Performance Modeling and Evaluation of Distributed Component Systems using Queueing Petri Nets

Samuel Kounev
OPERA Group, Systems Research Group
University of Cambridge – Computer Laboratory

May 4, 2007
PEPA Club, LFCS, University of Edinburgh

Roadmap

- Research Interests
- Introduction to Queueing Petri Nets
- Performance Modeling Methodology
- Case Study: Modeling SPECjAppServer2004
- QPN Modeling Environment (QPME) Demo
- Concluding Remarks
Research Interests

1. Performance modeling and evaluation
2. System sizing and capacity planning
3. Software performance engineering
4. Benchmarking and experimental performance analysis
5. Performance tuning and optimization
6. Autonomic computing and self-managed systems

Target Domains

1. Distributed component-based systems
2. Enterprise middleware
3. Java EE and related technologies
4. Large-scale e-business applications
5. Event-based systems
6. Grid computing environments
7. XML-based Web services
8. Service Oriented Architectures
9. RFID and EPCglobal-related applications
Current Projects

1. Performance modeling and evaluation of event-based systems
2. SPECjms2007 - benchmark for message-oriented middleware
3. Autonomic QoS management in Grid computing and SOA using online performance models
4. QPME - Queueing Petri Net Modeling Environment
5. Performance and scalability analysis of SAP’s Web application server (Netweaver)

Motivation

- Distributed component systems increasingly ubiquitous.
- Quality of service requirements of crucial importance!
- System architects and deployers faced with questions such as:
  - Which platform would provide the best cost/performance ratio for a given application?
  - How do we ensure that the selected platform does not have any inherent scalability bottlenecks?
  - For a given application design, what performance would the application exhibit under the expected workload?
  - How do we ensure that the application does not have any inherent scalability bottlenecks?
Distributed Component System (DCS)

If \( n = 1000 \)
\[ k=? \quad m=? \quad p=? \]
so that all SLAs are fulfilled.

Space of Performance Models
Queueing Networks vs. Petri Nets

- **Queueing Networks**
  - Very powerful for modelling **hardware contention** and scheduling strategies. Many efficient analysis techniques available.
  - Hard to model blocking, synchronization, simultaneous resource possession and **software contention** aspects.

- **Stochastic Petri Nets**
  - Suitable both for qualitative and quantitative analysis.
  - Easy to model blocking, synchronization, simultaneous resource possession and software contention aspects.
  - However, no direct means for modelling queues.

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:** 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

---

Modeling using Queueing Networks

“Perf. modeling and evaluation of large-scale J2EE applications”, CMG-2003

- Benchmark deployment modeled using Queueing Networks (QNs).
- Two problems encountered:
  - Poor model expressiveness: no way to accurately model asynchronous processing and software contention.
  - Large non-product form QNs not tractable.
**Modeling using Queueing Petri Nets**


**Modeling using Queueing Petri Nets (2)**

- **Excellent model expressiveness** for both hardware and software aspects of system behavior.

<table>
<thead>
<tr>
<th>Description</th>
<th>Model</th>
<th>Measured</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBS-CPU workload</td>
<td>100%</td>
<td>105%</td>
<td>15%</td>
</tr>
<tr>
<td>DBS-CP0 workload</td>
<td>100%</td>
<td>105%</td>
<td>15%</td>
</tr>
<tr>
<td>WebServer ServiceLoad</td>
<td>18.42</td>
<td>19.42</td>
<td>5.3%</td>
</tr>
<tr>
<td>WebServer Response Time</td>
<td>0.0058s</td>
<td>0.0058s</td>
<td>0.0%</td>
</tr>
<tr>
<td>Thread Queue Length</td>
<td>17.14</td>
<td>1.8</td>
<td>87%</td>
</tr>
</tbody>
</table>

- **However, state space explosion problem:**
  - Model had to be restricted to part of the application.
  - Max 20 concurrent customers.
  - Models of realistic systems not tractable!
SimQPN – Simulator for QPNs

- Tool and methodology for analyzing QPNs using simulation.
- Provides a scalable simulation engine optimized for QPNs.
- Can be used to analyze models of realistic size and complexity.
- Light-weight and fast.
- Portable across platforms.
- Validated in a number of realistic scenarios.


