Performance Tuning Tips for OneWorld HTML Client on pSeries

The reduced total cost of ownership and an increasingly internet sophisticated enterprise user base has driven the demand of OneWorld's HTML solution. This increased demand in turn calls for a tuning guide that will optimize the performance of such system. For the best performance of the HTML client, parameters in the WebSphere Application Server, the HTTP Server, and the OneWorld JAS and enterprise servers need to be tuned. Many of these configurable parameters are based on the number of users.

Consider these parameter guidelines as a starting point. Your environment will be different from our lab and your results may vary. For example, if you also have fat or TSE clients, run a different mix of batch jobs (UBEs), or have other applications running on your system, additional iterations may be required to optimize the performance of your implementation. Keep in mind that there are steady improvements and fixes in the pSeries server software and OneWorld code so, over time, the guidelines established by our ongoing testing will continue to improve as well.

For each configurable parameter, we have included:

- an explanation
- Where practical, the actual settings for 300 users (150 users running distribution applications, 75 users running manufacturing applications, and 75 users running financial applications). You should extrapolate these settings for your environment based on your number of users.
- The recommendation summarized in general terms.

Test environment for establishing recommendations

We tested with AIX 4.3.3 on the pSeries server with WebSphere 3.5.4 and OneWorld Xe SP17. The hardware configuration that we tested was Physical 3-Tier (P3T), had one pSeries server set up as the OneWorld JAS server with WebSphere, and another pSeries server set up as the OneWorld enterprise server, and yet another pSeries server was set up as the OneWorld Database server.

All pSeries servers used in this testing had the level of software installed in the table below . This combination of software offered the latest fixes available at the time of the testing and is included for your reference. Keep in mind that software does change and improve over time, and we generally recommend that customers stay current on service packs.

Test environment update levels

Software for test environment	**Description**	**Service Pack:**
Aix	4.3.3	4.3.3.75
UDB Database	7.2.4	Fixpack 4*
OneWorld	Xe	SP17
HTTP	1.3.12	
WebSphere Std. 3.5.4	3.5.4	Fixpack 4*
Java	1.2.2	

* Check the JDE Knowledge Garden for latest versions of supported software.

WebSphere Application Server

Most of these settings are accomplished by starting the WebSphere Administrative Console.

1 WebSphere Application Server instances (or JVMs)

HTML Clients use a web browser running Internet Explorer 5.5 which connects through the HTTP Server running on the pSeries server to pass requests to WebSphere running OneWorld® JAS servlets and JSPs. We followed the JAS installation instructions for creating the JAS server instance (referred to as a JVM because each instance runs in its own Java Virtual Machine). We found that each JVM provided the best performance if it supported up to 300 users. Running 300 or fewer users per JVM reduces the memory contention within WebSphere and therefore reduces the response time for the HTML clients. If you have more than 300 users, you will need to create additional instances to provide for the additional users, as needed.

300 user example: we created one JVM to accommodate our 300 users.

Recommendation: Follow the JAS installation instructions and create one or more instances to run the JAS code for up to 300 users running in each instance.

2 Heap size memory settings

You can control how much of the pSeries server's total memory is available for the JAS server by varying the heap size. The way you vary the heap size for each instance is via the command line arguments within each JVM. WebSphere allows for an initial heap setting and a maximum heap setting. We found that WebSphere on the pSeries server performed best when both the initial and maximum heap size setting were the same.

In our test, we used both 750 MB and 1000 MB heap size. We found using anything higher than 1000 MB will cause excess garbage collection times, thus affecting overall response times. A 1000 MB heap size seams to be ideal for most situations. However, 750 MB would work when a shortage of physical memory is present. Response times may suffer sue to smaller heap size. You can review your heap size by using OneWorld's "web" saw.

300 user example: we set the JVM's command line parameters to include –Xms1000m for initial and –Xmx1000m for maximum.

From your WebSphere administrative console, expand your node (Machine Hostname), double click on your JVM (JDEJAS1). Review the command line arguments by scrolling left and right within them and either change or add the initial and maximum memory setting to reflect the sizes you have chosen. Then left click Apply and watch for the console message below that says Command "JDEJAS1.ModfiyAttributes" completed successfully. You must stop and restart your JVM for these changes to take effect.

Recommendation: for the heap size in each JVM, on the command line, set an initial heap of 1000 MB memory per user and a maximum heap of 1000 MB. Total memory utilization is effectively 3.3 MB per user at this setting.

3 Garbage collection setting

Java manages its memory usage by periodically cleaning up storage that is not in use, which is referred to as garbage collection. WebSphere loads the OneWorld® Java classes and garbage collects them as needed. Because these classes are constantly used in this environment, we set the command line arguments to prevent the classes from being garbage collected by using –noclassgc.

Review the command line arguments by scrolling left and right within them and add the -noclassgc argument, if it is not there already. Then left click Apply and watch for the console message below that says Command “JDEJAS1.ModfiyAttributes” completed successfully. You must then stop and restart your JVM for this change to take effect.

Recommendation: use the –noclassgc setting in the WebSphere instance command line arguments.

