IBM WebSphere Application Server Network Deployment for IBM i, Version 8.0

Tuning guide

Note

Before using this information, be sure to read the general information under “Notices” on page 61.

Compilation date: July 16, 2011

Contents

How to send your comments v

Changes to serve you more quickly vii

Chapter 1. Planning for performance 1
Application design consideration 1

Chapter 2. Taking advantage of performance functions 5

Chapter 3. Obtaining advice from the advisors 7
Why you want to use the performance advisors 7
Performance advisor types and purposes 8
Using the Performance and Diagnostic Advisor 12
Performance and Diagnostic Advisor configuration settings 13
Advice configuration settings 15
Viewing the Performance and Diagnostic Advisor recommendations 16
Starting the lightweight memory leak detection 17
Enabling automated heap dump generation 18
Using the performance advisor in Tivoli Performance Viewer 19
Performance advisor report in Tivoli Performance Viewer 20
Activating the heap monitor 20
Heap monitor default operation 21

Chapter 4. Tuning the application serving environment 23
Tuning parameter hot list 23
Directory conventions 24
Tuning TCP/IP buffer sizes 26
Tuning the JVM 27
Tuning the IBM virtual machine for Java 27
Directory conventions 36
Tuning transport channel services 37
Checking hardware configuration and settings 42
Tuning operating systems 43
Tuning IBM i systems 43
Tuning web servers for IBM i 44
Tuning web servers 45
Using Collection Services performance data 46
The manageWASCollectionServices script 48
processStats script 48
Directory conventions 49
Tuning the application server using pre-defined tuning templates 51

Chapter 5. Troubleshooting performance problems 55

Appendix. Directory conventions 59

Notices 61

Trademarks and service marks 63

Index 65

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-5250.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

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.

Changes to serve you more quickly

Print sections directly from the information center navigation

PDF books are provided as a convenience format for easy printing, reading, and offline use. The information center is the official delivery format for IBM WebSphere Application Server documentation. If you use the PDF books primarily for convenient printing, it is now easier to print various parts of the information center as needed, quickly and directly from the information center navigation tree.

To print a section of the information center navigation:

1. Hover your cursor over an entry in the information center navigation until the **Open Quick Menu** icon is displayed beside the entry.
2. Right-click the icon to display a menu for printing or searching your selected section of the navigation tree.
3. If you select **Print this topic and subtopics** from the menu, the selected section is launched in a separate browser window as one HTML file. The HTML file includes each of the topics in the section, with a table of contents at the top.
4. Print the HTML file.

For performance reasons, the number of topics you can print at one time is limited. You are notified if your selection contains too many topics. If the current limit is too restrictive, use the feedback link to suggest a preferable limit. The feedback link is available at the end of most information center pages.

Under construction!

The Information Development Team for IBM WebSphere Application Server is changing its PDF book delivery strategy to respond better to user needs. The intention is to deliver the content to you in PDF format more frequently. During a temporary transition phase, you might experience broken links. During the transition phase, expect the following link behavior:

- Links to Web addresses beginning with http:// work
- Links that refer to specific page numbers within the same PDF book work
- The remaining links will *not* work. You receive an error message when you click them

Thanks for your patience, in the short term, to facilitate the transition to more frequent PDF book updates.

Chapter 1. Planning for performance

How well a website performs while receiving heavy user traffic is an essential factor in the overall success of an organization. This section provides online resources that you can consult to ensure that your site performs well under pressure.

Procedure

- Consult the following web resources for learning.

  IBM® Patterns for e-Business

  IBM Patterns for e-business is a group of reusable assets that can help speed the process of developing Web-based applications. The patterns leverage the experience of IBM architects to create solutions quickly, whether for a small local business or a large multinational enterprise.

  Planning for availability in the enterprise

  Availability is an achievable service-level characteristic that every enterprise struggles with. The worst case scenario is realized when load is underestimated or bandwidth is overloaded because availability planning was not carefully conducted. Applying the information in this article and the accompanying spreadsheet to your planning exercises can help you avoid such a scenario.

  Hardware configurations for WebSphere® Application Server production environments
  This article describes the most common production hardware configurations, and provides the reasons for choosing each one. It begins with a single machine configuration, and then proceeds with additional configurations that have higher fault tolerance, horizontal scaling, and a separation of web and enterprise bean servers.
- Take advantage of performance functions to improve performance. You can use functions such as balancing workloads with clusters and using the dynamic cache to improve performance.

Application design consideration

This topic describes architectural suggestions for the design and tuning of applications.

The designing applications information contains the architectural suggestions and the implementation of applications. For existing applications, the suggestions might require changing the existing implementations. Tuning the application server and resource parameters can have the greatest effect on performance of the applications that are well designed.

Use designing applications considerations in this topic for tips to ensure your applications are thoughtfully designed and tuned. These considerations include websites and other ideas for finding best practices for designing WebSphere applications, particularly in the realm of WebSphere extensions to the Java Platform, Enterprise Edition (Java EE) specification.

best-practices: Use the following information as an architectural guide when implementing applications:

- Persistence
- Model-view-controller pattern
- Statelessness
- Caching
- Asynchronous considerations
- Third-party libraries

Persistence

Java EE applications load, store, create, and remove data from relational databases, a process commonly referred to as *persistence*. Most enterprise applications have significant database access. The architecture and performance of the persistence layer is critical to the performance of an application. Therefore, persistence is a very important area to consider when making architectural choices that require trade-offs related to performance. This guide recommends first focusing on a solution that has clean architecture. The clean architecture considers data consistency, security, maintenance, portability, and the performance of that solution. Although this approach might not yield the absolute peak performance obtainable from manual coding a solution that ignores the mentioned qualities of service, this approach can achieve the appropriate balance of data consistency, maintainability, portability, security, and performance.

Multiple options are available in Java EE for persistence: Session beans using entity beans including container-managed persistence (CMP) or bean-managed persistence (BMP), session beans using Java Database Connectivity (JDBC), and Java beans using JDBC. For the reasons previously mentioned, consider CMP entity persistence because it provides maximum security, maintenance, and portability. CMP is also recommended for good performance. Refer to the Tune the EJB container section of the tuning application servers topic on tuning enterprise beans and more specifically, CMP.

If an application requires using enterprise beans not using EJB entities, the persistence mechanism usually involves the JDBC API. Because JDBC requires manual coding, the Structured Query Language (SQL) that runs against a database instance, it is critical to optimize the SQL statements that are used within the application. Also, configure the database server to support the optimal performance of these SQL statements. Finally, usage of specific JDBC APIs must be considered including prepared statements and batching.

Regardless of which persistence mechanism is considered, use container-managed transactions where the bean delegates management of transactions to the container. For applications that use JDBC, this is easily achieved by using the session façade pattern, which wraps all JDBC functions with a stateless session bean.

Finally, information about tuning the connection over which the EJB entity beans or JDBC communicates can be found in the Tune the data sources section of the tuning application servers topic.

Model-view-controller pattern

One of the standard Java EE programming architectures is the model-view-controller (MVC) architecture, where a call to a controller servlet might include one or more child JavaServer Pages (JSP) files to construct the view. The MVC pattern is a recommended pattern for application architecture. This pattern requires distinct separation of the view (JSP files or presentation logic), the controller (servlets), and the model (business logic). Using the MVC pattern enables optimization of the performance and scalability of each layer separately.

Statelessness

Implementations that avoid storing the client user state scale and perform the best. Design implementations to avoid storing state. If state storage is needed, ensure that the size of the state data and the time that the state is stored are kept to the smallest possible values. Also, if state storage is needed, consider the possibility of reconstructing the state if a failure occurs, instead of guaranteeing state failover through replication.

Specific tuning of state affects HTTP session state, dynamic caching, and enterprise beans. Refer to the follow tuning guides for tuning the size, replication, and timing of the state storage:
- Session management tuning
- EJB tuning tips
- Tuning dynamic cache with the cache monitor

Caching

Most Java EE application workloads have more read operations than write operations. Read operations require passing a request through several topology levels that consist of a front-end web server, the web container of an application server, the EJB container of an application server, and a database. WebSphere Application Server provides the ability to cache results at all levels of the network topology and Java EE programming model that include web services.

Application designers must consider caching when the application architecture is designed because caching integrates at most levels of the programming model. Caching is another reason to enforce the MVC pattern in applications. Combining caching and MVC can provide caching independent of the presentation technology and in cases where there is no presentation to the clients of the application.

Network designers must consider caching when network planning is performed because caching also integrates at most levels of the network topology. For applications that are available on the public Internet, network designers might want to consider Edge Side Include (ESI) caching when WebSphere Application Server caching extends into the public Internet. Network caching services are available in the proxy server for WebSphere Application Server, WebSphere Edge Component Caching Proxy, and the WebSphere plug-in.

Asynchronous considerations

Java EE workloads typically consist of two types of operations. You must perform the first type of operation to respond to a system request. You can perform the second type of operation asynchronously after the user request that initiated the operation is fulfilled.

An example of this difference is an application that enables you to submit a purchase order, enables you to continue while the system validates the order, queries remote systems, and in the future informs you of the purchase order status. This example can be implemented synchronously with the client waiting for the response. The synchronous implementation requires application server resources and you wait until the entire operations complete. If the process enables you to continue, while the result is computed asynchronously, the application server can schedule the processing to occur when it is optimal in relation to other requests. The notification to you can be triggered through email or some other interface within the application.

Because the asynchronous approach supports optimal scheduling of workloads and minimal server resource, consider asynchronous architectures. WebSphere Application Server supports asynchronous programming through Java EE Java Message Service (JMS) and message-driven beans (MDB) as well as asynchronous beans that are explained in the Tuning Java Message Service and Tuning MDB topics.

Third-party libraries

Verify that all the libraries that applications use are also designed for server-side performance. Some libraries are designed to work well within a client application and fail to consider server-side performance concerns, for example, memory utilization, synchronization, and pooling. It is suggested that all libraries that are not developed as part of an application undergo performance testing using the same test methodologies as used for the application.

Additional reference:

IBM WebSphere Developer Technical Journal: The top 10 (more or less) Java EE best practices

Improve performance in your XML applications, Part 2

Chapter 2. Taking advantage of performance functions

This topic highlights a few main ways you can improve performance through a combination of product features and application development considerations.

Procedure

- Use one of the following considerations to improve performance.

  Balancing workloads with clusters

  Clusters are sets of servers that are managed together and participate in workload management. The servers that are members of a cluster can be on different host machines, as opposed to the servers that are part of the same node and must be located on the same host machine. A cell can have no clusters, one cluster, or multiple clusters.

  Using the dynamic cache service to improve performance

  The dynamic cache service improves performance by caching the output of servlets, commands, and JavaServer Pages (JSP) files. Dynamic caching features include cache replication among clusters, cache disk offload, Edge-side include caching, and external caching, which is the ability to control caches outside of the application server, such as that of your web server.

- Ensure your applications perform well.

  Take advantage of architectural suggestions and coding best practices to ensure that your applications perform well. See the information about application design considerations and the information on designing applications to learn more about ways you can improve performance of your applications.

Chapter 3. Obtaining advice from the advisors

Advisors provide a variety of recommendations that help improve the performance of your application server.

Before you begin

The advisors provide helpful performance as well as diagnostic advice about the state of the application server.

About this task

Tuning WebSphere Application Server is a critical part of getting the best performance from your website. However, tuning WebSphere Application Server involves analyzing performance data and determining the optimal server configuration. This determination requires considerable knowledge about the various components in the application server and their performance characteristics. The performance advisors encapsulate this knowledge, analyze the performance data, and provide configuration recommendations to improve the application server performance. Therefore, the performance advisors provide a starting point to the application server tuning process and help you without requiring that you become an expert.

The Runtime Performance Advisor is extended to also provide diagnostic advice and is now called the Performance and Diagnostic Advisor. Diagnostic advice provides useful information regarding the state of the application server. Diagnostic advice is especially useful when an application is not functioning as expected, or simply as a means of monitoring the health of application server.

Procedure

- Decide which performance advisor is right for the purpose, Performance and Diagnostic Advisor or Tivoli® Performance Viewer advisor.
- Use the chosen advisor to periodically check for inefficient settings, and to view recommendations.
- Analyze Performance Monitoring Infrastructure data with performance advisors.

What to do next

Additionally, you can use the heap monitor feature to monitor the Java Virtual Machine (JVM) heap size of a WebSphere Application Server profile in comparison to pool size. The feature is available for new WebSphere Application Server profiles or profiles that are created after you update to the WebSphere Application Server. For existing WebSphere Application Server profiles, there is a script available to add the feature. See the heapMonitor script information.

Why you want to use the performance advisors

The advisors analyze the Performance Monitoring Infrastructure (PMI) data of WebSphere Application Server using general performance principles, best practices, and WebSphere Application Server-specific rules for tuning. The advisors that are based on this information provide advice on how to set some of your configuration parameters to better tune WebSphere Application Server.

The advisors provide a variety of advice on the following application server resources:

- Object Request Broker service thread pools
- Web container thread pools
- Connection pool size
- Persisted session size and time
- Data source statement cache size

- Session cache size
- Dynamic cache size
- Java virtual machine heap size
- DB2® Performance Configuration wizard
- Connection use violations

For example, consider the data source statement cache. It optimizes the processing of *prepared statements* and *callable statements* by caching those statements that are not used in an active connection. (Both statements are SQL statements that essentially run repeatable tasks without the costs of repeated compilation.) If the cache is full, an old entry in the cache is discarded to make room for the new one. The best performance is generally obtained when the cache is large enough to hold all of the statements that are used in the application. The PMI counter, prepared statement cache discards, indicates the number of statements that are discarded from the cache. The performance advisors check this counter and provide recommendations to minimize the cache discards.

Another example is thread or connection pooling. The idea behind pooling is to use an existing thread or connection from the pool instead of creating a new instance for each request. Because each thread or connection in the pool consumes memory and increases the context-switching cost, the pool size is an important configuration parameter. A pool that is too large can hurt performance as much as a pool that is too small. The performance advisors use PMI information about current pool usage, minimum or maximum pool size, and the application server CPU utilization to recommend efficient values for the pool sizes.

The advisors can also issue diagnostic advice to help in problem determination and health monitoring. For example, if your application requires more memory than is available, the diagnostic adviser tells you to increase the size or the heap for application server.

Performance advisor types and purposes

Two performance advisors are available: the Performance and Diagnostic Advisor and the performance advisor in Tivoli Performance Viewer.

The Performance and Diagnostic Advisor runs in the Java virtual machine (JVM) process of application server; therefore, it does not provide expensive advice. In a stand-alone application server environment, the performance advisor in Tivoli Performance Viewer runs within the application server JVM.

The performance advisor in Tivoli Performance Viewer provides advice to help tune systems for optimal performance and provide recommendations on inefficient settings by using collected Performance Monitoring Infrastructure (PMI) data. Obtain the advice by selecting the performance advisor in Tivoli Performance Viewer.

In a WebSphere Application Server, Network Deployment environment, the performance advisor in Tivoli Performance Viewer runs within the JVM of the node agent and can provide advice on resources that are more expensive to monitor and analyze. The Tivoli Performance Viewer advisor requires that you enable performance modules, counters, or both.

