

TIBCO ActiveMatrix® BPM Performance Tuning Guide

*Software Release 4.2
August 2017*

Document Update: December 2017

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO ActiveMatrix BPM, TIBCO Administrator, TIBCO Business Studio, TIBCO Enterprise Message Service, TIBCO General Interface, TIBCO Hawk, TIBCO iProcess, TIBCO JasperReports, TIBCO Spotfire, TIBCO Spotfire Server, and TIBCO Spotfire Web Player are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2005-2017 TIBCO Software Inc. All rights reserved.

TIBCO Software Inc. Confidential Information

Contents

TIBCO Documentation and Support Services	7
Tuning Overview	9
System Monitoring	10
Monitoring and Alerting	10
CPU Usage	11
Memory	13
Disk	13
Network	14
Operating System Configuration	15
Maximum Open File Descriptors	15
Non-Uniform Memory Access (NUMA) Systems	15
Database Tuning	16
Oracle Tuning	16
Database Disk Configuration	16
Redo Logs	16
Memory Configuration	17
Configuring Locking Operation on Retry When Receiving/Retrieving a Message	17
Parameter Configuration	17
Monitoring	18
SQL Execution Plans	18
Tune SQL Execution Plans	19
DB2 Tuning	20
Microsoft SQL Server Tuning	20
Virtual Machines	21
Virtual Machine Stuns	21
Time Drift	21
Java Virtual Machines (JVMs)	22
JVM Monitoring	22
JConsole	22
JVisualVM	23
jstat	23
JVMInfo	23
Hawk	25
Oracle JVM Options	25
Default JVM Settings	25
Changing the Value of -Xmx	26

Changing the Value of MaxPermSize	26
Other JVM Settings	27
Java Heap Memory	27
Permanent Heap	27
Thread Stack Size	28
Garbage Collection	28
Miscellaneous JVM Options	29
Information Resources	30
IBM JVM Options	30
AIX Settings	31
JVM Threads	32
ActiveMatrix BPM Configuration	33
Setting the ActiveMatrix BPM Tuning Options	33
TIBCO Configuration Tool (TCT)	33
Transaction Log File	34
BPM Threads	34
Sequence Caching	34
Auditing Levels	35
JDBC Resources	35
Maximum Connections	36
Prepared Statement Cache Size	36
Turning off JDBC Connection Validation	37
Deallocating Unused JDBC Connections	38
HTTP Connectors	38
Memory Usage	39
HTTP Client	39
Background Processing	39
Process Manager (PM)	40
Business Resource Management (BRM)	40
Event Collector (EC)	40
Statistics Collector (SC)	40
Thread Pools	41
Process Engine (PVM)	41
User Application (UserApp)	42
BPM Filtering and Sorting of Work Lists and Audit Entries	42
BPM Nodes	42
Configuring How Often the Process Engine Looks for Work Requests	43
Locking of Signal Definitions	43
Expired Signals Job	44

Maximum Listener Count	44
Configuring the Maximum Number of Tasks per Process Instance	44
Configuring Cleanup of the PVM_WORK_ITEM or PVM_REQUEST_QUEUE Database Tables	45
JVM Properties Reference	46
Load Balancer	56
Process Design	57
LDAP	59
Windows Tools	60

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site. To directly access documentation for this product, double-click the following file:

`TIBCO_HOME/release_notes/TIB_amx-bpm_version_docinfo.html`

where *TIBCO_HOME* is the top-level directory in which TIBCO products are installed. On Windows, the default *TIBCO_HOME* is `C:\tibco`. On UNIX systems, the default *TIBCO_HOME* is `/opt/tibco`.

The following documents for this product can be found on the TIBCO Documentation site:

- TIBCO ActiveMatrix BPM SOA Concepts
- TIBCO ActiveMatrix BPM Concepts
- TIBCO ActiveMatrix BPM Developer's Guide
- TIBCO ActiveMatrix BPM Web Client Developer's Guide
- TIBCO ActiveMatrix BPM Tutorials
- TIBCO ActiveMatrix BPM Business Data Services Developer Guide
- TIBCO ActiveMatrix BPM Case Data User Guide
- TIBCO ActiveMatrix BPM Event Collector Schema Reference
- TIBCO ActiveMatrix BPM - Integration with Content Management Systems
- TIBCO ActiveMatrix BPM SOA Composite Development
- TIBCO ActiveMatrix BPM Java Component Development
- TIBCO ActiveMatrix BPM Mediation Component Development
- TIBCO ActiveMatrix BPM Mediation API Reference
- TIBCO ActiveMatrix BPM WebApp Component Development
- TIBCO ActiveMatrix BPM Administration
- TIBCO ActiveMatrix BPM Performance Tuning Guide
- TIBCO ActiveMatrix BPM SOA Administration
- TIBCO ActiveMatrix BPM SOA Administration Tutorials
- TIBCO ActiveMatrix BPM SOA Development Tutorials
- TIBCO ActiveMatrix BPM Client Application Management Guide
- TIBCO ActiveMatrix BPM Client Application Developer's Guide
- TIBCO Openspace User's Guide
- TIBCO Openspace Customization Guide

- TIBCO ActiveMatrix BPM Organization Browser User's Guide (Openspace)
- TIBCO ActiveMatrix BPM Organization Browser User's Guide (Workspace)
- TIBCO ActiveMatrix BPM Spotfire Visualizations
- TIBCO Workspace User's Guide
- TIBCO Workspace Configuration and Customization
- TIBCO Workspace Components Developer Guide
- TIBCO ActiveMatrix BPM Troubleshooting Guide
- TIBCO ActiveMatrix BPM Deployment
- TIBCO ActiveMatrix BPM Hawk Plug-in User's Guide
- TIBCO ActiveMatrix BPM Installation: Developer Server
- TIBCO ActiveMatrix BPM Installation and Configuration
- TIBCO ActiveMatrix BPM Log Viewer
- TIBCO ActiveMatrix BPM Single Sign-On
- Using TIBCO JasperReports for ActiveMatrix BPM

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

Tuning Overview

Tuning and diagnostic techniques for ActiveMatrix BPM systems should be performed regularly so that you can optimize the performance of the system.

Because usage patterns and volumes evolve over time, regular monitoring and maintenance should be a planned and expected activity.



Most of the examples in this guide assume a Linux installation using an Oracle database. Equivalent tools for other platforms are listed [here](#).

System Monitoring

Monitoring the system is crucial to understanding whether memory, CPU, disk, or network are limiting the performance the system. These key aspects should be monitored on both the ActiveMatrix BPM and database systems on a regular basis.

There are many tools available to monitor systems. On Linux, `collectl` (<http://collectl.sourceforge.net/>) can run as a background service continually collecting CPU, memory, disk, process and network statistics that can then be examined at a later date, as well as providing live statistics.

```
[mjs@poker] collectl
```

#<-----CPU----->				-----Disks-----				>-----Network----->			
#cpu	sys	inter	ctxsw	KBRead	Reads	KBWrit	Writes	netKBi	pkt-in	netKBo	pkt-out
37	37	382	188	0	0	27144	254	45	68	3	21
25	25	366	180	20	4	31280	296	0	1	0	0
25	25	368	183	0	0	31720	275	2	20	0	1

Indications of a stressed system are:

- Constantly high CPU (greater than 90%)
- Disk writes or reads reaching the throughput of the sub-system
- Total network traffic reaching the bandwidth limit

There are many other tools available, and you should use whichever tool you are comfortable with that can gather the information required during testing.

Monitoring and Alerting

For short running tests, CPU, disk, and network information is generally analyzed on completion of the tests. For long running tests or, more specifically, user acceptance or production systems, it will be necessary to issue alerts when CPU usage is too high, disks are too busy and so on.

There are a number of applications available on Linux to monitor the operating system, as well as JVMs and databases.

For example:

- TIBCO Hawk[®] http://www.tibco.com/multimedia/ds-hawk_tcm8-731.pdf
- Nagios <http://www.nagios.com/solutions/linux-monitoring>
- Zabbix <http://www.zabbix.com/product.php>
- Cacti http://www.cacti.net/what_is_cacti.php

On Windows, the supplied Performance Monitor can be used to monitor the systems and add alerts if necessary. See the following articles for more information on using Performance Monitor:

- http://www.windowsnetworking.com/articles-tutorials/windows-2003/Windows_2003_Performance_Monitor.html
- <http://technet.microsoft.com/en-us/library/cc722414.aspx>

Also, there is an agent available for Nagios that enables the combining of system information from different platforms.

You will only be able to determine the exact effects of some of the configuration options, and therefore the most suitable settings for these options on your system, after trying them out. If you are thinking of changing any of the options described, TIBCO therefore recommends that you run a test against the

system to see how the Java Virtual Machine memory and the CPU usage, on both the database and the ActiveMatrix BPM machines, are affected by different values.

CPU Usage

CPU usage is a good indicator of overall system performance.

A useful text-mode interactive tool for monitoring CPU usage is **htop** (<http://htop.sourceforge.net/index.php?page=main>). This provides enhanced information that the **top** command does not provide.

The screenshot shows the htop interface with the following details:

- System Statistics:**
 - Tasks: 75, 32 thr; 1 running
 - Load average: 0.01 0.07 0.08
 - Uptime: 1 day, 13:27:36
 - Mem: 630/3819MB
 - Sup: 0/0MB
- Process List:**

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	Command
17721	hisham	40	0	3976	2032	1596	S	0.0	0.1	ssh -p 2222 hisham
17720	root	40	0	7452	4664	2348	S	0.0	0.1	python ./main.py p
2675	hisham	40	0	49836	27860	16656	S	0.0	0.7	pidgin
1232	root	40	0	1860	340	228	S	0.0	0.0	/bin/dhccpd wlan0 -h par
1197	root	40	0	3776	900	656	S	0.0	0.0	wpa_supplicant -B -i wla
1193	hisham	40	0	30092	16176	9568	S	0.0	0.4	/usr/bin/python -0 /usr/
1187	hisham	40	0	20584	5716	4252	S	0.0	0.1	xfce4-settings-helper
1183	hisham	40	0	21156	8820	6868	S	0.0	0.2	xfdesktop
1181	hisham	40	0	19684	6240	5120	S	0.0	0.2	Thunar --daemon
1179	hisham	40	0	31796	12464	8700	S	0.0	0.3	xfce4-panel
1608	hisham	40	0	31796	12464	8700	S	0.0	0.3	xfce4-panel
1207	hisham	40	0	33892	12024	8868	S	0.0	0.3	/System/Index/lib/xfc
1214	hisham	40	0	33892	12024	8868	S	0.0	0.3	/System/Index/lib/
1206	hisham	40	0	0	0	0	Z	0.0	0.0	xfce4-battery-p
1204	hisham	40	0	23420	12060	8296	S	0.0	0.3	/System/Index/libexec

The following table shows some example CPU usage scenarios

Scenario	Possible Causes
High database CPU usage and low ActiveMatrix BPM usage	The database system could be underpowered for the ActiveMatrix BPM system configuration and tests being performed. It can also indicate that some SQL queries being performed are using excessive CPU, most probably due to Oracle using an incorrect execution plan for a SQL query. See Database for more detail on this.

Scenario	Possible Causes
Low database CPU usage and high ActiveMatrix BPM usage	<p>If pageflows are used, it is normal for the CPU usage of the ActiveMatrix BPM system to be larger than that of the database, taking into account the relative size of each system. This can also be an indication that the JVM for the BPM nodes has run out of or is low on memory.</p> <p>When a JVM runs out of heap it can spend a large amount of its time performing garbage collection in order to free up memory. This can result in high CPU usage on the ActiveMatrix BPM system, but low throughput (seen as low CPU on the database machine or slow client responses).</p>
High database CPU usage and high ActiveMatrix BPM usage	<p>The system could be working correctly, but masking other issues such as those mentioned above. If CPU is high on the ActiveMatrix BPM machine, first determine how many threads are running within the BPM node's JVM by either: using JConsole/JVisualVM, taking a thread dump with jstack, or through Administrator (not available prior to ActiveMatrix BPM 2.2.0).</p> <p>If the CPU is high and there are a large number of threads within the JVM (greater than 500), the system might be spending a lot of time context switching, which results in threads getting little actual CPU time. This can lead to slower throughput. Sometimes adding more threads can actually slow performance of a system.</p>
Low database CPU usage and low ActiveMatrix BPM usage	<p>Either the system is not being exercised by the clients enough, or there is a bottleneck somewhere in the system (for example, I/O or network).</p> <p>There may also be contention for resources within the ActiveMatrix BPM nodes. For example, the number of connections in a JDBC connection pool may not be large enough, or the number of HTTP client connections insufficient.</p> <p>Correlate the expected performance, from the client perspective, using system monitoring (disk, CPU, and network), JVM monitoring, as well as logs from the system, database, and ActiveMatrix BPM.</p>

Memory

For a system to perform well, sufficient physical memory must be available for all processes running on the system (for example, TIBCO Enterprise Message Service™, Apache LDAP, as well as the ActiveMatrix BPM system and BPM nodes).

The main item to check for is that no swapping is taking place, or at least none that involves the ActiveMatrix BPM processes. Swapping to disk can cause a dramatic drop in performance, especially if memory is being swapped to slow disk systems, which are also being shared by the operating system, EMS, database, ActiveMatrix BPM, and so on.

The utility `vmstat` (<http://www.linuxjournal.com/article/8178>) or `collectl` can be used to monitor memory and swapping.

```
[root@uk-caswell-centos ~]# collectl --vmstat
waiting for 1 second sample...
#procs -----memory (KB)----- --swaps-- --io-- --system-- --cpu-----
# r b swpd free buff cache inact active si so bi bo in cs us sy id wa
0 0 0 6965M 107M 447M 377M 173M 0 0 0 100 93 122 0 0 99 0
0 0 0 6965M 107M 447M 377M 173M 0 0 0 0 65 120 0 0 100 0
0 0 0 6965M 107M 447M 377M 173M 0 0 0 0 70 111 0 0 100 0
0 0 0 6965M 107M 447M 377M 173M 0 0 0 0 50 101 0 0 99 0
0 0 0 6965M 107M 447M 377M 173M 0 0 0 1516 138 116 0 0 99 0
0 0 0 6965M 107M 447M 377M 173M 0 0 0 0 154 695 0 0 99 0
```

Disk

The usage and configuration of disks can have a marked impact on the performance of the ActiveMatrix BPM system, especially with respect to logs for EMS, the database, BPM, or the ActiveMatrix HOWL (transaction) logs. An understanding of the physical structure of the disk systems being used by various components of the system is needed to identify potential performance bottlenecks.

