IBM AIX performance and tuning tips for Oracle's JD Edwards EnterpriseOne web server

Applies to JD Edwards EnterpriseOne 9.0 with tools release 8.98 or 9.1

Diane Webster
IBM Oracle International Competency Center

August 2012

Table of contents

Change history

Version	Date	Editor	Editing description
1.0	08/09/2012	Diane Webster	Original version

Executive Summary

The IBM AIX® operating system with POWER7® processor-based Power Systems™ hardware combined with WebSphere® and JD Edwards EnterpriseOne application software is a powerful technology stack that can allow you to reduce the total cost of ownership (TCO) of your ERP implementation. As Oracle continues to enhance and enrich the functionality of JD Edwards EnterpriseOne, it becomes increasingly necessary to optimize the performance of the system to maintain a good end user experience. The AIX web server with WebSphere is one component that should be tuned to ensure good performance. With proper tuning and configuration, the IBM AIX web server can provide not only a good end-user experience, but also a lower total cost of ownership.

This paper provides web server tuning recommendations based on tests performed at the IBM Oracle International Competency Center. The tests were performed with JD Edwards EnterpriseOne 9.02 with tools release 8.98 and tools release 9.1 using WebSphere Application Server 7.0 on AIX 7.1.

Oracle support for WebSphere Application Server

At the time this document is being written, Oracle supports the following editions and versions of WebSphere Application Server for use with JD Edwards EnterpriseOne:

- WebSphere Application Server Network Deployment 6.1 (IBM Technology for Java 32 bit JVM)
- WebSphere Application Server Network Deployment and Express 7.0 (IBM Technology for Java 32 bit and 64 bit JVMs)

Note that the editions and versions supported are dependent on the operating system release and the application release. For complete information, please refer to the following Oracle website: https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=705335.1

Note: This web site requires a userid and a password

The tuning guidance in this document has been reviewed for all of the above WebSphere releases and is expected to remain valid for future versions. References to WebSphere in this document which don't specify an edition or release should be understood as any edition of WebSphere supported by Oracle for use with JD Edwards EnterpriseOne.

Note that Network Deployment includes additional functionality for clustering, high availability, and load balancing. It can be used to either configure multiple instances of the application server on the same system ("vertical clustering") or multiple instances on multiple systems ("horizontal clustering"). Vertical clustering can help environments with high JVM memory requirements as it allows separate tuning and management for each JVM. Horizontal clustering is not generally used in EnterpriseOne environments. Base and Express editions do not support clustering.

Introduction

When using WebSphere and the AIX 7.1 operating system as the technology stack for the JD Edwards EnterpriseOne web-tier, you should tune and configure the environment to maximize the system resources. For the best end-user performance, the WebSphere Application Server, the HTTP server, and the JD Edwards EnterpriseOne JAS configuration should be reviewed. In most cases, tuning will be determined based on the expected number of concurrent users. During the lab testing, 1000 interactive users were used as the baseline. When applying the recommendations to your system, factor in the number of users expected for your environment.

Consider the tuning guidelines in this document as a starting point. Your environment will be different from the lab and your results may vary. For example, if you run a different mix of applications or if your users work with large reports or data extractions, additional tuning may be required to optimize the performance of your implementation. Over time, additional tuning may be required to keep current with changes your workload and changes in the application code and technology stacks.

Useful web sites and references to other documents are included throughout this paper. For additional documents, please refer to Appendix A, Additional information.

Test environment

In the tests, the web server was run on AIX 7.1 and the POWER7 processor-based Power Systems hardware with WebSphere 7.0 and JD Edwards EnterpriseOne 9.02. The configuration was a virtual three-tier environment. The backend application server and database server was an IBM Power Systems server running IBM i. A high-level diagram of the environment is shown below in Figure 1.

