ACR: **AUTOMATIC CHECKPOINT/RESTART** FOR **SOFT AND HARD ERROR PROTECTION.**

Xiang Ni, Esteban Meneses, Nikhil Jain, Sanjay Kale

Parallel Programming Lab, UIUC
CONTENTS

☐ MOTIVATION
☐ ACR FRAMEWORK
☐ OPTIMIZATION
☐ EXPERIMENTAL RESULTS
☐ CONCLUSION
BACKGROUND

TASK

NODE A

NODE B

NODE C
BACKGROUND

TASK

LOCAL CHECKPOINT

NODE A

NODE B

NODE C
BACKGROUND

TASK

LOCAL CHECKPOINT

NODE A

NODE B

NODE C
BACKGROUND

TASK

LOCAL CHECKPOINT

GLOBAL CHECKPOINT

NODE A

NODE B

NODE C

BUDDY
New challenge: soft error.

An unintended change in the state of an electronic device that alters the information that it stores without destroying its functionality.

Detectable and correctable: single bit flip

Detectable and uncorrectable: double bit flips

Undetectable, uncorrectable and incorrect program outcome
MOTIVATION

☐ New challenge: soft error.

☐ An unintended change in the state of an electronic device that alters the information that it stores without destroying its functionality.

☐ Detectable and correctable: single bit flip

☐ Detectable and uncorrectable: double bit flips

☐ Undetectable, uncorrectable and incorrect program outcome
MOTIVATION

- Why soft failure rate will increase?
  - Computer electronic’s sensitivity to radiation increases as their dimensions and operating voltage decreases
  - The requirements for high performance and low power.
- What may happen if soft failure rate keeps increasing?
MOTIVATION
MOTIVATION

Utilization
Number of Sockets
Soft Error Rate per Socket (FIT)

Vulnerability

NO FT
MOTIVATION

![Graph showing utilization, vulnerability, and soft error rate per socket. The graphs compare NO FT (normal operation without fault tolerance) and CR (corrective response).]
REPLICATION ENHANCED CHECKPOINTING

**Replica 1**

**Task**
- Node A
- Node B

**Local Checkpoint**
- Node A
- Node B

**Replica 2**

**Task**
- Node A
- Node B

**Local Checkpoint**
- Node A
- Node B
REPLICATION ENHANCED CHECKPOINTING

**Replica 1**
- Task
- Local Checkpoint
  - Node A
  - Node B

**Replica 2**
- Task
- Local Checkpoint
  - Node A
  - Node B
REPLICATION ENHANCED CHECKPOINTING

REPLICA 1

BUDDY

REPLICA 2

TASK

LOCAL CHECKPOINT

NODE A

NODE B

NODE A

NODE B
REPLICATION ENHANCED CHECKPOINTING

Comparison for soft data corruption

REPLICA 1

BUDDY

REPLICA 2

TASK

LOCAL CHECKPOINT

NODE A

NODE B

NODE A

NODE B
REPLICATION ENHANCED CHECKPOINTING

Job Starts

replica 1

T1

transfer checkpoint for soft error detection

replica 2

T2

hard error detected by replica 2

replica 2 sends checkpoints to replica 1 for recovery

T3

soft error detected, both replicas roll back

application execution

checkpoint

recovery

TIME

replica 1

replica 2

Wednesday, November 20, 13
DIFFERENT WAYS TO RESTART FROM HARD ERROR

Strong resilience
Recover from the previous checkpoint
DIFFERENT WAYS TO RESTART FROM HARD ERROR

Medium resilience
Healthy replica schedules an immediate checkpoint
DIFFERENT WAYS TO RESTART FROM HARD ERROR

Weak resilience
Recover from the next checkpoint
AUTOMATIC CHECKPOINT DECISION

☐ “Schedule” immediate checkpoint

☐ Asynchronous application: no barriers

☐ Program will hang after restart if inconsistent states are

☐ Solution: asynchronous consensus based on scheme available in Charm++*

*MENON ET.AL “AUTOMATED LOAD BALANCING INVOCATION BASED ON APPLICATION CHARACTERISTICS” CLUSTER, 2012
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE
LOCAL MAXIMUM PROGRESS

TASK A

1

TASK B

1
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

TASK A

1

TASK B

1
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

TASK A

1

TASK B

1
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

ACR: FINDING MAXIMUM ITERATIONS

TASK A

1

2

TASK B

1
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

ACR: FINDING MAXIMUM ITERATIONS

TASK A

1

2

TASK B

1
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

ACR: FINDING MAXIMUM ITERATIONS

TASK A

1

2

TASK B

1
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

ACR: FINDING MAXIMUM ITERATIONS

PAUSE

TASK A

1

2

TASK B

1

Wednesday, November 20, 13
AUTOMATIC CHECKPOINT DECISION

ACR: UPDATE LOCAL MAXIMUM PROGRESS

ACR: FINDING CHECKPOINT ITERATIONS

CHECKPOINT ITERATION DECIDED

1

PAUSE

Wednesday, November 20, 13
AUTOMATIC CHECKPOINT DECISION

- Task A
  1. ACR: UPDATE LOCAL MAXIMUM PROGRESS
  2. ACR: FINDING CHECKPOINT ITERATIONS
  3. ACR: BROADCAST CHECKPOINT ITERATION 3

- Task B
  1. 
  2. PAUSE

Wednesday, November 20, 13
AUTOMATIC CHECKPOINT DECISION

**Task A**
1. ACR: Update Local Maximum Progress
2. PAUSE
3. ACR: Broadcast Checkpoint Iteration 3
4. ACR: Ready to Checkpoint

**Task B**
1. ACR: Finding Checkpoint Iterations
2. 3

Wednesday, November 20, 13
AUTOMATIC CHECKPOINT DECISION

1. ACR: UPDATE LOCAL MAXIMUM PROGRESS
2. ACR: FINDING CHECKPOINT ITERATIONS
3. ACR: BROADCAST CHECKPOINT ITERATION 3

Task A:
1. 1
2. PAUSE
3. 3

Task B:
1. 1
2. 2
3. 3
BASE PERFORMANCE

![Bar chart showing base performance with number of cores per replica on the x-axis and time in seconds on the y-axis. The chart includes a checkpoint feature.]

