PushToTest

SOA Scalability and Developer Productivity Knowledge Kit

This presentation summarizes PushToTest's experience building The Kit and running performance and scalability tests against the implementations.

Download the Kit at: http://www.pushtotest.com/Downloads/kits/soakit.html

PushToTest

SOA Scalability and Developer Productivity Knowledge Kit

Background and Goals

Software architects and developers make choices of XML parsing techniques, service libraries, encoding techniques, and protocols when building Service Oriented Architecture (SOA.) Each choice has an impact on scalability and performance of the finished service. The SOA Scalability and Developer Productivity Knowledge Kit ("The Kit") has three goals:

1) Explain the changing landscape of APIs, libraries, encoding techniques and protocols to software architects and developers. PushToTest tracks these and finds that today's generation of technology choices will change in the near term.
2) Identify and use real-world scenarios that inform software architects and developers of the most appropriate technology choices based on the goals of the intended service.
3) Deliver code that is compatible with your existing knowledge of building functional and scalability tests, including black-box, unit testing, and agile testing methods.

In "The Kit" you will find these resources:

1) A Developer's Journal describing in detail:
   - Detailed use cases and test scenarios
   - Design Decisions and Trade-offs
   - XML and Java Binding Implementation Stories
   - Client-side Software To Call The Implemented Services
   - Server-side Software That Implement The Services
   - Use Case Scenario Specific Findings
   - Installation and Performance Tuning
2) Complete source code to each use case and test scenario; including Ant build scripts so you may build The Kit in your own environment.
3) Pre-built JAR and WAR files to run immediately in your own environment. PushToTest publishes kits for BEA WebLogic Server 8.1 and 9.0, Oracle Application Server 10g, JBoss 4, and IBM WebSphere 6. All of the kits are distributed under free open-source license.
4) TestMaker and XS Test scripts to stage a scalability and performance test of each use case and test scenario.

Download the Kit at: http://www.pushtotest.com/Downloads/kits/soakit.html

Agenda

- Project Goals
- Methodology
- Results
  - XML Binding Compiler, Streaming XML Parser, DOM
- Summary Observations
- Configuration Settings
- Distribution

PushToTest

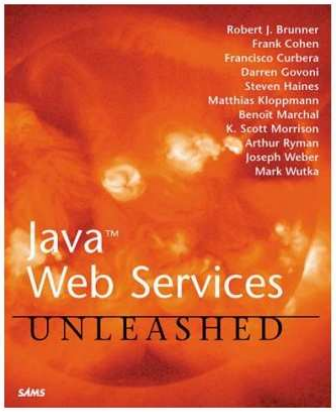
The "Go To" Company

- Bridge Between Enterprises and Tool Vendors in SOA Space
- TestMaker Platform
- Deliver Custom Test Solutions
- Run Scalability Studies
- Training: Dev, QA, IT

JAVA™ TESTING and DESIGN

FROM UNIT TESTING TO AUTOMATED WEB TESTS

FRANK COHEN


Robert J. Brunner
Frank Cohen
Francisco Curbera
Darren Govoni
Steven Haines
Matthias Kloppmann
Benoit Marchal
K. Scott Morrison
Arthur Ryman
Joseph Weber
Mark Wutka

Java™ Web Services UNLEASHED

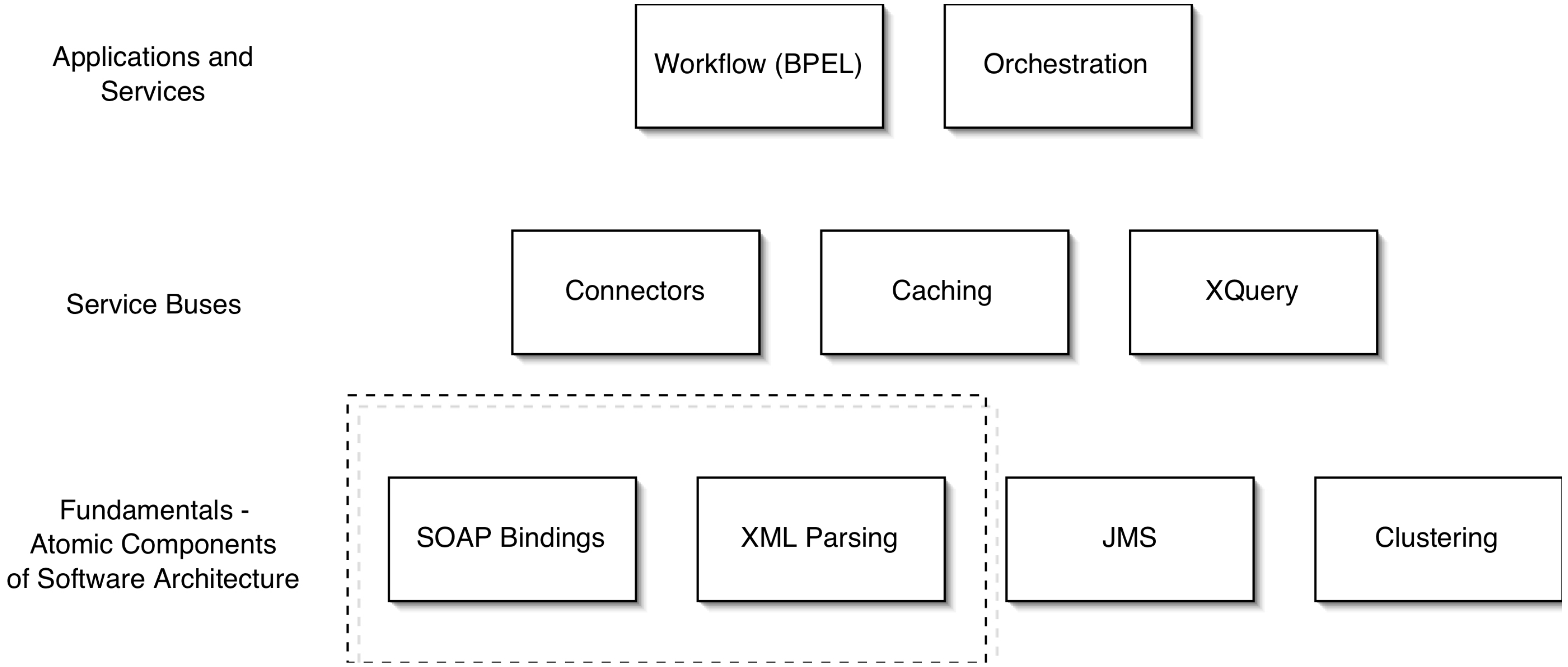
SAMS

bea™ Sun microsystems IBM® Microsoft®

PushToTest

Scope

- Large group of possible tests
- Focused on SOA as core



Applications and Services

Workflow (BPEL)

Orchestration

Service Buses

Connectors

Caching

XQuery

Fundamentals - Atomic Components of Software Architecture

SOAP Bindings

XML Parsing

JMS

Clustering

Methodology

- Atomic Test of SOA Binding/XML Parsing
- 1-Tier Application
- Parameters: App Server, XML Parser, SOAP Bindings, Concurrent Requests, Payload Size
- Use Published Instructions to Construct
- HP Supplied Server Equipment
- Results and Code Immediately Useful to Customers

Server Basic Configuration

Server	JVM	Tunning
WLS 8.1	Jrockit 1.4.2_05	-Xmx1024 / Production Mode
JBOSS 4.0.1	1.5.0_03	-Xmx1024 / Java Tune
OAS 10g	1.4.2_03	-Xmx1024 / Java Tune
WLS 9	Jrockit 1.5.0_03	-Xmx1024 / Production Mode
WS 6	1.4.2_03	-Xmx1024 / No PMI / Java Tune

