IBM Software Group

Advance Performance Tuning for WebSphere Application Server – Part1

Surya Duggirala: suryadu@us.ibm.com

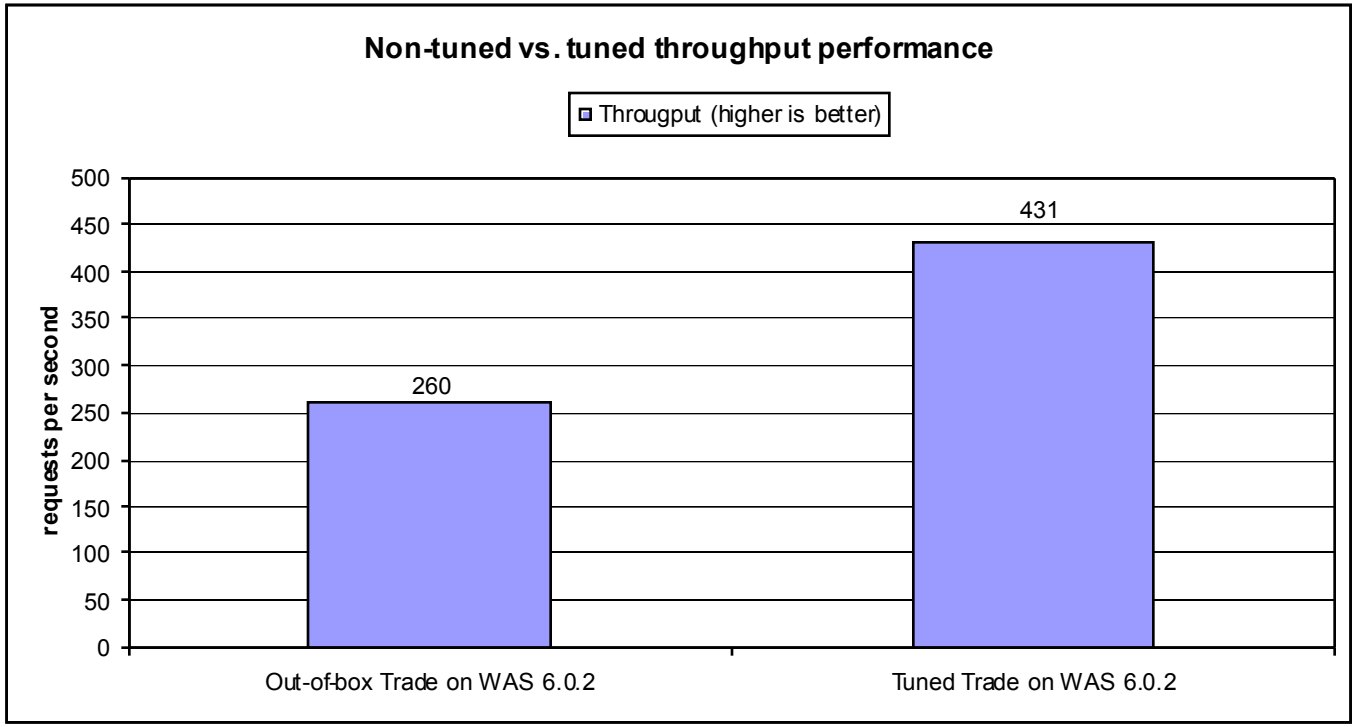
Bhushan Lokhande: lokhande@us.ibm.com

WebSphere® Support Technical Exchange

Purpose of meeting

- After completing this session, you should be able to:
  - Understand how to tune and use key J2EE components for optimal performance
  - Identify performance issues unique to administrators
  - Discuss performance issues and trade-offs

Out-of-box vs. Tuned application performance



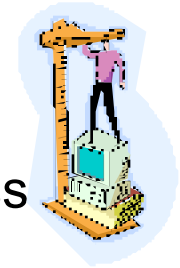
66% faster performance after tuning!!

