

WebSphere WebSphere Process Server for z/OS
Version 6.2.0

Tuning for z/OS

IBM

WebSphere WebSphere Process Server for z/OS
Version 6.2.0

Tuning for z/OS

IBM

Note

Before using this information, be sure to read the general information in the Notices section at the end of this document.

25 June 2010

This edition applies to version 6, release 2, modification 0 of WebSphere Process Server for Multiplatforms (product number 5724-L01) and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, send an e-mail message to doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright IBM Corporation 2005, 2009.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

PDF books and the information center

PDF books are provided as a convenience for printing and offline reading. For the latest information, see the online information center.



As a set, the PDF books contain the same content as the information center.

The PDF documentation is available within a quarter after a major release of the information center, such as Version 6.0 or Version 6.1.

The PDF documentation is updated less frequently than the information center, but more frequently than the Redbooks®. In general, PDF books are updated when enough changes are accumulated for the book.

Links to topics outside a PDF book go to the information center on the Web. Links to targets outside a PDF book are marked by icons that indicate whether the target is a PDF book or a Web page.

Table 1. Icons that prefix links to topics outside this book

Icon	Description
	<p>A link to a Web page, including a page in the information center.</p> <p>Links to the information center go through an indirection routing service, so that they continue to work even if target topic is moved to a new location.</p> <p>If you want to find a linked page in a local information center, you can search for the link title. Alternatively, you can search for the topic id. If the search results in several topics for different product variants, you can use the search result Group by controls to identify the topic instance that you want to view. For example:</p> <ol style="list-style-type: none">1. Copy the link URL; for example, right-click the link then select Copy link location. For example: <code>http://www14.software.ibm.com/webapp/wsbroker/redirect?version=wbpm620&product=wesb-dist&topic=tins_apply_service</code>2. Copy the topic id after <code>&topic=</code>. For example: <code>tins_apply_service</code>3. In the search field of your local information center, paste the topic id. If you have the documentation feature installed locally, the search result will list the topic. For example: <div data-bbox="370 1367 1175 1560" style="border: 1px solid black; border-radius: 10px; padding: 10px;"><p>1 result(s) found for</p><p>Group by: None Platform Version Product Show Summary</p><p>Installing fix packs and refresh packs with the Update Installer</p></div> <ol style="list-style-type: none">4. Click the link in the search result to display the topic.
	<p>A link to a PDF book.</p>

Contents

PDF books and the information center	iii	Chapter 8. Advanced tuning	29
Figures	vii	Tracing and monitoring considerations	29
Tables	ix	Configure WorkArea Service's maximum send and receive	29
Chapter 1. Introduction	1	Disable the CEI emitter validateEvent property	29
Chapter 2. Performance tuning methodology	3	Chapter 9. Tuning for large objects	31
Chapter 3. Tuning checklist	5	Reduce or eliminate other processing while processing a large object	31
Chapter 4. Tuning parameters	7	Chapter 10. Tuning for maximum concurrency	33
Tracing and logging flags	7	Tune edge components for concurrency	33
64-bit mode.	7	MDB ActivationSpec	33
Java tuning parameters	8	Configure listener port	34
Java GC policy	8	Configure thread pool sizes	34
Java heap sizes	8	Configure dedicated thread pools for MDBs	34
Workload profile	9	Tune intermediate components for concurrency	35
Workload manager service class	9	Configure JMS and JMS queue connection factories	35
MDB ActivationSpec.	9	Configure DataSource options	36
MQ listener port.	10	DataSource prepared statement cache size	36
MDB throttle.	10	Chapter 11. Messaging tuning.	37
Thread pool sizes	10	Choose datastore or filestore for message engines	37
JMS connection pool sizes	11	Set data buffer sizes (discardable or cached)	37
DataSource Connection Pool Size	12	Move message engine data stores to a high performance DBMS.	37
DataSource prepared statement cache size	12	Setting up the data stores for the messaging engines	38
Messaging engine properties.	13	Create the DB2 database and load the data store schemas	38
Minimize security	13	Create the data sources for the messaging engines	39
Disable automatic synchronization for network deployment	13	Change the data stores of the messaging engines.	40
Chapter 5. Tuning business processes	15	Chapter 12. WebSphere MQ tuning.	41
Tuning long-running processes	16	Make judicious use of tracing	41
Planning messaging engine settings	16	Employ message management	41
Fine-tuning the messaging provider	17	Set the CCSID	41
Improving the performance of business process navigation.	17	Configure appropriate logging	41
Tuning microflows	18	Enable MQ connection pooling in WebSphere Application Server	41
Chapter 6. Tuning business processes that contain human tasks.	21	Chapter 13. Database: general tuning	43
Reduce concurrent access to human tasks	21	Place database log files on a fast disk subsystem	43
Reduce query response time.	21	Place logs on separate device from table space containers	43
Avoid scanning whole tables	22	Chapter 14. Database: DB2 specific tuning	45
Optimize task and process queries.	22	Update database statistics	45
Chapter 7. Tuning Business Process Choreographer Explorer	25	Set buffer pool sizes correctly	45
Tuning the Business Process Choreographer		Maintain proper table indexing.	46
Explorer reporting function	26	Size log files appropriately	46

Chapter 15. Advanced Java Heap Tuning	47
Monitoring garbage collection	47
Setting the heap size for most configurations	48
Setting the heap size when running multiple JVMs on one system	48
Reduce or increase heap size if out of memory errors occur	49

Chapter 16. Common references	51
Chapter 17. z/OS specific references	53
Notices	55
Index	59

Figures

Tables

- | | | | |
|--|-----|---|----|
| 1. Icons that prefix links to topics outside this book | iii | 3. Messaging engine data source names | 39 |
| 2. Messaging engine schema names | 38 | 4. New datasource names | 39 |

Chapter 1. Introduction

To optimize performance you must change the default system settings. This section lists several areas to consider during system tuning. This includes tuning the WebSphere Business Process Management products, and also other products in the system (for example DB2). The documentation for each of these products contains information about performance, capacity planning, and configuration.

This documentation provides you with guidance for performance considerations in a variety of operational environments. Assuming that all these issues have been addressed from the perspective of the actual product, additional levels of performance implications are introduced at the interface between these products and the products covered in this section.

A number of configuration parameters are available to the system administrator. This section identifies several specific parameters observed to affect performance, but it does not address all parameters that are available. For a complete list of configuration parameters and possible settings, refer to the relevant product documentation.

The next section describes a methodology to use when tuning a deployed system. It is followed by a basic tuning checklist that enumerates the major components and their associated tuning concepts.

The subsections that follow address tuning in more detail, first describing several tuning parameters and their suggested setting (where appropriate), and finally providing advanced tuning guidelines for more detailed guidance for key areas of the system. While there is no guarantee that following the guidance in this chapter immediately provides acceptable performance, it is likely that degraded performance can be expected if these parameters are incorrectly set.

The last set of topics in this section contain references to related documentation that can prove valuable when tuning a particular configuration.

Chapter 2. Performance tuning methodology

It is recommended that you use a system-wide approach to performance tuning a WebSphere® Business Process Management environment. System performance tuning requires training and experience and is not going to be exhaustively described. Some key aspects of tuning that are particularly important are described.

It is important to note that tuning includes every element of the deployment topology:

- Physical hardware topology choices
- Operating system parameters tuning
- WebSphere Process Server, WebSphere Application Server, and Messaging Engine tuning.

The methodology for tuning can be stated very simply as an iterative loop:

- Select a set of reasonable initial parameter settings
- Run the system
- Monitor the system to obtain metrics that indicate whether performance is being limited
- Use monitoring data to guide further tuning changes
- Repeat until done

You can now examine each item in turn:

- Pick a set of reasonable initial parameter settings
- Use the tuning checklist for a systematic way to set parameters
- Monitor the system to determine system health and determine the need for further tuning. The following items must be monitored:
 - Each physical system in the topology including front end and back end servers like web servers, and database servers
 - CPU utilization, memory utilization, disk utilization, network utilization using relevant z/OS® tools such as SMF and RMF™
 - Each JVM process started on a physical system, for example WebSphere Process Server controller, servant, adjunct
 - Verbosegc statistics
- For each WebSphere Process Server server, use TPV (Tivoli® Performance Viewer) to monitor the following:
 - For each data source, the data connection pool utilization
 - For each JCA resource connector, the connection and session pool statistics
 - For each thread pool (webcontainer, default, work managers), the thread pool utilization
- Use monitoring data to guide further tuning changes.

This is a vast topic which requires skill and experience. This phase of tuning requires the analyst to look at the collected monitoring data, detect performance bottlenecks, and do further tuning. The key characteristic about this phase of tuning is that it is driven by monitoring data collected in the previous phase.

Examples of performance bottlenecks include, but are not limited to:

- Excessive use of physical resources like CPU, disk, memory. These can be resolved either by adding more physical resources, or rebalancing the load more evenly across the available resources.
- Excessive use of virtual resources. Examples include heap memory, connection pools, and thread pools. For these, tuning parameters must be used to remove the bottlenecks.

Chapter 3. Tuning checklist

Use the following checklist to tune your system:

Common

- Disable tracing and monitoring when possible
- Move databases from the default Derby to a high performance DBMS such as DB2 for z/OS
- Where practical, do not enable security
- Use appropriate hardware configuration for performance measurement. The environment must preferably consist of dedicated resources to enable repeatable performance tests to be executed. Shared resources, or resources or resources with an availability which can be influenced by other systems
- Do not run the production server in development mode
- Do not use the Unit Test Environment (UTE) for performance measurement
- Configure MDB activation specs
- Configure for clustering (where applicable)
- Configure thread pool sizes
- Configure data sources : connection pool size, prepared statement cache size
- Disable validation in CEI emitter

Messaging and Message Bindings

- Optimize activation specification (JMS)
- Optimize queue connection factory (JMS)
- Configure connection pool size (JMS)
- Optimize listener port configuration (MQJMS, MQ)
- Configure SIBus data buffer sizes

Database

- Place database table spaces and logs on a fast disk subsystem
- Place logs on devices that are separate to table space containers
- Maintain current indexes on tables
- Update database statistics
- Set log file sizes correctly
- Optimize the buffer pool size
- Set the RRULOCK DB2 parameter to YES to prevent tablespace locks on WebSphere Process Server tables.

