Chapter 11 of *WebSphere Application Server V7 Administration and Configuration Guide*, SG24-7615

WebSphere Application Server V7: Monitoring the Runtime

Being able to measure and monitor system interactions helps IT in providing business continuity. Monitoring capabilities play a key role in successfully managing enterprise systems. In WebSphere® Application Server, there are a number of tools that can contribute to an organizations monitoring strategy and provide insights into the performance of the application server.

In this chapter, we provide an introduction to these toolsets.

We cover the following topics:

Overview

IT environments are complex, involving many different servers working together to deliver the electronic functions of business. In a single user interaction, it is typical that information can be retrieved from many systems. Consider the very simple distributed WebSphere Application Server environment in Figure 1.

The stars in the figure highlight that even a simple Web application request can pass through a whole series of dependent servers in order to successfully complete a request. JEE is a component based architecture, requiring a request to interact and use ‘n’ number of these components to complete. Monitoring system components and their performance can become complex, yet is critical to understanding the overall performance of an application.

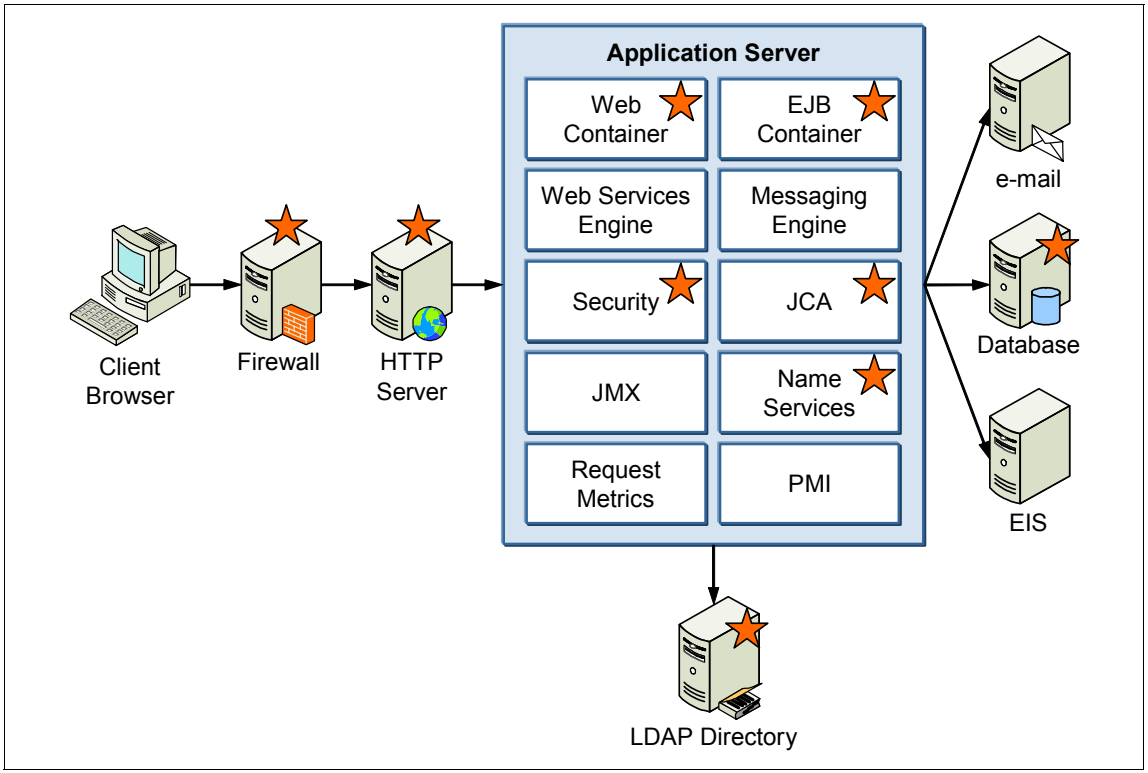


Figure 1 Simple Web system topology

Monitoring the systems contributes to overall systems management by:

- Establishing an understanding of the performance baseline and of what runtime behaviors constitute “normal” operations

- Measuring performance and identifying poorly performing systems and components
- Identifying service failures, and can assist in root cause identification

WebSphere Application Server monitoring tools rely primarily on information gathered from two core data infrastructures:

- Performance Monitoring Infrastructure (PMI), which is a collection of statistical agents scattered through out the application server that gather statistical data on the performance of the application server components.

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/cprf_pmidata.html

- Request metrics, which are primarily a set of timing agents that track a request as it navigates the components of the application server. A key differentiation of request metrics is that they are measured at the request level. The focus of a request metric is to record the time spent by individual requests in different components of the application and at the end of the request provide a record of where time was spent in the request.

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tprf_requestmetrics.html

Monitoring scenarios

This chapter demonstrates the use of the system monitoring infrastructures by using different monitoring scenarios, which are summarized in Table 1.

Note: These scenarios cover several common uses of the monitoring tools, but it should be understood that many of the different types of data are not explicitly discussed. This chapter provides an introduction to the tool sets and a way for new administrators to get started, and serves as a reminder for experienced administrators about some of the tools that they might not have utilized in some time.

Table 1 Monitoring Scenario Summary

Monitoring Scenario	Chapter Reference
How is monitoring data activated and what are the monitoring choices? - PMI data defaults - Enabling request metrics	"Enabling monitoring infrastructures" on page 6 - "PMI defaults and monitoring settings" on page 6 - "Enable request metrics" on page 15

Monitoring Scenario	Chapter Reference
What are the tools that I can use for understanding the collected data? ► Tivoli® Performance Viewer.	“Viewing the monitoring data” on page 22
Understanding how application(s) are interacting with a database. This scenario helps identify data that will help investigate: ► Is the database responding fast enough? ► Is there enough connections to the database? ► Are the connections being returned to the pool?	“Database interactions” on page 34
Understanding JCA connection pool utilization. This scenario helps administrators understand: ► What is the response like for the JCA connection pools. ► Are there enough connections to support the system interactions? ► Are the connections being returned to the pool?	“Database interactions” on page 34 “JCA interactions” on page 35
What about threading resources. Is there sufficient JMS, EJB™, and Web threads allocated in the server thread pools? This scenario helps administrator understand: ► How to monitor and improve system related throughput. ► Current limits on system concurrency.	“Threading resources” on page 36
Monitoring memory allocation and garbage collection. JVM™ memory tuning is vital to application server performance, and this scenario introduces administrators to tools that assist in making memory tuning choices.	“JVM memory usage” on page 40
Finding bottlenecks in response time, Services, EJB, Web and response times. Understanding of component response time helps in the identification of application bottleknecks and performance issues.	“Request level details” on page 41
Special features of ITCAM for WebSphere in WebSphere Application Server v7.0	“ITCAM for WebSphere” on page 47

The monitoring infrastructure scenarios are demonstrated using the example environment shown in Figure 2.

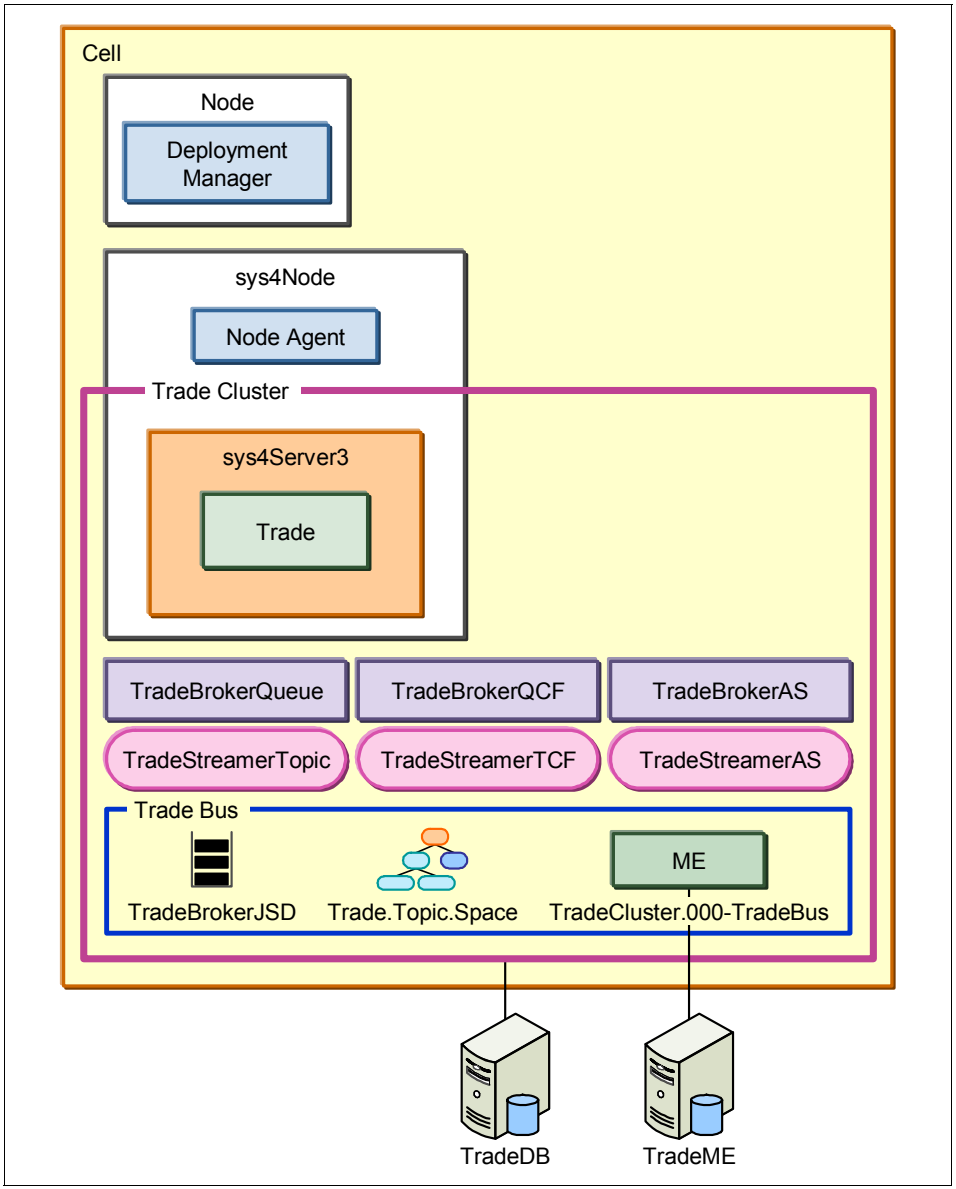


Figure 2 Example monitoring environment

Note: No special tuning has been done in the test environment. The Trade performance application was simply installed into a base server configuration, with the exception of default memory was changed.

Enabling monitoring infrastructures

This section shows you how to enable the PMI monitoring infrastructure and the request metrics that provide the performance data.

PMI defaults and monitoring settings

The enabling of PMI data is managed on a server-by-server basis. In the administrative console, do the following steps:

1. Navigate to the **Performance Monitoring Infrastructure (PMI)** menu item in the **Monitoring and Tuning** navigation menu.
2. Select the link for server for that you want to manage the PMI controls for. In this example, sys4Server3 is the server. Figure 3 shows the PMI configuration panel for the server.

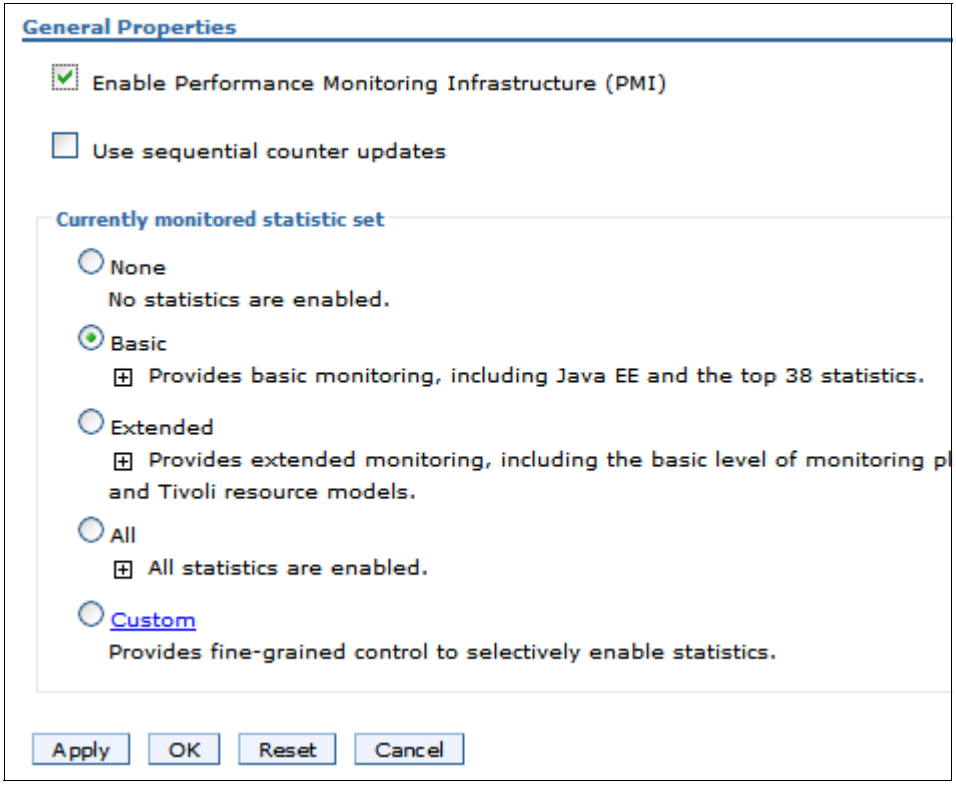


Figure 3 Default PMI settings

On this panel it is worthy to note that:

– PMI is enabled by default.

– The default statistical set is the basic set.

The PMI data can be changed at runtime using settings in the Runtime tab.

Note: Disabling and enabling of PMI data requires a server restart.

The enabling and disabling of PMI is not available on the Runtime tab. However, the monitoring level can be set to None in a server with PMI enabled using the Runtime tab.

Understanding the sets of PMI statistics

The PMI statistic sets represent a group of individual statistical agents. The types of statistics that PMI can collect are classified.

Information about these classes can be found in the WebSphere Information Center on the *PMI data classification* page at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/rprf_dataclass.html

Everyone working in system administration knows that every action executed in an environment has a cost. Monitoring is no different, and for PMI, the cost of monitoring is impacted primarily by two factors:

1. The amount of data that is monitored.
2. The overhead of individual performance metrics. Not all metrics have the same collection cost.

With PMI, there are multiple sets of statistics that can be enabled as shown in Figure 3 on page 7. These sets of statistics are:

- None
- Basic
- Extended
- All
- Custom

None and All are fairly self explanatory, so here we take a closer look at the options provided by Basic, Extended, and Custom.

Basic statistic set

The Basic statistic set is the default setting, The basic setting is configured with the intention of providing an overall understanding of application server health, including statistics as outlined in the JEE specification, as well as other common performance hotspots and key monitoring points for JEE applications. Later, we discuss how to determine the overhead and level of a statistic (see “Getting more information about statistics sets” on page 12).

Figure 4 shows the list of PMI counters that are active for the basic PMI data level. For details on each counter, refer to “Getting more information about statistics sets” on page 12.

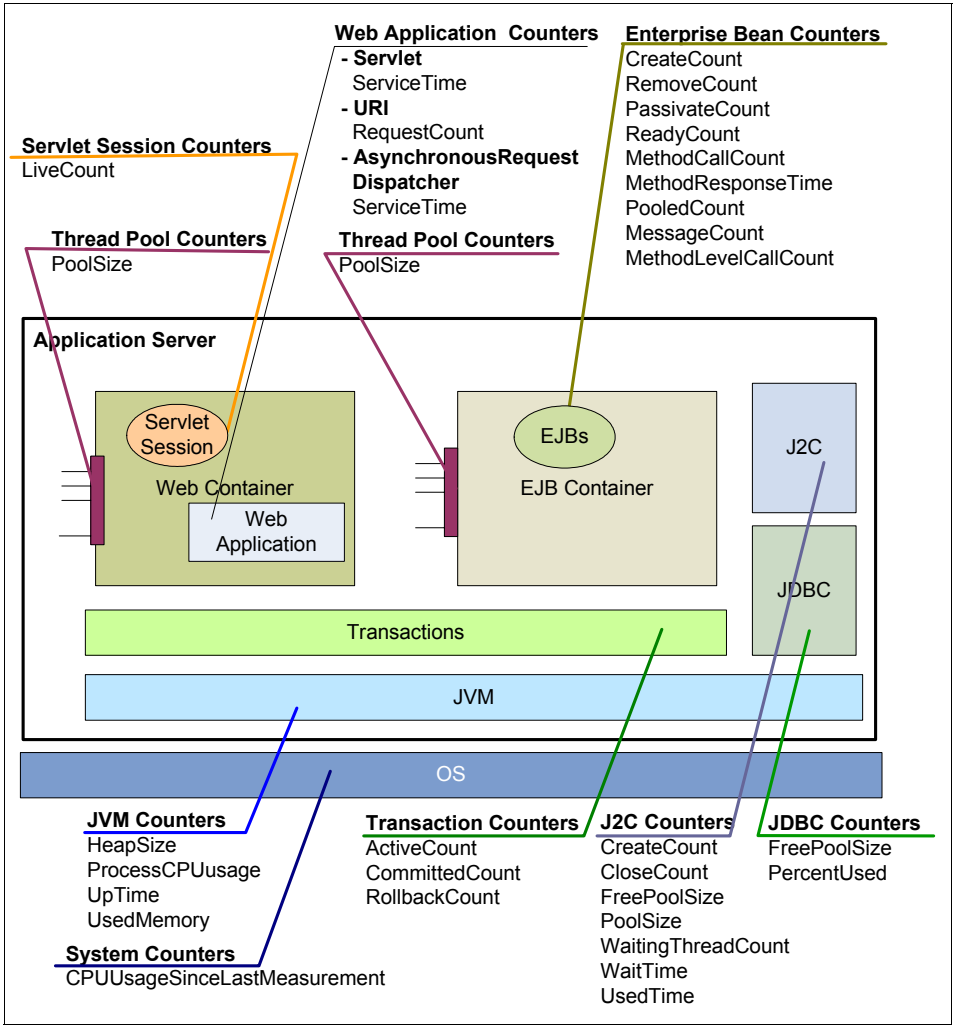


Figure 4 PMI basic counters

Extended statistic set

The extended PMI data set has the basic set as well as some additional statistics with a particular emphasis on statistics that look at the load on the server and the servers response to the load being applied. The statistical agents in the extended set might or might not apply to a JEE application depending on the individual application architecture and environment configurations.