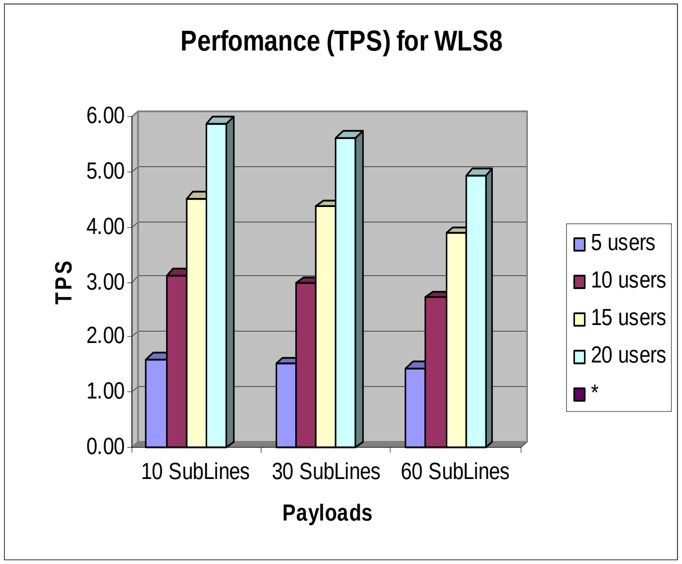
- All servers –Xms and –Xmx memory settings were the same, to reduce memory allocation overhead.
- Jboss had a serious memory leak with JVM 1.4.2, so JVM 1.5 was used for that server. WLS servers used JRockit
- All servers executed using production mode or –server directive, which optimizes runtime but increases server initialization.
- Logging, debugging and monitor components were disabled to reduce overhead.
- All servers were restarted after each scenario, to clean up resource allocations.

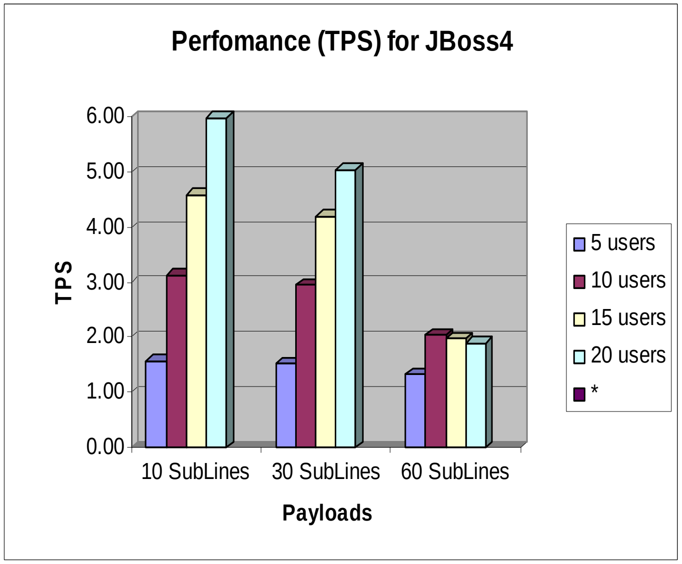
XML Binding Compiler Scenario TV Dinner

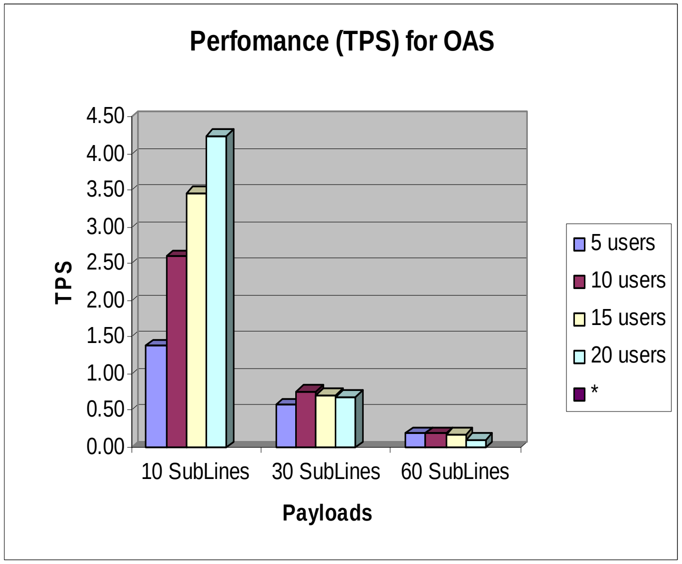
- Parts Ordering Service
- OAGI Business Object Documents
    - Automotive Industry Schema (STAR BODs)
- Request: Process Purchase Order
- Response: Acknowledgement
- Large payload sizes: (100K to 5 Mbytes)
- Named element approach to XML Parsing
- XML Representation: String

