Fault Tolerant MPI
Protocols and Implementations

Brian J. Argauer
Stephen R. Byers

May 23, 2006

Multiple Processor Systems EECC 756
Dr. Muhammad Shaaban
Outline

- Motivation for Fault Tolerance
- Techniques
  - Checkpoint
  - Message Logging
- Implementations
  - MPICH-V1
  - MPICH-V2
  - MPICH-VCL
- CoCheck Framework
- Conclusion
Fault Tolerant Motivation

- Current trend toward larger clusters, distributed and GRID computing
- Source of Failure
  - Nodes
  - Network
  - Human Factors
- Thousands of nodes reduces MTBF to hours or minutes
Techniques: Checkpoint

- **Capture entire state of task**
  - Application, Stack, Allocated memory, etc.

- **Program Failure**
  - Kill survivors
  - Restart from last consistent and complete set of checkpoints

- **Coordinated**
  - Expensive
  - All tasks stop message passing
  - Write to disks simultaneously
  - Continue Message Passing

- **Uncoordinated**
  - Nodes Checkpoint at different times
  - In-flight messages retained via explicit logging
Techniques: Message Logging

- **Pessimistic Log**
  - Transaction logging
  - No incoherent states can be reached
  - Can handle an unbounded number of faults

- **Optimistic Log**
  - Log messages
  - Assume part of log lost when faults occur
  - Either rollback entire application if too many faults or assume only 1 fault at a time can occur in system

- **Causal Log**
Checkpointing is efficient with low fault frequency!

Message logging is efficient higher fault frequency!
MPICH-V Introduction

- **Traditional MPI**
  - Static Resources
  - Limited Error Handling
  - Node failure stalls or slows down other nodes

- **MPICH-V**
  - Research effort to provide MPI implementation based on MPICH
  - Automatic fault tolerant MPI library
  - Implementations: MPICH-V1, MPICH-V2, MPICH-VCausal, MPICH-VCL

MPICH-V

MPI Implementation for Volatile resources
Fault Tolerant Overview

Automatic
- Co-Check
- Starfish
- clip
- MPICH-VCL
- Optimistic recovery in distributed systems a fault with coherent checkpointing [SY85]
- MPICH-V/Causal
- Sender-Based Message Logging 1 fault, sender-based [ZJ87]
- MPICH-V1/V2
- log based causal log
- pessimistic log

Non-Automatic
- Manetho
- Egida
- FT-MPI
- MPI/FT redundancy of tasks [BNC+01]
- MPICH/FT synchronization server [LNLE00]
MPICH-V1 Goals

- Volatility Tolerance
  - Redundancy
  - Task Migration
- Highly Distributed
  - Scalable
  - Asynchronous Checkpointing
  - No Global Synchronization
- Inter-administration domain communications
  - Security Tools for GRID Deployment
  - Use non-protected relay between client and server nodes if client and server both fire walled
**MPICH-V1 Overview**

- Designed from standard MPI Implementation
- Run existing MPI applications without modification
- Suitable for very large scale computing using heterogeneous networks
- Uncoordinated Checkpoint
- Remote Pessimistic Message Logging
MPICH-V1 Architecture

- **Checkpoint Server**
  - Store and provide task images
  - Images sent to CS as generated by nodes
  - Image clone of running process on given node
MPICH-V1 Architecture

- **Channel Memory**
  - Storage of in-transit messages
  - Repository services

- **Dispatcher**
  - Resource Scheduling
  - Task Management
MPICH-V1 Performance

- Periodic checkpoint every 130s for each task
- Base execution without checkpoint and without fault

- Total execution time (in seconds)

- Number of faults during execution of bt.A.9
Fault Tolerant Overview

Automatic

Co-Check

Optimistic recovery in distributed systems

Manetho

a fault with coherent checkpointing

Egida

[SY85]

[EZ92]

[RAV99]

FT-MPI

Non-Automatic

MPICH-V1/V2

Sender-Based Message Logging

1 fault, sender-based

MPI/FT

redundancy of tasks

BNC+01

MPICH-V/Causal

log based causal log

MPI/FT

communication protocol

[MIL00]

MPICH-VCL

Checkpoint based

optimistic log

(sender based)
MPICH-V2

- Pessimistic Logging for large clusters
- Uncoordinated Checkpoint
- Nodes store messages they send locally
- Event Loggers store sequence of received messages for each node
Fault Tolerant Overview

Automatic
- Optimistic recovery in distributed systems after faults with coherent checkpointing [SY85]
- Manetho on faults [EZ92]
- Egida [RAV99]

Non-Automatic
- FT-MPI

Level
- Co-Check

Framework
- Starfish environment of MPI [AP99]
- CliP semi-transparent checkpoint [CLP97]

API
- MPICH-VCL
- MPICH-V/Causal
- MPICH-V1/V2

Communication Library
- Checkpoint based
- Optimistic log (sender based)
- Log based causal log
- Pessimistic log

Other Method
- Method
- MPI FT redundancy of tasks [BNC+01]
- MPJ EI fault-tolerant server [LNIE00]
MPICH-VCL

- Newest MPICH-V
- Designed for extra low latency dependent applications
- Coordinated Checkpoint
MPICH-VCL Performance

- NAS Benchmark BT Class B, 25 Nodes, Fast Ethernet
- Performance crossover point between checkpoint and message logging: 1 fault every 3 minutes
Fault Tolerant Overview

### Automatic
- Co-Check
- Optimistic recovery in distributed systems a fault with coherent checkpointing [SY85]
- Manetho a fault [EZ92]
- Egida [RAV99]

### Non-Automatic
- FT-MPI
- MPICH-V/Causal
- MPICH-V1/V2
- Sender-Based Message Logging 1 fault sender-based [IZ87]
- MPICH-V/Causal
- MPI/FT redundancy of ticks [BNC+01]
- Other method

Comm. lib
- MPICH-VCL
- Checkpoint based
- Optimistic log (sender based)
- Log based causal log
- Pessimistic log
CoCheck Framework

- Abstraction Framework
- Above message passing layer
- Easily adaptable and portable to different MPI implementations through the use of wrapper functions
- Provides consistency
- Considers checkpointing & process migration
- tuMPI
State Consistency Problem

- Processes A, B, C
- Circles = Events, Arrows = Message Sending
- S, S’, S” = Checkpoint Snapshots
- Notice S” is inconsistent
Clearing Communication Lines

- Uses Coordinator Process
- Sends “Ready-Message” (RM) when checkpoint or migration is needed.
- If process receives RM, assumes no more communication
- Once all RMs are received... can checkpoint or migrate
- On restart... check for messages in buffer
Performance & Future Research

- Single processor migration results
  - vs. num processors & size of checkpoint image
- 8 Machines
  - Mix of Sun SparcStation 2 and Sparc 10
- Dominating Factor
  - Image Size
- Future Considerations
  - Automatic load performance and balancing
Conclusion

- Current trend
  - Increasing cluster size
  - Lower MTBF
  - Fault tolerance increasingly important
- Fault tolerant implementations in MPI offer assortment of solutions
- New research yielding new improvements & ideas to enhance efficiency and robustness of fault tolerant systems.
Questions?

References:

- G. Stellner, CoCheck: Checkpointing and Process Migration for MPI, Institut fur Informatik der Technischen Universität München, München, Germany
- A. Bouteiller, P. Lemarinier, G. Krawezik, F. Cappello, Coordinated Checkpoint versus Message Log for Fault Tolerant MPI, LRI, Université de Paris Sud, Orsay, France
- W. Gropp, E. Lusk, Fault Tolerance in MPI Programs, Argonne National Laboratory, Argonne IL