

IBM WebSphere Web Multi-Platform Configuration

WebSphere Application Server Performance



What Will Be Covered



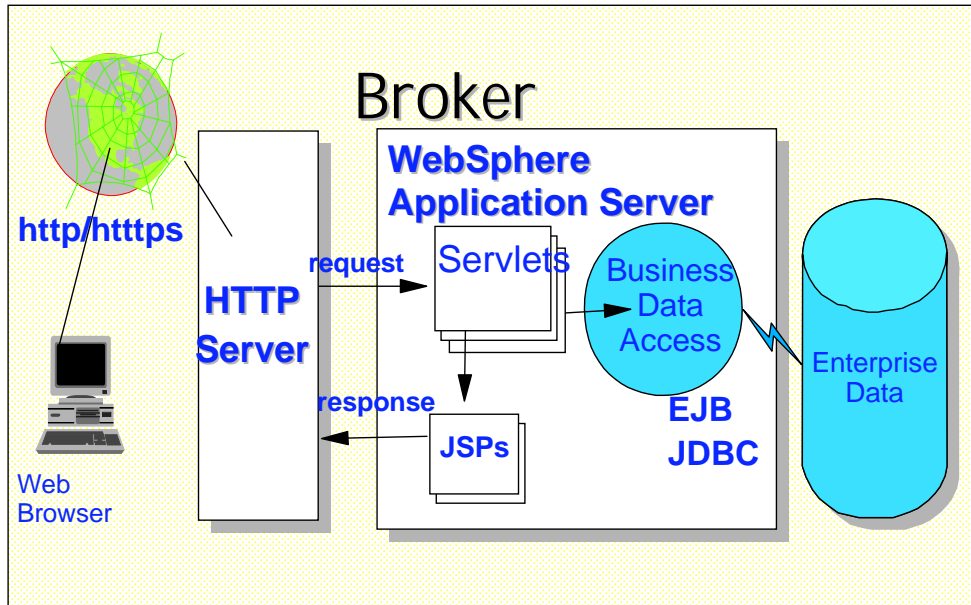
- Performance Benchmarks
 - ▲ What to Expect
 - ▲ How to Scale
- Improving Performance
 - ▲ Tuning
 - ▲ Troubleshooting
- For More Information



Performance Benchmarks



Broker Application Topology



- The Broker application is used for tests conducted by the WebSphere Performance Team in Raleigh, NC.
- This sample application is designed to test the WebSphere Application Server for scalability, performance and competitiveness.

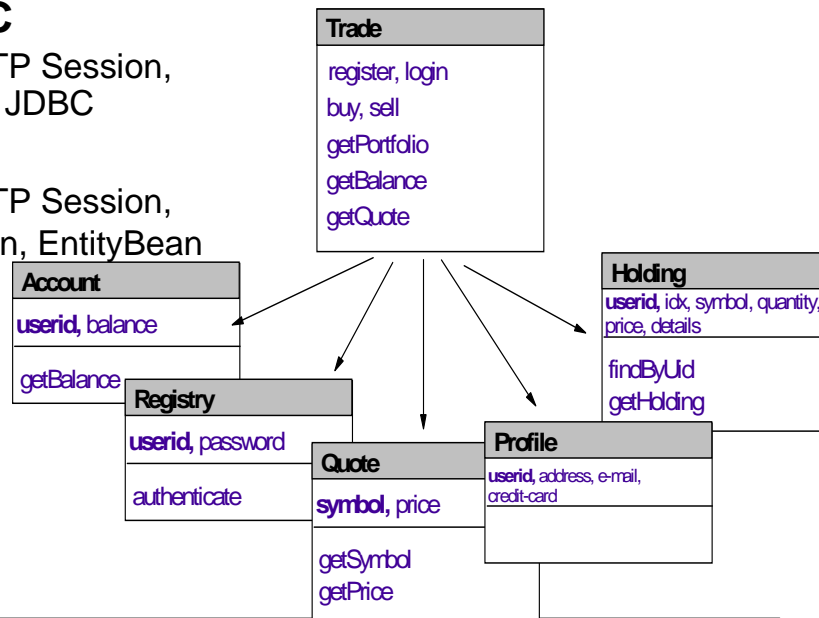
Broker Test Application

Broker JDBC

- ▶ Servlet, HTTP Session, DataBeans, JDBC

Broker EJB

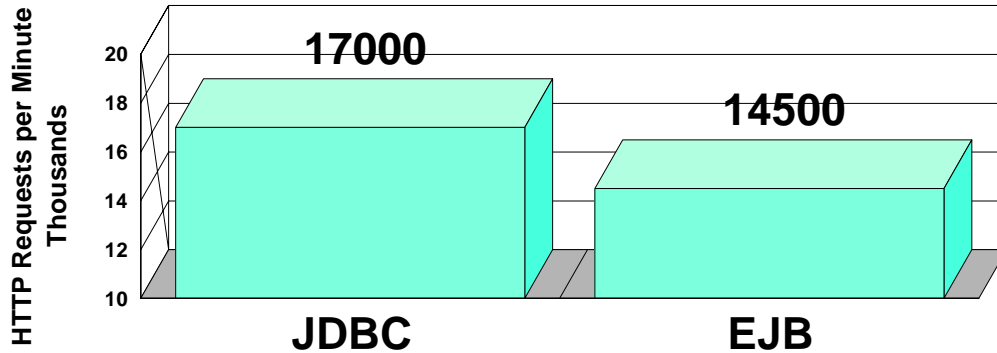
- ▶ Servlet, HTTP Session, SessionBean, EntityBean



- The Broker application simulates an online stock trading web site.
 - The Trade servlet is the controller for the application.
 - The five other objects are data objects: Account, Registry, Quote, Profile, and Holding.
- When the Broker application is configured to use pure JDBC, these 5 objects are data access beans that use JDBC prepared SQL statements to manipulate data within a database table.
- When the Broker application is configured to use EJBs, these 5 objects are container-managed Entity EJBs.

