Parallel Simulation of Queueing Petri Nets

Jürgen Walter, Simon Spinner, Samuel Kounev
Dept. of Computer Science, University of Würzburg

SimuTools Conference,
August 25th 2015, Athens, Greek
Queueing Petri Nets are used for performance modeling and analysis.

Desire for performance prediction at run time.

Multi-core-processors are standard, but SimQPN is still sequential.
Queueing Petri Nets

- Queueing Petri Nets (QPN)
  - Petri Nets (PN)
  - Queueing Networks (QN)
- Model Parts
  - Places, Transitions, Token, Queues

Queueing Petri Net Modeling Environment (QPME)

- SimQPN
  - Batch/means
  - Replication/deletion

http://tools.descartes/qpme

Motivation  Foundations  Approach  Case Studies  Conclusions
Concurrent Simulation

- Concurrent Simulation
  - Parallel Simulation
  - Distributed Simulation

- Logical Process (LP)

- Synchronization
  - Conservative
  - Optimistic

- Lookahead

Focus on parallel simulation

Simulate subparts of simulation model

Motivation Foundations Approach Case Studies Conclusions

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
How to Parallelize Simulation

APPROACH
Parallelization Levels

**Application Level**
- Parallel execution of different simulation runs [Pawlinkowski94]

**Functional Level**
- Execution of helper functions (e.g. random number generation) parallel to simulation
- For basic mathematical models the existence of helper functions is an indicator for inefficient code [Jürgens97]

**Event Level**
- Parallel execution of one simulation run
  - Decomposition into Logical Processes
  - Lookahead
  - Synchronization

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
Decomposition

- Spatial decomposition
- Minimum regions [Chiola93]
- Merging rules [Chiola93]
Token emittance hard to predict for several queueing strategies

Solution: Presampling of scheduling times [Wagner89]
  - Limit number of tokens
  - Lower bound on service time distribution
Parallel simulation works on a theoretical basis for every kind of model.

However:
- Event processing in few microseconds
- Synchronization overhead is too high for multiple models

Fujimoto: „Parallel Simulation: Will the field Survive?“
What works in Practice

- Closed workload models

- Open workload models
  - Can be processed similar to a batch process
  - No predecessor \( \Rightarrow \) When to synchronize?
    - Technical Solution: Virtual time steps
  - Conservative parallelization to reduce overheads
Virtual Time Steps

Motivation
Foundations
Approach
Case Studies
Conclusions

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
Process Overview

Parallel Simulation

Motivation Foundations Approach Case Studies Conclusions

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
Decomposition

- Decomposition into Minimum Regions
- Merge Workload Generators
- Merge Cyclic Connected LPs
- Merge LPs with Ordinary Input Places into Predecessors
- Merge LPs Until Number of LPs < Number of Cores

- Minimization of Communication
- Prevention of Overflow / Balancing of Synchronization
- Transformation to DAG
- Minimization of Communication
- Balancing of Synchronization
Java SE Barriers perform bad on small time slices

Barrier synchronization in Java [Ball03]
  - active wait + hierarchical barriers

Barrier synchronization available at:
http://net.cs.uni-bonn.de/wg/cs/applications/jbarrier/
Contributions

- QPN decomposition into a directed acyclic graph
  - Applicability of existing Petri Net rules
  - Introduction of a merging algorithm that merges cyclically connected subparts, applies some rules and finally merges greedy

- Parallel simulation method optimized for open workload models

- Implementation of parallel SimQPN version
  - Application level
  - Event level
Evaluation

CASE STUDIES
Case Study: Application Level

- Similar curve for all tested models
Case Study: Small Model

- Model provided by a big cloud provider
- Even more reduced …
Case Study: Small Model

- Model provided by a big cloud provider
- Average speedup 1.91
Case Study: SPECj App Server

- Decomposition with heuristics into four logical processes
- Speedup of 2.45 but we expect decomposition not to be optimal
### Network Model

<table>
<thead>
<tr>
<th>Number of Threads / LPs</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup</td>
<td>1.61</td>
<td>1.92</td>
<td>2.22</td>
</tr>
</tbody>
</table>

Motivation - Foundations - Approach - Case Studies - Conclusions

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
Case Study: Artificial Model

- Model Choice
  - Speedup heavily depends on model characteristics
  - Use of a generated model
  - Example shows 3x2 model

Motivation
Foundations
Approach
Case Studies
Conclusions

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
Synchronization Interval Length

- Model: 6 x [length of the lane]
- Less synchronization, higher speedup
- Speedup depends on model

Case Study: Artificial Model
Case Study: Artificial Model

Barrier Contention

- Model: [number of lanes] x 10
- More LPs, more contention for the barrier
Summary

- **Actions**
  - Survey of techniques
  - Parallel simulation engine
    - Event level (for open workload models)
    - Application level

- **Benefits**
  - Parallel simulation runs faster than sequential.
  - SimQPN is applicable to more scenarios.

- **Future Work**
  - Improve decomposition
  - Apply to more case studies
Thank You!

Code & more info:
http://tools.descartes/qpme

juergen.walter@uni-wuerzburg.de
http://se.informatik.uni-wuerzburg.de
<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Title</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reference</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>