Table 1. Performance and Diagnostic Advisor and Tivoli Performance Viewer advisor. The following chart shows the differences between the Performance and Diagnostic Advisor and the Tivoli Performance Viewer advisor:

	Performance and Diagnostic Advisor	**Tivoli Performance Viewer advisor**
Start location	Application server	Tivoli Performance Viewer client
Invocation of tool	Administrative console	Tivoli Performance Viewer
Output	• The `SystemOut.log` file • The administrative console • JMX notifications	Tivoli Performance Viewer in the administrative console

Table 1. Performance and Diagnostic Advisor and Tivoli Performance Viewer advisor (continued). The following chart shows the differences between the Performance and Diagnostic Advisor and the Tivoli Performance Viewer advisor:

Frequency of operation	Configurable	When you select refresh in the Tivoli Performance Viewer administrative console
Types of advice	Performance advice: • Object Request Broker (ORB) service thread pools • Web container thread pools • Connection pool size • Persisted session size and time • Prepared statement cache size • Session cache size • Memory leak detection Diagnostic advice: • Connection factory diagnostics • Data source diagnostics Connection usage diagnostics • Detection of connection use by multiple threads • Detection of connection use across components	Performance advice: • ORB service thread pools • Web container thread pools • Connection pool size • Persisted session size and time • Prepared statement cache size • Session cache size • Dynamic cache size • Java virtual machine (JVM) heap size • DB2 Performance Configuration wizard

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Performance and Diagnostic Advisor

Use this topic to understand the functions of the Performance and Diagnostic Advisor.

The Performance and Diagnostic Advisor provides advice to help tune systems for optimal performance and is configured using the WebSphere Application Server administrative console or the wsadmin tool. Running in the Java virtual machine (JVM) of the application server, the Performance and Diagnostic Advisor periodically checks for inefficient settings and issues recommendations as standard product warning messages. These recommendations are displayed both as warnings in the administrative console under Runtime Messages in the WebSphere Application Server Status panel and as text in the application server `SystemOut.log` file. Enabling the Performance and Diagnostic Advisor has minimal system performance impact.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

The Performance and Diagnostic Advisor provides performance advice and diagnostic advice to help tune systems for optimal performance, and also to help understand the health of the system. It is configured using the WebSphere Application Server administrative console or the wsadmin tool. Running in the Java virtual machine (JVM) of the application server, the Performance and Diagnostic Advisor periodically checks for inefficient settings and issues recommendations as standard product warning messages. These recommendations are displayed as warnings in the administrative console under Runtime Messages in the WebSphere Application Server Status panel, as text in the application server `SystemOut.log` file, and as Java Management Extensions (JMX) notifications. Enabling the Performance and Diagnostic Advisor has minimal system performance impact.

From WebSphere Application Server, Version 6.0.2, you can use the Performance and Diagnostic Advisor to enable the lightweight memory leak detection, which is designed to provide early detection of memory problems in test and production environments.

The advice that the Performance and Diagnostic Advisor gives is all on the server level. The only difference when running in a WebSphere Application Server, Network Deployment environment is that you might receive contradictory advice on resources that are declared at the node or cell level and used at the server level.

For example, two sets of advice are given if a data source is declared at the node level to have a connection pool size of {10,50} and is used by two servers (server1 and server2). If server1 uses only two connections and server2 uses all fifty connections during peak load, the optimal connection pool size is different for the two servers. Therefore, the Performance and Diagnostic Advisor gives two sets of advice (one for server1 and another for server2). The data source is declared at the node level and you must make your decisions appropriately by setting one size that works for both, or by declaring two different data sources for each server with the appropriate level.

Read the using the performance and diagnostic advisor information for startup and configuration steps.

Diagnostic alerts:

In WebSphere Application Server Version 8.0 the Performance and Diagnostic Advisors are extended to provide more diagnostic alerts to help common troubleshoot problems.

Several alerts are made available to monitor connection factory and data sources behavior. Some of these alerts are straightforward and easy to comprehend. Others are much more involved and are intended for use by IBM support only.

ConnectionErrorOccured diagnostic alert

When a resource adapter or data source encounters a problem with connections such that the connection might no longer be usable, it informs the connection manager that a connection error occurred. This causes the destruction of the individual connection or a pool purge, which is the destruction of all connections in the pool, depending on the pool purge policy configuration setting. An alert is sent, indicating a potential problem with the back-end if an abnormally high number of unusable connections are detected.

Connection low-percent efficiency diagnostic alert

If the percentage of time that a connection is used versus held for any individual connections drops below a threshold, an alert is sent with a call stack.

Cross-Component Use JCA Programming Model Violation Diagnostic Alert

When you enable cross-component use detection, the application server raises an alert when a connection handle is used by a Java EE application component that is different from the component that originally

acquired the handle through a connection factory. This condition might inadvertently occur if an application passes a connection handle in a parameter or an application obtains a handle from a cache that is shared by multiple application components. If components use a connection handle in this manner, this might result in problems with application or data integrity. Enable the alert to detect the cross-component connection use during development to identify and avoid potential application problems.

Local transaction containment (LTC) nesting threshold exceeded diagnostic alert

For LTC definition, see the Local transaction containment (LTC) and Transaction type and connection behavior information, and Default behavior of managed connections in WebSphere Application Server topic.

If a high number of LTCs are started on a thread before completing, an alert is raised. This alert is useful in debugging some situations where the connection pool is unexpectedly running out of connections due to multiple nested LTCs holding onto multiple shareable connections.

Multi-Thread Use JCA Programming Model Violation Diagnostic Alert

Multi-thread use detection raises an alert when an application component acquires a connection handle using a connection factory, and then the component uses the handle on a different thread from which the handle was acquired. If you use a connection in this manner, this behavior might cause data integrity problems, especially if an application uses a connection handle on a thread that is not managed. Enable the alert to detect multi-thread connection usage during application development.

Pool low-percent efficiency diagnostic alert

If the average time that a connection is held versus used for the all connections in the pool drops below a threshold, an alert is sent.

Serial reuse violation diagnostic alert

For information on what serial reuse is, see the transaction type and connection behavior information. Some legitimate scenarios exist, where a serial reuse violation is appropriate, but in most cases this violation is not intended and might lead to data integrity problems.

If this alert is enabled, any time a serial reuse violation occurs within an LTC, an alert is sent.

Surge mode entered or exited diagnostic alert

When surge mode is configured, an alert is sent whenever surge mode engages or disengages. See the surge mode documentation for more information.

Stuck connection block mode entered or exited diagnostic alert

When stuck connection detection is configured, an alert is sent whenever stuck connection blocking starts or stops. See the stuck connection information.

Thread maximum connections exceeded diagnostic alert

When one or more LTCs on a thread ties too many managed connections, or poolable connections for data sources an alert is issued.

Using the Performance and Diagnostic Advisor

The advisors analyze the Performance Monitoring Infrastructure (PMI) data of WebSphere Application Server using general performance principles, best practices, and WebSphere Application Server-specific rules for tuning.

Procedure

1. Ensure that PMI is enabled, which is default. If PMI is disabled, see the enabling PMI using the administrative console information. To obtain advice, you must first enable PMI through the administrative console and restart the server. The Performance and Diagnostic Advisor enables the appropriate monitoring counter levels for all enabled advice when PMI is enabled. If specific counters exist that are not wanted, or when disabling the Performance and Diagnostic Advisor, you might want to disable PMI or the counters that the Performance and Diagnostic Advisor enabled.
2. If running WebSphere Application Server, Network Deployment, you must enable PMI on both the server and the administrative agent, and restart the server and the administrative agent.
3. Click **Servers** > **Application servers** in the administrative console navigation tree.
4. Click *server_name* > **Performance and Diagnostic Advisor Configuration**.
5. Under the **Configuration** tab, specify the number of processors on the server. This setting is critical to ensure accurate advice for the specific configuration of the system.
6. Select the **Calculation Interval**. PMI data is taken over time and averaged to provide advice. The calculation interval specifies the length of time over which data is taken for this advice. Therefore, details within the advice messages display as averages over this interval.
7. Select the **Maximum Warning Sequence**. The maximum warning sequence refers to the number of consecutive warnings that are issued before the threshold is updated. For example, if the maximum warning sequence is set to `3`, then the advisor sends only three warnings, to indicate that the prepared statement cache is overflowing. After three warnings, a new alert is issued only if the rate of discards exceeds the new threshold setting.
8. Specify **Minimum CPU for Working System**. The minimum central processing unit (CPU) for a working system refers to the CPU level that indicates a application server is under production load. Or, if you want to tune your application server for peak production loads that range from 50-90% CPU utilization, set this value to `50`. If the CPU is below this value, some diagnostic and performance advice are still issued. For example, regardless of the CPU level if you are discarding prepared statements at a high rate, you are notified.
9. Specify **CPU Saturated**. The CPU saturated level indicates at what level the CPU is considered fully utilized. The level determines when concurrency rules no longer increase thread pools or other resources, even if they are fully utilized.
10. Click **Apply**.
11. Click **Save**.
12. Click *server_name* > **Performance and Diagnostic Advisor Configuration**.
13. Click the **Runtime** tab.
14. Click **Restart**. Select **Restart** on the Runtime tab to reinitialize the Performance and Diagnostic Advisor using the last configuration information that is saved to disk.

    This action also resets the state of the Performance and Diagnostic Advisor. For example, the current warning count is reset to zero (0) for each message.
15. Simulate a production level load. If you use the Performance and Diagnostic Advisor in a test environment, do any other tuning for performance, or simulate a realistic production load for your application. The application must run this load without errors. This simulation includes numbers of concurrent users typical of peak periods, and drives system resources, for example, CPU and memory, to the levels that are expected in production. The Performance and Diagnostic Advisor provides advice when CPU utilization exceeds a sufficiently high level only. For a list of IBM business partners that provide tools to drive this type of load, see the performance: resource for learning information.

16. Select the check box to enable the Performance and Diagnostic Advisor.

    Tip: To achieve the best results for performance tuning, enable the Performance and Diagnostic Advisor when a stable production-level load is applied.
17. Click **OK**.
18. Select **Runtime Warnings** in the administrative console under the Runtime Messages in the Status panel or look in the `SystemOut.log` file, which is located in the following directory:

    profile_root/logs/*server_name*

    Some messages are not issued immediately.
19. Update the product configuration for improved performance, based on advice. Although the performance advisors attempt to distinguish between loaded and idle conditions, misleading advice might be issued if the advisor is enabled while the system is ramping up or down. This result is especially likely when running short tests. Although the advice helps in most configurations, there might be situations where the advice hinders performance. Because of these conditions, advice is not guaranteed. Therefore, test the environment with the updated configuration to ensure that it functions and performs better than the previous configuration.

    Over time, the advisor might issue differing advice. The differing advice is due to load fluctuations and the runtime state. When differing advice is received, you need to look at all advice and the time period over which it is issued. Advice is taken during the time that most closely represents the peak production load.

    Performance tuning is an iterative process. After applying advice, simulate a production load, update the configuration that is based on the advice, and retest for improved performance. This procedure is continued until optimal performance is achieved.

What to do next

You can enable and disable advice in the Advice Configuration panel. Some advice applies only to certain configurations, and can be enabled only for those configurations. For example, unbounded Object Request Broker (ORB) service thread pool advice is only relevant when the ORB service thread pool is unbounded, and can only be enabled when the ORB thread pool is unbounded. For more information on Advice configuration, see the advice configuration settings information.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Performance and Diagnostic Advisor configuration settings

Use this page to specify settings for the Performance and Diagnostic Advisor.

To view this administrative page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Performance and Diagnostic Advisor Configuration** under the Performance section.

Enable Performance and Diagnostic Advisor Framework

Specifies whether the Performance and Diagnostic Advisor runs on the server startup.

The Performance and Diagnostic Advisor requires that the Performance Monitoring Infrastructure (PMI) be enabled. It does not require that individual counters be enabled. When a counter that is needed by the Performance and Diagnostic Advisor or is not enabled, the Performance and Diagnostic Advisor enables it automatically. When disabling the Performance and Diagnostic Advisor, you might want to disable

Performance Monitoring Infrastructure (PMI) or the counters that Performance and Diagnostic Advisor enabled. The following counters might be enabled by the Performance and Diagnostic Advisor:

- ThreadPools (module)
  - Web Container (module)
    - Pool Size
    - Active Threads
  - Object Request Broker (module)
    - Pool Size
    - Active Threads
- JDBC Connection Pools (module)
  - Pool Size
  - Percent used
  - Prepared Statement Discards
- Servlet Session Manager (module)
  - External Read Size
  - External Write Size
  - External Read Time
  - External Write Time
  - No Room For New Session
- System Data (module)
  - CPU Utilization
  - Free Memory

Enable automatic heap dump collection

Specifies whether the Performance and Diagnostic Advisor automatically generates heap dumps for post analysis when suspicious memory activity is detected.

Calculation Interval

Specifies the length of time over which data is taken for this advice.

PMI data is taken over an interval of time and averaged to provide advice. The calculation interval specifies the length of time over which data is taken for this advice. Details within the advice messages display as averages over this interval. The default value is automatically set to four minutes.

Maximum warning sequence

The maximum warning sequence refers to the number of consecutive warnings that are issued before the threshold is relaxed.

For example, if the maximum warning sequence is set to 3, the advisor only sends three warnings to indicate that the prepared statement cache is overflowing. After three warnings, a new alert is only issued if the rate of discards exceeds the new threshold setting. The default value is automatically set to one.

Number of processors

Specifies the number of processors on the server.

This setting is helpful to ensure accurate advice for the specific configuration of the system. Depending your configuration and system, there may be only one processor utilized. The default value is automatically set to two.

Minimum CPU For Working System

The minimum CPU for working system refers to the point at which concurrency rules do not attempt to free resources in thread pools.

There is a set of concurrency alerts to warn you if all threads in a pool are busy. This can affect performance, and it may be necessary for you to increase them. The CPU bounds are a mechanism to help determine when an application server is active and tunable.

The Minimum CPU for working system sets a lower limit as to when you should consider adjusting thread pools. For example, say you set this value to 50%. If the CPU is less than 50%, concurrency rules *do not* try to free up resources by decreasing pools to get rid of unused threads. That is, if the pool size is 50-100 and only 20 threads are consistently used then concurrency rules would like to decrease the minimum pool size to 20.

CPU Saturated
The CPU Saturated setting determines when the CPU is deemed to be saturated.

There is a set of concurrency alerts to warn you if all threads in a pool are busy. This can affect performance, and it may be necessary for you to increase them. The CPU bounds are a mechanism to help determine when an application server is active and tunable.

The CPU saturated setting determines when the CPU has reached its saturation point. For example, if this is set to 95%, when the CPU is greater than 95% the concurrency rules *do not* try to improve things, that is, increase the size of a thread pool.

Advice configuration settings
Use this page to select the advice you wish to enable or disable.

To view this administrative page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. Under the Performance section, click **Performance and Diagnostic Advisor Configuration** > **Performance and Diagnostic Advice Configuration**.

Advice name
Specifies the advice that you can enable or disable.

Advice applied to component
Specifies the WebSphere Application Server component to which the advice applies.