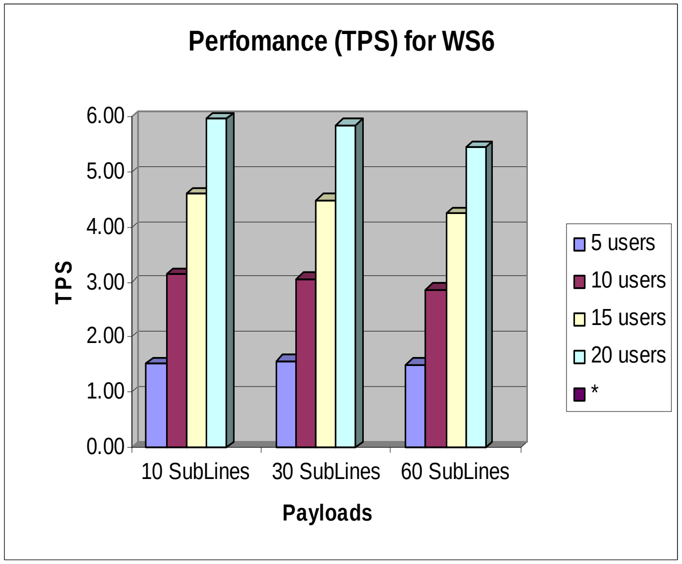
PushToTest

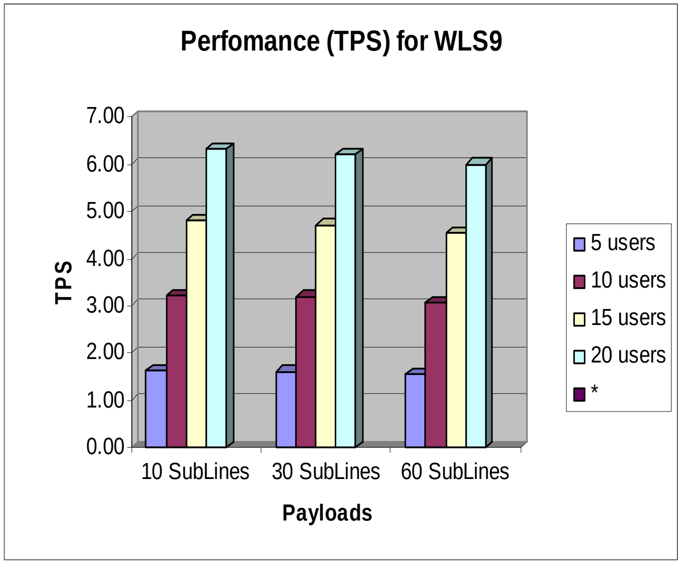
XML Binding Compiler Results



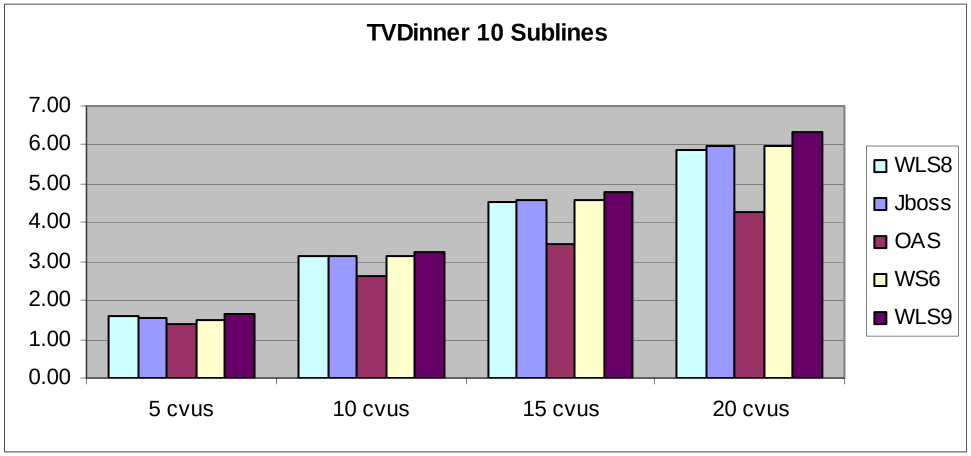


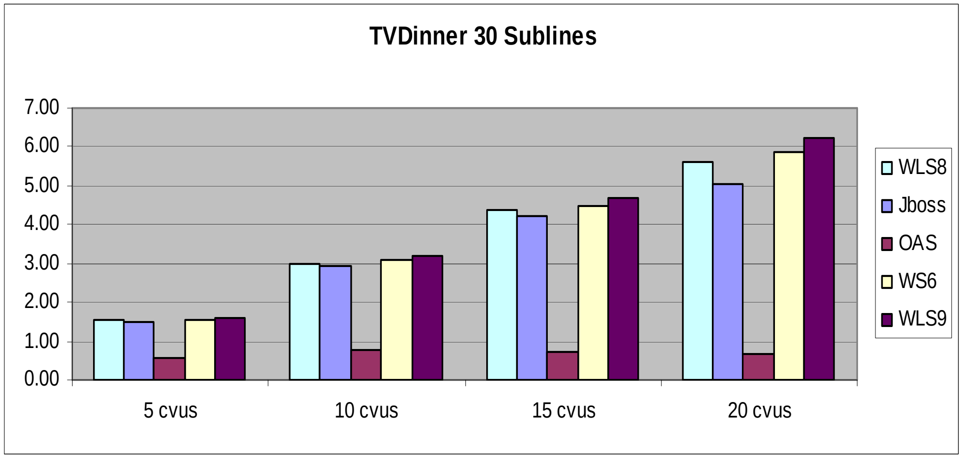


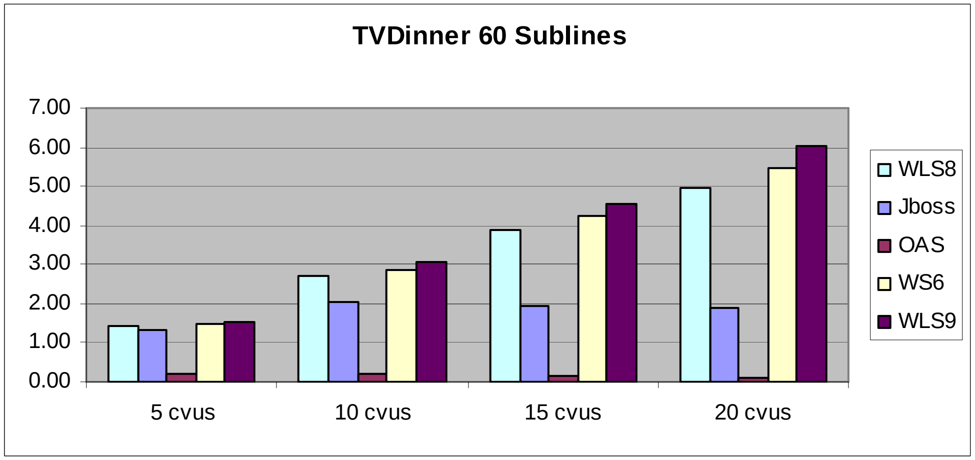




XML Binding Compiler Comparison


TVDinner 10 Sublines

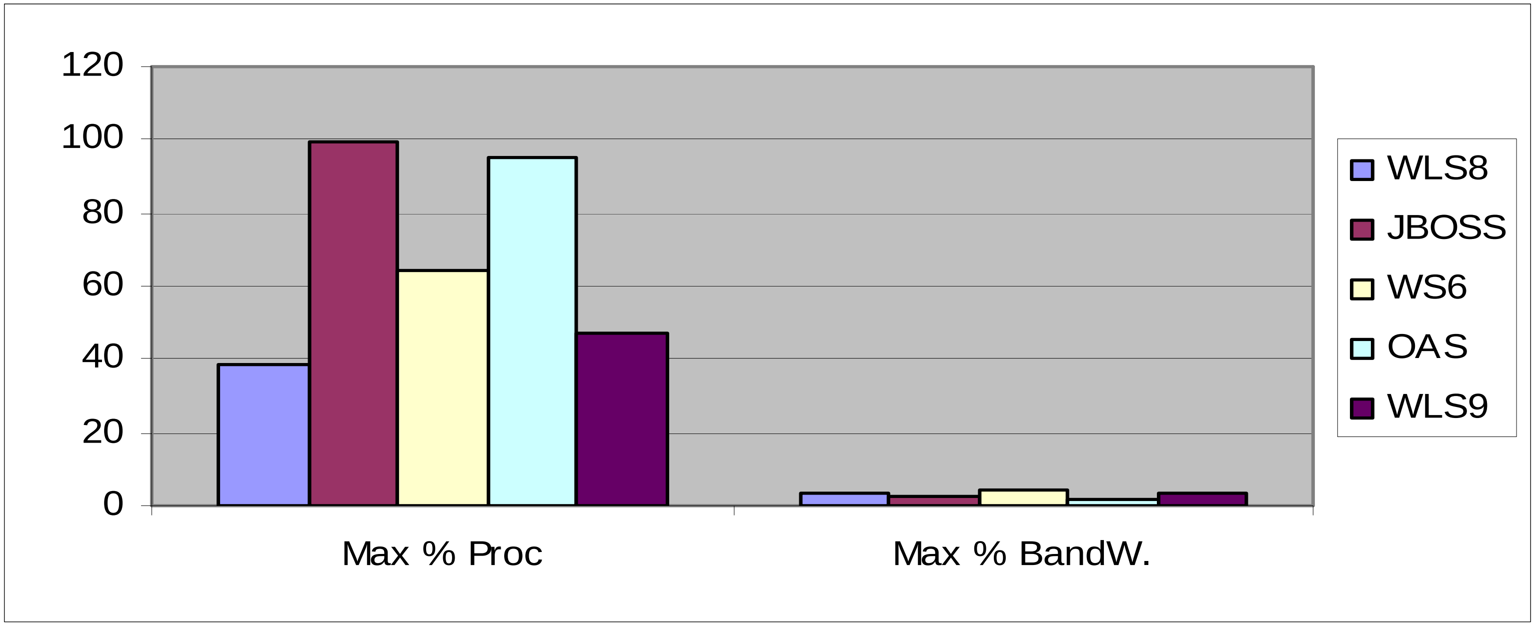

TVDinner 30 Sublines


TVDinner 60 Sublines

5 cvus		WLS 8.1	Jboss 4.0.1	OAS 10g	WS 6	WLS 9
	10 SubLines	1.00	0.97	0.86	0.94	1.01
	30 SubLines	1.00	0.99	0.38	1.03	1.06
	60 SubLines	1.00	0.92	0.13	1.04	1.07
10 cvus						
	10 SubLines	1.00	1.00	0.83	1.00	1.03
	30 SubLines	1.00	0.99	0.25	1.03	1.07
	60 SubLines	1.00	0.74	0.07	1.05	1.12
15 cvus						
	10 SubLines	1.00	1.01	0.76	1.01	1.06
	30 SubLines	1.00	0.96	0.16	1.03	1.08
	60 SubLines	1.00	0.50	0.04	1.09	1.17
20 cvus						
	10 SubLines	1.00	1.02	0.72	1.02	1.07
	30 SubLines	1.00	0.90	0.12	1.04	1.10
	60 SubLines	1.00	0.38	0.02	1.10	1.21