JDBC vs. EJB Database Access

Broker Application - Windows NT



System configuration: NetFinity 5500 400MHz 2-way 2048M RAM, Windows NT 4.0, 100Mb Ethernet
Server: IIS+WAS 3.0 Adv, IBM JDK 117p
Application Server Parameters: -ms128m; -mx128m
Test Driver: AKstress; 100 client load, 25,000 page hits <1K data transmitted per page hit

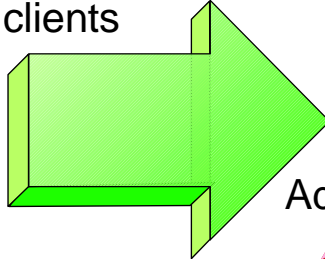


- When the Broker application is configured with data access beans that use JDBC to access the database, performance is about 15% greater than when configured using EJBs.
- Using EJB technology impacts performance, but offers portability and flexibility that are critically important to large, complex e-business applications.

WebSphere Scaling Alternatives

To Improve:

- ▲ Response time
- ▲ Throughput
- ▲ Concurrent clients



Add:

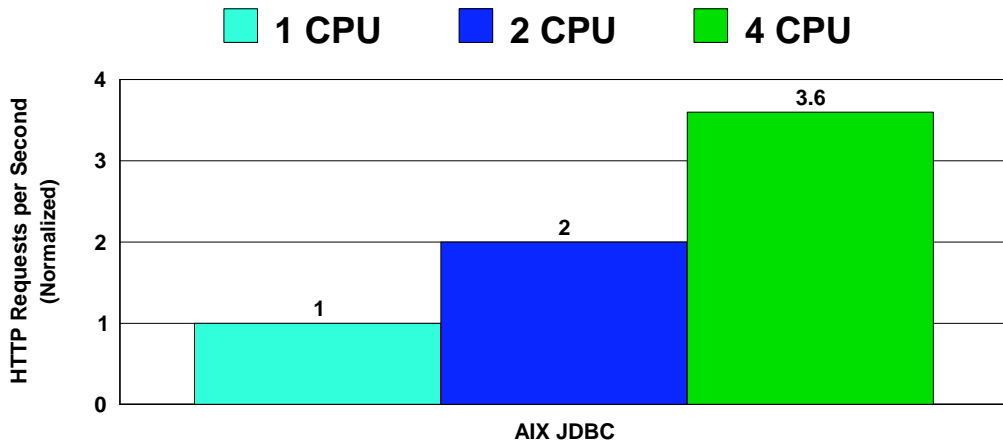
- ▲ Faster processor
- ▲ More processors
- ▲ More memory
- ▲ More servers



When your operating system, network, database, and application are at peak performance, adding capacity (faster processors, more processors, more memory, more servers) is the best way to increase performance.

- You should know how long it takes to serve a static HTML page in your configuration. A simple static page will show the best performance number you can hope to achieve.

Vertical Scaling Without Clones



CPU	100%	100%	99%
-----	------	------	-----

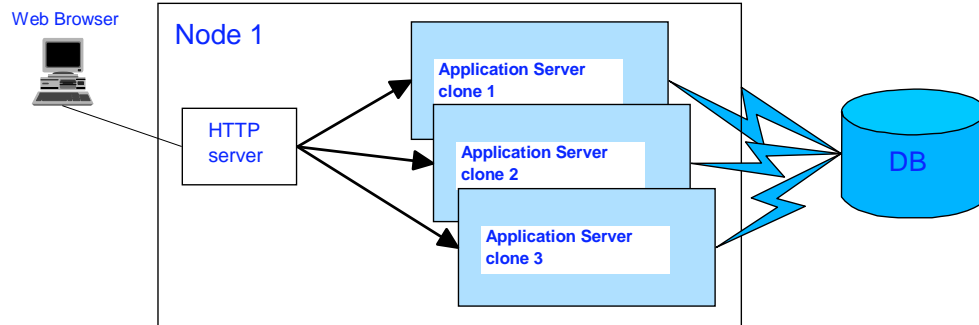
AIX System configuration: RS/6000 F50 166MHz 4-way, 1024M RAM, AIX 4.3.2, 100MbEthernet
AIX Application Server: IHS+WAS Adv 3.0, IBM JDK 116
Test Driver: AKstress; 100 client load, 25,000 page hits <1K data transmitted per page hit



- Adding more processors provides near linear scaling. You may see a greater percentage performance increase when going from 1 processor to 2 processors than you get going from 2 processors to 4 processors.

Note: These figures were obtained on a single node by disabling all but one processor and taking measurements. Additional processors were enabled and measurements taken with each addition. The test performed attempts to stress the CPU(s) to reach 100% CPU usage.