Advice type
Categorizes the primary indent of a piece of Advice.

Use Advice type for grouping, and then enabling or disabling sets of advice that is based upon your purpose. Advice has the following types:

- Performance: Performance advice provides tuning recommendations, or identifies problems with your configuration from a performance perspective.
- Diagnostic: Diagnostic advice provide automated logic and analysis relating to problem identification and analysis. These types advice are usually issued when unexpected circumstances are encountered by the application server.

Performance impact
Generalizes the performance overhead that an alert might incur.

The performance impact of a particular piece of advice is highly dependant upon the scenario being run and upon the conditions meet. The performance categorization of alerts is based upon worst case scenario measurements. The performance categorizations are:

- **Low:** Advice has minimal performance overhead. Advice might be run in test and production environments. Cumulative performance overhead is within run to run variance when all advice of this type is enabled.
- **Medium:** Advice has measurable but low performance overhead. Advice might be run within test environments, and might be run within production environments if deemed necessary. Cumulative performance overhead is less than 4% when all advice of this type is enabled.

- **High:** Advice impact is high or unknown. Advice might be run during problem determination tests and functional tests. It is not run in production simulation or production environments unless deemed necessary. Cumulative performance overhead might be significant when all advice of this type is enabled.

Advice status
Specifies whether the advice is stopped, started, or unavailable.

The advice status has one of three values: **Started**, **Stopped** or **Unavailable**.
- **Started**: The advice is enabled.
- **Stopped**: The advice is not enabled.
- **Unavailable**: The advice does not apply to the current configuration, for example, persisted session size advice in a configuration without persistent sessions.

Viewing the Performance and Diagnostic Advisor recommendations
Runtime Performance Advisor uses Performance Monitoring Infrastructure (PMI) data to provide recommendations for performance tuning.

About this task

The Performance and Diagnostic Advisor uses Performance Monitoring Infrastructure (PMI) data to provide recommendations for performance tuning. Running in the Java virtual machine (JVM) of the application server, this advisor periodically checks for inefficient settings, and issues recommendations as standard product warning messages.

Procedure

The Performance and Diagnostic Advisor recommendations are displayed in two locations:
1. The WebSphere Application Server `SystemOut.log` log file.
2. The Runtime Messages panel in the administrative console. To view this administrative page, click **Troubleshooting** > **Runtime Messages** > **Runtime Warning**.

Example

The following log file is a sample output of advice on the `SystemOut.log` file:

```
[4/2/04 15:50:26:406 EST] 6a83e321 TraceResponse W CWTUN0202W:
Increasing the web container thread pool Maximum Size to 48
might improve performance.

Additional explanatory data follows.

Average number of threads: 48.

Configured maximum pool size: 2.

This alert has been issued 1 time(s) in a row.
The threshold will be updated to reduce the
overhead of the analysis.
```

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Starting the lightweight memory leak detection

Use this task to start the lightweight memory leak detection using the Performance and Diagnostic Advisor.

Before you begin

If you have a memory leak and want to confirm the leak, or you want to automatically generate heap dumps on Java virtual machines (JVM) in WebSphere Application Server, consider changing your minimum and maximum heap sizes to be equal. This change provides the memory leak detection more time for reliable diagnosis.

About this task

To start the lightweight memory leak detection using the Performance and Diagnostic Advisor, perform the following steps in the administrative console:

Procedure

1. Click **Servers > Application servers** in the administrative console navigation tree.
2. Click *server_name* **> Performance and Diagnostic Advisor Configuration**.
3. Click the **Runtime** tab.
4. Enable the Performance and Diagnostic Advisor Framework.
5. Click **OK**.
6. From the Runtime or Configuration tab of Performance and Diagnostic Advisor Framework, click **Performance and Diagnostic Advice configuration**.
7. Start the memory leak detection advice and stop any other unwanted advice.

Results

The memory leak detection advice is started.

Important: To achieve the best results for performance tuning, start the Performance and Diagnostic Advisor when a stable production level load is running.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

What to do next

You can monitor any notifications of memory leaks by checking the `SystemOut.log` file or Runtime Messages. For more information, see the “Viewing the Performance and Diagnostic Advisor recommendations” on page 16 topic.

Lightweight memory leak detection

This topic describes memory leaks in Java applications and introduces lightweight memory leak detection.

Memory leaks in Java applications

Although a Java application has a built-in garbage collection mechanism, which frees the programmer from any explicit object deallocation responsibilities, memory leaks are still common in Java applications.

Memory leaks occur in Java applications when unintentional references are made to unused objects. This occurrence prevents Java garbage collection from freeing memory.

The term *memory leak* is overused; a memory leak refers to a memory misuse or mismanagement. Old unused data structures might have outstanding references but are never garbage collected. A data structure might have unbounded growth or there might not be enough memory that is allocated to efficiently run a set of applications.

Lightweight memory leak detection in WebSphere Application Server

Most existing memory leak technologies are based upon the idea that you know that you have a memory leak and want to find it. Because of these analysis requirements, these technologies have significant performance burdens and are not designed for use as a detection mechanism in production. This limitation means that memory leaks are generally not detected until the problem is critical; the application passes all system tests and is put in production, but it crashes and nobody knows why.

WebSphere Application Server has implemented a lightweight memory leak detection mechanism that runs within the WebSphere Performance and Diagnostic Advisor framework. This mechanism is designed to provide early detection of memory problems in test and production environments. This framework is not designed to provide analysis of the source of the problem, but rather to provide notification and help generating the information that is required to use analysis tools. The mechanism only detects memory leaks in the Java heap and does not detect native leaks.

The lightweight memory leak detection in WebSphere Application Server does not require any additional agents. The detection relies on algorithms that are based on information that is available from the Performance Monitoring Infrastructure service and has minimal performance overhead.

Enabling automated heap dump generation

Use this task to enable automated heap dump generation. This function is not supported when using a Sun Java virtual machine (JVM) which includes WebSphere Application Server running on HP-UX and Solaris operating systems. You need to research taking heap dumps on Sun JVMs or call IBM Support.

Before you begin

Although heap dumps are only generated in response to a detected memory leak, you must understand that generating heap dumps can have a severe performance impact on WebSphere Application Server for several minutes.

About this task

To help you analyze memory leak problems when memory leak detection occurs, use the Heap Analysis Tools for Java™. Use the Heap Analysis Tools component (also known as Heap Analyzer) to perform Java application heap analysis and object create profiling (size and identification) over time. Heap Analyzer includes information about:

- Java virtual machine (JVM) heap growth or size
- The objects being created that include type of object, count and object size, object heap size
- The application "Heap Footprint" for memory sizing and performance considerations
- Includes a call stack for every snapshot when running in profile mode so objects created can be correlated to functions in the application.

The Heap Analyzer tool is a component of the iDoctor for IBM i suite of performance monitoring tools

Use the heap monitor feature to monitor the JVM heap size of a WebSphere Application Server profile in comparison to pool size.

Using the performance advisor in Tivoli Performance Viewer

The performance advisor in Tivoli Performance Viewer provides advice to help tune systems for optimal performance and provides recommendations on inefficient settings by using the collected Performance Monitoring Infrastructure (PMI) data.

About this task

Obtain advice by clicking **Performance Advisor** in Tivoli Performance Viewer. The performance advisor in Tivoli Performance Viewer provides more extensive advice than the Performance and Diagnostic Advisor. For example, Tivoli Performance Viewer provides advice on setting the dynamic cache size, setting the Java virtual machine (JVM) heap size and using the DB2 Performance Configuration wizard.

Procedure

1. Enable data collection and set the PMI monitoring level to Extended.

   The monitoring levels that determine which data counters are enabled can be set dynamically, without restarting the server. These monitoring levels and the data selected determine the type of advice you obtain. The performance advisor in Tivoli Performance Viewer uses the extended monitoring level; however, the performance advisor in Tivoli Performance Viewer can use a few of the more expensive counters (to provide additional advice) and provide advice on which counters can be enabled.

   For example, the advice pertaining to session size needs the PMI statistic set to All. Or, you can use the PMI Custom Monitoring Level to enable the Servlet Session Manager SessionObjectSize counter. The monitoring of the SessionSize PMI counter is expensive, and is not in the Extended PMI statistic set. Complete this action in one of the following ways:

   a. PMI settings.

   b. Enabling Performance Monitoring Infrastructure using the wsadmin tool.

2. In the administrative console, click **Monitoring and Tuning** > **Performance Viewer** > **Current Activity**.
3. Simulate a production level load. Simulate a realistic production load for your application, if you use the performance advisor in a test environment, or do any other performance tuning. The application must run this load without errors. This simulation includes numbers of concurrent users typical of peak periods, and drives system resources, for example, CPU and memory to the levels that are expected in production. The performance advisor only provides advice when CPU utilization exceeds a sufficiently high level. For a list of IBM business partners providing tools to drive this type of load, see the performance: resource for learning information.
4. Log performance data with Tivoli Performance Viewer.
5. Clicking **Refresh** on top of the table of advice causes the advisor to recalculate the advice based on the current data in the buffer.
6. Tuning advice displays when the Advisor icon is chosen in the Tivoli Performance Viewer Performance Advisor. Double-click an individual message for details. Because PMI data is taken over an interval of time and averaged to provide advice, details within the advice message display as averages.

   Note: If the Refresh Rate is adjusted, the Buffer Size must also be adjusted to enable sufficient data to be collected for performing average calculations. Currently 5 minutes of data is required. Hence, the following guidelines intend to help you use the Tivoli Performance Advisor:

   a. You cannot have a Refresh Rate of more than 300 seconds.

   b. RefreshRate * BufferSize > 300 seconds. Buffer Size * Refresh Rate is the amount of PMI data available in memory and it must be greater than 300 seconds.

   c. For the Tivoli Performance Advisor to work properly with Tivoli Performance Viewer logs, the logs must be at least 300 seconds of duration.

   For more information about configuring user and logging settings of Tivoli Performance Viewer, refer to the configuring Tivoli Performance Viewer settings information.

7. Update the product configuration for improved performance, based on advice. Because Tivoli Performance Viewer refreshes advice at a single instant in time, take the advice from the peak load time. Although the performance advisors attempt to distinguish between loaded and idle conditions, misleading advice might be issued if the advisor is enabled while the system is ramping up or down. This result is especially likely when running short tests. Although the advice helps in most configurations, there might be situations where the advice hinders performance. Because of these conditions, advice is not guaranteed. Therefore, test the environment with the updated configuration to ensure it functions and performs well.

   Over a period of time the advisor might issue differing advice. The differing advice is due to load fluctuations and run-time state. When differing advice is received, you need to look at all advice and the time period over which it was issued. You must take advice during the time that most closely represents the peak production load.

   Performance tuning is an iterative process. After applying advice, simulate a production load, update the configuration that is based on the advice, and retest for improved performance. This procedure is continued until optimal performance is achieved.

Performance advisor report in Tivoli Performance Viewer

View recommendations and data from the performance advisor in Tivoli Performance Viewer by clicking the Advisor link in Tivoli Performance Viewer for a server.

For more information on how to use the performance advisor in Tivoli Performance Viewer, see the article, Using the performance advisor in Tivoli Performance Viewer.

Message

Specifies recommendations for performance tuning.

Click the message to obtain more details.

Performance data in the upper panel

Displays a summary of performance data for WebSphere Application Server. Data here corresponds to the same period that recommendations were provided for. However, recommendations might use a different set of data points during analysis than the set that is displayed by the summary page.

The first table represents the number of requests per second and the response time in milliseconds for both the web and Enterprise JavaBeans containers.

The pie graph displays the CPU activity as percentage busy and idle.

The second table displays the average thread activity for the web container and Object Request Broker (ORB) thread pools, and the average database connection activity for connection pools. The activity is expressed as the number of threads or connections busy and idle.

Activating the heap monitor

This task describes the steps used to activate the heap monitor. Heap monitor is used with WebSphere Application Server profiles to monitor heap size of a profile in comparison to pool size.

Before you begin

About this task

You can use the heap monitor feature to monitor the Java Virtual Machine (JVM) heap size of a WebSphere Application Server profile in comparison to pool size. The feature is available for new WebSphere Application Server profiles or profiles that are created.

To check if a WebSphere Application Server profile has the heap monitor enabled and to activate it if necessary, perform the following steps.

Procedure

1. Start the server for the WebSphere Application Server profile.
2. Run the heapMonitor script with the -status flag. For example, for a WebSphere Application Server Version 7 profile named default, enter the following command in the Qshell environment:

   ```
   /QIBM/ProdData/WebSphere/AppServer/V61/ND/bin/heapMonitor -profileName default -status
   ```

   The output should look similar to the following:

   ```
   WASX7209I: Connected to process "server1" on node MYSERVER using SOAP connector;
    The type of process is: UnManagedProcess
    WASX7303I: The following options are passed to the scripting
     environment and are available as argument that is stored in the argv
     variable: "[status, server1]"
    HEAP0002I: The heap monitor is disabled.
    $
   ```

3. To enable the heap monitor for this example, enter the following command in the Qshell environment:

   ```
   /QIBM/ProdData/WebSphere/AppServer/V61/ND/bin/heapMonitor -profileName default -enable
   ```

   The output should look similar to the following:

   ```
   WASX7209I: Connected to process "server1" on node MYSERVER using SOAP connector;
    The type of process is: UnManagedProcess
    WASX7303I: The following options are passed to the scripting environment and are
     available as argument that is stored in the argv
     variable: "[enable, server1]"
    HEAP0005I: Enabling the heap monitor...
    HEAP0003I: The heap monitor has been enabled.
    $
   ```

4. Stop and start the server.

Results

The following message typically appears in the Display Message command (DSPMSG QSYSOPR):

```
HEAP MONITOR STARTED FOR 012500/QEJBSVR/SERVER1 IN SUBSYSTEM qwas8 IN POOL
  *BASE POOL ID=2 POOLSIZE(B)=1687994368 RESERVED(B)=778240 HEAP
  TOTAL(B)=202276864 FREE(B)=67037600 USEDHEAP=135239264
  OS400.GC.HEAP.SIZE.MAX(KB)  =240000000
```

The heap monitor is activated.

Heap monitor default operation

The heap monitor follows default operation behavior as described in this file.

An active heap monitor typically sends a message to the QSYSOPR message queue when the WebSphere Application Server profile starts. For example, the Display Message command (DSPMSG QSYSOPR) displays the following message:

```
HEAP MONITOR STARTED FOR 012500/QEJBSVR/SERVER1 IN SUBSYSTEM qwas8 IN POOL
  *BASE POOL ID=2 POOLSIZE(B)=1687994368 RESERVED(B)=778240 HEAP
  TOTAL(B)=202276864 FREE(B)=67037600 USEDHEAP=135239264
  OS400.GC.HEAP.SIZE.MAX(KB)  =240000000
```

In default operation, a similar message displays ENDED instead of STARTED when the WebSphere Application Server profile is ended. For example:

```
HEAP MONITOR ENDED FOR 012500/QEJBSVR/SERVER1 IN SUBSYSTEM
 qwas8 IN POOL *BASE POOL ID=2 POOLSIZE(B)=6662139904 RESERVED(B)=5165056
 HEAP TOTAL(B)=312999936 FREE(B)=168637264 USEDHEAP=144362672
 OS400.GC.HEAP.SIZE.MAX(KB)  =240000000
```

The Display Log command (DSPLOG LOG(QHST) MSGID(CPI8859)) shows all STARTED and ENDED messages in the history log.