Java

- Set the heap/nursery sizes to manage memory efficiently
- Choose the appropriate garbage collection policy

Chapter 4. Tuning parameters

This section lists performance tuning parameters commonly used in tuning the products covered in this report. Some flags or check boxes are common to all or a subset of the products, while others are specific to a particular product.

Tracing and logging flags

Most tracing and monitoring is controlled using the administrative console. Make sure that the appropriate level of tracing/monitoring is set for PMI monitoring, logging, and tracing using the administrative console.

About this task

To adjust tracing from the console perform the following steps:

Procedure

Procedure

1. Open the administrative console and navigate to **Troubleshooting > Logging and Tracing > (Server-name) > Change Log Detail Levels**.
2. Set both the configuration and runtime to ***=all=disabled**.
3. Tracing can also be disabled from the z/OS console by issuing a modify command for the server. For example, to disable all java traces from the z/OS console issue the command **f (server name),tracejava=*=all=disabled**.
4. Monitoring can be controlled from the administrative console. To disable monitoring from the console, navigate to **Monitoring and Tuning > Performance Monitoring Infrastructure (PMI) > (Server-name)** and select **None**.

64-bit mode

WebSphere Process Server 6.2.0 run on WebSphere Application Server for z/OS 6.2.0, which can be configured to run in 64-bit mode. Activating 64-bit mode introduces performance overhead. Some of the measurements that are described experienced significant performance degradations when running in 64-bit mode with no apparent benefit. The performance degradations garbage collection statistics showed that the Java heap was not under stress, so there was no benefit to be gained by increasing the heap size beyond values that were available in 31-bit mode.

When the heap is under stress, for example when dealing with high concurrency or large objects/messages, 64-bit mode provides the ability to increase the heap size to a point where some relief can be realized. However, this has not been evaluated for this tuning section of the information center.

Activating 64-bit mode involves setting the **Run in 64 bit JVM mode** option in the general properties for the server. In addition, the **AMODE=64** parameter must be specified on the z/OS start command for the server. See the topic Starting a server from the MVS console for more information.

Java tuning parameters

In this section we list a few frequently used Java Virtual Machine (JVM) tuning parameters.

For a complete list, consult the JVM tuning section in the WAS for z/OS V6.1 Tuning Guide

Access the JVM administration page by opening the administrative console and navigating to **Servers > Application Servers > your server name > Server Infrastructure > Java and Process Management > Process Definition > (Control or Adjunct or Servant) > Additional Properties > Java Virtual Machine.**

Java GC policy

The default garbage collection (GC) algorithm is Mark-Sweep-Compact. In some cases, the generational concurrent (gencon) algorithm delivers better performance with a tuned nursery size, as discussed in the next section.

To change the GC policy to gencon, add **-Xgcpolicy:gencon** to the Generic JVM arguments on the Java Virtual Machine administration panel.

For further information, see information on tuning storage and the JVM in the WAS for z/OS V6.1 Tuning Guide.

Java heap sizes

Garbage collection (GC) is the process of freeing unused objects so that portions of the JVM heap can be reused. Because the Java language specification does not provide explicit `delete()` or `free()` byte codes, it is very important to occasionally detect and delete objects which no longer have active references and free that space for reuse.

Garbage collection is triggered automatically when there is a request for memory, for example when objects are created, and the request cannot be readily satisfied from the free memory available in the heap (allocation failure). Garbage collection can also be activated programmatically via a Java class library `System.gc()` call. In this case garbage collection occurs immediately and synchronously.

Use the appropriate Java heap size for production environments. To change the default Java heap sizes, set the Initial Heap Size and Maximum Heap Size explicitly on the Java Virtual Machine administrative panel.

If Generational Concurrent Garbage Collector is used, the Java heap is divided into a new area (nursery) where new objects are allocated and an old area (tenured space) where longer lived objects reside. The total heap size is the sum of the new area and the tenured space. The new area size can be set independently from the total heap size. Typically the new area size must be set around $\frac{1}{2}$ of the total heap size. The relevant parameters are:

- `-Xmns<size>` : initial new area size
- `-Xmnx<size>` : maximum new area size
- `-Xmn<size>` : fixed new area size

Workload profile

The number of servant threads can be configured by setting the workload profile for the server. The number of threads for each profile setting is determined using a formula involving the number of processors that are available.

Refer to the WebSphere for z/OS Information Center for more information. Take care when allocating threads to the WebSphere Process Server for z/OS server. On z/OS there is a standard thread setting for ISOLATE, IOBOUND, CPUBOUND, or LONGWAIT that must be examined based on the workload.

To set the workload profile, open the administrative console and navigate to **Servers > Application servers > (Server-name) > Container Settings - Container Services - ORB Service > z/OS additional settings** and select the workload profile from the drop-down list.

Workload manager service class

The z/OS operating system assigns machine resources to process the requests based on the workload manager service class which has been assigned. Ensure that the workload manager has been configured to classify the requests to receive an appropriate level of service.

The workload can be classified using a workload manager subsystem type of CB.

More fine grained workload classification can be accomplished using the workload classification file. For more information, see the WebSphere Application Server Information Center.

MDB ActivationSpec

JMS activation specifications are used to bind application MDBs to the message queues which drive the MDB onMessage methods. Two activation specification parameters of particular interest are the maximum batch size and the maximum concurrent endpoints.

For each JMS export component there is an MDB and its corresponding ActivationSpec (JNDI name: module name/export component name_AS). The maximum concurrent endpoints specifies the number of messages a particular MDB can process concurrently, but be aware that each active MDB instance runs on a thread from the container thread pool. The default value for the JMS export MDB is 10, which means that up to 10 BOs from the JMS queue can be delivered to the MDB threads concurrently.

The maximum batch size in the activation specification also has an impact on performance. The default value is 1. The maximum batch size value determines how many messages are taken from the messaging layer and delivered to the application layer in a single step (this does not mean that this work is done within a single transaction, and therefore this setting does not influence transactional scope). Increasing this value for activation specifications associated with long-running business processes can improve the efficiency of MDB message processing.

You can configure the activation specification parameters in the administrative console by navigating to **Resources > JMS > Activation specifications > (ActivationSpec name)**.

MQ listener port

For the MQ or MQJMS binding, a Listener Port is used for configuring the delivery of inbound messages to WebSphere Process Server or WebSphere Enterprise Service Bus. Listener Ports are the equivalent of JMS Activation Specifications for binding application MDBs to message queues. They have similar parameters which can be configured in the Message Listener Service part of the WAS administrative console.

The Maximum Sessions parameter is equivalent to the JMS activation specification maximum concurrent endpoints, and determines the level of concurrency for MDB processing on a particular Listener Port.

The Maximum Messages parameter determines how many times an MDB's onMessage method can be called in the same context.

The Maximum Sessions value must be less than the message listener thread pool value. The effect of the MDB throttle must also be considered when setting this parameter.

You can set the listener port parameters by opening the administrative console and navigating to **Servers > Application servers > (Server-name) > Message Listener Service > Listener Ports > (ListenerPort-name)**.

MDB throttle

On WebSphere Application Server for z/OS, a control of the MDB processing is the MDB throttle which is used to prevent the z/OS WLM work queue becoming full of messages waiting to be processed by worker threads in the WebSphere Application Server servant. The MDB throttle limits how far a message listener port reads ahead down a JMS queue (or topic) to ensure that the backlog of messages to be processed on the WLM request queue does not become too large.

The high and low thresholds for the MDB throttle are determined by the value of the maximum sessions parameter of the listener port. The high threshold is set to the maximum sessions value for the listener port. When this amount of messages are queued up on the WLM work request queue, no further requests are queued until the number of concurrent requests drops to the low threshold, which by default is 50% of the maximum sessions value.

To avoid the situation in which there are idle threads in the WebSphere Application Server servant while the MDB throttle is blocking further requests from being queued, the recommendation is that the maximum sessions value is set to twice the number of worker threads in the servant.

To set the maximum sessions value, open the administrative console and navigate to **Servers > Application servers > (Server-name) > Message Listener Service > Listener Ports > (ListenerPort-name)**.

Thread pool sizes

WebSphere uses thread pools to manage concurrent tasks. To set the maximum size property of a thread pool, open the administrative console and navigate to **Servers > Application servers > server name > Additional Properties > Thread Pools > (thread pool name)**.

You typically need to tune the following thread pools:

- Default
- ORB.thread.pool
- WebContaineran interesting point

If a work manager is used set the **Maximum number of threads** to a value high enough to prevent the thread pool from running out of threads. You can set this value by opening the administrative console and navigating to **Asynchronous beans > work managers**.

The WebSphere MQ messages make use of message listeners to invoke actions against MDBs. To enable concurrent execution of MDB instances and listener ports, the message listener thread pool must be configured. The thread pool must be large enough to support any listener port maximum sessions value. Remember that the thread pool setting is server wide, and is set by navigating to the following area of the administrative console: **Application servers > (Server-name) > Message Listener Service > Thread Pool**.

One symptom of insufficient concurrency is processor idleness. Vary the concurrency to achieve maximum processor utilization and throughput. For information about how to ensure that sufficient worker threads are available, see Tuning for Subsystems in the WebSphere Application Server for z/OS Tuning Guide.

JMS connection pool sizes

You can increase the maximum connection pool size to allow for greater concurrency. Note that if only one deployed application is obtaining connections from a particular connection factory, there is no benefit in increasing the maximum queue connection pool size beyond the maximum concurrency as configured within the activation specification.

