Performance of Web Services

Prof. Mario Jeckle
University of Applied Sciences Furtwangen

Dr. Ingo Melzer, RIC/ED
DaimlerChrysler Forschungszentrum Ulm
**Contents**

- Introduction
- Methodology
- Results and Interpretation
  - Java Remote Method Invocation (RMI)
  - Common Object Request Broker Architecture (CORBA)
  - SOAP
- Summary
Introduction

Looking at:

• RMI
• CORBA
• SOAP
• HTTP

• Response Time
• Network Traffic
• CPU Load

Influenced by
Test Configuration

- Standards: SOAP v1.1, HTTP/1.1
- Ethernet: 10Mbit/s, Connected by hub
- Client configuration
  - AMD Athlon 1800Mhz
  - SUN J2SE v1.4.1_02-b06
  - Apache Axis v1.1 RC2
  - Windows XP Home (v5.1 Build2600.xpshp1.020828-1920, SP1)
- Server configuration
  - Dual AMD Athlon 1200Mhz
  - SUN J2SE v1.4.1_02-b06
  - Apache Axis v1.1 RC2
  - Jakarta Tomcat v4.1.26
  - Apache v1.3.27
  - Perl v5.8.0
  - Suse Linux v8.1 (Kernel 2.4.19)
Measurements

• More than 1,200,000 single data points (RMI, CORBA and SOAP)

• What we measured
  • Response time
  • CPU load

• Points of interest
  • Scalability
  • Faults

• No Optimization
  • No process modifications
  • No low level timing optimizations

• No other network traffic
Methodology - Goals

• No side effects
  • I.e. no additional system or communication requirements
• Scalable methodology
• Technology/Toolkit agnostic
• Different communication styles
  • RPC- and document style
• Multiple protocols tested
  • HTTP, TCP, ICMP
Methodology - Our Ping Service

- Modeled after ICMP Echo-Service (Ping, RFC 792)
  - Symmetric Network load
  - No logic on server
  - Virtually no requirements on execution environments
- Only three lines of code in Perl
- Portable Java-Client
Protocol Hierarchy

- SOAP
- HTTP, ...
- RMI
- CORBA
- TCP, ...
- ICMP
- IP, ...
- Ethernet, ...
- Wire

Direct comparison of results is difficult because they operate on different conceptual levels.
Results I: Package Size

Payload size: 0 bytes

- SOAP: 1298 bytes
- CORBA: 620 + 869 bytes
- RMI: 308 + 1056 bytes
- ICMP: 106 bytes

[Bar chart showing package sizes for SOAP, CORBA, RMI, and ICMP]
Results I: Package size

Payload size: 10,000 bytes
Results I: Package Size

- **SOAP**
  - 0 bytes payload
    - As measured: 1298 bytes
  - 20,000 bytes payload (10,000 request + 10,000 response)
    - Expected value: 21298 bytes
    - As measured: 21300 bytes

- **CORBA**
  - 0 bytes payload
    - As measured: 869 bytes
  - 20,000 bytes payload (10,000 request + 10,000 response)
    - Expected value: 20869 bytes
    - As measured: 20818 bytes

- **RMI**
  - 0 bytes payload
    - As measured: 1056 bytes
  - 20,000 bytes payload (10,000 request + 10,000 response)
    - Expected value: 21056 bytes
    - As measured: 20851 bytes

**Conclusion:** package size scales linear with payload size
Results II: Response Time

HTTP Overhead causes higher response times for SOAP packages
Results II: Response Time

Response time grows exponentially
Results II: Response Time

Observation: reproducible, extreme fluctuations
Explanation: TCP Retransmission-Time (RTO) exceeded

measurment inaccuracy: 18%
Results II: Response Time

Perl over TCP

Java over HTTP
**Results II: Response Time**

- AXIS SOAP, Web Service not precompiled (JWS deployment).
- First call requires compiling;
  - Response time is 16 times higher
- Compiled code is cached, but only for a short time
Results II: Response Time

- Precompiled code is much faster on the first call
Results II: Response Time, Different Transport Protocols

- ICMP-Ping
- RMI
- CORBA
- SOAP over HTTP
- SOAP over TCP

Payload Size [bytes]

Response Time [ms]
Results III: Load

- Server
  - SPEC: 340
  - 42 calls/sec
  - 34 calls/sec

- Client
  - SPEC: 250

- Server
  - SPEC: 274
  - 47 calls/sec
  - 38 calls/sec

- Client
  - SPEC: 350
Results III: Load

• What we measured
  • 10000 calls with 3000 byte payload each
  • Client/server performance according to published SPEC2000 data

• Results
  • 40% faster client: 10% more calls
  • 20% faster server: 20% more calls

• Conclusion
  • SPEC2000 value of server makes good estimate for web service performance
## Results III: Load

<table>
<thead>
<tr>
<th>Server</th>
<th>Spec 2000</th>
<th>Expected Performance [calls/sec, 3000 bytes payload]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium III 750 Mhz</td>
<td>340</td>
<td>47</td>
</tr>
<tr>
<td>Pentium IV 1,3 Ghz</td>
<td>561</td>
<td>77</td>
</tr>
<tr>
<td>Pentium IV 2,66 Ghz</td>
<td>957</td>
<td>132</td>
</tr>
<tr>
<td>Dual Intel Xeon 2,8 Ghz</td>
<td>926</td>
<td>128</td>
</tr>
<tr>
<td>Dual AMD Opteron 1,8 Ghz</td>
<td>1023</td>
<td>141</td>
</tr>
<tr>
<td>Intel Xeon 3,06 Ghz</td>
<td>1113</td>
<td>155</td>
</tr>
</tbody>
</table>
Conclusion

- Network load (influenced by package size) can have a large impact on overall performance.
- SOAP often performs much better than expected.
- HTTP is the bottleneck.
- SPEC values deliver good estimates for server performance.
- There is lots of potential for performance optimization.
- SOAP Ping is a good benchmark for comparing SOAP implementations.