XML Binding Compiler Hardware Performance

	Client		Server	
Server	**Max % Proc**	**Max % BandW.**	**Max % Proc**	**Max % BandW.**
WLS8	63.65	2.22	39	3.29
JBOSS	8.65	1.26	99.2	2.67
WS6	17.8	2.45	64	3.9
OAS	40.7	0.49	95	1.74
WLS9	17	2.6	47	3.6



PushToTest

Conclusions - XML Binding Compiler

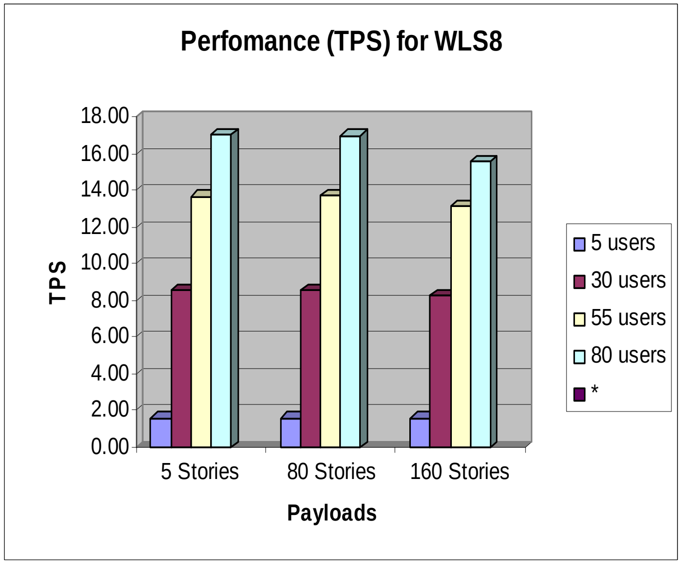
- WLS 9 was ahead of all others on all the tests
- WebLogic 8.1 and WebSphere 6 second best
- JBoss matched and even rebased WLS 8.1 a couple of times with small payloads, but lowered the performance for larger payloads
- Oracle Application Server suffered due to poor String processing
- JBoss and OAS consumed almost all processor time in the higher load tests
- WLS 8.1 was the most efficient in processor use

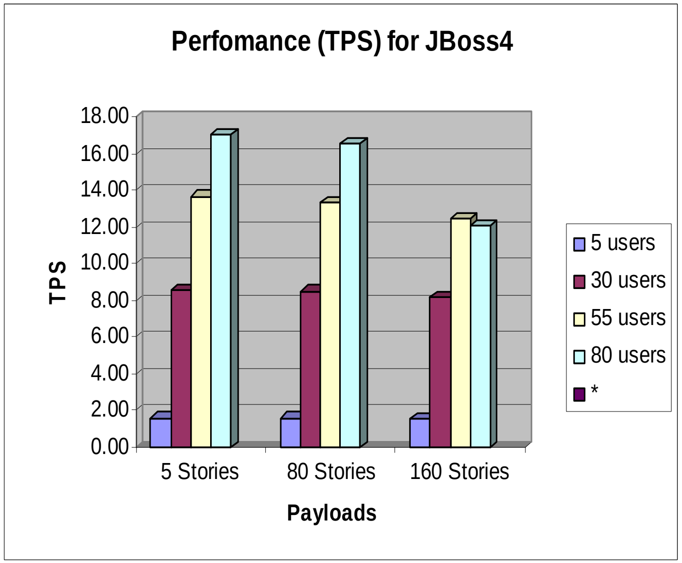
Streaming XML Parser Scenario Sushi Boats

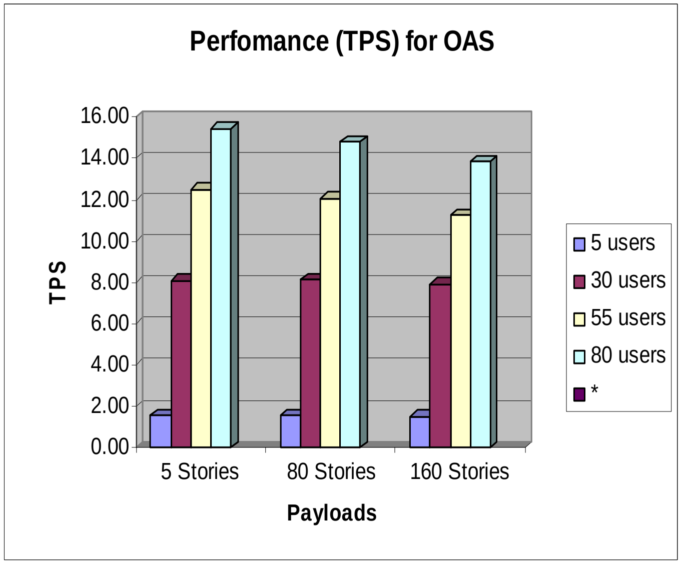
- Portal application receives news postings
- Multiple news stories in each request
  - Find interesting stories, skip others
- StAX approach to XML parsing
- XML Representation: String