4 JAS servlet connections

Within the instance, we configured the servlet running the OneWorld® JAS code to control how many concurrent servlet request that a JVM can accept. Because we allow 300 max client in the HTTP Server, we set the maximum number of connections for the servlet to be 300. The maximum should be equal to or less than the number of HTTP Server child process. This setting should not exceed 300 per instance because each user uses a thread and we only want to allow 300 threads per instance. Because more of a user’s time is spent on data input than running the servlet, changing this setting did not significantly affect performance.

300 user example: we set the maximum number of connections for the servlet to 300.

From your WebSphere administrative console, expand your node (Machine Hostname), expand your JVM (JDEJAS1), left click the JDEJASServlet, then left click on the Advanced tab, and change the max connections. Then left click Apply and watch for the console message below that says Command “JDEJASServlet.ModifyAttributes” completed successfully. You must then stop and restart your JVM for this change to take effect.

Recommendation: set the number of OneWorld® servlet connections to equal the number of users connecting to the WebSphere instance, with a maximum of 300.

5 Auto Reload

Auto reload on Web Applications should be set to false unless required. If it is required set the interval as high as possible. the auto reload is available under the servlets advanced properties or default.

HTTP Server

All of the tuning in this section for the HTTP Server involves changing the HTTP configuration file. The following shows how to access the HTTP configuration file using VI Editor.

On the pSeries server command line, type in and hit Enter:

```
cd /usr/HTTPServer/conf
```

and then open the “httpd.conf” file with the vi editor:

```
vi httpd.conf
```

At this point you now have the “httpd.conf” file open and ready to edit.

When finished with the httpd.conf file type the following to save your changes:

```
:wq
```

If you wish to exit without saving your changes type:

```
:q!
```

6 Server-Pool size regulation

Rather than making you guess how many server processes you need, Apache dynamically adapts to the load it see. That is, it tries to maintain enough server processes to handle the current load, plus a few spare servers to handle transient load spikes (e.g., multiple simultaneous request from a single Internet Explorer browser).

It does this by periodically checking how many servers are waiting for a request. If there are fewer than MinSpareServers, it creates a new spare. If there are more than MaxSpareServers, some of the spares die off.

300 user example:

```
MinSpareServers 5
MaxSpareServers 50
```

Recommendation: The default settings shown above should be sufficient for 300 users.

7 MaxClients

Limit on total number of servers running, i.e., limit on the number of clients who can simultaneously connect. If this limit is ever reached, clients will be locked out, so it should not be set too low. It is intended mainly as a brake to keep a runaway server from taking AIX with it as it spirals down.

```
300 user example:

MaxClients 500
```

Recommendation: Set this number equal to the total number of HTML clients connecting to the system. You must then stop and restart your HTTP Server for this change to take effect.

OneWorld® Xe JAS server

All of the tuning in this section for the OneWorld® Xe JAS Server involves changing the JAS.INI file. The following shows how to access the JSA.INI file using VI Editor.

On the pSeries server command line, type in and hit Enter:

```
cd /u01/jdedwardsoneworld/b7333/ini
```

and then open the “JAS.INI” file with the vi editor:

```
vi JAS.INI
```

At this point you now have the “httpd.conf” file open and ready to edit.

When you are finished editing the file and wish to close and save the file type:

```
:wq
```

8 JDBC Connection pool settings in the jas.ini

JDBC is the database interface used between the OneWorld® JAS code running on WebSphere and the database enterprise server. We ran with various numbers of initial, minimum, maximum, and pool growth settings. These settings affect the number of QZDASOINIT and/or QSQSRVR jobs that get used. Connection pooling allows HTML clients to use fewer of these jobs than fat and TSE clients because they can be reused. While running with 300 users, we found that, having a base amount for minimum number connections and small growth rate up to a maximum equal to the number of users, delivered the best performance. Initial connections can be set to one for every 10 users.

300 user example:

```
[CONNECTION POOL]
MaxConnection=300
MinConnection=5
PoolGrowth=5
InitialConnection=30  (300 users/10 connections per user=30)
```

Recommendation: if the JDE userids are all mapped to a single pSeries server system user profile (proxy userid), set initial connections to 1 for every 10 users, minimum connections to 5, pool growth to 5, and maximum connections equal to the number of users, up to a maximum of 400 per instance.

various user count example:

```
[CONNECTION POOL]
MaxConnection=2
MinConnection=1
PoolGrowth=1
InitialConnection=1
```

Recommendation: if the JDE userids are all mapped to separate pSeries server user profiles, set initial connections to 1, minimum connections to 1, pool growth to 1, and maximum connections to 2. (This method for connection pooling may be eliminated for future releases of OneWorld.)

9 JDENET connections in the jas.ini

The connections between the OneWorld® JAS server and the enterprise server are JDENET connections, the same as any fat client or TSE user. The number of JDENET connections should be the same as the number of users that are being run in the JVM (maximum of 300). This is controlled via a setting in the jas.ini file. This setting did not significantly improve our performance.

```
300 user example:

[JDENET]
maxPoolSize=300
```

Recommendation: set the same number of JDENET connections, maxPoolSize, as there are users running in the JVM (maximum of 300).