THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS. THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE PRODUCTS IN THIS MANUAL ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR STA GROUP REPRESENTATIVE FOR A COPY.

The implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB’s public domain version of the UNIX operating system. © 1981 Regents of the University of California. All rights reserved.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies are considered uncontrolled copies and the original online version should be referred to for latest version.

© 2018 STA Group LLC. All rights reserved.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>3</td>
</tr>
<tr>
<td>Preface</td>
<td>5</td>
</tr>
<tr>
<td>Audience</td>
<td>5</td>
</tr>
<tr>
<td>Related Documentation</td>
<td>5</td>
</tr>
<tr>
<td>Obtaining Documentation and Submitting a Service Request</td>
<td>5</td>
</tr>
<tr>
<td>Document Change History</td>
<td>6</td>
</tr>
<tr>
<td>Optimizing TWA Performance</td>
<td>7</td>
</tr>
<tr>
<td>TWA Hardware Requirements</td>
<td>7</td>
</tr>
<tr>
<td>Configuration Definition</td>
<td>7</td>
</tr>
<tr>
<td>Master Configuration</td>
<td>8</td>
</tr>
<tr>
<td>Client Manager Configuration</td>
<td>8</td>
</tr>
<tr>
<td>Master and Client Manager DB Configuration</td>
<td>8</td>
</tr>
<tr>
<td>Java Client Configuration</td>
<td>8</td>
</tr>
<tr>
<td>Adapter Configuration</td>
<td>8</td>
</tr>
<tr>
<td>Transporter Configuration</td>
<td>9</td>
</tr>
<tr>
<td>Performance Matrix</td>
<td>9</td>
</tr>
<tr>
<td>Monitoring TWA</td>
<td>11</td>
</tr>
<tr>
<td>Overview</td>
<td>11</td>
</tr>
<tr>
<td>Monitoring with JConsole</td>
<td>11</td>
</tr>
<tr>
<td>Connecting JConsole</td>
<td>11</td>
</tr>
<tr>
<td>Viewing a System Overview</td>
<td>13</td>
</tr>
<tr>
<td>Viewing Memory Usage</td>
<td>14</td>
</tr>
<tr>
<td>Viewing Active Threads</td>
<td>15</td>
</tr>
<tr>
<td>Viewing a VM Summary</td>
<td>16</td>
</tr>
<tr>
<td>Monitoring Scheduling Activity</td>
<td>17</td>
</tr>
<tr>
<td>Viewing MBeans</td>
<td>17</td>
</tr>
<tr>
<td>Viewing Connected Users</td>
<td>18</td>
</tr>
<tr>
<td>Viewing Job Activity</td>
<td>19</td>
</tr>
<tr>
<td>Viewing Master Status</td>
<td>20</td>
</tr>
<tr>
<td>Viewing Queue Activity</td>
<td>21</td>
</tr>
<tr>
<td>Viewing Event Activity</td>
<td>22</td>
</tr>
<tr>
<td>Monitoring the Overall Message System</td>
<td>23</td>
</tr>
<tr>
<td>Monitoring a Message Queue</td>
<td>24</td>
</tr>
<tr>
<td>Monitoring a Message Thread</td>
<td>25</td>
</tr>
<tr>
<td>Monitoring a Message Thread Pool</td>
<td>26</td>
</tr>
<tr>
<td>Monitoring Schedule Compiling</td>
<td>27</td>
</tr>
<tr>
<td>Master Status-Compile Status</td>
<td>27</td>
</tr>
<tr>
<td>Monitoring the Queue Manager Compiler</td>
<td>28</td>
</tr>
<tr>
<td>Monitoring the Message Queue Compiler</td>
<td>29</td>
</tr>
</tbody>
</table>
Monitoring Adapter/Agent Connections .............................................................. 30
  Viewing All Connections and Statuses .......................................................... 30
  Monitoring Adapter Connections via JConsole ............................................. 31
Monitoring the TWA Java Application Performance ........................................ 32
  Monitoring the Cache Sync ........................................................................ 32
  Viewing the Cache Sync Logging ................................................................. 33
  Viewing the Client Manager Output Log ....................................................... 34

Tuning TWA ................................................................................................ 35
  Overview ........................................................................................................ 35
  Configuration and Tuning ............................................................................ 35
    Memory ........................................................................................................ 35
    CPU ............................................................................................................ 35
    JMS ............................................................................................................. 35
    Master Messaging ....................................................................................... 38
    WebClient Performance Parameters ......................................................... 38
  Size-Based Guidelines for Tuning Parameters ............................................. 39
    Small Configuration .................................................................................... 39
    Medium Configuration .............................................................................. 41
    Large Configuration .................................................................................. 42

Transporter Performance ................................................................................ 45
  General Best Practices ................................................................................. 45
  Transporter Job Read Options ..................................................................... 45
Preface

The purpose of this document is to help you troubleshoot performance issues with Tidal Workload Automation.

Audience

This guide is for administrators who install, configure, monitor, and tune the Tidal Workload Automation for better performance.