Number of Cores per Replica

Time (s)

JACOBI3D BGP
BASE PERFORMANCE

![Graph showing time vs number of cores per replica for JACOBI3D BGP](image)

- Number of Cores per Replica: 1k, 2k, 4k, 8k, 16k
- Time (s) on the y-axis: 0, 0.5, 1, 1.5, 2, 2.5, 3
- checkpoint

Wednesday, November 20, 13
OPTIMIZATION: TOPOLOGY AWARE MAPPING
OPTIMIZATION: TOPOLOGY AWARE MAPPING

(a) Default-mapping
1) Reduce the inter-replica communication distance.
2) Trade-off between inter and intra replica communication.

(a) Default-mapping

(b) Column-mapping

(c) Mixed-mapping
OPTIMIZATION: CHECKSUM

- TRANSFER THE CHECKSUM OF 1 INTEGER INSTEAD OF THE WHOLE CHECKPOINTS
- FLOATING POINT ROUND-OFF ERROR
## EXPERIMENTAL RESULTS: MINI-APPLICATIONS

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Description</th>
<th>Configuration per core</th>
<th>Memory Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi3D</td>
<td>7-point stencil</td>
<td>64<em>64</em>128</td>
<td>High</td>
</tr>
<tr>
<td>HPCCG</td>
<td>Unstructured implicit finite element method</td>
<td>40<em>40</em>40</td>
<td>High</td>
</tr>
<tr>
<td>LULESH</td>
<td>Unstructured explicit mesh</td>
<td>32<em>32</em>64</td>
<td>High</td>
</tr>
<tr>
<td>LeanMD</td>
<td>Short-range non-bonded force calculation in NAMD</td>
<td>4000 atoms</td>
<td>Low</td>
</tr>
<tr>
<td>miniMD</td>
<td>Mimic the performance in LAMMPS</td>
<td>1000 atoms</td>
<td>Low</td>
</tr>
</tbody>
</table>
EXPERIMENTAL RESULTS: CHECKPOINT

**JACOBI3D AMPI**

**LEANMD**
EXPERIMENTAL RESULTS: CHECKPOINT

**LULESH**

**HPCCG**

**MINIMD**
EXPERIMENTAL RESULTS: RESTART

JACOB13D AMPI

LEANMD
EXPERIMENTAL RESULTS: RESTART

![Graphs showing experimental results for LULESH, HPCCG, and MINIMD.](image-url)
• Failures are injected according to Weibull process: shape parameter 0.6
• 19 failures are injected in a 30mins Jacobi3d run. 14 failures are injected in the first half while 5 failures are injected in the second half.
• Real failures injected: node becomes irresponsive
• Automatic restart: with the support of spare nodes and thus no need to submit the job again
• Runtime system observes the change in failure rate based on the failure history.
• Adjusts the optimum checkpoint period based on Daly’s model.
• Checkpoint period changes from 6s in the beginning to 17s in the end.
ADAPTING TO FAILURES

- Runtime system observes the change in failure rate based on the failure history.
- Adjusts the optimum checkpoint period based on Daly’s model.
- Checkpoint period changes from 6s in the beginning to 17s in the end.
ADAPTING TO FAILURES

• Runtime system observes the change in failure rate based on the failure history.
• Adjusts the optimum checkpoint period based on Daly’s model.
• Checkpoint period changes from 6s in the beginning to 17s in the end.
CONCLUSION

- Protection for both hard errors and silent data corruption is important
- Replication enables synergistic handling of errors
- Good scalability
- Automatic checkpoint decision
  - Restart from hard errors
  - Adapting to different failure distributions
THANKS!

☐ QUESTIONS?

☐ CONTACT AT xiangni2@illinois.edu

☐ MORE INFO AT http://charm.cs.illinois.edu/research/ft
OVERHEAD

Overhead per Replica (%) vs. Number of Sockets per Replica for JACOBI3D and LEANMD.

- JACOBI3D
- LEANMD

Number of Sockets per Replica:
- 1k
- 4k
- 16k

Overhead per Replica (%):
- Default
- Default + Checksum
- Column
- Column + Checksum

Categories:
- Strong
- Medium
- Weak
MODELING OF UTILIZATION AND VULNERABILITY

Utilization per Replica vs Number of Sockets per Replica

Weak $\delta = 15s$
Medium $\delta = 15s$
Strong $\delta = 15s$
Weak $\delta = 180s$
Medium $\delta = 180s$
Strong $\delta = 180s$

Probability of Undetected Soft Errors vs Number of Sockets per Replica

Weak $\delta = 180s$
Medium $\delta = 180s$
Weak $\delta = 15s$
Medium $\delta = 15s$