Balancing Speculative Loads in Parallel Discrete Event Simulation

Eric Mikida
Brief PDES Description

• Simulation made up of Logical Processes (LPs)
• LPs process events in timestamp order
• Synchronization is conservative or optimistic
• Periodically compute global virtual time (GVT)
Optimistic Execution
Optimistic Execution
Optimistic Execution
Optimistic Execution
Optimistic Execution
Optimistic Execution
Performance Metrics

Event Rate = $\frac{E_{\text{committed}}}{s}$

Event Efficiency = $\frac{E_{\text{committed}}}{E_{\text{total}}}$

Load Balance = ??
What is “load”? 

• Charm++ automatically measures CPU time 
  – Makes sense when all work is useful work 
  – Relies on principle of persistence 
  – Balances CPU time per PE
What is “load”?

• Charm++ automatically measures CPU time
  – Makes sense when all work is useful work
  – Relies on principle of persistence
  – Balances CPU time per PE

Does this make sense in a speculative setting?
Example

<table>
<thead>
<tr>
<th>LP 1</th>
<th>LP 2</th>
<th>LP 3</th>
</tr>
</thead>
</table>
| Executed: 5  
Committed: 4  
Rolled Back: 0 | Executed: 5  
Committed: 0  
Rolled Back: 4 | Executed: 5  
Committed: 0  
Rolled Back: 0 |
Example

LP 1
- Executed: 5
- Committed: 
- Rolled Back:

LP 2
- Executed: 5
- Committed: 
- Rolled Back:

LP 3
- Executed: 5
- Committed: 
- Rolled Back:
Example

LP 1
- Executed: 5
- Committed: 4
- Rolled Back:

LP 2
- Executed: 5
- Committed: 0
- Rolled Back:

LP 3
- Executed: 5
- Committed: 0
- Rolled Back:
Example

LP 1
- Executed: 5
- Committed: 4
- Rolled Back: 0

LP 2
- Executed: 5
- Committed: 0
- Rolled Back: 4

LP 3
- Executed: 5
- Committed: 0
- Rolled Back: 0
Example

<table>
<thead>
<tr>
<th>LP 1</th>
<th>LP 2</th>
<th>LP 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executed: 5</td>
<td>Executed: 5</td>
<td>Executed: 5</td>
</tr>
<tr>
<td>Committed: 4</td>
<td>Committed: 0</td>
<td>Committed: 0</td>
</tr>
<tr>
<td>Rolled Back: 0</td>
<td>Rolled Back: 4</td>
<td>Rolled Back: 0</td>
</tr>
</tbody>
</table>

Roughly the same CPU time spent executing events
Example

LP 1
- Executed: 5
- Committed: 4
- Rolled Back: 0

LP 2
- Executed: 5
- Committed: 0
- Rolled Back: 4

LP 3
- Executed: 5
- Committed: 0
- Rolled Back: 0

Roughly the same CPU time spent executing events
How does the load balancer differentiate?
How does GVT affect balance?

Count-Based GVT
• GVT computed every X events
• Doesn’t attempt to bound optimism
• Can lead to poor event efficiency

Leash-Based GVT
• GVT computed every X units of virtual time
• Keeps virtual times balanced across PEs
• Can lead to poor CPU balance across PEs
Benchmarks

**PHOLD**
- Common PDES benchmark
- Executing an event causes a new event for a random LP
- Changing event distribution causes imbalance

**Traffic**
- Simulates a grid of intersections
- Events are cars arriving, leaving, and changing lanes
- Cars travel from source to destination
GVT Trigger Event Rates

64 nodes of Vesta (BG/Q)

PHOLD:
- 32 LPs per rank
- 50% remote events

Traffic:
- 64K intersections
- 1M cars
GVT Trigger Comparison

Efficiency (%)

<table>
<thead>
<tr>
<th>PHOLD</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Leash</td>
</tr>
</tbody>
</table>

64 nodes of Vesta (BG/Q)

PHOLD:
- 32 LPs per rank
- 50% remote events

Traffic:
- 64K intersections
- 1M cars
Our Load Balancing Goal

• Make sure all PEs have useful work
  – Balance the CPU load
  – Only count useful work

• Maintain a high event efficiency
  – Balance rate of progress
  – Leads to less overall work
Redefine “Load” for PDES

**Past-Looking Metrics**
- CPU Time
- Current Timestamp
- Committed Timestamp
- Committed Events
- Potential Committed Events

**Future-Looking Metrics**
- Next Timestamp
- Pending Events
- Weighted Pending Events
- “Active” Events
PHOLD Event Rate

Event Rate (millions/s)

Count

Leash

2.5x Speedup

4.5x Speedup

Event Rate (millions/s)

No LB

CPU Time

Committed TS

Current TS

Next TS

Committed Events

Pending Events

Active Events

2.5x Speedup

4.5x Speedup

Count

Leash
PHOLD Efficiency

<table>
<thead>
<tr>
<th>Efficiency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
</tr>
<tr>
<td>Leash</td>
</tr>
</tbody>
</table>

- No LB
- CPU Time
- Committed TS
- Current TS
- Next TS
- Committed Events
- Pending Events
- Active Events
Traffic Event Rate

Event Rate (millions/s)

Count

Leash

3x Speedup

3x Speedup

3x Speedup

Event Rate

No LB
CPU Time
Committed TS
Current TS
Next TS
Committed Events
Pending Events
Active Events
Traffic Efficiency

Efficiency (%)

- Count
- Leash

- No LB
- CPU Time
- Committed TS
- Current TS
- Next TS
- Committed Events
- Pending Events
- Active Events
What’s next?

• Better visualization/analysis tools
• More diverse set of models
• Conservative synchronization
• Vector load balancing strategies
• Adaptive load balancer
• Combine with GVT work