Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications

Authors:
Md. Mohsin Ali and Peter E. Strazdins

Research School of Computer Science
The Australian National University
Canberra, ACT 0200, Australia

Presented By:
Md. Mohsin Ali

April 24 2013
High Performance Computing

application areas
- Atmosphere, Earth, Environment
- Bioscience, Biotechnology, Genetics
- Chemistry, Molecular Sciences
- Computer Science, Mathematics
- Advanced graphics and virtual reality, etc.

Science and Engineering

Industrial and Commercial
High Performance Computing

- component

- hundreds of thousands of processing elements to concurrently execute

- millions of threads

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications

Md. Mohsin Ali
Size of HPC systems are becoming larger to meet current demand
Size of HPC systems are becoming larger to meet current demand

... But

System size — Probability (component failure)
Size of HPC systems are becoming larger to meet current demand

... But

System size — Probability (component failure)

System size — Hardness of achieving parallelism
Frequency of Failure

*G. Gibson, B. Schroeder, J. Digney, 2007

*G. Gibson, B. Schroeder, J. Digney, 2007

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications  |  Md. Mohsin Ali
Introduction

Frequency of Failure

Reasons of Failure

*G. Gibson, B. Schroeder, J. Digney, 2007*
checkpoint/restart
Ways of Failure Recovery

**checkpoint/restart**

- I/O bottleneck
- Up to 25% of overhead in current petascale systems
Ways of Failure Recovery

repliCation
replication

disadvantages

- need to keep data consistency
- hard to find out proper places for replication
- scalability is degraded
Ways of Failure Recovery

message logging
message logging

- Disadvantages:
  - Same as replication but reduced message size
  - Performance degradation caused by synchronization
Ways of Failure Recovery

Algorithm-based

Standard AMFT

- Read data
- Codify data
- Calculations

Correctness test

if (success)
  acceptance
  test
  output results
algorithm-based

advantages

- error detection, correction, and repeated computation are within the algorithm executing within a processing element (PE)
- errors are propagated on less number of PE
“... partial differential equations (PDEs) are the basis of all physical theorems.”

Bernhard Riemann (1826-1866)
“... partial differential equations (PDEs) are the basis of all physical theorems.”

Bernhard Riemann (1826-1866)

Solution of PDEs

Time-evolving numerical methods
“... partial differential equations (PDEs) are the basis of all physical theorems.”

Bernhard Riemann (1826-1866)

Solution of PDEs

Time-evolving numerical methods

Parallel version for complex PDEs
“... partial differential equations (PDEs) are the basis of all physical theorems.”

Bernhard Riemann (1826-1866)

Solution of PDEs

Time-evolving numerical methods

Parallel version for complex PDEs

Even a single process failure postpone whole computation
“... partial differential equations (PDEs) are the basis of all physical theorems.”

Bernhard Riemann (1826-1866)

Solution of PDEs

Time-evolving numerical methods

Parallel version for complex PDEs

Even a single process failure postpone whole computation

More component on system causes more failure (and more complexity)
Goal

- Design and implementation of time-evolving application
  - tolerate process failure
  - achieve high scalability
- To learn the usability of fault-tolerant semantics of FT-MPI
Challenges

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications

Md. Mohsin Ali
Challenges

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications

Md. Mohsin Ali
How to detect processes failure?
How to detect processes failure?

How to determine which processes are failed?
How to detect processes failure?
How to determine which processes are failed?
How to recover failed processes?
How to detect processes failure?
How to determine which processes are failed?
How to recover failed processes?
How to recover “lost state info” of recovered processes?
How to detect processes failure?
How to determine which processes are failed?
How to recover failed processes?
How to recover “lost state info” of recovered processes?
How to continue time-step from the point of failure?
How to detect processes failure?
How to determine which processes are failed?
How to recover failed processes?
How to recover “lost state info” of recovered processes?
How to continue time-step from the point of failure?
How to retain scalability?
How to detect processes failure?

✓ MPI_ERR_OTHER semantics of FT-MPI
How to detect processes failure?