Figure 5 shows the extended metrics that are additional to those of the basic configuration.

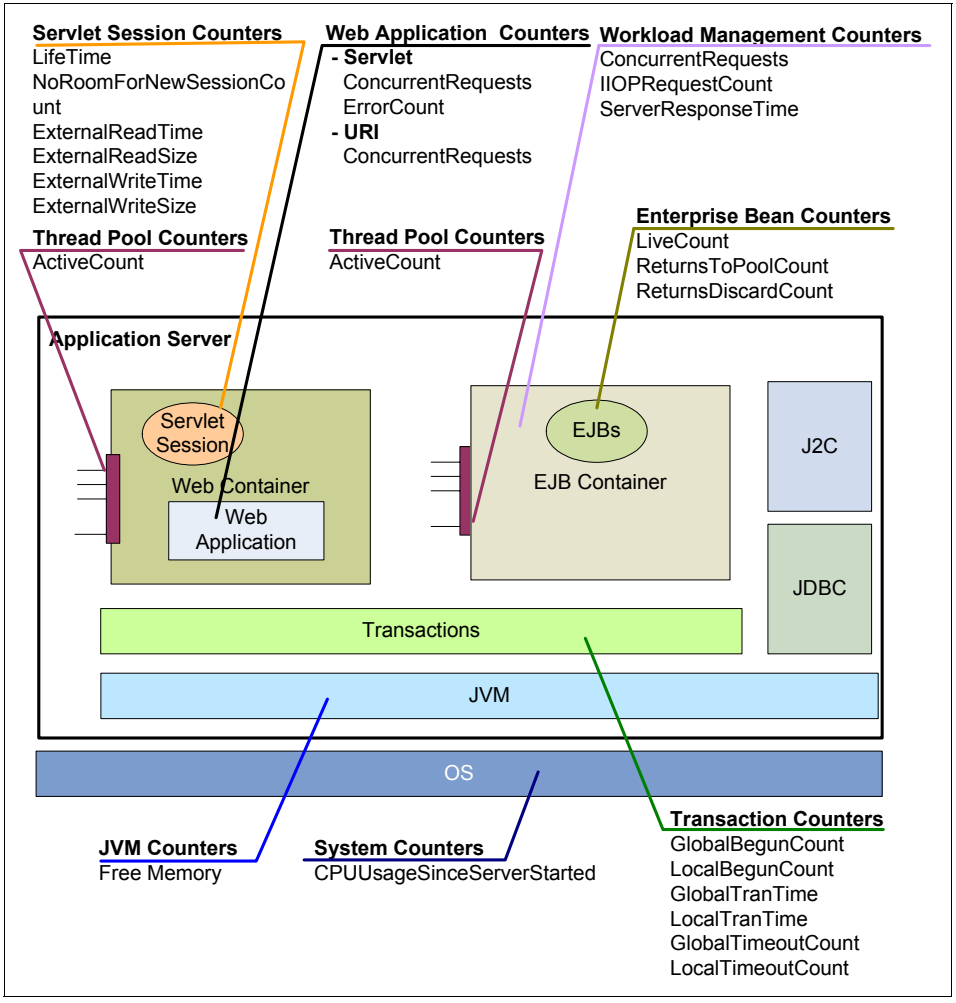


Figure 5 PMI extended counters

Custom statistic set

The Custom PMI data collection set allows the administrator to choose the counters that are most appropriate for the application(s) that are deployed on the server. Each counter is individually activated. This is the most powerful configuration but requires that the administrator spend some time reviewing the available statistical counters and also that the administrator understands the type of counter that is useful for the applications.

For example, consider the counters activated for ServletSession if the extended data set is selected. The counters are:

- LiveCount
- LifeTime
- NoRoomForNewSessionCount
- ExternalReadTime
- ExteranlReadSize
- ExternalWriteTime
- ExteranlWriteSize

The **NoRoomForNewSessionCount** counter only applies if the **Allow overflow** from the Web container session management was changed from its default value of true. This attribute is shown in Figure 6.

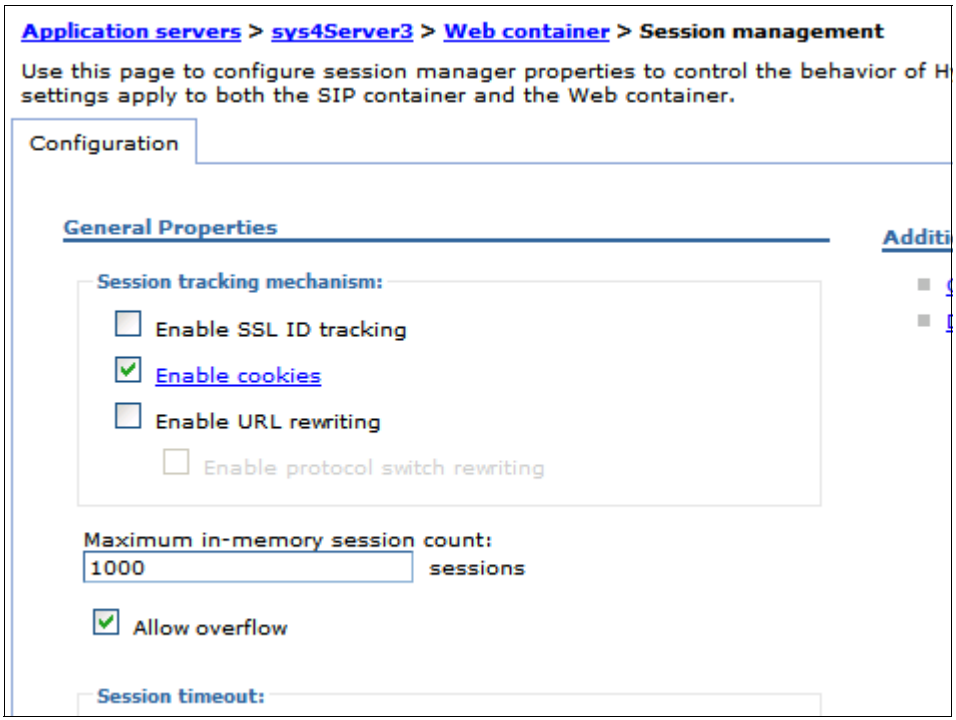


Figure 6 Allow overflow default is true

Similarly, the counters related to external session management only apply if session persistence is configured. Hence the activation of the extended session information does little for an application where the overflow option is not modified and session persistence is not configured.

With a custom metric approach, the administrator could choose to simply add LiveCount and LifeTime counters as well as perhaps choosing other metrics of interest, such as the TimeoutInvalidationCount, to measure how many sessions are being timed out rather than logged off.

Tip for using custom PMI settings: The counters used for the basic set can be customized to form a baseline for the custom counter activation. They should be supplemented with additional counters that are relevant for the application types that are being deployed.

Overhead of PMI

The actual overhead of each statistic level varies depending on the particular applications on the server and load that is being executed by the server. In the WebSphere Information Center, each counter has a documented qualitative overhead level to indicate the type of overhead it will incur (see “Getting more information about statistics sets” on page 12). This is not intended to prevent administrators from using counters with high overhead. It is important to remember that the overhead is a relative measurement, and the administrator needs to balance the need for the data versus the overhead incurred to enable a particular counter temporarily or for the long term.

Note: If running in a stand alone server configuration, it is important to remember that the data collection and statistical counters are all working in the same JVM as the applications. In this circumstance, the administrator should anticipate that the activation of additional PMI data can incur additional CPU and memory usage by the JVM.

The approximate overhead of the PMI statistic sets are as follows:

- Basic overhead up to 2%
- Extended overhead up to 3%
- All overhead of up to 6%
- Custom will depend on the counters enabled but it is reasonable to expect somewhere between 2%-6%

Getting more information about statistics sets

The WebSphere Information Center has extensive information to assist the administrator in understanding exactly which metrics are set for a particular level, and to appreciate the potential overheads of using the statistics.

If we consider the number of components that make up an application server as shown in Figure 7 on page 13, it should be no surprise that there are many PMI counters available to help monitor the application server.

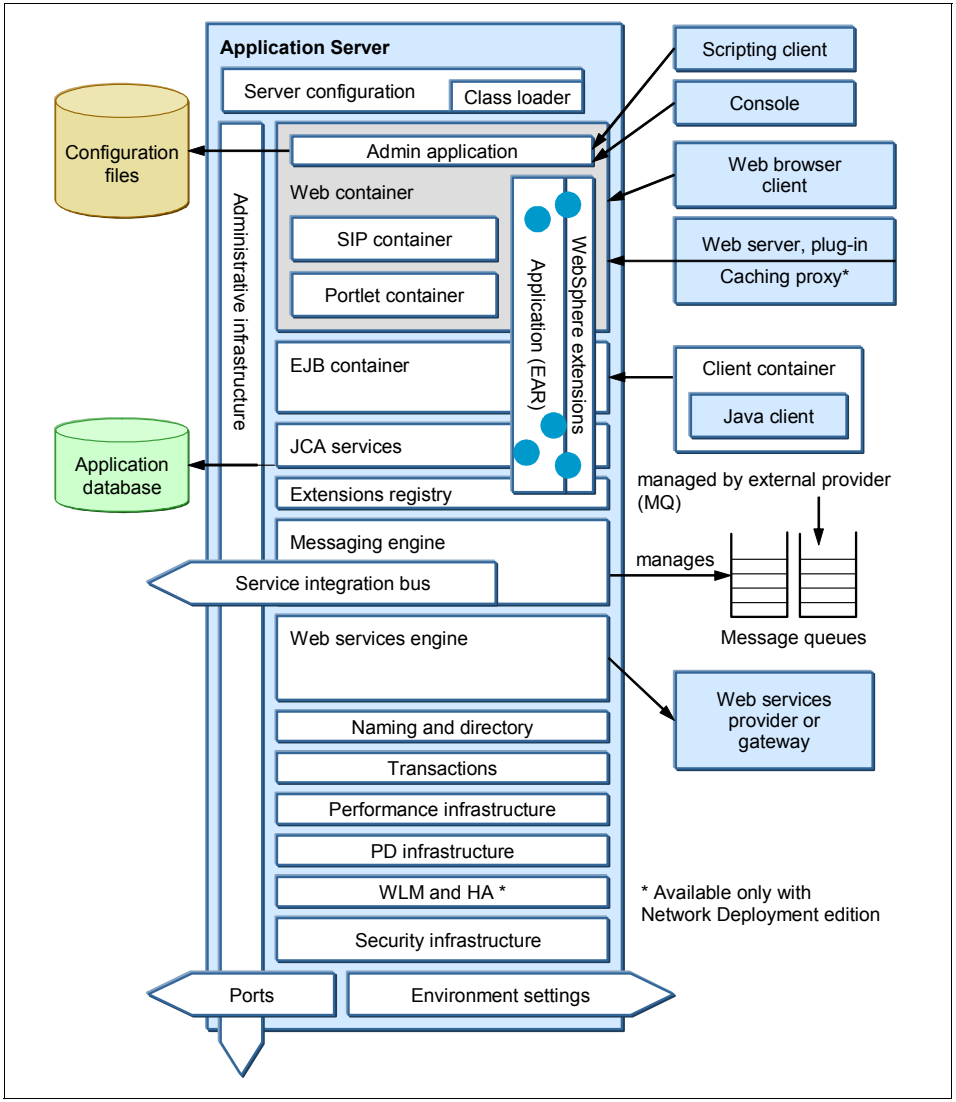


Figure 7 Application server components

The WebSphere Information Center provides a summary of PMI counters to help administrators understand the variety of counters that are available in each of the different counter classifications in the application server. A good place to start with gathering information is the article, *Enabling PMI data collection*, at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/tprf_pmi_encoll.html

Figure 8 shows the links found at the bottom of this article, taking you to a page with more information about each counter.

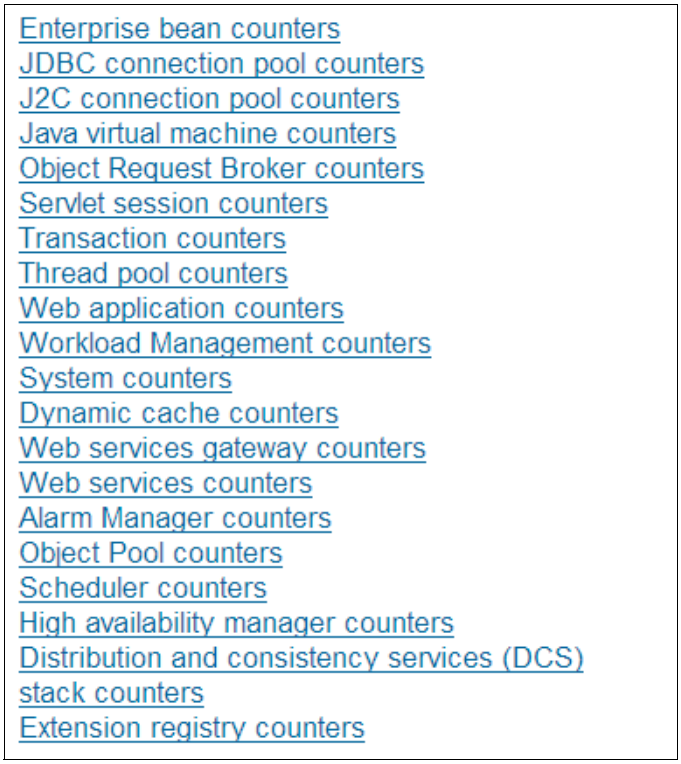


Figure 8 PMI counter types

From the links to the counter classifications, it is possible to navigate and view the individual counters that each counter classification contains.

The following topics in the information center can provide more information:

- General PMI data organization:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/rprf_dataorg.html`

- WebSphere Application Server supports the eclipse framework for extensible applications. A key part of this framework is the implementation of the Extension registry. These counters are only relevant when referring to extensible applications. For more information, see:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.iseries.doc/info/iseries/ae/cweb_extensions.html`

- Service integration bus counters:
  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/rprf_sibcounter.html

Enable request metrics

The enabling of request metrics is a cell wide configuration and when activated, it is activated for all servers in the cell. In the administrative console:

1. Navigate to the **Request Metrics** menu item in the **Monitoring and Tuning** navigation menu (Figure 9).

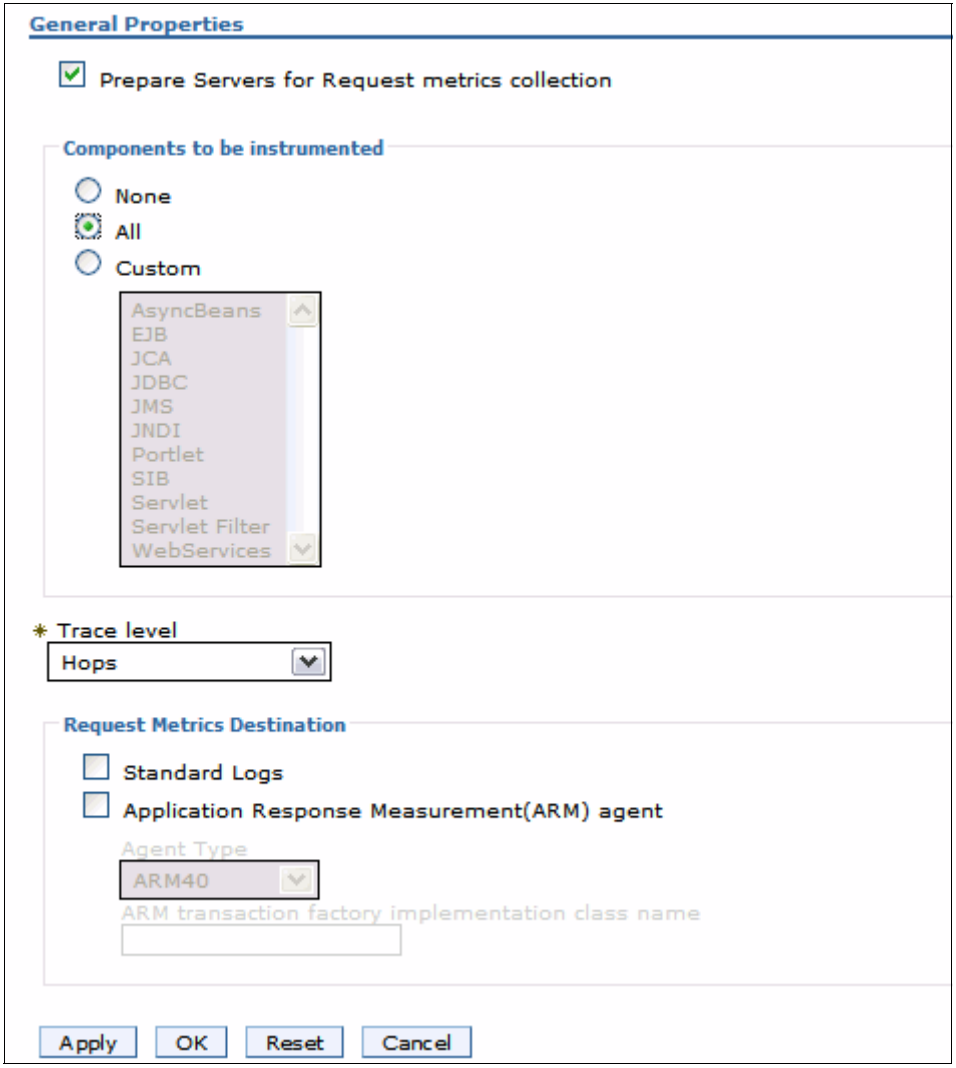


Figure 9 Request metrics panel

Request metrics are enabled by:

a. Checking the **Prepare servers for request metrics collection**

b. Choosing a monitoring level from the **Components to be instrumented** section of the panel.

c. Choosing a trace level, and

d. Choosing a destination from the **Request Metrics destination** section of the **Request Metrics** panel.