Caution – Different environments will yield different results

Agenda

- JVM
- HttpSession
- EJB
- JMS
- Questions

JVM Sizing

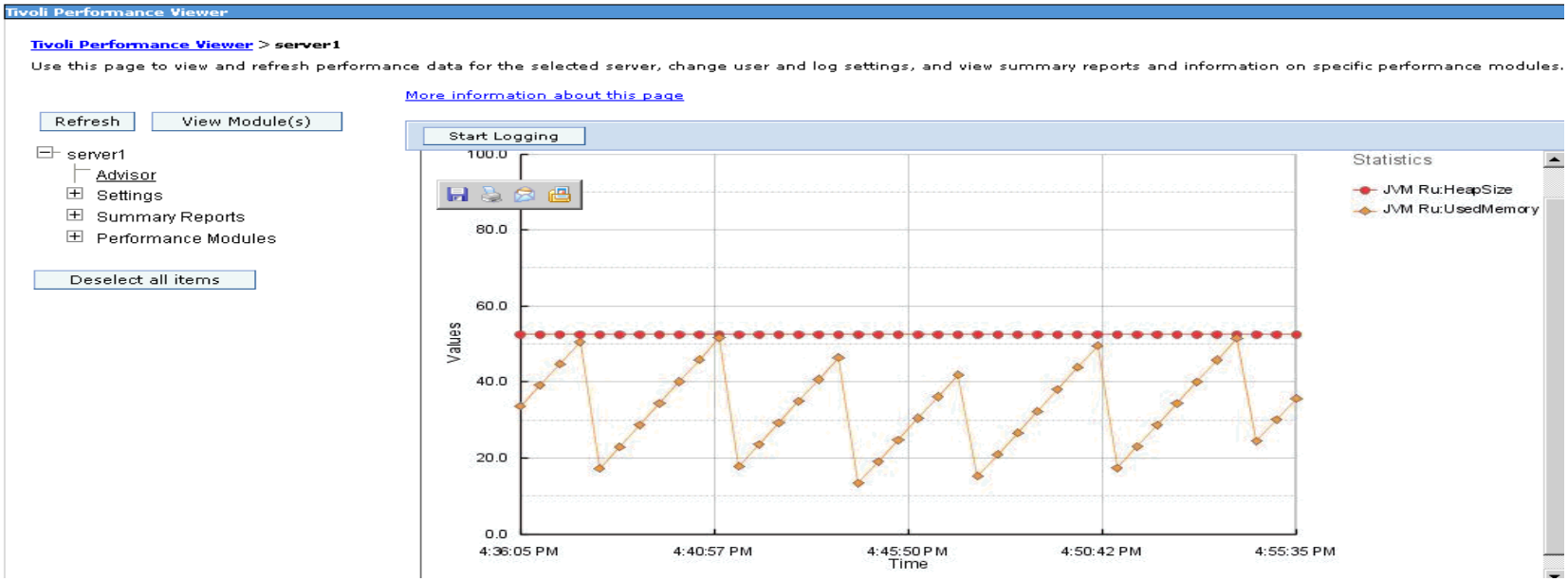


- Ensure that required resources are available to WAS JVM process
  - Always keep the JVM(s) within the physical memory of the server
- Some starting points:
  - -Xmx=512M for WAS
  - -Xmx=1GB for WP
- Minimum heapsize
  - Production systems set –Xms lower than –Xmx
    - Gives headroom for emergencies
    - In theory, results in more efficient object table
  - “Burst” testing requires –Xms=-Xmx
- 1.4x IBM JVMs more efficient than predecessors
  - Shorter GC pauses, even for large heaps
  - More tuning parameters does not equate to better performance!

Tuning Garbage Collection

Smaller heap size	Shorter but frequent GCs
Large heap size	Less frequent but longer GCs

- Average time between garbage collections should be at least 5-6 times the average duration of a single garbage collection



Monitoring GC

Application Servers > server1 > Process Definition >

Java Virtual Machine

Advanced Java virtual machine settings.