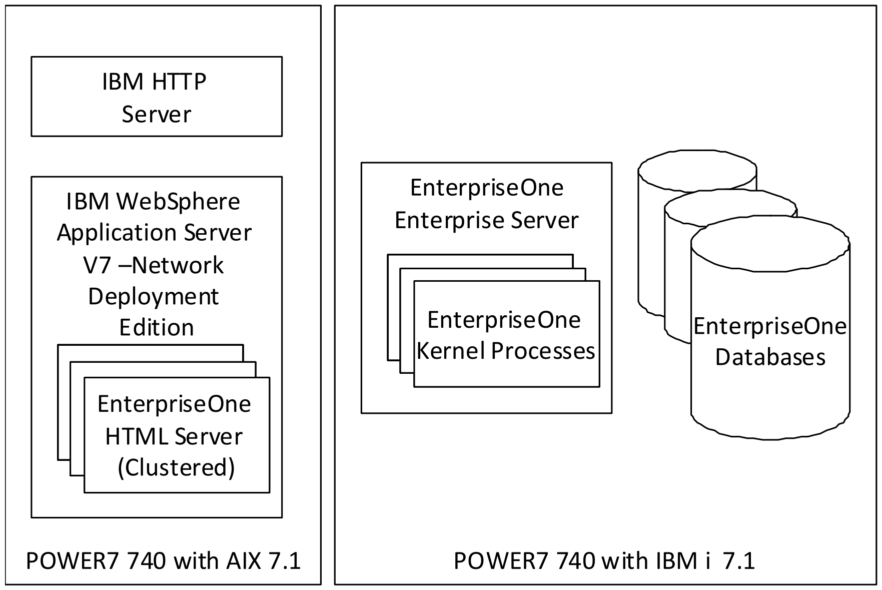

IBM HTTP Server
IBM WebSphere Application Server V7 –Network Deployment Edition
EnterpriseOne HTML Server (Clustered)
POWER7 740 with AIX 7.1
EnterpriseOne Enterprise Server
EnterpriseOne Kernel Processes
EnterpriseOne Databases
POWER7 740 with IBM i 7.1

Figure 1: Test Environment Configuration

Table 1 below shows the details of the software versions used. These were the latest fixes available at the time of the testing. Before doing any tuning, verify that your environment meets Oracle's published MTRs (minimum technical requirements) for JD Edwards EnterpriseOne. The MTR website is listed in Appendix A.

Software for test environment	Description	Service Pack:
AIX	7.1	Service Pack 4
IBM HTTP Server	7.0	
WebSphere	7.0	Fixpack 19
JD Edwards EnterpriseOne	9.02	Tools 8.98.4.7 and 9.1.0.4

Table 1: Test environment software

HTTP Server tuning parameters

All of the tuning in this section for the HTTP server involves changing the HTTP configuration file, httpd.conf, located in the /conf directory of your HTTP Server installation directory.

Configuring Simultaneous Client Connections

On the HTTP server, it is important to properly configure the number of threads available to process client requests. If the web server runs out of threads, requests will wait for a thread to become available. This can cause unnecessary and unexpected delays in web request processing. It is therefore important to review the HTTP Server configuration and tune the thread configuration appropriately.

Before we discuss the configuration, it will help to have a general understanding of how the HTTP Server handles requests on AIX. A running instance of IBM HTTP Server consists of a single-threaded parent process that creates and manages one or more multi-threaded child process. HTTP requests are served by threads running in the child process. Each simultaneous connection consumes a thread.

On AIX, there are several parameters to review to ensure sufficient threads to process client requests. The first parameter is the MaxClients directives. This controls the upper limit on the number of concurrent connections the web server can handle. ThreadsPerChild is the second important directive. This controls how many threads each child process creates. The third directive, ServerLimit, controls the number of child processes that can be created. At runtime, the upper limit of child processes is MaxClients divided by ThreadsPerChild.

Figure 2 shows the default HTTP Server configuration for the threads.

```
<IfModule worker.c>
ThreadLimit             25
ServerLimit             64
StartServers             1
MaxClients             600
MinSpareThreads         25
MaxSpareThreads         75
ThreadsPerChild         25
MaxRequestsPerChild      0
</IfModule>
```

Figure 2: Default HTTPD.Conf file

The default configuration will support up to 600 concurrent connections. Even though in this case, the ServerLimit is set to 64, the maximum number of child processes that will be created is 24 (600 max clients/25 threads per child). Based on the default configuration, MaxClients can be increased to 1600 without increasing the value for ServerLimit.

Recommendation: If the number of concurrent users is less than 600, the default http server configuration is sufficient. Depending on the httpd.conf file, a single HTTP server instance can send requests to multiple WebSphere Application Server ports or cluster members. Be sure factor that into calculations for user counts. To verify you are not running out of threads, check the error_log file located in the folder: <IBM HTTP Server root directory>logs. The error log will contain one of the following error message if you are running out of threads:

"server is within MinSpareThreads of MaxClients, consider raising the MaxClients setting"

or

"server reached MaxClients setting, consider raising the MaxClients setting".

