Impact of Non-optimal Checkpoint Intervals on Overall Application Efficiency in Cluster Computing: A Simulation-Based Study

William M. Jones, PhD
Computer Science Department
Coastal Carolina University
Myrtle Beach, South Carolina

http://www.parl.clemson.edu/beosim
The Issue

- Application efficiency
  - Throughput
  - Turnaround times
  - Heavy loads
- Optimal checkpoint intervals
- AMTTI estimation
  - Running estimate
- Sensitivity to error
  - Non-optimal checkpoint interval
- Simulation-based study
  - Discrete event-driven
  - LANL's Pink Cluster

Parameter Estimation

Checkpoint Interval

Application Efficiency

Throughput, TAT, etc
LANL's PINK Cluster Workload

PINK (01/2007 to 11/2007) Job Stats

- 1916 compute nodes
- 50K jobs arrive to queue
- Poisson arrival process
- Run-times extrapolated

<table>
<thead>
<tr>
<th>Num CPUs</th>
<th>% Total Jobs</th>
<th>% Total Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>128</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>256</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>512</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>1024</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
PINK Failure Model (Assumptions)

- Independent
- MTBF / node
- Single failure mode
- One failure
  - Halts at most 1 job
- Zero repair time
  - Job added to HOQ
- Checkpoint / Restart
  - 10 min dump time
  - 10 min restart time
- Simplistic
  - First pass
Checkpointing Metrics:
Average Application Efficiency

\[
\text{Avg. } AE = \frac{\sum_{i=0}^{N-1} Ts_i n_i}{\sum_{i=0}^{N-1} Tr_i n_i} = 0.75
\]

\[
\text{Avg. } AE = \frac{\sum_{i=0}^{N-1} Ts_i}{N} = 0.8
\]
Setting Checkpoint Interval For Each Job in Simulation

\[ AMTTI_i = \frac{T}{n_i}, \quad T = \frac{1}{\lambda} \]

\[ Tc_i = \sqrt{2} \delta AMTTI_i \]

\[ Tc_i = Tc_i \times Err \]

\[ (Tc_i < \delta) \lor (Tc_i > Ts_i) \rightarrow \text{no cp'ing} \]
Application Efficiency vs Checkpoint Interval vs System MTBF
LANL Pink Simulation w/ DumpTime/Rerstart=10mins, 50K jobs

Weighted Avg. App. Efficiency (solve time/run time)

Err (i.e. ratio of CP interval used to "optimal" CP interval)

SMTBF's
250 min
230 min
210 min
190 min
170 min
150 min
130 min
110 min
90 min
70 min
50 min
30 min
Application Efficiency vs Checkpoint Interval vs System MTBF
LANL Pink Simulation w/ DumpTime/Restart=10mins, 50K jobs

~8% drop

LANL Resilience Seminar – July 2009
Back Calculate Loss of Application Efficiency as Function of Error in AMTTI

\[ T_{c,i} = \sqrt{2} \delta AMTTI_i \]

Assume AMTTI is the sole source of error in Tc

A factor of 2 error in Tc would result from a factor of 4 error in AMTTI
Loss In App. Efficiency vs Error In AMTTI Estimate vs System MTBF
LANL Pink Simulation w/ DumpTime/Restart=10mins, 50K jobs

Loss In Weighted Avg. App. Efficiency (w.r.t. Eff. at Optimal CP Interval)

Ratio of AMTTI Estimate Used to Actual AMTTI Calculation

SMTBF’s
230 min
230 min
210 min
190 min
170 min
150 min
130 min
110 min
90 min
70 min
50 min
30 min
Sensitivity of an Application to Accuracy of the MTTI Estimate

Solid Lines: Approximate Optimum Checkpoint Interval, Dotted Lines: Exact

\[ \frac{M}{\delta} = \begin{array}{ccccccccccccc}
1200 & 200 & 60 & 25 & 12.5 & 6.25 & 3.25 & 1.75 & 1 & 0.5 \\
\end{array} \]

\[ t_c = \sqrt{2\delta M} \]

