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 modelling 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

[Bause93a] [Bause93b]

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
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
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
Theory versus Practice

- 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 ➔ When to synchronize?
    - Technical Solution: Virtual time steps
  - Conservative parallelization to reduce overheads
Virtual Time Steps

Motivation
Foundations
Approach
Case Studies
Conclusions
Process Overview

Motivation
Foundations
Approach
Case Studies
Conclusions

Jürgen Walter - Parallel Simulation of Queueing Petri Nets
Decomposition

- Decomposition into Minimum Regions
  - Minimization of Communication
- Merge Workload Generators
  - Prevention of Overflow / Balancing of Synchronization
- Merge Cyclic Connected LPs
  - Transformation to DAG
- Merge LPs with Ordinary Input Places into Predecessors
  - Minimization of Communication
- Merge LPs Until Number of LPs < Number of Cores
  - Balancing of Synchronization

Motivation  Foundations  Approach  Case Studies  Conclusions
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 Graph
  - Applicability of existing Petri Net rules
  - Introduction of a greedy merging algorithm that merges cyclically connected subparts

- Parallel Simulation method optimized for open workload models

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

- QPN lookahead improvement by the use of queueing network best practices
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
<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>
Model Choice

- Speedup heavily depends on model characteristics
- Use of a generated model
- Example shows 3x2 model
Case Study: Artificial Model

Synchronization Interval Length

- Model: $6 \times \text{[length of the lane]}$
- Less synchronization, higher speedup
- Speedup depends on 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>Details</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reference</td>
<td>Title</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
Related Work

- Queueing Networks
  - [Wagner89] [Lin90]

- Concurrent Simulation Algorithms
  - [Fujimoto99] [Fujimoto00]

- Parallel QPN Simulation

- Petri Nets
  - [Chiola93], [Nketsa01], [Fang07] and [Jürgens97]
To model an unlimited queue with several token colors, it is necessary to create a GSPN with unlimited number of places to represent the waiting line.
GSPN Models of Queueing Stations

M/M/1/3-FCFS queue with 2 token colors:

- \((4 + B + B \times \text{NumColors})\) places needed for queue capacity \(B\)!
- For non-exponentially distributed firing delays or multiple servers, further places/transitions would be needed.
Decomposition

- Structure Analysis
  - Branching
  - Lane
My Research in Short

- **Problem:**
  - Desire for increased QPN analysis speed
  - Sequential QPN simulation can not exploit multi core hardware

- **Idea**
  - Provide a parallel simulation engine for QPNs

- **Benefit**
  - Simulation runs faster
  - Improved applicability at runtime scenarios

- **Actions**
  - Identify suitable parallelization techniques
  - Implement these techniques
  - Evaluate the performance improvement
Evaluation

- Validation
- Test for Speedup
Validation

- Basic Validation Idea
  - Baseline sequential simulator
  - Fixed seed for pseudo random number generators
  - Models from case studies

- Application Level

- Event Level
  - Explainable deviations as expected
Least squares regression based on Utilization Law

Known measurable

- $U_{↓i}$ average utilization in measurement period $i$
- $X_{↓i,c}$ average throughput of workload class $c$ in measurement period $i$

Resource demand $D_{↓c}$ of workload class $c$

Utilization Law for $C$ workload classes:

$$U_{↓i} = X_{↓i,1} \cdot D_{↓1} + \ldots + X_{↓i,C} \cdot D_{↓C}$$