Related Documentation

For a list of all Tidal Workload Automation guides, see the Tidal Workload Automation Documentation Overview of your release on tidalautomation.com at:

http://docs.tidalautomation.com/

Note: We sometimes update the documentation after original publication. Therefore, you should also review the documentation on tidalautomation.com for any updates.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see What’s New in Tidal Product Documentation at:

https://docs.tidalautomation.com/rss

Subscribe to What's New in Tidal Product Documentation, which lists all new and revised Tidal technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.
Document Change History

The table below provides the revision history for the Performance and Tuning Guide..

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Issue Date</th>
<th>Reason for Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.1 (SP2)</td>
<td>May 2015</td>
<td>- Added Monitoring TWA Java Application section. Enlarged screenshots to make them readable. General editing.</td>
</tr>
<tr>
<td>6.2.1 (SP3)</td>
<td>March 2016</td>
<td>- Restructured document into functional chapters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the JVMARGS hyphens that incorrectly showed as en-dashes.</td>
</tr>
<tr>
<td>6.3</td>
<td>August 2016</td>
<td>- Rebranded &quot;Cisco Tidal Enterprise Scheduler&quot; to &quot;Cisco Workload Automation (CWA)&quot;.</td>
</tr>
<tr>
<td>6.3.1</td>
<td>May 2017</td>
<td>- Added new configuration parameters for Large, Medium, and Small environments.</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Oct 2017</td>
<td>- Enhanced web scrolling performance.</td>
</tr>
<tr>
<td>6.3.3</td>
<td>January 2018</td>
<td>- Rebranded Cisco Workload Automation (CWA)&quot; to Tidal Workload Automation (TWA)&quot;.</td>
</tr>
</tbody>
</table>
1 Optimizing TWA Performance

Optimization and tuning is an exact science which is why it is critical to be able to recognize which parts of the system are being stressed (monitoring) and then knowing what parameters should be adjusted to reduce that stress (tuning). This chapter describes:

- TWA Hardware Requirements
- Performance Matrix

Monitoring tasks are described in the Monitoring TWA, page 11 chapter.

Tuning tasks are described in the Tuning TWA, page 35 chapter.

See Transporter Performance, page 45 for the best practices and configuration options for the Transporter.

TWA Hardware Requirements

Hardware requirements are provided below for small, medium, and large systems as defined in Configuration Definition for these TWA components:

- Master Configuration, page 8
- Client Manager Configuration, page 8
- Master and Client Manager DB Configuration, page 8
- Java Client Configuration, page 8
- Adapter Configuration, page 8
- Transporter Configuration, page 9

Note: All memory, CPU, and disk requirements in this section are for TWA-related components only, and do not take into account any additional OS/application requirements.

Configuration Definition

When referring to configuration sizes in the following sections, we use these definitions:

<table>
<thead>
<tr>
<th>Configuration Size</th>
<th>Jobs Definition</th>
<th>DB Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Configuration</td>
<td>1 - 3000</td>
<td>&lt; 4 GB</td>
</tr>
<tr>
<td>Medium Configuration</td>
<td>3,000 - 20,000</td>
<td>&lt; 16 GB</td>
</tr>
<tr>
<td>Large Configuration</td>
<td>20,000 and less than 100 K</td>
<td>&gt; 32GB</td>
</tr>
</tbody>
</table>

Also see Size-Based Guidelines for Tuning Parameters, page 39.
Master Configuration
Required for the TWA Master only (see Note in TWA Hardware Requirements, page 7):

<table>
<thead>
<tr>
<th>Master Configuration</th>
<th>Memory</th>
<th>CPU Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>8GB</td>
<td>4</td>
</tr>
<tr>
<td>Medium</td>
<td>16GB</td>
<td>8</td>
</tr>
<tr>
<td>Large</td>
<td>24GB</td>
<td>16</td>
</tr>
</tbody>
</table>

Client Manager Configuration
Required for the TWA Client Manager only (see Note in TWA Hardware Requirements, page 7):

<table>
<thead>
<tr>
<th>Client Manager Configuration</th>
<th>Memory</th>
<th>CPU Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>12 GB</td>
<td>8</td>
</tr>
<tr>
<td>Medium</td>
<td>24 GB</td>
<td>16</td>
</tr>
<tr>
<td>Large</td>
<td>32 GB</td>
<td>24</td>
</tr>
</tbody>
</table>

Master and Client Manager DB Configuration
Minimum sizes required by the TWA Master and Client Manager databases:
- MS SQL Server: 128 MB Data, 32 MB Log
- Oracle: 400 MB Data, 300 MB Index, 200 MB Temp
As the number of jobs or logs increases, DBA should tune the DB accordingly.

Java Client Configuration
Required for the TWA Java Client only (see Note in TWA Hardware Requirements, page 7):

<table>
<thead>
<tr>
<th>Java Client</th>
<th>Memory</th>
<th>CPU Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>4 GB</td>
<td>4</td>
</tr>
<tr>
<td>Medium</td>
<td>8 GB</td>
<td>4</td>
</tr>
<tr>
<td>Large</td>
<td>16 GB</td>
<td>8</td>
</tr>
</tbody>
</table>

Adapter Configuration
For each TWA adapter instance, allocate the following extra RAM:
Transporter Configuration

Required for the TWA Transporter only (see Note in TWA Hardware Requirements, page 7):

<table>
<thead>
<tr>
<th>Adapter</th>
<th>Memory*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1 GB</td>
</tr>
<tr>
<td>Medium</td>
<td>1 GB</td>
</tr>
<tr>
<td>Large</td>
<td>2 GB</td>
</tr>
</tbody>
</table>

Performance Matrix

The tables below help you link common catalysts that impact performance with the areas that need to be monitored/tuned to get better performance.