- Checkpoint Efficiency
- \( \frac{\tilde{M}}{M} = \frac{\text{MTTI (estimated)}}{\text{MTTI (actual)}} \)
Daly states “… we do not need to be overly concerned about the precision of our checkpoint restart interval approximation […] or our measurement of its dependent dump time and application MTTI parameters.”
Application Efficiency Still Matters

Heavy System Load

Dramatic Impact on Queue Time

Turnaround Time = Queue Time + Execution Time

Besides

Energy Usage

But what about at larger scales where down time >> AMTTI?
Future Work

- New cluster (larger systems) runs
  - Realistic failure stats
- Add new failure modes
  - Failures that impact more than one job
  - AMTTI << SMTBF
- Add AMTTI on-line parameter estimator
  - Initialize to a guess
  - As AMTTI is refined, Tc becomes closer to optimal
- Map jobs with resilience in mind
  - Schedule contiguous across switch, etc

PART 2 of Presentation
Checkpoint / Migration Job Scheduling Next
Parallel Job Scheduling

- Local execution
- Job migration
- **Job co-allocation**
  - Map across boundaries
  - Sharing resources
  - Network BW contention
- Can help or hurt
- Some example scenarios ...
Multi-site Co-allocation

Cluster 1

Cluster 2

Processor Pool

Job 1

Job 2

Internet
Scheduling w/o co-allocation

Cluster 1
- Running job
- Waiting job 1

Cluster 2
- Running job
- Waiting job 2

Makespan without co-allocation
Scheduling w/ co-allocation

makespan without co-allocation

makespan with co-allocation

improvement

Cluster 2

running job

Cluster 1

waiting job 1

running job

waiting job 2

waiting job 1

waiting job 2

Time
Co-allocation w/ slowdown

makespan without co-allocation

makespan with co-allocation and some NW congestion

makespan with co-allocation and no NW congestion

improvement

execution slowdown

Time

Cluster 1

running job

waiting job 1

waiting job 2

Cluster 2

running job

waiting job 1

waiting job 2
Idealized Multi-cluster Model

- Internal switches
- Central switch
- Clusters
IPC Pattern Model

All-to-all personalized

2D
How Much Info Is “Enough”

A1 performance w.r.t. Information Availability

Max improvement possible: 36%
A1 Best: 31%
Summary of Previous Results
Improvement Over Migration Only w/ No Estimate Inaccuracy

% improvement

Max (0% Error)

150 Mbps
300 Mbps
400 Mbps
What happens if there is significant inaccuracy in the user-provided bandwidth estimates?
Turnaround Time vs Info w/ PPBW = 150 Mbps

Job turnaround time

Information Availability (% of jobs)
Turnaround Time vs Info vs Error w/ PPBW = 300
Turnaround Time vs Info w/ PPBW = 300 Mbps

Job turnaround time

Information Availability (% of jobs)

Migration Only
@ +/- 100% Error
@ +/- 80% Error
@ +/- 50% Error
@ +/- 0% Error
Impact of +/- 100% Estimate Inaccuracy

% Improvement Over Migration Only

Max (0% Error)

w/o CPing

150 Mbps

300 Mbps

400 Mbps

7/6/09

LANL Resilience Seminar – July 2009
What potential gain might there be to employ checkpointing and run-time job migration to mitigate network over-subscription?
Turnaround Time vs CP Overhead vs Error w/ PPBW = 150 Mbps

Job Turnaround Time (mins)
Turnaround Time vs Error w/ PPBW = 150 Mbps

+/-% Error in User Estimate

Job Turnaround Time

Equivalent RMS % Error in User Estimate

- w/o CPing
- CPing w/ OH=00 min
- CPing w/ OH=05 min
- CPing w/ OH=10 min
- CPing w/ OH=20 min
- CPing w/ OH=30 min
- Migration Only
Turnaround Time vs CP Overhead vs Error w/ PPBW = 400 Mbps

Job Turnaround Time (mins)

CP overhead (mins)

+/-% Error in User Estimate
Turnaround Time vs Error w/ PPBW = 400 Mbps

+/-% Error in User Estimate

Job Turnaround Time

Equivalent RMS % Error in User Estimate

- w/o CPing
- CPing w/ OH=00 min
- CPing w/ OH=05 min
- CPing w/ OH=10 min
- CPing w/ OH=20 min
- CPing w/ OH=30 min
- Migration Only
Results Summary at 5 and 10 Minute CP Overhead at 100% Estimate Error

