Darmstadt University of Technology
Databases & Distributed Systems Group

Performance Modeling of Distributed E-Business Applications using Queueing Petri Nets

Samuel Kounev and Alex Buchmann

{skounev,buchmann}@informatik.tu-darmstadt.de

DAGSTUHL Graduiertenkolleg Workshop, June 18, 2003

Motivation

- Modern **E-Business Systems** are gaining in size and complexity, which makes it difficult for deployers to estimate the size and capacity of the deployment environment needed to meet SLAs.
- Deployers are faced with questions such as the following:
  - Does the system scale? Are there potential system bottlenecks?
  - What is the maximum load level that the system is able to handle?
  - What would the avg. response time, throughput and utilization be under the expected workload?

Motivation (contd.)

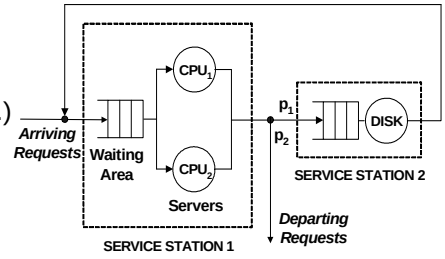
- The main problem is **how to predict system performance under a particular workload**. Performance models are increasingly used for this purpose.
- **Queueing Networks** and **Generalized Stochastic Petri Nets** are among the most popular models exploited, but they both have some serious disadvantages.
- In this paper we look at a new modelling formalism – **Queueing Petri Nets (QPNs)**, which eliminates these disadvantages.
- We study a real-world e-business system and show how QPN models can be exploited for performance analysis.

Agenda

- Traditional Queueing Networks and Petri Nets
- (Hierarchical) Queueing Petri Nets
- Case Study: SPECjAppServer2001
- QPN Performance Models
- Model Analysis and Validation
- Summary and Conclusions

Queueing Networks (QNs)

- **QN**: Set of interconnected queues
- **Queue** = waiting area and servers
- Scheduling strategies(FCFS,PS,...)
- Single-class vs. multi-class
- Open, closed or mixed

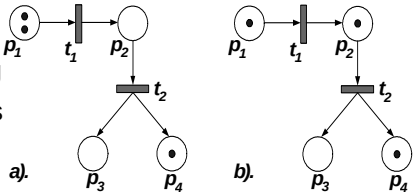


PROS: Very powerful for modelling **hardware contention** and scheduling strategies. Many efficient analysis techniques available.

CONS: Not suitable for modelling blocking, synchronization, simultaneous resource possession and **software contention** in general. Although *Extended QNs* provide some limited support for the above, they are very restrictive and inaccurate.

Petri Nets (PNs)

- **PN**: places, tokens and transitions. marking, transition enabling/firing
- **CPNs**: allow tokens of different colors and transition modes
- **GSPNs**: allow timed transitions
- **CGSPNs**: CPNs + GSPNs

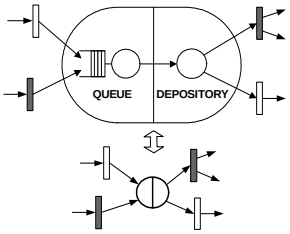


PROS: Suitable both for qualitative and quantitative analysis. Lend themselves very well to modelling blocking, synchronization, simultaneous resource possession and software contention.

CONS: No direct means for modelling scheduling strategies. Not as many algorithms/tools for efficient quantitative analysis are available as for Queueing Networks.

Queueing Petri Nets (QPNs = QNs + PNs)

- Introduced by **Falko Bause** in 1993.
- Combine Queueing Networks and Petri Nets
- Allow integration of queues into places of PNs
- Ordinary vs. Queueing Places
- **Queueing Place** = Queue + Depository



PROS: Combine the modelling power and expressiveness of QNs and PNs. Facilitate the modelling of both hardware and software aspects of system behavior in the same model.