Vertical Scaling With Clones



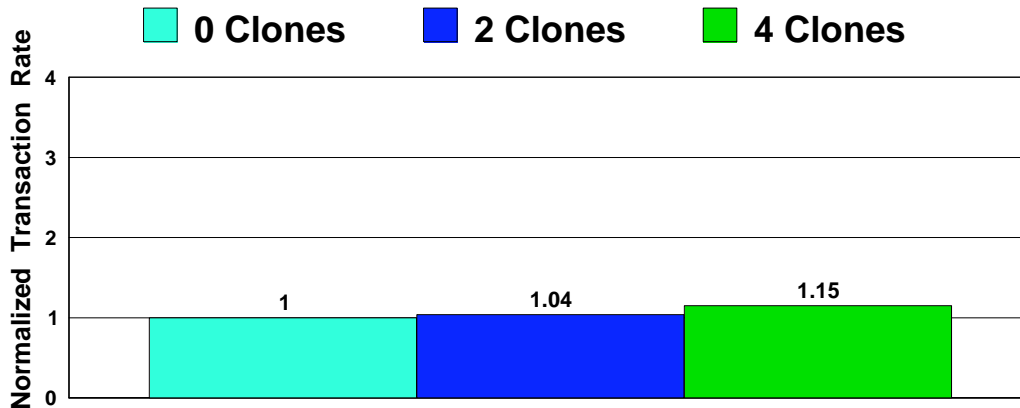
Multiple WebSphere Application Servers on one machine

- ▲ Plug-in provides load balancing across clones
- ▲ Higher transaction rates on large SMP machines



- Vertical scaling with clones is also known as Servlet Clustering on one node.
- Higher transaction rates are achieved on machines with 12 or more processors and multiple clones of the application.
- Adding clones will increase performance on a single node **only** if that machine is not already running at 100% CPU capacity.

Vertical Scaling with Clones (4-way processor)



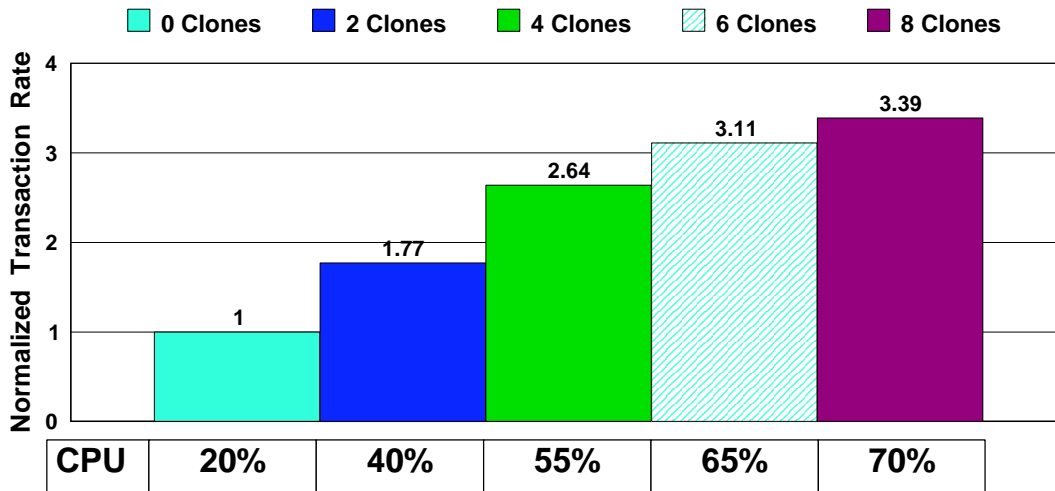
CPU%	CPU%	CPU%	CPU%
Near 100%	Near 100%	Near 100%	Near 100%

NT System configuration: NetFinity 7000, 1024M RAM, NT 4.0, 100MbEthernet
NT Application Server: IHS+WAS Adv 3.0, IBM JDK 117
Test Driver: AKstress; 100 client load, <1K data transmitted per page hit



- The slight gain in performance is due to garbage collection. Each clone has its own JVM, which performs garbage collection independently of the other clones.
- Note that CPU use is near 100% in each case from 0 to 4 clones.
- Moral of the story: if you're already at or near 100% CPU use, adding clones produces nominal results.

Vertical Scaling with Clones (12-way processor)



AIX System configuration: RS/6000 S80, 12-way, AIX 4.3.2, 100MbEthernet
AIX Application Server: IHS+WAS Adv 3.0, IBM JDK 116
Test Driver: AKstress; 100 client load, <1K data transmitted per page hit

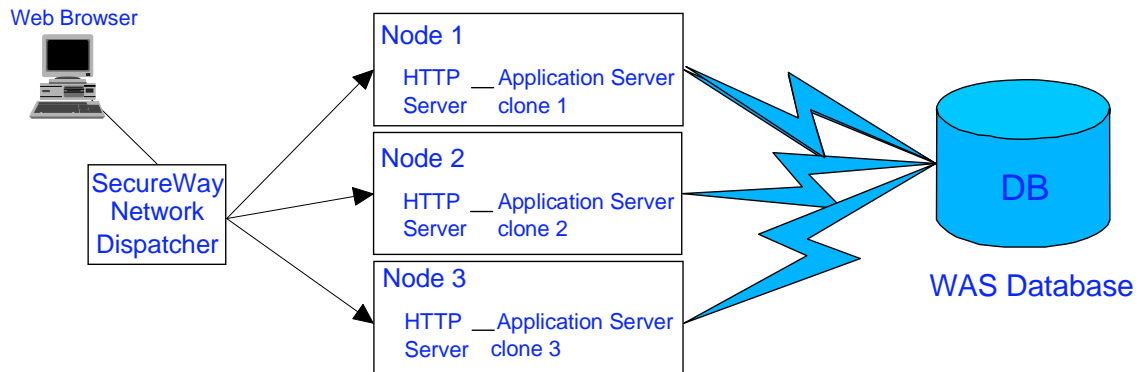


- Note that CPU utilization never reaches 100%.
- You get diminishing performance returns as you add more clones.