**Table 1**

<table>
<thead>
<tr>
<th>Catalyst</th>
<th>CPU</th>
<th>Memory</th>
<th>JMS</th>
<th>Cache Read</th>
<th>Cache Write</th>
<th>Cache Tuning</th>
<th>Cache Sync</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schedule Activity</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schedule Compile</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache Sync</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For example, in the table above, if you are experiencing performance problems with large schedules, the items you should tune or monitor would be the CPU, the memory, your JMS, and the Cache Write.

**Table 2**

<table>
<thead>
<tr>
<th>Catalyst</th>
<th>CPU</th>
<th>Memory</th>
<th>JMS</th>
<th>DB Connections</th>
<th>Message Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Managers</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Adapter Connections</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Agents</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Schedule Activity</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Schedule Compile</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
**Table 2**

<table>
<thead>
<tr>
<th>Catalyst</th>
<th>CPU</th>
<th>Memory</th>
<th>JMS</th>
<th>DB Connections</th>
<th>Message Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitions</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logs and History</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cache Sync</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Monitoring TWA

Overview

This chapter covers the tasks you can use to monitor TWA performance:

- Monitoring with JConsole, page 11
- Monitoring Scheduling Activity, page 17
- Monitoring Schedule Compiling, page 27
- Monitoring Adapter/Agent Connections, page 30
- Monitoring the TWA Java Application Performance, page 32
- Monitoring the Cache Sync, page 32

Monitoring with JConsole

JConsole is a diagnostic tool that comes with the standard JDK. It allows you to connect directly to a running JVM and monitor many performance metrics, including memory/CPU usage. JConsole can also be used to access diagnostic modules exposed by each individual application called MBeans. Both the Master and Client Manager expose MBeans.

Connecting JConsole

Before connecting JConsole to either the Client Manager or Master, make sure the following property is set in clientmgr.props or master.props.

JmxOn=Y

If you are running JConsole on the same machine as the JVM you are connecting to, the JVM will be listed in JConsole.
If you are connecting JConsole to a JVM running on a remote machine, type in the remote JVM's machine host name and port (the default Client Manager port is 1100).

**Note:** You can change the default port for the JVM by setting the following property in clientmgr.props or master.props.  
*JmxRmiPort*=1200

**Note:** To avoid random port generation, add *JmxRemotePort*=1098 along with existing *JmxRmiPort* in master.props or clientmgr.props. For JConsole, ensure that you use the *JmxRmiPort*.  

![Remote Process:](image-url)
Viewing a System Overview

The Overview tab of JConsole provides an overview of JVM’s memory, threads, and CPU usage.

To view a system overview of the vital stats for the JVM, select the Overview tab of the console.

Figure 1  Viewing Vital Stats for JVM
Viewing Memory Usage

To view memory usage, select the Memory tab of the console.

The memory tab provides more detailed information about JVM’s memory use, allowing you to determine if the JVM has sufficient memory for the application that is running.

Note: For a normal running JVM, you should see memory use increase and decrease in the short term. However, if you see that memory use is increasing in the long term, it may indicate a memory that will eventually result in an out-of-memory termination.
Viewing Active Threads

To view active threads and potential deadlocks, select the **Threads** tab of the console.

This screen provides stack traces for every thread in the JVM. A stack trace shows exactly what a thread is doing at the time of the trace. This screen allows you to automatically detect thread deadlocks.

**Figure 3**  Viewing active threads
Viewing a VM Summary

For viewing a virtual machine summary and for basic technical support information, select the VM Summary tab of the console.
Monitoring Scheduling Activity

Viewing MBeans

For viewing custom monitoring modules (MBeans), select the MBeans tab of the console.

Figure 5 Viewing MBeans
Viewing Connected Users

For viewing connected users, click the **Connections** tab on the Master Status pane.

**Figure 6**  
Viewing connected users
Viewing Job Activity

For viewing current job activity, select **Operations > Job Activity** from the Navigation tree to view the Job Activity pane.

**Figure 7**  Viewing Job Activity
Viewing Master Status

For viewing an overview of the Master status, select **Operations > Master Status** from the Navigation tree to view the Master Status pane, and then click **Overview**.

**Figure 8  Viewing the Master Status**
Viewing Queue Activity

For viewing queue activity, choose Operations > Master Status from the Navigation tree to view the Master Status pane, and then click Queue.

Figure 9  Viewing Queue Activity
Viewing Event Activity

To view event activity, choose Operations > Event Activity from the Navigation tree to view the Event Activity pane.

Figure 10  Viewing Event Activity
Monitoring the Overall Message System

For monitoring the overall message system, click the MBeans tab on the Java console, and then select Attributes from the tree to view the attribute values.

Figure 11  Monitoring the overall message system
Monitoring a Message Queue

For monitoring a message queue, click the MBeans tab on the Java console, and then choose `MessageQueue > Attributes` from the tree to view the attribute values associated with the message queues.

**Figure 12  Monitoring a message queue**
Monitoring a Message Thread

For monitoring a message thread, click the MBeans tab on the Java console, and then choose MessageThread > threadname > Attributes from the tree to view the attribute values associated with the message thread.