The default operation monitors the size of the Java virtual machine (JVM) Garbage Collection (GC) heap against the following:

- The size of the effective memory pool.
- The size of the memory pool size minus the reserved size.

It also issues a message if the effective memory pool size exceeds 85, 90, 95, or 100 percent. For example:

```
048241/QEJBSVR/SERVER1 GC HEAP USES 95% OF THE NON-RESERVED POOL.  JVM GC
   HEAP SIZE(KB)  EFFECTIVE POOLSIZE(KB):840282 882444.
 048241/QEJBSVR/SERVER1 GC HEAP USES 110% OF THE NON-RESERVED POOL.  JVM GC
   HEAP SIZE(KB)  EFFECTIVE POOLSIZE(KB):974601 882392.
```

The first number is the size of the heap, such as 840282 or 974601. The second number is the effective pool size (or non-reserved pool size), such as 882444 or 882392. The Display Log command (DSPLOG LOG(QHST) MSGID(CPF9898)) shows warning messages in the history log.

For the maximum Garbage Collection heap size, the default operation is to monitor the size of the JVM Garbage Collection heap so that it does not exceed 85, 90, or 95 percent of the maximum. For example:

```
048358/QEJBSVR/USER JAVA USED 88% OF THE GC HEAP.  USED HEAP SIZE(KB)
  AND MAX HEAP(KB):909088 1024001.
```

The maximum heap size is 1024001 Kbytes (-Xmx1000m), and the used heap size is 909088 Kbytes.

Chapter 4. Tuning the application serving environment

Use this topic to understand the benefits of tuning for optimal performance. Learn about about the tunable parameters of the major WebSphere Application Server components and how these parameters affect performance.

Before you begin

WebSphere Application Server provides tunable settings for its major components so that you can adjust the runtime environment to match the characteristics of your application. Applications can run successfully without changing the default values for these tuning parameters. Other applications might need changes, for example, a larger heap size, to achieve optimal performance.

Performance tuning can yield significant gains in performance even if an application is not optimized for performance. However, correcting shortcomings of an application typically results in higher performance gains than are possible with just altering tuning parameters. Many factors contribute to a high performing application.

About this task

Procedure

1. Run the applyPerfTuningTemplate.py script as the starting point for improving the performance of a specific application server. This python-based tuning script, along with one of its template files, applies the recommended performance tuning settings for a typical development, production, or environment that is ready for immediate use. The applyPerfTuningTemplate.py script is located in the *app_server_root*/bin directory. The associated templates and properties files are located in the *app_server_root*/scriptLibraries/perfTuning/V70 directory.
2. Use the performance advisors, the suggested procedures or parameters in the tuning parameter hot list, and the information on troubleshooting performance problems to optimize your WebSphere Application Server instances to their fullest extent.

   Performance advisors
   The performance advisors use the Performance Monitoring Infrastructure (PMI) data to suggest configuration changes to Object Request Broker (ORB) service thread pools, web container thread pools, connection pool size, persisted session size and time, prepared statement cache size, and session cache size. The Runtime Performance Advisor runs in the application server process, while the other advisor runs in the Tivoli Performance Viewer. For more information, see the documentation about using the Performance and Diagnostic Advisor and use the performance advisor in Tivoli Performance Viewer.

   Tuning parameter hot list
   Review the documentation about the tuning parameter hot list. These parameters have an important impact on performance. Because these parameters are application-dependent, the parameter settings for specific applications and environments can vary.

   Troubleshooting performance
   To save you time detecting problems and help you troubleshoot performance problems, see the documentation about troubleshooting performance.

Tuning parameter hot list

The following hot list contains recommendations that have improved performance or scalability, or both, for many applications.

WebSphere Application Server provides several tunable parameters and options to match the application server environment to the requirements of your application.

- **Review the hardware and software requirements**

  It is critical for proper functionality and performance to satisfy the minimum hardware and software requirements. Refer to IBM WebSphere Application Server supported hardware, software, and APIs website which details hardware and software requirements.
- **Check hardware configuration and settings**

  Check network connections to make sure that they are running at their highest speed. For more information, see Chapter 4, “Tuning the application serving environment,” on page 23.
- “Tuning IBM i systems” on page 43

  Operating system configuration plays a key role in performance. For example, adjustments such as TCP/IP parameters might be necessary for your application.

  IBM Power Systems Performance Capabilities Reference IBM i operating system Version 6.1

  Web Performance Advisor
- **Set the minimum and maximum Java virtual machine (JVM) heap sizes**

  Many applications need a larger heap size then the default for best performance. It is also advised to select an appropriate GC policy based on the application's characteristics.
- **Use a type 2 JDBC driver for local data access and a type 4 (or pure Java) JDBC driver for remote data access**

  In general, the type 2 JDBC driver is recommended. Use the link above to view a list of database vendor-specific requirements, which can tell you if a type 4 JDBC driver is supported for your database.

  See the *Administering applications and their environment* PDF for more information.
- **Enable the pass by reference option**

  Use applications that can take advantage of the pass by reference option to avoid the cost of copying parameters to the stack.
- **Ensure that the transaction log is assigned to a fast disk**

  Some applications generate a high rate of writes to the WebSphere Application Server transaction log. Locating the transaction log on a fast disk or disk array can improve response time

  See the *Administering applications and their environment* PDF for more information.
- **Tune related components, for example, database**

  In many cases, some other component, for example, a database, needs adjustments to achieve higher throughput for your entire configuration.

  For more information, see the *Administering applications and their environment* PDF for more information.
- **Disable functions that are not required**

  For example, if your application does not use the web services addressing (WS-Addressing) support, disabling this function can improve performance.

  Attention: Use this property with care because applications might require WS-Addressing MAPs to function correctly. Setting this property also disables WebSphere Application Server support for the following specifications, which depend on the WS-Addressing support: Web Services Atomic Transactions, Web Services Business Agreement and Web Services Notification.

  To disable the support for WS-Addressing, refer to Enabling Web Services Addressing support for JAX-RPC applications
- **Review your application design**

  You can track many performance problems back to the application design. Review the design to determine if it causes performance problems.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server Network Deployment is the `/QIBM/ProdData/WebSphere/AppServer/V8/ND` directory.

java_home

Table 2. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*`x`* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server Network Deployment is the `/QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/`*`profile_name`* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server Network Deployment is the `/QIBM/UserData/WebSphere/AppServer/V8/ND` directory.

The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Tuning TCP/IP buffer sizes

WebSphere Application Server uses the TCP/IP sockets communication mechanism extensively. For a TCP/IP socket connection, the send and receive buffer sizes define the receive window. The receive window specifies the amount of data that can be sent and not received before the send is interrupted. If too much data is sent, it overruns the buffer and interrupts the transfer. The mechanism that controls data transfer interruptions is referred to as flow control. If the receive window size for TCP/IP buffers is too small, the receive window buffer is frequently overrun, and the flow control mechanism stops the data transfer until the receive buffer is empty.

About this task

Flow control can consume a significant amount of CPU time and result in additional network latency as a result of data transfer interruptions. It is recommended that you increase buffer sizes to avoid flow control under normal operating conditions. A larger buffer size reduces the potential for flow control to occur, and results in improved CPU utilization. However, a large buffer size can have a negative effect on performance in some cases. If the TCP/IP buffers are too large and applications are not processing data fast enough, paging can increase. The goal is to specify a value large enough to avoid flow control, but not so large that the buffer accumulates more data than the system can process.

The default buffer size is 8 KB. The maximum size is 8 MB (8096 KB). The optimal buffer size depends on several network environment factors including types of switches and systems, acknowledgment timing, error rates and network topology, memory size, and data transfer size. When data transfer size is extremely large, you might want to set the buffer sizes up to the maximum value to improve throughput, reduce the occurrence of flow control, and reduce CPU cost.

Buffer sizes for the socket connections between the web server and WebSphere Application Server are set at 64KB. In most cases this value is adequate.

Flow control can be an issue when an application uses either the IBM Developer Kit for Java(TM) JDBC driver or the IBM Toolbox for Java JDBC driver to access a remote database. If the data transfers are large, flow control can consume a large amount of CPU time. If you use the IBM Toolbox for Java JDBC driver, you can use custom properties to configure the buffer sizes for each data source. It is recommended that you specify large buffer sizes, for example, 1 MB.

Some system-wide settings can override the default 8 KB buffer size for sockets. With some applications, for example, WebSphere Commerce Suite, a buffer size of 180 KB reduces flow control and typically does not adversely affect paging. The optimal value is dependent on specific system characteristics. You might need to try several values before you determine the ideal buffer size for your system.

To change the system wide value, perform the following steps:

Procedure

Tune the TCP/IP buffer sizes.

1. Change the TCP/IP configuration.

a. Run the Change TCP/IP Attribute, **CHGTCPA** command.
b. View and change the buffer sizes by pressing **F4** on the Change TCP/IP Attributes window. The buffer sizes are displayed as the TCP receive and send buffer sizes. Type new values and save your changes.

2. Recycle TCP/IP, then monitor CPU and paging rates to determine if they are within recommended system guidelines.

Results

Repeat this process until you determine the ideal buffer size.

What to do next

The TCP/IP buffer sizes are changed. Repeat this process until you determine the ideal buffer size.

For more information about TCP/IP performance, see Chapter 5 of the Performance Capabilities Reference. Links to several editions of the Performance Capabilities Reference are in the Performance Management Resource Library.

Tuning the JVM

Tuning the IBM virtual machine for Java

An application server is a Java based server and requires a Java virtual machine (JVM) environment to run and support the enterprise applications that run on it. As part of configuring your application server, you can configure the Java SE Runtime Environment to tune performance and system resource usage. This topic applies to IBM virtual machines for Java.

Before you begin

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

- Determine the type of JVM on which your application server is running.

  Issue the dspwasinst command from the profile_root/bin directory. The output from this command contains the JAVA_HOME setting and other information about the JVM enabled for your application server profile.

  If your application server is running on the IBM i Java Developer Kit 6.0 JVM, which is also known as the Classic JVM, see the topic *Tuning the Classic JVM (IBM i)*.

  Use the managesdk command, if you want to enable your application server profile to use a different JVM.

  Use the enablejvm command, if you want to enable your application server profile to use a different JVM.

- Verify that:
  1. The most recent supported version of the JVM is installed on your system.
  2. The most recent service update is installed on your system. Almost every new service level includes JVM performance improvements.

About this task

Each JVM vendor provides detailed information on performance and tuning for their JVM. Use the information provided in this topic in conjunction with the information that is provided with the JVM that is running on your system.

A Java SE Runtime Environment provides the environment for running enterprise applications and application servers. Therefore the Java configuration plays a significant role in determining performance and system resource consumption for an application server and the applications that run on it.

The IBM virtual machine for Java Version 6.0 includes the latest in Java Platform, Enterprise Edition (Java EE) specifications, and provides performance and stability improvements over previous versions of Java.

Even though JVM tuning is dependent on the JVM provider you use, there are some general tuning concepts that apply to all JVMs. These general concepts include:

- Compiler tuning. All JVMs use Just-In-Time (JIT) compilers to compile Java byte codes into native instructions during server runtime.
- Java memory or heap tuning. Tuning the JVM memory management function, or garbage collection, is a good starting point for improving JVM performance.
- Class loading tuning.
- Start up versus runtime performance optimization

The following steps provide specific instructions on how to perform the following types of tuning for each JVM. The steps do not have to be performed in any specific order.

Procedure

1. Optimize the startup and runtime performance.

   In some environments, such as a development environment, it is more important to optimize the startup performance of your application server rather than the runtime performance. In other environments, it is more important to optimize the runtime performance. By default, IBM virtual machines for Java are optimized for runtime performance, while HotSpot-based JVMs are optimized for startup performance.

   The Java Just-in-Time (JIT) compiler impacts whether startup or runtime performance is optimized. The initial optimization level that the compiler uses influences the length of time that is required to compile a class method, and the length of time that is required to start the server. For faster startups, reduce the initial optimization level that the compiler uses. However if you reduce the initial optimization level, the runtime performance of your applications might decrease because the class methods are now compiled at a lower optimization level.

   - **-Xquickstart**

     This setting influences how the IBM virtual machine for Java uses a lower optimization level for class method compiles. A lower optimization level provides for faster server startups, but lowers runtime performance. If this parameter is not specified, the IBM virtual machine for Java defaults to starting with a high initial optimization level for compiles, which results in faster runtime performance, but slower server starts.

     You can set this property on the Java virtual machine panel using the administrative console. For details, read the information about Java virtual machine settings.

Default	High initial compiler optimization level
Recommended	High initial compiler optimization level
Usage	Specifying `-Xquickstart` improves server startup time.

2. Verify the classpath as part of the Java configuration.

In the General Properties section of the Configuration Tab, enter the classpath in the text box of the Classpath option. Make sure that the classpath points to only the classes you need. If possible, the classes that are referenced most frequently should be located near the front of the path.

Sometimes poor performance is caused by a missing class. The class loader looks in its tables of already loaded classes and if the class is not found to be already loaded it will search for it. This search process can cause a high amount of I/O activity to the HFS volumes. To determine if this is the problem you can collect CTRACE records from the file system. Once you determine which class is not being found you can repair the problem by providing the class or by removing the need for it.

3. Configure the heap size.

   The Java heap parameters influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes garbage collection to take longer.

   The JVM uses defined thresholds to manage the storage that it is allocated. When the thresholds are reached, the garbage collector is invoked to free up unused storage. Therefore, garbage collection can cause significant degradation of Java performance. Before changing the initial and maximum heap sizes, you should consider the following information:

   - In the majority of cases you should set the maximum JVM heap size to a value that is higher than the initial JVM heap size. This setting allows for the JVM to operate efficiently during normal, steady state periods within the confines of the initial heap. This setting also allows the JVM to operate effectively during periods of high transaction volume because the JVM can expand the heap up to the value specified for the maximum JVM heap size. In some rare cases, where absolute optimal performance is required, you might want to specify the same value for both the initial and maximum heap size. This setting eliminates some overhead that occurs when the JVM expands or contracts the size of the JVM heap. Before changing any of the JVM heap sizes, verify that the JVM storage allocation is large enough to accommodate the new heap size.
   - Do not make the size of the initial heap so large that while it initially improves performance by delaying garbage collection, when garbage collection does occur, the collection process affects response time because the process has to run longer.

   To use the administrative console to configure the heap size:

   a. In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*.
   b. In the Server Infrastructure section, click **Java and process management > Process definition > Java virtual machine**.
   c. Specify a new value in either the **Initial heap size** or the **Maximum heap size** field.

      You can also specify values for both fields if you need to adjust both settings.