The tool `collectl` can be used to monitor disks.

```
collectl -sD
# DISK STATISTICS (/sec)
#
#Name <-----reads-----> <-----writes-----> <-----averages-----> Pct
      KBytes Merged IOs Size KBytes Merged IOs Size RWSIZE QLen Wait SvcTim Util
sda      0      0      0      0      291      67      5      58      58      0      0      0      0
sdb      0      0      0      0      0      0      0      0      0      0      0      0      0
dm-0      0      0      0      0      291      0      73      4      4      0      1      0      0
dm-1      0      0      0      0      0      0      0      0      0      0      0      0      0
hda      0      0      0      0      0      0      0      0      0      0      0      0      0
```

Disk Setup	Possible Problems
Database/ActiveMatrix BPM/EMS files on OS partition	Most systems do not have a large disk partition for the operating system. If the database installation, EMS or ActiveMatrix BPM are using the operating system partition, there may be issues, especially if the same partition is used for memory swapping.
Oracle data and redo log files on same partition	ActiveMatrix BPM makes heavy use of transactions resulting in Oracle redo logs being a potential performance bottleneck. Ideally the Oracle data and redo files should be located on separate partitions.
EMS files and BPM installation on same partition	Both EMS and BPM make heavy use of files for logging. For optimum performance BPM logging and ActiveMatrix HOWL logs should be located on separate partitions to EMS.

Disk Setup	Possible Problems
ActiveMatrix HOWL logs and BPM logging on same partition	By default the ActiveMatrix HOWL logs and all BPM logging is configured to the same partition. The location of the HOWL logs can be changed using the <code>amx.node.txlogdir</code> property. See <i>TIBCO ActiveMatrix SOA Administration</i> for more information.



For ActiveMatrix BPM installations running against Microsoft SQL, take into account the location of the MS-DTC log files, which by default are on the same partition as the operating system on the Microsoft SQL system (for example, C:\windows\system32\MSDtc). If the default location for MS-SQL database files is being used, these will also exist under the OS partition.

Network

Providing sufficient network reliability and capacity are essential for the good performance and operation of any system.

The amount of data passed between ActiveMatrix BPM and the database, as well as that used for client interactions with ActiveMatrix BPM, is usually well below the bandwidth of most networks. However, the quality of the network and round-trip times merit close attention.

For all systems, and especially for larger systems with multiple BPM nodes (either across multiple machines or all on the same machine), monitor the network utilization.

A tool that presents live information is iftop (<http://www.ex-parrot.com/pdw/iftop/>), which gives a summary of interfaces and network traffic:

	10b	100b	1.00kb	10.0kb	100kb	1.00Mb
mythic.beasts.org	=>	camhfw02.camh.net		357Kb	269Kb	355Kb
	<=			11.1Kb	8.12Kb	10.7Kb
mythic.beasts.org	=>	host71-89.pool80117.inter		223Kb	260Kb	172Kb
	<=			4.83Kb	4.77Kb	3.13Kb
mythic.beasts.org	=>	66.196.72.58		0b	31.6Kb	7.91Kb
	<=			0b	771b	193b
mythic.beasts.org	=>	204.109.64.2		11.7Kb	13.0Kb	3.25Kb
	<=			208b	698b	175b
mythic.beasts.org	=>	213.122.26.58		11.7Kb	8.33Kb	2.08Kb
	<=			448b	596b	149b
mythic.beasts.org	=>	193.201.200.170		0b	4.00Kb	1.00Kb
	<=			0b	1.07Kb	274b
mythic.beasts.org	=>	62.253.164.70		0b	3.64Kb	932b
	<=			0b	961b	240b
mythic.beasts.org	=>	155.198.5.111		0b	3.72Kb	951b
	<=			0b	800b	200b
mythic.beasts.org	=>	128.91.201.64		0b	3.76Kb	962b
	<=			0b	741b	185b
<hr/>						
TX:	cumm:	3.35MB	peak:	690Kb	rates:	607Kb 609Kb 603Kb
RX:		124KB		37.9Kb		16.8Kb 21.7Kb 19.9Kb
TOTAL:		3.47MB		712Kb		623Kb 631Kb 623Kb

One aspect of networking that is harder to assess is the latency of the network between the ActiveMatrix BPM system and the database. Latency refers to a time delay; for example, the gap between when a device requests access to a network and when it receives permission to transmit. The network may be very quick and have plenty of bandwidth, but if the latency is high, performance will be impacted.

Oracle have many documents on network performance, in particular, refer to section 7.3.8 of Oracle Configuration Best Practices (<http://www.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10726/configbp.htm>) says the following:

[the] infrastructure must have the following characteristics:

- Sufficient bandwidth to accommodate the maximum redo generation rate
- Minimal latency to reduce the performance impact on the production database
- Multiple network paths for network redundancy

The required bandwidth of a dedicated network connection is determined by the maximum redo rate of the production database.

Operating System Configuration

Most operating systems need little or no extra configuration for smaller ActiveMatrix BPM systems. However, for larger ActiveMatrix BPM installations, you will probably need increase various configuration settings.

A good starting point for this is the operating system configuration settings recommended for an Oracle installation (http://www.oracle-base.com/articles/11g/oracle-db-11gr2-installation-on-oracle-linux-5.php#manual_setup), especially maximum open file descriptors.

Maximum Open File Descriptors

When running large user centric tests, or on a system that has a number of BPM nodes running under the same user, consider increasing the maximum number of open file descriptors that are configured for the user.

The `ulimit -n` command can be used to display the user's current configuration.

For smaller systems, consider a soft limit of 4096. For larger tests, this has should be increased in the `/etc/security/limits.conf` file to:

```
* soft nofile 8196
* hard nofile 65536
```

Non-Uniform Memory Access (NUMA) Systems

In a NUMA system, the greater the distance between a processor and a memory bank, the slower the processor's access to that memory bank. Performance-sensitive applications should therefore be configured so that they allocate memory from the closest possible memory bank.

Having a single process, the JVM, span multiple NUMA nodes could be detrimental to performance. This can be alleviated by having multiple, small JVM processes used on multiple NUMA node systems (as opposed to a single large process), with each JVM being restricted to a single NUMA node.

The command `numactl` (<http://linux.die.net/man/8/numactl>) can be used to view the available nodes on the system, as well as controlling to which nodes a process is allowed to run on.

- NUMA systems and process allocation is unlikely to be an issue on most systems, but it is something to bear in mind when investigating performance issues
- On Windows, this information and control is available through Task Manager. Right-clicking on a process allows you to control the affinity of the process, and also onto which logical processors and/or NUMA nodes the process can run.



Database Tuning

You should closely monitor the performance of your database, and should be prepared to tune the database to ensure best performance. As with the other configuration options described, the best values to select vary from one installation to another.

Oracle Tuning

Database Disk Configuration

Database disk configuration and usage is the biggest bottleneck for an ActiveMatrix BPM system. Since BPM (excepting Pageflows) makes heavy use of the database, ensuring that the disk system for the database is well configured is essential, as well as understanding how the physical disks are configured on which the Oracle data and temp files are located.

Ensure that the main database files for ActiveMatrix BPM are on a fast disk system, separated from the operating system and other heavily used databases. The database files for the ActiveMatrix BPM database need to be resized according to the expected amount of data to be used during system operation.

The following SQL can be used to list tablespaces, their sizes, and the percentage free:

```
set linesize 140
SELECT df.tablespace_name TABLESPACE, df.total_space TOTAL_SPACE,
fs.free_space FREE_SPACE, df.total_space_mb TOTAL_SPACE_MB,
(df.total_space_mb - fs.free_space_mb) USED_SPACE_MB,
fs.free_space_mb FREE_SPACE_MB,
ROUND(100 * (fs.free_space / df.total_space),2) PCT_FREE
FROM (SELECT tablespace_name, SUM(bytes) TOTAL_SPACE,
ROUND(SUM(bytes) / 1048576) TOTAL_SPACE_MB
FROM dba_data_files
GROUP BY tablespace_name) df,
(SELECT tablespace_name, SUM(bytes) FREE_SPACE,
ROUND(SUM(bytes) / 1048576) FREE_SPACE_MB
FROM dba_free_space
GROUP BY tablespace_name) fs
WHERE df.tablespace_name = fs.tablespace_name(+)
ORDER BY fs.tablespace_name;
```

The specific BPM data and temp files should be on separate disks. Monitoring of the operating disks on the Oracle system and use of the Oracle Enterprise Manager's (OEM) I/O monitoring can help identify any bottlenecks.

Redo Logs

Since ActiveMatrix BPM makes heavy use of transactions, the configuration and location of the Oracle redo logs are critical. These files hold information about changes to the database.

For more information about Oracle redo logs, see http://docs.oracle.com/cd/B28359_01/server.111/b28310/online redo001.htm#ADMIN11302

The location, size, and number of redo log files can impact ActiveMatrix BPM performance. If Oracle reports that there is excessive log switching, increase the size of the redo log files.

To get an Oracle report for a test, run the Automatic Database Diagnostic Monitor (ADDM) report for a given period based on Oracle's snapshots. For more information on the ADDM reporting, see <http://www.oracle-base.com/articles/10g/automatic-database-diagnostic-monitor-10g.php>. This can be used to pick up heavy SQL being performed that might require a different execution plan or changes to indexes on tables.

Memory Configuration

The amount of memory configured for Oracle has an impact on the performance of ActiveMatrix BPM. If Oracle has to frequently go to the disk to satisfy queries, rather than doing so from memory, ActiveMatrix BPM throughput is affected.

Oracle reports can be used to see whether memory is adequately sized for specific test runs. For more information about memory management in Oracle, see http://docs.oracle.com/cd/B28359_01/server.111/b28310/memory.htm#BGBCBHEJ.

Configuring Locking Operation on Retry When Receiving/Retrieving a Message

Locking is enabled by default. Disabling it will help to avoid database row lock contention when dealing with high volumes of correlation and event handlers. It might introduce delays in processing messages when a message is being delivered at the exact same time as a corresponding receiver task.

To change the setting of locking and other related properties that are described below, use ActiveMatrix Administrator to update the Java Virtual Machine (JVM) configuration for the ActiveMatrix BPM nodes, as explained in the *TIBCO ActiveMatrix BPM SOA Administration* guide. You need to add (and set) the relevant properties on the **JVM Configuration** page within the **Configuration** tab. When you have done this, the properties are added to each BPM node's `.tra` file.

To disable locking system-wide set `com.tibco.bx.lockOperation` to **false**. If locking is disabled then a background job that checks the messages that have not been claimed yet will be started. To control its execution frequency set `com.tibco.bx.bpel.bg.unclaimedMsgProcessor.interval` (default is 30 minutes, lowest possible value is 1 minute - P1). The `com.tibco.bx.bpel.bg.unclaimedMsgProcessor.numMsgs` property controls a maximum number of messages that a job will pick up per execution (default is 500, minimum 50).

If system-wide locking is enabled then `com.tibco.bx.lockPerOperation=true` can be used to enable finer-grained control by using the following properties:

- module level: `com.tibco.bx.lockOperation.ModuleName` - if this property is defined (whether it is true or false) then the operation level property is not going to take effect.
- operation level: `com.tibco.bx.lockOperation.ModuleName.PortType.OperationName` (where `PortType` is in the form of `{NamespaceURI}localPart`).



The background job will **not** be running when system-wide locking is enabled, which introduces a possibility of an unclaimed message and a stuck instance in case of concurrent processing as mentioned above.

Parameter Configuration

As the number of BPM nodes and users increase on systems, you may need to change the values of the Oracle database parameters **open_cursors** and **processes**.

open_cursors

If the value of **open_cursors** is too low, the following message may be displayed when the system is under load:

```
ORA-01000: maximum open cursors exceeded
```

The default value is 50. To see the current value, enter the following Oracle command:

```
SHOW PARAMETERS cursors;
```

On an ActiveMatrix BPM system, TIBCO recommends setting this value to a minimum of 400. To change the current value, enter the following command (specific to your Oracle installation):

```
ALTER SYSTEM SET open_cursors=400 scope=spfile;
```

If the error message is displayed again, consider increasing the value.

processes

The parameter **processes** specifies the maximum number of operating system user processes that can simultaneously connect to Oracle.

The default value is 100. To see the current value, enter the following Oracle command:

```
SHOW PARAMETERS processes;
```

To determine whether you should increase the current value, you can add together all background processes (such as locks, job queue processes, and parallel execution processes) and the [Maximum Connections](#) values for each JDBC resource that will use the database. To change the current value, enter the following command (specific to your Oracle installation):

```
ALTER SYSTEM SET processes=1500 scope=spfile;
```

Monitoring

Monitoring the database during and after test runs will help understand where any issues with disk, memory, SQL, and configuration might be.

The Oracle Automatic Workload Repository (AWR - <http://www.oracle-base.com/articles/10g/automatic-workload-repository-10g.php>) and ADDM reporting tools provide views on the system during and after test runs.

Oracle Enterprise Manager Database Control (OEMDC - http://docs.oracle.com/cd/E11882_01/server.112/e10803/config_intro.htm#HABPT5145) provides a set of tools to monitor and manage your database.



Some aspects of these tools need a separate license from Oracle.

OEMDC provides useful recommendations for SQL performance, indexes and configurations. For example:

- OEMDC can highlight use of an inefficient SQL execution plan for work list queries, resulting in slow performance.
- OEMDC may recommend additional indexes depending on what work list filters and sort criteria are in use.
- Depending on processes and data being used by BPM, OEMDC can suggest other tables that may also benefit from additional indexes.

Tuning analysis data should be generated on the production system. Any recommendations should be implemented on a test system to determine their impact, by running the recommendations against the analysis data copied from your production system, before applying them to the production system.

SQL Execution Plans

Performance issues can occur if particular SQL queries against ActiveMatrix BPM database tables are taking a long time to execute because they are using inefficient SQL execution plans.

To enable the database optimizer to calculate the best execution plans, the ActiveMatrix BPM database objects should have their statistics refreshed at regular intervals. This can be accomplished by allowing the database to automatically collect and refresh the statistics (**GATHER_STATS_JOB**). You may need to collect the statistics manually if large updates have been performed, for example when a large number of instances have been started after going live.

You can make use of the `Explain Plan` statement to display the execution plan that shows how Oracle will carry out a given database query, and test to see if this is in fact the best plan available. Note that:

- If database access is slow, you may need to refresh stale database statistics.
- Your application architect and database administrator may find it helpful to understand how users are to filter and order both work lists and audit trails, so that they can configure database indexes or add additional database indexes if necessary. See [BPM Filtering and Sorting of Work Lists and Audit Entries](#) for more details.