If either of these messages occurs, you need to increase the MaxClients value. Ensure that the new ratio of maxclients/threads per child is a whole number and is less than the ServerLimit value. In some instance, you may need to change the ThreadsPerChild and/or the MaxServer parameters. This is especially true if the HTTP instance does SSL processing. There are several HTTP Server tuning guides which give more information on more advanced tuning. This additional information can be found in the technote titled "Tuning IBM HTTP Server to maximize the number of client connections to WebSphere Application Server". The reference to this document is located in Appendix A.

HTTP Server error logging

Logging is useful during the initial startup of the web client environment and for debugging problems. When the production environment is running and stable, extraneous logging can consume additional processing power. To maximize performance, reduce logging to the minimum required. The HTTP Server provides two different types of logging: error logging and access logging. The error log can be configured to capture various types of errors. By default on AIX, the HTTP Server will capture all warnings, errors, and critical conditions. The recommendation is to change this setting so that only errors and critical conditions are captured in the error log. For error logging level, change the LogLevel from "warn" to "error" to limit the amount of error logging done by the HTTP Server. This will control the number of messages logged in the error log. Remember the error level may be increased to debug for more verbose logging in the event of HTTP Server problems.

Recommendation: Set LogLevel to an appropriate level depending on the condition of your environment.

Example: Set LogLevel *error*

The access log captures all HTTP requests to the system. Depending on the number of HTTP requests processed by the system, this log can become very large. As a result, the performance recommendation is to disable all HTTP access logging. Since the access log tracks all access to the system, it can be used to determine the location of a web attack. Security considerations need to be balanced with the performance impact of access logging.

Recommendation: Set access logging to an appropriate level depending on requirements in your environment.

Example: Comment out the access logging:

```
#CustomLog logs/access.log combined
```

Enabling compression

The HTTP Server supports data compression of html documents and images delivered to browser clients. Compression of web traffic to the client decreases the size data packets sent over the network and can improve end user response time. Remote users connected to the network over a wide area network (WAN) generally see the most performance gains. Users connected on high speed local area networks (LAN) see smaller performance improvements. Compression and decompression consume slightly more CPU resources but is recommended when the network is relatively slow.

It is enabled with a DEFLATE directive in the httpd.conf file. The directive is present by default in the http server configuration file httpd.conf. However, it is commented out. You will need to un-comment the directive. A second directive, AddOutputFilterByType is also required to specify the types of content to be compressed.

Recommendation: In the httpd.conf file uncomment the line for the LoadModule deflate_module directive. Add the location directive (as shown below) to specify the types of files to be compressed. Restart the HTTP Server to enable compression.

Example:

```
LoadModule deflate_module modules/mod_deflate.so

<Location />

AddOutputFilterByType DEFLATE text/html text/plain text/css text/xml application/x-javascript

</Location>
```

WebSphere Application Server tuning parameters

Most of the settings below should be altered by accessing the WebSphere Administrative Console using a web browser.