When configured, the servers must be restarted for request metrics to be enabled. The servers must also be stopped when disabling request metrics.

Understanding component instrumentation and trace levels

Trace levels and component instrumentation work together to determine if the request is instrumented. The component instrumentation levels are shown in Figure 10.

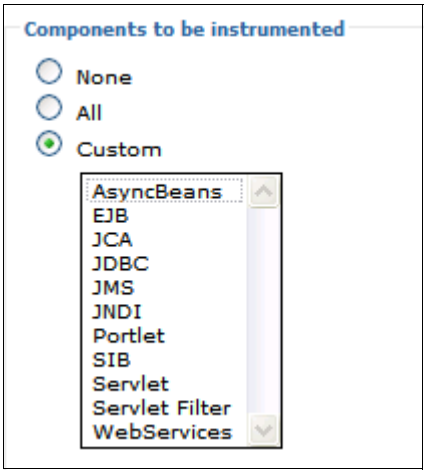


Figure 10 Components to be instrumented

If you select **All**, all components will be monitored based on trace level settings.

If you select **Custom**, you can select the components to be monitored. Data will be collected from the components if the trace level also calls for the capturing of data from this component.

Note: When a component is defined as an edge component, meaning the request enters or exits the application server through the component, then this component is instrumented even is it is not selected as part of the custom component listing.

Working in conjunction with the component instrumentation levels is the trace level (Figure 11).

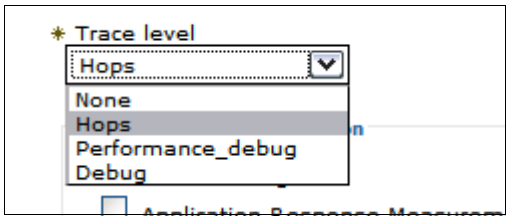


Figure 11 Request metric trace levels

The following trace levels are possible:

- None: No instrumentation is generated.
- Hops: Generates instrumentation information about process boundaries only. When this setting is selected, you see the data at the application server level, not the level of individual components such as enterprise beans or servlets.
- Performance_debug: Generates the data at Hops level and the first level of the intra-process servlet and Enterprise JavaBeans™ (EJB) call (for example, when an inbound servlet forwards to a servlet and an inbound EJB calls another EJB). Other intra-process calls like naming and service integration bus (SIB) are not enabled at this level.
- Debug: Provides detailed instrumentation data, including response times for all intra-process calls. Note that requests to servlet filters will only be instrumented at this level.

Note: Working with instrumentation and trace levels are further explored later in the chapter (see “Request level details” on page 41).

Important: Request metrics are checked starting with the HTTP plug-in for some Web related settings. The HTTP-plug-in configuration must be regenerated and propagated after enabling request metrics.

Using request metric filters

One final way that can be used to control the request metric instrumentation is to use request metric filters. Filters provide a way to specifically target flows and components to reduce the overhead of broad monitoring and to also make it easier to analyze the captured data by reducing the amount data that is captured.

It is important to understand, however, that filters are implemented as edge component filtering, not as intra-component processing, so an EJB filter will not be effective if the EJB is always invoked from a servlet. In this case it is the URI that needs filtering, not the EJB. Filters should be applied on edge components.

Filters are configured by selecting the **filters** link from the Additional properties section of the **Request metrics** panel. This navigates the administrator to the Request metric filter panel shown in Figure 12.

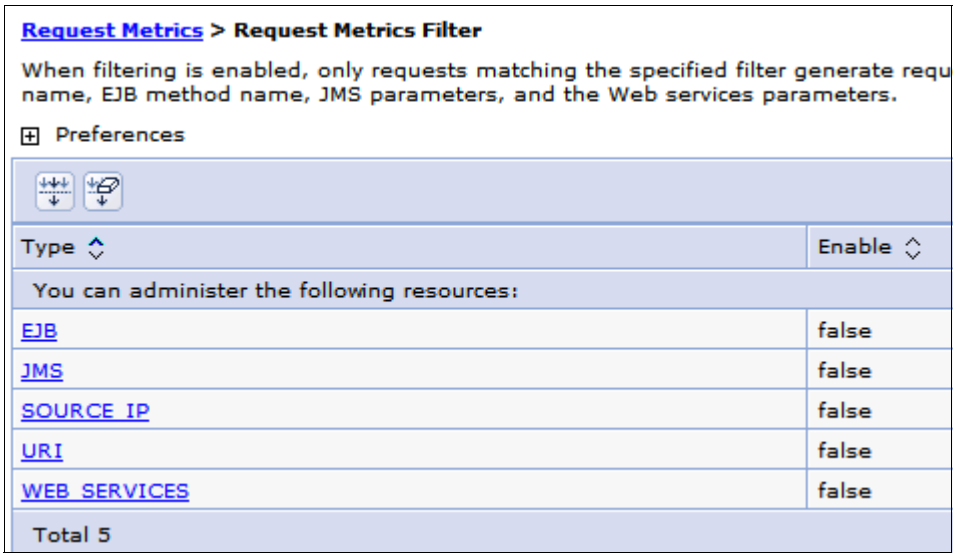


Figure 12 Request Metric Filter panel

Using filters, fine grained controls can be applied to the different edge types of EJB, JMS, IP address, URI, and Web services. The first step is to specify the filter. An example of each type can be seen by navigating the administration console for that type of filter. For example, selecting the URI link in Figure 12 takes you to the URI panel shown in Figure 13.

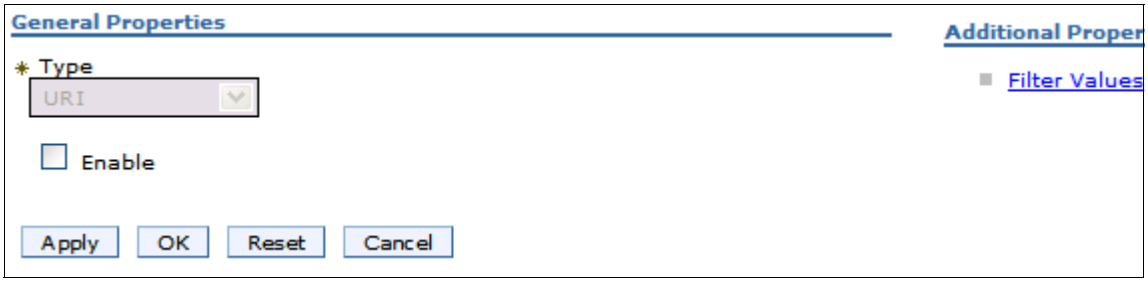


Figure 13 Uri panel

The enable check box must be checked for the filters to be enabled. Then the filters will be used along with the component and trace level settings to determine which components are instrumented.

Note: Enabling filters requires an application server restart.

Select the **Filter Values** link in the filter panel to add or edit filters. This is also where the default example filters are displayed (Figure 14).

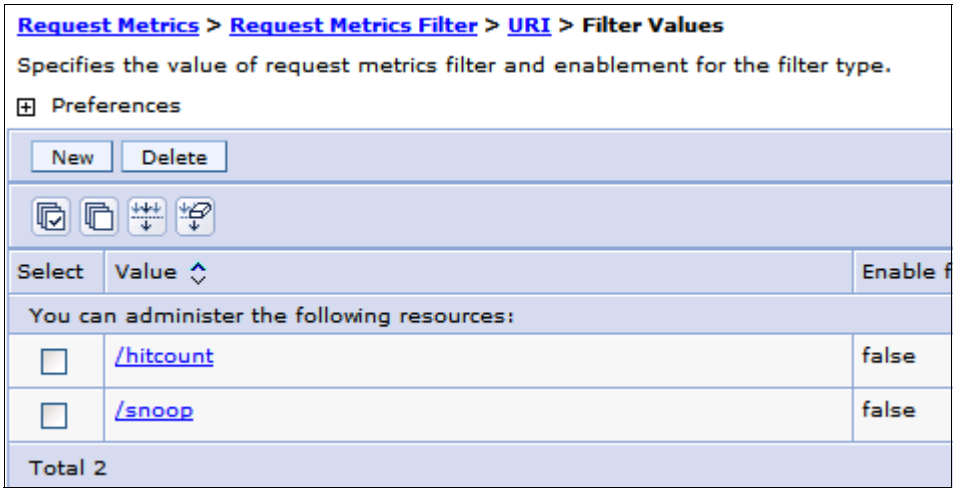


Figure 14 Default filter values displayed in filter value panel

Each filter type has its own syntax that is appropriate for the type. For example, the EJB filter specifies a method class or package that sets the scope of the filtering. Figure 15 shows the example URI filer value supplied for EJB filters.

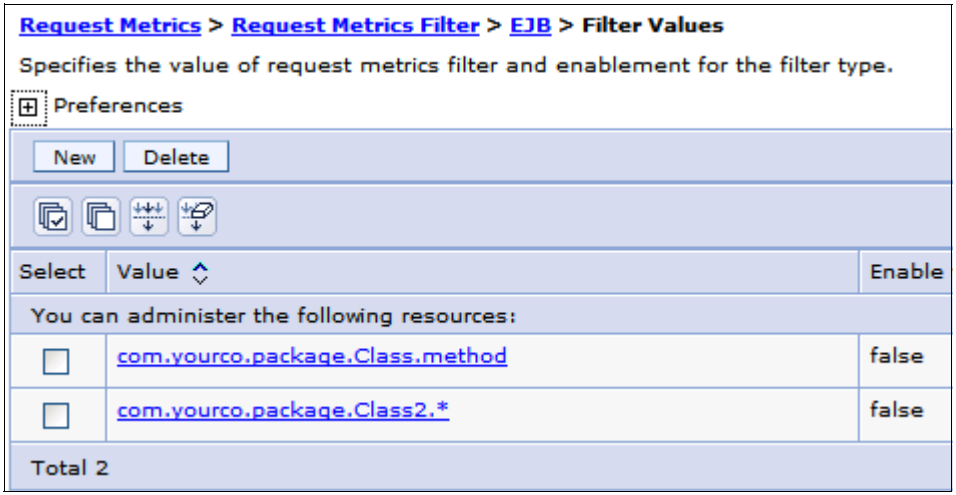


Figure 15 EJB default filter values

It is possible to use wildcard settings in the filters if desired, as shown in the second entry in Figure 15 on page 20.

Tip: To enable/disable a filter it requires the restarting of a server. It is important to plan the component levels and filters that an application might require to minimize the need to stop and restart servers.

Destination type considerations

The final consideration when configuring the request metrics is where the metrics will be gathered. There are two types of supported destinations,

- Data can be logged with standard logs. In this configuration the instrumented components are logged to the SystemOut.log file.
- Data can also be collated in an Application Response Measurement (ARM) data collector. In this case, the data is normally then moved to a monitoring system for analysis and display (for example, using IBM® Tivoli Monitoring Transaction Performance).

Tip: When configuring ARM agents for use with the application server, follow the installation instructions provided with the specific agent. For more information about ARM agents, see:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/cprf_arm.html`

Both logging types can be activated at once. Writing to standard logs is not recommended as a long term monitoring strategy because the overhead can be higher than is desirable.

Overhead of request metrics

The overhead of request metrics can vary significantly based on the components being monitored and the complexity of the request execution within the monitored components. There are no specific metrics on what the overhead is, but it is reasonable to assume that request metrics on every request and component might incur more overhead than is desired. We recommend that an organization consider and plan carefully the interactions that it wants to monitor, then measure the specific overhead associated with configuring request metrics for these components.

Viewing the monitoring data

WebSphere Application Server provides an interface for viewing the monitored data. The interface is the Tivoli Performance Viewer (TPV) found in the administrative console.

Starting TPV monitoring and configuring settings

To work with the TPV from the administrative console, follow these steps:

1. Navigate to the Tivoli Performance Viewer panel, by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity**.
2. Select the check-box of server(s) that are to be monitored and click the **Start Monitoring** button (Figure 16).

Tip: If you are only starting monitoring on a single server, monitoring can be started by simply clicking the server link and navigating into the TPV viewer.

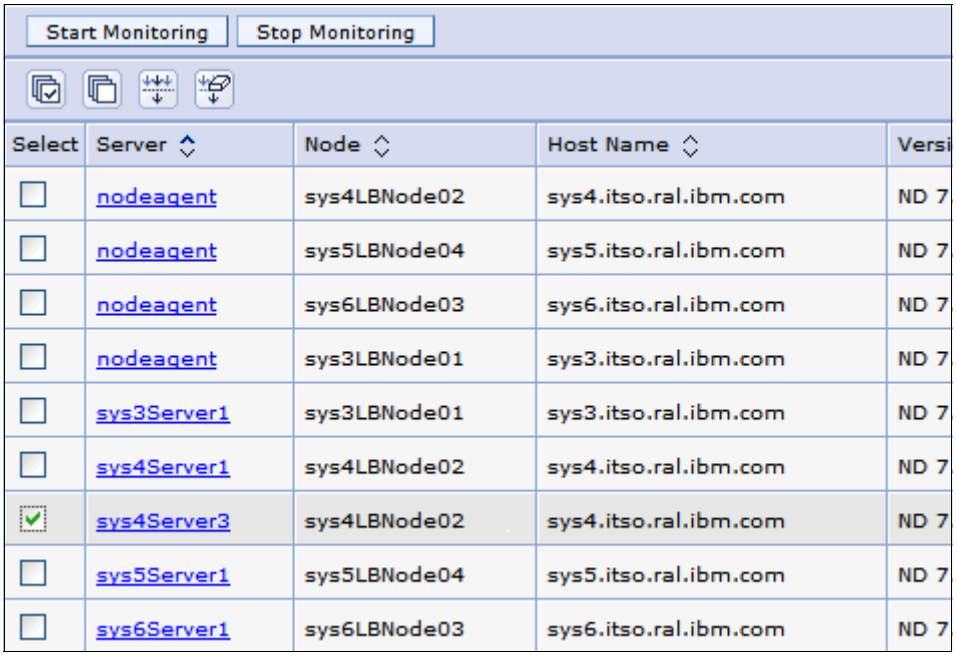


Figure 16 Start Server monitoring

After monitoring is started, a message will be returned in the messages section of the panel, and the Status column of the server will be updated to Monitored.

3. The PMI data can only be observed one server at a time when using a single user session. Select the server name link to navigate to the Tivoli Performance Viewer panel (Figure 17).

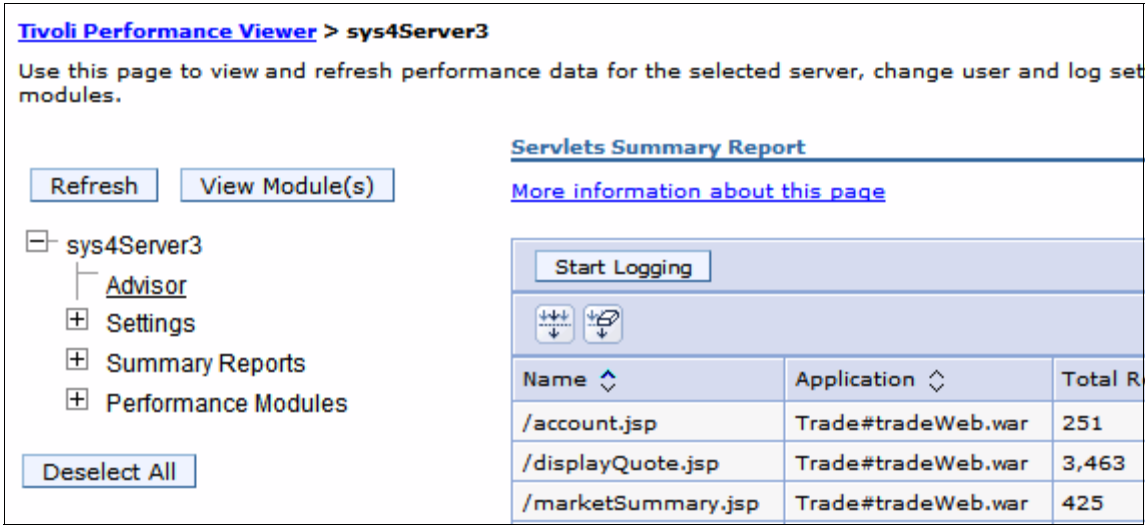


Figure 17 Tivoli Performance Viewer

The default view for this panel shows the Servlet Summary Report panel indicating recent servlet activity, as well as the TPV tree navigation panel.