If work list processing performance is slow, and deteriorating over time, this could be because a query to get a work list is executing slowly. This can impact other operations, resulting in error messages such as the following:

- In the BPM node log: [httpConnector_205] [ERROR]
org.hibernate.util.JDBCExceptionReporter - ORA-02049: timeout: distributed transaction waiting for lock
- In the TIBCO Workspace or TIBCO Openspace log: <error code="INTERNAL_SERVICE_FAULT_ALLOCATE_AND_OPEN_WORK_ITEM"message="could not load an entity:[com.tibco.n2.brm.orgentity.config.OrganisationalEntityConfig#AD66DB42-13DF-4609-B3A9-5B6D93543D74]"xmlns=""><parameter>org.hibernate.exception.SQLGrammarException: could not load an entity:[com.tibco.n2.brm.orgentity.config.OrganisationalEntityConfig#AD66DB42-13DF-4609-B3A9-5B6D93543D74]</parameter> </error>

Tune SQL Execution Plans

Use Oracle's SQL profiling capability to tune the SQL execution plans of inefficient SQL queries that are used by ActiveMatrix BPM.

Procedure

1. Collect fresh performance statistics from objects with either stale or non-existent statistics, by running a command similar to the following:

```
BEGIN
  DBMS_STATS.gather_schema_stats (
    ownname          => 'BPMUSER',
    estimate_percent  => dbms_stats.auto_sample_size,
    method_opt        => 'for all indexed columns',
    options           => 'GATHER AUTO',
    cascade           => TRUE);
END;
/
```

2. Run an Automatic Workload Repository (AWR) report against two database snapshots to identify any SQL queries that are taking a long time to execute.
3. For each offending SQL query:
 - a) Generate a SQL profiling report for the SQL statement (using the sqltrpt.sql tool).
 - b) Implement any SQL profile(s) that the report recommends to produce an improved execution plan for this SQL statement.

For further information, see the Oracle documentation.

DB2 Tuning

If you are using a DB2 database, the default configuration values are likely to be satisfactory for most systems. IBM provides the DB2 Utilities Suite which may be of use. For further information, see the IBM documentation.

Microsoft SQL Server Tuning

There are many areas that can be looked at for Microsoft SQL Server performance tuning, only a few of which are considered here as potential key areas. A DBA should understand these and many other areas that can be monitored to improve performance given a specific usage profile (depending on the design of their BPM applications).

There are a number of useful resources on SQL Server monitoring and maintenance:

- <http://www.sqlskills.com/help/accidental-dba/>
- <https://www.simple-talk.com/sql/database-administration/eight-steps-to-effective-sql-server-monitoring/>
- <http://www.mssqltips.com/sqlservertip/1861/sql-server-monitoring-scripts-with-the-dmvs/>

Database Engine Tuning Advisor (<http://msdn.microsoft.com/en-us/library/hh231122.aspx>), supplied with SQL Server, can analyze your database and recommend ways that you can optimize query performance. Activity Monitor (<http://technet.microsoft.com/en-us/library/hh212951.aspx>), also supplied with SQL Server, is useful to monitor the expensive SQL queries, resources waits, and file I/O. You can also view the Explain Plan for top SQL queries. Microsoft also supplies a Best Practices Analyzer for SQL Server (<http://www.microsoft.com/en-us/download/details.aspx?id=15289>). You can use this tool to scan your SQL Server systems and verify that common best practices have been implemented.

There is information available that can help you to find out the usage of indexes on your system. It can give you statistics on seeks, scans, lookups, and updates, which can help to identify heavily-used indexes as well as indexes that are unused or duplicated:

- <http://blogs.msdn.com/b/ialonso/archive/2012/10/08/faq-around-sys-dm-db-index-usage-stats.aspx>
- <http://www.sqlskills.com/blogs/kimberly/removing-duplicate-indexes/>
- http://www.sqlskills.com/blogs/kimberly/sp_helpindex2-to-show-included-columns-2005-and-filtered-indexes-2008-which-are-not-shown-by-sp_helpindex/

For information about missing indexes on SQL Server, see the following:

- <http://blogs.msdn.com/b/bartd/archive/2007/07/19/are-you-using-sql-s-missing-index-dmvs.aspx>
- <http://www.sqlperformance.com/2013/06/t-sql-queries/missing-index>

Virtual Machines

Running the database or ActiveMatrix BPM within a virtual machine can have an impact on performance. Virtual machines can be configured to share physical disks, memory, and so on. If these resources are allocated incorrectly (for example, both virtual machines sharing a physical disk or memory, performance can be degraded).

Examine the configuration of the virtual machines involved, assess the physical resources available, and how many virtual machines are sharing these resources.



For TIBCO's position on virtualization support, see Late Breaking News document LBN1-7058UQ, available from TIBCO Support. TIBCO is not in a position to provide customers with recommendations on configuring virtual systems, or estimating the performance overhead that virtual systems may levy. However, the information presented here is based on experience gained at customer sites.

Virtual Machine Stuns

VMware's Distributed Resource Scheduler (DRS/SDRS) can move a virtual machine depending on the need and availability of resources. This will "stun" a virtual machine, which can also occur during a virtual machine backup or snapshot delete.

Here are some VMware knowledge base articles on this subject:

- Virtual machine performance degrades while a vMotion is being performed - http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2007595
- A snapshot removal can stop a virtual machine for long time - http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1002836
- Taking a snapshot with virtual machine memory renders the virtual machine to an inactive state - http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1013163

Be aware of how a virtual machine environment might result in performance degradation when vMotion events are performed, snapshots removed, and backups taken. If necessary, reserve resources for the ActiveMatrix BPM and database systems in order to improve performance.

Time Drift

Time drift can happen on systems if the clocks are not kept in synchronization. This is more of an issue on virtual machines due to resource contention and "stuns". When time is brought back into synchronization, there could be possible "read timed out" exceptions thrown from the socket layers.

VMware have many articles on timekeeping and time drift:

- [Time in virtual machine drifts due to hardware timer drift](#)
- [Timekeeping best practices for Windows](#)
- [Timekeeping in VMWare Virtual Machines](#)

Java Virtual Machines (JVMs)

The ActiveMatrix BPM system runs within a Java Virtual Machine (JVM), and the settings specified for the JVM can have a great impact on the performance and stability of the overall system.

Various performance monitoring tools are described in [JVM Monitoring](#).

JVM options vary between manufacturers. In ActiveMatrix BPM:

- Installations on all platforms except AIX use the Oracle JVM. For this, see [Oracle JVM Options](#).
- Installations on AIX use the IBM JVM. For this, see [IBM JVM Options](#).

The amount of memory required by the JVM will be directly affected by the setting of other tuning options, such as the numbers of threads and of database connections. For example, if monitoring shows that the JVM is running low on memory, consider steps such as increasing the maximum heap size, or perhaps decreasing the number of threads, as explained in [JVM Threads](#).

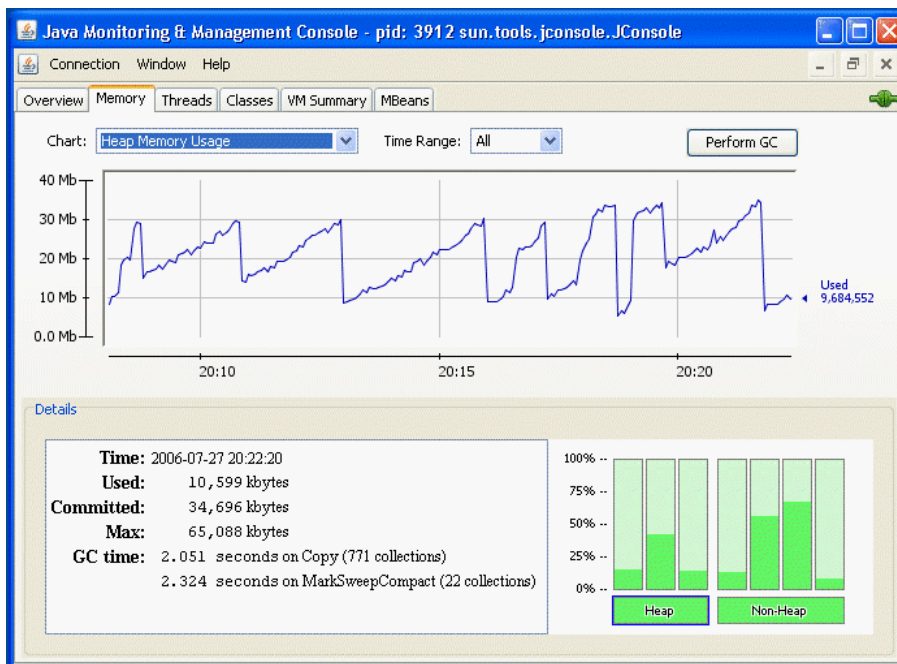
JVM Monitoring

You can use JConsole, JVisualVM, jstat, JVMInfo, or Hawk to get information about the memory, threads, and classes within a JVM.

JConsole

JConsole is a graphical monitoring tool using the JMX specification to provide information about the performance and resource consumption of the JVM. It is bundled with a JDK.

The main information to check in JConsole is the memory consumption and memory pools.



The bar chart on the lower right-hand side shows the memory consumed by the memory pools in heap and non-heap memory. The bar turns red when the memory used exceeds the memory usage threshold.

For more information, see <http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html>.

JVisualVM

JVisualVM is a tool that provides a visual interface for viewing detailed information about Java applications while they are running on a JVM. It is bundled with JDK 6 update 7 and later.

When you use JVisualVM, it is safest to use the `-Xshare:off` command-line option. This option disables class sharing, and if it is not used there is a possibility that the JVM being profiled may crash.

For more information, see <http://docs.oracle.com/javase/7/docs/technotes/guides/visualvm/>.

jstat

jstat (Java Virtual Machine Statistics Monitoring Tool) is a console tool that displays performance statistics for a JVM.

The `-gcutil` option identifies whether the JVM is spending a lot of time garbage collecting due to low memory. For example, the following command attaches to process 21891 and takes 7 samples at 250 millisecond intervals:

```
jstat -gcutil 21891 250 7
```

S0	S1	E	O	P	YGC	YGCT	FGC	FGCT	GCT
12.44	0.00	27.20	9.49	96.70	78	0.176	5	0.495	0.672
12.44	0.00	62.16	9.49	96.70	78	0.176	5	0.495	0.672
12.44	0.00	83.97	9.49	96.70	78	0.176	5	0.495	0.672

FGC (the number of full GC events) is an indicator of how busy the JVM is doing garbage collection. If this number increases quickly (for example, every 15 or 30 seconds or more frequently), it indicates memory issues.

The above output also indicates the eden (E), old (O) and permgen (P) memory % utilization.

For more information, see <http://docs.oracle.com/javase/6/docs/technotes/tools/share/jstat.html>.

JVMInfo

JVMInfo (ActiveMatrix BPM JVMInfo Utility) is a command-line tool that you can use to view the current status of a JVM process, for example memory, threads, or classes. You can use it to get a summary, or to monitor the process continuously.

Setup

JVMInfo uses the Java Runtime Environment (JRE) used by ActiveMatrix BPM. However, this JRE no longer ships with the libattach shared library (for example `libattach.so` on Linux). So, you must locate libattach in a separate JDK's JRE and add this location to the shared library path. For example, on Linux, `LD_LIBRARY_PATH` can be altered in the shell to include the location of the library:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/jdk1.7.0_25/jre/lib/amd64
```

Similar changes to the library path can be made on other platforms if required.

Running JVMInfo

On Windows, the path of JVMInfo is:

```
TIBCO_HOME\bpm\version\bin\jvminfo.bat
```

On UNIX, the path of JVMInfo is:

```
TIBCO_HOME/bpm/version/bin/jvminfo
```

Usage

```
jvminfo [-v] [-b] <process_ID> [<interval> [<count>]]
```

```
jvminfo -t <process_ID>
```


jvminfo -l

The options and variables have the following meanings:

-v	Verbose output.
-t	Dump thread information.
-l	List the JVMs running.
-b	When in Verbose mode, output the memory values in bytes. This gives a more precise figure than when -b is omitted.
<process_ID>	Process ID of the JVM.
<interval>	Sampling interval (in seconds).
<count>	Number of samples to take before terminating.

Example of Non-verbose Output

In non-verbose mode (-v is *not* used), memory values are displayed in bytes:

```
C:\...\bin>jvminfo.bat 4068 1 4
Running JVMs are: [7084, 4856, 4068, 5920]
DATETIME,PID,OSNAME,OSVERSION,OSARCH,NAME,VENDOR,VERSION,HEAPCOMMITTED,HEAPUSED,HEAP
MAX,NONHEAPCOMMITTED,NONHEAPUSED,NONHEAPMAX,THREADSCURRENT,THREADSPEAK,THREADSDAEMON
,THREADSTOTAL,CLASSESLOADED,CLASSESTOTAL,CLASSESUNLOADED
2013/06/04 09:53:35,4068,Windows 7,6.1,amd64,Java HotSpot(TM) 64-Bit Server VM,Sun
Microsystems Inc.,20.5-
b03,558956544,279721064,1037959168,233111552,230496032,318767104,72,72,65,545,24250,
26148,1898
2013/06/04 09:53:36,4068,Windows 7,6.1,amd64,Java HotSpot(TM) 64-Bit Server VM,Sun
Microsystems Inc.,20.5-
b03,558956544,282121792,1037959168,233111552,230512120,318767104,78,78,71,551,24250,
26148,1898
2013/06/04 09:53:37,4068,Windows 7,6.1,amd64,Java HotSpot(TM) 64-Bit Server VM,Sun
Microsystems Inc.,20.5-
b03,558956544,284522520,1037959168,233111552,230512280,318767104,84,84,77,557,24250,
26148,1898
```

Example of Verbose Output

In verbose mode (-v is used), memory values are displayed in bytes, kilobytes, megabytes, or gigabytes, as appropriate:

```
C:\...\bin>jvminfo.bat -v 4068
Running JVMs are: [5676, 4856, 4068, 5920]
JVM PID = 4068
Local connector address = service:jmx:rmi://127.0.0.1/stub/
r00ABXNyAC5qYXZheC5tYW5hZ2VtZW50LnJlbW90ZS5ybWkuUk1JU2VydmVySW1wbF9TdHViAAAAAAAAAIC
AAB4cgAaamF2YS5ybWkuc2VydmVyLlJlbW90ZVN0dWlp/tzJi
+FLGgIAAHhyABxqYXZhLnJtaS5zZXJ2ZXIuUmVtb3RlT2JqZWNO02G0kQxhMx4DAAB4cHc3AAAtVbmljYXN0U
mVmMgAADDE5Mi4xNjguMS41MQAAyV7+nQFW8GfdPhCX3t0AAAE/DmJvd4ABAHg=
2013/06/04 09:51:43
=====
OperatingSystem:
  name = Windows 7
  version = 6.1
  arch = amd64
Runtime:
  name = Java HotSpot(TM) 64-Bit Server VM
  vendor = Sun Microsystems Inc.
  version = 20.5-b03
Heap:
```



```

    committed = 1.2 GB (23.13%)
    used = 807.2 MB (15.55%)
    max = 5.1 GB
NonHeap:
    committed = 27.8 MB (73.13%)
    used = 27.4 MB (72.30%)
    max = 38 MB
Threads:
    current = 36
    peak = 36
    daemon = 29
    total = 509
Classes:
    loaded = 24241
    total = 26139
    unloaded = 1898
Memory Pool (Code Cache):
    used = 65.9 KB (1.07%)
    max = 6 MB
.
.
.

```

Example of Verbose Output with Memory Values in Bytes

In verbose mode with memory values in bytes (-v -b is used), *all* memory values are displayed in bytes:

```

C:\...\bin>jvminfo.bat -v -b 4068
Running JVMs are: [5676, 4856, 4068, 5920]
.
.
.
Heap:
    committed = 1,258,971,136 B (23.13%)
    used = 952,866,680 B (17.51%)
    max = 5,442,174,976 B
NonHeap:
    committed = 233111552 B (73.13%)
    used = 230461848 B (72.30%)
    max = 318767104 B
.
.
.

```

Hawk

TIBCO Hawk is a tool for monitoring and managing distributed applications and operating systems. It enables you to monitor JVMs and log files on multiple BPM nodes.

For more information, see "Monitoring ActiveMatrix BPM Using TIBCO Hawk Rulebases" in the *TIBCO ActiveMatrix BPM Administration* guide.

Oracle JVM Options

Default JVM Settings

ActiveMatrix BPM is delivered with default JVM settings.