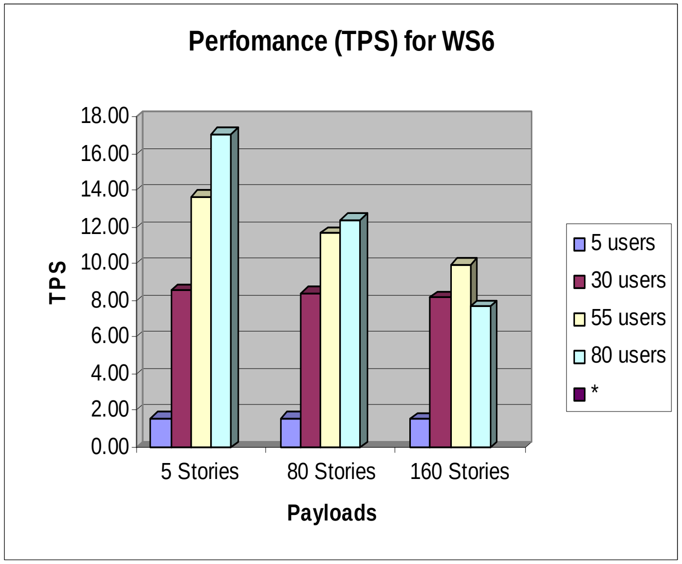
PushToTest

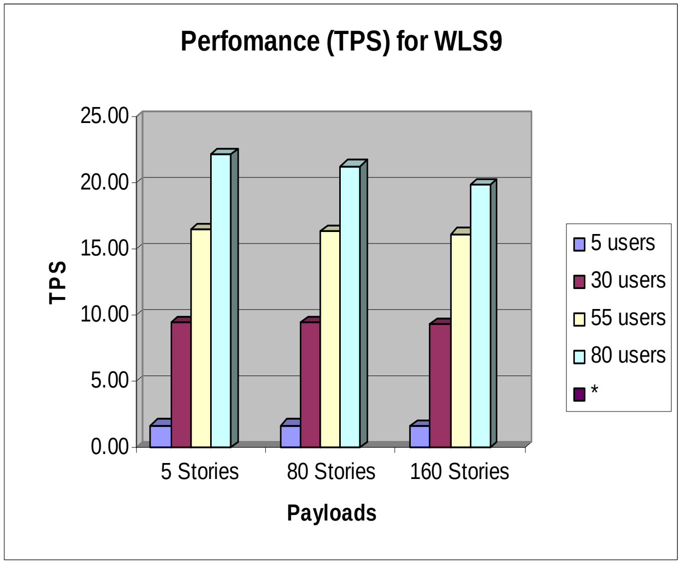
Streaming XML Parsing Results


Perfomance (TPS) for WLS8

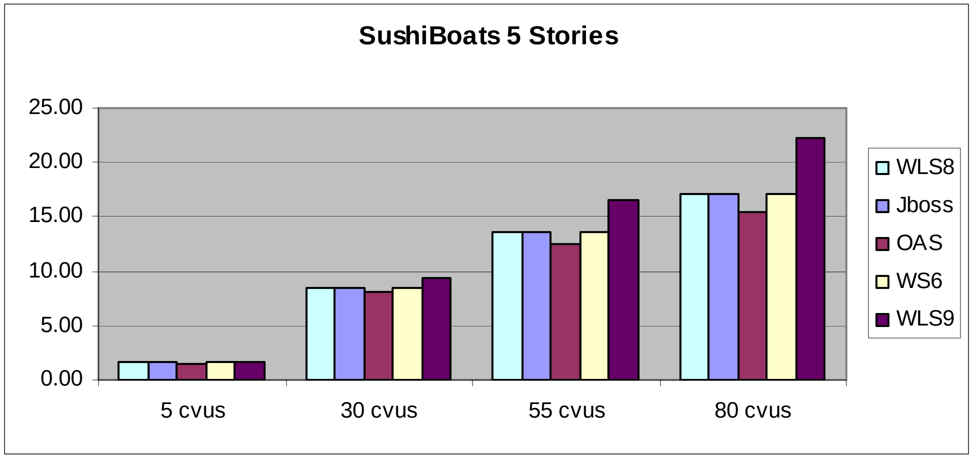

Perfomance (TPS) for JBoss4

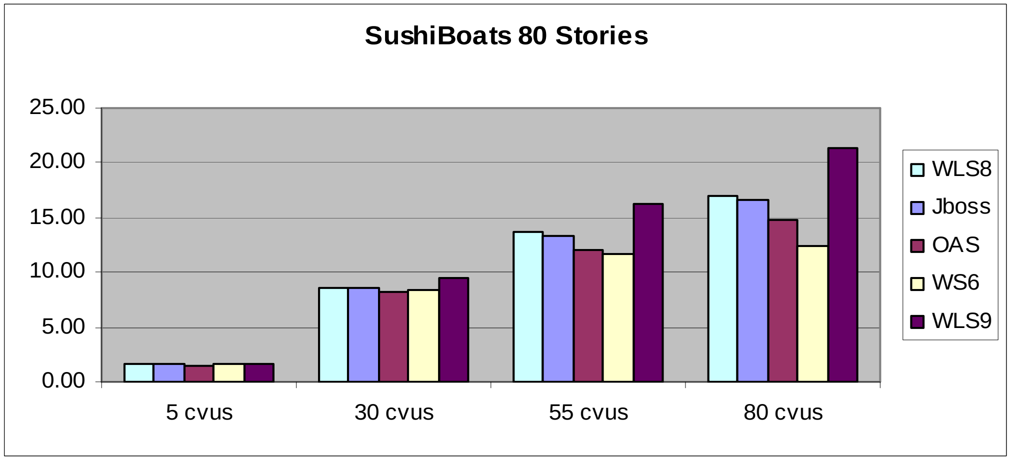

Perfomance (TPS) for OAS

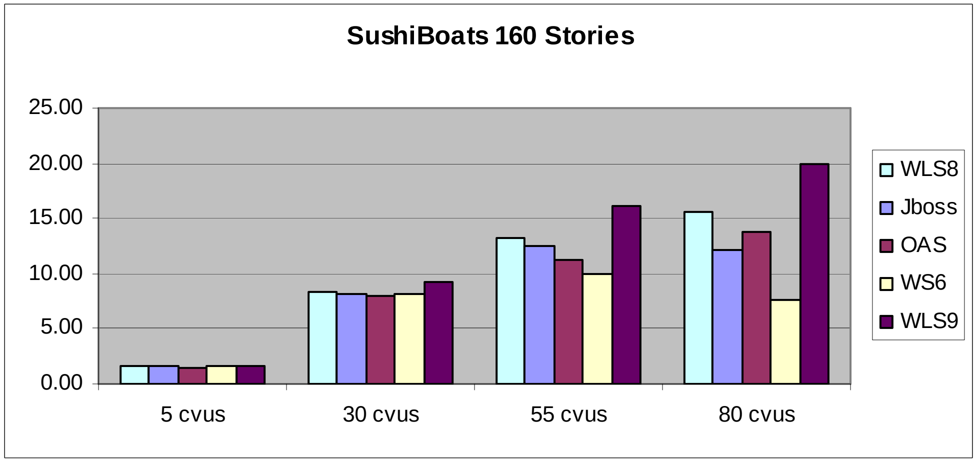

Perfomance (TPS) for WS6


Perfomance (TPS) for WLS9

Streaming XML Parsing Comparison



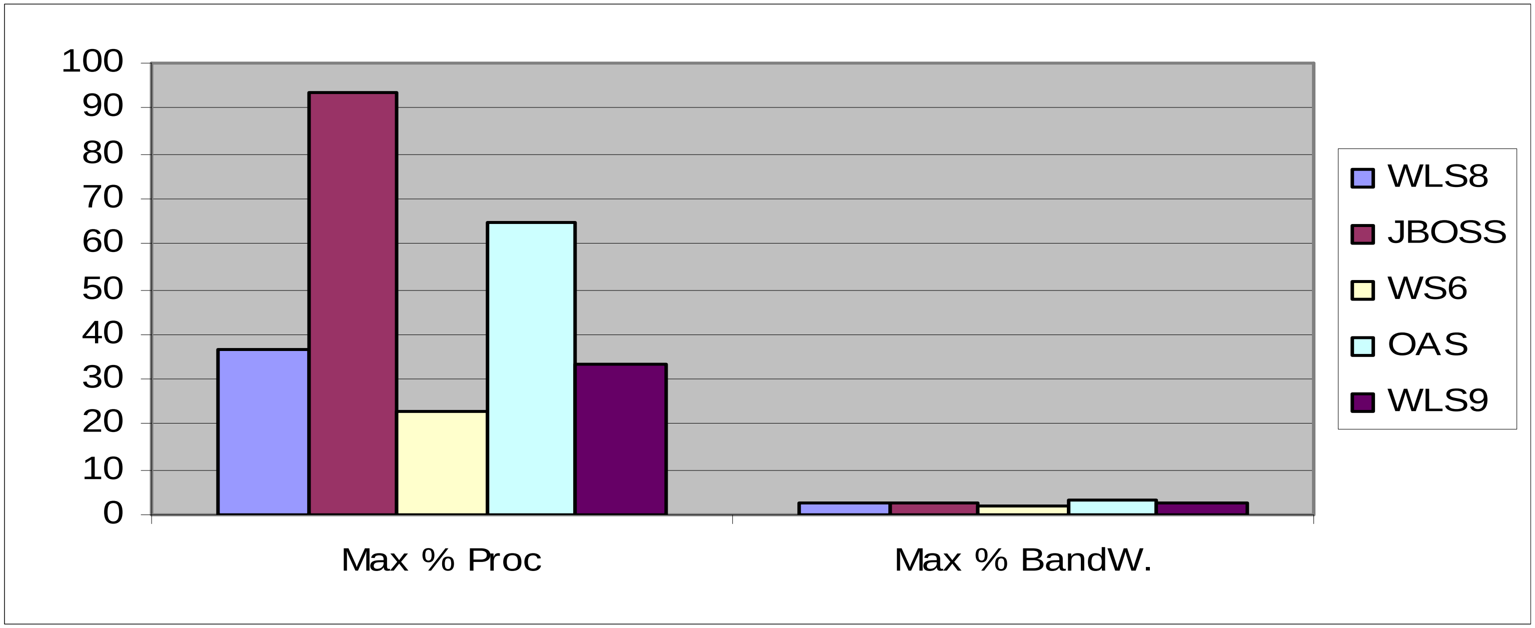




5 cvus		WLS 8.1	Jboss 4.0.1	OAS 10g	WS 6	WLS9
	5 Stories	1.00	1.00	0.95	1.00	1.03
	80 Stories	1.00	1.01	0.95	1.01	1.03
	160 Stories	1.00	0.97	0.93	0.97	1.01
30 cvus						
	5 Stories	1.00	1.00	0.95	1.00	1.10
	80 Stories	1.00	0.99	0.95	0.99	1.10
	160 Stories	1.00	0.99	0.95	0.99	1.12
55 cvus						
	5 Stories	1.00	1.00	0.91	1.00	1.20
	80 Stories	1.00	0.97	0.88	0.85	1.19
	160 Stories	1.00	0.95	0.86	0.76	1.23
80 cvus						
	5 Stories	1.00	1.00	0.91	1.00	1.30
	80 Stories	1.00	0.98	0.87	0.73	1.25
