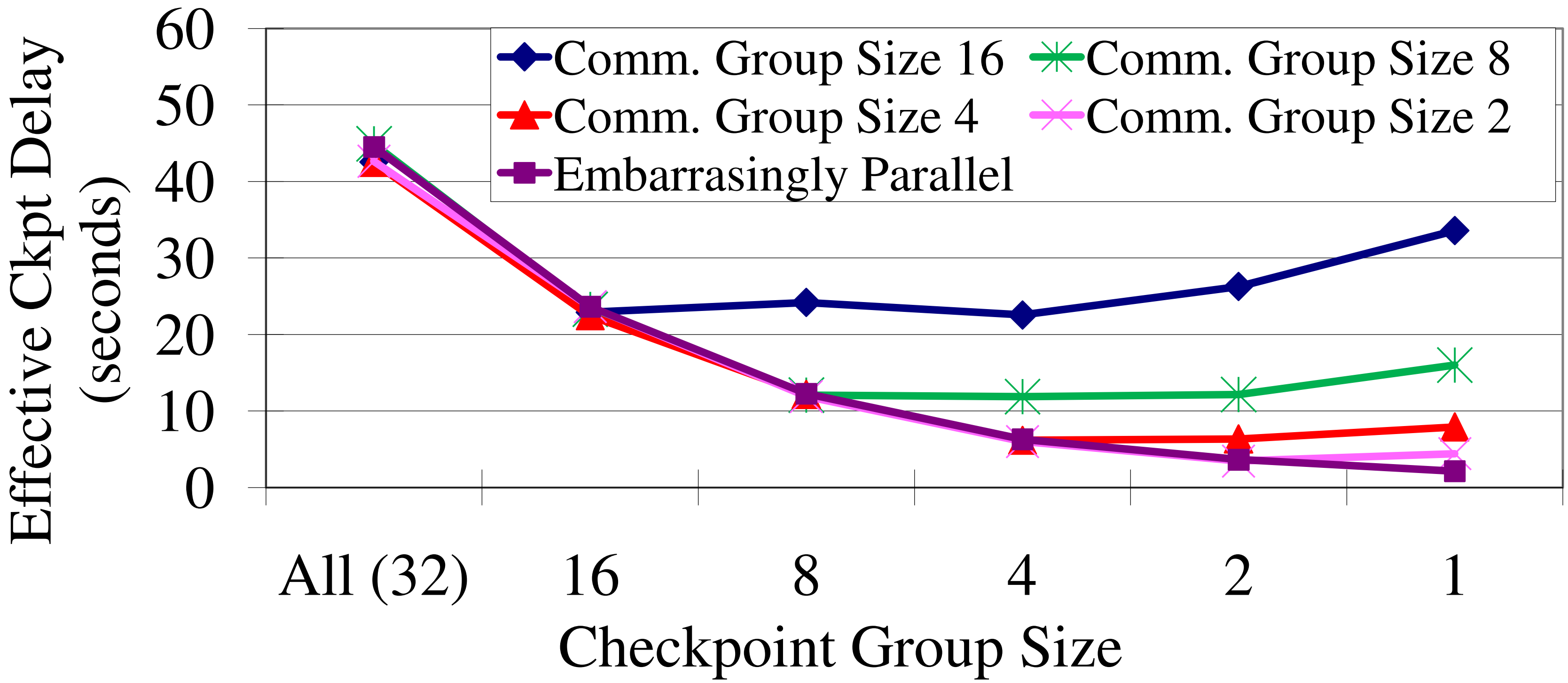
Related Works

- Other checkpointing protocols/designs
  - Uncoordinated checkpointing
  - Causal checkpointing
  - Staggered checkpointing
- Other techniques to reduce checkpoint delay
  - Diskless checkpointing
  - Incremental checkpointing
- On MPI
  - MPICH-V, V2, Vcl, Vcausal, etc.
  - OpenMPI (LAM/MPI, FT-MPI)
  - Charm++ and AMPI