Tuning garbage collection

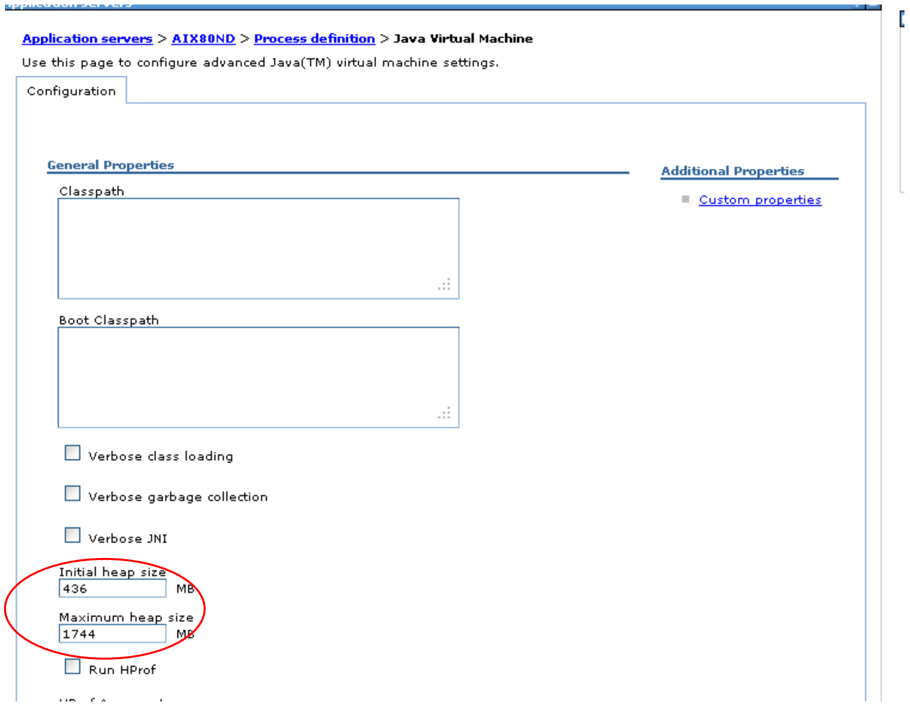
Heap size memory settings

Tuning the WebSphere heap size is an important part of managing performance of JD Edwards EnterpriseOne. A heap that is too small causes large amounts of garbage collection which use additional system resources. A heap that is too small can also result in “out of memory” errors which cause failures for end users. A heap that is too large can consume memory that is better used by other processes on the system. It can also limit the number of WebSphere instances that a server can support. WebSphere supports configuration for both the initial heap size and maximum heap size. In all tests, we set the minimum and maximum to different values. This is consistent with the WebSphere 7.0 JVM tuning guidelines provided in IBM WebSphere documentation. With this approach, the JVM can operate efficiently during steady state periods using the initial heap. However, during periods of high transaction volumes, the JVM can expand the heap up to the maximum JVM heap size. This approach eliminates some overhead that occurs during JVM expansion or contraction.

The initial heap size setting for JAS is based on the number of active users. An active user is defined as one who clicks OK or other button two to three times per minute. Lab tests show that for the best overall performance, the maximum heap size should be set at 1744 MB with a minimum heap size of 436 MB. For the 64-bit installation of WebSphere, unless there are specific issues which result in out of memory conditions, or more than 400 users are run in the JVM, use the same heap configuration as on the 32-bit version of WebSphere. The appropriate heap size may vary depending on the type of applications being run as well as the number of users.

Recommendation: Set the initial heap size to 436 MB and set the maximum heap size to 1744 MB.

To change the initial and maximum heap sizes using the WebSphere Administration Console use the path: Expand Servers->Click Server Types-> Click Application Server-> Click Desired Server Name->Expand Java and Process Management->Click Process Definition->Click Java Virtual Machine.


Application servers > AIX80ND > Process definition > Java Virtual Machine
Use this page to configure advanced Java(TM) virtual machine settings.

Initial heap size: 436 MB
Maximum heap size: 1744 MB

Figure 3: Initial and maximum heap size configuration

Tune garbage policy

IBM Technology for Java™ JVM's provides several options for managing garbage collection. One of these is called "generational concurrent" which is abbreviated as "gencon". It divides the JVM heap storage between short lived and long lived (also known as tenured) objects. This method of garbage collection helps minimize the time that the JVM pauses to wait to garbage collection. For applications that have many short lived objects, it can result in shorter garbage collection pauses while still providing good throughput.

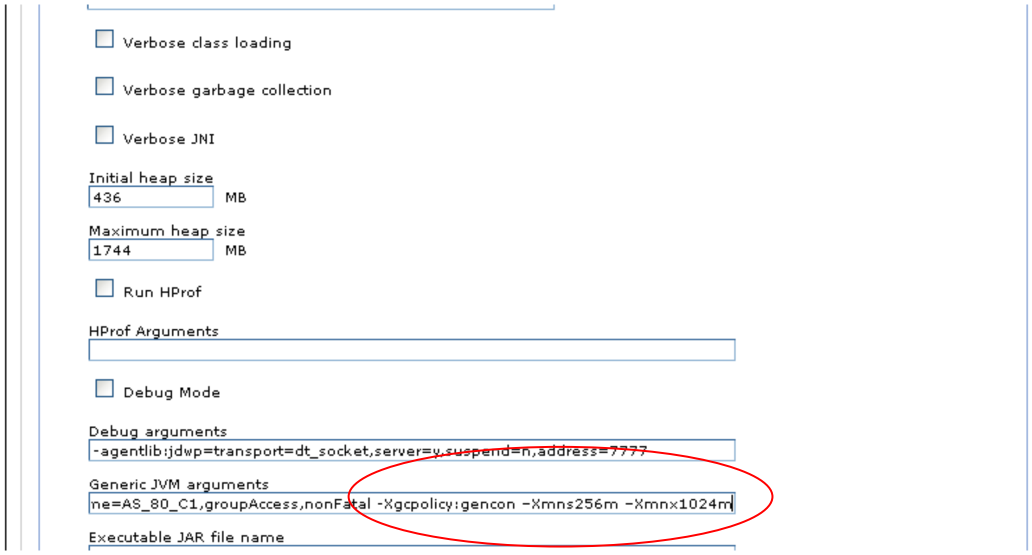
An analysis of JD Edwards EnterpriseOne JVMs during the lab tests showed that the largest number of objects reclaimed during garbage collection was the short-lived objects. Therefore, to reduce the number of times that garbage collections occurred, we configured the JVM with to allow for a large nursery.