The maximum connection pool size specifies the maximum number of physical connections that you can create in this pool. These are the physical connections to the backend resource. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use returns to the pool, or a `ConnectionWaitTimeout` exception is issued.

For example, if the **Maximum Connections** value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in `Connection Timeout` for a physical connection to become free. The threads waiting for connections to underlying resource are blocked until the connections are freed up and allocated to them by the pool manager. If no connection is freed in the specified interval, a `ConnectionWaitTimeout` exception is issued.

If **Maximum Connections** is set to 0, the connection pool is allowed to grow infinitely. This causes the **Connection Timeout** value to be ignored.

When listener ports are involved, the connection factory pool sizes need to take this into consideration. The connection pool of the connection factory must be larger than the number of listener ports, and the session pool of the connection factory must be large enough to cover the maximum sessions value for any listener port using the connection.

The maximum connection pool size can be set from the administrative console under **Resources** as follows:

- **Resources > JMS > Connection factories > (factory name)**
- **Resources > JMS > Queue connection factories > (factory name)**

From the connection factory administrative panel, open **Additional Properties > Connection pool properties**. Set maximum connections property for the max size of the connection pool.

DataSource Connection Pool Size

You can adjust the minimum and maximum number of connections to DB2[®] for z/OS, and the wait for connection timeout value. :

Open the administrative console and navigate to **Resources > JDBC > Data sources > (Data-source-name)**.

The maximum size of the DataSource connection pool is limited by the value of Maximum connections property, which can be configured by opening the administrative console and navigating to **Additional Properties -> Connection pool properties** in the DataSource panel.

The following DataSources typically need to be tuned:

- BPE DataSource for BPE DB
- SCA Application Bus ME DataSource
- SCA System Bus ME DataSource
- CEI Bus ME DataSource
- ESB Logger Mediation DataSource
- WBI DataSource
- The Event and Event Catalog DataSources

DataSource prepared statement cache size

The prepared statement cache optimises the processing of prepared SQL statements and callable statements.

The size of the cache can be tuned to avoid cache entries from being discarded. You can tune the size of the cache by opening the administrative console and navigating to **Resources > JDBC > Data sources > (Data-source-name) > WebSphere Application Server data source properties**

BPEL macroflows (long-running processes) make extensive use of the BPEDB data source for persisting data relevant to the process. Persisting various data results in the usage of many different statements, far more than the default capacity of the data source's cache. This results in an excessive number of cache misses, making the caching ineffective. This problem can be resolved by increasing the size of the cache.

The BPC container in WebSphere Process Server uses prepared statements extensively. The statement cache size of this DataSource must be at least 128. The number of cache discards for a data source can be monitored using the Tivoli performance viewer in the administrative console.

Messaging engine properties

Two message engine custom properties can impact the messaging engine performance.

The `DiscardableDataBufferSize` is the size in bytes of the data buffer used when processing best effort non persistent messages. The purpose of the discardable data buffer is to hold message data in memory as it is never written to the data store for this quality of service. Messages that are too large to fit into this buffer are discarded.

The `CachedDataBufferSize` is the size in bytes of the data buffer used when processing all messages other than best effort non persistent messages. The purpose of the cached data buffer is to optimize the performance by caching in memory data that might otherwise need to be read from the data store.

You can set the `DiscardableDataBufferSize` and `CachedDataBufferSize` by opening the administrative console and navigating to **Service Integration > Buses > (Bus-name) > Messaging Engines > (Messaging-engine-name) > Custom properties**.

- `sib.msgstore.discardableDataBufferSize`
- `sib.msgstore.cachedDataBufferSize`

Minimize security

The security features of WebSphere Application Server provide important protection against unauthorized use of resources. Whilst not advocating that security be completely disabled, there are various security mechanisms available, each of which add processing overhead.

You must give careful consideration to which of the WebSphere Application Server security features are activated. These are configured in the administrative console under the security section, however some aspects of security are configured at application assembly and deployment time.

Disable automatic synchronization for network deployment

In a network deployment configuration the node agent synchronizes the deployment manager configuration data with that of the node. If this synchronization occurs too frequently, overall system performance can be effected.

You can adjust the frequency of the synchronization for network deployment or disable automated synchronization by opening the administrative console and navigating to **System administration > Node agents > nodeagent > File synchronization service**.

Chapter 5. Tuning business processes

Use this task to improve the performance of business processes.

Before you begin

After successfully running business processes, you can perform this task to improve performance.

Procedure

Procedure

1. Define how to measure the baseline performance, and which measurements you want to optimize.

For example, for some business applications, it is preferable to reduce the response time for end-users under peak-load conditions. For other applications, the rate that the system can process transactions might be more important than the actual duration of each transaction.

2. Make baseline measurements.

Make the baseline measurements under conditions of load, time-of-day, and day-of-week that are appropriate for tuning your application. Normally, the most important baseline measurements are the throughput and response times. Throughput values are measured after a specific bottleneck capacity is reached, for example 100% CPU load, disk I/O at maximum, or network I/O at 100%. Reliable response time values are best measured for a single process instance during low server utilization.

3. Tune the application.

Applications can contain multiple processes. Because microflows perform better than long-running processes, if persistence is not necessary, and the functionality can be handled single threaded in one transaction, choose to model microflows instead of long-running processes. Also consider separating branches of a long-running process into microflows. Furthermore synchronous service invocations are usually faster than asynchronous service invocations. So for performance reasons, prefer synchronous service invocations although this is not the default behavior in long-running processes.

In long-running processes you can change transaction boundaries. In most cases, performance can be improved by reducing the number of transaction boundaries. However, the optimal number of transaction boundaries can only be found out by performance testing. Because serializing and deserializing data is also expensive, consider using parallel execution paths in your processes instead of serializing activities, and minimizing the size and complexity of data that is part of your process. Also minimize the number of events that are emitted.

4. Tune the processes.

Depending on whether your application uses long-running processes or microflows, perform one of the following steps:

- To tune long-running processes, perform the steps that are described in "Tuning long-running processes" on page 16. These processes tend to run for a long time, but can be interrupted by events or human interaction. Their

performance therefore depends on the performance of the Business Process Choreographer database and the messaging service.

- To tune microflows, perform the steps that are described in “Tuning microflows” on page 18. These processes tend to run for only a short time. They use the database only for audit logging, if enabled, and to retrieve the template information. These processes involve no human interaction.
5. Review the current configuration for performance bottlenecks that can be eliminated.
Possibilities to consider include:
 - Installing more processors, more memory, and faster disks.
 - Storing the database logs on different physical disks from the data, and distributing the data on several disks.
 - Using DB2, rather than Cloudscape, for optimal performance.
 6. Repeat the benchmark measurements under similar load conditions to those of the baseline measurements.
Keep a permanent record of the application performance measurements to objectively measure any future changes in performance.

Results

The business processes are configured to run measurably faster.

Tuning long-running processes

Use this task to improve the performance of long-running business processes.

About this task

Long-running processes can include user-interaction, asynchronous invocations, multiple receives, picks, and event handlers, for example; they use database and messaging subsystems for storing persistent states. The following topics describe how to improve the performance of long-running processes.

Planning messaging engine settings

Use this task to plan your initial settings for the messaging engines.

About this task

To achieve the best performance for long-running processes, tune the messaging system for maximum performance of persistent messages. For the data store back-end types, file store is preferred because it performs well. Use a database data store if your environment runs in a cluster and you cannot use a file store.

If you use the service integration capabilities of WebSphere Application Server, follow the instructions given in Setting tuning properties for service integration in the WebSphere Application Server for z/OS information center, to set up and tune the data stores for the messaging engines.

Results

Your messaging engines are operating optimally.

Fine-tuning the messaging provider

Use this task to improve the performance of your messaging provider.

Procedure

Procedure

If you use the service integration capabilities of WebSphere Application Server, refer to in the WebSphere Application Server information center.

Results

The performance of your messaging provider is improved.

Improving the performance of business process navigation

You can tune the performance of long-running processes by enabling performance optimizations, and tuning various configuration parameters.

About this task

A long-running process spans multiple transactions. By default, a transaction is triggered by a Java™ Messaging Service (JMS) message. To improve the performance of process navigation, you can configure Business Flow Manager to use a work-manager-based implementation for triggering transactions instead of JMS messages. Regardless of whether you use JMS or work-manager navigation mode, you can tune the size of the intertransaction cache.

The following summarizes the characteristics of the two process navigation modes:

JMS message-based navigation

Process navigation handled by JMS messages that are controlled by the process navigation message-driven bean (MDB).

- If the topology is set up so that the messaging engines are local to the application, a process navigates with server affinity unless it is triggered by external events, such as asynchronous messages or human tasks.
- If the topology is set up so that multiple servers in an application cluster use a single remote messaging engine, then navigation within a process is distributed over the servers in the cluster.

Work-manager-based navigation

Process navigation is handled by a thread pool that is controlled by the work manager. Normal navigation of a process instance is done completely with server affinity.

To ensure transactional integrity, the messages that trigger navigation steps are stored in the Business Process Choreographer database. A background recovery thread periodically scans these messages, and if messages exist that are older than a specified maximum age, it sends them to the JMS queue to be picked up by the process navigation MDB. Business Process Choreographer guarantees that each message is executed exactly once.

If an error occurs that causes the navigation step to roll back, the navigation of the process reverts to JMS-controlled navigation.

Server affinity means that navigation within a process instance happens on one WebSphere Application Server, unless an asynchronous service is invoked, a wait

or timeout condition is encountered, a receive or pick activity is activated, or a human task is executed. These events can cause navigation within a process to continue on another WebSphere Application Server.