Figure 13  Monitoring a Message Thread
Monitoring a Message Thread Pool

For monitoring a message thread pool, click the MBeans tab on the Java console, and then choose MessageThreadPool > poolname > Attributes from the tree to view the attribute values associated with the message thread pool.

Figure 14 Monitoring a Message Thread Pool
Monitoring Schedule Compiling

Master Status-Compile Status

For monitoring the Master compile status, choose **Operations > Master Status** from the Navigation tree to view the Master Status pane, and then click **Overview**. You can view the compile percentage in the Poll Activity pane as displayed below.

**Figure 15** Viewing the Master compile status
Monitoring the Queue Manager Compiler

For monitoring the queue manager compiler, select the MBeans tab on the Java console, and then select **CompilerQueueManager > Attributes** from the tree to view the attribute values associated with the queue manager compiler.

**Figure 16  Monitoring the queue manager compiler**
Monitoring the Message Queue Compiler

For monitoring the queue message compiler, select the MBeans tab on the Java console, and then choose **MessageQueue > Attributes** from the tree to view the attribute values associated with the queue message compiler.

**Figure 17** Monitoring the message queue compiler
Monitoring Adapter/Agent Connections

Viewing All Connections and Statuses

To view all connections and their statuses, choose Administration > Connections from the Navigation tree to view the Connections pane.

Figure 18 Viewing All Connections and their statuses
Monitoring Adapter Connections via JConsole

For monitoring the adapter connections, select the **MBeans** tab on the Java console, and then choose **Node > Service > Attributes** from the tree to view the attribute values associated with the adapter connection.

**Figure 19  Viewing adapter connections via JConsole**
Monitoring the TWA Java Application Performance

JConsole is recommended for capturing/monitoring any performance related issue with the Java Client application. JConsole, a GUI-based software, does not provided a way to capture performance data in a non-interactive way. To capture JMX MBean Attribute values as displayed below, for the purposes of generating alarms when the software exceeds thresholds, use the following command-line alternatives to the JConsole.

- Jmxterm (http://wiki.cyclopsgroup.org/jmxterm)
- cmdline-jmxclient (http://crawler.archive.org/cmdline-jmxclient)

Jmxterm retrieves MBean Attribute values in an interactive shell, whereas cmdline-jmxclient retrieves it non-interactively. The syntax of cmdline-jmxclient is as follows:

```
Z:\>java -jar cmdline-jmxclient-0.10.3.jar
Usage: java -jar cmdline-jmxclient.jar USER:PASS HOST:PORT [BEAN] [COMMAND]
Options:
USER:PASS Username and password. Required. If none, pass '-'.
E.g. 'controlRole:secret'
HOST:PORT Hostname and port to connect to. Required. E.g. localhost:8081.
```

List registered beans if only USER:PASS and this argument.

BEANNAME Optional target bean name. If present we list available operations and attributes.

COMMAND Optional operation to run or attribute to fetch. If none supplied, all operations and attributes are listed. Attributes begin with a capital letter: e.g. ‘Status’ or ‘Started’. Operations do not. Operations can take arguments by adding an ‘=’ followed by comma-delimited params. Pass multiple attributes/operations to run more than one per invocation.

Monitoring the Cache Sync

To monitor the Cache sync, open DSP.props and set the cache logging level to FINE as displayed below.

**Figure 20 Monitoring the Cache sync**

![Figure 20](image)
Viewing the Cache Sync Logging

To view the Cache sync logging, open the log file located in the Log folder.

Figure 21  Viewing the Cache sync logging
Viewing the Client Manager Output Log

To view the Client Manager output, open the `clientmgr.out` log file located in the Log folder.

**Figure 22** Viewing the Client Manager Output

```
08/27 01:26:43:403] Client Manager: version 6.3.0.146
08/27 01:26:43:403] Java version: 1.8.0_91
08/27 01:26:43:403] Java Virtual Machine version: 25.91-b15
08/27 01:26:43:403] Start time: 08/27/16 01:26:43:403
08/27 01:26:43:403] [-----------------------------------------------]
08/27 01:26:43:403] Maximum number of log files = 50
08/27 01:26:43:408] Added a log file called 'RegularFile'.
08/27 01:26:43:554] Retrieved a log file called 'RegularFile'
08/27 01:26:43:461] 2016-08-27 01:26:45,461: INFO: JMX connector: Logging initialized @5061ms
08/27 01:26:52:948] -- Loading CSS plugins: /sites-6.0.0.0.jar
08/27 01:26:13:784] Maximum number of log files = 50
08/27 01:26:33:785] Added a log file called 'tes-6.0'
08/27 01:27:04:467] ClientNode Backup Server = [tcp://]
%20Files/TIDAL/ClientManager/webapps/Client AVAILABLE/webapps\Client]
08/27 01:30:54:624] Client initialized.
08/27 01:31:23:313] Retrieved a log file called 'RegularFile'
08/27 01:31:35:639] Retrieved a log file called 'tes-6.0'
08/27 01:24:26:957] Retrieved a log file called 'tes-6.0'
08/27 01:25:03:81] Retrieved a log file called 'tes-6.0'
08/27 01:36:02:569] Primary objects synchronized in 63 seconds.
08/27 01:26:08:229] Secondary objects synchronized in 5 seconds.
```
3

Tuning TWA

Overview

This chapter covers the tasks you can use to tune TWA performance:

- Configuration and Tuning, page 35
- Size-Based Guidelines for Tuning Parameters, page 39

Configuration and Tuning

This section will walk you through how to tune the application (either Master or Client Manager) to get better performance.