Configuration

General Properties	
Classpath	
Boot Classpath	
Verbose class loading	☐
Verbose garbage collection	☑
Verbose JNI	☐

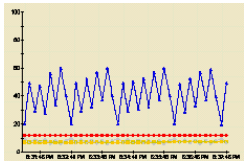
```
<af type="tenured" id="86" timestamp="Mon May 22 15:31:16 2006" intervalms="2066.497">
  <minimum requested_bytes="32" />
  <time exclusiveaccessms="0.449" />
  <tenured freebytes="0" totalbytes="268435456" percent="0" >
    <soa freebytes="0" totalbytes="268435456" percent="0" />
    <loa freebytes="0" totalbytes="0" percent="0" />
  </tenured>
  <gc type="global" id="86" totalid="86" intervalms="2073.651">
    <refs_cleared soft="0" weak="362" phantom="0" />
    <finalization objectsqueued="1692" />
    <timesms mark="237.430" sweep="7.839" compact="0.000" total="245.582" />
    <tenured freebytes="173378944" totalbytes="268435456" percent="64" >
      <soa freebytes="173378944" totalbytes="268435456" percent="64" />
      <loa freebytes="0" totalbytes="0" percent="0" />
    </tenured>
  </gc>
  <tenured freebytes="173377928" totalbytes="268435456" percent="64" >
    <soa freebytes="173377928" totalbytes="268435456" percent="64" />
    <loa freebytes="0" totalbytes="0" percent="0" />
  </tenured>
  <time totalms="246.569" />
</af>
```

GC interval 2066 ms

Memory freed after GC cyde = 64%

GC Duration 246 ms

Tuning Garbage Collection



- Start with -Xms and -Xmx set to 512 MB
  - Apply stress.
  - Increase both values until memory utilization reaches stable level.
  - This is the amount of memory required by the application.
- Set –Xmx close to the stabilization level.
  - Example - If stabilization level is 1.1 GB, set –Xmx to 1024 MB
- Set –Xms 50% of –Xmx
- Monitor Garbage collection statistics.
- Tune –Xms and –Xmx values to optimize time spent in garbage collection.

Controlling Memory Fragmentation with the IBM JVM

- Pinned and dosed objects are unmovable during compaction
  - Example: Native calls generate pinned objects
- Heap fragmentation could become an issue
  - Especially true with large objects
  - OutOfMemory occurs even if there are free space in the heap
- pCluster is an area for pinned objects
  - 16K by default
  - Newly created pCluster are 2K in size
  - Can be reset using –Xp<iiii>[K][,<oooo>[K]]
    - iiii = initial pCluster size
    - oooo for the size of subsequent overflow pClusters
- kCluster is an area of storage for class blocks
  - Default size = 1280 (entries) x 256 (bytes)
  - Can be reset via –Xk<size>

Agenda

- JVM
- HttpSession
- EJB
- JMS
- Questions

HttpSession Management

- Determine the average HttpSession size
  - Session data shares the JVM heap with applications. Large HttpSessions impact scalability
  - Use the Tivoli Performance Viewer (TPV) during *test phase*
- Size the session cache correctly
  - Overflow sessions not retrieved as efficiently
  - Written to external store if persistence enabled
- Consider enhancements for persistence
  - Use time-based writing (default)
    - Some exposure for data loss
    - Significant performance gains
  - Write only updates, not the full session repeatedly

