



Scaling Up the JBoss Application Server.

Peter Johnson
 JBoss World 2005
 March 1, 2005




Imagine it. Done.

> Consulting.
 > Systems Integration.
 > Outsourcing.
 > Infrastructure.
 > Server Technology.

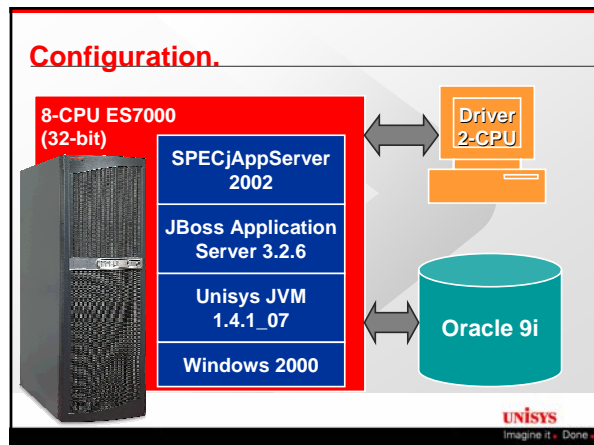


Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusion




Imagine it. Done.

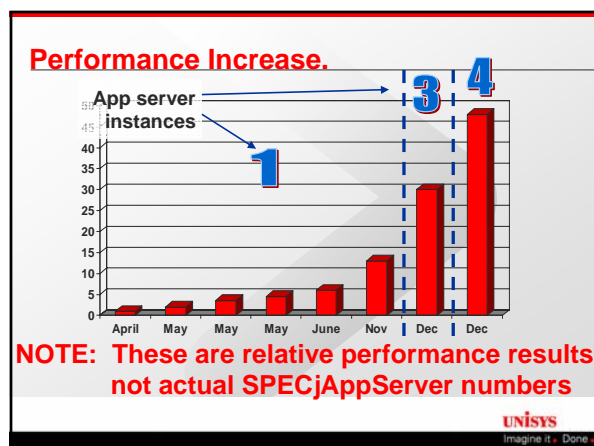



SPECjAppServer2002.

- > Industry standard client/server benchmark
 - Measures J2EE application server performance
 - Complete end-to-end web application
 - EJB 2.0
 - JDBC to relational database
 - Web services
- > Metrics:
 - TOPS – Total Operations Per Second
 - \$ per TOPS – price/performance
- > <http://www.spec.org/jAppServer2002/docs/FAQ.html>




Imagine it. Done.





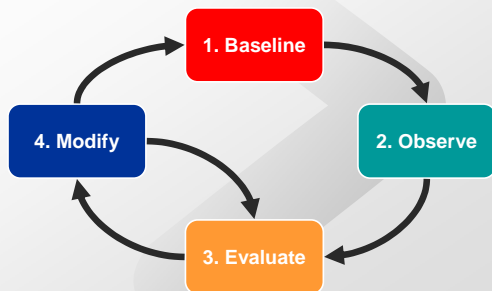
Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



Imagine it. Done.

Methodology.



UNISYS
Imagine it. Done.

SPECjAppServer2002 Statistics.

- > **Injection Rate (IR)** – benchmark driver parameter
 - Identifies pressure place on the application
 - Starts at 1 and goes up from there
 - Start at 1, and then increment by 5 or 10.
- > **Total Operations Per Second (TOPS)**
 - Benchmark result
 - Should be ~1.7 TOPS/IR
- > **Example**
 - IR=50, TOPS=60, TOPS/IR=1.1 → evaluate & modify
 - IR=50, TOPS=83, TOPS/IR=~1.7 → increase IR to 60
- > <http://www.spec.org/jAppServer2002>

UNISYS
Imagine it. Done.

Tuning Philosophy.

- > There is always another bottleneck
 - > Sometimes, reducing the bottleneck is sufficient
 - > Everything is fair game (almost):
 - Add more hardware (CPUs, NICs, memory, RAID disk, etc.)
 - Tweak the JVM options
 - Tweak the database settings
 - Tweak the application server settings
 - Tweak the application settings
 - ~~Modify the application~~
- Cost is taken into account in SPECjAppServer results
- Not allowed with standard benchmarks

UNISYS
Imagine it. Done.

Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



UNISYS
Imagine it. Done.

Hardware Tuning.

- > Plan for growth
- > Memory – you never have enough
 - 4GB (or more) on each server
- > Monitor system usage
 - Perfmon
 - Top
 - Others (commercial tools)
- > CPU usage
 - If over 80% => add another CPU

UNISYS
Imagine it. Done.

More Hardware Tuning.

- > Network considerations
 - Private network(s)
 - Fast NICs, multiple NICs
 - Set speed on NICs, don't use auto-detect
 - Routers and network configuration
- > Fast hard drives on database server
 - RAID array, 15,000 RPM SCSI
- > Watch out for domino effect
 - If app server host now has more capacity, beware the db host
- > Don't forget the driver system

UNISYS
Imagine it. Done.

Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



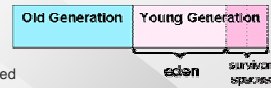
UNISYS

Imagine it. Done.

HotSpot™ Heap Primer.

> Young Generation

- Eden
 - Where new objects are created
- Survivor spaces
 - Where garbage collector places objects that are still in use



> Old Generation

- Where tenured objects are placed

> References

- <http://java.sun.com/docs/hotspot>

UNISYS

Imagine it. Done.

JVM Heap Tuning.

> Set the heap sizes

- Set min and max sizes to same value
- Set young generation to 1/3 size of heap
- Ex: -Xms1000m -Xmx1000m
 - XX:newSize=300m -XX:MaxNewSize=300m

> Monitor heap usage

- -verbose:gc
 - Basic before/after garbage collection heap sizes
- -XX:PrintHeapAtGC
 - Detailed before/after garbage collection heap size

> Set the heap sizes appropriately

UNISYS

Imagine it. Done.