When using maximum heap size of 1744 MB, we specified the following gencon settings in the HTML server JVM's:

-Xgcpolicy:gencon –Xmns256m –Xmnx1024m

The two parameters specify the minimum and maximum storage for the short lived portion (nursery) of the JVM heap. This allows the JVM to grow the amount of storage required for short lived objects when required.

This is configured on the same screen where the heap sizes are specified.



Verbose class loading [ ]

Verbose garbage collection [ ]

Verbose JNI [ ]

Initial heap size: 436 MB

Maximum heap size: 1744 MB

Run HProf [ ]

HProf Arguments:

Debug Mode [ ]

Debug arguments: -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777

Generic JVM arguments: ne=AS_80_C1,groupAccess,nonFatal -Xgcpolicy:gencon -Xmns256m -Xmnx1024m

Executable JAR file name

Figure 4: Specify the gencon garbage collection policy

Tuning garbage collection threads

By default the classic JVM used in WebSphere 7.0 will use multiple threads to perform garbage collection. The default for garbage collection will be the number of logical processors on the system. On systems which use simultaneous multi-theading (SMT), there can be mulitple threads or logical processors per physical processor. For example, POWER7 processor-based hardware could have 2 physical cores and 4 threads per core. There would then be 8 logical processors. By default the number of threads used for garbage collection would then be 8. A best practice is set the garbage collection threads equal to the number of physical cores. In the POWER7 example above, you would set the number of threads for garbage collection equal to 2.

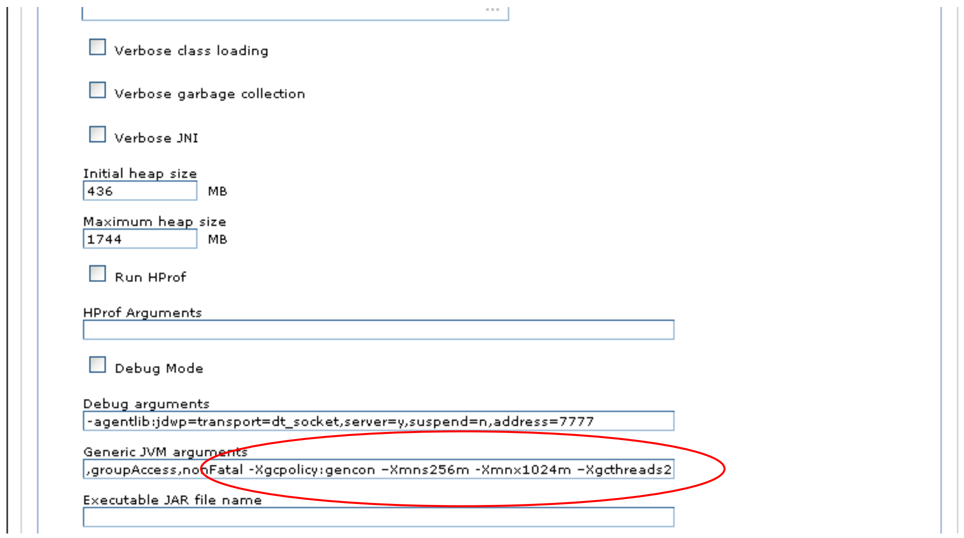
Recommendation: To set this, you will use the –Xgcthreads argument in the Generic JVM Arguments field.

Specify `–Xgcthreads` as follows:

-Xgcthreads<number of threads>

Note: Do not add a space between --`Xgcthreads` and the value for the number of processors.

Figure 5 shows the configuration for 2 cores.