HttpSession Persistence

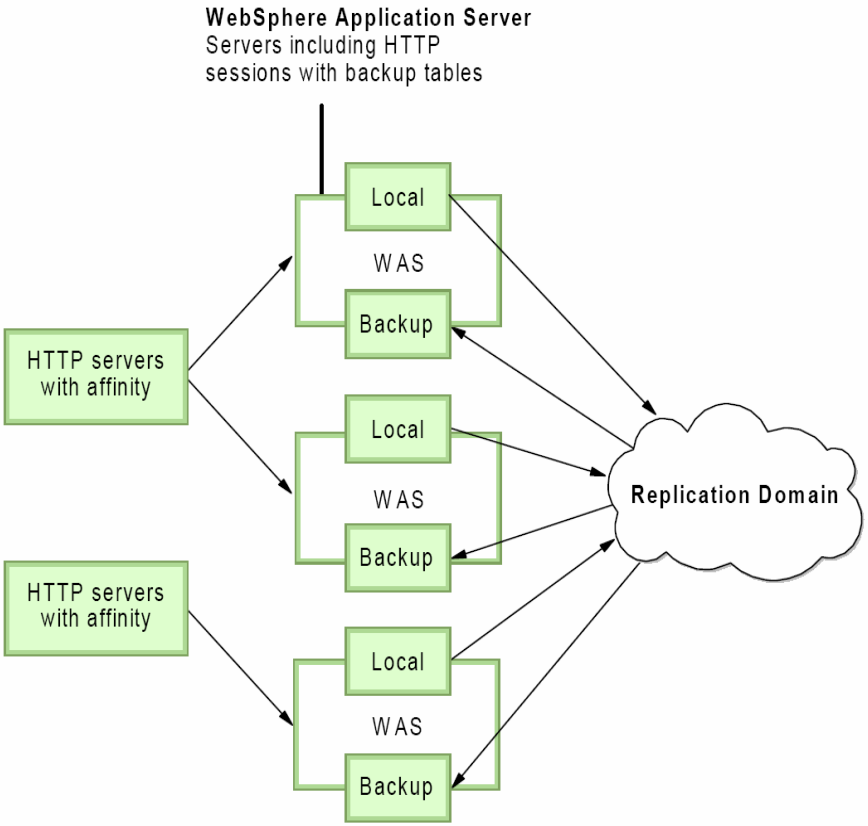
- Memory-to-Memory replication
  - Memory-to-Memory best for:
    - Team environments
    - Smaller sites with well-understood loading. Sessions are going to live in heap space.
  - Simplified configuration
  - Is Memory-to-Memory Copy faster than Database Persistence in V6?
    - **No**
  - Most expensive persistence operation remains **Network Serialization**
- Database persistence more scalable and reliable
  - No practical limitations on supported sessions
  - Database solution stores session data to a hard drive

Memory-to-Memory replication

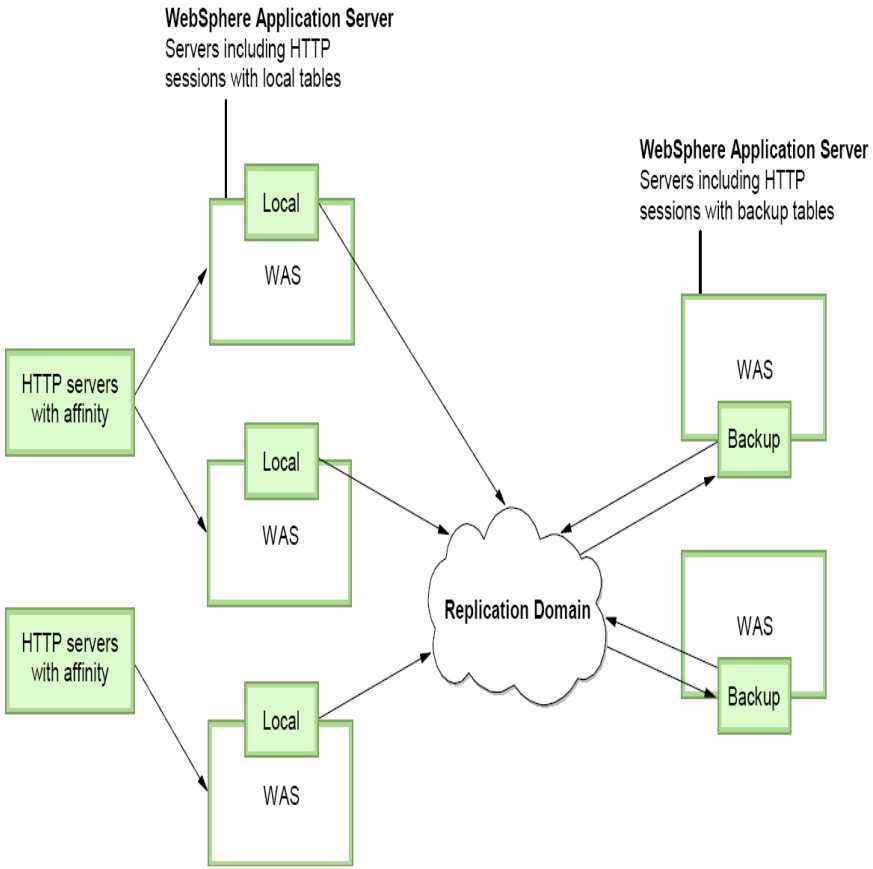
- Topology changes
  - No more replicas and partitions, instead we have client and servers in a replication domain
- One Data Replication Service (DRS) instance runs on a server for each Replication Domain
- A DRS instance can run in three modes
  - Server mode
  - Client mode
  - Both mode
- Replication domain topologies for memory-memory replication
  - Peer-to-peer replication
  - Client/server replication

