Modeling and Simulation of Error Recovery in Concurrent Processing Systems

Dr. Silvia Heubach

Department of Mathematics and Computer Science

California State University Los Angeles
Introduction

- Model and Parameters
- Iterative versus Selective Rollback
- Simulation
- Cost Derivation
- Results of Comparison
- Conclusion
Model Description

- Global checkpointing with $n$ processes
- Inter-communications between processes after exponential times with parameters $\lambda_{ij}$
- Failures after exponential times with parameters $\phi_i$
- Acceptance tests after exponential times with parameters $\alpha_i$
Iterative versus Selective Rollback

Iterative Rollback
- Rolls back to most recent checkpoint (CP) and attempts recovery
- If recovery fails from CP k, processes are rolled back to CP k-1

Selective Rollback
- Selects checkpoint for recovery based on distribution of latency times
- Pairs of checkpoints are compared for smaller expected cost of recovery

Both methods recover eventually if failure occurred after CP 1
Cost of Recovery from Checkpoint \( k \)

- \( CT = cycle\ time = C + CL \)
  - \( C \) = time between checkpoints
  - \( CL \) = time to load checkpoint

- \( tot = \#\ of\ currently\ established\ checkpoints \)

- \( d = time\ between\ acceptance\ test\ and\ last\ checkpoint \)

- \( T(k) = cost\ of\ recovery\ from\ CP\ k \)
  - \( T(k) = (tot - k) CT + CL + d \)
Total Cost of Recovery (Iterative Rollback)

- CP \( r \) = checkpoint to which processes must be rolled back for recovery
  - Failure between CP \( k \) and CP \( k+1 \)
    \( \Rightarrow r = k \)
  - Failure before CP \( 1 \)
    \( \Rightarrow r = 1 \)

- Total cost of recovery
  \( TOI = T(r) + T(r+1) + \ldots + T(\text{tot}) \)
Selective Rollback

- Choice of checkpoint based on latency distribution

- Simulation creates empirical distribution function for latency distribution

- Simulation based on events with exponential waiting times
  - next event after exponential time with parameter

\[ \mu = \sum_{j=1}^{n} \sum_{i=1, i\neq j}^{n} \lambda_{ij} + \sum_{i=1}^{n} \varphi_i + \sum_{i=1}^{n} \alpha_i \]
Simulation

- 5 Simulations (up to time 200 hours)

- Ranges for average times (in hours) between events

- Simulation based on events with exponential waiting times
  - Inter-communications [0.25, 1]
  - Failures [10, 40]
  - Acceptance tests [1, 2]
Simulation

Sample parameter values for simulation 1

<table>
<thead>
<tr>
<th>$\lambda_{ij}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>3.50</td>
<td>1.85</td>
<td>3.92</td>
</tr>
<tr>
<td>2</td>
<td>1.84</td>
<td>*</td>
<td>2.30</td>
<td>3.30</td>
</tr>
<tr>
<td>3</td>
<td>3.34</td>
<td>1.16</td>
<td>*</td>
<td>2.80</td>
</tr>
<tr>
<td>4</td>
<td>3.77</td>
<td>3.05</td>
<td>1.43</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\varphi_i$</th>
<th>$\alpha_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.080</td>
<td>0.087</td>
</tr>
<tr>
<td>2</td>
<td>0.098</td>
<td>0.949</td>
</tr>
<tr>
<td>3</td>
<td>0.080</td>
<td>0.971</td>
</tr>
<tr>
<td>4</td>
<td>0.066</td>
<td>0.770</td>
</tr>
</tbody>
</table>
Latency Times

- Combined Latency Times

- Latency Distribution
  - median ~ .43
Latency Times

\[ P(k) := P(\text{failure occurred after CP } k) = P(\text{latency} < (\text{total} - k + 1) \ C) \]

\[ C(l,k) = \text{cost of successful recovery from CP } l \text{ given that the first recovery attempt starts at CP } k > l \]

\[ EC(k) = \text{expected cost of recovery given recovery from CP } k \text{ was unsuccessful} \]

\[ EC(k) = [P(1) - P(2)] C(1,k-1) + [P(2) - P(3)] C(2,k-1) + \ldots + [P(k-1) - P(k)] C(k-1,k-1) \]
Selective Rollback Algorithm