Procedure

Procedure

1. Configure Business Flow Manager to use work-manager-based process navigation.

In the administrative console, perform the following steps:

- a. Navigate to **Servers** → **Clusters** → *cluster_name* or **Servers** → **Application servers** → *server_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Business Flow Manager Configuration** → **Configuration**.
 - b. Select the **Enable advanced performance operations** option. Now you can change the values of the following configuration parameters:
 - Message pool size
 - Maximum age for stalled messages
 - Recovery interval for stalled messages
 - Maximum process time on thread
 - Intertransaction cache size
2. Optional: Increase the maximum number of threads available to the workflow manager.

Business Flow Manager requires two threads for internal processing. The remaining threads are available for process navigation. Start with one additional thread for each processor. If you increase the thread pool size, you also need to increase the connection pool size for the Business Process Choreographer database (BPEDB) and for the connection factory (BPECFC). To change the maximum number of threads, perform the following using the administrative console.

 - a. Navigate to **Resources** → **Asynchronous beans** → **Work managers** → **BPENavigationWorkManager**.
 - b. Under **Thread pool properties**, change the value of **Maximum number of threads**.
 - c. Set the value of **Work request queue size** to be the same as the value of **Maximum number of threads**.
 3. Save your changes.
 4. Restart the server to activate your changes.

Results

The work manager now controls your process navigation.

Tuning microflows

Use this task to improve the performance of microflows.

About this task

Microflows run in memory, without any user-interaction or persistent messaging support. Database access is required only if audit logging or Common Event Infrastructure (CEI) are enabled for the microflow. The processing of a microflow

occurs in a single thread, and normally, in a single transaction. The performance of microflows mainly depends on the services called. However, if the memory available for the server is too small, the performance of microflows will be reduced.

Procedure

Procedure

1. Tune the Java Virtual Machine (JVM) heap size.
By increasing the Java heap size, you can improve the throughput of microflows, because a larger heap size reduces the number of garbage collection cycles that are required. Keep the value low enough to avoid heap swapping to disk.
2. Tune the JVM garbage collection. The generational garbage collector policy achieves the best throughput. This policy is activated as a generic JVM argument in the JVM settings. Set the initial value of the collection to half of the total heap size. For example, `-Xgcpolicy:gencon -Xmn512M` activates the policy for a heap size of 1024 MB.
3. Tune the Object Request Broker (ORB) thread pool size. If remote clients connect to the server-side ORB, make sure that there are enough threads available. Navigate to **Application servers** → *server_name* → **ORB Service** → **z/OS additional settings** → **Workload Profile**.
4. Tune the default thread pool size. To increase the number of microflows that can run concurrently, you must increase the default thread pool size. To change the value, using the administrative console, navigate to **Servers** → **Application Servers** → *server_name* → **Add properties** → **Thread pools** → **Default**.

Results

Your microflows are running as fast as possible under the current environment and loading conditions.

Chapter 6. Tuning business processes that contain human tasks

There are various ways to improve the performance of business processes that contain human tasks.

The following topics describe how to tune business processes that contain human tasks.

Reduce concurrent access to human tasks

When two or more people try to claim the same human task, only one person will succeed. The other person is denied access.

Only one person can claim a human task. If several people attempt to work with the same human task at the same time, the probability of collision increases. Collisions cause delays, because of lock waits on the database or rollbacks. Some ways to avoid or reduce the incidence of collision are as follows:

- If concurrent access is high, limit the number of users who can access a particular human task.
- Avoid unnecessary human task queries from clients, by using intelligent claim mechanisms. For example, you might take one of the following steps:
 - Try to claim another item from the list if the first claim is unsuccessful.
 - Always claim a random human task.
 - Reduce the number of potential owners for the task, for example, by assigning the task to a group with fewer members.
 - Limit the size of the task list by specifying a threshold on the query used to retrieve the list. Also consider using filtering to limit the number of hits. You can filter for properties of a task, for example, only showing tasks with priority one or tasks that are due within 24 hours from now. For an inline task, you can also filter for business data that is associated with the task using custom properties or query properties. To perform such filtering, you must specify an appropriate WHERE clause on the query that retrieves the task list.
 - Minimize or avoid dynamic people queries, that is, ones that use replacement variables.
 - Use a client caching mechanism for human task queries, to avoid running several queries at the same time.

Reduce query response time

Reduce the time that the database takes to respond to queries.

When you use a custom client, make sure that the queries set a threshold. From a usability viewpoint, retrieving hundreds or thousands of items is typically undesirable, because the larger the number of database operations, the longer the task takes to complete, and because a person can manage only a small number of results at a time. By specifying a threshold, you minimize database load and network traffic, and help to ensure that the client can present the data quickly.

A better way to handle a query that returns a large number of items might be to rewrite the query, to return a smaller result set of items. You can do this by querying work items for only a certain process instance or work items with only a certain date.

You can also reduce the query result by using filter criteria.

Avoid scanning whole tables

When you use the query application programming interfaces (APIs), to list the objects in the database, you can specify filters that narrow the results you want to retrieve. In these filters, you can specify the values and ranges of object attributes.

When database queries are processed, the filter information is translated into WHERE clauses in a Structured Query Language (SQL) statement. These WHERE clauses map the object attributes to column names in the affected database tables.

If your query specifies a filter that does not translate to an indexed table column, the SQL statement will probably cause the table to be scanned. This scanning impacts performance negatively and increases the risk of deadlocks. Although this performance impact can be tolerated if it happens only a few times a day, it could adversely affect efficiency if it takes place several times a minute.

In such circumstances, a custom index can dramatically reduce the impact. In a real customer situation, a custom index helped to reduce the API response time from 25 seconds to 300 milliseconds. Instead of reading 724 000 rows of the database table, only six rows had to be read. If you define a volatile flag for the instance tables in the Business Process Choreographer database, this helps the DB2 optimizer to decide on an appropriate data access plan. This flag specifies that the index is always used instead of a table scan, even with empty or almost empty tables.

Depending on the filter criteria that you specify, some columns might not be included in an index. If this is the case, and if a table scan is used, resulting in slow query performance, check the access path of the statement, using DB2 Explain, for example. If necessary, define a new index.

Optimize task and process queries

The query and queryAll API calls for retrieving task and process lists can result in complex SQL queries that include combinations of multiple database tables. An optimized representation of the data helps to address performance requirements, particularly for human workflow applications where multiple users access task lists concurrently.

About this task

If Business Process Choreographer is tuned for queries, response times usually are in the region of subseconds on an adequately sized system, even under high load. You can apply standard database calculations to calculate the response time of queries.

High-volume human workflow scenarios are best tuned with query tables. Query tables provide a precalculated set of data that is relevant for specific queries. For example, query properties must be joined by the database with tasks or process

instances when the query runs. If query tables are used, these SQL joins do not need to be calculated anymore at query execution time.

The implementation and maintenance effort for query tables is higher than for standard database tuning techniques. Carefully consider standard database optimization techniques, such as indexes, log file distribution, and memory, before you use query tables.

Two approaches to query tables are supported: materialized views and custom tables. Decide whether to use materialized views or custom tables based on maintenance costs, development costs, and your requirements on the currency of the data that is returned by task and process list queries:

Procedure

- Use materialized views to take advantage of the asynchronous update mechanism, which provides optimal query and process navigation performance.
 - Updates occur only when the materialized view is used
 - Setup, use, and maintenance is relatively simple
 - Can be implemented without changes to the application source code
- Use custom tables to include data from other applications in standard queries using the query or the queryAll interface. Additionally, custom tables can be used to provide an optimized representation of the data that is needed for task and process queries
 - Database triggers or other techniques can be used to synchronously update a custom table which is optimized for task and process list queries
 - Queries must be changed to query the data provided in the custom table

Chapter 7. Tuning Business Process Choreographer Explorer

The following suggestions provide various ways to improve the performance of the Business Process Choreographer Explorer.

Procedure

Procedure

1. Use query tables for customized views.

Business Process Choreographer Explorer provides default query table definitions for each of the predefined views. However, to exploit the full potential of query tables, you should also use query tables as the basis for your customized views. If you have customized views that do not use query tables, consider creating query tables that contain exactly those properties and filters that you need for your business scenario, and redefining those views based on the new query tables.

Avoid using customized views that are not based on query tables. Restrict the use of searches that are not based on query tables to one-off searches where you need the flexibility to define specific filter criteria.

2. Consider increasing the maximum heap size of the server.

Web clients naturally increase the load on your system. The more clients that are connected to your server, the more objects that have to be kept in memory. Therefore consider increasing the maximum heap size of your server. This improves the response time of your application, and increases the maximum number of users that can work in parallel with the application.

3. Tune the Web container thread pool.

The size of the thread pool and the thread inactivity timeout can affect the performance of the Web container. To change these settings navigate to the following area of the administrative console: **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then in the **Additional Properties** section, click **Thread Pools** → **WebContainer**.

- a. Adjust the maximum and minimum pool size.

All HTTP requests for Web client applications are processed using threads from the Web container thread pool. You can adjust the minimum and maximum pool size to influence the performance of your Web client.

The number of maximum threads in the pool does not represent the number of requests your application server can process concurrently. If all of the threads in the pool are in use, additional requests are queued until they can be assigned to a thread. If a client request waits for a thread to be assigned, the response time increases for the client. However, if the maximum number is set too high, the system might get overloaded resulting in an even worse response time for the clients. It might also cause other applications to slow down dramatically.

To determine whether changing the container size might result in a performance gain, you can use Tivoli Performance Viewer to monitor the load on the threads (PercentMaxed counter) and the number of active threads (ActiveThreads counter) for the Web container module. If the value of the PercentMaxed counter is consistently in double digits, then the Web container might be a bottleneck. In this case, increase the number of

threads. If the number of active threads is lower than the number of threads in the pool, decreasing the thread pool size might result in a performance gain.

- b. Adjust the thread inactivity timeout.

The thread inactivity timeout defines after how many milliseconds of inactivity that should elapse before a thread is reclaimed. Set this timeout to a low value, for example, 1, so that many users can work concurrently without having to wait for a free thread in the thread pool. A value of 0 indicates no wait time.