Note: The different ways that data can be viewed is discussed in “Exploring TPV data views” on page 26. But before exploring the data, let us first take a look at the settings menu to examine the User and Logging settings.

4. The user settings are reached by expanding the Settings category and selecting **User**. This panel helps you control how much data is retained and how often data samples are taken in the live system (Figure 18).

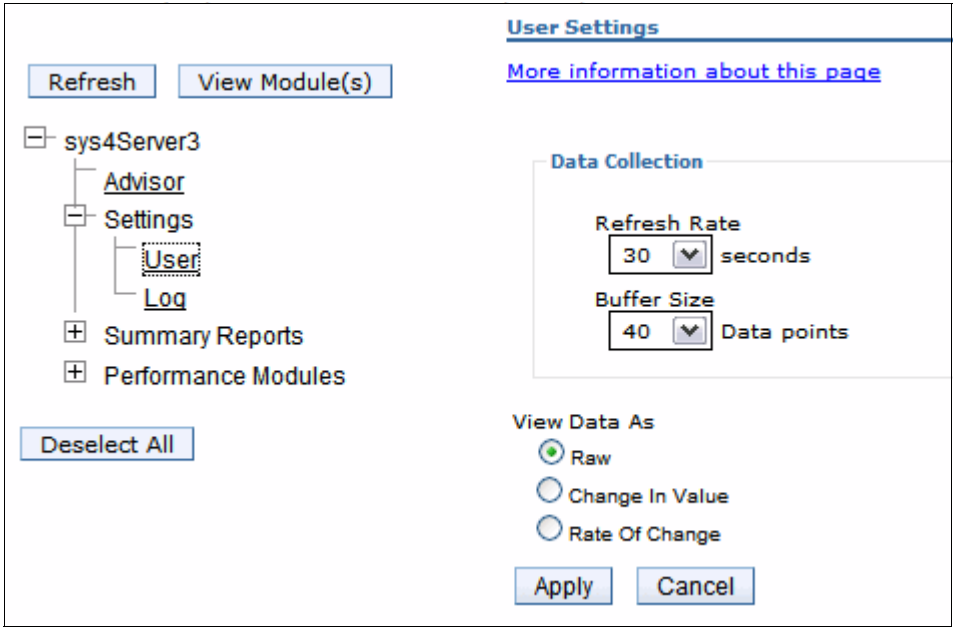


Figure 18 User settings

The user settings are very significant and can have a direct impact on the performance of the server. The two key configuration settings are in the Data Collection section of the panel:

- The refresh rate indicates the interval between data sampling. Higher frequency rates mean that the server will gather and report on statistics more frequently, adding load to the servers that are collecting data.
- The buffer size indicates the number of data points that are kept. More data points simply mean that the PMI data will require more memory.

For a standalone server this means that PMI data at high frequency and high buffer re-initiation will need more processing time and more memory.

In a distributed server model the load is shared, but some settings might still need to be tuned. The data is collected at the node level and stored in memory on the node agent. Thus if a node has many servers, the memory requirements of the node agent will need to be adjusted.

Also in a distributed server configuration, the data is viewed from the deployment manager. Thus, to process the data, the deployment manager should have adequate memory and CPU resources also. Consider also that more than one administrator might be observing data at once.

5. A powerful feature of TPV is the ability to record PMI data and then replay the data later in a different deployment manager as if it were in real time or with options to fast forward and rewind.

   Logging is started by simply clicking the **Start Logging** button shown in the TPV viewing panel (Figure 17 on page 23). However, before clicking this button, the Log settings must be configured. The Log settings are found by selecting **Settings** → **Log** as shown in Figure 19.

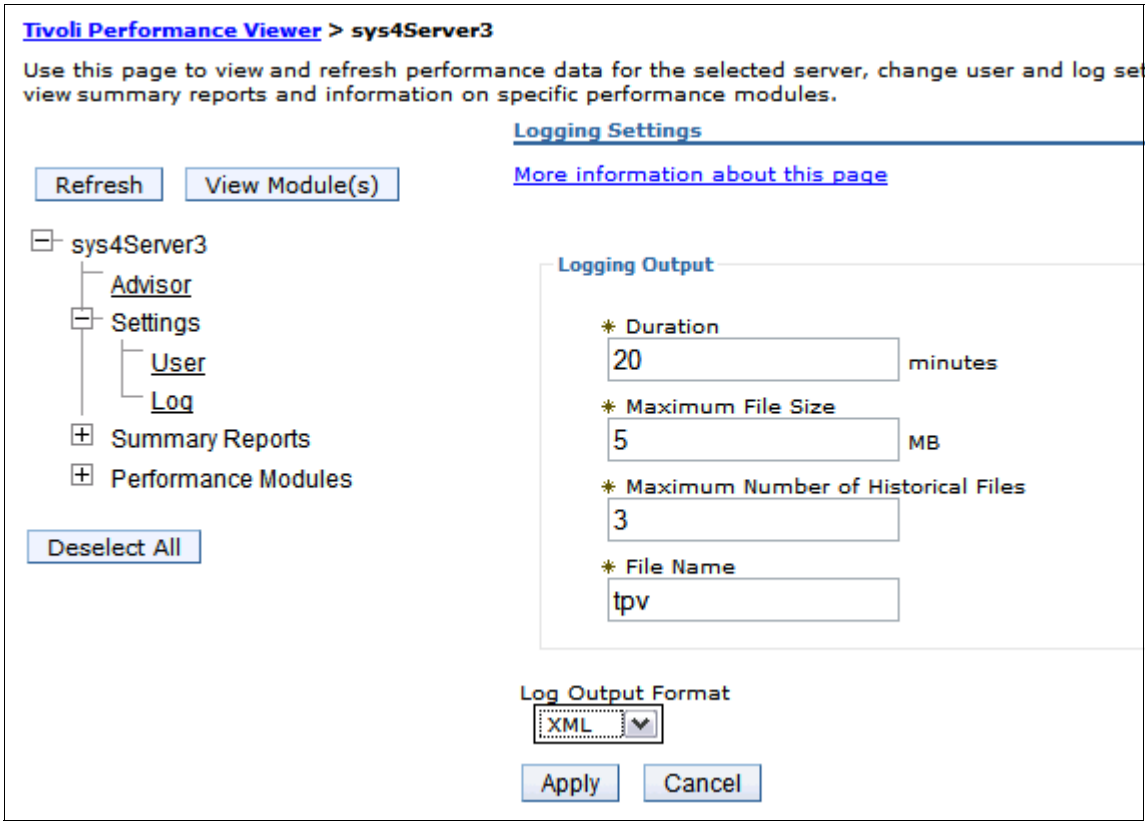


Figure 19 Log Settings

Examining the log settings that are available, the administrator is faced with several configuration choices:

– Duration:

  The logging of PMI data has with it a certain amount of overhead (resulting from logging to a file, the buffering of data in memory, and disk usage for log storage...). It is not intended to be used as a long term production monitoring strategy. Thus when logging is enabled, it is configured to be disabled after a period of time.

PMI data logging used in short durations can be used to capture runtime characteristics that might need further investigation of for sharing with development and troubleshooting specialists.

– Maximum file size and Maximum number of historical files:

  The settings for maximum file size and the number kept that are appropriate for your environment will depend on two conditions:

  - How much PMI data is enabled
  - The data sampling frequency (as configured in the user settings).

– File name:

  The server name and the time at which the log is started is appended to the file name specified to help users identify a log file

– Log output format:

  The other configuration item of consequence is the format type. The default is XML but the binary format requires a smaller footprint on the disk. If logging larger amounts of data, the binary logging format might be more suitable.

Exploring TPV data views

Tivoli Performance Viewer has three primary types of data that can be viewed

- Summary Reports
- Performance Modules
- Advisors

Summary reports

The summary reports provide a general overview in a tabulated format of the current system performance. The reports that are available include:

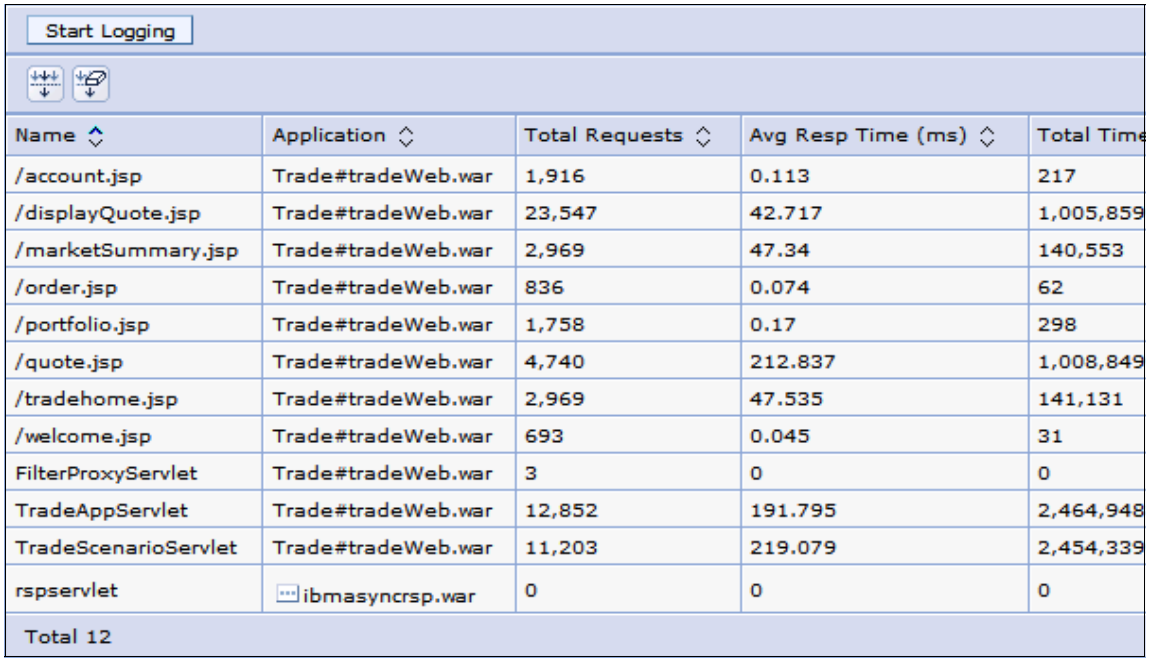
- Servlets
- EJBs
- EJB Methods
- Connection Pool
- Thread Pool

Servlet and EJB summary reports can be useful for identifying the application resources that are most busy, and to the extent that averages can be used, to identify candidates that might warrant further investigation as to their performance. Together with request metrics, results from the summary reports can help identify possible candidates for request metric filters.

The information for Connection Pool and Thread Pool utilization, while useful, can also be easily determined by monitoring the metrics in the performance modules.

Note: For summary reports to be available, the PMI data must be reported at a sufficiently detailed level. For the basic PMI data level, only the Servlets and EJBs reports are available. Higher or custom PMI settings must be specified for the other reports.

Figure 20 shows an example of the Servlets report.

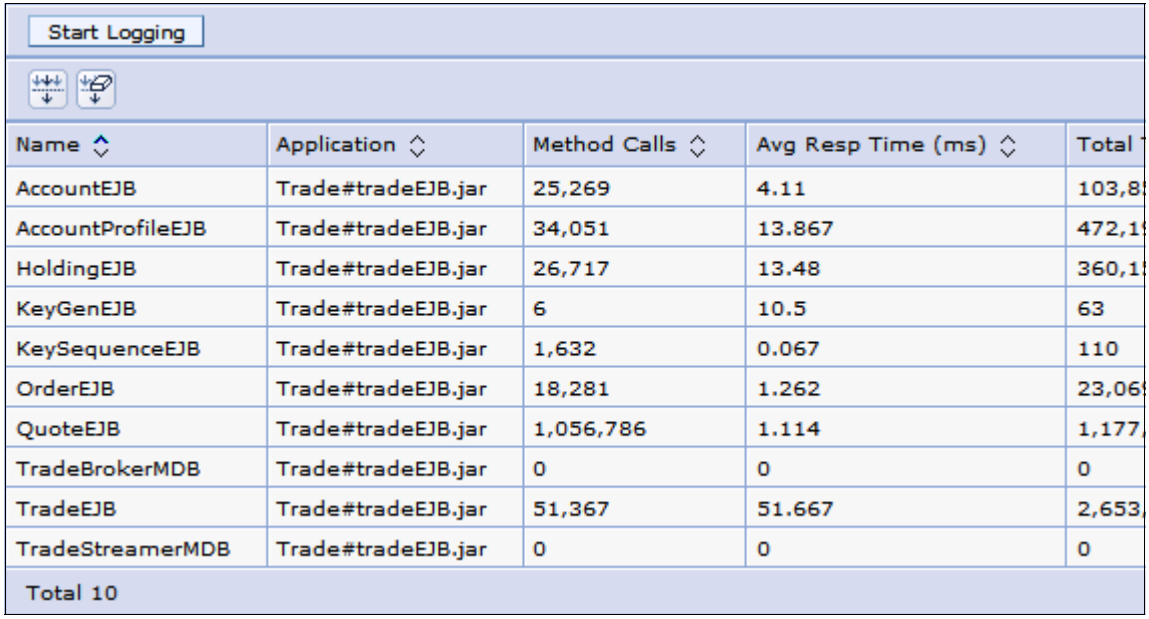

Start Logging

Name	Application	Total Requests	Avg Resp Time (ms)	Total Time
/account.jsp	Trade#tradeWeb.war	1,916	0.113	217
/displayQuote.jsp	Trade#tradeWeb.war	23,547	42.717	1,005,859
/marketSummary.jsp	Trade#tradeWeb.war	2,969	47.34	140,553
/order.jsp	Trade#tradeWeb.war	836	0.074	62
/portfolio.jsp	Trade#tradeWeb.war	1,758	0.17	298
/quote.jsp	Trade#tradeWeb.war	4,740	212.837	1,008,849
/tradehome.jsp	Trade#tradeWeb.war	2,969	47.535	141,131
/welcome.jsp	Trade#tradeWeb.war	693	0.045	31
FilterProxyServlet	Trade#tradeWeb.war	3	0	0
TradeAppServlet	Trade#tradeWeb.war	12,852	191.795	2,464,948
TradeScenarioServlet	Trade#tradeWeb.war	11,203	219.079	2,454,339
rspservlet	ibmasyncrsp.war	0	0	0

Total 12

Figure 20 Servlets summary report

Figure 21 shows an example of the EJBs summary report.


Start Logging

Name	Application	Method Calls	Avg Resp Time (ms)	Total
AccountEJB	Trade#tradeEJB.jar	25,269	4.11	103,8
AccountProfileEJB	Trade#tradeEJB.jar	34,051	13.867	472,1
HoldingEJB	Trade#tradeEJB.jar	26,717	13.48	360,1
KeyGenEJB	Trade#tradeEJB.jar	6	10.5	63
KeySequenceEJB	Trade#tradeEJB.jar	1,632	0.067	110
OrderEJB	Trade#tradeEJB.jar	18,281	1.262	23,06
QuoteEJB	Trade#tradeEJB.jar	1,056,786	1.114	1,177
TradeBrokerMDB	Trade#tradeEJB.jar	0	0	0
TradeEJB	Trade#tradeEJB.jar	51,367	51.667	2,653
TradeStreamerMDB	Trade#tradeEJB.jar	0	0	0

Total 10

Figure 21 EJB summary

Tip: This summary data can be used to identify EJBs that get invoked very often. If the system is working correctly, the slowest bean on average might be worth investigating if the time exceeds expected SLAs, and so on.

It can also be useful to identify what work is not happening. In the preceding snapshot, there are two message-driven beans that have not been invoked. This, in itself, might be unusual and warrant investigation.

Performance modules

Performance modules provide a tracking mechanism for each of the PMI counters that are active. These counters are categorized under their different PMI data classifications. The data can be viewed as a table or graphically displayed using the embedded Adobe® SVG viewer in a graphical display.

The performance modules provide a powerful runtime view of the data as it is being recorded to allow the administrator to analyze the current system health.

The data that can be displayed is limited depending on the PMI level that has been configured. The administrator can select one or more metrics for the current PMI level as shown in Figure 22, and then select the **View Modules** button at the top of the panel.