Memory

These parameters tune the amount of memory the application has to use to get work done. If an application does not have enough memory to work with, it could have very poor performance or in the worst case get out of memory errors and fail.

The following parameters apply to all Java applications:

- -Xmn = size of young generation (1/4 size of heap)
- -Xmx = max heap size
- -Xms = initial heap size (guarantees JVM has that much memory)
- -Xss = thread stack size (increase if getting stack overflow exceptions)
- JVMARGS= -Xmn1024m -Xms28672m -Xmx28672m

CPU

The system CPU is not readily tunable from our application or the JVM itself, however, know that the CPU is an important resource that directly impacts performance. If a system does not have enough CPUs or CPU power, applications can run very slowly across the board. Upgrade your system to more CPUs or faster CPUs if the system monitor consistently shows the CPU meter very high.

JMS

Both the Master and Client Manager rely on the Java Message Service (JMS) for internal and cross communication with each other. You can think of JMS as the communication link between the Master and Client Manager. That is why JMS can greatly impact performance, especially if there is a lot of data moving back and forth between the Master and Client Manager, such as the primary and secondary cache synchronization.
JMS sessions process all messages such as during the use of one worker thread per session. The following threads are affected:

- **MinSessionPoolSize** - minimum number of ActiveMQ sessions kept pooled. Having sessions available to handle requests reduces the cost of allocating sessions on demand.
- **MaxSessionPoolSize** - maximum number of ActiveMQ sessions kept pooled.
  
  Note: If more sessions are needed to process messages, the system will still allocate them. This setting only limits the number of sessions allowed to be pooled.
- **MaxConcurrentMessage** - maximum number of ActiveMQ messages allowed to be processed concurrently. This setting is important for increasing throughput and utilizing all the cores on a system when there are many messages waiting in the ActiveMQ queues.

The following Message Brokers handle all JMS traffic:

- **MessageBroker.MemoryLimit** - how much memory (in MB) to allocate to ActiveMQ for storing in-flight messages. If queues become full, ActiveMQ will page messages to disk, which is more expensive than keeping them in memory.
- **MessageBroker.TempLimit** - how much memory (in MB) to allocate to ActiveMQ for storing temporary messages. If queues become full, ActiveMQ will page messages to disk, which is more expensive than keeping them in memory.
- **MessageBroker.StoreLimit** - how much disk space (in MB) to allocate to ActiveMQ for storing in-flight messages when memory is full.

The following *master.props* properties are what you would tune for DSP to Master message traffic for all DSP connections:

- **MinSessionPoolSize** (5)
- **MaxSessionPoolSize** (10)
- **MaxConcurrentMessages** (10)

The following *master.props* properties are what you would tune for DSP to Master message traffic per DSP connections:

- **ClientConnection.MinSessionPoolSize** (2)
- **ClientConnection.MaxSessionPoolSize** (5)
- **ClientConnection.MaxConcurrentMessages** (5)

The following *master.props* properties are what you would tune for Remote Master to Master message traffic per Remote Master:

- **RemoteMasterClient.MinSessionPoolSize** (2)
- **RemoteMasterClient.MaxSessionPoolSize** (5)
- **RemoteMasterClient.MaxConcurrentMessages** (5)

The following *master.props* properties are what you would tune for Master to Remote Master message traffic per Remote Master:

- **RemoteMasterServer.MinSessionPoolSize** (2)
- **RemoteMasterServer.MaxSessionPoolSize** (5)
- **RemoteMasterServer.MaxConcurrentMessages** (5)

The following *tes-6.0.dsp* properties are what you would tune for Master to DSP message traffic:
- **ClientNode.MinSessionPoolSize (5)**
- **ClientNode.MaxSessionPoolSize (10)**
- **ClientNode.MaxConcurrentMessages (10)**

The following `tes-6.2.dsp` properties are what you would tune for Fault Monitor to DSP message traffic:

- **FTNode.MinSessionPoolSize (2)**
- **FTNode.MaxSessionPoolSize (5)**
- **FTNode.MaxConcurrentMessages (5)**

The following `tes-6.2.dsp` properties are what you would tune for Cache Read connections shared among all threads:

- **DataCache.ReadConnectionsMin (2)** – minimum number of JDBC connections kept pooled to read from the data cache. Having connections always available to handle read requests reduces the cost of allocating connections on demand.
- **DataCache.ReadConnectionsMax (4)** – maximum number of JDBC connections allowed to be allocated concurrently to read from the data cache. If this number is exceeded, read requests will be queued and blocked. Recommend increasing to handle more concurrent users.

The following `tes-6.2.dsp` properties are what you would tune for Cache Write connections shared among all threads:

- **DataCache.WriteConnectionsMin (4)** – minimum number of JDBC connections kept pooled to write to the data cache. Having connections always available to handle write requests reduces the cost of allocating connections on demand.
- **DataCache.WriteConnectionsMax (8)** – maximum number of JDBC connections allowed to be allocated concurrently to write to the data cache. If this number is exceeded, write request will be queued and blocked. Recommend increasing to handle more data update activity between the Master and DSP.