- Max (0% Error)
- CPing (5 min)
- CPing (10 min)
- w/o CPing

% Improvement Over Migration Only

<table>
<thead>
<tr>
<th>Speed</th>
<th>150 Mbps</th>
<th>300 Mbps</th>
<th>400 Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>22</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>CPing (5 min)</td>
<td>20</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>CPing (10 min)</td>
<td>15</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>w/o CPing</td>
<td>10</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Turnaround Time vs CP Overhead w/ PPBW=150 Mbps

@ 80% Error w/o CPing
@ 80% Error w/ CPing
@ 100% Error w/o CPing
@ 100% Error w/ CPing
Migration Only

Job Turnaround Time (mins)

Checkpoint Overhead (mins)
Turnaround Time vs CP Overhead w/ PPBW=300 Mbps

Job Turnaround Time (mins)

@ 80% Error w/o CPing
@ 80% Error w/ CPing
@ 100% Error w/o CPing
@ 100% Error w/ CPing
Migration Only

Checkpoint Overhead (mins)
Thank you!

Questions?

William M. Jones
http://www.parl.clemson.edu/beosim
What could some error be 'good'?

\[ T = Q + E \]
Simulation Framework

- Parallel Job Model
  - Computation
  - Communication

- Multi-cluster Model
  - ICN
  - Nodes

- Workload Generator

- Queueing System

- Intelligent Schedulers
  - Job characterization
  - Network topology
  - Fairness policies

- Performance Evaluation
  - Job turnaround time
  - Fairness
Initial Results
Algorithm Comparison, Synthetic Workload

Job turnaround time (sec)

First-Fit
B1
B3
A1
Zero Comm. Cost
Migration Only

PPBW (Mbps)
# Algorithm Run-Time Analysis

<table>
<thead>
<tr>
<th>Module</th>
<th>Complexity</th>
<th>Time (μSec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>$O(n \cdot p^m)$</td>
<td>69.6</td>
</tr>
<tr>
<td>B1</td>
<td>$O(m \cdot \log(m) + n)$</td>
<td>0.98</td>
</tr>
<tr>
<td>B2</td>
<td>$O(m \cdot \log(m) + n)$</td>
<td>0.89</td>
</tr>
<tr>
<td>B3</td>
<td>$O(m \cdot \log(m) + n)$</td>
<td>1.28</td>
</tr>
<tr>
<td>B4</td>
<td>$O(p)$</td>
<td>1.10</td>
</tr>
</tbody>
</table>
Ensuring Fairness

- Disparate workload intensities
- Different cluster sizes
- Unfair resource sharing
- Overload remote clusters
- Worse than not participating
- Technique to control fairness

Diagram:

- Cluster 1: $Y >> X \text{ jobs/day}$
- Cluster 2: $Y >> X \text{ jobs/day}$
- Cluster 3: $Y >> X \text{ jobs/day}$
- Cluster 4: $X \text{ jobs/day}$
Fairness Via Conservative Backfilling

- Out of order execution
- No delay to start time
- Two-tiered approach
- Backfill local jobs first
- Local backfill schedules
- Consider remote job backfill
- Constrain non-local node use
- Prevents local job starvation
Fairness, 3 Clusters (100, 100, 100) w/ Increasing Load, 1 Cluster (100) w/ Fixed Load

Job turnaround time (sec)

A1 on G
A1+B on G
A1+B on FC
A1+B on OC
No Share on FC

Arrival Rate on the first 3 clusters (Jobs/Week/Cluster)
What about fault-tolerance?

Suppose you could detect that an error occurred, migrate the job, and restart the job from last checkpoint.

How quickly would you need to determine that an interrupt occurred?
Fig. 4. Turnaround time as a function of CPdelta at various failure rates. *Note that as failures become more frequent, the overall performance becomes more sensitive to reductions in the detection latency, CPdelta.*
Fig. 6. Job runtime as a function of CPdelta. Here the initial job runtime distribution is the same for all classes of jobs. This is done to illustrate the expected relationship between MTBF and execution time due to multiple single-application interruptions. Note the similar improvement due to a reduction in CPdelta.