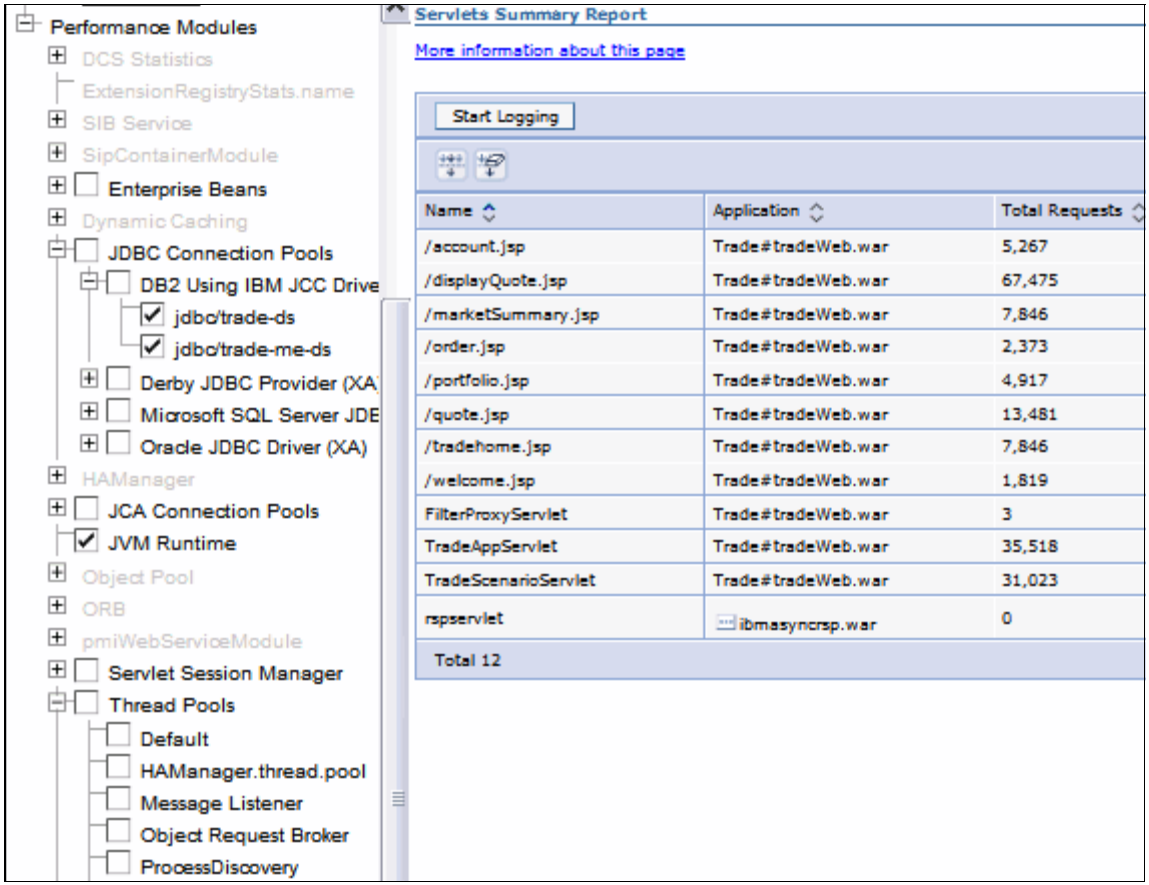


Figure 22 Performance modules

After the performance modules are selected, the data panel provides a view of the data that is being collected. By default, the graphical view is used. Each most recent data point and the graph key for the different counters can be seen under the graph. The graph can then be customized to include or exclude counters from the chosen set of PMI data. Figure 23 shows an example of the PMI data displayed in the example environment.

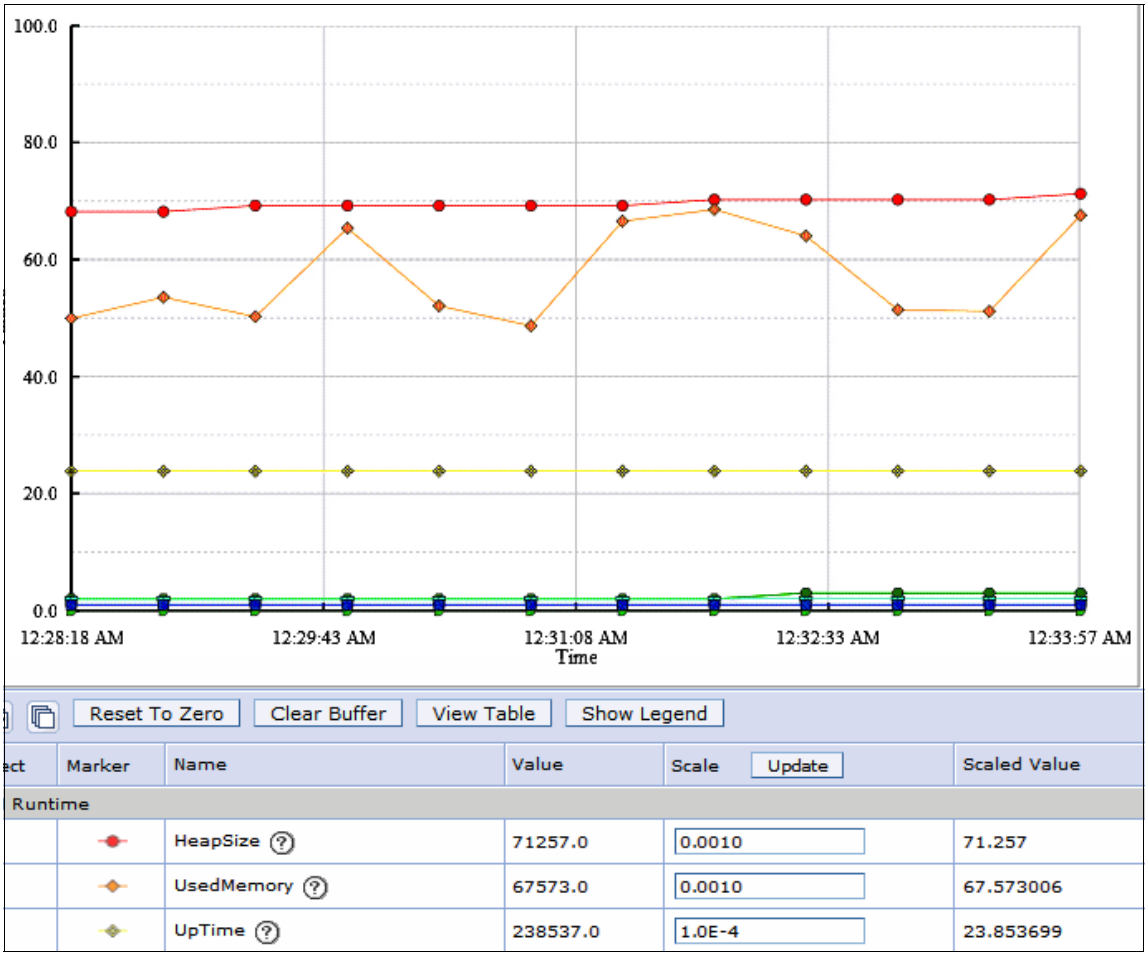


Figure 23 View modules, graphical data view

To view the data in table format, click the **View table** button. Scales can also be adjusted by changing the scale and clicking the **Update** button.

Performance advisors

The last of the TPV data sets contains the TPV performance advisors. These advisors analyze the data using rules that are pre-configured by IBM based on recommended practice and performance observations. The advisors provide tuning recommendations to help improve the performance.

The types of items that TPV will provide advice on include several well known performance hot spots:

- 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

Advisors are more of a tuning aid than a monitoring tool set and are not suitable for use in production environments. But advisors can be a useful aid in identifying well known performance hotspots in the current server configurations in testing.

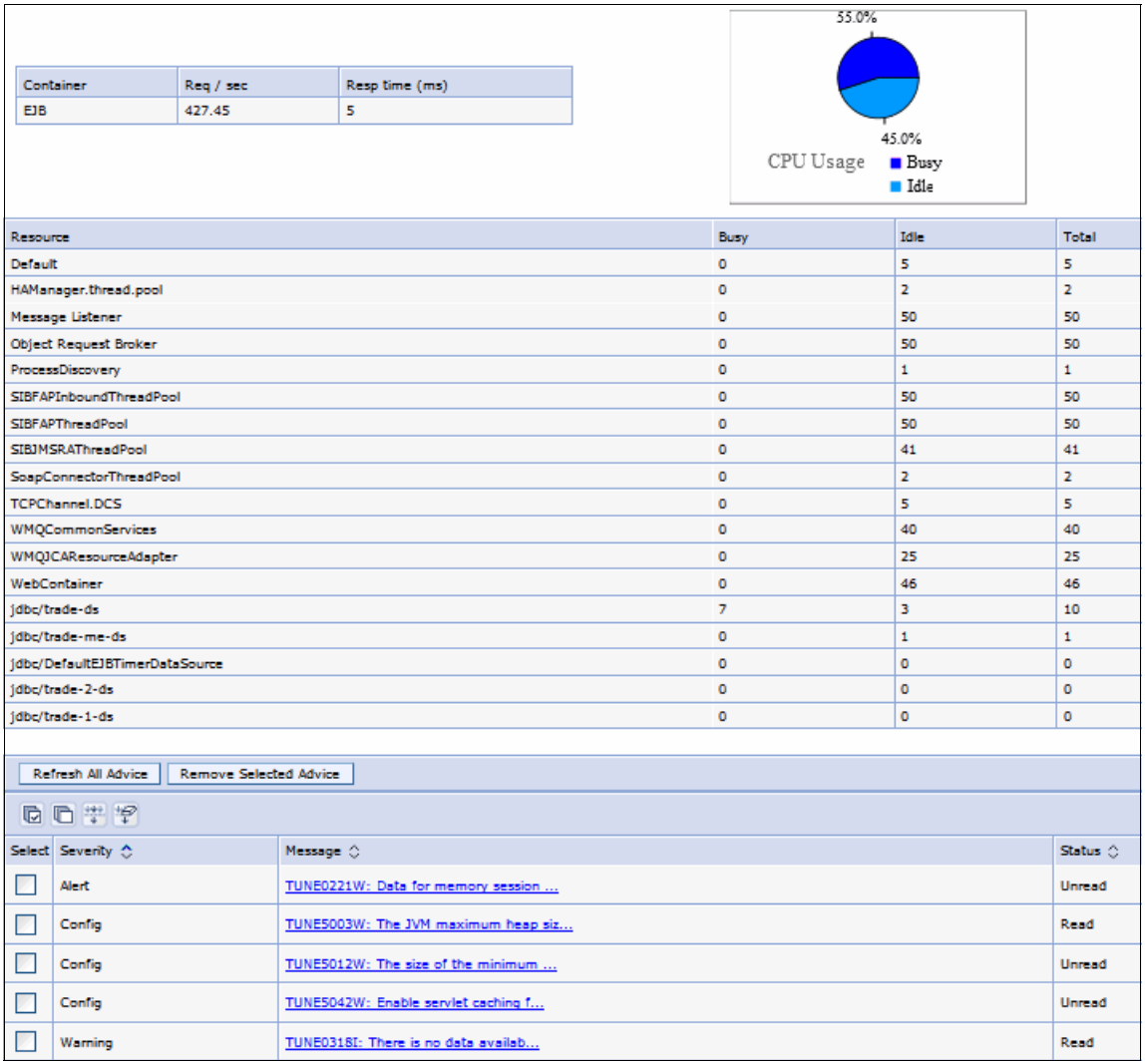
Advisors are best used when:

- A reasonable load can be driven to the application server utilizing significant CPU (50+%).
- You want help in tuning a server while establishing initial performance benchmarks

For more information about the advisors, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/cprf_whyuseperfadvisors.html

To view the Advisors panel, click the **Advisor** link in the TPV menu panel. Figure 24 shows an example of what you will see.



Container	Req / sec	Resp time (ms)
EJB	427.45	5

Resource	Busy	Idle	Total
Default	0	5	5
HAManager.thread.pool	0	2	2
Message Listener	0	50	50
Object Request Broker	0	50	50
ProcessDiscovery	0	1	1
SIBFAPInboundThreadPool	0	50	50
SIBFAPThreadPool	0	50	50
SIBJMSRAThreadPool	0	41	41
SoapConnectorThreadPool	0	2	2
TCPChannel.DCS	0	5	5
WMQCommonServices	0	40	40
WMQJCAResourceAdapter	0	25	25
WebContainer	0	46	46
jdbc/trade-ds	7	3	10
jdbc/trade-me-ds	0	1	1
jdbc/DefaultEJBTimerDataSource	0	0	0
jdbc/trade-2-ds	0	0	0
jdbc/trade-1-ds	0	0	0

Select	Severity	Message	Status
☐	Alert	TUNE0221W: Data for memory session ...	Unread
☐	Config	TUNE5003W: The JVM maximum heap siz...	Read
☐	Config	TUNE5012W: The size of the minimum ...	Unread
☐	Config	TUNE5042W: Enable servlet caching f...	Unread
☐	Warning	TUNE0318I: There is no data availab...	Read

Figure 24 Advisors view

From within the advisors view, the different advice statements can be selected and further analyzed by the administrator. Figure 25 shows an example of an advice panel.

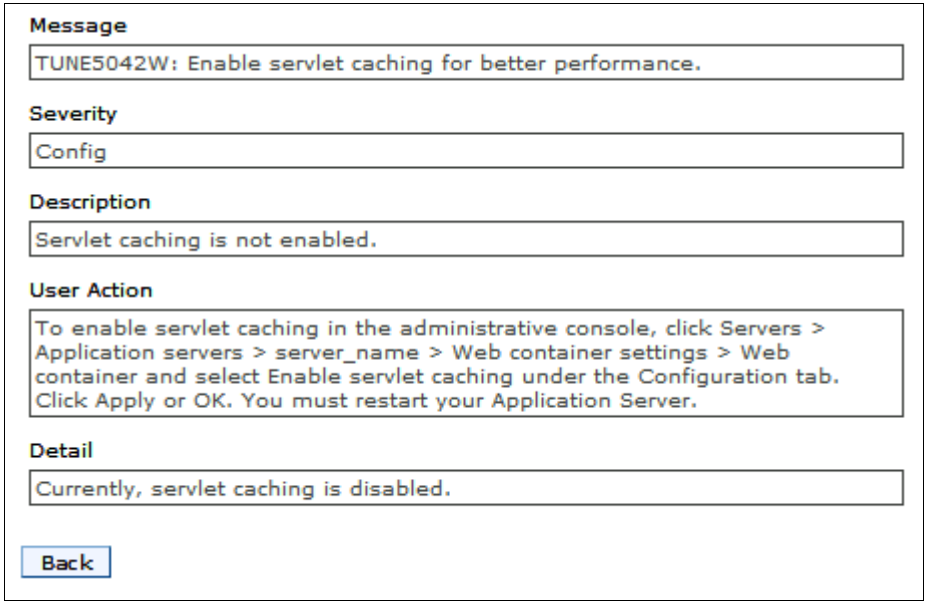


Figure 25 Advice panel

The Advice panel provides a clear description and recommendation to aid the administrator in quickly addressing well understood performance considerations.

Note: Advisors are not further used in this chapter. Our focus in this chapter is monitoring, and advisors really fall into a tuning toolset more than monitoring.

Monitoring scenarios

In this section, a variety of monitoring data will be used to provide examples and information about what that data tells an administrator.

Important: While the statistical data that is observed in PMI and request metrics provides a powerful understanding of what is occurring in the environment, it is very important to remember that the counters and statistics are mostly averages. You must always verify that the statistics make sense for your environment. For example, if an EJB seems to be having good response time, but it is throwing and catching an exception shortly after entering the bean, it would still appear to be having excellent response time. Response time alone is not enough to indicate a healthy system.

Database interactions

One of the hotspots for monitoring is interactions with external servers, especially interactions with databases. PMI provides a good set of metrics at even the basic level for these types of interactions.

Consider the data snapshot shown in Figure 26. This snapshot was taken in the example environment while the server was under load. The report is viewed by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **Performance Modules** → **JDBC™ Connection Pools**. The monitoring level in effect was the Basic level.

Select	Marker	Name	Value	Scale Update
jdbc/trade-ds				
☑		CreateCount	0.0	1.0
☑		CloseCount	0.0	1.0
☑		PoolSize	10.0	1.0
☐		FreePoolSize	1.0	1.0
☐		WaitingThreadCount	4.0	1.0
☐		PercentUsed	90.0	1.0
☐		UseTime	39.785263	1.0E20
☐		WaitTime	124.26131	1.0E20

Figure 26 Basic PMI for database

From a monitoring perspective, this provides several very useful statistics that can help the administrator understand if the database interaction is healthy. For measuring runtime health, the following counters are very beneficial:

- PercentUsed indicates if the database connections are being over utilized or under utilized.

  Depending on the application and capacity of the database, it is typically not a good sign if the database is 100% utilized. If a connection pool becomes 100% utilized all of the time after the system has been tuned, either the database might need more capacity to support more connections, or some type of error is occurring, for example, the application might have a connection leak. Utilization is a good indicator that database connections need some attention.

- Similarly, it is not unreasonable to have waiting threads on the data source. After all the resource is shared. From a monitoring perspective it is a combination of the waiting thread count and the wait time that make an interesting combination.

  If the wait time and waiting thread count grow over time, this is an indication that the database might be responding more slowly than desired, or there are insufficient connections available to support the load. How long is too long a wait time depends on application service level agreements and whether wait times occur all the time or only on occasion under exceptional load.

- UseTime can help understand how much time is spent communicating with the database and can help to indicate if database response is degrading.

If the administrator believes there are response time errors with the database, request metrics can be useful in diagnosing which components the application is spending it time.

JCA interactions

JCA is the more generic form of the JDBC, and is used with standard adapters that comply with the JCA standard. As part of this, the adapter can have connection pooling the same as used with JDBC. Probably the most common JCA adapter other than JDBC is that of JMS connections, whether connecting to the service integration bus or connecting to an external JMS provider such as WebSphere MQ.

Hence, when working with JCA or JMS connections, the same indicators as used for the JDBC and as discussed in “Database interactions” on page 34 are relevant, with the exception of counters that are JDBC specific, of course. When monitoring, it is beneficial in particular to monitor the waiting threads and their wait time, and the current pool size.

Threading resources

Assuming that there is enough CPU and memory, while simplistic, it is reasonable to surmise that thread pools along with resource connections are a major factor in understanding the limits of throughput on the application server. The various thread pools in the application server control the entry points for requests into the system. If the pool is exhausted, then requests to the system are queued and have to wait. From a monitoring perspective, it is preferred that thread pools are not constantly exhausted and running at their maximum.

But before monitoring thread pools, the administrator needs to first be aware of what types of thread pools their application uses. The following application topologies provide some insight into what needs to be considered in understanding which thread pools might be important to application runtime throughput.