The following `tes-6.2.dsp` properties are what you would tune for Cache Tuning tradeoff between performance and memory/disk usage:

- **DataCache.PageCacheSize (50000)** – number of pages (of size `DataCache.PageSize`) in memory allocated to the data cache. The in-memory data cache allows for the fastest possible read/write access. Thus, for very large data caches, it is recommended that the data cache be given as much memory as possible.
- **DataCache.PageSize (4096)** – size (in bytes) of each page in the data cache. The data cache stores records in page-size chunks. Larger pages may improve read/write access when the data cache needs to fetch records from disk, with the added cost of a larger data cache in-memory and on disk. Another factor to keep in mind is the OS disk block size. Usually, it is recommended to keep the data cache page size the same as the OS disk block size.
- **DataCache.CacheSizeDefault (1000)** – number of objects each table in the data cache is allowed to keep in memory. More objects equals faster read/write access from and to the data cache.

The following `tes-6.2.dsp` properties are what you would tune for Cache Syncing, which affects the data sync between the Master database and Client Manager (DSP) cache.

- **CacheSynchronizer.Purge (N)** – Y to purge leftover deleted records during sync. Leftover records may exist in the cache if the Master deletes the records from its database when the Client Manager is not connected.
- **CacheSynchronizer.NumThreads (4)** – number of concurrent threads spawned to sync the cache. More threads increase throughput, but use more CPU and memory. Recommend to set at or below number of physical cores on machine.
- **CacheSynchronizer.StreamCommitSize (1000)** – number of records committed to the cache in a batch. Larger batches improve throughput, but use more memory.
The following *master.props* properties are what you would tune for the connections used to read/write to the Master database:

- Shared between normal Master operation PLUS cache sync (both can be high I/O).
- Each Client Manager can configure X number of sync threads = Master needs X number of DB connections.
- DatabaseConnections (20)

**Master Messaging**

Master messaging impacts only the Master, but it has a big impact on the performance of the Master and thus indirectly the Client Manager as well. The Master is designed to be a heavily multi-threaded application. By tuning the messaging parameters, the Master is better able to utilize the threading capabilities of the system.

All work in the Master is performed by the following message threads:

- **Message Queues**
  
  Before a message is sent to an object, it is first posted to one of the application’s message queues. Each queue or set of queues is allocated for a specific type of message. For example, the default queues are for general messages, the compiler queue is for compile messages, and the communication queues are for communication messages. Each queue has a priority that determines how often its messages get processed. Higher priority queues have their messages processed more frequently than lower priority ones.

- **Message Threads**

  Message threads are the workhorse of the application. They are responsible for pulling messages out of queues and then executing them. Depending on the application configuration, there may be anywhere from a dozen to a hundred threads running inside the application. Generally speaking, more threads equal better performance, since each CPU can execute a thread at the same time as another CPU. So, technically, a dual core system can process twice as many messages as a single core system. However, in reality, because each thread eats up a bit of memory and CPU, performance is expected to degrade when there are too many threads for the system to handle.

- **Message Thread Pools**

  A message thread pool groups together a set of messages threads with a set of message queues. The intent is to force threads from a pool to only process messages for queues from the same pool. This guarantees if threads in one pool are busy, messages in another pool will still get processed by free threads in that pool.

To configure the Master messaging:

- MessageThreads = general workhorse threads
- SpecialMessageThreads = adapter-related threads
- EventMessageThreads = event-related threads
- CommThreads = communication threads

**Note:** As a best practice, retain no more than 20 M message log records for better performance.

**WebClient Performance Parameters**

The following *tes-6.0.0.0.dsp* properties are what you would tune for the WebClient performance:

- JdbcFetchSize – Allows you to specify the number of rows fetched with each database round trip for a query. Setting the fetch size overrides the row-prefetch setting and affects the subsequent queries executed through that statement object. The default value is 100.
- **JdbcBatchSize** – The client manager maintains a temporary table to track the filtered records in the job and job activity screen. This parameter controls the size of batch insertion into the temporary table. The default value is 100.

- **JobThreadSliceSize** – When the user in the Webclient or REST API, requests the job or job run data, the Client manager API uses parallel threads to fetch the child information of the top level job group. For a flat hierarchy with lot of top level groups, this tends to take over most of the thread resources that interact with database for one user request.

  This parameter allows the user to configure the number of top level groups child information to fetch in single thread. The higher number reduces the number of threads required to satisfy one user request and allows the system to be more responsive for many concurrent users. Higher number though might increase the response time for one user, as the child information fetch for top level job group is less concurrent per user.

  The JobThreadSliceSize with a value of 8 was tested across different environments (small, medium, and large). An optimal result was obtained based on the collected metrics.

- **GridScrollingDataFetchDelay** – Defines the time delay to load the page when scrolling in data grids and drop-downs. By default, the delay value is 1000. The allowed values are >= 400 and <=1000 milliseconds. The recommended value is 400 milliseconds.

  If the delay is reduced, the user interface will be more responsive. But, more page request for a given amount of scroll is sent to the client manager with increased load on the client manager or database.