■ **Step 1** (Initialization)
  - \( k = \min \{ m, \text{total} \}, \ P = 0, \ \text{TOS} = 0 \)

■ **Step 2** (Termination Condition)
  - If \( k = 1 \), go to Step 4

■ **Step 3** (Comparison)
  - Compare expected cost of recovery from CP \( k \) and \( k-1 \). If cost from CP \( k-1 \) smaller, then \( k := k-1 \); go to Step 2

■ **Step 4** (Rollback; updating of variables)
  - Roll back to CP \( k \) and attempt recovery
  - \( P := P(k), \ \text{TOS} = \text{TOS} + T(k) \)
  - If recovery successful, resume normal operation; otherwise
    - if \( k > 1 \), go to Step 2
    - else indicate that recovery is not possible
Selective Rollback Algorithm

■ Step 3 (Comparison)
  – If \((P(k-1) - P) T(k-1) + (1 - P(k-1)) EC(k-1) \leq (P(k) - P) T(k) + (1 - P(k)) EC(1)\), then \(k:=k-1\); go to Step 2

■ Step 4 (Rollback; updating of variables)
  – \(P:= P(k), \ TOS = TOS + T(k)\)

  • If recovery is unsuccessful from CP \(k\), the probabilities \(P(l)\) are replaced by conditional probabilities \((P(l)-P(k))/(1-P(k))\) and the expected values \(EC(k)\) are likewise divided by \((1-P(k))\). This is achieved by setting \(P = P(k)\).

  • Cost is updated to reflect the accumulated cost
Cost Comparison

- Number of checkpoints selected to achieve a given recovery level (0.9 or 0.95) using quantiles of latency distribution

- Values for C ranging from 0.05 to 0.45 (~ median of latency distribution)

- 30 simulations for each level of recovery and choice of C

- If no recovery was possible, simulation stopped
Results $C = 0.1$

Selective $\mathbb{H}$L vs. Iterative $\mathbb{H}$L Roll Back
$C = 0.1$ Recovery Level = 0.9

Selective $\mathbb{H}$L vs. Iterative $\mathbb{H}$L Roll Back
$C = 0.1$ Recovery Level = 0.95
Results for Averaged Cost vs. C

Average Selective ΗL vs. Average Iterative ΗL
Roll Back at Recovery Level = 0.9

Average Selective ΗL vs. Average Iterative ΗL
Roll Back at Recovery Level = 0.95
Conclusion

- Selective rollback has smaller or equal cost in all cases
- Difference most pronounced for small cycle time
- Checkpoint selection
  - m = total number of checkpoints
  - \( \hat{m} \) = checkpoint to which processes roll back

<table>
<thead>
<tr>
<th></th>
<th>90%</th>
<th></th>
<th></th>
<th>95%</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>m</td>
<td>( \hat{m} )</td>
<td>C</td>
<td>m</td>
<td>( \hat{m} )</td>
</tr>
<tr>
<td>.05</td>
<td>19</td>
<td>4</td>
<td></td>
<td>.05</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>.10</td>
<td>11</td>
<td>4</td>
<td></td>
<td>.10</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>.15</td>
<td>8</td>
<td>4</td>
<td></td>
<td>.15</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>.20</td>
<td>6</td>
<td>3</td>
<td></td>
<td>.20</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>.25</td>
<td>5</td>
<td>3</td>
<td></td>
<td>.25</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>.30</td>
<td>5</td>
<td>3</td>
<td></td>
<td>.30</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>.35</td>
<td>4</td>
<td>2</td>
<td></td>
<td>.35</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>.40</td>
<td>4</td>
<td>4</td>
<td></td>
<td>.40</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>.45</td>
<td>4</td>
<td>4</td>
<td></td>
<td>.45</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Practical Issues

- Simulation can be enhanced by dynamically adapting the latency distribution

Actual Implementation

- Initially use iterative method with small C to create data for approximate latency distribution

- Use approximate distribution function to implement selective rollback
Future Work

- Error bounds on difference between actual latency distribution and approximate distribution

- Theoretical distribution and related parameter estimation?