Performance Modeling Methodology

1. Establish performance modeling objectives.
2. Characterize the system in its current state.
3. Characterize the workload.
4. Develop a performance model.
5. Validate, refine and/or calibrate the model.
6. Use model to predict system performance.
7. Analyze results and address modeling objectives.

Roadmap

- Research Interests
- Introduction to Queueing Petri Nets
- Performance Modeling Methodology
- **Case Study: Modeling SPECjAppServer2004**
- QPN Modeling Environment (QPME) Demo
- Concluding Remarks

SPECjAppServer2004 Business Model

- Dealers
- Suppliers
- Customer Domain
- Corporate Domain
- Supplier Domain
- Manufacturing Domain
SPECjAppServer2004 Business Domains

**CUSTOMER DOMAIN**
Order Entry Application
- Place Order
- Get Order Status
- Get Customer Status
- Cancel Order

**MANUFACTURING DOMAIN**
Parts ➔ Planned Lines ➔ Vehicles
- Create Large Order
- Schedule Work Order
- Update Work Order
- Complete Work Order

**SUPPLIER DOMAIN**
Purchase Parts ➔ Deliver Parts
- Select Supplier
- Send Purchase Order
- Deliver Purchase Order

**CORPORATE DOMAIN**
Customer, Supplier and Parts Information
- Register Customer
- Determine Discount
- Check Credit

**OSG Java Subcommittee**

OPERA Group © S. Kounev
**SPECjAppServer2004 Application Design**

**Benchmark Components:**

1. **EJBs** – J2EE application deployed on the System Under Test (SUT)
2. **Supplier Emulator** – web application emulating external suppliers
3. **Driver** – Java application emulating clients interacting with the system and driving production lines

- RDBMS used for persistence
- Asynchronous-messaging used for inter-domain communication
- Throughput is function of chosen Transaction Injection Rate
- Performance metric is \( JOPS = \text{JAppServerOpsPerSecond} \)

**SPECjAppServer2004 Application Design (2)**

**SPECjAppServer Driver made up of two components:**

1. **DealerEntry Driver:**
   - Emulates automobile dealers interacting with the system.
   - Exercises the dealer and order-entry applications using 3 business transaction types: Browse, Purchase and Manage.
   - Each transaction emulates a client session.
   - Communicates with the SUT through HTTP.

2. **Manufacturing Driver:**
   - Drives production lines in the manufacturing domain.
   - Exercises the manufacturing application.
   - Unit of work is WorkOrder.
   - Communicates with the SUT through RMI.
**SPECjAppServer2004 Application Design (2)**

- **Driver**
  - Client JVM
  - HTTP / RMI

- **SUT**
  - EJB X
  - EJB Y
  - EJB Z
  - ReceiverSes
  - BuyerSes

- **Supplier Emulator**
  - Emulator Servlet
  - Web Container
  - HTTP

- **J2EE AppServer**

---

**Sample Deployment Environment (IBM)**

- **WebSphere Application Server V6.1**
- IBM DB2 8.1 FP6, ESE
- IBM HTTP Server 2.0.47
- IBM Site Selector 5.1

- **SUT**
  - IHS
  - SUT
  - IBM eServer System i50 2 x Intel 3.20GHz 2x4GB RAM
  - IBM eServer xSeries 690 4 x IBM Power 4 x 1.45GHz 2x1GB RAM
  - IBM eServer xSeries 440 4 x IBM Power 4 1.4GHz 2x4GB RAM
  - IBM eServer xSeries 440 4 x IBM Power 4 1.4GHz 2x4GB RAM
  - IBM eServer xSeries 335 4 x IBM Power 4 1.4GHz 2x4GB RAM
  - IBM eServer xSeries 335 4 x IBM Power 4 1.4GHz 2x4GB RAM
  - IBM eServer xSeries 335 4 x IBM Power 4 1.4GHz 2x4GB RAM
  - IBM eServer xSeries 335 4 x IBM Power 4 1.4GHz 2x4GB RAM

- **DB2 Database**
  - With 2 arrays
  - WebSphere Deployment Mgr
  - IMS Server

---
Sample Deployment Environment (Sun)