<u>Change in Clones</u>	<u>Performance Increase</u>
• 0 - 2	77%
• 2 - 4	49%
• 4 - 8	28%

- These tests were performed using JDK 1.1.6. Preliminary tests with versions of WebSphere that support JDK 1.1.8 or JDK 1.2 show scaling improvements, in general.

Horizontal Scaling - Multi-Node



Multiple HTTP/WAS Servers

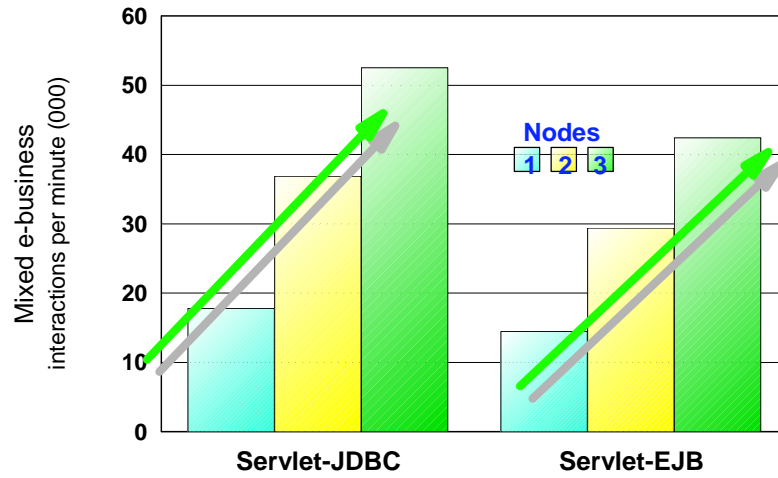
- ▲ SecureWay Network Dispatcher balances load across HTTP servers
- ▲ Near linear scaling for number of users supported



- Horizontal scaling can be used with several workload management configurations:

- TCP/IP spraying with Network Dispatcher
- Servlet Redirector
- OSE Remote

Horizontal Scaling - Performance



System configuration: NetFinity 5500 400MHz 2-way, 2G RAM, NT 4.0 100MbEthernet
Server: Microsoft IIS + WAS 3.0, JDK 1.1.7p
Test Driver: AKstress 300 client load, <1K data transmitted



Improving Performance



Improving Performance



- It all starts with the application
- There is no "magic" knob



Application Tuning



- Applications must be designed for performance
 - ▲ Avoid serialization
 - ▲ Minimize size and complexity of objects
 - ▲ Avoid string concatenation
 - ▲ Careful object creation and reuse

- Application rewrites can yield 2x to 5x performance improvements



No Magic Knob



- The OS, HTTP Server, application, network, and database must be performing at their peak.
- Performance tuning depends on the application and the environment into which it is deployed.
- Tuning inside WebSphere may yield a 5% to 20% performance gain.

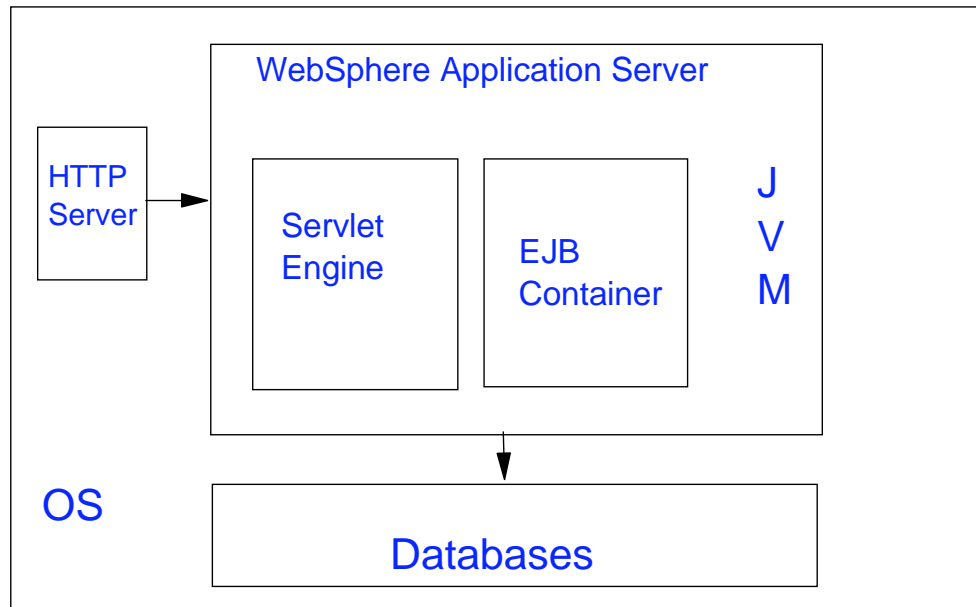


- Remember, the greatest performance gains are obtained by ensuring the application is well written.