4. Decrease the search limit for large lists.

If you are working with large task or process lists, you might want to decrease the search limit for lists to avoid gathering data that is not accessed by users. To change this setting, navigate to the following area in the administrative console: either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**, then change the setting.

Tuning the Business Process Choreographer Explorer reporting function

The time required to generate a report can vary, and depends on many factors. The following suggestions provide various ways to improve the performance of report generation.

Update your database statistics

For DB2 databases, updating the database statistics when you have a populated production database can improve the performance dramatically.

- To update the statistics for a DB2 database, enter the following commands:

```
RUNSTATS ON TABLE schema_prefix.EVENT_ACT_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;  
RUNSTATS ON TABLE schema_prefix.EVENT_PRC_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;  
RUNSTATS ON TABLE schema_prefix.INST_ACT_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;  
RUNSTATS ON TABLE schema_prefix.INST_PRC_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;  
RUNSTATS ON TABLE schema_prefix.OPEN_EVENTS_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;  
RUNSTATS ON TABLE schema_prefix.QUERY_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;  
RUNSTATS ON TABLE schema_prefix.SLICES_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
```

For large databases, for example, with more than 500 000 process instances, use the WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL statement when you run the RUNSTATS utility.

Where *schema_prefix* is the name of the database schema that was used when the database for the Business Process Choreographer Explorer reporting function was created. For more information about updating the database statistics, refer to the documentation for your database.

Reduce the number of emitted events

In WebSphere Integration Developer, you can define the logging of activities or processes at a very detailed level. Activity audit events are relevant for reporting only if events are also generated for the process that contains the activity. Activity events that cannot be associated with a process are ignored by the event collector application, and they are not stored in the database. To reduce the number of emitted events, perform the following steps:

1. Select the process templates that you want to audit, and disable the emission of events for processes that you are not interested in.
2. Select the activities of this process template that you want to audit. Check whether you can omit some of the events without impacting your report results.

To get an accurate picture of an activity or a process, you should either audit all or none of the event types.

Use the SQL user-defined functions implementation

To create reports, you must install some specific user-defined functions (UDFs) in the Business Process Choreographer Explorer reporting database. The UDFs are provided as an SQL-based implementation and a Java-based implementation. The SQL implementation performs faster than the Java implementation, but has some disadvantages. If you are using the Java implementation, consider switching to the SQL implementation.

For more information about the advantages and disadvantages of the SQL and Java implementations, read about selecting a user-defined function.

Increase timeout values

It can take a long time to generate a report. If it takes too long, a transaction timeout or a connection timeout of the JDBC driver might occur. If this happens, increase the timeout values as follows:

1. In the administrative console, navigate to **Servers** → **Application servers** → *server_name* → **Transaction Service**.
2. If the **Total transaction lifetime timeout** value is less than the **Maximum transaction timeout** value, make it the same.
3. If you are still experiencing performance problems, set the **Total transaction lifetime timeout** value to 0 and increase the **Maximum transaction timeout** value.
4. If you are still experiencing performance problems, set both the **Total transaction lifetime timeout** and the **Maximum transaction timeout** values to 0, and increase the value of the connection timeout for the JDBC driver. To do this, navigate to the connection pool properties for your data source under **JDBC** → **JDBC providers** > *JDBC provider* → **Data sources** → *data_source_name* → **Connection pool properties**, and increase the **Connection timeout** value.

In a server cluster, you must adjust the transaction timeout values for all of the cluster members.

Delete unnecessary data

The report performance depends on the amount of instance and event data in the reporting database. Performance is reduced if large amounts of data are queried to produce the report. Report performance can improve if you reduce the number of process and activity instances that are in the reporting database. Regularly deleting unnecessary or old information can help to improve performance.

Chapter 8. Advanced tuning

Use the following topics to configure advanced tuning.

Tracing and monitoring considerations

The ability to configure tracing and monitoring at different levels for a variety of system components has proven to be extremely valuable during periods of system analysis or debugging. The WebSphere BPM product set provides rich monitoring capabilities, both in terms of business monitoring via the Common Event Interface (CEI) and audit logging, and system performance monitoring via PMI and ARM. While these capabilities provide insight into the performance of the running solution, these features can degrade overall system performance and throughput. **Therefore, use tracing and monitoring judiciously and when possible, turn it off entirely to ensure optimal performance.**

Most tracing and monitoring is controlled via the WebSphere Application Server administrative console. Make sure that the appropriate level of tracing/monitoring is set for PMI monitoring, logging, and tracing via the administrative console.

Use the administrative console to check that the Audit logging and Common Event Infrastructure logging check boxes are clear in the Business Process container, unless these capabilities are required for business reasons. For performance work, it is recommended that monitoring be turned off entirely unless the data being recorded is of value.

The WebSphere Integration Developer (WID) is also used to control event monitoring. Check the event monitor tab for your components and business processes to ensure that event monitoring is applied carefully.

Configure WorkArea Service's maximum send and receive

When making Synchronous SCA calls across the JVM, the underlying infrastructure serializes the workarea in the request, and then deserializes the workarea from the reply. The default workarea service as well as the workarea partition service makes use of the user specified size limits as an upper limit for sizes that can be sent or received by a workarea. The possible values are 0 (no limit), -1 (default) or a non-zero positive number. It has been found that setting the sizes to "0" (no limit) is beneficial to performance as it circumvents the costly checking of the outbound or incoming workarea size.

Disable the CEI emitter validateEvent property

The default behavior of CEI event emitters is to validate the events to ensure that all fields are correctly filled in. This is unnecessary for events generated by the WebSphere Process Server for z/OS system. To disable event validation, a custom property needs to be added to the emitter factory profile.

To disable event validation, open the administrative console and navigate to **Service Integration -> Common Event Infrastructure -> Event Emitter Factories -> Default Common Event Infrastructure emitter**. Add a new property with the name **validateEvent** and set the value to **false**.

Chapter 9. Tuning for large objects

Use the following topics to configure tuning for large objects.

Reduce or eliminate other processing while processing a large object

One way to allow for larger object sizes is to limit concurrent processing within the JVM. You must not expect to process a steady stream of the largest objects possible concurrently with other WebSphere Process Server for z/OS and WebSphere adapter activities. The operational assumption that needs to be made when considering large objects is that not all objects are “large” or “very large” and that large objects do not arrive very often, perhaps once or twice per day. If more than one very large object is being processed concurrently the likelihood of failure increases dramatically.

The size and number of the normally arriving smaller objects affects the amount of Java heap memory consumption in the system. In general the heavier the load on a system when a large object is being processed the more likely memory problems will occur.

Chapter 10. Tuning for maximum concurrency

For most high volume deployments on server-class hardware, there are many operations which take place simultaneously. Tuning for maximum concurrency ensures that the server accepts enough load to saturate its CPU(s). One sign of an inadequately tuned configuration is when additional load does not result in additional CPU utilization, while the CPUs are not fully used. To optimize these operations for maximum concurrency, the general guideline is to follow the execution flow and remove bottlenecks one at a time.

Note: Higher concurrent processing means higher resource requirements (memory and number of threads) on the server. It needs to be balanced with other tuning objectives, such as the handling of large objects, handling large numbers of users, and providing good response time.

Tune edge components for concurrency

The first step is to ensure that Business Objects are handled concurrently at the edge components of WebSphere Process Server. If the input BOs come from the adapter, make sure the adapter is tuned for concurrent delivery of input messages.

If the input BOs come from WebServices export binding or direct invocation from JSP or Servlets, make sure the WebContainer thread pool is right sized. If the input BOs come from the messaging, the ActivationSpec (MDB bindings) and Listener ports (MQ or MQJMS bindings) need to be tuned.

MDB ActivationSpec

JMS activation specifications are used to bind application MDBs to the message queues which drive the MDB onMessage methods. Two activation specification parameters of particular interest are the maximum batch size and the maximum concurrent endpoints.

For each JMS export component there is an MDB and its corresponding ActivationSpec (JNDI name: module name/export component name_AS). The maximum concurrent endpoints specifies the number of messages a particular MDB can process concurrently, but be aware that each active MDB instance runs on a thread from the container thread pool. The default value for the JMS export MDB is 10, which means that up to 10 BOs from the JMS queue can be delivered to the MDB threads concurrently.

The maximum batch size in the activation specification also has an impact on performance. The default value is 1. The maximum batch size value determines how many messages are taken from the messaging layer and delivered to the application layer in a single step (this does not mean that this work is done within a single transaction, and therefore this setting does not influence transactional scope). Increasing this value for activation specifications associated with long-running business processes can improve the efficiency of MDB message processing.

You can configure the activation specification parameters in the administrative console by navigating to **Resources > JMS > Activation specifications > (ActivationSpec name)**.

Configure listener port

If the MQ or MQ/JMS bindings are used, a Listener Port is used to configure the delivery of inbound messages to WebSphere Process Server for z/OS. In this case, the Maximum sessions property serves an identical purpose as the maxConcurrency parameter in the ActivationSpec for JMS bindings. Increase this value to an appropriate value.

Note: A listener port is created when the SCA module containing the MQ or MQ/JMS binding is deployed to the server.

Configure thread pool sizes

The sizes of thread pools have a direct impact on the ability of a server to run applications concurrently. For maximum concurrency, the thread pool sizes must be set to optimal values. Increasing the maxConcurrency or Maximum sessions parameters only enables the concurrent delivery of BOs from the JMS or MQ queues. In order for the WebSphere Process Server for z/OS server to process multiple requests concurrently, you must also increase the corresponding thread pool sizes to allow higher concurrent execution of these Message Driven Bean (MDB) threads.

MDB work is dispatched to threads allocated from the default thread pool. Note that all MDBs in the application server share this thread pool, unless a different thread pool is specified. This means that the default thread pool size must be significantly larger than the maxConcurrency of any individual MDB.

Threads in the Web Container thread pool are used for handling incoming HTTP and Web Services requests. This thread pool is shared by all applications deployed on the server and must be tuned to a higher value than the default.