MPI_ERR_OTHER semantics of FT-MPI

How to determine which processes are failed?

Attribute catching mechanism of MPI
How to detect processes failure?

- MPI_ERR_OTHER semantics of FT-MPI

How to determine which processes are failed?

- Attribute catching mechanism of MPI

How to recover failed processes?

- Creating new processes with the same rank as previous
  - FT_MPI_CHECK_RECOVER and
  - MPI_Comm_dup semantics of FT-MPI
How to recover “lost state info” of recovered processes?

How to continue time-step from the point of failure?

1D advection with periodic boundary condition

Replaced by
How to recover “lost state info” of recovered processes?

How to continue time-step from the point of failure?

Every two-way exchange is going through master and save “state info” on it.

“lost state info” of recovered processes are recovered from master by FT-MPI process restart.

1D advection with periodic boundary condition

Replaced by

worker
How to recover “lost state info” of recovered processes?

How to continue time-step from the point of failure?

“lost state info” of recovered processes are recovered from master by FT-MPI process restart

time-stepping is continued from one step backwards
Sending from Master (Process 0) is Failed
Sending from Master (Process 0) is Failed
Sending from Master (Process 0) is Failed

How to Tackle Challenges
Sending from Master (Process 0) is Failed

How to Tackle Challenges
Sending from Master (Process 0) is Failed
Sending from Worker (Process > 0) is Failed

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications | Md. Mohsin Ali |
How to Tackle Challenges

Sending from Worker (Process > 0) is Failed

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications | Md. Mohsin Ali | 43
How to Tackle Challenges

Sending from Worker (Process > 0) is Failed

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications | Md. Mohsin Ali | 44
How to Tackle Challenges

Sending from Worker (Process > 0) is Failed

![Diagram showing the process of fault tolerance in time-evolving applications](Image)

- Iteration 0
- Iteration 1

- $P_0$
- $P_1$
- $P_2$
- $P_3$
- ... $P_n$

- If receive from resend
- Waiting to receive from $P_2$
- Failed
- Recovery time
- Resend to $P_0$
- (invalid data)
- ...
- ...
- ...
- ...

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications | Md. Mohsin Ali |
Sending from Worker (Process > 0) is Failed

if receive from resend otherwise
waiting to receive from $P_2$

Failed

time

resend to $P_0$
(invalid data)

Algorithm-Based Master-Worker Model of Fault Tolerance in Time-Evolving Applications | Md. Mohsin Ali |
How to Tackle Challenges

How to retain scalability?

✓ Scalability is very low in this master-worker model
Overhead of FT-MPI over Open MPI

- # cores 16 (total)
- # nodes 4 (total)
- Memory 4 GB (each node)
- standard GigE Switch

<table>
<thead>
<tr>
<th># Grid Points</th>
<th># Time Steps</th>
<th>Open MPI (Sec)</th>
<th>FT-MPI (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15360</td>
<td>38400</td>
<td>41</td>
<td>61</td>
</tr>
<tr>
<td>30720</td>
<td>76800</td>
<td>173</td>
<td>204</td>
</tr>
<tr>
<td>46080</td>
<td>115200</td>
<td>382</td>
<td>441</td>
</tr>
</tbody>
</table>
Scalability and Recovery Time

- **Scalability achieved for 16 cores (4 nodes) = 15% (very low)**

- **Recovery time**
  - 1 worker process failed = 1 sec
  - 4 worker processes failed = 2 sec
  - 8 worker processes failed = 3 sec
  - 15 worker processes failed = 5 sec
Checkpointing after each $T$ time-steps on a specific node.
Future Work

Checkpoint after each T time-steps on separate nodes
Conclusion

System size — Probability (component failure)

System size — Hardness of achieving parallelism

- Process failure detection by FT-MPI
- Failed process restart by FT-MPI
- Algorithm-based fault tolerance technique for data recovery
- Overhead of FT-MPI compared to Open MPI is low
- Recovery time is less
- Master-worker model is not so scalable, but can be used as a prototype
Thank You!

Any questions?