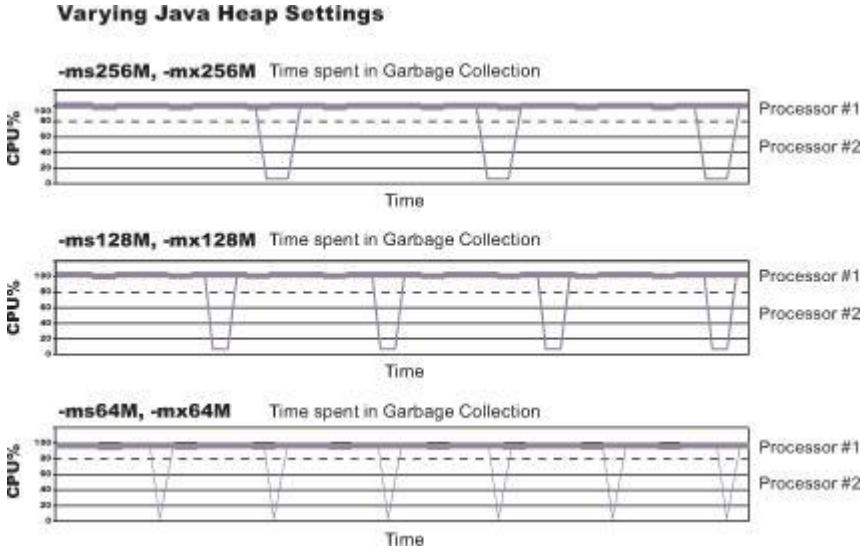
      For performance analysis, the initial and maximum heap sizes should be equal.

      The Initial heap size setting specifies, in megabytes, the amount of storage that is allocated for the JVM heap when the JVM starts. The Maximum heap size setting specifies, in megabytes, the maximum amount of storage that can be allocated to the JVM heap. Both of these settings have a significant effect on performance.

      If you are tuning a production system where you do not know the working set size of the enterprise applications that are running on that system, an appropriate starting value for the initial heap size is 25 percent of the maximum heap size. The JVM then tries to adapt the size of the heap to the working set size of the application.

      The following illustration represents three CPU profiles, each running a fixed workload with varying Java heap settings. In the middle profile, the initial and maximum heap sizes are set to 128 MB. Four garbage collections occur. The total time in garbage collection is about 15 percent of the total run. When the heap parameters are doubled to 256 MB, as in the top profile, the length of the work time increases between garbage collections. Only three garbage collections occur, but the length of each garbage collection is also increased. In the third profile, the heap size is reduced to 64 MB and exhibits the opposite effect. With a smaller heap size, both the time between garbage

collections and the time for each garbage collection are shorter. For all three configurations, the total time in garbage collection is approximately 15 percent. This example illustrates an important concept about the Java heap and its relationship to object utilization. A cost for garbage collection always exists when running enterprise applications.



Run a series of tests that vary the Java heap settings. For example, run experiments with 128 MB, 192 MB, 256 MB, and 320 MB. During each experiment, monitor the total memory usage. If you expand the heap too aggressively, paging can occur.

Use the IBM i WRKSYSSTS command to check for paging. If paging occurs, reduce the size of the heap or add more memory to the system.

When all the runs are finished, compare the following statistics:
- Number of garbage collection calls
- Average duration of a single garbage collection call
- Ratio between the length of a single garbage collection call and the average time between calls

If the application is not over utilizing objects and has no memory leaks, the state of steady memory utilization is reached. Garbage collection also occurs less frequently and for short duration.

If the heap free space settles at 85 percent or more, consider decreasing the maximum heap size values because the application server and the application are under-utilizing the memory allocated for heap.

d. Click **Apply**.

e. Click **Save** to save your changes to the master configuration.

f. Stop and restart the application server.

You can also use the following command-line parameters to adjust these settings. These parameters apply to all supported JVMs and are used to adjust the minimum and maximum heap size for each application server or application server instance.

- **-Xms**

  This parameter controls the initial size of the Java heap. Tuning this parameter reduces the overhead of garbage collection, which improves server response time and throughput. For some applications, the default setting for this option might be too low, which causes a high number of minor garbage collections.

Default	50 MB
Recommended	Workload specific, but higher than the default.
Usage	Specifying `-Xms256m` sets the initial heap size to 256 MB.

- **-Xmx**

  This parameter controls the maximum size of the Java heap. Increasing this parameter increases the memory available to the application server, and reduces the frequency of garbage collection. Increasing this setting can improve server response time and throughput. However, increasing this setting also increases the duration of a garbage collection when it does occur. This setting should never be increased above the system memory available for the application server instance. Increasing the setting above the available system memory can cause system paging and a significant decrease in performance.

Default	By default, the JVM dynamically calculates the Java heap size based on the available memory in the system.
Recommended	Workload specific, but higher than the default value, depending on the amount of available physical memory.
Usage	Specifying `-Xmx512m` sets the maximum heap size to 512 MB.

  Note: Specify a value for the **-Xmx** parameter to reduce possible out-of-memory issues.

- **-Xlp**

  Use this parameter with the IBM virtual machine for Java to allocate the heap when using large pages, such as 16 MB pages. Before specifying this parameter, verify that your operating system is configured to support large pages. Using large pages can reduce the CPU overhead needed to keep track of heap memory, and might also allow the creation of a larger heap.

- **–Xlp64k**

  This parameter can be used to allocate the heap using medium size pages, such as 64 KB. Using this virtual memory page size for the memory that an application requires can improve the performance and throughput of the application because of hardware efficiencies that are associated with a larger page size.

  IBM i and AIX® provide rich support around 64 KB pages because 64 KB pages are intended to be general purpose pages. 64 KB pages are easy to enable, and applications might receive performance benefits when 64 KB pages are used instead of 4 KB pages, which is the default setting. This setting can be changed without changing the operating system configuration. However, it is recommended that you run your application servers in a separate storage pool if you enable the use of 64KB pages.

Default	4 KB
Recommended	-Xlp64k enables the 64 KB page size support. IBM i POWER5+ systems, and IBM i Version 6, Release 1, support a 64 KB page size.

4. Tune Java memory.

   Enterprise applications written in the Java language involve complex object relationships and use large numbers of objects. Although, the Java language automatically manages memory associated with object life cycles, understanding the application usage patterns for objects is important. In particular, verify that the following conditions exist:
   - The application is not over utilizing objects
   - The application is not leaking objects
   - The Java heap parameters are set properly to handle a given object usage pattern

   a. Check for over-utilization of objects.

      You can review the counters for the JVM run time, that are included in Tivoli Performance Viewer reports, to determine if an application is overusing objects. You have to specify the -XrunpmiJvmtiProfiler command-line option, as well as the JVM module maximum level, to enable the Java virtual machine profiler interface, JVMTI, counters.

The optimal result for the average time between garbage collections is at least five to six times the average duration of a single garbage collection. If you do not achieve this number, the application is spending more than 15 percent of its time in garbage collection.

If the information indicates a garbage collection bottleneck, there are two ways to clear the bottleneck. The most cost-effective way to optimize the application is to implement object caches and pools. Use a Java profiler to determine which objects to target. If you can not optimize the application, try adding memory, processors and clones. Additional memory allows each clone to maintain a reasonable heap size. Additional processors allow the clones to run in parallel.

b. Test for memory leaks.

Memory leaks in the Java language are a dangerous contributor to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently until the heap is exhausted and the Java code fails with a fatal out-of-memory exception. Memory leaks occur when an unused object has references that are never freed. Memory leaks most commonly occur in collection classes, such as Hashtable because the table always has a reference to the object, even after real references are deleted.

High workload often causes applications to crash immediately after deployment in the production environment. If an application has memory leaks, a high workload can accelerate the magnification of the leakage and cause memory allocation failures to occur.

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list provides insight on how to interpret the results of your memory leak testing:

- **Long-running test**

  Memory leak problems can manifest only after a period of time, therefore, memory leaks are found easily during long-running tests. Short running tests might provide invalid indications of where the memory leaks are occurring. It is sometimes difficult to know when a memory leak is occurring in the Java language, especially when memory usage has seemingly increased either abruptly or monotonically in a given period of time. The reason it is hard to detect a memory leak is that these kinds of increases can be valid or might be the intention of the developer. You can learn how to differentiate the delayed use of objects from completely unused objects by running applications for a longer period of time. Long-running application testing gives you higher confidence for whether the delayed use of objects is actually occurring.
- **Repetitive test**

  In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the number of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

  You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is easier. First, the memory usage before running the module is recorded. Then, a fixed set of test cases are run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection must be suggested when recording the actual memory usage by inserting System.gc() in the module where you want garbage collection to occur, or using a profiling tool, to force the event to occur.
- **Concurrency test**

  Some memory leak problems can occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to memory leaks because of the added complication in the program logic. Careless programming can lead to kept or

not-released references. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

Consider the following points when choosing which test cases to use for memory leak testing:

- A good test case exercises areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario can suggest creation of data spaces, such as adding a new record, creating an HTTP session, performing a transaction and searching a record.
- Look at areas where collections of objects are used. Typically, memory leaks are composed of objects within the same class. Also, collection classes such as Vector and Hashtable are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the get method of a Hashtable object does not remove its reference to the retrieved object.

You can use the Tivoli Performance Viewer to help find memory leaks.

For optimal results, repeat experiments with increasing duration, such as 1,000, 2,000, and 4,000 page requests. The Tivoli Performance Viewer graph of used memory should have a jagged shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following conditions is appears in the graph:

- The amount of memory used immediately after each garbage collection increases significantly. When this condition occurs, the jagged pattern looks more like a staircase.
- The jagged pattern has an irregular shape.
- The gap between the number of objects allocated and the number of objects freed increases over time.

Heap consumption that indicates a possible leak during periods when the application server is consistently near 100 percent CPU utilization, but disappears when the workload becomes lighter or near-idle, is an indication of heap fragmentation. Heap fragmentation can occur when the JVM can free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small free memory areas in the heap to larger contiguous spaces.

Another form of heap fragmentation occurs when objects that are less than 512 bytes are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation until a heap compaction occurs.

Heap fragmentation can be reduced by forcing compactions to occur. However, there is a performance penalty for forcing compactions. Use the Java -X command to see the list of memory options.

5. Tune garbage collection

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection typically consumes from 5 to 20 percent of total run time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application.

Monitoring garbage collection while a fixed workload is running, provides you with insight as to whether the application is over using objects. Garbage collection can even detect the presence of memory leaks.

You can use JVM settings to configure the type and behavior of garbage collection. When the JVM cannot allocate an object from the current heap because of lack of contiguous space, the garbage collector is invoked to reclaim memory from Java objects that are no longer being used. Each JVM vendor provides unique garbage collector policies and tuning parameters.

You can use the **Verbose garbage collection** setting in the administrative console to enable garbage collection monitoring. The output from this setting includes class garbage collection statistics. The format of the generated report is not standardized between different JVMs or release levels.

To adjust your JVM garbage collection settings:

a. In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*.
b. In the Server Infrastructure section, click **Java and process management > Process definition > Java virtual machine**
c. Enter the –X option you want to change in the **Generic JVM arguments** field.
d. Click **Apply**.
e. Click **Save** to save your changes to the master configuration.
f. Stop and restart the application server.

The following list describes the –X options for the different JVM garbage collectors.

The IBM virtual machine for Java garbage collector.

A complete guide to the IBM implementation of the Java garbage collector is provided in the *IBM Developer Kit and Runtime Environment, Java2 Technology Edition, Version 5.0 Diagnostics Guide*. This document is available on the developerWorks® website.

Use the Java -X option to view a list of memory options.

- **-Xgcpolicy**

  The IBM virtual machine for Java provides four policies for garbage collection. Each policy provides unique benefits.

  Note: While each policy provides unique benefits, for WebSphere Application Server V8, gencon is the default garbage collection policy. Previous versions of the application server specify that optthruput is the default garbage collection policy.

  – gencon is the default policy. This policy works with the generational garbage collector. The generational scheme attempts to achieve high throughput along with reduced garbage collection pause times. To accomplish this goal, the heap is split into new and old segments. Long lived objects are promoted to the old space while short-lived objects are garbage collected quickly in the new space. The gencon policy provides significant benefits for many applications. However, it is not suited for all applications, and is typically more difficult to tune.

  – optthruput provides high throughput but with longer garbage collection pause times. During a garbage collection, all application threads are stopped for mark, sweep and compaction, when compaction is needed. The gencon policy is sufficient for most applications.

  – optavgpause is the policy that reduces garbage collection pause time by performing the mark and sweep phases of garbage collection while an application is running. This policy causes a small performance impact to overall throughput.

  – subpool is a policy that increases performance on multiprocessor systems, that commonly use more then 8 processors. This policy is only available on IBM System i® System p® and System z® processors. The subpool policy is similar to the gencon policy except that the heap is divided into subpools that provide improved scalability for object allocation.

Default	gencon
Recommended	gencon
Usage	Specifying `Xgcpolicy:gencon` sets the garbage collection policy to gencon.

Setting **gcpolicy** to `gencon` disables concurrent mark. You should get optimal throughput results when you use the gencon policy unless you are experiencing erratic application response times, which is an indication that you might have pause time problems

Setting **gcpolicy** to `optavgpause` enables concurrent mark with its default values. This setting alleviates erratic application response times that normal garbage collection causes. However, this option might decrease overall throughput.

- **-Xnoclassgc**

  By default, the JVM unloads a class from memory whenever there are no live instances of that class left. The overhead of loading and unloading the same class multiple times, can decrease performance.

  gotcha: You can use the `-Xnoclassgc` argument to disable class garbage collection. However, the performance impact of class garbage collection is typically minimal, and turning off class garbage collection in a Java Platform, Enterprise Edition (Java EE) based system, with its heavy use of application class loaders, might effectively create a memory leak of class data, and cause the JVM to throw an Out-of-Memory Exception.

  If you use this option, whenever you redeploy an application, you should always restart the application server to clear the classes and static data from the pervious version of the application.

Default	Class garbage collection is enabled.
Recommended	Do not disable class garbage collection.
Usage	Specify `Xnoclassgc` to disable class garbage collection.

6. **Enable class sharing in a cache.**

   The share classes option of the IBM implementation of the Java 2 Runtime Environment (J2RE) Version 1.5.0 lets you share classes in a cache. Sharing classes in a cache can improve startup time and reduce memory footprint. Processes, such as application servers, node agents, and deployment managers, can use the share classes option.

   This option is enabled by default in the application server. To clear the cache, either call the app_server_root/bin/clearClassCache utility or stop the application server and then restart the application server.

   If you need to disable the share classes option for a process, specify the generic JVM argument -Xshareclasses:none for that process:

   a. In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*.
   b. In the Server Infrastructure section, click **Java and process management > Process definition > Java virtual machine**
   c. Enter `-Xshareclasses:none` in the **Generic JVM arguments** field.
   d. Click **OK**.
   e. Click **Save** to save your changes to the master configuration.
   f. Stop and restart the application server.

Default	The Share classes in a cache option are enabled.
Recommended	Leave the share classes in a cache option enabled.
Usage	Specifying `-Xshareclasses:none` disables the share classes in a cache option.