ORB thread pool threads are employed for running ORB requests, for example remote EJB calls. The thread pool size must be large enough to handle requests coming in through the EJB interface, for example certain human task manager APIs.

Configure dedicated thread pools for MDBs

The default thread pool is shared by many WebSphere Application Server tasks. It is sometimes necessary to separate the execution of JMS MDBs to a dedicated thread pool. Follow the steps below to change the thread pool used for JMS MDB threads.

1. Create a new thread pool, for example MDBThreadPool, on the server.
2. Open the Service Integration Bus (SIB) JMS Resource Adapter administrative panel with server scope from **Resources > Resource Adapters > Resource adapters**. If the adapter is not shown, navigate to **Preferences** and set the **Show built-in resources** check box.
3. Repeat steps 1 and 2 for SIB JMS Resource Adapters with node and cell scope.
4. Restart the server for the change to be effective.

SCA Module MDBs for asynchronous SCA calls use a separate resource adapter, the Platform Messaging Component SPI Resource Adapter. Follow the same step as above to change the thread pool to a different one.

Note: Even with a dedicated thread pool, all MDBs associated with the resource adapter still share the same thread pool. However, they do not have to compete with other WAS tasks that also use the default thread pool.

Tune intermediate components for concurrency

If the input BO is handled by a single thread from end to end, the tuning for the edge components is normally adequate. In many situations, however, there are multiple thread switches during the end to end execution path. It is important to tune the system to ensure adequate concurrency for each asynchronous segment of the execution path.

Asynchronous invocations of an SCA component use an MDB to listen for incoming events that arrive in the associated input queue. Each SCA module defines an MDB and its corresponding activation specification (JNDI name: `sca/module name/ActivationSpec`).

Note: The SCA module MDB is shared by all asynchronous SCA components within the module, including SCA export components. Take this into account when configuring the `ActivationSpec`'s `maxConcurrency` property value. SCA module MDBs use the same default thread pool as those for JMS exports.

The asynchrony in a long running business process occurs at transaction boundaries. BPE defines an internal MDB and its `ActivationSpec`: `BPEInternalActivationSpec`. The `maxConcurrency` parameter needs to be tuned following the same guideline as for SCA module and JMS export MDBs (described above). There is one `BPEInternalActivationSpec` in the WebSphere Process Server server.

Configure JMS and JMS queue connection factories

Multiple concurrently running threads can cause a bottleneck with resources such as JMS and database connection pools if these resources are not tuned properly. The maximum connections pool size specifies the maximum number of physical connections that can be created in this pool. These are the physical connections to the backend resource, for example a DB2 database. Once the thread pool limit is reached, no new physical connections can be created and the requester waits until a physical connection that is currently in use is returned to the pool, or a `ConnectionWaitTimeout` exception is issued.

For example, if the maximum connections value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in connection timeout for a physical connection to become free. The threads waiting for connections to underlying resource are blocked until the connections are freed up and allocated to them by the pool manager. If no connection is freed in the specified interval, a `ConnectionWaitTimeout` exception is issued.

If maximum connections is set to 0, the connection pool is allowed to grow infinitely. This also has the side effect of causing the connection timeout value to be ignored. The general guideline for tuning connection factories is that their max connection pool size needs to match the number of concurrent threads multiplied by the number of simultaneous connections per thread.

For each JMS, MQ, or MQJMS Import, there is a Connection Factory created during application deployment (for example `SAPSAPAdpt.JMSImport_CF` in

ContactManager). The maximum connections property of the JMS Connection Factory's connection pool must be large enough to provide connections for all threads concurrently executing in the import component, the default value is 10.

Configure DataSource options

The maximum connections property of DataSources must be large enough to allow concurrent access to the databases from all threads. Typically there are a number of DataSources configured in WebSphere Process Server for z/OS servers, for example BPEDB datasource, WPSDB datasource, and Message Engine DB datasources. Set the maximum connection property for each dataSource to match the maximum concurrency of other system resources.

DataSource prepared statement cache size

The prepared statement cache optimises the processing of prepared SQL statements and callable statements.

The size of the cache can be tuned to avoid cache entries from being discarded. You can tune the size of the cache by opening the administrative console and navigating to **Resources > JDBC > Data sources > (Data-source-name) > WebSphere Application Server data source properties**

BPEL macroflows (long-running processes) make extensive use of the BPEDB data source for persisting data relevant to the process. Persisting various data results in the usage of many different statements, far more than the default capacity of the data source's cache. This results in an excessive number of cache misses, making the caching ineffective. This problem can be resolved by increasing the size of the cache.

The BPC container in WebSphere Process Server uses prepared statements extensively. The statement cache size of this DataSource must be at least 128. The number of cache discards for a data source can be monitored using the Tivoli performance viewer in the administrative console.

Chapter 11. Messaging tuning

Use the following topics to configure messaging tuning.

Choose datastore or filestore for message engines

Message Engine persistence is backed by a database. For a stand-alone configuration of WebSphere Process Server for z/OS version 6.2 the persistence storage of BPE and SCA buses can be backed by the file system (filestore). The choice of filestore must be made at profile creation time. Use the Profile Management Tool to create a new stand alone enterprise service bus profile or stand alone process server profile. Choose **Profile Creation Options > Advanced profile creation > Database Configuration**, select the check box **Use a file store for Messaging Engine (MEs)**. When this profile is used, filestores are used for BPE and SCA service integration buses.

Set data buffer sizes (discardable or cached)

The DiscardableDataBufferSize is the size in bytes of the data buffer used when processing best effort non persistent messages. The purpose of the discardable data buffer is to hold message data in memory because this data is never written to the data store for this quality of service. Messages that are too large to fit into this buffer are discarded.

The CachedDataBufferSize is the size in bytes of the data buffer used when processing all messages other than best effort non persistent messages. The purpose of the cached data buffer is to optimize performance by caching in memory data that might otherwise need to be read from the data store.

To set the DiscardableDataBufferSize and CachedDataBufferSize values, open the administrative console and navigate to **Service integration > Buses > bus name > Messaging engines > messaging engine name > Additional Properties > Custom properties**.

Move message engine data stores to a high performance DBMS

For better performance, the message engine data stores must use production quality databases, such as DB2, rather than the default Derby. Each datastore is located in its own database. For DB2, this is not optimal from an administrative point of view. There are already many databases in the system and adding four more databases increases the maintenance and tuning effort substantially. The solution proposed here uses a single DB2 database for all four data stores. The individual data stores/tables are completely separate and each message engine acquires an exclusive lock on its set of tables during startup. Each message engine uses a unique schema name to identify its set of tables.

Setting up the data stores for the messaging engines

Each of messaging engine is by default configured to use a data store in Derby. Each data store is located in its own database. When considering DB2 for z/OS this is not optimal from an administrative point of view. There are already many databases in the system and adding four more databases substantially increases the maintenance and tuning effort.

The solution proposed here uses a single DB2 for z/OS database for all four data stores. The individual data stores/tables are separated by assigning each a unique schema name to the set of tables associated with each messaging engine. This allows each messaging engine to acquire an exclusive lock on its set of tables during startup.

Create the DB2 database and load the data store schemas

Instead of having a DB2 database per messaging engine we put all messaging engines into the same database using different schemas to separate them, as is shown below.

Note: The table space names must be unique.

Table 2. Messaging engine schema names

Schema	Messaging Engine
SCASYS	LPAR01-server1.SCA.SYSTEM.box01Node01Cell.Bus
SCAAPP	LPAR01-server1.SCA.APPLICATION. box01Node01Cell.Bus
CEIMSG	LPAR01-server1.CommonEventInfrastructure_Bus
BPCMSG	LPAR01-server1.BPC.box01Node01Cell.Bus

Create one schema definition for each message engine with the following command on Windows®. <WAS Install> represents the WebSphere Process Server Installation directory and <user> represents the user name.

```
<WAS Install>\bin\sibDDLGenerator.sh -system db2 -version 8.1 -platform zOS  
-statementend ; -schema BPCMSG -user <user>createSIBSchema_BPCMSG.ddl
```

Repeat this command for each schema/messaging engine.

To distribute the database across several disks, edit the created schema definitions and put each table in a table space named after the schema used for example SCAAPP becomes SCANODE_TS, CEIMSG becomes CEIMSG_TS and so on. The schema definition must look like this after editing:

```
CREATE SCHEMA CEIMSG;  
CREATE TABLE CEIMSG.SIBOWNER (  
    ME_UUID VARCHAR(16),  
    INC_UUID VARCHAR(16),  
    VERSION INTEGER,  
    MIGRATION_VERSION INTEGER  
) IN CEIMSG_TB;  
CREATE TABLE CEIMSG.SIBCLASSMAP (  
    CLASSID INTEGER NOT NULL,  
    URI VARCHAR(2048) NOT NULL,  
    PRIMARY KEY(CLASSID)  
) IN CEIMSG_TB;
```

It is possible to provide separate table spaces for the various tables here. Optimal distribution depends on application structure and load characteristics. In this example one table space per datastore was used.

After creating all schema definitions and defined table spaces for the tables, create a database named 'SIB'.

Create the table spaces and distribute the containers across available disks by issuing the following command for a system managed table space:

For z/OS, DDL is provided with the product for tailoring by your database administrator.

Place the database log on a separate disk if possible.

Create the schema of the database by loading the four schema definitions into the database.

Create the data sources for the messaging engines

Create a data source for each message engine and configure each message engine to use the new datastore using the administrative console.

The following table shows the default state:

Table 3. Messaging engine data source names

Messaging Engine	JDBC Provider
box01-server1.SCA.SYSTEM.box01Node01Cell.Bus	Derby JDBC Provider (XA)
box01-server1.SCA.APPLICATION.box01Node01Cell.Bus	Derby JDBC Provider
box01-server1.CommonEventInfrastructure_Bus	Derby JDBC Provider
box01-server1.BPC.box01Node01Cell.Bus	Derby JDBC Provider

Create a new JDBC provider "DB2 Universal JDBC Driver Provider" for the non-XA datasources first if it is missing. The XA DB2 JDBC Driver Provider must exist if BPC was configured correctly for DB2.