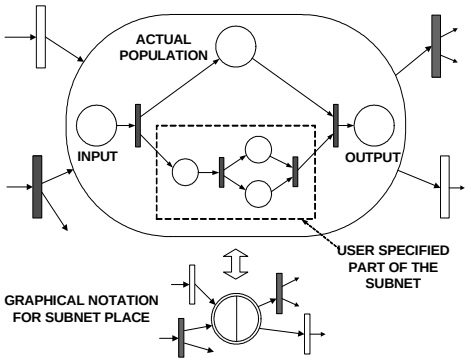
CONS: Extremely difficult to analyze! Analysis suffers the **state space explosion** problem and this imposes a limit on the size of the models that are analyzable.

Hierarchical Queueing Petri Nets (HQPNs)

- Allow hierarchical model specification
- **Subnet Place**: contains a nested QPN
- Structured analysis methods alleviate the state space explosion problem and enable larger models to be analyzed.

Analysis Tools for HQPNs

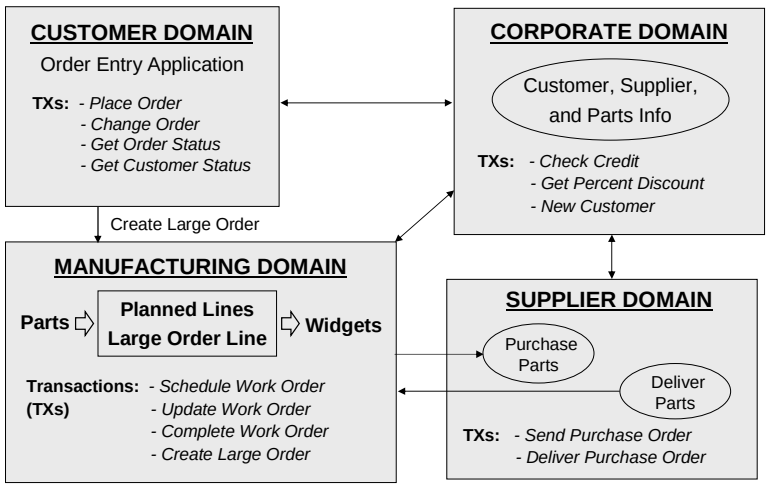
Currently only one tool available:
The **HQPN-Tool** from the University of Dortmund.
Supports a number of structured analysis methods.
Available free of charge for non-commercial use.



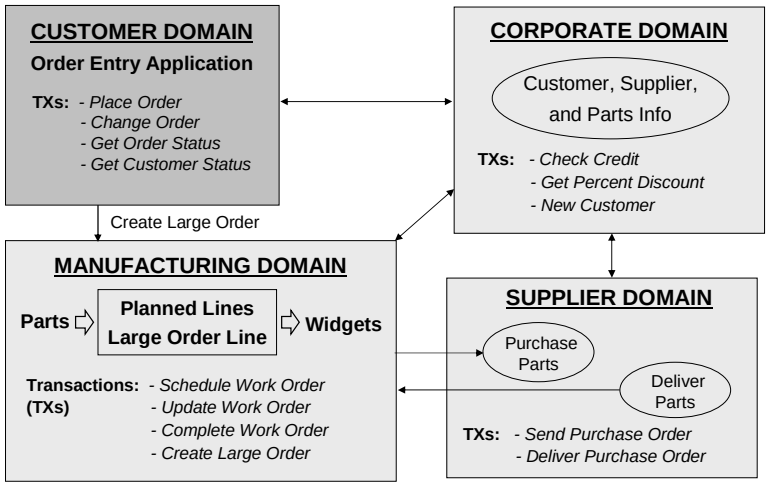
The SPECjAppServer2001 Benchmark

- Heavy-duty B2B E-Commerce Benchmark
- Successor of Sun's ECperf™ 1.1 Benchmark
- Measures performance and scalability of J2EE App. Servers
- Developed by **SPEC OSG Java Subcommittee.**
- For more info visit: http://www.spec.org/osg/jAppServer/