System Under Test

Database Server

Sun Fire E20K
20 x 1.5 GHz dual x86
Dual Xeon 3.0 GHz
60 GB hard disk

Application Servers

7 x Sun Fire T2000
1 x 2.2 GHz UltraSPARC T2
32 GB memory
4 x 70 GB 10K SAS Drive

Sun StorEdge 7500
1 x 256 MB RAM
1 x 8 GB SAS Drive
1 x 56 GB RAID 0 Drive

Net Gear GSMB2K
24 Port Switch

Case Study - Deployment Environment

Dealers

HTTP

Internet

1 GBit LAN

Oracle 9i Server

2xAMD MP2000+
2GB RAM

HTTP Load Balancer

1 x AMD XP2000+ CPU, 1GB

HTTP

Suppliers

ALH

JDBC

WebLogic 8.1 Cluster

Each node with 1 x AMD XP2000+ CPU, 1GE

OPERA Group © S. Kounev
1. Establish Modeling Objectives

**Normal Conditions:** 72 concurrent dealer clients (40 Browse, 16 Purchase, 16 Manage) and 50 planned production lines in the mfg domain.

**Peak Conditions:** 152 concurrent dealer clients (100 Browse, 26 Purchase, 26 Manage) and 100 planned production lines in the mfg domain.

**Goals:**

- Predict system performance under normal operating conditions with 4 and 6 application servers.
- Predict how much system performance would improve if the load balancer is upgraded with a slightly faster CPU.
- Study the scalability of the system as the workload increases and additional application server nodes are added.
- Determine which servers would be most utilized under heavy load and investigate if they are potential bottlenecks.

2. Characterize the System

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Balancer</td>
<td>WebLogic 5.1 Server (IBM iSeries/AS/400)</td>
</tr>
<tr>
<td></td>
<td>1 x AMD Athlon XP2000+ CPU</td>
</tr>
<tr>
<td></td>
<td>1 GB RAM, SunOS Solaris 9</td>
</tr>
<tr>
<td>App. Server Cluster Nodes</td>
<td>WebLogic 5.1 Server</td>
</tr>
<tr>
<td></td>
<td>1 x AMD Athlon XP2000+ CPU</td>
</tr>
<tr>
<td></td>
<td>1 GB RAM, SunOS Solaris 9</td>
</tr>
<tr>
<td>Database Server</td>
<td>Oracle 9i Server</td>
</tr>
<tr>
<td></td>
<td>2 x AMD Athlon XP2000+ CPU</td>
</tr>
<tr>
<td></td>
<td>2 GB RAM, SunOS Solaris 9</td>
</tr>
<tr>
<td>Local Area Network</td>
<td>1 GB Switched Ethernet</td>
</tr>
</tbody>
</table>
3. Characterize the Workload

1. Basic Components: Dealer Transactions and Work Orders.
2. Workload Classes: Browse, Purchase, Manage, WorkOrder and LgrOrder.
3. Inter-Component Interactions:

![Inter-component interactions diagram]

3. Characterize the Workload (2)

Describe the processing steps (subtransactions).
3. Characterize the Workload (3)

Workload Service Demand Parameters (ms)

<table>
<thead>
<tr>
<th>Category</th>
<th>Normal Conditions</th>
<th>Peak Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WorkOrder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LargeOrder</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Characterize the Workload (4)

Workload Intensity Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Normal Conditions</th>
<th>Peak Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Clients</td>
<td>64</td>
<td>102</td>
</tr>
<tr>
<td>Processor Clients</td>
<td>16</td>
<td>28</td>
</tr>
<tr>
<td>Memory Clients</td>
<td>125</td>
<td>25</td>
</tr>
<tr>
<td>Planned Lines</td>
<td>80</td>
<td>160</td>
</tr>
<tr>
<td>Doctor Think Time</td>
<td>2 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>Mig Think Time</td>
<td>10 sec</td>
<td>16 sec</td>
</tr>
</tbody>
</table>
4. Develop a Performance Model

Assume 2 AS nodes available.