In the first example, consider a clustered application server environment where the deployed application has both Web and EJB components deployed together in the JVM (Figure 27). For performance reasons, WebSphere Application Server will use process affinity when invoking the EJB's, sending requests from the Web container to the EJB container in the same JVM.

In this scenario, threads are not swapped and the requests are executed on Web container threads. Even though EJBs are extensively used in the application, the ORB thread pool would not be used in this application topology. Web container threads in this topology control the concurrency of the application.This is the topology used by the sample application and is common to many JEE applications.

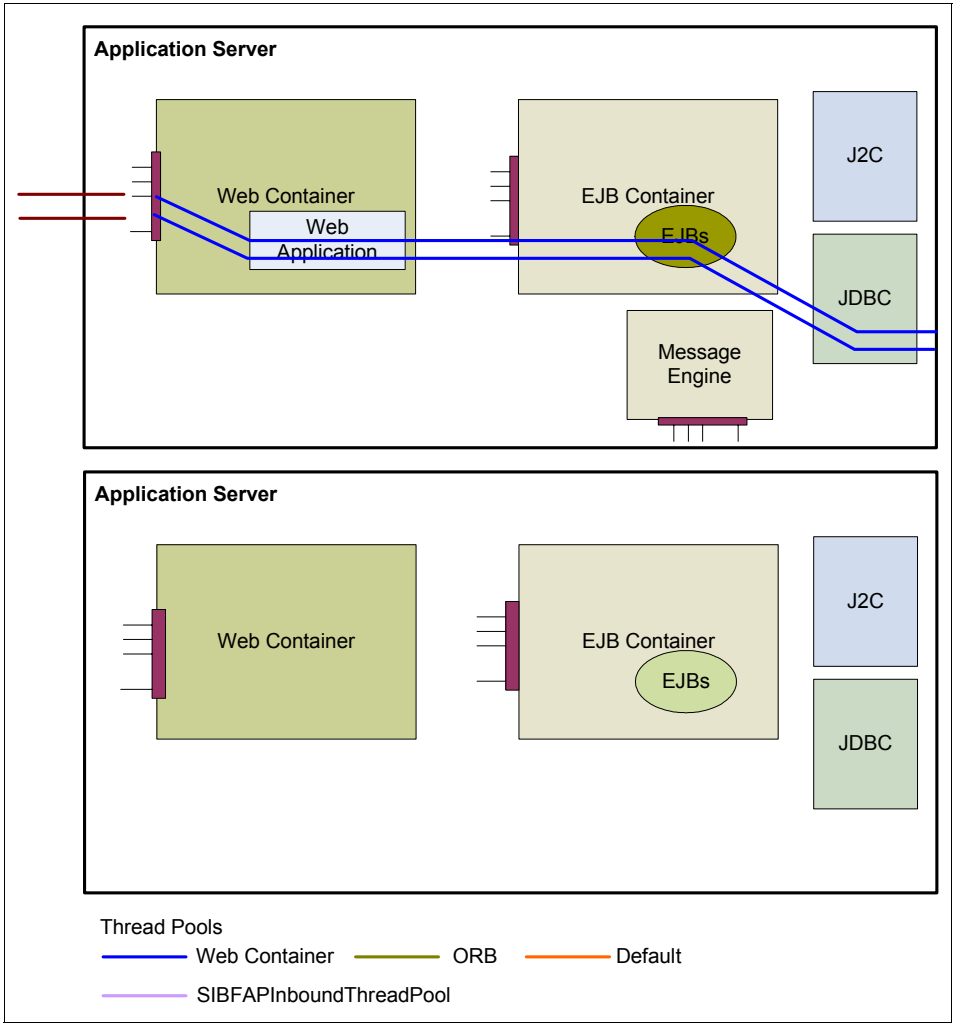


Figure 27 Web application

But what if the EJB components were deployed in a separate JVM (Figure 28)?

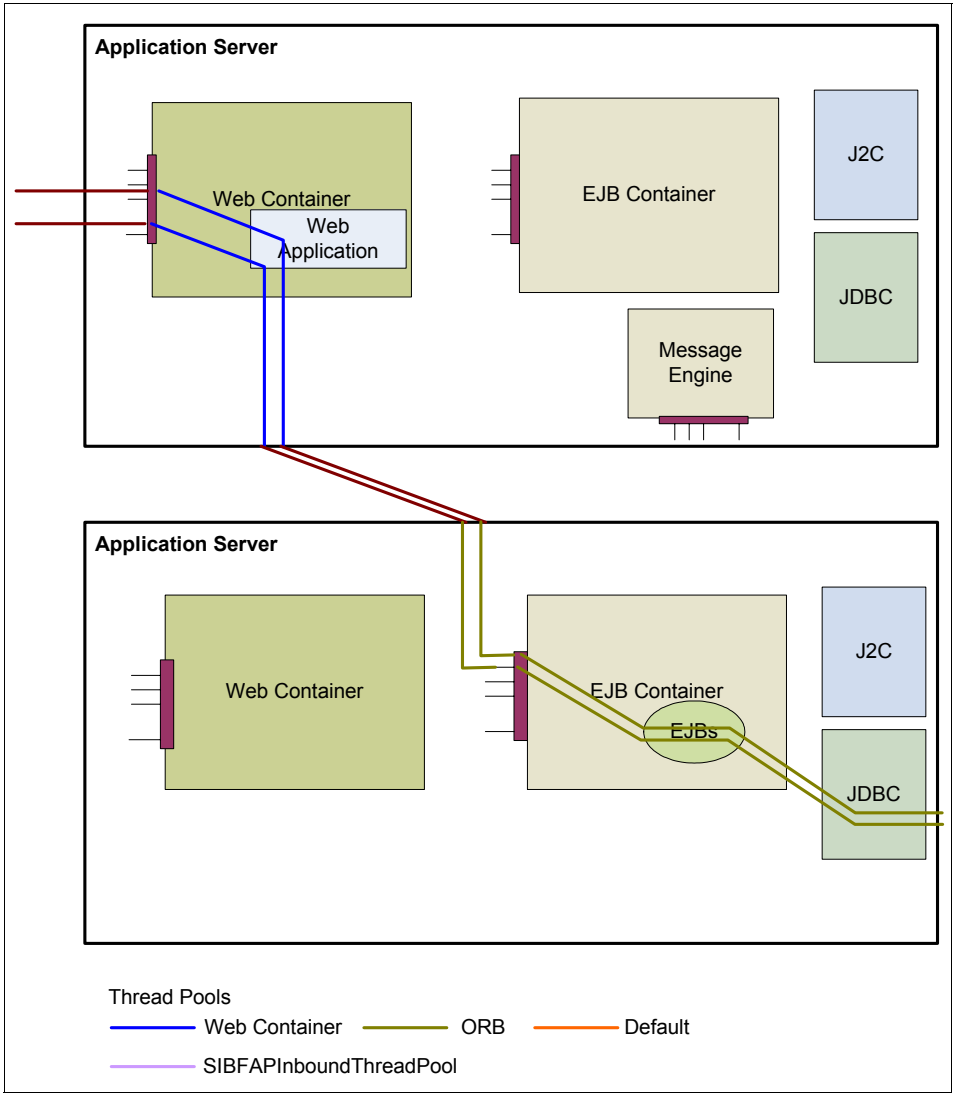


Figure 28 Distributed application model

In such a scenario, the ORB thread pool in each application server would become equally important as the Web container thread pool. Request throughput would now be controlled by the Web container, the ORB thread pool and other parameters such as memory and connections to external resources.

Different environment resources use different thread pools and a key consideration to understand as an administrator is what are the components that the deployed applications will interact with, and what is the likely environmental impact.

Tivoli performance viewer provides a number of resources that can assist with understanding ThreadPool utilization, and we recommend consulting the Thread Counters to learn more.

Figure 29 shows a graphic where the red line represents the Web container active thread pool count available when PMI is activated at the basic level. The report is viewed by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **Performance Modules**. Expand the **Thread Pools** section and select **Web container**.

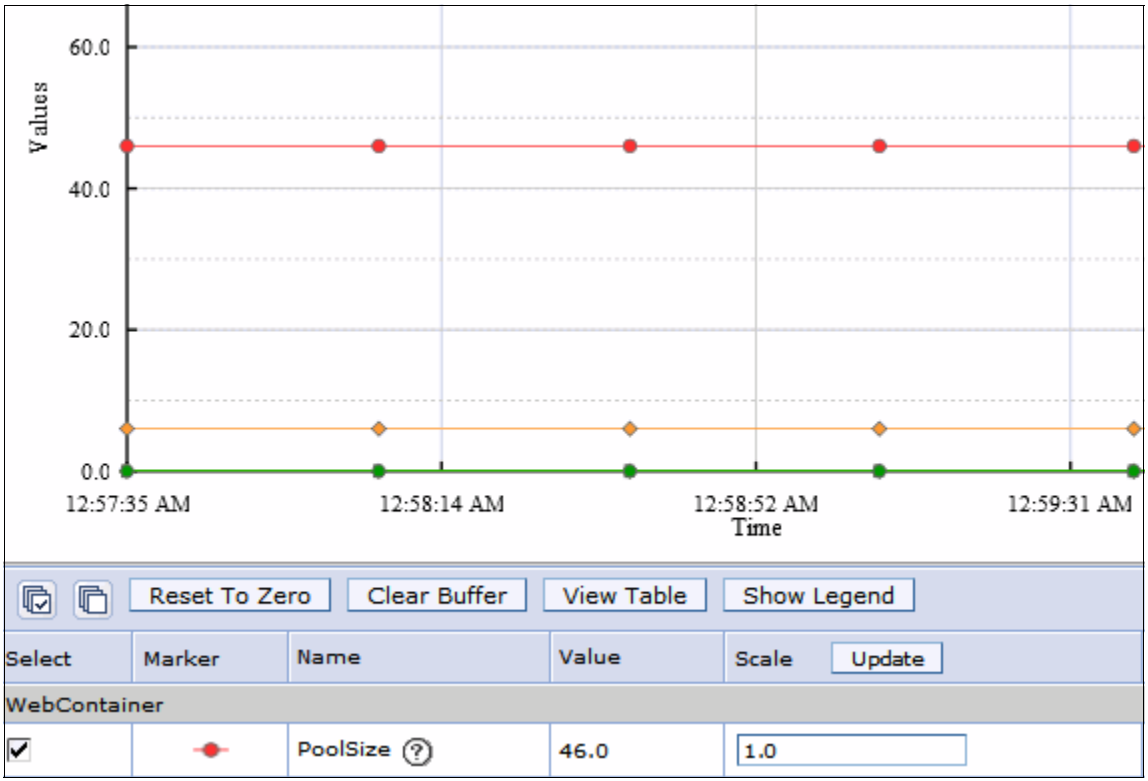


Figure 29 Thread Pool counters

JVM memory usage

Environmentally, the way the applications and application servers utilize memory is very important to the overall server performance. As a performance hotspot, it should be no surprise that the application server PMI data provides some basic level monitoring capabilities.

The JVM Used memory counter can be monitored and displayed in the TPV graph. Figure 30 shows an example of this data.

The report can be viewed by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **Performance Modules**, then selecting **JVM Runtime.**

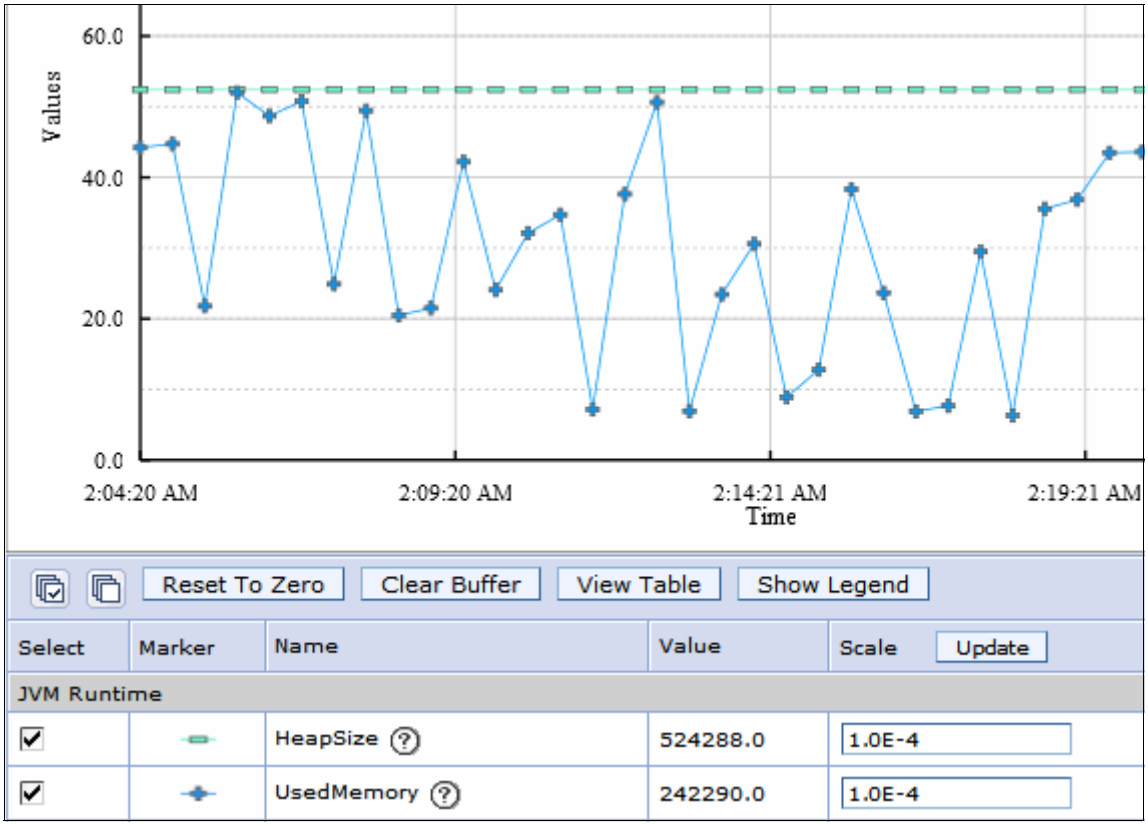


Figure 30 JVM memory usage

This graphical view is a nice way to visually check that memory is not constantly growing and that over time, garbage collection stabilizes it. It also gives a good indication of how frequently garbage collection might be occurring.

However, when tuning memory sizes, we recommend that verbose garbage collection and a combination of the application server support tools available in IBM Support Assistant are utilized for the analysis of the memory usage.

Note: TPV provides a good representation of what is occurring with the JVM memory, but the IBM Garbage Collection and Memory Visualizer tool and the IBM Pattern Modelling and Analysis for Java Garbage Collector tool provide more insightful help on memory related configurations to assist tuning memory. For example, you can be provided with advice on heap sizes and garbage collection algorithms based on the patterns observed in the garbage collection log.

These tools are available as add-ons to the IBM Support Assistant. Use the IBM support assistant setup wizards to download these and other Support tools. For more information, see:

- IBM Support Assistant, at:

  http://www-01.ibm.com/software/support/isa/

- IBM Education Assistant module for The IBM Garbage Collection and Memory Visualizer, at:

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v7/was/7.0/ProblemDetermination/WASv7_GCMVOverview/player.html

Request level details

When examining performance data, especially when tuning a server, it can be useful to have more detailed data. This is particularly true when first trying to pin down and explore bottlenecks. Request metrics can provide this lower level information, showing where time is spent in a request.

Manually creating an application server perceived response time graph (Figure 31 on page 42) can help illustrate the response time of each component as the request is processed. The request metrics at the outer most or edge component are taken as well as all inner components, as each one handles the request. Each new component represents a narrower view excluding the work done by the components that proceed it.

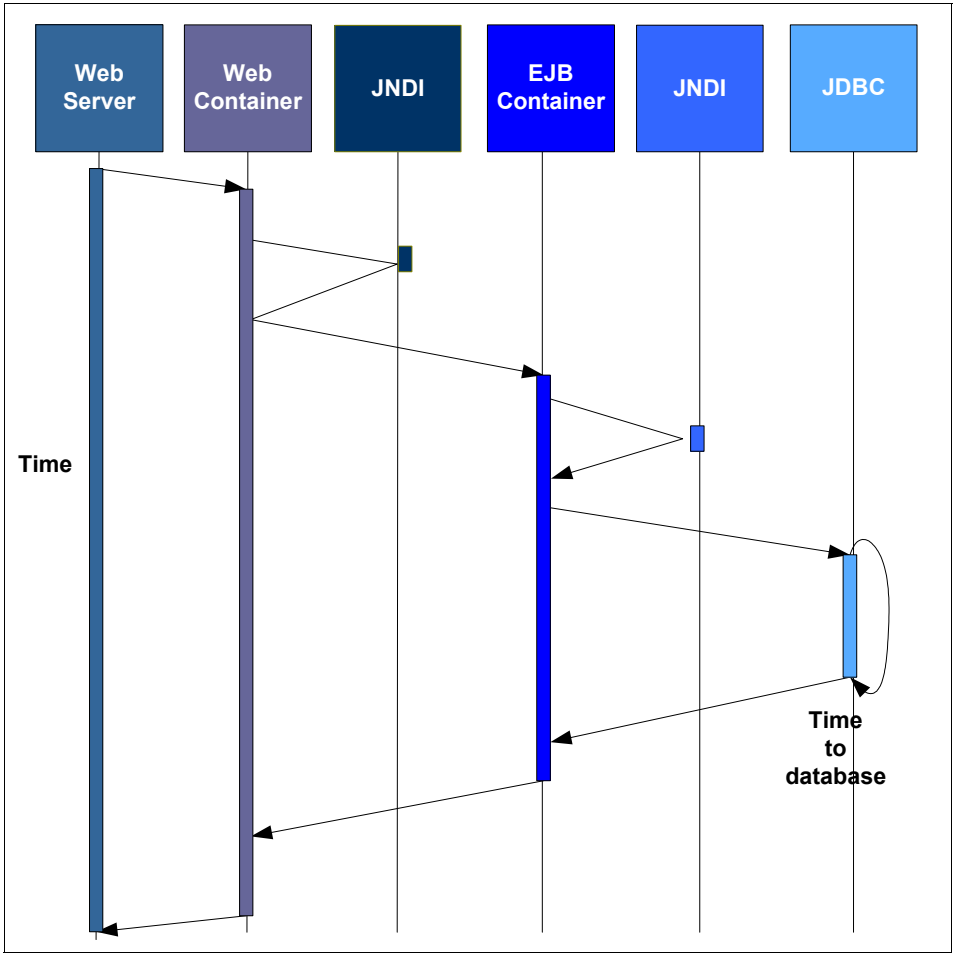


Figure 31 Time view of request metrics

Tip: The metrics, when printed to the standard logs, are printed from right to left with the innermost statistic reported first.