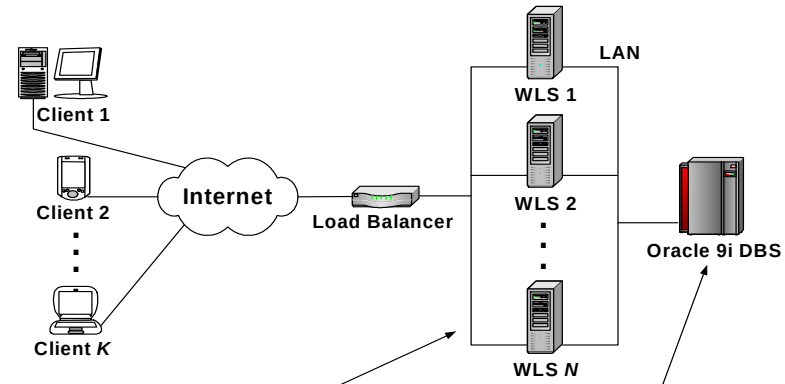
SPECjAppServer2001 Business Model



Case Study: Order Entry Application



Deployment Environment



WebLogic Server 7.0 Cluster
Each node equipped with:
AMD XP 2000+ CPU, 1 GB RAM
Running on SuSE Linux 8.0

Oracle 9i (9.0.1) Database Server
Hosting the SPECjAppServer DB
1,7 GHz AMD XP CPU, 1 GB RAM
Running on Red Hat Linux 7.2

Capacity Planning Issues

We are interested in finding answers to the following questions:

- What level of performance does the system provide under load?
- Average response time, throughput and utilization = ?
- Are there potential system bottlenecks?
- How many application servers would be needed to guarantee adequate performance?

Need also optimal values for the following configuration parameters:

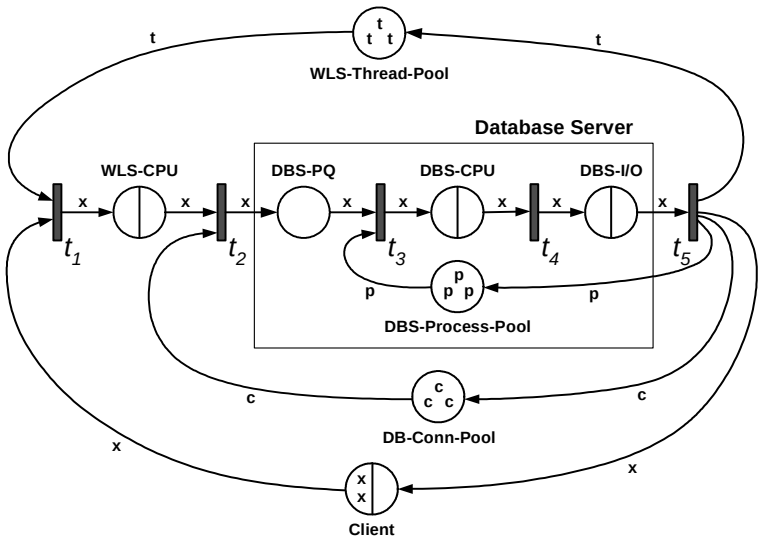
- Number of **threads** in WebLogic (WLS) thread pools
- Number of **connections** in WLS database connection pools
- Number of **processes** of the Oracle server instance

Workload Characterization

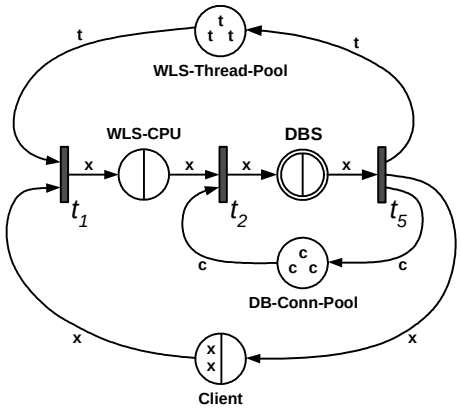
1. Describe the types of requests (**request classes**) that arrive at the system: NewOrder, ChangeOrder, OrderStatus, CustStatus.
2. Identify the hardware and software **resources used** by each request class: **HW:** WLS-CPU, Network, DBS-CPU, DBS-Disk, **SW:** WLS Thread, DB Connection, DBS Process.
3. Measure the total service time (**service demand**) of each request class at each processing resource:

TX-Type	WLS-CPU	DBS-CPU	DBS-I/O
NewOrder	70ms	53ms	12ms
ChangeOrder	26ms	16ms	6ms
OrderStatus	7ms	4ms	0ms
CustomerStatus	10ms	5ms	0ms

First Cut System Model

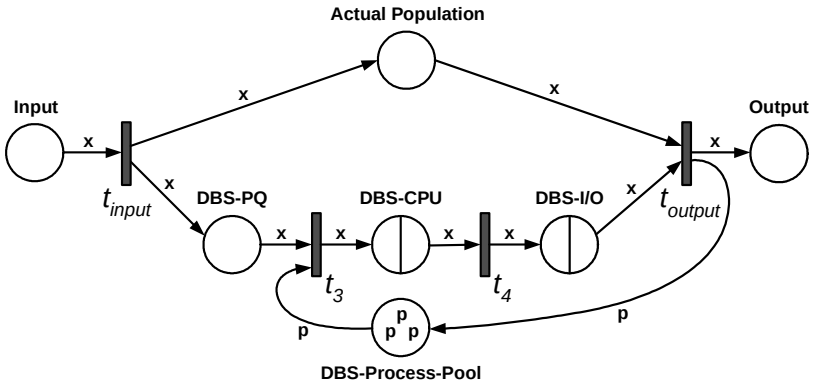


Hierarchical System Model: High-Level QPN



- We isolate the database server and model it using a separate QPN, represented by subnet place „DBS“ above.
- The above QPN is called **High-Level QPN (HLQPN)** of our hierarchical model.

Hierarchical System Model: Low-Level QPN



- The nested DBS subnet of our HQPN - called **Low-Level QPN (LLQPN)**.
- Places Input, Output and Actual Population are standard for each subnet.

Scenario 1: Single Request Class

- Single request class – the NewOrder TX
- 80 concurrent clients with avg. client think time of 200ms
- 60 WLS Threads, 40 JDBC Connections, 30 Oracle processes

PLACE	$\mathcal{N}$	$\mathcal{U}$	$\mathcal{X}$	$\mathcal{R}$ $[ms]$
Client_Q	2.85	0.94	14.28	200
Client_D	17.14	1.00	-//-	1200
WLS-CPU_Q	56.67	1.00	-//-	3967
WLS-CPU_D	0.00	0.00	-//-	0
DBS-PQ	0.00	0.00	-//-	0
DBS-CPU_Q	3.11	0.75	-//-	218
DBS-I/O_Q	0.20	0.17	-//-	14
WLS-Thread-Pool	0.00	0.00		
DB-Conn-Pool	36.67	1.00		
DBS-Process-Pool	26.67	1.00		

⇦ Analysis Results

Modelling Error ⇨

METRIC	Model	Measured	Error
WLS-CPU Utilization	100%	100%	0%
DBS-CPU Utilization	75%	65%	15%
NewOrder Throughput	14.28	13.43	6.3%
NewOrder Resp.Time	5399ms	5738ms	5.9%
Thread Queue Length	17.14	18	4.7%