	160 Stories	1.00	0.78	0.89	0.49	1.28

Streaming XML Parsing Hardware Performance

	Client		Server	
Server	**Max % Proc**	**Max % BandW.**	**Max % Proc**	**Max % BandW.**
WLS8	58.73	2.28	36.52	2.65
JBOSS	87.4	1.96	93.65	2.4
WS6	81.77	1.63	23	2
OAS	38.35	1.9	65	3.4
WLS9	51.48	2.3	33.4	2.86



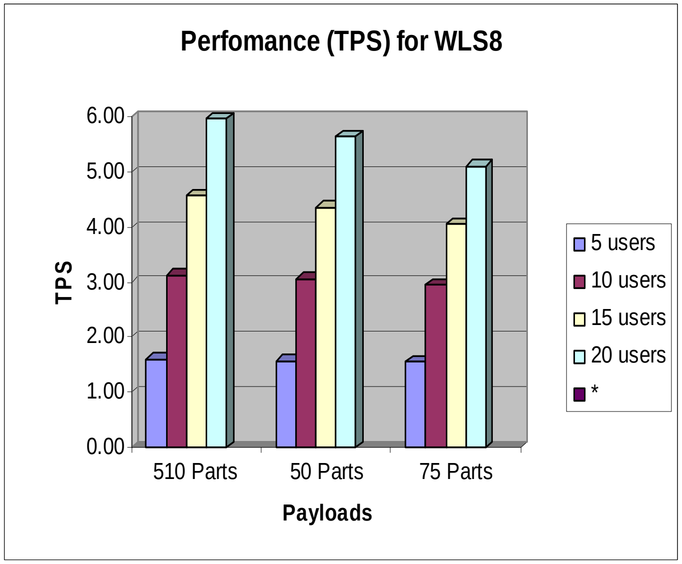
Streaming XML Parsing Conclusions

- For low StAX requirements (5 stories), all servers performed similar
- For higher StAX requirements (160 stories) WLS 9 performed better at high concurrency (80 cvus) on all cases
- WS6 used WLS 8.1 StAX implementation but even so, performance was decreased perceptively at high load
- JBoss used the most processor time, WS6 the most efficient in processor use

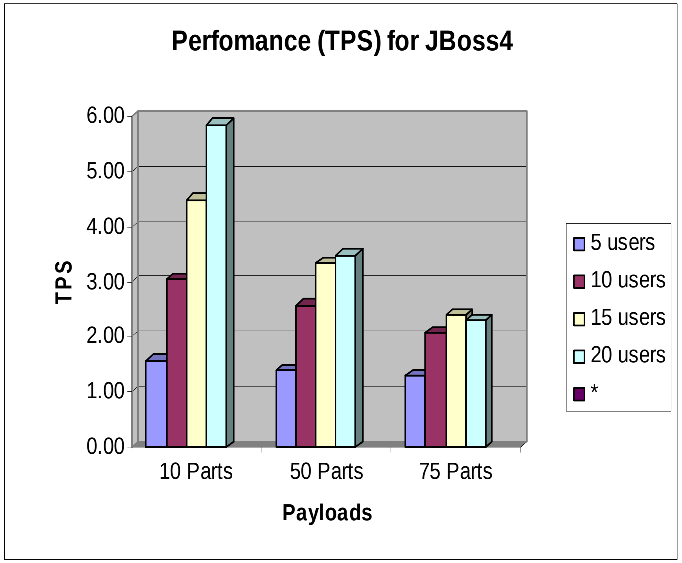
DOM Parsing Scenario

- Order Request Validation Service
- Validate every element in a request
- Xerces/DOM approach to XML parsing
- XML Representation: DOM-SOAP Element