```

-server -XX:MaxPermSize=1024m -XX:+UseParNewGC
-XX:MaxNewSize=256m -XX:NewSize=256m
-Xms4096m -Xmx4096m
-XX:SurvivorRatio=128 -XX:MaxTenuringThreshold=0
-XX:+UseTLAB
-XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:+CMSPermGenSweepingEnabled

```

The only settings that you are likely to need to change are **-Xmx** and **MaxPermSize**.



Do not change the values of any other JVM options unless you have investigated all other solutions. **Do not** try to change any such values without input from TIBCO Support.

Changing the Value of -Xmx

The **-Xmx** option is one of the JVM settings you might need to change.

What it does

The **-Xmx** option limits the maximum size of the Java heap. This is a continuous memory region where all Objects data—instances of class, primitives, and references—is stored. It forms a large part of the *process heap*. Properly tuning this parameter can reduce the overhead of garbage collection; if it is set too low, the result can be a large number of minor garbage collections. This reduces server response time and throughput.

Original value

When you initially configure ActiveMatrix BPM you can select one of the following default values for **-Xmx** using the **Maximum Java Heap Size** field in TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

For development systems: 1024 mb.

For production systems: 4096 mb.

Monitoring

Regular monitoring of your JVM is necessary to show whether or not you need to change the value of this option. If the heap memory is being used up, try increasing the value of **-Xmx**. This may (depending on the setting of the **-Xss** [Thread Stack Size](#) option) enable more threads or connections to be created. However, do not use a value so large that it would cause swapping or impact garbage collection performance, as this would then drastically reduce the performance of BPM system.

Changing the value

You can change it using the TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).



Do not set this to less than the value for **-Xms** (see [Java Heap Memory](#)), or BPM will be unable to create a JVM.

Changing the Value of MaxPermSize

The **-XX:MaxPermSize** option is one of the JVM settings you might need to change.

What it does

The **-XX:MaxPermSize** option specifies the maximum size for the *permanent generation*, which is the memory holding objects such as classes and methods. Properly tuning this parameter can reduce memory issues in the permanent generation. It can also reduce the need for major garbage collections (minor garbage collections do not affect the permanent generation). This reduces server response time and throughput.

Original value

When you initially configure ActiveMatrix BPM you can select one of the following default values for **MaxPermSize** using the **Max Java PermGen Size** field in TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

For development systems: 256 mb.

For production systems: 1024 mb.

Monitoring

Regular monitoring of your JVM is necessary to show whether or not you need to change the value of this option.

Changing the value

You can change it using the TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

Other JVM Settings

You are unlikely to need to change any other JVM settings. However, this section discusses some other settings for completeness.



Do not change the values of any other JVM options unless you have investigated all other solutions. **Do not** try to change any such values without input from TIBCO Support.

Java Heap Memory

The *Java heap* is a continuous memory region where all Objects data—instances of class, primitives, and references—is stored. It forms a large part of the *process heap*.

What it does

The **-Xms** option controls the initial size of the Java heap. Properly tuning this parameter reduces the overhead of garbage collection, improving server response time and throughput. For some applications, the default setting for this option might be too low, resulting in a high number of minor garbage collections.

Original value

When you initially configure ActiveMatrix BPM you can select one of the following default values for **-Xms** using the **Minimum Java Heap Size** field in TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

For development systems: 512mb.

For production systems: 1024 mb.



TIBCO recommends that you do not change this value without consulting TIBCO Support.

For the **maximum** heap size, see [Changing the Value of -Xmx](#).

Permanent Heap

Permanent heap memory is where the classes, methods, and similar data are stored. This is controlled by two settings: **-XX:MaxPermSize** is discussed in [Changing the Value of MaxPermSize](#). The other setting is **-XX:PermSize**.

What it does

The **-XX:PermSize** option controls the section of the heap that is reserved for the permanent generation and holds all of the reflective data for the JVM.

Original value

By default, **-XX:PermSize** is set to **4096** mb.

Changing the Value

This size should only be increased to optimize the performance of applications that dynamically load and unload a lot of classes. Setting **-XX:PermSize** to a larger value eliminates the overhead of increasing this part of the heap.



TIBCO recommends that you do not change this value without consulting TIBCO Support.

Thread Stack Size

Each thread in the JVM is assigned a stack. The stack size therefore limits the number of threads that you can have. Too big a stack size and you will run out of memory as each thread is allocated more memory than it needs. If the stack space is too small, eventually you will see an exception **class java.lang.StackOverflowError**.

- **-Xss** This Java option determines the size of the thread stack. You can set it in ActiveMatrix Administrator using the **Java Thread Stack Size** property on the **Configuration** tab for the node. For details, see "Editing a Node" and "Node Configuration Reference" in the *TIBCO ActiveMatrix BPM SOA Administration* guide.

For example, setting **Java Thread Stack Size** to **1** would add the following to the JVM argument string, setting the thread size to 1 megabyte:

```
-Xss1m
```

Garbage Collection

Garbage collection (GC) is a mechanism provided by JVMs to reclaim heap memory space from objects which are taking up that memory but are not currently being used. The following parameters can be set to configure garbage collection.



The terminology used in the following points assumes that you are familiar with the vocabulary of garbage collection. If you are not, consult some of the references listed in [Information Resources](#).

- **-XX:+UseParallelGC** This command minimizes GC pauses by enabling parallel garbage collection for the section of the memory allotted to "new generation" space. The algorithm that it uses is tuned for heap sizes over 10 gigabytes, on multi-CPU machines. Issuing this command on such a multi-processor system can decrease the amount of time that it takes the JVM to complete a partial garbage collection cycle.
- **-XX:ParallelGCThreads** The Java UseParallelGC option uses as many garbage collector threads as there are processors/cores on the system. Use this option to control the number of threads used. For example:

```
-XX:ParallelGCThreads=2
```
- **-XX:+DisableExplicitGC** Some applications make explicit GC calls thinking that it will make their application faster. In some cases this can lead to frequent unnecessary GC calls using up system resources. This Java option disables such explicit calls.
- **-XX:+UseParNewGC** This enables the parallel copying collector. Like the original copying collector, this is a stop-the-world collector. However this collector parallelizes the copying collection over multiple threads, which is more efficient than the original single-thread copying collector for multi-CPU machines (though *not* for single-CPU machines). This algorithm potentially speeds up young generation collection by a factor equal to the number of CPUs available, when compared to the original single-threaded copying collector.
- **-XX:SurvivorRatio** The SurvivorRatio parameter controls the size of the two survivor spaces. For example, **-XX:SurvivorRatio=6** sets the ratio between each survivor space and eden to be 1:6, so that each survivor space will be one-sixth of the young generation. The default for Solaris is 32. If survivor spaces are too small, copying collection overflows directly into the old generation. If survivor spaces are too large, they will be empty. At each GC, the JVM determines the number of

times that an object can be copied before it is tenured, called the *tenure threshold*. This threshold is chosen to keep the survivor space half full.

- **-XX:MaxTenuringThreshold** Allows short-lived objects a longer time period to die in the young generation (and hence, to avoid promotion). A consequence of this setting is that minor GC times can increase because of the additional objects to copy. This value and survivor-space sizes may need to be adjusted so as to balance overheads of copying between survivor spaces and tenuring objects that are going to live for a long time. The default settings are:

```
SurvivorRatio=1024
MaxTenuringThreshold=0
```

These settings cause all survivors of a scavenge to be promoted. This can place a lot of pressure on the single concurrent thread collecting the tenured generation.

Note that if it is used with **-XX:+UseBiasedLocking**, this setting should be 15.

- **-XX:+UseTLAB** This option uses thread-local object allocation.
- **-XX:+UseConcMarkSweepGC** Sets the garbage collector policy to the concurrent (low pause time) garbage collector (also known as CMS).
- **-XX:+CMSClassUnloadingEnabled**
-XX:+CMSPermGenSweepingEnabled If you enable CMSClassUnloadingEnabled the GC will sweep PermGen as well, and will remove classes which are no longer used.
- **-XX:NewSize**
-XX:MaxNewSize The young generation is set by a policy that bounds the size from below by NewSize and bounds it from above by MaxNewSize. As the young generation grows from NewSize to MaxNewSize, both eden and the survivor spaces grow.

Miscellaneous JVM Options

This section contains some Java options that may or may not have an impact on performance of a JVM.

- **-XX:ReservedCodeCacheSize=128m** This is an option for the Java just-in-time compiler. Basically it sets the maximum size for the compiler's code cache. (InitialCodeCacheSize similarly sets the initial size of the same cache.)

The cache can become full, which results in Java warnings like this:

```
CodeCache is full. Compiler has been disabled, Try increasing the code cache
size using XXReservedCodeCacheSize=
```

In extreme cases this may be followed by an `OutOfMemoryError` such as:

```
exception: java.lang.OutOfMemoryError occurred dispatching signal SINT to handler
```

In such cases the VM may need to be forcibly terminated.

In most installations the default value (which depends on the platform) for this option will not need changing, but occasionally a very large workload might produce warnings or errors similar to the above.

- **-server** JVMs based on Sun's Hotspot technology initially compile class methods with a low optimization level. These JVMs use a simple compiler and an optimizing JIT compiler. Normally the simple JIT compiler is used. However you can use this option to use the optimizing compiler instead. This change will significantly increase the performance of the server, but at the cost that the server takes longer to warm up when the optimizing compiler is used.
- **-Xconcurrentio** This is an undocumented option, which generally helps programs with many threads, particularly on Solaris. The main feature turned on with **-Xconcurrentio** is to use LWP-based synchronization instead of thread-based synchronization. This may enable certain applications to speed up significantly.
- Correlation timeout settings:
 - **com.tibco.bx.bpel.bg.pendingMsgProcessor.numMsgs** This option defines how many messages to process at a time. The default (and minimum setting) for this property is 500.

- **com.tibco.bx.bpel.bg.pendingMsgProcessor.interval** This option defines how often to check for expired messages. This interval property can be set as periodic in minutes or daily. For example, it would be **P30** for every 30 minutes, **P45** for 45 minutes and so forth. For a daily setting at 4 PM every day, this can be set as **D16**. The default (and minimum setting) for the interval is 30 minutes.

Information Resources

There are many websites that go into much detail about JVM configuration and how to monitor a JVM.

- Oracle's JVM option guide: <http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html>
- Oracle's white paper on JDBC memory management and JVM configuration: <http://www.oracle.com/technetwork/database/enterprise-edition/memory.pdf>
- Sun's memory management document for JVMs: http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf
- Garbage Collection in the JVM: <http://java.sun.com/docs/hotspot/gc1.4.2/faq.html>
- JVM Garbage Collection in Oracle: <http://onlineappsdba.com/index.php/2008/11/18/jvm-tuning-garbage-collection-in-oracle-apps-11i/>
- Advanced JVM tuning: <http://themindstorms.wordpress.com/2009/01/21/advanced-jvm-tuning-for-low-pause/>
- JVM switches for tuning: <http://performance.netbeans.org/howto/jvmswitches/>

IBM JVM Options

As with other JVMs, the settings used for IBM's JVM on an AIX platform can impact on the performance and stability of the overall system.

For more information, see IBM's JVM tuning guide: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=%2Fcom.ibm.websphere.express.doc%2Finfo%2Fexp%2Fae%2Ftprf_tunejvm.html

Consider the values of the following IBM JVM options:

- **-Xcompressedrefs** (64-bit only) The use of compressed references improves the performance of many applications because objects are smaller, resulting in less frequent garbage collection and improved memory cache utilization. Certain applications might not benefit from compressed references. Test the performance of your application with and without compressed references to determine if they are appropriate. For default option settings, see the JVM command-line options.
See: http://publib.boulder.ibm.com/infocenter/javadoc/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/understanding/mm_compressed_references.html
- **-Xjit:codetotal=262144** Maximum size of memory (KB) allocated for compiled data. See:
[IV26401: REDUCED PERFORMANCE AFTER HEAVY CLASS RE-LOADING](#)
[IZ73689: 64-BIT JAVA RUNS VERY SLOWLY WHEN JIT CODE CACHE IS FULL](#)
- **-Xcodecache32m** This option sets the size of each block of memory that is allocated to store the native code of compiled Java methods. See:
<http://publib.boulder.ibm.com/infocenter/javadoc/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/appendixes/cmdline/xcodecache.html>
- **-Xgcpolicy:gencon** The **gencon** policy uses a concurrent mark phase combined with generational garbage collection to help minimize the time that is spent in any garbage collection pause. This policy is particularly useful for applications with many short-lived objects, such as transactional applications

- **-Xmnx2048m** Sets the maximum size of the new area to the specified value when using **-Xgcpolicy:gencon**.
- **-Xmox2048m** Sets the maximum size of the old (tenure) heap to the specified value when using **-Xgcpolicy:gencon**.
- **-Xlp** Requests the JVM to allocate the Java object heap with large pages.

AIX Settings

You should consider the values of the AIX settings listed in this section.

- **LDR_CNTRL=USERREGS** Specifying the **USERREGS** option tells the system to save all general-purpose user registers across system calls made by an application. This can be helpful in applications doing garbage collection. See:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.prftungd/doc/prftungd/misc_tun_params.htm
- **AIXTHREAD_SCOPE=S** Choose to use a different thread model. Setting the environment variable **AIXTHREAD_SCOPE=S** for that process, we can set the thread model to 1:1. See:
http://swervedba.wordpress.com/2011/05/14/importance-of-aixthread_scope-in-oracle-3/
- **AIXTHREAD_MUTEX_DEBUG=OFF** Maintains a list of active mutexes for use by the debugger. If the program contains a large number of active mutexes and frequently creates and destroys mutexes, this might create higher overhead for maintaining the list of mutexes. Leaving the variable set to **OFF** disables the list. See:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.prftungd/doc/prftungd/thread_supp_tun_params.htm
- **AIXTHREAD_RWLOCK_DEBUG=OFF** Maintains a list of read-write locks for use by the debugger. If the program contains a large number of active read-write locks and frequently creates and destroys read-write locks, this might create higher overhead for maintaining the list of read-write locks. Setting the variable to **OFF** will disable the list. See:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.prftungd/doc/prftungd/thread_supp_tun_params.htm
- **AIXTHREAD_COND_DEBUG=OFF** Maintains a list of condition variables for use by the debugger. If the program contains a large number of active condition variables and frequently creates and destroys condition variables, this might create higher overhead for maintaining the list of condition variables. Setting the variable to **OFF** disables the list. See:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.prftungd/doc/prftungd/thread_supp_tun_params.htm
- **sedmgr -c exempt tibamx_BPMNode-INT** (64-bit only) Exempts the process from the Stack Execution Disable (SED) mechanism. The SED facility is available only in the AIX 64-bit kernel operating systems. See:
<http://www-03.ibm.com/systems/power/software/aix/security/feature/sed.html>
<http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.cmds/doc/aixcmds5/sedmgr.htm>
- **SPINLOOPTIME=128** Controls the number of times to retry a busy lock before yielding to another processor (only for **libpthreads**). The default is **40**. If threads are going to sleep often (lot of idle time), then the **SPINLOOPTIME** value might not be high enough.
- **YIELDLOOPTIME=32** Controls the number of times to yield the processor before blocking on a busy lock (only for **libpthreads**). The processor is yielded to another kernel thread, assuming there is another runnable kernel thread with sufficient priority. If threads are going to sleep often (lot of idle time), then the **YIELDLOOPTIME** might not be high enough.
- **AIXTHREAD_MUTEX_FAST=ON** Enables the use of the optimized mutex locking mechanism. If the program experiences performance degradation due to heavy mutex contention, then setting this

variable to **ON** will force the pthread library to use an optimized mutex locking mechanism that works only on process private mutexes.

JVM Threads

One issue that can affect performance is the number of threads running in the JVM. With mediation it is possible to have thousands of threads running in the JVM, all competing for memory and CPU, as well as increased contention on various mutexes within the JVM and ActiveMatrix.

Large numbers of threads can dramatically increase the memory used by the JVM, especially with database connection pools and prepared statement caching. Depending on the configuration, the Oracle JDBC driver can also use a large amount of memory.

If you see large numbers of threads (for example, more than 500), examine the use of mediation in ActiveMatrix BPM and deployed processes. Also examine other non-BPM applications that have been deployed to the node.

You can decrease the number of Process Engine threads, as explained in [Process Engine \(PVM\)](#).

Consider restricting the number of threads available to httpConnector resources by using a worker thread pool, as explained in [HTTP Connectors](#).

For more information about thread pools, including how to reduce thread numbers, see [Thread Pools](#).

ActiveMatrix BPM Configuration

There are a number of different aspects to the configuration of ActiveMatrix BPM that can impact performance: files, threads, cache sizes, threads, connectors, and so on.

Setting the ActiveMatrix BPM Tuning Options

The various tuning options described are set in different ways.

These include:

- By using the TIBCO Configuration Tool (TCT). This enables you to set system sizing options, depending on the type of system.
- By using ActiveMatrix Administrator. Some tuning options are controlled from this program, which is documented in the *TIBCO ActiveMatrix BPM SOA Administration* guide.
- By setting the values of properties in `.properties` files. For more information, see the *TIBCO ActiveMatrix BPM Administration* guide.

TIBCO Configuration Tool (TCT)

In the TIBCO Configuration Tool (TCT), the Sizing Configuration page of the Create TIBCO ActiveMatrix BPM Server wizard enables you to set values for system sizing, depending on the type of system that you are tuning.

From the **Environment** list, select either **Development** or **Production**. This setting determines the values that are displayed in the other fields on this page. These values are suitable for either a typical development system or a typical production system, respectively.

You can set the following options:

Database Connection Pool Size

The maximum number of database connections available in each of the BPM DataSource resource templates. If the number of engine, client, or EMS driven threads exceed this value, they may have to wait for a database connection to become available (waiting for 30 seconds by default and throwing errors in the log if one wasn't available after waiting). For more information, see [Maximum Connections](#).

Database Statement Cache Size

The number of prepared statements that can be cached by each connection. Caching these statements improves performance but uses more memory. It is unlikely this value needs altering. For more information, see [Prepared Statement Cache Size](#).

Number of Process Engine Threads

The number of PE background threads to be run in the BPM node JVM. This directly affects the throughput of instance starts, continuation and completion. It is likely that these threads will use the most CPU, and there is not much benefit in having more than 27 engine threads configured. For more information, see [Process Engine \(PVM\)](#).

Max Java PermGen Size

The maximum size of the JVM's permanent generation memory space. This is memory outside of the main JVM heap and holds class definitions, associated metadata, and static instances. Depending on the number and type of applications deployed, this value may need altering later through the ActiveMatrix Administrator JVM configuration page. This can be monitored through JConsole, JVisualVM, or the TIBCO JVMInfo command line utility. For more information, see [Changing the Value of MaxPermSize](#).

Java Heap Size (Minimum/Maximum)

The heap is the main memory area used by the JVM. The heap is divided into generations; young holds short-lived objects that are created and immediately garbage collected, and

objects that persist longer are moved to the old generation. Getting the maximum heap size configured correctly is important, and is dependent on various configurations of the system (for example, database connection and thread pool sizes, PE threads, and concurrent clients). This needs to be monitored through JConsole, JVisualVM or the TIBCO JVMInfo command line utility. For more information about the *minimum* heap size, see [Java Heap Memory](#). For more information about the *maximum* heap size, see [Changing the Value of -Xmx](#).

For more information about the fields on this page, see the section about the TIBCO ActiveMatrix BPM Server wizard in the ActiveMatrix BPM installation documentation.

Transaction Log File

The transaction log file stores the information for the distributed transactions managed by transaction manager (TM) within a TIBCO Host instance, so that this data may be used in transaction recovery.

See "Transaction Recovery" in the ActiveMatrix SOA Administration documentation for more information on how transactions are handled.

If the writing of transaction data to this log file is slow, it can have an impact on overall performance, blocking other threads. If you encounter this problem on your system, it is therefore advantageous for the transaction log file to be placed on a separate fast disk, away from the BPM logging, EMS, the operating system, and so on. The faster the disk, and the fewer other applications using it, the better for performance. In order to guarantee that the transaction data is available for recovery in the event of a system failure, this must be a persistent and fault-tolerant device, such as an SSD or a flash drive.

To change the location of the transaction log file to such a device, set the following JVM option within the node's TRA file:

```
-Damf.node.txlogdir="drive:/folder_name"
```

The folder specified in the option, *folder_name*, must exist.

BPM Threads

It is beneficial to keep the number of BPM threads configured low: thread pools, mediation, httpConnectors, and so on.

Too many threads competing for CPU can reduce the actual processing time, with the operating system spending a large amount of time swapping threads in and out.

Sequence Caching

There are several ID sequence caches in ActiveMatrix BPM, maintained by different components of the system.

These caches store unique identifiers for various entities within the system. When a new entity of the appropriate type is required, its ID can be assigned from the cache without needing a call to the database.

The size chosen for a cache can have an impact on system performance. For example, a cache size that is too large will take up resources unnecessarily. But if a cache size is too small, the component will need to call more frequently to the database for a new batch of identifiers.

The sizes of these caches are defined by the values assigned to properties in various BPM properties files. For more information, see the section about BPM Properties Files in the *TIBCO ActiveMatrix BPM Administration* guide, in particular the subsections about the Business Resource Management, Calendar, and Directory Engine properties.

Auditing Levels

ActiveMatrix BPM enables you to configure which of the messages produced by the system are audited: that is, which messages are collected and stored centrally by the Event Collector component.

For more information about levels of auditing, see the section about configuring ActiveMatrix BPM auditing in the *TIBCO ActiveMatrix BPM Administration* guide.

Setting up full auditing will of course mean that many more messages are collected centrally than if you use the default lesser level of auditing provided, with correspondingly more use of system resources. The section about editing the audit rules file in the *TIBCO ActiveMatrix BPM Administration* guide describes how you can tailor the level of auditing to include or exclude individual message IDs. If your monitoring of the system shows that the volume of audit messages is causing a problem, you could try excluding more message IDs. Note that:

- The use of system resources is governed by the number of messages that are audited, not by the number of message IDs listed. One message type that is generated frequently may cause more traffic than a list of a dozen rarely-used types.
- Some messages occur in pairs, one saying that an activity has started, the other saying that it has finished. For example:
 - DAC_COMPONENT_ENT_START — Starting Deadline and Calendar
 - DAC_COMPONENT_EX_START — Deadline and Calendar Started

The first message indicates that a Start operation is beginning, the second that it has been carried out.

In some cases, you might find it useful to exclude the first message of such a pair, thus reducing the number of messages but still being informed when the operation has taken place.

JDBC Resources

There are two types of JDBC resource template, **DataSource** and **DataSourceDirect**. The only difference is that the **DataSource** templates use **XA** transaction co-ordination for the transactions performed with the database, while the **DataSourceDirect** templates do not.



When you reconfigure the JDBC resource templates you **must** restart the BPM node.

There are three resource templates (and by default one resource instance for each template):

- **DataSourceBDS** Used by Business Data Services (BDS) to persist case data to the case data store via an XA-based transaction.
- **DataSource** Used by all other BPM components for database operations required when processing work items or organizational information within an XA transaction.
- **DataSourceDirect** Used by all other components for database operations required when processing work items or organizational information within a non-XA transaction.

Use of XA transactions in TIBCO ActiveMatrix BPM 4.0 has been significantly reduced (providing improved system performance). As part of this change, the following JDBC resource templates are deleted as part of an upgrade from a pre-4.0 BPM system:

- **DataSourcePE** Used by Process Engine (PE) for database operations required when starting and processing process instances within an XA transaction.
- **DataSourceEC** Used by Event Collector (EC) to persist audit events to the database via an XA-based transaction.
- **DataSourceDirectPE** Used by Process Engine for database operations required when starting and processing process instances within a non-XA transaction.
- **DataSourceDirectEC** Used by Event Collector (EC) to persist audit events to the database via a non-XA based transaction.



Process Engine and Event Collector components now share the use of the common DataSource and DataSourceDirect database connection pools with other BPM components, instead of having their own pools. An end-to-end item of work is now processed as a single transaction, no matter what components are involved.

If you have upgraded from a pre-4.0 BPM system, you should review and possibly increase the [Maximum Connections](#) or [Prepared Statement Cache Size](#) values for the DataSource and DataSourceDirect resource templates to take account of this, in line with the performance of your upgraded system. As there are now less connection pools you may need to increase the number of connections per pool. See *TIBCO ActiveMatrix BPM Installation and Configuration* for more information about how to do this.

Maximum Connections

This setting controls the maximum number of database connections within the pool for the data source template.

Original value

When you initially configure ActiveMatrix BPM you can select one of the following default values for the **Database Connection Pool Size** field in TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

For development systems: 10 threads

For production systems: 50 threads

Monitoring

Monitor your system's performance, and if you find that BPM is running slowly you may wish to consider increasing this value.

Changing the value

Too small a value for the pool size, compared with the number of threads using it, will result in threads waiting for a connection to become available. This wait time is up to 60 seconds, and so can have big impact on performance. However too large a value may use up too much memory within the JVM, especially if the [Prepared Statement Cache Size](#) is configured.

For information about setting the value of this option, see [TIBCO Configuration Tool \(TCT\)](#).

Prepared Statement Cache Size

When executing SQL statements against the database there are typically two phases required: prepare and execute. The *prepare* phase parses, compiles, and performs query optimizations on the SQL

statement. The *execute* phase binds values to the parameters of the statement, and then the database management system executes the statement.

Having a prepared statement cache means that the prepare phase is reduced or eliminated altogether when performing SQL statements, reducing the number of calls to and time within the database. However, it uses more memory.

What it does

The **Database Statement Cache Size** defines the number of prepared database statements that will be cached by *each* connection in the DataSource resource template pool.

Original value

When you initially configure ActiveMatrix BPM you can select one of the following default values for the **Database Statement Cache Size** field in TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

For development systems: 0 statements

For production systems: 100 statements

Changing the value

The larger the prepared statement cache, the more SQL statements that can use it and the greater the benefit. But each area makes a finite number of SQL statements, so having a very large value, such as 200, will be counterproductive since it will be using more memory within the JVM.

Too small a value, such as **10**, and prepared statements will continually be swapped out of the cache in order to make space for others.

The default values provided for the **Database Statement Cache Size** parameter in TCT should be satisfactory for most systems, depending on the memory available to the JVM. If there are issues with JVM memory then you may consider turning this cache off by setting a value of **0**. (This is the default setting for a development system.)

Turning off JDBC Connection Validation

By default, the ActiveMatrix platform validates a JDBC connection before executing a SQL statement over that connection. On large systems this validation can have a significant performance impact, and a performance benefit can be obtained by disabling the platform's JDBC connection validation.

To turn off JDBC connection validation, make the following configuration changes to **only** those JDBC resource templates that have been configured for XA transaction co-ordination (see [JDBC Resources](#)):

Procedure

1. In ActiveMatrixAdministrator, select **Shared Objects > Resource Templates**.
2. Select in turn each of the JDBC resource templates that is identified in the **Description** column as an **XA DataSource**.
3. On the **Advanced** tab, go to **Host Type Properties** and click **Add**.
4. In the **Name** column, enter **validateConnectionOnAcquire**. In the **Value** column, enter **false**.
5. In the same way, add **connectionPoolValidateConnection** and set it to **false**. (If **true**, this property causes connections to be validated regardless of how **validateConnectionOnAcquire** is set.)
6. Click **Save**.

Deallocating Unused JDBC Connections

By default, the ActiveMatrix platform validates a JDBC connection before executing a SQL statement over that connection. On large systems this validation can have a significant performance impact, and a performance benefit can be obtained by deallocating any unused connections.

To deallocate unused connections make the following configuration changes to **only** those JDBC resource templates that have been configured for XA transaction co-ordination (see [JDBC Resources](#)):

Procedure

1. In ActiveMatrixAdministrator, select **Shared Objects > Resource Templates**
2. Select in turn each of the JDBC resource templates that is identified in the **Description** column as an **XA DataSource**.
3. Select the **Advanced** tab.
4. Under **Host Type Properties**, set **POOL_IDLE_TIMEOUT** to a suitable low value, such as **1** (meaning one minute).
5. Set **POOL_MIN_SIZE** to **0**.
6. Click **Save**.

Result

This procedure internally deallocates any connection that has been idle for more than one minute. This should be quicker than any firewall or database setting, so it should never result in any invalid connections. However if your connection pool does often have one-minute gaps in activity, this will result in a lot of connection allocation/deallocation overhead. You may need to experiment with suitable values.

HTTP Connectors

HTTP Connector resource templates (by default there is one named **httpConnector**) control the number of threads that are available to process client requests. The default is 20.

For guidance on this value in the template, see the information about the Acceptor Threads property in the section about HTTP Connector in the *TIBCO ActiveMatrix BPM SOA Administration* guide.

There is only a single resource instance for the BPM node, so with a large number of users connected and performing work the default value may be inadequate. The number of threads in this resource template need not be the same as the number of clients performing work. Testing the system and monitoring client call times will give a good indication of whether this needs to be changed.

Again, as for all thread configuration options, be aware that more threads will require more JVM memory and will impact other threads through increased CPU requirements.

If you have large numbers of users processing work items, the number of **worker threads** that are required to process client requests will also be high, and can exceed the number of database connections available. This results in failures to perform the required call. Since the thread will wait for some time when no available connections are found, clients attempting to connect will also experience timeouts.

If you encounter this issue, you can create a new thread pool and configure the HTTP Connector resource template to use it. Configure the thread pool to have a maximum number of threads that is **less** than the **Maximum Connections** value defined on the associated JDBC DataSource resource templates. User request threads will not then be waiting for database connections, since only the limited number of threads will be created.

To do this:

Procedure

1. Open ActiveMatrix Administrator.
2. Select **Shared Objects > Resource Templates**.
3. Select the appropriate HTTP Connector resource template and click the **Advanced Configuration** tab.
4. Click the **new** link by the **Worker Thread Pool** field. The **Add Resource Template** dialog displays. Note that the **Type** is set to **Thread Pool**.
5. Enter a **Name** and optionally a **Description**.
6. Set the **Max Pool Size** to your required value.
7. Click **Save**.

Memory Usage

Without a thread pool on HTTP connectors there is also the possibility of excessive heap usage, causing potential “out of memory” errors, when clients are performing requests.

There can be sharp growth in heap usage whilst calls are being performed up until the JVM decides to perform garbage collection. Depending on the amount of heap to be freed the garbage collection may cause a “stun” of the virtual machine for a short while, impacting clients and throughput of the BPM node.

This is caused by excessive object cycling due to the use of new threads for client calls rather than re-using threads from a pool. Using a pool of threads avoids thread-specific data held in various different packages being created and then thrown away for a single client call to TIBCO ActiveMatrix BPM.

For more information on thread pools and how they can be configured, see [Thread Pools](#)

HTTP Client

HTTP client resource templates (RT) are used by web service calls, whether with mediation or an external system. Each RT has a maximum total connections per host and overall. The default values are 2 and 20, respectively.

The screenshot shows the 'HTTPClient' configuration window with the 'Advanced Configuration' tab selected. The window has tabs for 'General', 'SSL', 'Advanced Configuration', 'Proxy Configuration', and 'Resource Instances'. The 'Advanced Configuration' tab contains several settings:

- Accept Redirect (optional)**: ☐ Yes
- Reuse Address (optional)**: ☐ Yes
- Disable Connection Pooling (optional)**: ☐ Yes
- Suppress TCP Delay (optional)**: ☒ Yes
- Stale Check (optional)**: ☐ Yes
- Buffer Size (B) (optional)**:
- Connection Retrieval Timeout (ms) (optional)**:
- Local Socket Address (optional)**:
- Maximum Total Connections (optional)**:
- Maximum Total Connections per Host (optional)**:

At the bottom of the dialog are three buttons: 'Save', 'Revert', and 'Restore Default'.

If there are a lot of external calls being made, the default values may need to be modified.

Background Processing

Much of the work performed by an ActiveMatrix BPM system is performed in the background, such as starting and processing instances, scheduling work items, persisting events, and gathering statistics.

There are four main ActiveMatrix BPM components that perform background work:

- Process Manager (PM)
- Business Resource Management (BRM)
- Event Collector (EC)
- Statistics Collector (SC)

When users have finished using the system or if testing has finished, processing may still be being performed by any of these areas due to a backlog of work having built up over time. This backlog of work can be monitored during and after client work is performed to see whether ActiveMatrix BPM is keeping up with the work load being put through, or if the backlog is growing.

You can configure the thread pools that are associated with some of these components, as explained in [Thread Pools](#).

Process Manager (PM)

PM performs the starting, continuation, and completion of process instances. It can be driven from a number of sources such as client requests or asynchronous EMS queues.

The main way to monitor the backlog of PM work is to run the following query:

```
SELECT COUNT(1) FROM PVM_REQUEST_QUEUE WHERE status=0;
```

This returns all rows that have been scheduled but have yet to be picked up by the worker threads.

Business Resource Management (BRM)

BRM is involved in the scheduling, processing, and completion of work items, along with retrieving and processing work lists.

Most BRM background work is scheduling, rescheduling and cancelling work items that are generated by PM from user tasks in the process flow. Any backlog in work is held in the asynchronous EMS queues. For example, the following EMS queue contains requests to schedule, reschedule, and cancel work items:

```
AMX_SV.2.7.WorkManager.implementation.brm.S.AsyncWorkItemSchedulerService
```

Messages posted to this queue are processed by the `amx.bpm.userapp.threadpool`. For more information, see [Thread Pools](#).

Event Collector (EC)

EC is involved in the persisting of audit events generated mainly from PM and BRM.

These events are asynchronous and are held in an EMS queue. For example:

```
AMX_SV.2.7.EventCollector.S.EventCollectorAuditService
```

Messages posted to this queue are processed by the `amx.bpm.userapp.threadpool`. For more information, see [Thread Pools](#).

Statistics Collector (SC)

SC is driven from the EC audit events, generating process instance and work item statistics that can be used to run reports against.

The SC backlog is contained in the database, and the following query can be used to determine its size:

```
SELECT COUNT(1) FROM EC_EVENT_TRIGGER;
```

Background threads in the BPM node generate these statistics. The batch size used when these threads process triggers is controlled through the `BRM.sc.properties` configuration file. For more information about properties files, see the *TIBCO ActiveMatrix BPM Administration* guide.

Thread Pools

The ActiveMatrix BPM application contains a number of different thread pools that perform different functions within the BPM system.

Apart from the Process Engine (PVM) thread pool, which can be configured individually, each thread pool is configured collectively with a minimum and maximum number of threads by setting the **amx.bpm.userapp.threadpool** in the **Thread Pool** resource template. In general all the thread pools serve to limit the number of threads for EMS messages to a level that the system can handle comfortably. In many cases, the default values used by the system as delivered will prove to be satisfactory.

Depending on the system work load, sometimes too many of these threads are running and using excessive CPU and memory. However, most of the time, the number of these threads running in parallel is quite low. You can view these threads in a thread dump of the JVM (jstack) or through JConsole or JVisualVM.

Note that these threads also consume database connections, so these threads could exceed the maximum number of configured database connections, especially when taking into account the HTTP connector threads running.

Process Engine (PVM)

The Process Engine background thread pool contains the main process instance processing threads.

Original value

The **Number of Process Engine Threads** option specifies how many threads can be started in this pool. It also governs the rate at which new work items are scheduled (though not directly since they are scheduled asynchronously) going through the EMS system.

When you initially configure ActiveMatrix BPM you can select one of the following default values for the **Number of Process Engine Threads** field in TIBCO Configuration Tool - see [TIBCO Configuration Tool \(TCT\)](#).

For development systems: 5 threads

For production systems: 18 threads

Monitoring

Monitor your system's performance, and if you find any of the message queues getting too large you may wish to consider increasing this value.

Changing the value

You can set this value by using the **Number of Process Engine Threads** option in the TCT, or directly by setting the **pe.engineThreadCount** substitution variable on the BPM application (by default, **amx.bpm.app**).

The more threads configured for the Process Engine background thread pool, the more process instances that can be started, processed, and completed in parallel. However more threads put strain on other resources. Too many engine threads can, under heavy load, swamp the CPU, reducing performance of areas such as processing of user tasks, client processing of work items or pageflows. If this is the case, consider reducing the number of engine threads or, if possible, moving PE to a different BPM node on a different physical machine.

Too many engine threads within the same BPM node may not increase throughput when there is no restriction on the CPU. Testing of a straight through processing (STP) process has shown that more than 27 engine threads do not improve performance. A better approach is to add a new BPM node on the same machine, system resources permitting. The main reason for this is contention within platform,

which is why we recommend using a number of small BPM nodes on the same machine, rather than one large BPM node.

As a general guide, most systems are likely to see the best performance with a value greater than 5.

User Application (UserApp)

These threads process requests sent, via asynchronous EMS messages, to any customer application from other components in the BPM system.

Original value

By default the core thread pool size is **20**, with a maximum of **50** threads.

Monitoring

Monitor your system's performance, and if you find that BPM is running slowly you may wish to consider increasing this value.

Changing the value

You can set this value directly by setting the **amx.bpm.userapp.threadpool** in the **Thread Pool** resource template.

Reducing the number of threads within this pool may impact the speed at which user applications operate. Increasing the number of threads may increase the speed at which user applications run, but the extra CPU usage required can result in impacting the performance of other areas.

BPM Filtering and Sorting of Work Lists and Audit Entries

When installed the BPM database is created to provide a system balanced for both execution and query/filter performance; however some installations that make heavy use of filtering in TIBCO ActiveMatrix BPM may benefit from either tuning of existing database indexes or the creation of new indexes. Examining database reports from representative loads indicates whether new indexes are required, or updates to existing ones.

Work Item Ordering and Filtering

Work items can be filtered and ordered on any number of columns that are present on the **brm_work_items** table. On a standard installation only the most commonly queried database columns on this table are indexed. If Work List ordering and filtering is being applied to columns that aren't indexed, for example listing all work items that have **attribute19** greater than a specified date, then both filter and overall system performance may benefit if the database administrator adds an index to the **attribute19** column.

Audit Trail Querying

The **ec_event** indexes are configured at install time to benefit queries used by built-in functionality, such as the Graphical Audit Trail. Custom queries carried out either via Event Viewer or custom applications may require additional indexes to perform optimally. It is also important to consider the contents of the **ec_event** view, as different uses of AMXBPM can result in different types of data being more, or less prevalent in the **ec_event** view, for this reason it is important to look at both the query and the index being used to obtain the required data.

BPM Nodes

Another aspect of ActiveMatrix BPM configuration that has an effect on performance is the number and size of BPM nodes.

Adding multiple BPM nodes gives better throughput compared with a single node. This highlights the extra contention within a single JVM as the number of threads is increased. Using multiple, smaller

BPM nodes makes better use of the system's CPU and memory, as well as reducing any contention within a single JVM.

If load balancing is being used for client access to these ActiveMatrix BPM nodes, whether on the same machine or on separate ones, if one node crashes or the host machine goes down, the clients can still access a node and process work.

Configuring How Often the Process Engine Looks for Work Requests

The Process Engine maintains a queue of work requests to process. Depending on the number of items in the queue, the engine retrieves further requests from the PVM_REQUEST_QUEUE database table. Although the settings that control the frequency of this retrieval are suitable for most installations, you may want to change them to increase efficiency.

Procedure

- Use ActiveMatrix Administrator to update the Java Virtual Machine (JVM) configuration for the ActiveMatrix BPM nodes, as explained in the *TIBCO ActiveMatrix BPM SOA Administration* guide. Add (and set) one or more of the following properties on the **JVM Configuration** page within the **Configuration** tab:

Property	Description
<code>com.tibco.bx.groupRQ.batchSize</code>	The number of new requests to retrieve from the PVM_REQUEST_QUEUE table at a time. Default: 25.
<code>com.tibco.bx.groupRQ.fillLevel</code>	The maximum number of requests in the request queue. When this number is reached, the engine stops checking the PVM_REQUEST_QUEUE table for new requests. Default: 50.
<code>com.tibco.bx.groupRQ.lowWaterLevel</code>	The minimum number of requests in the request queue. When this number is reached, the engine starts checking the PVM_REQUEST_QUEUE table for new requests. Default: 10.
<code>com.tibco.bx.groupRQ.idleSleep</code>	If there are no requests in the PVM_REQUEST_QUEUE table, this is the amount of time (in ms) that the engine sleeps before it checks the table again for new requests. Default: 100.

Locking of Signal Definitions

Locking only occurs for signal definitions with a correlation timeout. Both first initialization of the event handler and the persistent signal creation are locked on the signal definition.

This behavior is configurable (similar to message locking) - if `com.tibco.bx.lockSignal` system property is set to true (default value), then finer grain control can be configured by setting `com.tibco.bx.lockPerSignal` property to true.

If locking per signal is enabled, then you can configure each signal application's locking strategy by setting a corresponding property in the form of `com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion`.

If the application level property is not set, then you can configure each signal definition's locking strategy by setting a corresponding property in the form of `com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion.signalName`.

Expired Signals Job

Background job frequency can be configured with the `com.tibco.bx.bg.globalSignalProcessor.interval` JVM property. This property specifies how frequently expired global signals are cleaned up.

This is specified as either of the following:

- P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes the cleanup job to run every 60 minutes.
- D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes the cleanup job to run daily at 8 PM.

The default is P30.

Batch size for this job is configured with the `com.tibco.bx.bg.globalSignalProcessor.numMsgs` JVM property. This specifies how many expired signals should be processed at a time. The default value is 500.

Maximum Listener Count

A maximum of 100 global signal event handlers can listen on the same signal definition with the same correlation. This is configurable via `com.tibco.bx.maxGlobalSignalListenerCount` property.

If an event handler attempts to register itself after the limit is reached, an exception with error code 905025 is thrown: There are more than [{0}] listeners with correlation [{1}] for global signal definition [{2}].

Configuring the Maximum Number of Tasks per Process Instance

It is possible for a runaway process to generate a very large number of tasks within a single instance, resulting in a large number of EC events for the instance, as well as a large number of rows in the PVM database tables for the instance.

You can configure a threshold for the number of tasks per instance, at which point the process is halted, and the following is written to the BPM log file:

```
[WARN] - {MAX_TASKS_PER_INSTANCE_REACHED} - Instance [pvm:xxxxx] has reached the
maximum number of tasks allowed: [1,000].
```

And the following audit message is generated:

```
BX_INSTANCE_PROCESS_MAX_TASKS_REACHED Maximum number of tasks per instance reached.
```

The number of tasks threshold (default = 1000) can be configured using the `com.tibco.bx.maxTasksPerInstance` JVM property. You may want to configure this threshold if:

- your process design contains a large number of loops or tasks.
- you have long-running processes, whose task count may get large even though the process isn't running away.

If you set a high value, your ActiveMatrix BPM system could stall if process instances reach a runaway state, as it will take time to reach the threshold. If you know your task per instance count, you may wish to set the threshold to a lower value.

Procedure

1. Log in to TIBCO ActiveMatrix Administrator as `root` user and select **Infrastructure > Nodes**.
2. Select the node **BPMNode** and click the **Configuration** tab.

3. Click the **JVM Configuration** link.
4. Under the **Properties** heading, click **Add** to add a new JVM property.
5. Enter the property name as `com.tibco.bx.maxTasksPerInstance` and set the value as desired.
6. Click **Save** to save the changes.
7. Stop and restart **BPMNode**.

Configuring Cleanup of the PVM_WORK_ITEM or PVM_REQUEST_QUEUE Database Tables

When Process Manager processes/completes a row in the PVM_WORK_ITEM or PVM_REQUEST_QUEUE database tables, it sets that row's STATUS value to 2 (DONE). By default, those rows are immediately and automatically removed from the tables when Process Manager has completed processing them. Therefore, using the default behavior, you will not see rows in the PVM_WORK_ITEM or PVM_REQUEST_QUEUE database tables with a status of 2 (DONE).

However, you can configure the Process Manager so that it does not remove the rows automatically when processing of the row is done. This is done using the following JVM properties:

- `com.tibco.pvm.deleteWorkItem` - Set this property to "false" to prevent rows in the PVM_WORK_ITEM table from being removed automatically upon completion of a row. Conversely, set the property to "true" (or remove the property) to revert to default behavior.
- `com.tibco.pvm.deleteRequestQueueItem` - Set this property to "false" to prevent rows in the PVM_REQUEST_QUEUE table from being removed automatically upon completion of a row. Conversely, set the property to "true" (or remove the property) to revert to default behavior.

If you use the properties shown above to prevent automatic removal of completed rows, you can use the JVM properties listed below to configure background cleanup of completed rows:

- `com.tibco.bx.bg.workItemCleanup.batchCount` - The number of batches to process, in the background, to remove completed rows from the PVM_WORK_ITEM table. The background process stops when this count is reached, or the number of completed rows is less than the batchSize (see below). A value of -1 causes all completed rows to be removed.
Default = -1
- `com.tibco.bx.bg.workItemCleanup.batchSize` - The number of completed rows to remove from the PVM_WORK_ITEM table in each batch.
Default = 500 (minimum is 250)
- `com.tibco.bx.bg.workItemCleanup.interval` - The frequency to perform the background cleanup of the PVM_WORK_ITEM table. Specified as either of the following:
 - P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes the cleanup job to run every 60 minutes.
 - D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes the cleanup job to run daily at 8 PM.
 Default = P30 (minimum is P15)
- `com.tibco.bx.bg.requestQueueCleanup.batchCount` - The number of batches to process, in the background, to remove completed rows from the PVM_REQUEST_QUEUE table. The background process stops when this count is reached, or the number of completed rows is less than the batchSize (see below). A value of -1 causes all completed rows to be removed.
Default = -1
- `com.tibco.bx.bg.requestQueueCleanup.batchSize` - The number of completed rows to remove from the PVM_REQUEST_QUEUE table in each batch.

Default = 1000 (minimum is 500)

- `com.tibco.bx.bg.requestQueueCleanup.interval` - The frequency to perform the background cleanup of the PVM_REQUEST_QUEUE table. Specified as either of the following:
 - P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes the cleanup job to run every 60 minutes.
 - D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes the cleanup job to run daily at 8 PM.

Default = P30 (minimum is P1)

Note that if you elect to not use the automatic deletion of completed rows, and the number of such rows grows very large, performance may be impacted when the background cleanup job is processing, particularly if you've configured a large batch size.

JVM Properties Reference

Properties can be passed to the Java Virtual Machine (JVM) to affect the behavior of ActiveMatrix BPM.

JVM properties are set on the BPM node using ActiveMatrix Administrator. For more information, see [Configuring JVM Properties on the BPMNode](#).

The following table lists all of the JVM properties that can be set as a JVM argument using ActiveMatrix Administrator. Some of these properties are described in more detail in other topics. For those, links or citations are provided in the description.

JVM Property	Description
<code>com.tibco.bx.allow.null.msgpart</code>	<p>Controls whether an exception is thrown if a required parameter is not initialized (that is, it is null) on a web service call.</p> <ul style="list-style-type: none"> • true (default) - an exception is not thrown for a null message part. • false - an exception is thrown for a null message part.
<code>com.tibco.bx.arrayDelimiter</code>	<p>Specifies the delimiter between array elements provided in a <code>setAvailableProcessInstanceVariables</code> request. The default value for the delimiter is ", " (comma followed by a space) if this property is not set.</p> <p>For more information, see "SOAP API - <code>setAvailableProcessInstanceVariables</code>" in the <i>TIBCO ActiveMatrix BPM Developer's Guide</i>.</p>
<code>com.tibco.bx.autoDelete</code>	<p>Specifies whether process instances are automatically deleted from the database upon completion.</p> <ul style="list-style-type: none"> • yes - automatically delete process instances that are complete. • no - do not automatically delete process instances. <p>Also see com.tibco.bx.bpel.maintenance.purgeProcess.interval.</p> <p>Default = no</p>


JVM Property	Description
com.tibco.bx.autoDeleteFailedProcesses	<p>Specifies whether failed process instances are automatically deleted from the database.</p> <ul style="list-style-type: none"> • yes - automatically delete failed process instances. • no - do not automatically delete failed process instances. <p>For more information, see "Configuration of Error Handling Behavior for Process Instances" in the <i>TIBCO ActiveMatrix BPM Administration</i> guide.</p> <p>Default = no</p>
com.tibco.bx.bg.globalSignalProcessor.interval	<p>Specifies how frequently expired global signals are cleaned up. Specified as either of the following:</p> <ul style="list-style-type: none"> • P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes the cleanup job to run every 60 minutes. • D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes the cleanup job to run daily at 8 PM. <p>Default = P30</p> <p>Also see Expired Signals Job.</p>
com.tibco.bx.bg.globalSignalProcessor.numMsgs	<p>The batch size for expired global-signal cleanup jobs, that is, the number of expired signals that are to be processed at one time.</p> <p>Default = 500</p> <p>Also see Expired Signals Job.</p>
com.tibco.pvm.deleteWorkItem com.tibco.bx.bg.requestQueueCleanup.batchCount com.tibco.bx.bg.requestQueueCleanup.batchSize com.tibco.bx.bg.requestQueueCleanup.interval	<p>By default, rows in the PVM_REQUEST_QUEUE database table are removed when Process Manager has completed processing them.</p> <p>These properties can be used to override this default behavior. For details, see Configuring Cleanup of the PVM_WORK_ITEM or PVM_REQUEST_QUEUE Database Tables</p>
com.tibco.pvm.deleteRequestQueueItem com.tibco.bx.bg.workItemCleanup.batchCount com.tibco.bx.bg.workItemCleanup.batchSize com.tibco.bx.bg.workItemCleanup.interval	<p>By default, rows in the PVM_WORK_ITEM database table are removed when Process Manager has completed processing them.</p> <p>These properties can be used to override this default behavior. For details, see Configuring Cleanup of the PVM_WORK_ITEM or PVM_REQUEST_QUEUE Database Tables</p>

JVM Property	Description
<code>com.tibco.bx.bpel.bg.moduleCleanup.interval</code>	<p>Used to clean up any modules from the database that have been marked for deletion. For example, the last instance of a module has been migrated or completed so that the module can now be removed. Specified as either of the following:</p> <ul style="list-style-type: none"> • P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes the cleanup job to run every 60 minutes. • D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes the cleanup job to run daily at 8 PM. <p>Default = P10 (minimum is P5)</p>
<code>com.tibco.bx.bpel.bg.pendingMsgProcessor.interval</code>	<p>Specifies how frequently pending messages are cleaned up. Specified as either of the following:</p> <ul style="list-style-type: none"> • P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes the cleanup job to run every 60 minutes. • D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes the cleanup job to run daily at 8 PM. <p>Default = P30</p>
<code>com.tibco.bx.bpel.bg.pendingMsgProcessor.numMsgs</code>	<p>The batch size for pending-message cleanup jobs, that is, the number of expired signals that are to be processed at one time.</p> <p>Default = 500</p>
<code>com.tibco.bx.bpel.bg.unclaimedMsgProcessor.interval</code>	<p>If system-wide locking is disabled (<code>com.tibco.bx.lockOperation=false</code>), this property specifies how frequently a background job checks for, and processes, unclaimed messages. Specified as either of the following:</p> <ul style="list-style-type: none"> • P# - Where "P" indicates "periodic" and # is the number of minutes. For example, P60 causes it to check every 60 minutes. The lowest valid value is P1. • D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes it to check daily at 8 PM. <p>Default = P30</p> <p>For more information, see Configuring Locking Operation on Retry When Receiving/Retrieving a Message.</p>

JVM Property	Description
<code>com.tibco.bx.bpel.bg.unclaimedMsgProcessor.numMsgs</code>	<p>If system-wide locking is disabled (<code>com.tibco.bx.lockOperation=false</code>), this property controls the maximum number of unclaimed messages that will be picked up per execution. The minimum is 50.</p> <p>Default = 500</p> <p>For more information, see Configuring Locking Operation on Retry When Receiving/Retrieving a Message.</p>
<code>com.tibco.bx.bpel.maintenance.purgeProcess.interval</code>	<p>Used to automatically delete completed process instances, based on time, as follows:</p> <ul style="list-style-type: none"> • P# - Where "P" indicates "periodic" and # is a number of minutes. For example, P60 causes completed process instances to be deleted every 60 minutes. The lowest valid value is P60. • D# - Where "D" indicates "daily" and # is the hour number (1-24). For example, D20 causes completed process instances to be deleted daily at 8 PM. <p>Default = D1 (daily at 1 AM)</p> <p>Also see com.tibco.bx.autoDelete.</p>
<code>com.tibco.bx.engine.checkParms</code>	<p>Prior to version 2.2.0 of ActiveMatrix BPM, if a process called a reusable sub-process, and the sub-process had mandatory parameters, the system was not checking to ensure that the parameters contained a value. The sub-process would start without error. However, it could result in an error later in the sub-process.</p> <p>The issue described above was resolved in version 2.2.0. Mandatory parameters are now checked to ensure they are not null. If the main process calls a sub-process, and there are mandatory input parameters to the sub-process, the sub-process is not started.</p> <p>The <code>com.tibco.bx.engine.checkParms</code> property can be set to "no" to override this stricter enforcement of mandatory parameters. This can be used if you are using a pre-2.2.0 application that worked fine previously, but has issues with the stricter enforcement of mandatory parameters.</p>
<code>com.tibco.bx.engine.clearInMemoryTrackers</code>	<p>If true, an in-memory instance row is cleared from the PVM_INMEM_PROC_TRACKER table when the sub-process task that started it has completed.</p> <p>Default = true</p>
<code>com.tibco.bx.engine.recovery.failure.threshold</code>	<p>The number of seconds after which a process engine is considered failed and needs to be recovered.</p> <p>Default = 30</p>

JVM Property	Description
com.tibco.bx.engine.threadPool.size	<p>Specifies the number of PE threads in the pool for processing work. Note that this property setting overrides the engineThreadCount value set on the amx.bpm.app in ActiveMatrix Administrator.</p> <p>Default = 5</p>
com.tibco.bx.filterByOrgs	<p>If a resource is mapped to more than six organizations, and that resource starts a process instance, BX_TOO_MANY_ORG_IDS is written to the BPM.log. This property specifies whether BX_TOO_MANY_ORG_IDS is shown as a warning or an error. The valid settings are:</p> <ul style="list-style-type: none"> no (default) - BX_TOO_MANY_ORG_IDS is shown as a warning. yes - BX_TOO_MANY_ORG_IDS is shown as an error.
com.tibco.bx.groupRQ.batchSize	<p>Controls number of rows from PVM_REQUEST_QUEUE to read in a batch.</p> <p>Default = 25</p> <p>Also see Configuring How Often the Process Engine Looks for Work Requests.</p>
com.tibco.bx.groupRQ.fillLevel	<p>Maximum number of items in the in-memory queue; will read batchSize items (see com.tibco.bx.groupRQ.batchSize) until this is reached.</p> <p>Default = 50</p> <p>Also see Configuring How Often the Process Engine Looks for Work Requests.</p>
com.tibco.bx.groupRQ.idleSleep	<p>Number of milliseconds to sleep before trying again when no rows are found.</p> <p>Default = 100</p> <p>Also see Configuring How Often the Process Engine Looks for Work Requests.</p>
com.tibco.bx.groupRQ.lowWaterLevel	<p>The number of items in the in-memory queue, at which point the "PVM:DB Work Queue" is re-started.</p> <p>Default = 10</p> <p>Also see Configuring How Often the Process Engine Looks for Work Requests.</p>

JVM Property	Description
<code>com.tibco.bx.lockOperation</code>	<p>Specifies whether system-wide locking is enabled:</p> <ul style="list-style-type: none"> • <code>true</code> (default) - system-wide locking is enabled. • <code>false</code> - system-wide locking is disabled. <p>Also see com.tibco.bx.bpel.bg.unclaimedMsgProcessor.interval and com.tibco.bx.bpel.bg.unclaimedMsgProcessor.numMsgs.</p> <p>For more information, see Configuring Locking Operation on Retry When Receiving/Retrieving a Message.</p>
<code>com.tibco.bx.lockOperation.ModuleName</code>	<p>This is used in conjunction with com.tibco.bx.lockPerOperation to specify locking at the module level. If <code>com.tibco.bx.lockPerOperation=true</code>, this property can be set to <code>true</code> (to enable locking) or <code>false</code> (to disable locking) for the module specified in <i>ModuleName</i>.</p> <p>If this property is set (either <code>true</code> or <code>false</code>), it takes precedence over the <code>com.tibco.bx.lockOperation.ModuleName.PortType.OperationName</code> (see below), if specified.</p> <p>For more information, see Configuring Locking Operation on Retry When Receiving/Retrieving a Message.</p>