- **GridRecordFetchPageSize** – Defines the number of pages to load as per request for every data grid and drop-downs. By default, the page size value is 1. The allowed values are >= 1 and <=3. The recommended value is 2.

  If the browser height dictates 100 rows per page having a page value of 2, the rows fetched per request will be increased to 200.

  If the page size is increased, the user interface will have more records and the view will be loaded seamlessly, which may increase the load on the client manager or database to fetch more records.

  **Note:** In a large environment, values lower than the default values of GridScrollingDataFetchDelay and GridRecordFetchPageSize parameters, are not recommended.

### Size-Based Guidelines for Tuning Parameters

Small, medium, and large configurations require parameters to be tuned differently. Set the parameters as indicated in this section.

#### Small Configuration

tes-6.0.0.0.dsp
- CacheSynchronizer.NumThreads=2
- DataCache.ReadConnectionsMin=5
- DataCache.ReadConnectionsMax=10
- DataCache.WriteConnectionsMin=5
- DataCache.WriteConnectionsMax=10
- DataCache.PageCacheSize=16384
- DataCache.ConnectionPoolMinSize=5
DataCache.ConnectionPoolMaxSize=10
DataCache.StatementCacheSize=750
ClientNode.MinSessionPoolSize=5
ClientNode.MaxSessionPoolSize=10
ClientNode.MaxConcurrentMessages=10
JdbcFetchSize=200
JdbcBatchSize=300
JobThreadSliceSize=8
GridScrollingDataFetchDelay=400
GridRecordFetchPageSize=2

`clientmgr.props`
JVMARGS= -Xms2048m -Xmx8192m -XX:PermSize=1024m -XX:MaxPermSize=1024m
ClientSession.MinSessionPoolSize=5
ClientSession.MaxSessionPoolSize=10
ClientSession.MaxConcurrentMessages=10
DataSource.MinSessionPoolSize=5
DataSource.MaxSessionPoolSize=10
DataSource.MaxConcurrentMessages=10

`master.props`
MessageBroker.MemoryLimit=2048
MessageBroker.StoreLimit=32768
MinSessionPoolSize=250
MaxSessionPoolSize=2500
MaxConcurrentMessages=5
ClientConnection.MinSessionPoolSize=10
ClientConnection.MaxSessionPoolSize=50
ClientConnection.MaxConcurrentMessages=10
DatabaseConnections=35
MessageThreads=25
EventMessageThreads=25
CommThreads=25
SpecialMessageThreads=25
JVMARGS= -Xms2048m -Xmx8192m -XX:PermSize=256m -XX:MaxPermSize=256m
transporter.cmd
JVMARGS= -Xms1024m -Xmx4096m

transporter.props
READJOBS_PAGINATED=true
READJOBS_BATCHES=false
READJOBS_ALL=false
READ_BATCHES=10000
XPORTER_DEBUG=YES

Medium Configuration
tes-6.0.0.0.dsp
CacheSynchronizer.NumThreads=4
DataCache.ReadConnectionsMin=10
DataCache.ReadConnectionsMax=20
DataCache.WriteConnectionsMin=10
DataCache.WriteConnectionsMax=20
DataCache.PageCacheSize=131072
DataCache.ConnectionPoolMinSize=10
DataCache.ConnectionPoolMaxSize=20
DataCache.StatementCacheSize=1500
ClientNode.MinSessionPoolSize=10
ClientNode.MaxSessionPoolSize=20
ClientNode.MaxConcurrentMessages=10
JdbcFetchSize=200
JdbcBatchSize=300
JobThreadSliceSize=8
GridScrollingDataFetchDelay=400
GridRecordFetchPageSize=2

clientmgr.props
JVMARGS= -Xms4096m -Xmx20480m -XX:PermSize=2048m -XX:MaxPermSize=2048m
ClientSession.MinSessionPoolSize=10
ClientSession.MaxSessionPoolSize=20
ClientSession.MaxConcurrentMessages=10
DataSource.MinSessionPoolSize=10
DataSource.MaxSessionPoolSize=20
DataSource.MaxConcurrentMessages=10

**master.props**
MessageBroker.MemoryLimit=4096
MessageBroker.StoreLimit=65536
MinSessionPoolSize=500
MaxSessionPoolSize=5000
MaxConcurrentMessages=10
ClientConnection.MinSessionPoolSize=10
ClientConnection.MaxSessionPoolSize=100
ClientConnection.MaxConcurrentMessages=10
DatabaseConnections=75
MessageThreads=50
EventMessageThreads=50
CommThreads=50
SpecialMessageThreads=50

`JVMARGS= -Xms4096m -Xmx16384m -XX:PermSize=256m -XX:MaxPermSize=256m`

**transporter.cmd**
`JVMARGS= -Xms3072m -Xmx10240m`

**transporter.props**
READJOBS_PAGINATED=true
READJOBS_BATCHES=false
READJOBS_ALL=false
READ_BATCHES=10000
XPORTER_DEBUG=YES