7. Tune the configuration update process for a large cell configuration.

   In a large cell configuration, you might need to determine whether configuration update performance or consistency checking is more important. The deployment manager maintains a master configuration repository for the entire cell. By default, when the configuration changes, the product compares the configuration in the workspace with the master repository to maintain workspace consistency. However,

the consistency verification process can cause an increase in the amount of time to save a configuration change or to deploy a large number of applications. The following factors influence how much time is required:

- The more application servers or clusters that are defined in a cell, the longer it takes to save a configuration change.
- The more applications that are deployed in a cell, the longer it takes to save a configuration change.

If the amount of time required to change a configuration change is unsatisfactory, you can add the `config_consistency_check` custom property to your JVM settings and set the value of this property to false.

a. In the administrative console, click **System administration > Deployment manager**.
b. Under Server Infrastructure, select Java and Process Management, and then click **Process Definition**.
c. Under Additional Properties, click **Java Virtual Machine > Custom Properties > New**.
d. Enter `config_consistency_check` in the Name field and `false` in the Value field.
e. Click **OK** and then save these changes to the master configuration.
f. Restart the server.

Note: The `config_consistency_check` custom property affects the deployment manager process only. It does not affect other processes including the node agent and application server processes. The consistency check is not performed on these processes. However, within the `SystemOut.log` files for these processes, you might see a note that the consistency check is disabled. For these non-deployment manager processes, you can ignore this message.

If you are using the wsadmin command wsadmin -conntype none in local mode, you must set the **config_consistency_check** property to `false` before issuing this command.

What to do next

Continue to gather and analyze data as you make tuning changes until you are satisfied with how the JVM is performing.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server Network Deployment is the /QIBM/ProdData/WebSphere/AppServer/V8/ND directory.

java_home

Table 3. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root
The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server Network Deployment is the /QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/*profile_name* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server Network Deployment is the /QIBM/UserData/WebSphere/AppServer/V8/ND directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Tuning transport channel services

The transport channel services manage client connections and I/O processing for HTTP and JMS requests. These I/O services are based on the non-blocking I/O (NIO) features that are available in Java. These services provide a highly scalable foundation to WebSphere Application Server request processing. Java NIO-based architecture has limitations in terms of performance, scalability, and user usability.

Therefore, integration of true asynchronous I/O is implemented. This implementation provides significant benefits in usability, reduces the complexity of I/O processing, and reduces that amount of performance tuning you must perform.

About this task

Key features of the new transport channel services include:

- Scalability, which enables the product to handle many concurrent requests
- Asynchronous request processing, which provides a many-to-one mapping of client requests to web container threads
- Resource sharing and segregation, which enables thread pools to be shared between the web container and a messaging service
- Improved usability
- Incorporation of autonomic tuning and configuration functions

Changing the default values for settings on one or more of the transport channels associated with a transport chain can improve the performance of that chain.

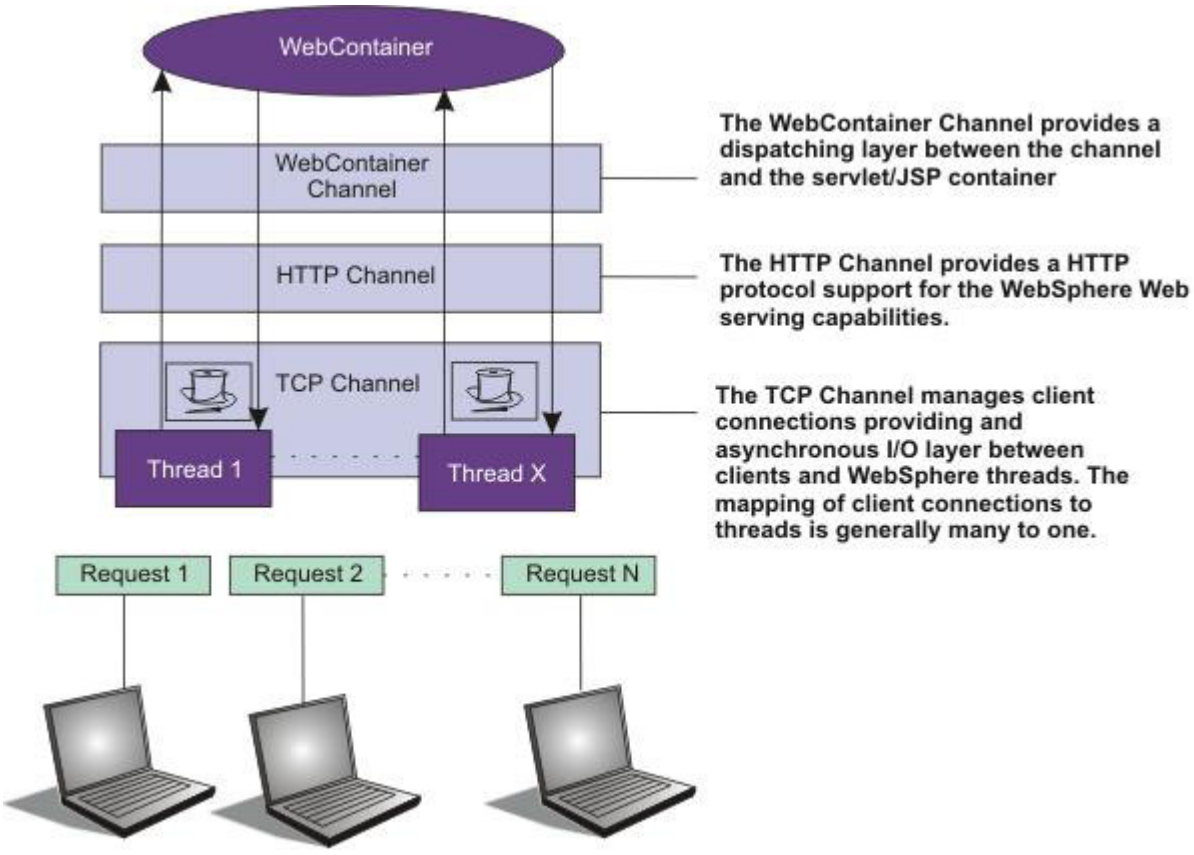


Figure 1. Transport Channel Service

Procedure

- Adjust TCP transport channel settings. In the administration console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Ports**. Then click **View associated transports** for the appropriate port.
  1. Select the transport chain whose properties you are changing.

2. Click the TCP transport channel defined for that chain.
3. Lower the value specified for the Maximum open connections property. This parameter controls the maximum number of connections that are available for a server to use. Leaving this parameter at the default value of `20000`, which is the maximum number of connections, might lead to stalled websites under failure conditions, because the product continues to accept connections, thereby increasing the connection, and associated work, backlog. The default should be changed to a significantly lower number, such as `500`, and then additional tuning and testing should be performed to determine the optimal value that you should specify for a specific website or application deployment.
4. If client connections are being closed without data being written back to the client, change the value specified for the Inactivity timeout parameter. This parameter controls the maximum number of connections available for a server use. After receiving a new connection, the TCP transport channel waits for enough data to arrive to dispatch the connection to the protocol-specific channels above the TCP transport channel. If not enough data is received during the time period specified for the Inactivity timeout parameter, the TCP transport channel closes the connection.

   The default value for this parameter is `60` seconds. This value is adequate for most applications. Increase the value specified for this parameter if your workload involves many connections and all of these connections cannot be serviced in 60 seconds.
5. Assign a thread pool to a specific HTTP port. Each TCP transport channel is assigned to a particular thread pool. Thread pools can be shared between one or more TCP transport channels as well as with other components. The default setting for a TCP transport channel is to have all HTTP-based traffic assigned to the `WebContainer` thread pool and all other traffic assigned to the `Default` thread pool. Use the **Thread pool** menu list to assign a particular thread pool to each TCP transport channel. The default setting for this parameter has all HTTP-based traffic assigned to the `WebContainer` thread pool and all other traffic is assigned to the `Default` thread pool. The information about thread pool collection describes how to create additional thread pools.
6. Tune the size of your thread pools. By default, a thread pool can have a minimum of `10` threads and a maximum of `50` maximum threads. To adjust these values, click **Thread pools** >*threadpool_name* and adjust the values specified for the Minimum Size and Maximum Size parameters for that thread pool.

   Typical applications usually do not need more than 10 threads per processor. One exception is if there is some off server condition, such as a very slow backend request, that causes a server thread to wait for the backend request to complete. In such a case, processor usage is low and increasing the workload does not increase processor throughput. Thread memory dumps show nearly all threads in a callout to the backend resource. If this condition exists, and the backend is tuned correctly, try increasing the minimum number of threads in the pool until you see improvements in throughput and thread memory dumps show threads in other areas of the run time besides the backend call.

   The setting for the Grow as needed parameter is changed unless your backend is prone to hanging for long periods of time. This condition might indicate that all of your runtime threads are blocked waiting for the backend instead of processing other work that does not involve the hung backend.

- Adjust HTTP transport channel settings. In the administration console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Ports**. Then click **View associated transports** for the appropriate port.
  1. Select the transport chain whose properties you are changing.
  2. Click the HTTP transport channel defined for that chain.
  3. Tune HTTP keep-alive.

     The Use persistent (keep-alive) connections setting controls whether connections are left open between requests. Leaving the connections open can save setup and teardown costs of sockets if your workload has clients that send multiple requests. The default value is true, which is typically the optimal setting.

If your clients only send single requests over substantially long periods of time, it is probably better to disable this option and close the connections right away rather than to have the HTTP transport channel setup the timeouts to close the connection at some later time.

4. Change the value specified for the Maximum persistent requests parameter to increase the number of requests that can flow over a connection before it is closed.

   When the **Use persistent connections** option is enabled, the Maximum persistent requests parameter controls the number of requests that can flow over a connection before it is closed. The default value is `100`. This value should be set to a value such that most, if not all, clients always have an open connection when they make multiple requests during the same session. A proper setting for this parameter helps to eliminate unnecessary setting up and tearing down of sockets.

   For test scenarios in which the client is never closed, a value of `-1` disables the processing which limits the number of requests over a single connection. The persistent timeout shuts down some idle sockets and protects your server from running out of open sockets.

5. Change the value specified for the Persistent timeout parameter to increase the length of time that a connection is held open before being closed due to inactivity. The Persistent timeout parameter controls the length of time that a connection is held open before being closed because there is no activity on that connection. The default value is `30` seconds This parameter is set to a value that keeps enough connections open so that most clients can obtain a connection available when they must make a request.
6. If clients are having trouble completing a request because it takes them more than `60` seconds to send their data, change the value specified for the Read timeout parameter. Some clients pause more than `60` seconds while sending data as part of a request. To ensure that they are able to complete their requests, change the value specified for this parameter to a length of time in seconds that is sufficient for the clients to complete the transfer of data. Be careful when changing this value that you still protect the server from clients who send incomplete data and thereby use resources (sockets) for an excessive amount of time.
7. If some of your clients require more than `60` seconds to receive data being written to them, change the value specified for the Write timeout parameter. Some clients are slow and require more than `60` seconds to receive data that is sent to them. To ensure that they are able to obtain all of their data, change the value specified for this parameter to a length of time in seconds that is sufficient for all of the data to be received. Be careful when changing this value that you still protect the server from malicious clients.

- Adjust the web container transport channel settings. In the administration console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Ports**. Then click **View associated transports** for the appropriate port.
  1. Select the transport chain whose properties must be changed.
  2. Click the web container transport channel defined for that chain.
  3. If multiple writes are required to handle responses to the client, change the value specified for the Write buffer size parameter to a value that is more appropriate for your clients. The Write buffer size parameter controls the maximum amount of data per thread that the web container buffers before sending the request on for processing. The default value is `32768` bytes, which are sufficient for most applications. If the size of a response is greater than the size of the write buffer, the response is chunked and written back in multiple TCP writes.

     If you must change the value specified for this parameter, make sure that the new value enables most requests to be written out in a single write. To determine an appropriate value for this parameter, look at the size of the pages that are returned and add some additional bytes to account for the HTTP headers.
- **Adjust the settings for the bounded buffer.**

  Even though the default bounded buffer parameters are optimal for most of the environments, you might want to change the default values in certain situations and for some operating systems to enhance performance. Changing the bounded buffer parameters can degrade performance. Therefore, make sure that you tune the other related areas, such as the web container and ORB thread pools, before deciding to change the bounded buffer parameters.

To change the bounded buffer parameters:

1. In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*.
2. In the Server Infrastructure section, click **Java and process management > Process definition > Java virtual machine**.
3. Click **Custom properties**.
4. Enter one of the following custom properties in the Name field and an appropriate value in the **Value** field, and then click **Apply** to save the custom property and its setting.
   - com.ibm.ws.util.BoundedBuffer.spins_take=*value*

     Specifies the number of times a web container thread can attempt to retrieve a request from the buffer before the thread is suspended and enqueued. This parameter enables you to trade off the cost of performing possibly unsuccessful retrieval attempts, with the cost to suspending a thread and activating it again in response to a put operation.

Default:	The number of processors available to the operating system minus 1.
Recommended:	Use any non-negative integer value. In practice, using an integer from 2 to 8 yields the best performance results.
Usage:	com.ibm.ws.util.BoundedBuffer.spins_take=6. Six attempts are made before the thread is suspended.

   - com.ibm.ws.util.BoundedBuffer.yield_take=true or false

     Specifies that a thread yields the processor to other threads after a set number of attempts to take a request from the buffer. Typically a lower number of attempts is preferable.

Default:	false
Recommended:	The effect of yield is implementation-specific for individual platforms.
Usage:	com.ibm.ws.util.BoundedBuffer.spins_take=*boolean value*

   - com.ibm.ws.util.BoundedBuffer.spins_put=*value*

     Specifies the number of attempts an InboundReader thread makes to put a request into the buffer before the thread is suspended and enqueued. Use this value to trade off between the cost of repeated, possibly unsuccessful, attempts to put a request into the buffer with the cost to suspend a thread and reactivate it in response to a take operation.

Default:	The value of com.ibm.ws.util.BoundedBuffer.spins_take divided by 4.
Recommended:	Use any non-negative integer value. In practice an integer 2 - 8 have shown the best performance results.
Usage:	com.ibm.ws.util.BoundedBuffer.spins_put=6. Six attempts are made before the thread is suspended.

   - com.ibm.ws.util.BoundedBuffer.yield_put=true or false

     Specifies that a thread yields the processor to other threads after a set number of attempts to put a request into the buffer. Typically a lower number of attempts is preferable.

Default:	false
Recommended:	The effect of yield is implementation-specific for individual platforms.
Usage:	com.ibm.ws.util.BoundedBuffer.yield_put=*boolean value*

– com.ibm.ws.util.BoundedBuffer.wait=*number of milliseconds*

Specifies the maximum length of time, in milliseconds, that a request might unnecessarily be delayed if the buffer is completely full or if the buffer is empty.

Default:	10000 milliseconds
Recommended:	A value of 10000 milliseconds usually works well. In rare instances when the buffer becomes either full or empty, a smaller value guarantee a more timely handling of requests, but there is usually a performance impact to using a smaller value.
Usage:	com.ibm.ws.util.BoundedBuffer.wait=8000. A request might unnecessarily be delayed up to 8000 milliseconds.

- Click **Apply** and then click **Save** to save these changes.

Checking hardware configuration and settings

An optimal hardware configuration enables applications to get the greatest benefit from performance tuning. The hardware speed impacts all types of applications and is critical to overall performance.