Two Specific Validation Scenarios:
1: 20 B, 10 P, 10 M, 30 PL
2: 40 B, 20 P, 30 M, 50 PL

Max. Modeling Error:
• For Throughput:  8.1%
• For Utilization:  10.2%
• For Resp. Times:  12.9%
### 6. Predict System Performance

#### ANALYSIS RESULTS FOR BEHAVIOR UNDER NORMAL CONDITIONS WITH 4 AND 6 APP SERVERS

<table>
<thead>
<tr>
<th>METHOD</th>
<th>4 App. Server Nodes</th>
<th>6 App. Server Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>Measured</td>
</tr>
<tr>
<td>( X_{L} )</td>
<td>7.365</td>
<td>7.462</td>
</tr>
<tr>
<td>( X_{R} )</td>
<td>3.119</td>
<td>3.264</td>
</tr>
<tr>
<td>( X_{M} )</td>
<td>3.617</td>
<td>3.668</td>
</tr>
<tr>
<td>( X_{W} )</td>
<td>4.511</td>
<td>4.566</td>
</tr>
<tr>
<td>( X_{R} )</td>
<td>0.313</td>
<td>0.314</td>
</tr>
</tbody>
</table>

| \( H_{L} \) | 929 ms | 982 ms | +5.5% | 929 ms | 976 ms | -5.5% |
| \( H_{M} \) | 131 ms | 146 ms | +11.6% | 131 ms | 146 ms | +11.6% |
| \( H_{R} \) | 21 ms | 26 ms | +24.7% | 21 ms | 26 ms | +24.7% |
| \( H_{W} \) | 10 ms | 11 ms | +9.1% | 10 ms | 11 ms | +9.1% |

| \( U_{L} \) | 38.3% | 38.0% | +1.3% | 38.2% | 38.0% | +1.3% |
| \( U_{M} \) | 39.8% | 39.5% | +1.8% | 39.5% | 39.5% | +1.8% |
| \( U_{R} \) | 18.8% | 18.5% | +1.6% | 18.7% | 18.5% | +1.6% |

### 6. Predict System Performance (2)

#### ANALYSIS RESULTS FOR BEHAVIOR UNDER PAIN CONDITIONS WITH 6 APP SERVERS

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Original Load Balance</th>
<th>Upgraded Load Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>Measured</td>
</tr>
<tr>
<td>( X_{L} )</td>
<td>10.850</td>
<td>10.842</td>
</tr>
<tr>
<td>( X_{R} )</td>
<td>3.891</td>
<td>4.015</td>
</tr>
<tr>
<td>( X_{M} )</td>
<td>4.984</td>
<td>4.985</td>
</tr>
<tr>
<td>( X_{W} )</td>
<td>9.364</td>
<td>9.250</td>
</tr>
<tr>
<td>( X_{R} )</td>
<td>0.497</td>
<td>0.490</td>
</tr>
</tbody>
</table>

| \( H_{L} \) | 165 ms | 165 ms | -0.0% | 165 ms | 165 ms | -0.0% |
| \( H_{M} \) | 21 ms | 21 ms | -0.0% | 21 ms | 21 ms | -0.0% |
| \( H_{R} \) | 111 ms | 111 ms | -0.0% | 111 ms | 111 ms | -0.0% |

| \( U_{L} \) | 39.3% | 38.0% | -1.2% | 39.3% | 38.0% | -1.2% |
| \( U_{M} \) | 33.8% | 33.8% | 0.0% | 33.8% | 33.8% | 0.0% |
| \( U_{R} \) | 28.6% | 28.6% | 0.0% | 28.6% | 28.6% | 0.0% |
### 6. Predict System Performance (3)

#### Analysis Results for Simulation Under Heavy Load with AFF Routed Model

<table>
<thead>
<tr>
<th></th>
<th>Heavy Load Scenario 1</th>
<th>Heavy Load Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
<td>Model</td>
<td>Measured</td>
</tr>
<tr>
<td>( N_A )</td>
<td>25.305</td>
<td>25.305</td>
</tr>
<tr>
<td>( X_F )</td>
<td>4.648</td>
<td>4.648</td>
</tr>
<tr>
<td>( X_M )</td>
<td>4.648</td>
<td>4.648</td>
</tr>
<tr>
<td>( X_Y )</td>
<td>8.948</td>
<td>8.948</td>
</tr>
<tr>
<td>( X_Y )</td>
<td>0.037</td>
<td>0.081</td>
</tr>
<tr>
<td>( R_m )</td>
<td>604ms</td>
<td>714ms</td>
</tr>
<tr>
<td>( R_f )</td>
<td>284ms</td>
<td>297ms</td>
</tr>
<tr>
<td>( R_M )</td>
<td>326ms</td>
<td>272ms</td>
</tr>
<tr>
<td>( R_Y )</td>
<td>111ms</td>
<td>172ms</td>
</tr>
<tr>
<td>( T_{L1 N} )</td>
<td>94.1%</td>
<td>91.0%</td>
</tr>
<tr>
<td>( T_{L2 N} )</td>
<td>54.5%</td>
<td>54.1%</td>
</tr>
<tr>
<td>( C_{L1 N} )</td>
<td>3.3%</td>
<td>4.3%</td>
</tr>
</tbody>
</table>