WebSphere Tuning



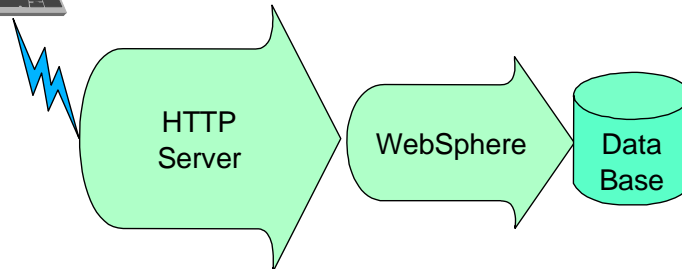
WebSphere Tuning



- Tuning WebSphere is interrelated with tuning of the HTTP server, JVM, Database, and OS.
- The *WebSphere V3 Performance Tuning Guide, SG24-5657*, is the best place to start when you want to get the best performance from your application environment.
- The most important relationships:
 - Number of threads in HTTP server (static and dynamic requests)
 - Number of threads in WebSphere--- controls pipe from HTTP server to WebSphere (plug-in will queue)
 - Number of database connections available for the application server (Use resource analyzer to view Database connections in use)

Coordinating Thread Pools

Web Browser



HTTP Threads \geq WAS Threads \geq DB Connections
More threads do not necessarily increase performance



- HTTP thread pool must be of sufficient size to handle static and dynamic requests
- WebSphere threads are specified in MaxConnections for Servlet Engine. For best performance, WebSphere connections should be set to a value less than or equal to the number of HTTP connections.
- The Database connection pool must be large enough to handle dynamic requests, but should be less than or equal to the number specified in the MaxConnections parameter for the Servlet Engine.

HTTP Server Tuning

- Throttle the WebServer to limit the number of simultaneous HTTP requests served
 - ▲ IBM HTTP Server
 - Set MaxClients parameter on AIX and Solaris to 50, then test at 40 and 60 (default is 150)
 - ▲ Netscape Enterprise Server
 - Set Active Threads high enough so that new connections will be serviced. (default is 512)
 - ▲ Failed connections on IIS under load
 - Raise ListenBackLog parameter to keep more requests in queue (default is 25)



On IBM HTTP Server, set the MaxClients parameter by editing the httpd.conf file in the <HTTP_home>/conf directory. Test at 50 clients, then try 40 clients and 60 clients and compare the results. The value of this directive can have a significant impact on your application performance, particularly if the value is set too high. Do not increase MaxClients if CPU use is at 100% and doing so causes server response time to exceed your response time criteria (default is 150).

On Netscape Enterprise Server, test the "maximum number of simultaneous requests parameter to see if it is too low.

- The default active thread limit is 512.
- This setting corresponds to the MaxRqThrottle parameter in the magnus.conf file.

Tuning the WebSphere Engine



WebSphere Tuning



- There are 5 areas of concern when tuning WebSphere:
 - ▲ JVM
 - ▲ Transport Queue
 - ▲ Servlet Auto Reload
 - ▲ EJB Container
 - ▲ ORB



JVM Heap Size

- Define the heap size for Admin Server and Application Server JVMs
 - ▲ Each clone has its own JVM

- Setting Min and Max heap size
 - ▲ For repeatable performance runs, set java.mx equal to java.ms (except on Solaris)
 - ▲ For production, set java.ms less than java.mx

- Tailor heap size to your needs
 - ▲ Too much heap extends garbage collection
 - ▲ Too little heap chokes the application



- Setting the minimum and maximum heap size the same during test runs produces highly repeatable results by eliminating heap growth.
Note: Sun recommends that you never set ms and mx to be the same.
- In the production environment, a good starting point would be to set the maximum heap size to 1/4 the total physical memory on the server. Set the minimum heap size to 1/2 the maximum heap size.
Note: Sun recommends that ms be set to somewhere between 1/10 and 1/4 of the mx setting.
- Remember, bigger is not necessarily better when it comes to heap size. Increasing heap size can increase the amount of time between garbage collection events. This can result in performance improvement. However, if mx is set too high, pause times will increase.
- In general, maximum heap size should not exceed 256MB without extensive testing using the `-gcverbose` command line argument.
- There is an ongoing debate about heap size:
 - Sun and Microsoft JVM folks say you should keep the heap small--(better control if they allocate it themselves).
 - JDKs are continuously improving.
 - Your mileage may vary.
- Set Minimum heap size to prevent garbage collection from occurring during startup.
- Set Maximum heap size to prevent the total of all heaps from exceeding physical memory.
- If the heap begins swapping to disk, Java performance suffers drastically.
- Default Min heap of 1MB makes startup slow