Memory-to-Memory replication topologies

Peer-to-peer replication



Client/server replication



Agenda

- JVM
- HttpSession
- EJB
- JMS
- Questions

EJBs

- New Performance Features in the product
  - Partial Updates for CMPs
    - Can substantially increase the performance in certain scenarios
    - May hurt the performance if not used properly
  - Keep Update Locks (KUL)
    - DB2 8.2 supports KUL and can realize significant performance boost with this option
  - Read Only Beans
    - A new type of entity bean that extends the caching options
    - Can experience significant performance boost for applications that use data that changes relatively infrequently

EJBs

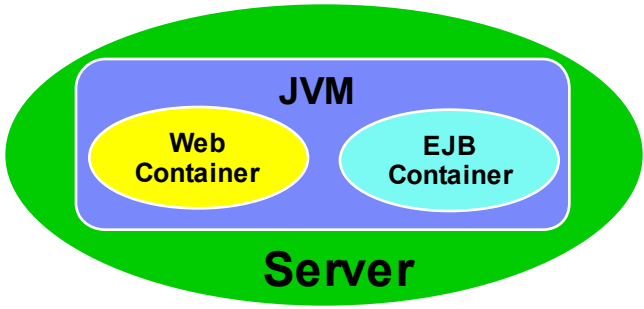
- EJB Container Cache tuning
  - Choose sufficient cache size to avoid frequent passivation
  - **EJB_Cache_Size** = (Largest number of Option B or C Entity Beans enlisted in a transaction * maximum number of concurrent transactions) + (Largest number of unique Option A Entity Beans expected to be accessed during typical application workload) + (Number of stateful Session Beans active during typical workload) + (Number of stateless SessionBean types used during typical workload)
- Use persistent Manager Batch Updates accessing multiple CMPs in a single transaction
  - -Dcom.ibm.ws.pm.batch=true
- Use deferred create for container managed persistence
- New in WAS v6.0
  - Lazy initialization of EJBs
  - Faster server startup with large EJB application