150 Browse Clients  
200 Browse Clients

---

### 6. Predict System Performance (4)

![Diagram](image)

---

OPERA Group © S. Kounev
6. Predict System Performance (4)

<table>
<thead>
<tr>
<th>METRIC</th>
<th>Heavy Load Sc. 3 with 15 Threads</th>
<th>Heavy Load Sc. 3 with 30 Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>Measured</td>
</tr>
<tr>
<td>$X_P$</td>
<td>28.607</td>
<td>27.323</td>
</tr>
<tr>
<td>$X_T$</td>
<td>4.501</td>
<td>4.220</td>
</tr>
<tr>
<td>$X_M$</td>
<td>4.489</td>
<td>4.387</td>
</tr>
<tr>
<td>$X_{NP}$</td>
<td>10.784</td>
<td>10.660</td>
</tr>
<tr>
<td>$X_T$</td>
<td>0.447</td>
<td>0.410</td>
</tr>
<tr>
<td>$R_B$</td>
<td>5495ms</td>
<td>5740ms</td>
</tr>
<tr>
<td>$R_T$</td>
<td>1674ms</td>
<td>1977ms</td>
</tr>
<tr>
<td>$R_M$</td>
<td>1685ms</td>
<td>1779ms</td>
</tr>
<tr>
<td>$R_{av}$</td>
<td>1125ms</td>
<td>1158ms</td>
</tr>
<tr>
<td>$U_{CB}$</td>
<td>100.0%</td>
<td>93.0%</td>
</tr>
<tr>
<td>$U_{AS}$</td>
<td>57.9%</td>
<td>57.8%</td>
</tr>
<tr>
<td>$U_{DB}$</td>
<td>41.6%</td>
<td>44.0%</td>
</tr>
<tr>
<td>$N_{LBQ}$</td>
<td>146</td>
<td>161</td>
</tr>
</tbody>
</table>

Sc.3: 300 B, 30 P, 30 M, 120 PL → Max Error 16.8%
Sc.4: 270 B, 90 P, 60 M, 120 PL → Max Error 15.2%

7. Analyze Results & Address Objectives

[Diagram showing performance metrics for different scenarios such as 8AS / HEAVY 4, 8AS / HEAVY 3, 6AS / PEAK / UPG. LB, 6AS / NORMAL, 4AS / NORMAL, etc., with different performance indicators marked by bars representing LB-C, AS-C, and DB-C.]
**Benefits of using QPNs**

1. QPN models allow the integration of hardware and software aspects of system behavior.
2. Using QPNs, DCS can be modeled accurately.
3. The knowledge of the structure and behavior of QPNs can be exploited for efficient simulation using SimQPN.
4. QPNs can be used to combine qualitative and quantitative system analysis.
5. QPN models have an intuitive graphical representation facilitating model development.

---

**QPME**

- A performance modeling tool based on QPNs
- QPN Editor (QPE) and Simulator (SimQPN)
- Based on Eclipse/GEF
- Provides a user-friendly graphical user interface
- Runs on all platforms supported by Eclipse
Summary & Conclusions

- Presented a systematic approach for performance prediction.
- Studied a representative application and predicted its performance under realistic load conditions.
- Model predictions were validated against measurements on the real system. The modeling error did not exceed 21.2%!

- QPN models can be exploited for accurate performance prediction in realistic scenarios.
- Proposed methodology provides a powerful tool for sizing and capacity planning.
Further Reading


Papers available for download at http://www.cl.cam.ac.uk/~sk507

Questions

Thank You for your Attention!

QUESTIONS?