**Large Configuration**

tes-6.0.0.0.dsp
CacheSynchronizer.NumThreads=8
DataCache.ReadConnectionsMin=50
DataCache.ReadConnectionsMax=100
DataCache.WriteConnectionsMin=50
DataCache.WriteConnectionsMax=100
DataCache.PageCacheSize=1048576
DataCache.ConnectionPoolMinSize=20
DataCache.ConnectionPoolMaxSize=40
DataCache.StatementCacheSize=7500
ClientNode.MinSessionPoolSize=50
ClientNode.MaxSessionPoolSize=100
ClientNode.MaxConcurrentMessages=10
JdbcFetchSize=200
JdbcBatchSize=300
JobThreadSliceSize=8
GridScrollingDataFetchDelay=400
GridRecordFetchPageSize=2

clientmgr.props
JVMARGS= -Xms6144m -Xmx24576m -XX:PermSize=3072m -XX:MaxPermSize=3072m
ClientSession.MinSessionPoolSize=50
ClientSession.MaxSessionPoolSize=100
ClientSession.MaxConcurrentMessages=10
DataSource.MinSessionPoolSize=50
DataSource.MaxSessionPoolSize=100
DataSource.MaxConcurrentMessages=10

master.props
MessageBroker.MemoryLimit=8192
MessageBroker.StoreLimit=65536
MinSessionPoolSize=1000
MaxSessionPoolSize=10000
MaxConcurrentMessages=25
ClientConnection.MinSessionPoolSize=25
ClientConnection.MaxSessionPoolSize=100
ClientConnection.MaxConcurrentMessages=25
DatabaseConnections=75
MessageThreads=75
EventMessageThreads=75
CommThreads=75
SpecialMessageThreads=75

JVMARGS= -Xms4096m -Xmx24576m -XX:PermSize=256m -XX:MaxPermSize=256m

transporter.cmd
JVMARGS= -Xms4096m -Xmx16384m

transporter.props
READJOBS_PAGINATED=true
READJOBS_BATCHES=false
READJOBS_ALL=false
READ_BATCHES=10000
XPORTER_DEBUG=YES
4

Transporter Performance

General Best Practices

Consider the following best practices while using the TWA Transporter:

- Use server-side filter to read specific jobs.
- Run only one instance of transporter at a time in a machine.
- Have small number of top level groups.
- Transport during off peak hours or when Client Manager usage is significantly less.

Transporter Job Read Options

Configurations have been made available to provide improved performance for unfiltered job reads. Multiple options are available for flexibility. Configuring these options may require tuning based on the customer environment. For tuning purpose, it would best to run the Transporter in debug mode with an open console so that you can view how the reads are performing.

To run the Transporter in debug mode, include `XPORTER_DEBUG=YES` in the Transporter.props file and run the `transporter.cmd` script located in bin.

The REST call `job.getList` has been replaced with the following options:

**Parameters Configured via Transporter.props**

Only one of the following parameters should be set to true at a time:

- `READJOBS_PAGINATED`
- `READJOBS_ALL`
- `READJOBS_BATCHES`

The `READ_BATCHES` parameter applies to `READJOBS_PAGINATED` or `READJOBS_BATCHES`.

If none of these parameters is set, the default configuration for read is (`READ_BATCHES=500`, `READJOBS_BATCHES=true`)

The `READ_BATCHES` parameter is used when reading paginated or batched reads.

The `READJOBS_PAGINATED` parameter determines whether to read jobs in pages.

The `READJOBS_BATCHES` parameter determines whether to read jobs in batches.

The `READJOBS_ALL` parameter determines whether to read all, given the min and max job ID.

**READJOBS_PAGINATED**

READJOBS_PAGINATED configures the Client Manager to return job data in pages, with the batches based on the `READ_BATCHES` value.
For example, READ_BATCHES=1000 and READJOBS_PAGINATED=true, tells the Client Manager to return job data in batches of 1000. This approach reduces the overhead on the Client Manager as data is sent in smaller batches. Increasing the READ_BATCHES value will reduce the number of requests sent to the Client Manager since the jobs are returned in larger batches.

Note: This approach may have less benefit given many jobs (i.e. 50K or more). The batching is done at the Client Manager level.

**READJOBS_BATCHES**

READJOBS_BATCHES reads jobs based on a given range of job IDs, where the range is specified via READ_BATCHES.

For example, if you have 50,000 job records whose job IDs start at 1 and end at 50000, and you have set READ_BATCHES=1000 and READJOBS_BATCHES=true, requests will be sent to the Client Manager to query job records in ranges, until no more records are returned, as follows.

```
jobid >=1 and jobid <=1001
jobid >=1002 and jobid <= 2002
jobid >=2003 and jobid <= 3003
...
```

If all the job IDs are sequential and start at 1, then each batch request will result in roughly 1000 records. However, if there are large gaps in the job IDs, due to mass job deletes for example, the request may return fewer results depending on where the job record ID falls in that range. While executing the read and running Transporter in the debug mode, if you find that very few or 0 records are returned given a READ_BATCHES configuration, then increasing this value will be necessary to reduce the number of requests that return 0 or few results.

Note: This approach appears to be more beneficial when there are many job records (50K or more).

**READJOBS_ALL**

READJOBS_ALL reads all jobs based on the first and last job ID. The result is that all jobs will be read in a single request. This approach is different from the job.getList call in that while both return all jobs, this request adds a query condition to the request, which seems to produce better performance. However, because all records are returned in a single request, the Client Manager will need to process all the records to send to Transporter.

Note: If there are many job records, the overhead on the Client Manager may be too high.