- [ ] Verbose class loading
- [ ] Verbose garbage collection
- [ ] Verbose JNI

Initial heap size: 436 MB

Maximum heap size: 1744 MB

- [ ] Run HProf

HProf Arguments:

- [ ] Debug Mode

Debug arguments: -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777

Generic JVM arguments: ,groupAccess,nonFatal -Xgcpolicy:gencon -Xmns256m -Xmnx1024m -Xgcthreads2

Executable JAR file name:

Figure 5: Settings for number of threads to use for garbage collection

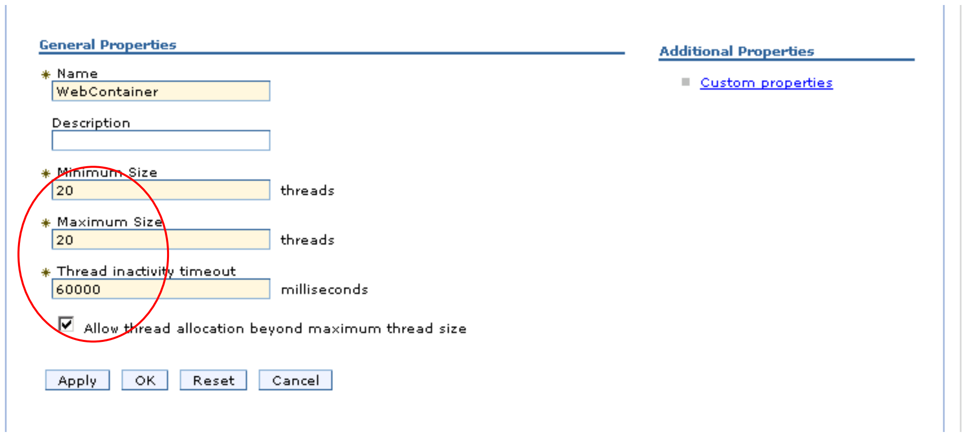
Note that multiple options can be specified in the Generic JVM arguments. For example, to follow the guidance for both the gcthread and gencon garbage collection, add following to the Generic JVM arguments field: -Xgcpolicy:gencon –Xmns256m -Xmnx1024m –Xgcthreads2

Container threads settings

WebSphere maintains a pool of threads that process requests from web clients. You can configure the minimum number of threads started and the maximum number of threads allowed in the pool. In WebSphere 7.0, new threading algorithms are used which make the threads much more efficient and fewer threads are required than in previous releases. A thread pool equal to 10-15% of the number of users supported in the JVM was sufficient in lab tests. Using a thread pool where min and max are equal or fixed results in lower overhead to manage the threads.

Recommendation: Set the minimum and maximum thread pool size equal to 10-15% of the number of active users. Enable “Allow thread allocation beyond maximum thread size”.

To navigate to the panel for thread settings, use the following path: expand Servers > click Application Servers > click the desired Server Name > click Additional Properties > click Thread Pools> click WebContainer. Check the “Allow thread allocation beyond maximum thread size” checkbox so that thread allocations are never restricted.



General Properties

* Name: WebContainer
Description:
* Minimum Size: 20 threads
* Maximum Size: 20 threads
* Thread inactivity timeout: 60000 milliseconds
[✓] Allow thread allocation beyond maximum thread size

Figure 6: Settings for Web Container thread pools.

Figure 6 shows the recommended values for 200 users. The minimum and maximum sizes are set to 10% of the total number of users. The checkbox "Allow thread allocation beyond maximum size" is also checked.

Summary

This white paper has covered some of the general guidelines for tuning the performance of JD Edwards EnterpriseOne web servers on the IBM AIX operating system when using the WebSphere platform. If used appropriately, these tuning tips can help ensure good overall web server performance. As with all tuning recommendations, adjustments may need to be made each individual user workload. For additional tuning resources for other components of a JD Edwards EnterpriseOne deployment, and for other platforms, refer to links provided in Appendix A.

Appendix A – Additional Information

Contact information

Please send questions or comments via email to IBM Oracle International Competency Center at

ibmoracle@us.ibm.com

References

General System Tuning

- Oracle Minimum Technical Requirements (MTRs) (userid and password required) https://metalink3.oracle.com/od/faces/secure/km/DocumentDisplay.jspx?id=747323.1
- IBM's HTTP Server Performance Tuning http://publib.boulder.ibm.com/httpserv/ihsdiag/ihs_performance.html#MaxClients
- Tuning IBM HTTP Server to maximize the number of client connections to WebSphere Application Server http://www-01.ibm.com/support/docview.wss?uid=swg21167658

JD Edwards EnterpriseOne

- JD Edwards EnterpriseOne documentation http://www.oracle.com/technology/documentation/jdedent.html

Additional Resources

- For additional copies of this document and for other white papers please visit http://www-03.ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs
- For additional information on integrated, collaborative enterprise solutions from IBM and Oracle's JD Edwards EnterpriseOne, please visit http://ibm.com/solutions/businesssolutions/oracle

Trademarks and special notices