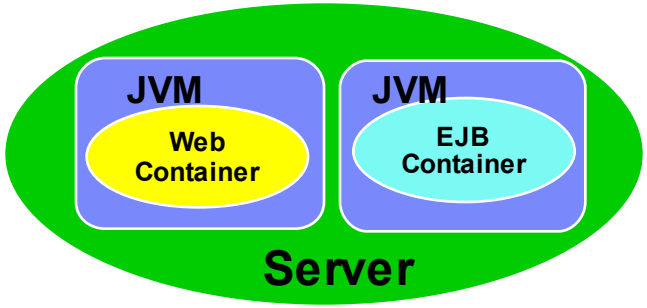
EJBs

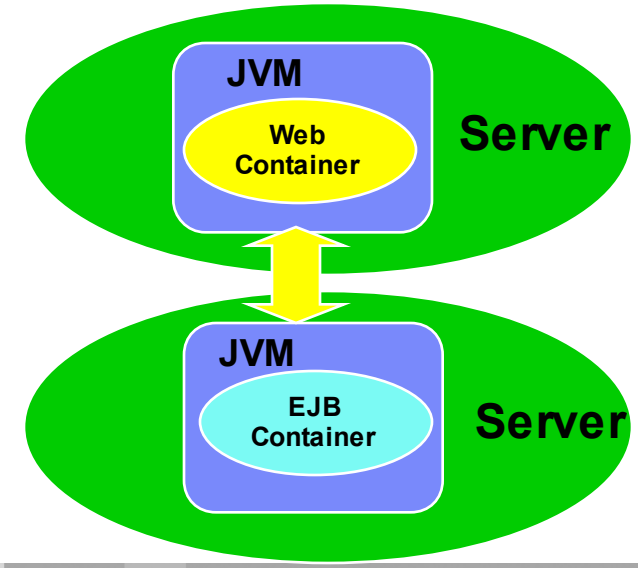
- Local Beans
  - Pass-by-reference J2EE API
  - Remote deployment restricted
  - Provides significant performance gains vs. pass by copy
    - Not significantly greater than pass-by-reference at ORB level
- ORB pass-by-reference
  - Set via Admin console
  - Applies pass-by-reference to all co-resident EJBs
    - If using same class loader, otherwise automatically pass-by-value
  - Beware of side effects!!
    - EJB spec assumes pass-by-value except for Local Beans
    - May actually modify parameter objects
- EJB coding may restrict deployment options!

EJB Deployment

- Co-locating EJB and Web Containers in the same JVM
  - High performing option
- Split-tier deployment
  - Performance degradation
  - High Availability option







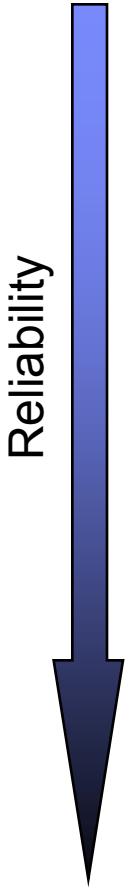
Agenda

- JVM
- HttpSession
- EJB
- JMS
- Questions

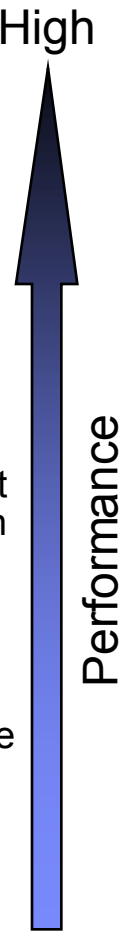
JMS

- JMS messaging support modes
  - In-process with WAS in V6
    - Reduces messaging overhead for light-weight messaging
    - Non-persistent messaging requires no external process interaction
    - Reduces overhead in serialization to database
  - Support with external MQSeries or embedded MQ
  - Support with any JMS compatible Messaging Engine
- Message integrity is normally the main criteria
  - Use transactional support for multiple, inter-dependent messages
  - Otherwise, non-transactional messaging (default) more efficient

JMS (Reliability vs performance)

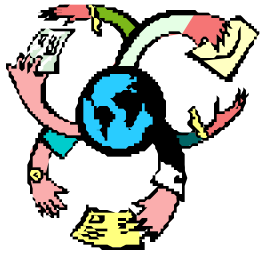


- BEST_EFFORT_NONPERSISTENT
  - Messages are never written to disk
  - throw away messages if memory cache over-runs
- EXPRESS_NONPERSISTENT
  - Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts
  - No acknowledgement that the ME has received the message
- RELIABLE_NONPERSISTENT
  - Same as Express_Nonpersistent, except, we have a low level acknowledgement message that the client code waits for, before returning to the application with an OK or not OK response
- RELIABLE_PERSISTENT
  - Messages are written asynchronously to persistent storage during normal processing, and stay persisted over server restarts.
  - If the server fails, messages might be lost if they are only held in the cache at the time of failure.
- ASSURED_PERSISTENT
  - Highest degree of reliability where assured delivery is supported



JMS

- 1 Phase Commit Optimization
  - Sharing connections with CMPs
  - Significant performance boost (about 15%)
  - Can not be used for BMPs and JDBC apps
  - Application Res Auth performs better than the Container Managed Res Auth



JMS

- MDB Performance tips
  - Tune the default thread pool
  - Deliver batches of messages to each MDB endpoint to improve performance
  - Change the default Cloudscape database to a remote DB2 database for better performance
  - Expect up to 50% higher throughput with remote DB2 compared to local DB2
  - Adjust the JDBC connection pool size and preparedStatementCache size

JMS

- Data store Performance tuning
  - Sufficiently increase the JDBC connection pool to cope with peak loads
  - Place database logs on a fast RAID array to reduce the write latency
  - consider putting the SIBxxx tables into a tablespace with 32KB pages
  - adjust the column width of the VARCHAR column to 32032 bytes.