Create four new JDBC datasources, one for CEI as an XA datasource, the remaining three as single-phase commit (non-XA) datasources.

The following table provides new names.

Table 4. New datasource names

Name of datasource	JNDI Name	Type of jdbc provider
CEIMSG_sib	jdbc/sib/CEIMSG	DB2 Universal (XA)
SCAAPP_sib	jdbc/sib/SCAAPPLICATION	DB2 Universal
SCASYSTEM_sib	jdbc/sib/SCASYSTEM	DB2 Universal
BPCMSG_sib	jdbc/sib/BPCMSG	DB2 Universal

Perform the following actions when you are creating a datasource:

- Clear the check box named 'Use this Data Source in container managed persistence (CMP)'

- Set a component-managed authentication alias
- Set the database name to the name used for the database created earlier for messaging for example 'SIB'
- Select a driver type : '2' or '4'. Per DB2 recommendations, use the JDBC universal driver type 2 connectivity to access local databases and type 4 connectivity to access remote databases. Note that a driver of type 4 requires a host name and valid port to be configured for the database.

Change the data stores of the messaging engines

Use the administrative console to change the data stores of the messaging engines.

In the navigation panel select **Service Integration -> Buses** and change the datastores for each bus/messaging engine displayed.

Put in the new JNDI and schema name for each datastore. Clear the **Create Tables** check box because the tables have already been created.

The server immediately restarts the message engine; the SystemOut.log shows the results of the change and also shows if the message engine starts successfully.

Restart the server and validate that all systems come up using the updated configuration.

Chapter 12. WebSphere MQ tuning

The following checklist details the major WebSphere MQ tuning considerations:

- Make judicious use of tracing
- Employ message management
- Set the CCSID
- Configure appropriate logging
- Enable MQ connection pooling in WebSphere Application Server

Make judicious use of tracing

Significant processor saving can be obtained by setting TRACE(G) OFF. Further gains can be obtained by starting the queue manager with trace off, rather than disabling it once the queue manager has started.

Employ message management

Separating short-lived and long-lived messages in different page sets can be beneficial. Small messages can be processed together in one unit of work, and if possible combine the transmission of many small messages into one larger message.

If you are using non-persistent messages, NPMSPEED(FAST) channels perform no logging if all the messages are non-persistent.

Set the CCSID

Performance can be degraded if the QMCCSID parameter is not set in the queue manager definition parameter dataset (xxxxZPRM).

Configure appropriate logging

Keep as much recovery log data as possible in the active logs on DASD. Ideally active log datasets must be allocated in low-usage volumes. The logs must be large enough so that log offloads occur at least every 30 minutes.

Enable MQ connection pooling in WebSphere Application Server

On the WebSphere MQ connection factory definition in WebSphere Application Server, the check box to Enable MQ Connection Pooling must be selected.

To do this, open the administrative console and navigate to **Resources > JMS Providers > WebSphere MQ > WebSphere MQ connection factory > (ConnFactory-name)**.

Chapter 13. Database: general tuning

Use the following topics to configure general tuning.

Place database log files on a fast disk subsystem

Databases are designed for high availability, transactional processing and recoverability. For performance reasons changes to table data cannot be written immediately to disk, these changes are made recoverable by writing to the database log. Updates are made to database log files when log buffers fill and at transaction commit time. As a result, database log files can be heavily used. More importantly, log writes often hold commit operations pending, meaning that the application is synchronously waiting for the write to complete. Therefore write access performance to the database log files is critical to overall system performance. We recommend that database log files be placed on a fast disk subsystem with write back cache.

Place logs on separate device from table space containers

A basic strategy for all database storage configurations is to place the database logs on separate physical disk devices from the table space containers. This reduces disk access contention between I/O to the table space containers and I/O to the database logs and preserves the mostly sequential access pattern for the log stream. Such separation also improves recoverability when log archival is employed.

Chapter 14. Database: DB2 specific tuning

Providing a comprehensive DB2 tuning guide is beyond the scope of this section of the information center. However, there are a few general rules that can assist in improving the performance of DB2 environments. In the sections below, we discuss these rules, and provide pointers to more detailed information.

A quick reference for DB2 performance tuning can be found in the developerWorks® article DB2 Tuning Tips for OLTP Applications.

The complete set of current DB2 tuning guidelines can be found in the DB2 Information Center.

The DB2 information center also contains a wealth of performance related information and is easily searched.

Update database statistics

It is important to update the statistics so that DB2 can optimize accesses to key tables. Statistics are used by the DB2 query optimizer to determine the access plan for evaluating a query. Statistics are maintained on tables and indexes. Examples of statistics include the number of rows in a table and the number of distinct values in a certain column of a table. DB2 8.2 contains new functionality called the DB2 automatic table maintenance feature, which runs the RUNSTATS command in the background as required to ensure that the correct statistics are collected and maintained. By default, this feature is not enabled. It can be turned on from the DB2 Control Center. Updating statistics allows the DB2 query optimizer to create better performing access plans for evaluating queries.

Set buffer pool sizes correctly

A buffer pool is an area of memory into which database pages are read, modified, and held during processing. Buffer pools improve database performance. If a needed page of data is already in the buffer pool, that page is accessed faster than if the page had to be read directly from disk. Consequently, the size of the DB2 buffer pools is critical to performance.

The amount of memory used by a buffer pool depends upon two factors: the size of buffer pool pages and the number of pages allocated. Buffer pool page size is fixed at creation time and can be set to 4, 8, 16 or 32 KB. The most commonly used buffer pool is IBMDEFAULTBP which has a 4 KB page size. Starting with version 8, DB2 recommends that the number of pages used by a buffer pool be set explicitly. This can be done using either the CREATE BUFFERPOOL or the ALTER BUFFERPOOL commands. The default number of pages is 250, resulting in a quite small total default buffer pool size of $4 \text{ KB} * 250 = 1000 \text{ KB}$.

Note: All buffer pools reside in database global memory, allocated on the database system. The buffer pools must coexist with other data structures and applications, all without exhausting available memory. In general, having larger buffer pools improves performance up to a point by reducing I/O activity. Beyond that point, allocating additional memory no longer improves performance. To choose appropriate buffer pool size settings, monitor database container I/O activity, by

using system tools or by using DB2 buffer pool snapshots. Avoid configuring large buffer pool size settings which lead to paging activity on the system.

Maintain proper table indexing

The WebSphere BPM products create a set of database indexes that are appropriate for many installations, however additional indexes are required in some circumstances. A database environment that requires additional indexes often displays performance degradation over time; in some cases the performance degradation can be profound. Environments that need additional indexes often exhibit heavy read I/O on devices holding the table space containers. To assist in determining which additional indexes could improve performance, DB2 provides the design advisor. The design advisor is available from the DB2 control center, or it can be started from a command line processor. The design advisor has the capability to help define and design indexes suitable for a particular workload.

Size log files appropriately

When using circular logging, it is important that the available log space permits dirty pages in the buffer pool to be cleaned at a reasonably low rate. Changes to the database are immediately written to the log, but a well tuned database coalesces multiple changes to a page before eventually writing that modified page back to disk. Changes recorded only in the log cannot be overwritten by circular logging. DB2 detects this condition and forces the immediate cleaning of dirty pages required to allow switching to a new log file. This mechanism protects the changes recorded in the log, however all application logging must be suspended until the necessary pages are cleaned.

DB2 avoids pauses when switching log files by proactively triggering page cleaning under control of the database level **softmax** parameter. The default value of 100 for softmax begins background cleaning activities when the gap between the current head of the log and the oldest log entry recording a change to a dirty page exceeds 100% of one log file in size. In extreme cases this asynchronous page cleaning cannot keep up with log activity, leading to log switch pauses which degrade performance.

Increasing the available log space gives asynchronous page cleaning more time to write dirty buffer pool pages and avoid log switch pauses. A longer interval between cleanings allows multiple changes to be coalesced on a page before it is written, which reduces the required write throughput by making page cleaning more efficient.

Available logspace is governed by the product of log file size and the number primary log files, which are configured at the database level. **logfilsiz** is the number of 4K pages in each log file. **logprimary** controls the number of primary log files. Use 10 primary log files which are large enough that they do not wrap for at least a minute in normal operation.

Increasing the primary log file size does have implications for database recovery. Assuming a constant value for softmax, larger log files mean that recovery can take more time. The softmax parameter can be lowered to counter this, but keep in mind that more aggressive page cleaning can also be less efficient. Increasing softmax gives additional opportunities for write coalescing at the cost of longer recovery time.

Chapter 15. Advanced Java Heap Tuning

Because the WebSphere BPM product set is written in Java, the performance of the Java Virtual Machine (JVM) has a significant impact on the performance delivered by these products. JVMs externalize multiple tuning parameters that can be used to improve both tooling and runtime performance. The most important of these are related to garbage collection and setting the Java heap size. This section details these topics.

Note: the products covered in this report use IBM® JVMs on most platforms, only Java 5 tunings are discussed here.

Following is a link to the IBM Java 5 Diagnostics Guide. The guide referenced above discusses many more tuning parameters than those discussed in this report, but most are for specific situations and are not of general use. For a more detailed description of IBM Java 5 garbage collection algorithms, see the section "Garbage Collectors" in the chapter titled "Understanding the IBM JDK for Java."

Monitoring garbage collection

To set the heap correctly you must determine how the heap is being used. This is easily done by collecting a verbosegc trace. A verbosegc trace prints garbage collection actions and statistics to stderr. The verbosegc trace is activated by using the Java run-time option verbosegc. Sample output is shown below.