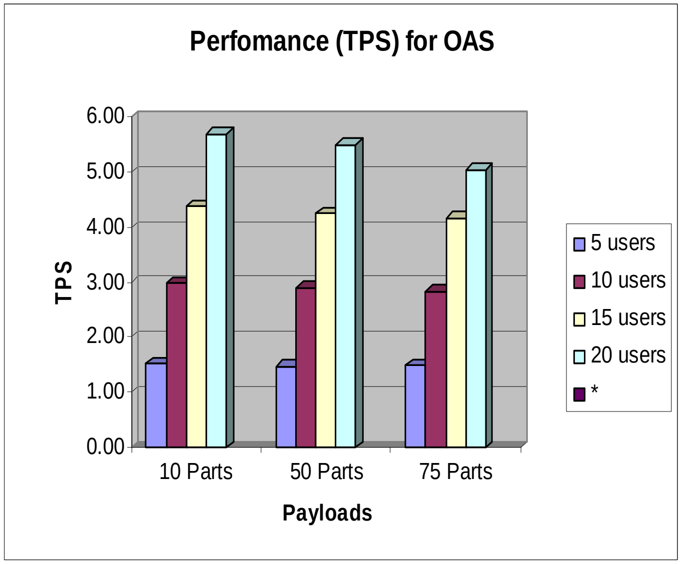
DOM Parsing Results



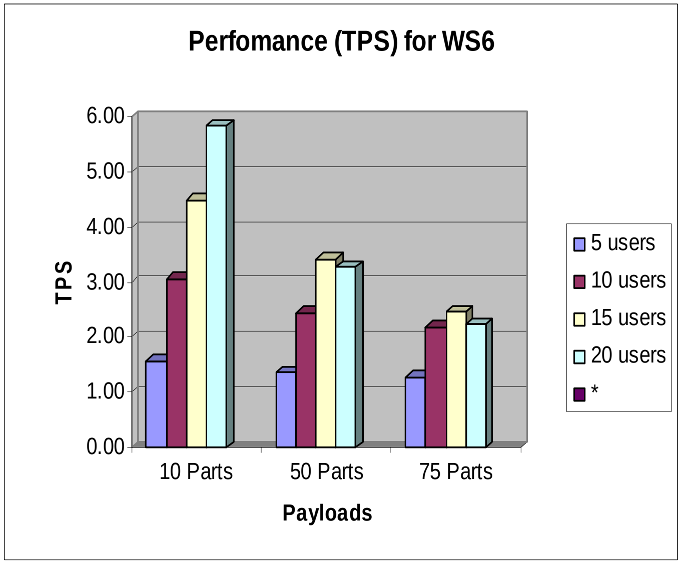
Perfomance (TPS) for WLS8



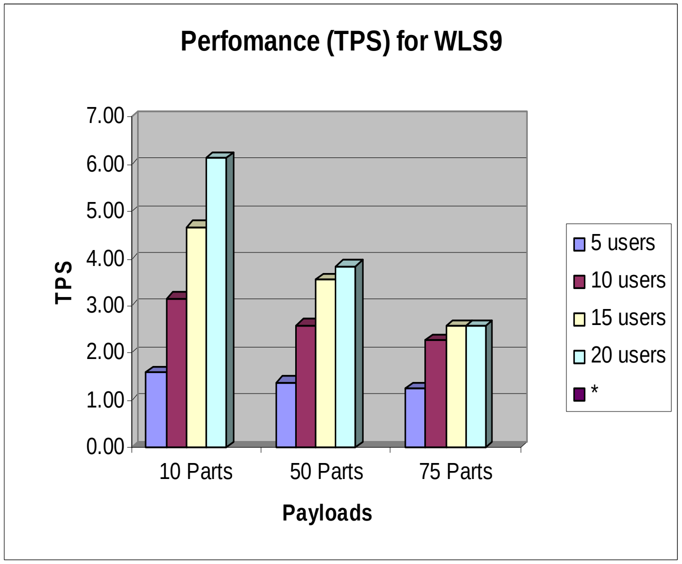
Perfomance (TPS) for JBoss4



Perfomance (TPS) for OAS

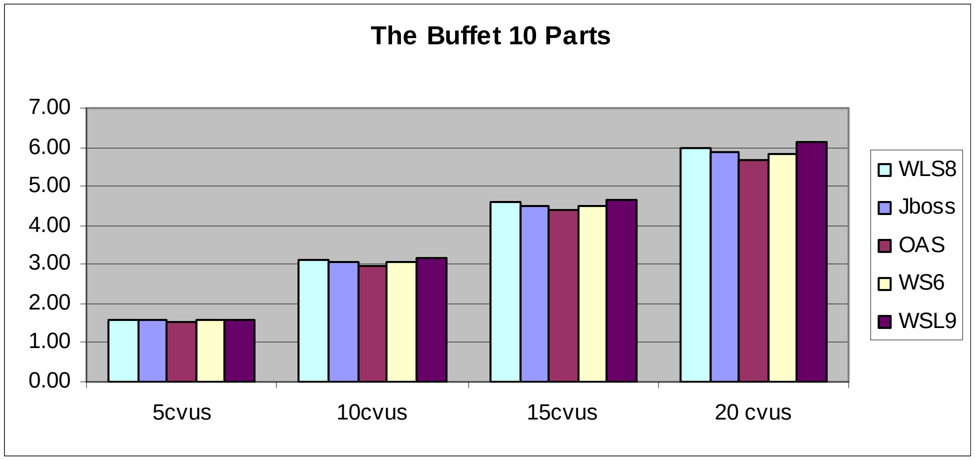


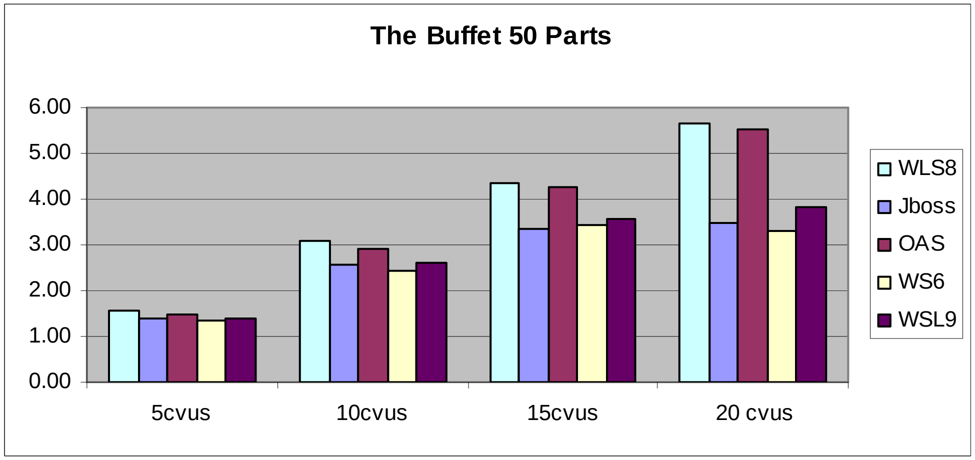
Perfomance (TPS) for WS6

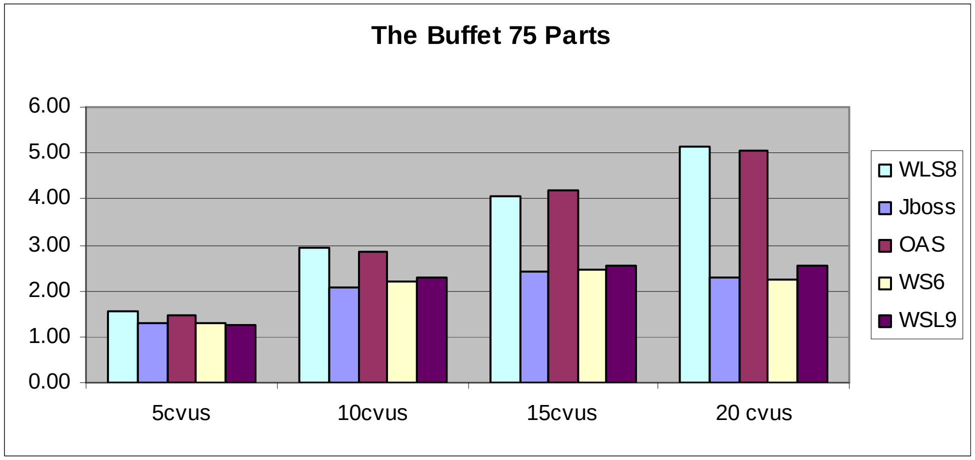


Perfomance (TPS) for WLS9

DOM Parsing Comparison



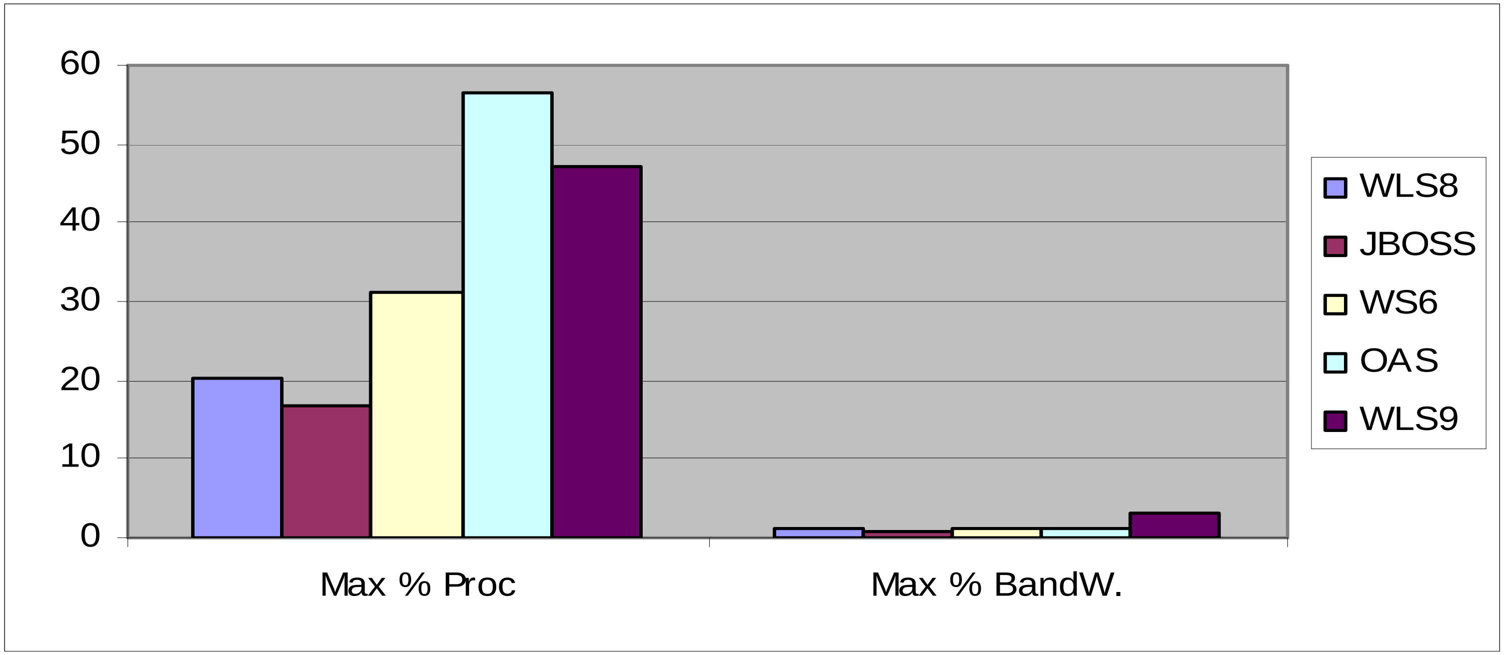




5 cvus		WLS 8.1	Jboss 4.0.1	OAS 10g	WS 6	WLS 9
	10 Parts	1.00	0.98	0.95	0.98	1.00
	50 Parts	1.00	0.89	0.94	0.86	0.88
	75 Parts	1.00	0.83	0.96	0.82	0.80
10 cvus						
	10 Parts	1.00	0.98	0.95	0.98	1.01
	50 Parts	1.00	0.84	0.95	0.80	0.85
	75 Parts	1.00	0.71	0.97	0.75	0.77
15 cvus						
	10 Parts	1.00	0.98	0.95	0.98	1.02
	50 Parts	1.00	0.76	0.98	0.78	0.81
	75 Parts	1.00	0.59	1.03	0.61	0.63
20 cvus						
	10 Parts	1.00	0.98	0.95	0.98	1.03
	50 Parts	1.00	0.62	0.97	0.58	0.68
	75 Parts	1.00	0.45	0.99	0.44	0.50

DOM Hardware Performance

	Client		Server	
Server	**Max % Proc**	**Max % BandW.**	**Max % Proc**	**Max % BandW.**
WLS8	40.85	0.73	20.3	1.04
JBOSS	91.2	0.4	16.78	0.66
WS6	82	0.34	31	1.11
OAS	10	0.67	56.4	1.17
WLS9	89	1.33	47	2.95



DOM Parsing Conclusions

- WLS 8.1 and OAS Show Higher TPS in larger CVUs, Payloads. OAS even surpasses slightly WLS in one case.
- WLS 8.1 and OAS use DOM in the SOAP stack
- JBoss, WS6 and WLS9 use a transform (SOAPElement to DOM) for adherence to JAX-RPC. That impacts in the performance.

Reliability and Efficiency

- JBoss in JVM 1.4.2 did Not Handle in Out-Of-Memory Conditions Gracefully. JVM 1.5 was used instead.

Use Case	Concurrent Requests	Payload Size	Result
TV Dinner	5	5	.68 TPS
	25	50	Server Hangs
	50	25	Server Hangs
The Buffet	20	70	2.36 TPS
	20	200	Server Hangs

PushToTest

Reliability and Efficiency

- StAX implementation is natively offered only in WebLogic 9.
- WLS9 StAX implementation presents a bug in the getAttributeValue() method.
- The native JAXB Implementation of OAS doesn’t support the OAGIS complexity
- Sun’s JAXB Implementation requires additional configuration to accept OAGIS schemas

Reliability and Efficiency

- The JBoss JAX-RPC/AXIS implementation requires the manipulation of XML descriptors to work with document style.
- WS6 generated clients for WS6 generated web Services failed the invocation in document style.