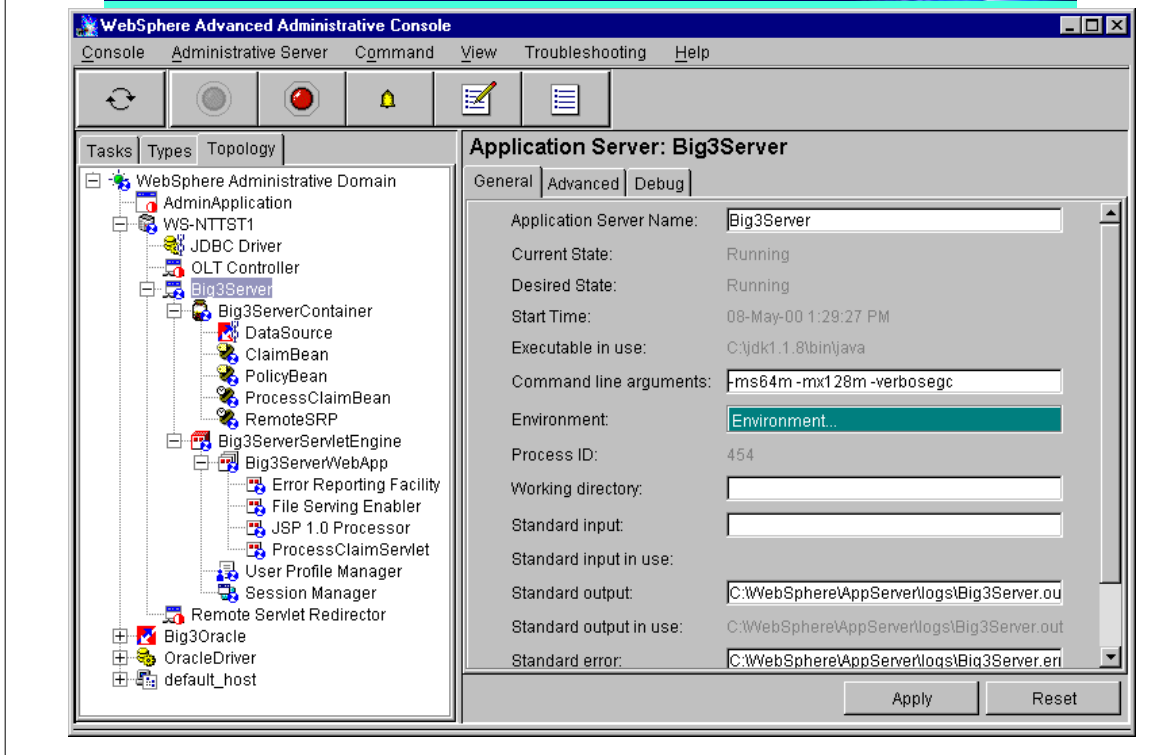
JVM Garbage Collection



- Garbage collection (GC) can take *several seconds* on a large heap.
- Efficient object management reduces GC times:
 - ▲ Eliminate excessive string use
 - ▲ Avoid unnecessary string concatenation
 - ▲ Reuse objects whenever possible
- Explicitly call GC if necessary



Set Application Server JVM Heap



Set Application Server JVM heap sizes with command line arguments on General properties tab.

- ms64m Minimum heap size(default is 1MB)
- ms128m Maximum heap size (default is 128MB)
- noasyncgc Turn off asynchronous garbage collection
- noclassgc Turn off class garbage collection
- verbosegc Verbose garbage collection for testing
 - Output for garbage collection is logged to the standard error file for the application server

Set Admin Server JVM Heap

- Admin Server JVM heap size is set with command line arguments in admin.config file

▲ Default value:

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
```

- This will use default minimum heap of 1MB
- Causes VERY slow startup of Admin Server
- Many garbage collections occur, and extend heap size many times

▲ Better value:

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-ms128m -mx128m
```

- Setting minimum heap size saves 50% of startup time on AIX



Set Application Server JVM Heap

- **Admin Server JVM heap size is set with command line arguments in the admin console.**

▲ A better value:

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-ms128m  
-mx128m
```

- Setting the -ms value the same as the -mx value works well for performance runs, but in production, your -ms value should be 1/2 to 1/4 the -mx value, based on platform.



Transport Queue

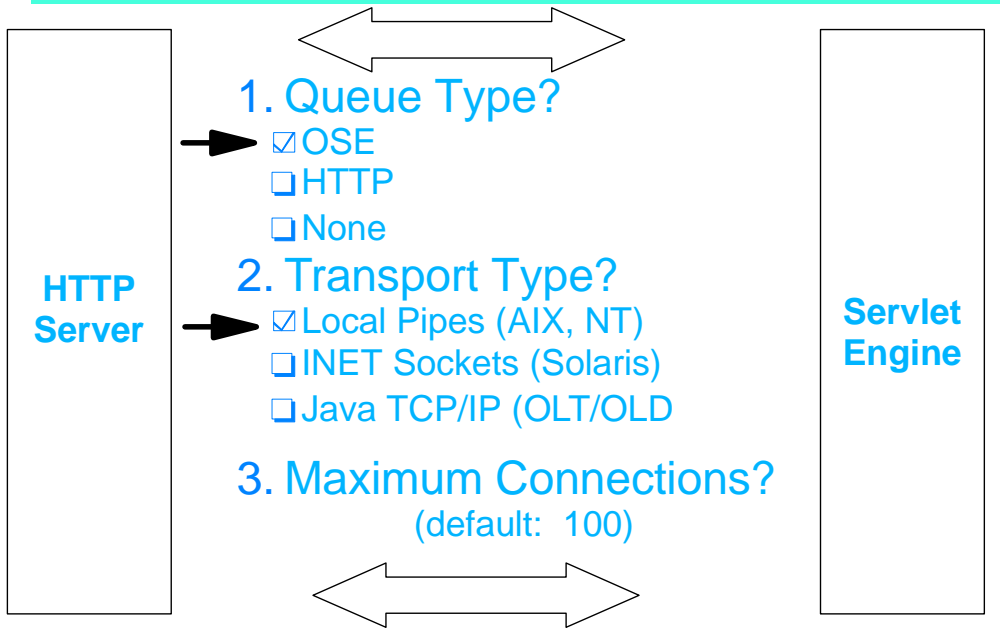


- To route servlet requests from the HTTP Server to the servlet engine, WebSphere establishes a transport queue between the HTTP server plug-in and each servlet engine.

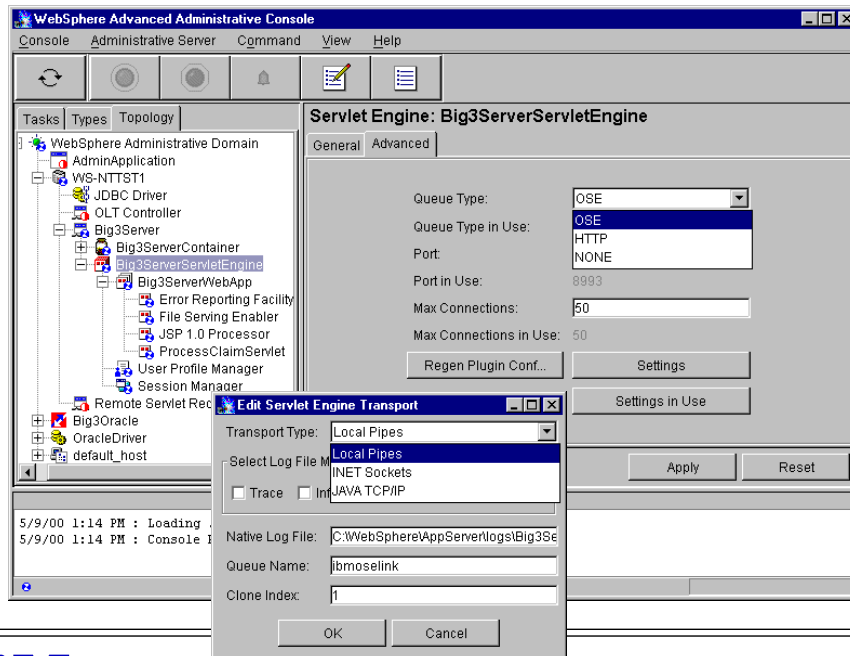
- You can adjust the transport queue to improve performance. Three parameters you should consider are:
 - ▲ Queue type
 - ▲ Transport type
 - ▲ Maximum connections



Transport Queue Tuning

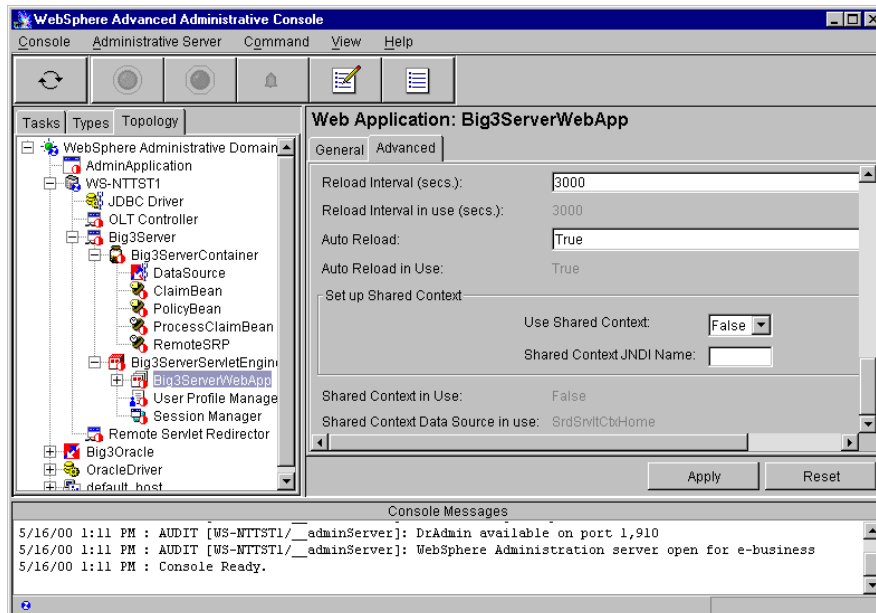


Transport Queue & Connections



- The queue type is very important. Whenever possible, you should use OSE for the queue type. OSE provides the best performance for the transport queue between the HTTP server plug-in and the WebSphere servlet engine.
- Set Servlet Engine transport queue options from Advanced properties sheet of the Servlet Engine.
- Click the Settings button to open Edit Servlet Engine Transport pop-up window.
- OSE Queue Transport Type Options
 - Local pipes performs better on AIX & Windows NT
 - INET Sockets is default on Solaris to handle load
- WebSphere threads specified in Max Connections for Servlet Engine. Optimum setting is typically slightly less than or equal to the number of threads or processes running in the HTTP server. The Max Connection setting specifies the number of connections to use for the communications channel between the HTTP server and the WebSphere engine.
- Make small (5 or fewer) changes Max Connections to find the optimal setting for your configuration.

Servlets Auto Reload



- Auto Reload of servlets simplifies testing by letting you quickly change servlets without restarting the Application Server. However, this dynamic ability to reload servlets and the associated polling has a negative effect on performance.
- Once you are in production mode, you should turn off Auto Reload by changing setting from True to False.
- Alternatively, you can adjust the Reload Interval much higher than the default.

ORB



- There are several settings that control internal ORB processing. These settings can be used to improve performance when EJBs are used.
- Normally, Remote Method Invocation (RMI) is Pass By Value.
- When both the client and the remote object are in the same JVM, for in-process calls, local copies can be disabled.



- Pass By Value causes a copy of the calling function's arguments to be used by the object being called, regardless of whether the call is in- or out-of-process.
- Disabling local copies eliminates the process cost of creating a "proxy object" and can lead to a performance savings of greater than 10% for the process.
- Disable local copies by adding the following directives to the command line arguments for your application server:
`-Djava.rmi.CORBA.UtilClass=com.ibm.CORBA.iiop.util`
`-Dcom.ibm.CORBA.iiop.noLocalCopies=true`

Using noLocalCopies



While using "noLocalCopies" can improve performance, there can also be unintended side effects. A possible unintended side effect is illustrated below:

```
void execute() {
    MyObject myObject = new MyObject();
    MyObject.setString("hello");
    remoteObject.foo(myObject);
    System.out.println(myObject.toString());
}
```