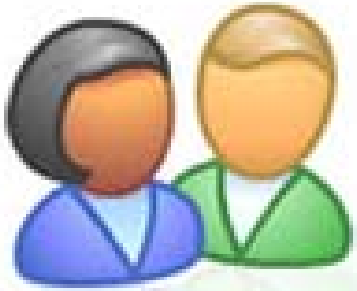
Summary

- New performance features should be exploited to improve the enterprise application performance wherever applicable.
- JVM heap size is the key tuning parameter. IBM JVM 1.4.x is more efficient than predecessors.
- Memory-to-Memory session replication is much improved, easier to configure and works best for smaller sites.
- Don’t use the highest Reliability level (if not necessary for the application) as it will lower the performance.
- For best performance of persistence messaging, data store tuning is essential.

Additional WebSphere Product Resources

- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: www.ibm.com/developerworks/websphere/community/
- Learn about other upcoming webcasts, conferences and events: www.ibm.com/software/websphere/events_1.html
- Join the Global WebSphere User Group Community: www.websphere.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant: www.ibm.com/software/info/education/assistant
- Learn about the Electronic Service Request (ESR) tool for submitting problems electronically: www.ibm.com/software/support/viewlet/ESR_Overview_viewlet_swf.html
- Sign up to receive weekly technical My support emails: www.ibm.com/software/support/einfo.html

Questions and Answers



Thank You!!

Additional Resources

- **Additional reports and papers:**
  - http://w3quickplace.lotus.com/wasperf
- **WebSphere Application Server Performance Web Site**
  - http://www.ibm.com/software/webservers/appserv/performance.html
  - Performance guides – includes papers describing how to maximize performance
- **Redbooks**
  - http://www.redbooks.ibm.com
  - Best Practices for High-Volume Web Sites
  - DB2 UDB/WebSphere Performance Tuning Guide
  - IBM WebSphere V6.0 Performance, Scalability, and High Availability WebSphere Handbook
  - WebSphere Version 6 Application Development Handbook
- **WebSphere Application Server InfoCenter**
  - http://www.ibm.com/software/webservers/appserv/infocenter.html
  - WebSphere Application Server V6 Tuning Guide
- **Joines, Stacy; Willenborg, Ruth; Hygh, Ken. Performance Analysis for Java Web Sites. Reading, MA: Addison-Wesley, 2003**

Performance Tuning Hot List

- Check hardware configuration and settings
- Review application design
- Tune the operating system
- Configure the Java heap size
- Tune WebSphere JDBC data sources
- Use a Type 4 (or pure Java) JDBC driver
- Enable the ‘pass by reference’ ORB option
- Ensure that the Transaction log is assigned to a fast disk.
- Tune related components such as: web server and database
- See the Tuning Performance section of the WebSphere V6 InfoCenter at http://publib.boulder.ibm.com/infocenter/ws60help/topic/com.ibm.websphere.nd.doc/info/welcome_nd.html

Tivoli Performance Viewer Overview

- Tivoli Performance Viewer provides visual monitoring for:
  - Application resources (servlets, Enterprise JavaBeans)
  - WebSphere Application Server runtime resources (thread pools, JVM memory)
  - Other products or applications that use the PMI API to provide metrics
- TPV for V6.0 now tightly integrated with WebSphere Admin Console
  - Thin client (browser access)
  - In-process with WebSphere
  - Minimal overhead
- New V6.0 capabilities
  - Monitoring and logging on per-server basis
  - Refresh rate and buffer size on a per-server and per-user basis
- Operations:
  - Polling
  - Buffering
  - Displaying
  - Logging

Performance Advisors Overview

- WebSphere Application Server provides tunable settings to optimize performance
- Tuning requires analysis and correct action
- Two performance Advisors provide analysis and recommend action
  - Runtime Performance Advisor
  - Tivoli Performance Advisor