<code>com.tibco.bx.lockOperation.ModuleName.PortType.OperationName</code>	<p>This is used in conjunction with com.tibco.bx.lockPerOperation to specify locking at the operation level. If <code>com.tibco.bx.lockPerOperation=true</code>, this property can be set to <code>true</code> (to enable locking) or <code>false</code> (to disable locking) for the operation specified in <i>OperationName</i>. The <i>PortType</i> must be in the form <code>{NamespaceURI}localPort</code>.</p> <p>Note that if the <code>com.tibco.bx.lockOperation.ModuleName</code> property (see above) is also specified, it takes precedence over this property.</p> <p>For more information, see Configuring Locking Operation on Retry When Receiving/Retrieving a Message.</p>
<code>com.tibco.bx.lockPerOperation</code>	<p>If system-wide locking is enabled (that is, <code>com.tibco.bx.lockOperation=true</code>), this property can be used to enable finer-grained control over locking. Set this property to <code>true</code>, then use either <code>com.tibco.bx.lockOperation.ModuleName</code> or <code>com.tibco.bx.lockOperation.ModuleName.PortType.OperationName</code> to specify locking at the module- or operation-level.</p> <p>For more information, see Configuring Locking Operation on Retry When Receiving/Retrieving a Message.</p>

JVM Property	Description
<code>com.tibco.bx.lockPerSignal</code>	<p>If signal definition locking is enabled (by setting <code>com.tibco.bx.lockSignal=true</code>), this property can be used to enable finer-grained control over locking. Set this property to true, then use either <code>com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion</code> or <code>com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion.signalName</code> to enable signal locking at the application- or signal-level.</p> <p>For more information see, Locking of Signal Definitions.</p>
<code>com.tibco.bx.lockSignal</code>	<p>Specifies whether locking is enabled on signal definitions.</p> <p>Default = true</p> <p>For more information see, Locking of Signal Definitions.</p>
<code>com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion</code>	<p>If locking per signal is enabled (by setting <code>com.tibco.bx.lockPerSignal=true</code>), this property can be used to enable signal locking at the application level, as follows:</p> <p><code>com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion</code></p> <p>For more information see, Locking of Signal Definitions.</p>
<code>com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion.signalName</code>	<p>If locking per signal is enabled (by setting <code>com.tibco.bx.lockPerSignal=true</code>), this property can be used to enable signal locking at the signal level, as follows:</p> <p><code>com.tibco.bx.lockSignal.SignalAppName.SignalAppVersion.signalName</code></p> <p>For more information see, Locking of Signal Definitions.</p>
<code>com.tibco.bx.management.queryMaxResultSize</code>	<p>The maximum number of process templates that are displayed in the TIBCO Workspace Start Process Instance dialog.</p> <p>Default = 500</p> <p>For more information, see "Setting How Many Templates Can Be Listed in the Workspace "Start Process Instance" Dialog" in the <i>TIBCO ActiveMatrix BPM Administration</i> guide.</p>
<code>com.tibco.bx.maxCancelChildrenBatchSize</code>	<p>The number of process instances to cancel in a batch.</p> <p>Default = 100</p> <div>  <p>Do not change the value of this property unless you are advised to by TIBCO Support.</p> </div>
<code>com.tibco.bx.maxGlobalSignalListenerCount</code>	<p>The number of global signal event handlers that can listen on the same signal definition with the same correlation.</p> <p>Default = 100</p> <p>For more information, see Maximum Listener Count.</p>

JVM Property	Description
com.tibco.bx.maxTasksPerInstance	<p>The maximum number of tasks that can be generated by a process instance. If this threshold is reached, the process instance is halted.</p> <p>Default = 1000</p> <p>For more information, see Configuring the Maximum Number of Tasks per Process Instance.</p>
com.tibco.bx.showExtendedPriorities	<p>By default, process instance priorities are shown in the event log as LOW, NORMAL, HIGH, or EXCEPTIONAL (which equate to 100, 200, 300, or 400, respectively). If a process instance priority is set to a value other than 100, 200, 300, or 400, it is shown in the event log as "CUSTOM(<i>value</i>)".</p> <p>If this property is set to "yes", and a process instance priority is set to a value other than 100, 200, 300, or 400, that integer value is shown in the event log, rather than CUSTOM(<i>value</i>).</p> <p>Note that this only applies to process instances whose priorities are set after this property is set to "yes". Process instances whose priorities were set to a value other than 100, 200, 300, or 400 prior to setting this property to "yes" still display as CUSTOM(<i>value</i>).</p> <p>Default = no</p>
com.tibco.bx.suspendOnError	<p>If this property is set to "yes", and an activity throws a Java exception, Process Manager checks to see if a number of criteria are true:</p> <ul style="list-style-type: none"> • The process instance is running against a process template that was deployed from a pre-3.5.10 version of TIBCO Business Studio. • There is no user catch error event in scope for the thrown exception. • The exception has not been raised by a user throw error event. <p>If all of these criteria are satisfied, the current transaction is rolled back and the process instance is placed in the SUSPENDED state. Otherwise:</p> <ul style="list-style-type: none"> • If a catch error event is in scope for the activity, processing continues there. • If no catch error event is in scope, the process instance is placed in the FAILED state and purged from the BPM runtime database. <p>Default = yes</p> <p>For more information, see "Configuration of Error Handling Behavior for Process Instances" in the <i>TIBCO ActiveMatrix BPM Administration</i> guide.</p>

JVM Property	Description
com.tibco.pvm.txVerification.enable	<p>This property provides a workaround for a Microsoft SQL Server defect where a commit returns successfully (in XA mode) even though the transaction has not completed. This defect can result in duplicate work items.</p> <p>If this property is set to "yes", a delay is introduced between transactions to ensure that the previous transaction has committed completely.</p> <p>Default = yes (if Microsoft SQL Server is being used, otherwise the default = no)</p> <p>If this property is set to "yes", the properties listed below can be used to configure the transaction delay.</p> <p>Also see "Setting Transaction Verification Property (SQL Server Only)" in the <i>TIBCO ActiveMatrix BPM Installation and Configuration Guide</i>.</p>
com.tibco.pvm.txVerification.retryInterval	<p>These properties are used in conjunction with the com.tibco.pvm.txVerification.enable property (see above) to configure the transaction delay.</p> <ul style="list-style-type: none"> • retryInterval - The number of milliseconds between checks to see if the previous transaction has committed completely. Default = 5 • timeout - The number of milliseconds it checks to see if the previous transaction has committed completely before stopping the checks. Default = 60000 (60 seconds) • warn - Each time a verification check fails, "Tx[%s] verification failed, will retry in %sms" is written to the log. If this property is false (the default), the log message is written to DEBUG. If this property is true, the log message is written to the WARN level. <p>If the check fails after the retryInterval, "Unable to verify transaction [%s], suspending the process" is logged.</p>
com.tibco.pvm.txVerification.timeout	
com.tibco.pvm.txVerification.warn	

Configuring JVM Properties on the BPMNode

JVM Properties are configured using TIBCO ActiveMatrix Administrator. Use the following procedure to add a new property, delete an existing property, or modify the value of an existing property.

1. In ActiveMatrix Administrator, select **Infrastructure > Nodes**.
2. In the **Nodes** list, select **BPMNode**.
3. In the lower pane, select the **Configuration** tab.
4. Select the **JVM Configuration** link.

The **Properties** section lists the existing JVM properties defined for the BPMNode. From this section, you can:

- Add a new JVM property by clicking **Add**, then entering the name and value of the property.
- Delete an existing JVM property by selecting the property in the list, then clicking **Delete**.

- Modify the value of an existing JVM property by selecting the property in the list, then changing the value in the **Value** field.
5. Click **Save**.
 6. Stop, then restart the BPMNode to have your changes take effect:
 - a. Select the **BPMNode** in the **Nodes** list.
 - b. Click **Stop**.
 - c. When the **Node State** column shows that the node is stopped, click **Install or Sync**.
This applies the configuration changes.
 - d. Click **Start**.

Load Balancer

It is important to consider the use of a load balancer to spread clients evenly across multiple BPM nodes, whether they are running on the same physical machine or spread across multiple machines.

One such load balancer is the Membrane load balancer (<http://www.membrane-soa.org/service-proxy-doc/4.4/soap-loadbalancing.htm>). Using the username passed in the SOAP request for sticky sessions means that regardless of how the clients are configured, they all have the same Membrane load balancer URL.

If a load balancer is used, there might be performance implications, depending on where it is located, the network being used, and the load on the machine hosting the balancer (assuming a software solution).

The use of a load balancer also needs to be tested to ensure that it behaves correctly if one or more of the BPM nodes are unavailable. Ensure that the load balancer correctly detects that a node is unavailable and pushes client requests to other available nodes.

Using a Load Balancer and Sticky Sessions with Pageflows

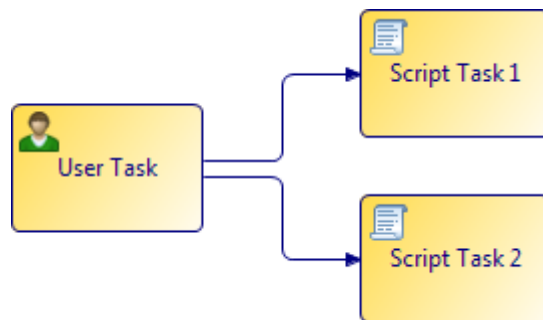
Pageflows are in-memory processes. When a pageflow is started, the state of the pageflow instance is cached, in memory, on the node where the pageflow is being executed. Subsequent calls to the pageflow must be routed through the same node for successful execution of the pageflow. Therefore, if you are operating in a distributed BPM system, you must use sticky sessions on a load balancer to ensure that all subsequent calls to a pageflow are routed to the same node.

For additional information, see "Pageflows" in *TIBCO ActiveMatrix BPM Concepts*.

Process Design

There are many aspects to process design that affect performance.

- **Process Data** - Size of data being passed around the process and passed to work items (number and type of fields, BOM number, complexity, and size) can have a marked impact on performance.
- **Scripts** - Examine the scripts in the process, including script tasks and those that are run as Process Manager (PM) and Work Manager Scripts. For script tasks, consider whether the contents can be moved into the PM complete script that is on the previous task. Doing so removes an entire task from the process flow, reducing audit data and processing required.
- **Database Tasks** - Examine the data that is selected, updated, inserted, or deleted. Consider different JDBC resources for different database tasks. This reduces contention when a database connection is required for the task. Check that the relevant indexes are set for the SQL operations performed.
- **Gateways** - Consider whether the gateways in the process are actually required. For example, a parallel split gateway may not actually be needed and can be replaced by just having two paths running directly from the task:



- **Web Service Tasks** - The location of web services, whether local or on a remote machine has an effect on performance. Service virtualization is an option to consider.

How well the service host is configured to support concurrent client requests affects performance as well.

The exact nature of the service calls (database calls, writes to disk) should be examined as well to determine if there are efficiencies to be made.

Use a monitoring tool such as the Membrane Monitor tool (<http://www.membrane-soa.org/soap-monitor/>) to intercept traffic between BPM and the web services to see if the data being passed could this be reduced.

- **Reply Tasks** - Determine if the reply tasks are necessary to the process flow. If no data is being returned, consider using the “reply immediate” setting for the process. This ensures that the client does not wait for the process instance to start in order to reach the reply task.
- **Chained Groups** - At the start of an embedded sub-process that is a chained group, the user task make an asynchronous call to BRM to start a new group. Once BRM has replied the instance is queued for processing again by an available process engine thread. The asynchronous way that chained groups are implemented means that you cannot expect work items to appear as soon as the instance is started; there will be some latency. If this proves problematic, consider revising the process.
- **User Task Offer Sets** - When allocating or re-offering a work item BRM has to move the offer set between the `brm_work_item_offer` and `brm_work_item_resources` table. This involves deleting and then inserting the rows in the database.

With a large offer set (>500) you may notice that the time to allocate or `allocateAndOpen` a work item (in the offered state) may take longer than opening a work item already allocated to you.

Also, with a large offer set, the scheduling of the work item will be slightly slower due to the increased number of database inserts that are performed. The impact is most obvious, however, during the allocation of the work item.

LDAP

Since all calls into the ActiveMatrix BPM server need to be checked for valid credentials, a well configured and performing Lightweight Directory Access Protocol (LDAP) server is essential.

Ensure that the LDAP server is external, that the LDAP attribute cache is correctly sized, and that the LDAP server is the one being used to authenticate users making SOAP requests. Consider increasing the number of threads running in the LDAP server.

Check the network configuration a capacity between the LDAP server and the ActiveMatrix BPM server.

Consider using the 'search' method for LDAP authentication. There are two forms for LDAP authentication: the 'bind' method and one that uses a search method. The advantage of the search method is that a connection need only be established once and then re-used for all authentications. In contrast, the 'bind' method requires that a connection is established for each authentication.

Windows Tools

There are several useful Windows tools for monitoring ActiveMatrix BPM systems.

On Windows the Performance Monitor can be used to monitor in real time or collect log data for later analysis. For more information, see <http://technet.microsoft.com/en-us/library/cc749249.aspx>.

- CPU/Processes - a more detailed alternative to Task Manager is Process Explorer (<http://technet.microsoft.com/en-gb/sysinternals/bb896653.aspx>).
- Network - Task Manager's Networking tab is useful for monitoring this, as well as Process Explorer.
- Disks - Windows has a comprehensive set of performance counters that can be gathered and monitored on physical and logical disks. This page explains the performance counters available for disk monitoring: <http://blogs.technet.com/b/askcore/archive/2012/03/16/windows-performance-monitor-disk-counters-explained.aspx>.