Other JVM Options.

> Use thread-local allocation blocks

- Section of young generation allocated to each thread
- -XX:+UseTLAB
- Watch the young generation size
 - # threads * TLAB size < young generation size

> Delay the distributed garbage collection (DGC)

- DGC performs full garbage collection every 60 seconds
- -Dsun.rmi.dgc.client.gcInterval=3600000
- -Dsun.rmi.dgc.server.gcInterval=3600000

UNISYS

Imagine it. Done.

Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



UNISYS

Imagine it. Done.

Turn off Hot Deployment.

- > The SECjAppServer application doesn't change
- > No other applications will be deployed

jboss-service.xml

~~<attribute name="ScanPeriod">5000</attribute>~~

<attribute name="ScanEnabled">false</attribute>

UNISYS

Imagine it. Done.

Remove Unused Services.

- Deploy SPECjAppServer to the 'default' configuration
- Remove the following services:
 - hsqldb-ds.xml
 - scheduler-service.xml
 - snmp-adaptor.sar
 - monitoring-service.xml
 - mail-service.xml
 - schedule-manager-service.xml
 - jms

Connection Pool Size.

- Size of connection pool related to injection rate
- For IR up to 50
 - $\text{Size} = (8 * \text{IR}) + 10\%$
 - Example: IR = 10, $\text{Size} = (8 * 10) + 10\% = 88$
- For IR over 50
 - $\text{Size} = ((8 * \text{IR}) + 10\%) * 0.1$
 - Example: IR = 100
 - $\text{Size} = ((8 * 100) + 10\%) * 0.1 = (880) * 0.1 = 88$
- Set <min-pool-size> and <max-pool-size> to same value

Other Data Source Configuration.

- Increase <blocking-timeout-millis>
 - Prevents failed transactions
 - Recommended starting setting: 30000
- <prepared-statement-cache-size>
 - Allows for prepared statement reuse
 - Recommended size: 200

New Persistence Manager.

- standardjboss.xml
 - Container name: cmp2.x jdbc2 pm
 - Persistence manager: JDBCStoreManager2
- Two-level cache
 - Entity beans stored in a transaction-level cache
 - Global cache holds "database rows"
- Optimistic locking recommended
- <http://www.jboss.org/wiki/Wiki.jsp?page=CMP2xJDBC2PM>

Other Configuration.

- Remove metrics collector interceptor
 - Standard Stateful Session Bean
 - Standard Stateless Session Bean

standardjboss.xml

```
<interceptor transaction="Bean" metricsEnabled="true">  
  org.jboss.ejb.plugins.MetricsInterceptor  
</interceptor>  
<interceptor transaction="Container" metricsEnabled="true">  
  org.jboss.ejb.plugins.MetricsInterceptor  
</interceptor>
```

Multiple Application Servers.

- Symptoms
 - Low system utilization on all tiers
 - No obvious bottlenecks
 - Fairly high injection rate
- Add a second application server instance
 - Separate hardware if app server system maxed out
 - On ES7000 – added to same system due to scale-up architecture
 - Beware of port conflicts
- Performance boost expectation
 - 2 app servers => about 2 * single app server
 - N app servers => not quite N * single app server

#	IR*
1	80
2	150
3	210

* - These are hypothetical injection rates and should not be interpreted as actual or observed rates.

Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



Oracle 9i Tuning.

- > Buffer cache
 - Use Oracle Enterprise Manager (OEM) to access the **Memory** configuration for the database
 - Click on the **Suggestion** button for suggested changes
 - Refer to Oracle tuning guide for more info
- > Processes
 - Symptom: exceptions stating Oracle out of processes
 - Use OEM to change **Processes** parameter: 1000-1500
 - Also change **Sessions** and **Transactions**

Oracle 9i Tuning, cont.

- > Symptom: Insufficient connections
- > Cause: stack size too big
- > Fix:
 1. Shutdown TNS and database services
 2. Find out size of existing stacks:
 - orastack tnslnr.exe
 - orastack oracle.exeShould see size of 1 MB
 3. Change stack size to 0.5 MB:
 - orastack tnslnr.exe 524288
 - orastack oracle.exe 524288
 4. Restart TNS and database services

WARNING
Do not go lower
than 0.5 MB

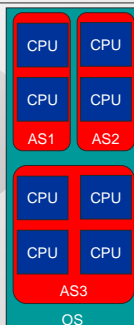
Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



Process and Thread Affinity.

- > Contain process/threads to run on certain processors
 - Keep threads within processor group if external cache
- > Process affinity
 - Useful if threads share many common objects
 - J2EE-based application servers
- > Thread affinity
 - Useful if threads are independent
 - Useful in scale-up scenario
- > Recommended: 2-4 CPUs/app server



Agenda.

- > Introduction
- > Methodology
- > Hardware Tuning
- > JVM Tuning
- > Application Server Tuning
- > Database Tuning
- > Miscellaneous Tuning
- > Conclusions



Concluding Thoughts.

- Significant performance and scalability improvements went into JBoss Application Server 3.2.6
- Performance testing is an iterative process – you don't run the benchmark just once
- There is always another bottleneck
- You too can tweak your system to improve its performance
- <http://www.jboss.org/wiki/Wiki.jsp?page=SPECJAppServer2002Tuning>

UNISYS
Imagine it • Done •

Questions?



➤ Consulting.
➤ Systems Integration.
➤ Outsourcing.
➤ Infrastructure.
➤ Server Technology.

UNISYS
Imagine it • Done •