Example verbosegc trace output

```
<af type="tenured" id="12" timestamp="Fri Jan 18 15:46:15 2008" intervalms="86.539">
  <minimum requested_bytes="3498704" />
  <time exclusiveaccessms="0.103" />
  <tenured freebytes="80200400" totalbytes="268435456" percent="29" >
    <soa freebytes="76787560" totalbytes="255013888" percent="30" />
    <loa freebytes="3412840" totalbytes="13421568" percent="25" />
  </tenured>
  <gc type="global" id="12" totalid="12" intervalms="87.124">
    <refs_cleared_soft="2" threshold="32" weak="0" phantom="0" />
    <finalization objectsqueued="0" />
    <timesms mark="242.029" sweep="14.348" compact="0.000" total="256.598" />
    <tenured freebytes="95436688" totalbytes="268435456" percent="35" >
      <soa freebytes="87135192" totalbytes="252329472" percent="34" />
      <loa freebytes="8301496" totalbytes="16105984" percent="51" />
    </tenured>
  </gc>
  <tenured freebytes="91937984" totalbytes="268435456" percent="34" >
    <soa freebytes="87135192" totalbytes="252329472" percent="34" />
    <loa freebytes="4802792" totalbytes="16105984" percent="29" />
  </tenured>
  <time totalms="263.195" />
</af>
```

It is tedious to parse the verbosegc output using a text editor. There are very good visualization tools on the Web that can be used for more effective Java heap analysis. The IBM Pattern Modeling and Analysis Tool (PMAT) for Java Garbage Collector is one such tool. It is available for free download from the IBM alphaWorks.

Setting the heap size for most configurations

This section contains guidelines for determining the appropriate Java heap size for most configurations. For most production applications, the IBM JVM Java heap size defaults are too small and must be increased. In addition, if the `-Xgcpolicy:gencon` option is used, the nursery size must be increased.

There are several approaches to setting optimal heap sizes; we outline one useful approach here. Set the initial heap size (`-Xms` option) to a reasonable value (for example, 256 MB), and the maximum heap size (`-Xmx`) option to something reasonable, but large (for example, 1024 MB). The maximum heap size must never force the heap to page. It is imperative that the heap always stays in physical memory. The JVM then tries to keep the GC time within reasonable limits by growing and shrinking the heap. The output from `verbosegc` must then be used to monitor GC activity.

If Generational Concurrent GC is used (`-Xgcpolicy:gencon`), the new area size can also be set to specific values. By default, the new size is a quarter of the total heap size or 64 MB, whichever is smaller. To improve performance, the nursery size must be a quarter of the heap size or larger, and it must not be capped at 64MB. New area sizes are set by JVM options: `-Xmn <size>`, `-Xmns<initialSize>`, and `-Xmnx<maxSize>`.

After the initial heap sizes are set, `verbosegc` traces must then be used to monitor GC activity. After analyzing the output, modify the heap settings accordingly. For example, if the percentage of time in GC is high and the heap has grown to its maximum size, throughput can be improved by increasing the maximum heap size. As a rule of thumb, greater than 10% of the total time spent in GC is generally considered high.

Note: Increasing the maximum size of the Java heap can not always solve this type of problem as it could be a memory over-usage problem. Conversely, if response times are too long due to GC pause times, decrease the heap size. If both problems are observed, an analysis of the application heap usage is required.

Setting the heap size when running multiple JVMs on one system

Each running Java program has a heap associated with it. Therefore if you have a configuration where more than one Java program is running on a single physical system, setting the heap sizes appropriately is of particular importance. If the sum of all of the virtual memory usage (including both Java heaps as well as all other virtual memory allocations) exceeds the size of physical memory, the Java heaps are subject to paging. As previously noted, this causes total system performance to degrade significantly. To minimize the possibility of this occurring, use the following guidelines:

- Collect a `verbosegc` trace for each running JVM
- Based on the `verbosegc` trace output, set the initial heap size to a relatively low value. For example, assume that the `verbosegc` trace output shows that the heap size grows quickly to 256 MB, and then grows more slowly to 400 MB. Based on this, set the initial heap size to 256 MB (`ms256m`)
- Based on the `verbosegc` trace output, set the maximum heap size appropriately. Do not set this value too low, or out of memory errors occur; the maximum heap size must be large enough to allow for peak throughput. Using the above example, a maximum heap size of 768 MB might be appropriate (`mx768m`)

- Do not set the heap sizes too low, because this causes garbage collections to occur frequently, which might reduce throughput. A `verbosegc` trace assists in determining this. A balance must be struck so that the heap sizes are large enough that garbage collections do not occur too often, while still ensuring that the heap sizes are not cumulatively so large as to cause the heap to page.

Reduce or increase heap size if out of memory errors occur

The `java.lang.OutOfMemory` exception is used by the JVM in a variety of circumstances, making it sometimes difficult to track down the source of the exception. There is no conclusive mechanism for telling the difference between these potential error sources, but a good start is to collect a trace using `verbosegc`. If the problem is a lack of memory in the heap, then this is easily seen in this output.

If, however, there is enough free memory when the `java.lang.OutOfMemory` exception is thrown, the next item to check is the finalizer count from the `verbosegc`. If these appear high then a subtle effect can be occurring whereby resources outside the heap are held by objects within the heap and being cleaned by finalizers. Reducing the size of the heap can alleviate this situation, by increasing the frequency with which finalizers are run. In addition, examine your application, to determine if the finalizers can be avoided, or minimized.

Note that Out Of Memory errors can also occur for issues unrelated to JVM heap usage, such as running out of certain system resources.. Examples of this include insufficient file handles or thread stack sizes that are too small.

In some cases, you can tune the configuration to avoid running out of native heap: try reducing the stack size for threads (the `-Xss` parameter). However, deeply nested methods can force a thread stack overflow in case of insufficient stack size.

For middleware products, if you are using an in-process version of the JDBC driver, it is usually possible to find an out-of-process driver that can have a significant effect on the native memory requirements. For example, you can use Type 4 JDBC drivers, MQSeries can be switched from Bindings mode to Client mode, and so on. See the documentation for the products in question for more details. For further information on tuning Java Heap for WAS 6.1 z/OS , see the WAS for z/OS Version 6.1 Tuning guide.

Chapter 16. Common references

The following list provides some useful reference material that can assist you when performing tuning activities with WebSphere Process Server.

1. Messaging best practices
2. DB2 best practices
3. DB2 tuning
4. DB2 Administration Guide: Performance:
 - <http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp>
 - http://www-306.ibm.com/software/data/db2/support/db2_9
5. Red Paper for Websphere Business Process Management 6.1 Performance Tuning
6. Business Process Choreographer (BPC) Tuning Paper for WebSphere Process Server 6.1
7. WebSphere Application Server Performance URL
8. Tuning Performance section of the WebSphere Application Server v6.1 Info Center (Includes tuning for messaging)
9. WebSphere Process Server V6 – Business Process Choreographer: Performance Tuning of Human Workflows Using Materialized Views
10. Modeling Efficient BPEL Processes (Document)
11. Using J2CA Adapters with WebSphere Enterprise Service Bus
12. Guided Tour of WID Part 1
13. WebSphere Enterprise Service Bus Support
14. Web Services Architectures and Best Practices
15. Best Practices for Web Services. Part1-Basics Papers 9 & 10 cover performance aspects.
16. Best Practices for Web Services. Part9- Performance considerations
17. Best Practices for Web Services. Part10- Other considerations which influence performance
18. WebSphere Journal article on dynamic caching services
19. IBM Java 5.0 Diagnostic Guide
20. Redbook: Production Topologies for WPS and WebSphere ESB V6

Chapter 17. z/OS specific references

The following sections list reference material that are useful when performing WebSphere Process Server performance tuning activities.

WebSphere performance best practices and resources

1. Performance Harness for Java Messaging Service

WebSphere Application Server reference material

1. WebSphere Application Server Performance Best Practices and Resources
2. WebSphere Process Integration Version 6.1.0 information center
3. Messaging best practices
4. WebSphere Application Server Performance URL
5. WebSphere Application Server for z/OS Version 6.1.0 product library page
6. WebSphere Application Server for z/OS Tuning guide Version 6.1.0
7. Tuning Performance section of the WebSphere Application Server for z/OS V6.1.0 information center (Includes tuning for messaging)

DB2 and DB2 for z/OS reference material

1. DB2 for z/OS and OS/390 product page
2. DB2 Universal Database for z/OS product overview
3. Best practices for tuning DB2 UDB V8.1 and its databases
4. A quick reference on DB2 tuning
5. DB2 Administration Guide: Performance
6. DB2 for z/OS and WebSphere for z/OS Redbook

WebSphere MQ reference material

1. WebSphere MQ for z/OS library
2. Performance Tuning for Java Messaging Service on WebSphere Application Server on z/OS
3. How the maximum sessions property on the listener port affects WebSphere Application Server performance
4. WebSphere MQ for z/OS product page

WebSphere Process Server and WebSphere Enterprise Service Bus reference material

1. z/OS Technical Overview: WebSphere Process Server and WebSphere Enterprise Service Bus
2. WebSphere Enterprise Service Bus Information Center
3. Getting Started with WebSphere Enterprise Service Bus V6
4. WebSphere Enterprise Service Bus and WebSphere Enterprise Service Bus version 6.0.x for z/OS Configuration Tools and Samples

WebSphere Integration Developer reference material

1. WebSphere Integration Developer information center
2. Modeling Efficient BPEL Processes

Web services reference material

1. Web Services Architectures and Best Practices
2. Best Practices for Web Services. Part9- Performance considerations
3. Best Practices for Web Services. Part10- Other considerations which influence performance
4. WebSphere Journal article on dynamic caching services

z/OS and System z reference material

1. z/OS Internet Library

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of

this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (or), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are registered trademark of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project
(<http://www.eclipse.org>).



IBM WebSphere Process Server for Multiplatforms, Version 6.1.2

Index



Printed in USA