Performance Analysis

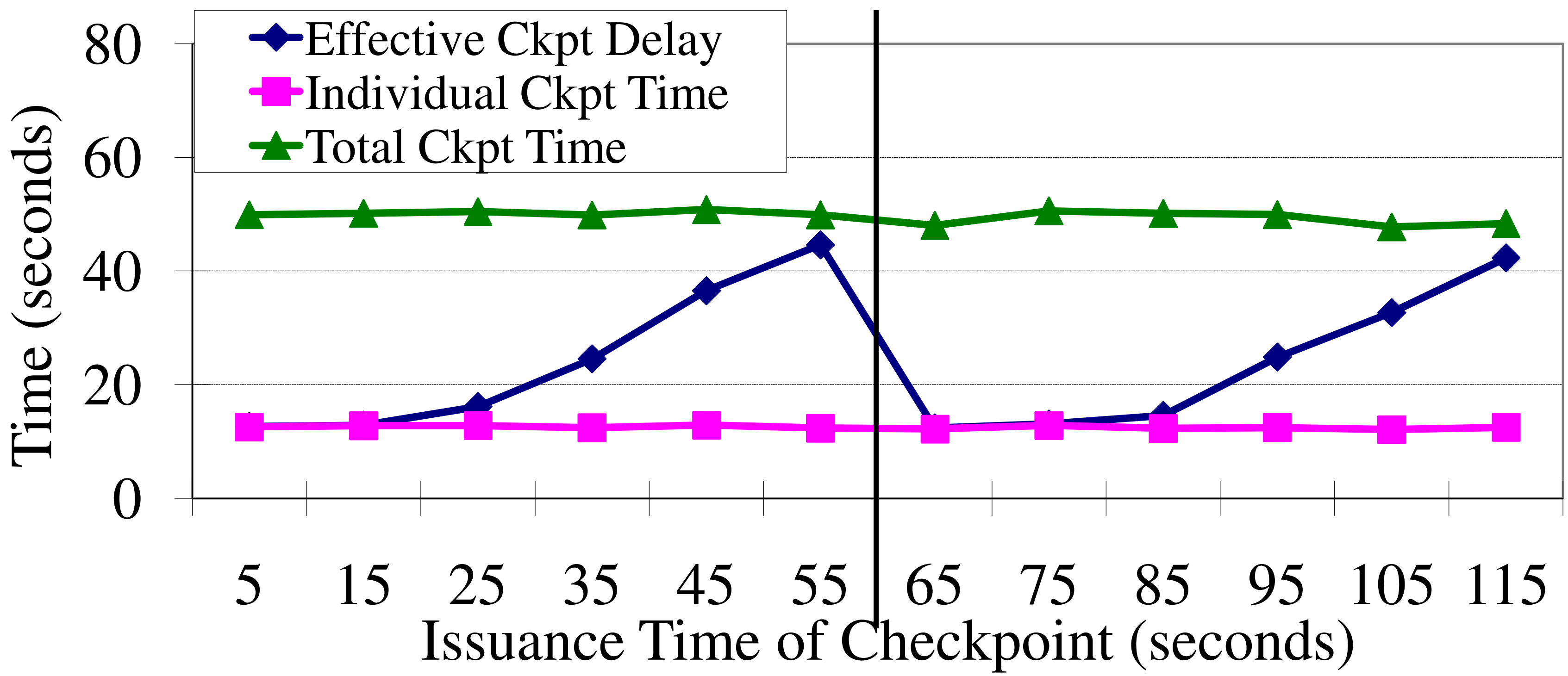
- Performance metrics
  - Effective ckpt delay: the increase in application running time caused by taking a checkpoint
  - Individual ckpt time: the downtime of individual processes for checkpointing, lower bound of effective delay
  - Total ckpt time: the time from ckpt request to ckpt finish, upper bound of effective delay.
- Two main factors affecting performance
  - How checkpointing group size matches with communication group size
  - Checkpoint placement: issuance time of checkpoint request

Checkpoint Group Size



- Processes communicate only within groups continuously with various group sizes.
- When checkpoint group covers more than one communication groups, reducing checkpointing group size will reduce the delay

Checkpoint Placement



- 32 processes, checkpoint group size = communication group size = 8, global barrier every minute.
- When checkpoint is placed close to synchronization point, group-based checkpointing reduces individual ckpt time greatly, but less in effective checkpoint delay.