```
RemoteObject...
void foo(MyObject obj) {
    obj.setString("hello world");
}
```



- Without "noLocalCopies," the method call to foo passes a copy of MyObject. When the foo method modifies the value of the String field (via setString), it has no bearing on the original object. This results in "hello" as the output.
- With "noLocalCopies," the method call to foo passes an object reference of MyObject. When the foo method modifies the value of the String field (via setString), it has an effect on the original object. This results in the output of "hello world."

Connection Pooling



- Connection pooling establishes a pool of connections that user servlets and EJBs can use.
- Once a connection is established, it is reused repeatedly so subsequent requests incur only a fraction of the cost of a connect/disconnect.
- Connection pooling is the recommended method to use when developing new servlets and EJBs.



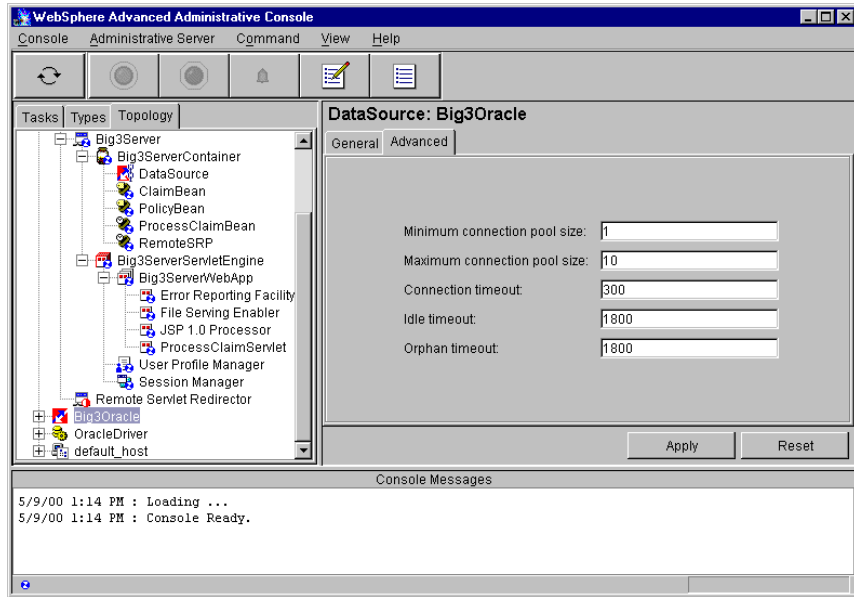
Database Connection Pooling



- Fewer connections may perform better
 - ▲ Each clone has its own connection pool.
 - ▲ Connections take up to 2M memory each.
 - ▲ Maximum database connection pool size is 5-10 times number of clones (default is 30)
- Place database on a node separate from the WebSphere server.
- Configure DB2 as "remote" for AIX & Solaris
 - ▲ Avoid shared memory limitations under load



Data Source Connection Pool



Session Management



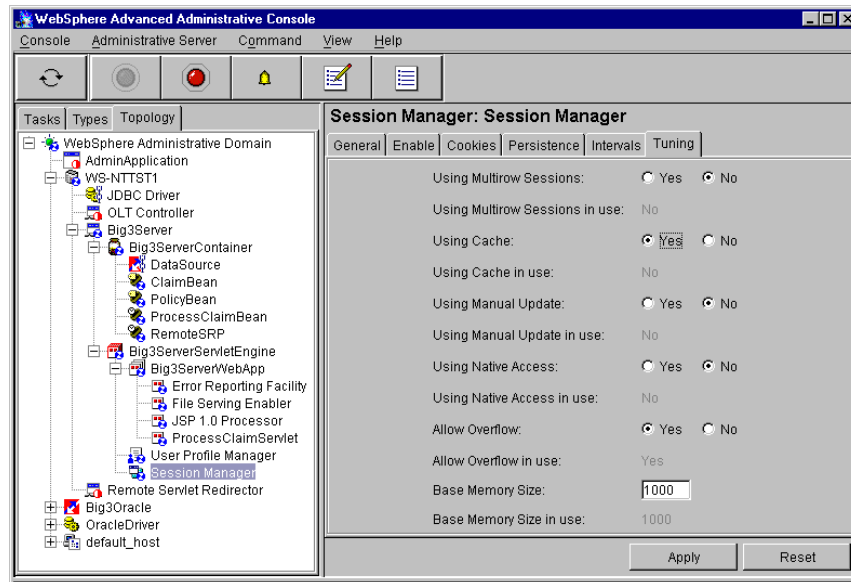
- Session to Database vs In-memory
 - ▲ Performance will degrade when persisting session data to the database
 - ▲ Session persistence required for cloning and clustering

- Session Timeout
 - ▲ Explicitly invalidate sessions when possible
 - ▲ Default session timeout 30 minutes
 - ▲ Lower this for active websites with no explicit application invalidation



- Storing sessions in a database will degrade performance compared with persisting session information to memory, but is less volatile.
- You can set the period of time a session is allowed to go unused. When that time expires, the session is considered invalid. Set this parameter from the Servlet Engine-->Session Manager.
- If possible, sessions should be closed with the Invalidate method. Since the Invalidate Time parameter affects the amount of memory used for sessions, setting this parameter correctly is extremely important.

Session Manager Tuning



- If you are storing sessions in a database, go to the Tuning tab in the Session Manager properties sheet.

For More Information...



- WebSphere Application Server Performance Whitepaper
<http://www.ibm.com/software/webservers/appserv/whitepapers.html>

- Redbooks (<http://www.redbooks.ibm.com>)
 - ▲ WebSphere V3 Performance Tuning Guide (SG24-5657)
 - ▲ San Francisco Performance Tips & Techniques (SG24-5368)

- Java Development information
 - ▲ <http://www.ibm.com/developer/java/>

- IBM Internal (<http://water.raleigh.ibm.com>)