- WLS9 JWS implementations require much less code.
- WLS9 generated WSDL in Wrapped style causes “uniqueness” problems to the Sun’s JAX-RPC wscompile.

PushToTest

Summary Observations

- XML Binding Compiler Scenario
  - WLS 9 Best Performance, Easiest to Code
  - WS6 surpasses slightly WLS8.1 in almost all cases, but WLS9 surpasses all the others.
  - JBoss presented problems with larger requests and used extensive CPU, going from 0.5 secs to around 10 secs in the largest payload.
  - OAS presented serious performance degradation with large String request in the dispatching servlet.

Summary Observations

- Streaming XML Parser Scenario
  - WLS 9 Outperforms, WS6 falls down.
  - No major differences in lower payload/cvus
  - Performance decreases (WS6, Jboss) with higher cvus and payload, both variables impact their performance.
  - OAS has a lower performance than the rest, partly because of its String problem.
  - WLS9 outperforms all the others
    - 30% over the reference (WLS 8.1)
    - 78% over the WS6 at 80 cvus/160 stories

Summary Observations

- DOM Scenario
  - DOM leads over SOAPElement
  - WLS 8.1 and OAS, using the natural DOM as transport, showed a improved difference on higher payloads.
  - JBoss and WS6 decreased their performance due to the conversion from SOAPElement to DOM.
  - WLS9 performance is slightly better than SOAPElement based ones, but falls far from WLS8.1 and OAS.

PushToTest

Summary Observations

- Development
  - The lack of a JBoss tool-chest for developing web services made it the most difficult to work with.
  - JWS made it easier to create the Web Service in WLS9. Others require manipulating descriptors.
  - WS6 tool-chest was a great help, but its generated descriptors needed manipulation to make them work on document style.
  - OAS tool-chest didn’t support document style.

Glossary

- STAR = Software Technology for Automotive Retailing industry consortium
- OAGIS = Open Applications Group, designed Business Object Document (BOD) XML schema for STAR
- Sublines = XML elements inserted into a STAR/OAGIS BOD to vary payload size
- CVUs = Concurrent Virtual Users

Resources

- Brian Bartel, bbartel@pushtotest.com
- Frank Cohen, fcohen@pushtotest.com
- http://www.pushtotest.com
- Download the Kit at:

  http://www.pushtotest.com/Downloads/kits/soakit.html