Scenario 1a: Same, but only with 40 Threads

PLACE	$\mathcal{N}$	$\mathcal{U}$	$\mathcal{X}$	$\mathcal{R}$ $[ms]$
Client_Q	2.85	0.94	14.28	200
Client_D	37.14	1.00	-//-	2601
WLS-CPU_Q	36.67	1.00	-//-	2568
WLS-CPU_D	0.00	0.00	-//-	0
DBS-PQ	0.00	0.00	-//-	0
DBS-CPU_Q	3.11	0.75	-//-	218
DBS-I/O_Q	0.20	0.17	-//-	14
WLS-Thread-Pool	0.00	0.00		
DB-Conn-Pool	36.67	1.00		
DBS-Process-Pool	26.67	1.00		

⇦ Analysis Results

Modelling Error ⇨

METRIC	Model	Measured	Error
WLS-CPU Utilization	100%	100%	0%
DBS-CPU Utilization	75%	65%	15%
NewOrder Throughput	14.28	13.41	6.4%
NewOrder Resp.Time	5401ms	5742ms	5.9%
Thread Queue Length	37.14	40	7.1%

- More contention for threads, but less contention for CPU time.
- In both cases, we can reduce the number of DB connections and DBS processes, since they are not effectively utilized.

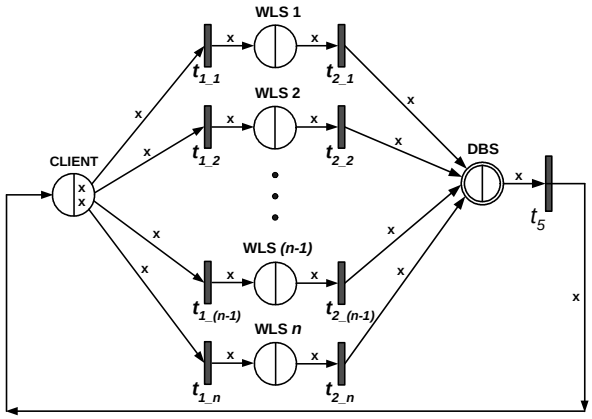
Scenario 2: Multiple Request Classes

- Two request classes – *NewOrder* and *ChangeOrder*
- Some simplifications needed to avoid explosion of the Markov Chain
- Assume that there are plenty of JDBC connections and DBS processes
- Drop places DB-Conn-Pool and DBS-Process-Pool
- 20 clients: 10 NewOrder and 10 ChangeOrder, Avg. think time = 1 sec
- Only 10 WLS Threads

METRIC	Model	Measured	Error
WLS-CPU Utilization	76%	77%	1.2%
DBS-CPU Utilization	54%	64%	15.6%
Avg.free WLS-Threads	6.68	7	4.5%
NewOrder Throughput	7.45	7.47	0.2%
NewOrder Resp. Time	341ms	318ms	7.2%
ChgOrder Throughput	9.22	9.15	0.7%
ChgOrder Resp. Time	84ms	104ms	19.2%

Scenario 3: Multiple Application Servers

- We modify the HLQPN to include multiple WLS places
- 30 NewOrder clients with avg. think time of 1 sec
- No contention for JDBC connections, DBS processes and WLS threads



Scenario 3: Modelling Error

METRIC	Model	Measured	Error
For 2 Application Servers			
WLS-CPU Utilization	64%	68%	6%
DBS-CPU Utilization	96%	91%	5%
NewOrder Throughput	18.28	17.56	4%
NewOrder Resp. Time	640ms	693ms	8%
For 3 Application Servers			
WLS-CPU Utilization	43%	44%	2%
DBS-CPU Utilization	98%	97%	1%
NewOrder Throughput	18.42	17.61	5%
NewOrder Resp. Time	623ms	673ms	7%

Summary and Conclusions

- QPN models enable us to **integrate both hardware and software aspects of system behavior** in the same model.
- Combining the expressiveness of Queueing Networks and Petri Nets, QPNs are not just powerful as a specification mechanism, but are also **very powerful as a performance analysis and prediction tool**.
- However, if this power is to be exploited to its full potential, **improved solution methods and software tools for QPNs are needed**, which enable larger models to be analyzed.

Thank You for Your Attention!

Questions?