About this task

For proper system sizing for WebSphere Application Server workloads, use the IBM Systems Workload Estimator.

You can check hardware configuration and settings such as disk speed, system memory and processor speed to gain performance benefits.

Procedure

Use the following considerations for selecting and configuring the hardware on which the application servers run:

1. **Optimize disk speed**
   - **Description:** Disk speed and the number of disk arms have a significant effect on application server performance in the following cases:
     – Your application is heavily dependent on database support .
     – Your application uses messaging extensively.
   - **Recommendation:** Use disk I/O subsystems that are optimized for performance, for example, Redundant Array of Independent Disks (RAID). Distribute the disk processing across as many disks as possible to avoid contention issues that occur with 1 or 2 disk systems. For more information about disk arms and how they can affect performance, see the iSeries® Disk Arm Requirements documentation.
2. **Increase processor speed and processor cache**
   - **Description:** In the absence of other bottlenecks, increasing the processing power can improve throughput, response times, or both. On WebSphere Application Server for IBM i, processing power can be related to the Commercial Processing Workload (CPW) value of the system. For more information about CPW values, see the Performance Management website.
3. **Increase system memory**
   - **Description:** If a large number of page faults occur, performing the following tasks to improve performance:
     – Increase the memory available to WebSphere Application Server.
     – Move WebSphere Application Server to another memory pool.
     – Remove jobs from the WebSphere Application Server memory pool

- **Recommendation:** To determine the current page fault level, run the Work with System Status (WRKSYSSTS) command from an IBM i command line. For information about the minimum memory requirements, see the IBM Support website.

4. **Increase system memory**
   - **Description:** If a large number of page faults occur, performing the following tasks to improve performance:
     – Increase the memory available to WebSphere Application Server.
     – Move WebSphere Application Server to another memory pool.
     – Remove jobs from the WebSphere Application Server memory pool
   - **Recommendation:** To determine the current page fault level, run the Work with System Status (WRKSYSSTS) command from an IBM i command line. For information about the minimum memory requirements, see the IBM Support website.
5. **Run network cards and network switches at full duplex**
   - **Description:** Run network cards and network switches at full duplex and use the highest supported speed. Full duplex is much faster than half duplex. Verify that the network speed of adapters, cables, switches, and other devices can accommodate the required throughput. Some websites might require multiple gigabit links.
   - **Recommendation** Make sure that the highest speed is in use on 10/100/1000 Ethernet networks.
6. **Verify that the activity levels for storage pools are sufficient**
   - **Description:** Verify that the activity levels for storage pools are sufficient. Increasing these values can prevent threads from transitioning into the ineligible condition.
   - **Recommendation**
     – To modify the activity level for the storage pool in which you are running WebSphere Application Server, run the following **WRKSYSSTS** command from the command line:

       ```
       WRKSYSSTS ASTLVL(*INTERMED)
       ```

     – Perform the following steps to set the QMAXACTLVL system value to a value equal to or greater than the total activity level for all pools, or *NOMAX:
       - Run the following **WRKSYSSTS** command from the command line:

         ```
         WRKSYSSTS ASTLVL(*INTERMED)
         ```

       - Adjust the value in the **Max Active** column.

Tuning operating systems

You can tune your operating system to optimize the performance of WebSphere Application Server.

About this task

Tuning parameters are specific to operating systems. Because these operating systems are not WebSphere Application Server products, be aware that the products can change and results can vary.

Note: Check your operating system documentation to determine how to make the tuning parameters changes permanent and if a reboot is required.

Procedure

Tune IBM i systems

Tuning IBM i systems

This topic describes how to tune the IBM i operating system to optimize the performance of WebSphere Application Server. Because the IBM i operating system is not a WebSphere Application Server product, be aware that the products can change and results can vary.

About this task

When you have a performance concern, check the operating system settings to determine if they are appropriate for your application.

Procedure

For detailed performance tuning, refer to the Tuning performance topic in the IBM i and System i Information Center.

Results

This tuning procedure improves performance of WebSphere Application Server on the IBM i operating system. After tuning your operating system for performance, consult other tuning topics for various tuning tips.

Tuning web servers for IBM i

The product provides plug-ins for several web server brands and versions. If you are running your web server on a non-IBM i platform, see the product documentation for performance tuning information.

About this task

For additional information, refer to Chapter 6 of the Performance Capabilities Reference Manual. This manual is available in the Performance Management Resource Library.

The IBM HTTP Server (powered by Apache) is a multi-process, multi-threaded server. To tune this web server:

Procedure

- Enable the access logs. The access logs record all incoming HTTP requests. Logging can degrade performance even though logging occurs in a separate process from the web server function.

  By default, the access log is disabled. It is recommended that you do not enable the access logs unless you need a record of all incoming HTTP requests.

  To enable the access logs:

  1. Open the IBM HTTP Server httpd.conf file, located in the /QIBM/ProdData/HTTPA/conf directory.
  2. Search for lines with the text CustomLog.
  3. Remove the hash mark (#) at the beginning of the line to enable a custom access log.
  4. Save and close the httpd.conf file.
  5. Stop and restart the IBM HTTP Server.
- Change the ThreadsPerChild directive setting. The ThreadsPerChild directive specifies the maximum number of concurrent client requests that the server processes at any time. The web server uses one thread for each request that it processes. The value specified for this directive does not represent the number of active clients.

  To change the ThreadsPerChild directive setting:

  1. Open the IBM HTTP Server httpd.conf file, located in the /QIBM/ProdData/HTTPA/conf directory.
  2. Search for the ThreadsPerChild directive.
  3. Change the setting. The default value is 40. It is recommended that you either use the default value or increase the value if you need to increase the number of concurrent client requests that the server can process at any time. You should not decrease the setting of this directive.
  4. Save and close the httpd.conf file.
  5. Stop and restart the IBM HTTP Server.

- Change the ListenBackLog directive setting. This directive specifies the length of the pending connections queue. When several clients request connections to the IBM HTTP Server, and all threads are in use, a queue is created to hold additional client requests.

  If you use the default Fast Response Cache Accelerator (FRCA) feature, the value specified for the ListenBackLog directive is ignored, because FRCA uses its own internal queue.

  To change the ListenBackLog directive setting:

  1. Open the IBM HTTP Server httpd.conf file, located in the /QIBM/ProdData/HTTPA/conf directory.
  2. Search for the ListenBackLog directive.
  3. Change the setting. For the IBM HTTP Server 1.3.26, the default setting is 1024 if FRCA is enabled, and 511 if FRCA disabled. It is recommended that you use these default values.
  4. Save and close the httpd.conf file.
  5. Stop and restart the IBM HTTP Server.

Tuning web servers

WebSphere Application Server provides plug-ins for several web server brands and versions. Each web server operating system combination has specific tuning parameters that affect the application performance.

About this task

Following is a list of tuning parameters specific to web servers. The listed parameters may not apply to all of the supported web servers. Check your web server documentation before using any of these parameters.

Procedure

- **Tune the IBM HTTP Server 2.0.47.1, Apache 2.0.48, IBM HTTP Server 6.0, and IBM HTTP Server 6.1.** Monitoring the CPU utilization and checking the IBM HTTP Server error_log and http_plugin.log files can help you diagnose web server performance problems.

  You can also configure the IBM HTTP Server to show a status page:
  - Edit the IBM HTTP Server httpd.conf file and remove the comment character (#) from the following lines in this file:

    ```
    #LoadModule status_module, modules/ApacheModuleStatus.dll,
    #<Location/server-status>
    #SetHandler server-status
    #</Location>
    ```
  - Save the changes and restart the IBM HTTP Server.
  - In a web browser, go to: http://*your_host*/server-status. Alternatively, click **Reload** to update status.
  - (Optional) If the browser supports refresh, go to http://*your_host*/server-status?refresh=5 to refresh every five seconds.

  All of these web servers allocate a thread to handle each client connection. Ensuring that enough threads are available for the maximum number of concurrent client connections helps prevent this tier from being a bottleneck. The settings for these web servers can be tuned by making changes to the httpd.conf file on the web server system.

  You can check the IBM HTTP Server error_log file to see if there are any warnings about having reached the maximum number of clients (MaxClients). There are several parameters, depending on the specific operating system platform, that determine the maximum number of clients the web server supports. See http://httpd.apache.org/docs-2.0/mod/mpm_common.html#maxclients for a description of the MaxClients parameters.
- **Support thousands of concurrent clients.** It is not unusual for a single IBM HTTP Server system to support thousands of concurrent clients. If your requirements are to support more concurrent clients than the number of threads that are supported by the web server operating system and hardware, consider using multiple web servers.

- **Respond to a Connection Refused error message**. Some clients might receive a Connection Refused error message if there is a sudden increase in the number of clients. Increasing the ListenBacklog and StartServer parameters can reduce or eliminate this error.
  - The ListenBacklog parameter indicates to the operating system the maximum allowed number of pending connections. Although the IBM HTTP Server default is 511, the actual value can be much higher or lower depending on the corresponding operating system parameter. To handle large numbers of simultaneous connections, this parameter and the corresponding OS parameter might need to be set to the number (possibly thousands) of expected simultaneous connections. (See the information about tuning operating systems for additional information on how to tune your operating system.
  - The StartServers parameter indicates the number of IBM HTTP Server processes to initially start. Pre-starting these IBM HTTP Server threads/processes reduces the chance of a user having to wait for a new process to start. You should set this parameter to a value equal to the MinSpareServers parameter so that the minimum number of IBM HTTP Server processes needed for this client load is started immediately.
- **Prevent the frequent creation and destruction of client threads/processes as the number of users change.** You can a use the MinSpareServers and MaxSpareServers to specify the minimum and maximum number of servers (client threads/processes) that can exist in an idle state. To prevent frequent creation and destruction of client threads/processes as the number of users change, set this range large enough to include the maximum number of simultaneous users.
- **Change the setting on the web server's Access logging parameter to reduce the load on the web server.** If you do not need to log every access to the Application Server, change the default value of the web server's Access logging parameter. This change will reduce the load on the web server.
- **Modify the settings of the Load balancing option and Retry interval web server plug-in properties to improve performance.** You can improve the performance of IBM HTTP Server (with the WebSphere web server plug-in) by modifying the following web server plug-in configuration properties:
  - Load balancing option, which specifies the load balancing option that the plug-in uses in sending requests to the various application servers associated with that web server.

    The goal of the default load balance option, Round Robin, is to provide an even distribution of work across cluster members. Round Robin works best with web servers that have a single process sending requests to the Application Server. If the web server is using multiple processes to send requests to the Application Server, the Random option can sometimes yield a more even distribution of work across the cluster.
  - Retry interval value, which specifies the length of time to wait before trying to connect to a server that has been marked temporarily unavailable.

    How can lowering the retry interval affect throughput ? If the plug-in attempts to connect to a particular application server and that application server is offline or in the process of restarting, the requests must wait for a timeout period. This process causes delayed responses for those requests. If you set the retry interval value too high, then an available application server is not utilized.

    Specify the retry interval value based on the following factors:

    - How long it will take for your application servers to restart
    - How averse you are to the delay caused by retrying too often
    - How important it is to utilize all of your application servers

  Making these changes can help the IBM HTTP Server to support more product users. To modify these properties, in the administrative console, click **Servers > Server Types > Web Servers >** *web_server_name* **> Plug-in properties > Request routing** .

Using Collection Services performance data

Collection Services, a component of WebSphere Application Server for IBM i, collects Performance Monitoring Infrastructure (PMI) data for stand-alone application servers. This PMI data is collected at specified intervals, providing a snapshot of activity during that time.

About this task

Perform the following steps enable Collection Services for WebSphere Application Server. When Collection Services is enabled, it writes the collected data into the following new database files:

- **QAPMWASCFG**

  Server configuration data. This data includes information about the different WebSphere Application Server jobs. This information is static and therefore does not change during life of the server. There is one record per server.
- **Server data**

  One record is created for each active server job per interval. Much of the data comes from WebSphere Application Server PMI data and transaction counters.
- **Application data**

  It contains one record for each application module per interval. WebSphere Application Server PMI counters are the source for many of the fields in this file. For the JavaServer Pages (JSP) section, the data in each field represents a sum over all JSPs running in a given application.
- **QAPMWASEJB EJB data**

  This data includes information about applications with Enterprise JavaBeans (EJB) running on the WebSphere Application Server. Each record represents one type of enterprise beans per application module per interval. If there is no bean activity for a particular EJB type, then no record is written.
- **QAPMWASRSC**

  Pooled resource data. This data includes information about pooled resources associated with WebSphere Application Server. Each record represents one pooled resource per interval. The type of pooled resource might be a Java Database Connectivity (JDBC) connection pool, a J2EE Connector (J2C) connection pool, or a thread pool. Not all fields are applicable to each pooled resource type. If a resource exists but is not being used (nothing is created, destroyed, allocated or returned), no record is written.

Procedure

1. Install and enable the Collection Services custom service:
   a. Type the following command on an IBM i command line to start the Qshell environment:
      ```
      STRQSH
      ```
   b. Type the following command:
      ```
      cd app_server_root/bin
      ```
   c. Type the following command to install and enable the Collection Services custom service:
      ```
      manageWASCollectionServices -profileName pName -server.name sName -enable
      ```

      where *pName* is the name of the WebSphere Application Server profile to manage and *sName* is the name of the Application Server to manage.
2. Enable PMI and set the level using the administrative console.
3. Stop and restart the application server.
4. Perform the following steps to configure Collection Services:
   - On an IBM i command line, type CFGPFRCOL (Configure Performance Collection) and press the F4 key.
   - Make note of the collection library specified to create the management collection object (Collection library parameter, LIB keyword).
   - Make note of whether or not database files are automatically created (Create database files parameter, CRTDBF keyword).
5. Type the following command to start the performance collection:
   ```
   STRPFRCOL
   ```

6. If you configured Collection Services with CRTDBF(*YES), new records are written to all of the above files at every collection interval. If CRTDBF(*NO) is specified, no database files are created automatically, you need to do it manually using the CRTPFRDTA command. Here are two examples:
   - To create all the Collection Services files, type the following command:
     ```
     CRTPFRDTA FROMMGTCOL(MYLIB/Q123456789)
     ```
   - To create the new WebSphere Application Server files only , type the following command:
     ```
     CRTPFRDTA FROMMGTCOL(MYLIB/Q123456789)
     ```

     CRTPFRDTA FROMMGTCOL(MYLIB/Q123456789) CGY(*WAS)

What to do next

For more information about Collection Services, see the IBM i and System i Information Center at: http://www.ibm.com/eserver/iseries/infocenter

The manageWASCollectionServices script

The manageWASCollectionServices script manages the WebSphere Application Server for IBM i Collection Services custom service, which collects select Performance Monitoring Infrastructure (PMI) data and other information for IBM i Collection Services.