Request metrics are ideal for helping to identify the slowest and worst performing request types and can assist in identifying areas where performance can be improved. As a second phase, data from more detailed monitoring for the request type can be broken down to provide insight on where the bulk of time is being spent and provide insight into what areas of the code warrant investigation.

Unlike PMI, there is no built in tool for request metric analysis, thus for these examples, the data will be extracted from the logs. IBM has tooling available in the Tivoli monitoring suites but it is not part of the application server offering.

Example 1 shows the request metrics captured for a single request in the standard log file. In this example, request metrics are enabled with a trace level of hops and all components instrumented. As you can see, it is quite verbose even with a small amount of detail active.

If you do not have tools, a simple editor with good search and parsing capabilities can be used to filter the request based on **reqid** or other fields.

Example 1 Simplified PMI data example

```
[6/2/09 2:37:12:187 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818051,event=1 type=JDBC
detail=javax.resource.spi.ManagedConnectionFactory.matchManagedConnections(Set, Subject,
ConnectionRequestInfo) elapsed=0
[6/2/09 2:37:12:187 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818052,event=1 type=JDBC
detail=javax.resource.spi.ManagedConnection.getConnection(Subject, ConnectionRequestInfo)
elapsed=0
[6/2/09 2:37:12:187 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818053,event=1 type=JDBC
detail=javax.resource.spi.XAResource.start(Xid, int) elapsed=0
[6/2/09 2:37:12:203 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818054,event=1 type=JDBC
detail=java.sql.PreparedStatement.executeQuery() elapsed=16
...
Lots of lines deleted for simplification
...
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818089,event=1 type=JDBC
detail=java.sql.PreparedStatement.executeQuery() elapsed=16
```

```
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818090,event=1 type=JDBC
detail=javax.resource.spi.XAResource.end(Xid, int) elapsed=0
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818091,event=1 type=JDBC
detail=javax.resource.spi.XAResource.commit(Xid, boolean) elapsed=0
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818092,event=1 type=JDBC
detail=javax.resource.spi.ManagedConnection.cleanup() elapsed=0
[6/2/09 2:37:12:234 EDT] 00000703 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 -
current:ver=1,ip=9.42.171.22,time=1243867419390,pid=6536,reqid=2818050,event=1 type=URI
detail=/trade/scenario elapsed=47
```

The timing data can manually be plotted as illustrated in Figure 32. Graphing complex applications might be an easy way to visualize where time is being spent. Plotting data manually can take some time. Interaction diagrams can be substituted and the timing plotted on the diagrams.

In this example, the request takes 47 milliseconds (not a long time). But it can be observed that the time for the prepared statement cache used about 70% of this time. Of note in this configuration is that only the edge metrics are captured. The edge being the JDBC data and the original servlet URI request.

Whatever representation is preferred the key understanding that is required is that request metrics provide a useful mechanisms for the analysis of where time is spent in requests that are being executed.

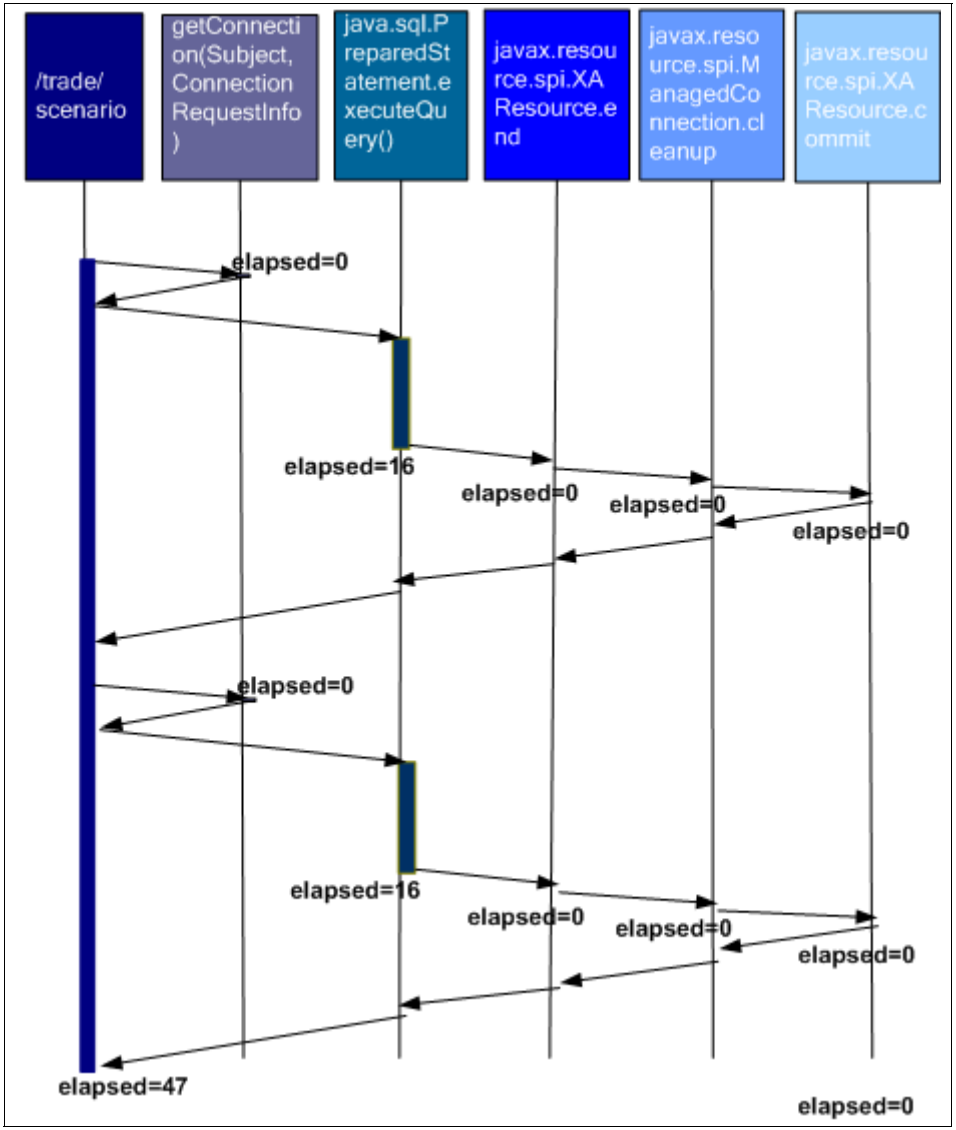


Figure 32 Hops time graph for trade application

As more detail is added, the picture becomes more complete. Instead of a trace level of hops, you can use trace level debug to gather more detail (Figure 33).

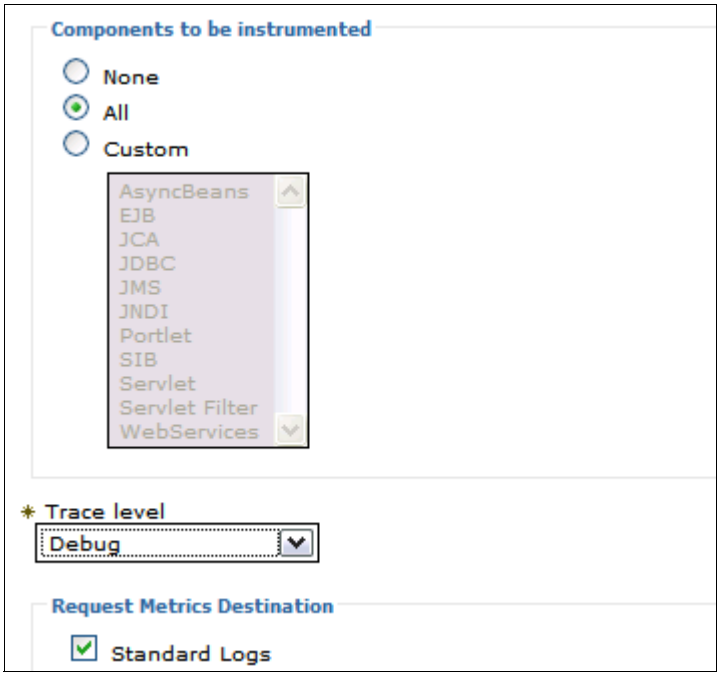


Figure 33 Request metrics configuration panel

Example 2 shows metrics captured at a trace level of debug.

Example 2 Additional component types now recorded

```
[6/2/09 3:43:59:312 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=645,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.findByPrimaryKeyForUpdate
elapsed=0
[6/2/09 3:43:59:312 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=650,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.getPrice elapsed=0
[6/2/09 3:43:59:312 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=651,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.getPrice elapsed=0
[6/2/09 3:43:59:328 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
```

```
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=652,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.QuoteBean.updatePrice elapsed=16
...
Lines deleted
...
[6/2/09 3:43:59:484 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=674,event=1 type=JDBC
detail=javax.resource.spi.XAResource.commit(Xid, boolean) elapsed=0
[6/2/09 3:43:59:484 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=675,event=1 type=JDBC
detail=javax.resource.spi.ManagedConnection.cleanup() elapsed=0
[6/2/09 3:43:59:484 EDT] 00000033 PmiRmArmWrapp I   PMRM0003I:
parent:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=567,event=1 -
current:ver=1,ip=9.42.171.22,time=1243928581406,pid=7904,reqid=644,event=1 type=EJB
detail=com.ibm.websphere.samples.trade.ejb.TradeBean.updateQuotePriceVolume
elapsed=172
```

For more information about these techniques, see:

- Why use request metrics? at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprf_requestmetrics.html

- Example, at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rprf_example.html

ITCAM for WebSphere

IBM Tivoli Composite Application Manager (ITCAM) for WebSphere is shipped with WebSphere Application V7. After being installed, it is embedded in the application server. ITCAM for WebSphere provides a subset of the ITCAM for Web resources. The following sections show how to configure and use ITCAM for WebSphere.

Installing the data collector

The first step in enabling ITCAM is to install and configure a data collector for the monitored servers. ITCAM for WebSphere can be installed from the WebSphere Application Server V7 installation launchpad.

For information about the installation, refer to the product installation documentation:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.itcamwas_wr.doc_6.2/welcome.html

Configuring ITCAM for WebSphere metrics

The last step of the installation process contains a check box to start the configuration tool that configures the servers for data collection. The configuration tool can be started immediately after installation using this option, or you can start it after installation.

The tool is located in *itcam_install*/config_dc/config_dc.

The TPV settings for the servers can also be adjusted to include ITCAM data during the configuration.

Follow the instructions in the data configuration wizard:

1. Click **Next** on the data collector configuration panel (Figure 34).

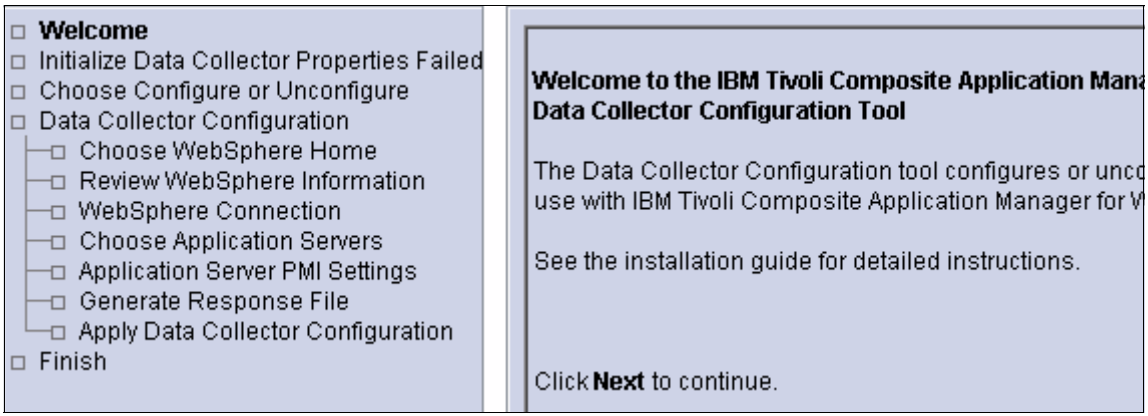


Figure 34 Collector configuration wizard

2. Select **Configure application servers for data collection** (Figure 35) and click **Next**.

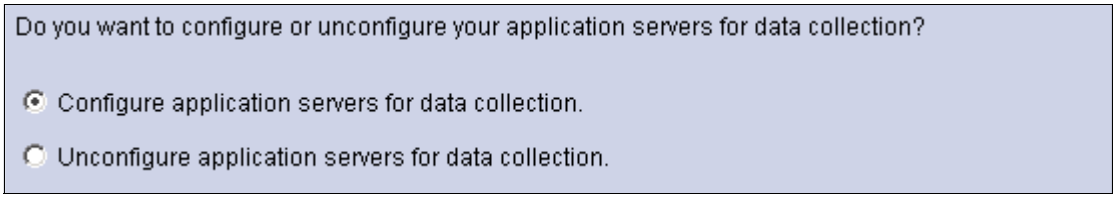


Figure 35 Configure application servers for data collection

3. Select the node that the collector is being configured for and click **Next** (Figure 36).

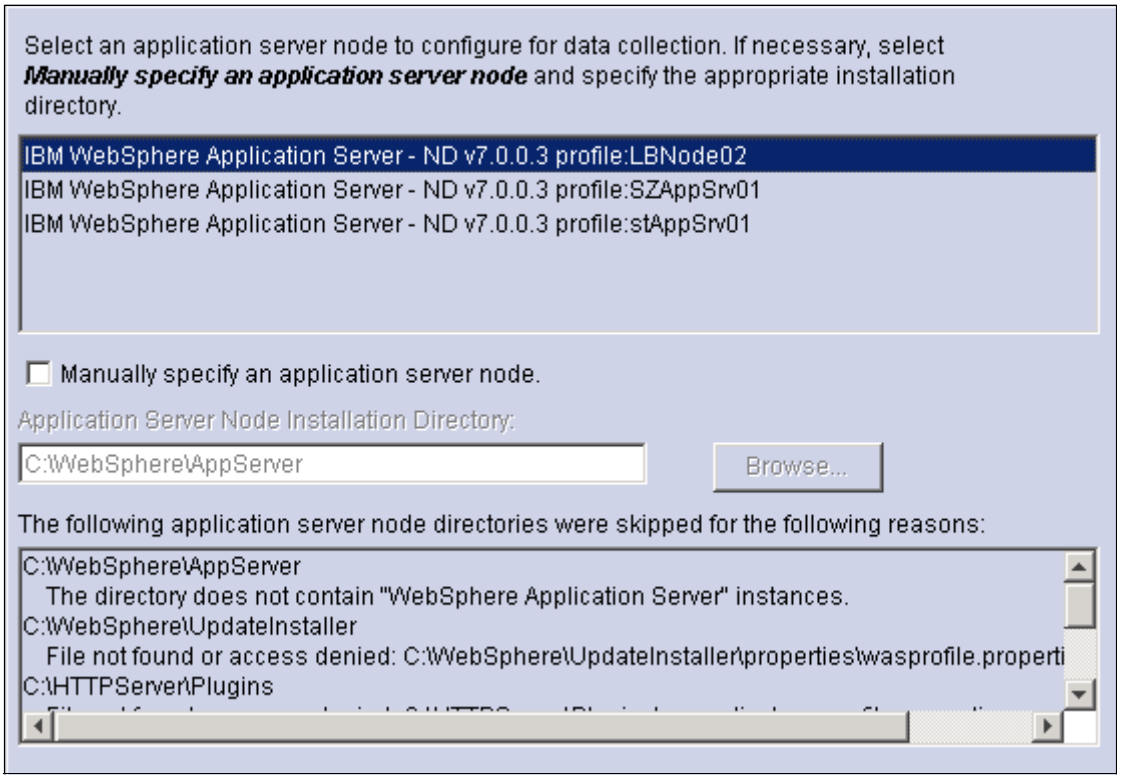


Figure 36 Data collector configuration

4. Click **Next** on the directory home panel.
5. Enter the administration port details (Figure 37 on page 50). For Network Deployment, this needs to be the information for the deployment manager configuration. Click **Next.**