Syntax

The syntax of the script is:

```
manageWASCollectionServices [ -profileName profile_name ] [ wsadmin_options ]
[ -server.name server_name ] [-enable|-disable|-status] [-help]
```

Parameters

The parameters of the script are:

- **-profileName**

  The name of the profile to manage. The default profile is used if no profile is specified.
- **-server.name**

  The name of the application server to manage. The default value is the name of the profile, or `server1` if the profile is named `default`.
- **-enable|-disable|-status**

  One of the following actions must be specified:
  - **-enable**

    Enables Collection Services on the specified server. If Collection Services is already enabled, it does nothing.
  - **-disable**

    Disables Collection Services on the specified server. If Collection Services is already disabled, it does nothing.
  - **-status**

    Displays whether Collection Services is enabled on the specified server.
- **-help**

  Displays help for the script.

processStats script

The processStats script collects and summarizes Performance Monitoring Infrastructure (PMI) data, saving the results to a text file. This file is imported into the IBM Systems Workload Estimator as the basis for a WebSphere Application Server estimate.

Syntax

```
processStats [ -host host_name ] [ -port soap_port ] [ -cycles cycles ]
[ sampleTime hours:minutes:seconds ] [ statLevel all|extended|base|noChange ]
[ -user user_name ] [ -password user_password ] [ -profileName profile_name ]
[-verbose] [-help | ? ]
```

Parameters

- **-host**

  Application Server host name. Default is `localhost`.
- **-port**

  The Application Server SOAP port. Default is `8880`.
- **-cycles**

  The total number of times statistics are sampled. Default is `12`.
- **-sampleTime**

  The length of the interval between samples. Default is 5 minutes. The time format is *hours:minutes:seconds*, where:
  - *hours* (required) is an integer less than or equal to 24
  - *minutes* (optional) is an integer thess than 60
  - *seconds* (optional) is a decimal less than or equal to 59.999

  Example: `0:12:3.23` specifies an interval of 12 minutes and 3.23 seconds.
- **-statLevel**

  The PMI statistics level. Accepted values are: `all`, `extended`, `base` and `noChange`. Default is `noChange`.
- **-user**

  The user IDof the WebSphere Application Server administrator when the server is running in secure mode.
- **-profileName**

  The name of the profile to run processStats. If `-profileName` is not specified, the default profile is used.
- **-verbose**

  Enables tracing.
- **-help**

  Displays help for the script..

For further information on IBM Systems Workload Estimator refer to: Collecting and Importing PMI Data .

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
: The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server Network Deployment is the `/QIBM/ProdData/WebSphere/AppServer/V8/ND` directory.

java_home

Table 4. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server Network Deployment is the `/QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/`*`profile_name`* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server Network Deployment is the `/QIBM/UserData/WebSphere/AppServer/V8/ND` directory.

The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Tuning the application server using pre-defined tuning templates

You can use the python-based tuning script, applyPerfTuningTemplate.py, along with one of its template files, to apply pre-defined performance tuning templates to your application server or cluster. The script, and these property-based template files are located in the <*WAS_HOME*>/scriptLibraries/perfTuning/V70 directory.

Before you begin

bprac: The configuration settings applied by this script and the associated tuning templates should be viewed as potential performance tuning options for you to explore or use as a starting point for additional tuning. The configuration settings that each of the pre-defined templates applies are geared towards optimizing common application server environments or scenarios. Typically, these settings improve performance for many applications.

Because optimizing for performance often involves trade-offs with features, capabilities, or functional behavior, some of these settings might impact application correctness, while other settings might be inappropriate for your environment. Please review the documentation below and consider the impact of these settings to your application inventory and infrastructure.

As with any performance tuning exercise, the settings configured by the predefined templates should be evaluated in a controlled preproduction test environment. You can then create a customized template to refine the tuning settings to meet the specific needs of your applications and production environment.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log , SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Typically, when you run the applyPerfTuningTemplate.py script, you will specify either the production.props template file or the development.props template file to apply against the target server or cluster.

- If you specify the production.props template file when you run the applyPerfTuningTemplate.py script, the script applies configuration settings that are appropriate for a production environment where application changes are rare and optimal runtime performance is important.
- If you specify the development.props template file when you run the applyPerfTuningTemplate.py script, the script applies configuration settings that are appropriate for a development environment where frequent application updates are performed and system resources are at a minimum.

In addition to these two common templates, a third template file, default.props, is provided to enable you to revert the server configuration settings back to the out-of-the-box defaults settings.

You can also create your own custom tuning template. To create a custom tuning template, copy one of the existing templates, modify the configuration settings to better fit the needs of your applications and environment, and then use the applyPerfTuningTemplate.py script to apply these customized settings. The script and properties files leverage the property file configuration management features that wsadmin provides, and can easily be augmented to tune additional server components. See the topic Using properties files to manage system configuration for more information.

About this task

Review the following table to see the configuration changes that occur based on the template file that you specify when you run the applyPerfTuningTemplate.py script. A blank cell in this table indicates that the listed parameter is not configured, or is configured back to the default settings for the server defaults.

Table 5. Tuning parameters and their template values. The table includes the tuning parameter and its value for the default template, the production template and the development template.

Parameter	Server default (default.props template file)	Production environment (production.props template file)	Development environment (development.props template file)
JVM Heap Size (MB) See the topic Tuning the IBM virtual machine for Java for more information about this setting.	50 min / 256 max	512 min / 512 max	256 min / 512 max
Verbose GC See the topic Tuning the IBM virtual machine for Java for more information about this setting.	disabled	enabled	enabled
JVM Diagnostic Trace (Generic JVM Arguments) See the topic Tuning the IBM virtual machine for Java for more information about this setting.		-Dcom.ibm.xml.xlxp.jaxb .opti.level=3	-Dcom.ibm.xml.xlxp.jaxb .opti.level=3
HTTP (9080) and HTTPS (9443) Channel maxKeepAliveRequests See the topic HTTP transport custom properties for more information about this setting.	100	10000	10000
TCP Channel maxOpenConnections	20000	500	500
TCP Channel listenBacklog	511	128	128
Development Mode See the topic Application server settings for more information about this setting.	disabled		enabled
Server Component Provisioning See the topic Application server settings for more information about this setting.	disabled	enabled	enabled

Table 5. Tuning parameters and their template values (continued). The table includes the tuning parameter and its value for the default template, the production template and the development template.

Parameter	Server default (default.props template file)	Production environment (production.props template file)	Development environment (development.props template file)
PMI Statistic Set See the topic Enabling PMI data collection for more information about this setting.	basic	none	none
Authentication Cache Timeout See the topic Authentication cache settings for more information about this setting.	10 minutes	60 minutes	60 minutes
Data Source Connection Pool Size* See the topic Connection pool settings for more information about this setting.	1 min / 10 max	10 min / 50 max	
Data Source Prepared Statement Cache Size* See the topic WebSphere Application Server data source properties for more information about this setting.	10	50	
ORB Pass-by-Reference** See the topic Request Broker service settings for more information about this setting.	disabled	enabled	enabled
Web Server Plug-in ServerIOTimeout	900	900	900
Thread Pools (Web Container, ORB, Default) See the topic Thread pool settings for more information about this setting.	50 min / 50 max, 10 min / 50 max, 20 min / 20 max		5 min / 10 max

Table notes:

* Indicates items that are tuned only if they exist in the configuration. For example, a data source connection pool typically does not exist until an application is installed on the application server. If these items are created after your run the script, they are given the standard server default values unless you specify other settings.

** Enabling ORB Pass-By-Reference can cause incorrect application behavior in some cases, because the Java EE standard assumes pass-by-value semantics. However, enabling this option can improve performance up to 50% or more if the EJB client and server are installed in the same instance, and your application is written to take advantage of these feature. The topic Object Request Broker service settings can help you determine if this setting is appropriate for your environment.

Following are a few subtle platform-specific tuning differences:

Procedure

- Start the wsadmin tool if it is not already running, and then complete one of the following actions to tune an application server or all of the application servers in a cluster.
- Run the applyPerfTuningTemplate.py script to tune a specific server or cluster of servers running in a production environment.

  ```
  wsadmin -f applyPerfTuningTemplate.py
  [-nodeName node_name -serverName server_name][clusterName cluster_name] -templateFile production.props
  ```

- Run the applyPerfTuningTemplate.py script to tune a specific server or cluster of servers running in a development environment.

  ```
  wsadmin -f applyPerfTuningTemplate.py
  [-nodeName node_name -serverName server_name][clusterName cluster_name] -templateFile development.props
  ```

- Run the applyPerfTuningTemplate.py script to change the settings for a server or a cluster back to the standard out-of-the-box default configuration settings.

  ```
  wsadmin -f applyPerfTuningTemplate.py
  [-nodeName node_name -serverName server_name][clusterName cluster_name] -templateFile default.props
  ```

What to do next

Conduct a performance evaluation, and tuning exercise to determine if you should further fine tune the server for your specific applications.

Chapter 5. Troubleshooting performance problems

This topic illustrates that solving a performance problem is an iterative process and shows how to troubleshoot performance problems.

About this task

Solving a performance problem is frequently an iterative process of:

- Measuring system performance and collecting performance data
- Locating a bottleneck
- Eliminating a bottleneck

This process is often iterative because when one bottleneck is removed the performance is now constrained by some other part of the system. For example, replacing slow hard disks with faster ones might shift the bottleneck to the CPU of a system.

Measuring system performance and collecting performance data

- Begin by choosing a *benchmark*, a standard set of operations to run. This benchmark exercises those application functions experiencing performance problems. Complex systems frequently need a warm-up period to cache objects, optimize code paths, and so on. System performance during the warm-up period is usually much slower than after the warm-up period. The benchmark must be able to generate work that warms up the system prior to recording the measurements that are used for performance analysis. Depending on the system complexity, a warm-up period can range from a few thousand transactions to longer than 30 minutes.
- If the performance problem under investigation only occurs when a large number of clients use the system, then the benchmark must also simulate multiple users. Another key requirement is that the benchmark must be able to produce repeatable results. If the results vary more than a few percent from one run to another, consider the possibility that the initial state of the system might not be the same for each run, or the measurements are made during the warm-up period, or that the system is running additional workloads.
- Several tools facilitate benchmark development. The tools range from tools that simply invoke a URL to script-based products that can interact with dynamic data generated by the application. IBM Rational® has tools that can generate complex interactions with the system under test and simulate thousands of users. Producing a useful benchmark requires effort and needs to be part of the development process. Do not wait until an application goes into production to determine how to measure performance.
- The benchmark records throughput and response time results in a form to allow graphing and other analysis techniques. The performance data that is provided by WebSphere Application Server Performance Monitoring Infrastructure (PMI) helps to monitor and tune the application server performance. See the information on why use request metrics to learn more about performance data that is provided by WebSphere Application Server. Request metrics allows a request to be timed at WebSphere Application Server component boundaries, enabling a determination of the time that is spent in each major component.

Locating a bottleneck

Consult the following scenarios and suggested solutions:

- **Scenario:** Poor performance occurs with only a single user.

  Suggested solution: Utilize request metrics to determine how much each component is contributing to the overall response time. Focus on the component accounting for the most time. Use Tivoli Performance Viewer to check for resource consumption, including frequency of garbage collections. You might need code profiling tools to isolate the problem to a specific method. See the *Administering applications and their environment* PDF for more information.
- **Scenario:** Poor performance only occurs with multiple users.

Suggested solution: Check to determine if any systems have high CPU, network or disk utilization and address those. For clustered configurations, check for uneven loading across cluster members.

- **Scenario:** None of the systems seems to have a CPU, memory, network, or disk constraint but performance problems occur with multiple users.

  Suggested solutions:
  - Check that work is reaching the system under test. Ensure that some external device does not limit the amount of work reaching the system. Tivoli Performance Viewer helps determine the number of requests in the system.
  - A thread dump might reveal a bottleneck at a synchronized method or a large number of threads waiting for a resource.
  - Make sure that enough threads are available to process the work both in IBM HTTP Server, database, and the application servers. Conversely, too many threads can increase resource contention and reduce throughput.
  - Monitor garbage collections with Tivoli Performance Viewer or the verbosegc option of your Java virtual machine. Excessive garbage collection can limit throughput.

Eliminating a bottleneck

Consider the following methods to eliminate a bottleneck:

- Reduce the demand
- Increase resources
- Improve workload distribution
- Reduce synchronization

Reducing the demand for resources can be accomplished in several ways. Caching can greatly reduce the use of system resources by returning a previously cached response, thereby avoiding the work needed to construct the original response. Caching is supported at several points in the following systems:

- IBM HTTP Server
- Command
- Enterprise bean
- Operating system

Application code profiling can lead to a reduction in the CPU demand by pointing out hot spots you can optimize. IBM Rational and other companies have tools to perform code profiling. An analysis of the application might reveal areas where some work might be reduced for some types of transactions.

Change tuning parameters to increase some resources, for example, the number of file handles, while other resources might need a hardware change, for example, more or faster CPUs, or additional application servers. Key tuning parameters are described for each major WebSphere Application Server component to facilitate solving performance problems. Also, the performance advisors page can provide advice on tuning a production system under a real or simulated load.

Workload distribution can affect performance when some resources are underutilized and others are overloaded. WebSphere Application Server workload management functions provide several ways to determine how the work is distributed. Workload distribution applies to both a single server and configurations with multiple servers and nodes.

See the *Administering applications and their environment* PDF for more information.

Some critical sections of the application and server code require synchronization to prevent multiple threads from running this code simultaneously and leading to incorrect results. Synchronization preserves correctness, but it can also reduce throughput when several threads must wait for one thread to exit the critical section. When several threads are waiting to enter a critical section, a thread dump shows these threads waiting in the same procedure. Synchronization can often be reduced by: changing the code to

only use synchronization when necessary; reducing the path length of the synchronized code; or reducing the frequency of invoking the synchronized code.

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server Network Deployment is the `/QIBM/ProdData/WebSphere/AppServer/V8/ND` directory.

java_home

Table 6. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server Network Deployment is the `/QIBM/UserData/WebSphere/AppServer/V8/ND/profiles/`*`profile_name`* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server Network Deployment is the `/QIBM/UserData/WebSphere/AppServer/V8/ND` directory.

The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

APACHE INFORMATION. This information may include all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see http://www.apache.org. You may obtain a copy of the Apache License at http://www.apache.org/licenses/LICENSE-2.0.

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

IBM Director of Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Trademarks and service marks

IBM, the IBM logo, and 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. For a current list of IBM trademarks, visit the IBM Copyright and trademark information Web site (www.ibm.com/legal/copytrade.shtml).

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Index

A

application server environment
  tuning 23
applications
  tuning 1

C

collection services
  performance 47

D

directory
  installation
    conventions 25, 36, 49, 59

J

JVM
  tuning 27

P

performance
  tuning 1, 7, 8
Performance and Diagnostic Advisor 9

T

Tivoli Performance Viewer
  tuning 19, 20
troubleshooting
  performance 55
tuning 12
  application server environment 23
  applications 1
  best practices 16
  buffer sizes 26
  diagnostic alerts 10
  heap dumps 18
  heap monitor 20, 21
  JVM 27
  memory leaks 17
  operating systems 43
    IBM i 44
  parameters 23
  performance 1, 5, 7, 8, 19, 47, 55
  Performance and Diagnostic Advisor 9
  settings 13, 15, 42
  transport channel services 38
  web server 45
    IBM i 44
  wsadmin scripts 48

W

web server
  tuning 45
    IBM i 44
wsadmin scripts
  managing collection services 48
  processing statistics 49