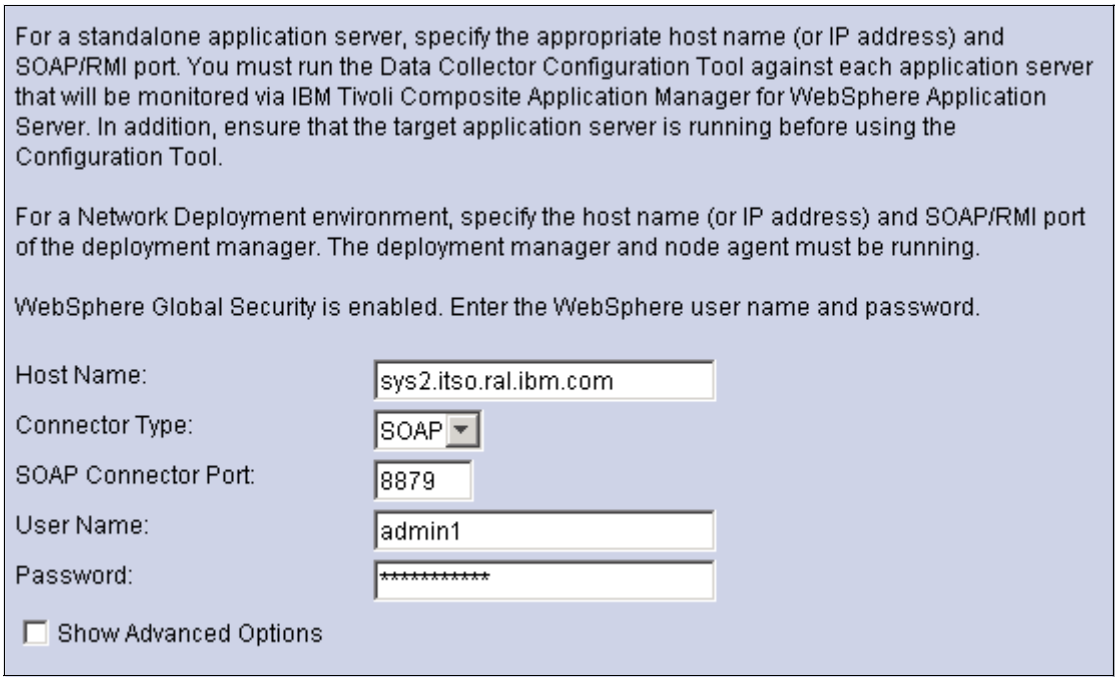


Figure 37 Enter administration port details

6. Specify which servers will have data collection configured and click **Next** (Figure 38).

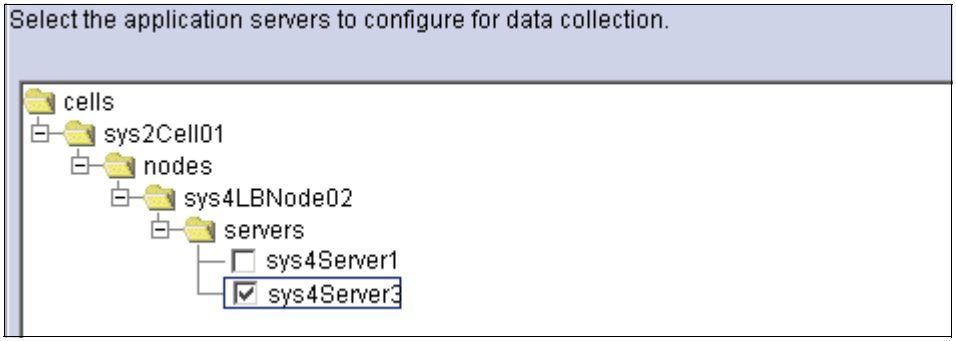


Figure 38 Choose servers to configure data collector for

7. As part of the data collection configuration, the PMI data settings can be modified to include ITCAM metrics, this can be done manually at another time or the administrator can allow the configuration tool to complete the modification (Figure 39). Click **Next**.

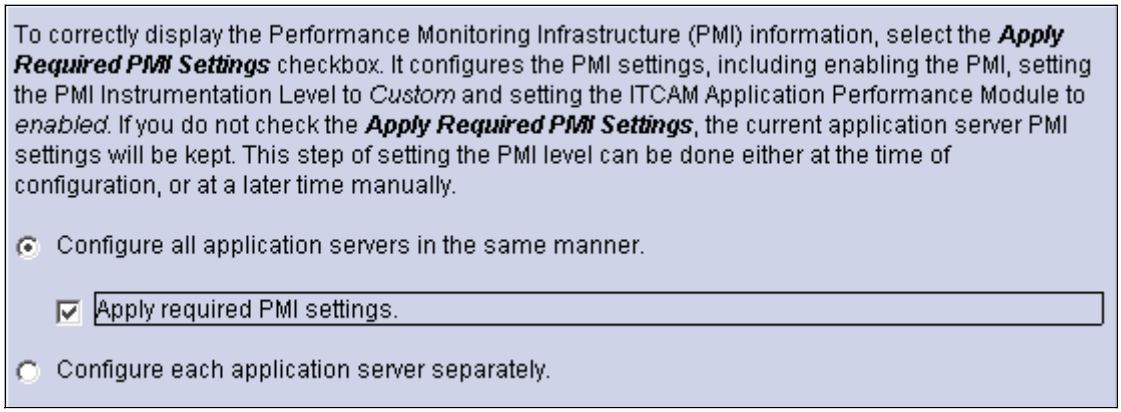


Figure 39 Modify PMI settings

8. Finalize the configuration steps (Figure 40) and click **Next**.

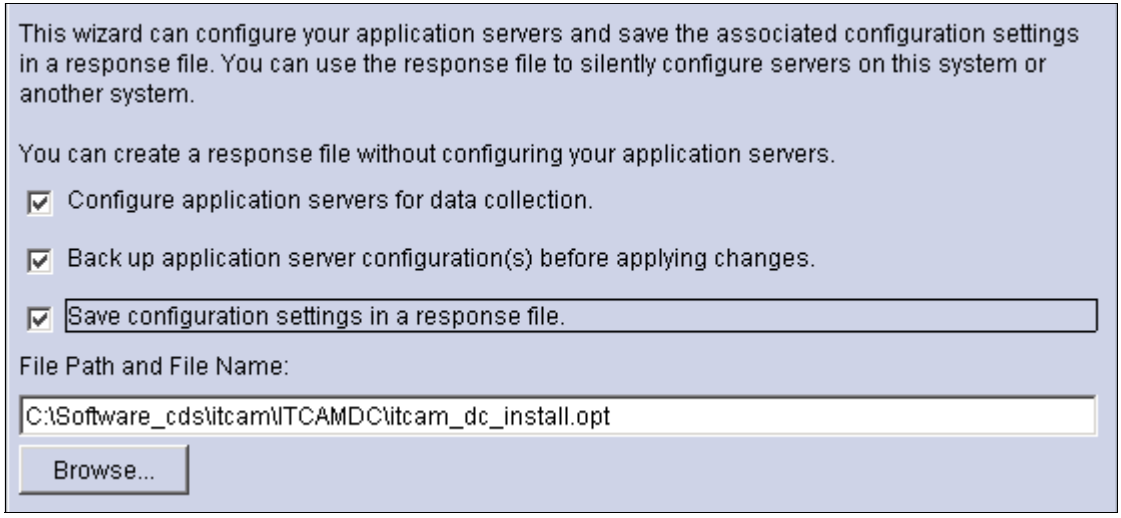


Figure 40 Configure the servers for data collection and create options file

9. Click **Finish**.
10. After the data collector configuration is complete, navigate in the WebSphere administration console to the Performance Monitoring Infrastructure panel for the configured server (Figure 41). A new **ITCAM for WAS** link is now in the Additional Properties section of the panel.

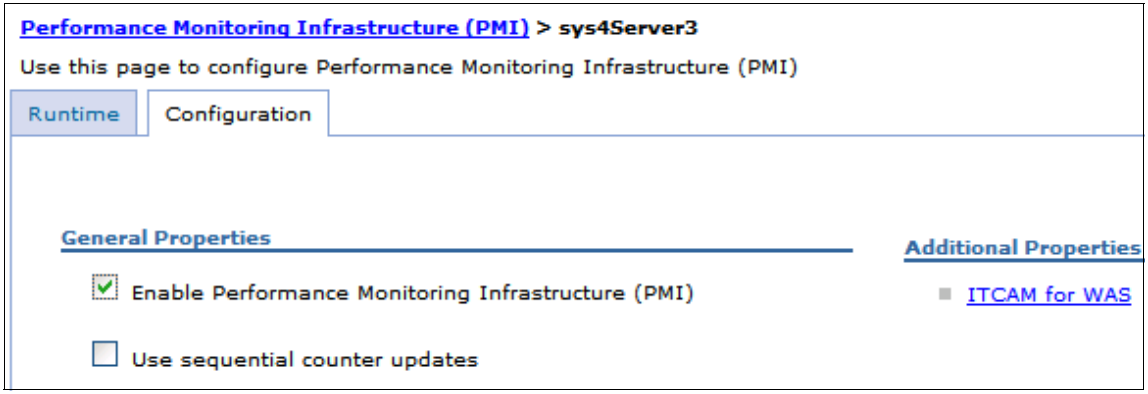


Figure 41 PMI configuration with ITCAM for WAS

11. Select the **ITCAM for WAS** link to open the ITCAM for WAS panel (Figure 42). This panel contains the setting that enables the data collector.

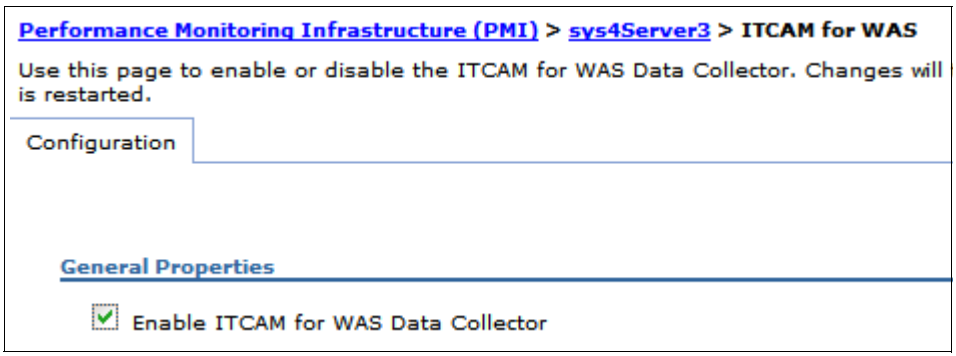


Figure 42 ITCAM for WAS panel

Note: This setting will only be enabled if you choose to apply the required PMI settings as shown in Figure 36 on page 49. Otherwise you will need to enable the ITCAM for WAS data collector and restart the server.

The PMI settings must also be configured at custom level.

12. Restart the servers for which the data collector was configured to make the metrics become active.

Viewing ITCAM for WebSphere data

This section assumes that the data collector has been installed and the ITCAM for WAS data collector is enabled as shown in Figure 42 on page 52.

1. Navigate to the Performance Monitoring Infrastructure panel for the server (Figure 43). Confirm that the **Custom** statistic set is selected.

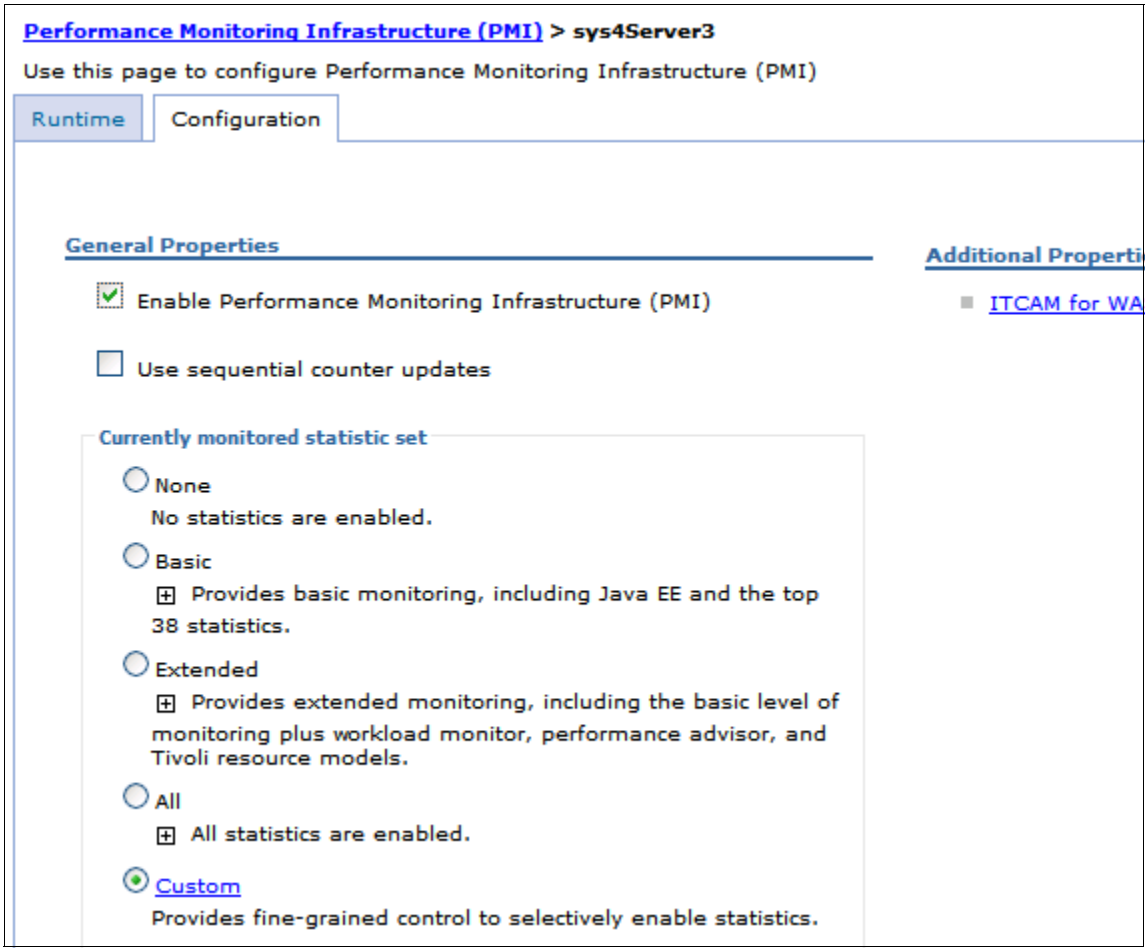


Figure 43 PMI Configuration panel

2. Click the **ITCAM for WAS** link and navigate to open the configuration panel (Figure 44).

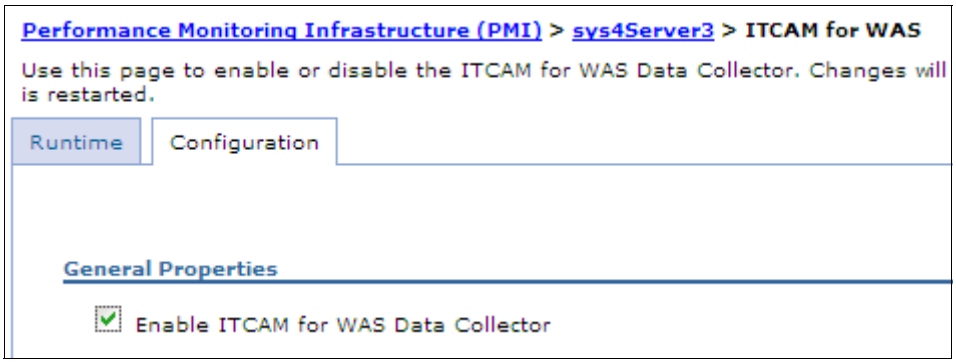


Figure 44 ITCAM for WAS configuration tab

3. Click the **Runtime** tab (Figure 45).

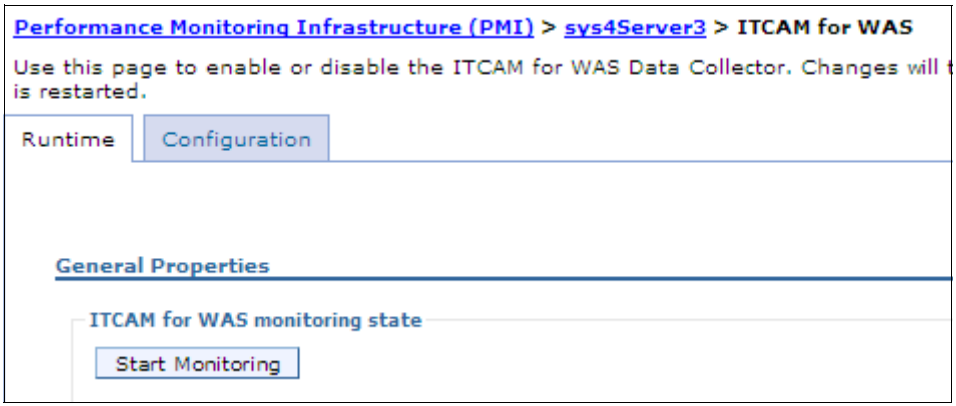


Figure 45 ITCAM for WAS runtime tab

4. Click the **Start Monitoring** button.

> **Notes:**
>
> You must start ITCAM for WAS monitoring on a server each server restart. It is not preserved in the configuration.
>
> After monitoring is started, the **Stop Monitoring** button is displayed instead. If you decide later to stop the ITCAM for WAS monitoring then return to this panel and click the button.

5. Click the server link (sys4Server3) in the navigation trail at the top of the panel to return to the PMI configuration panel.

6. Click the **Custom** link to navigate into the custom monitoring panel. In this panel select the **Runtime** tab. Click **ITCAM Application Performance** to show the ITCAM counters (Figure 46).

> **Note:** ITCAM counters are only visible in the Runtime tab.

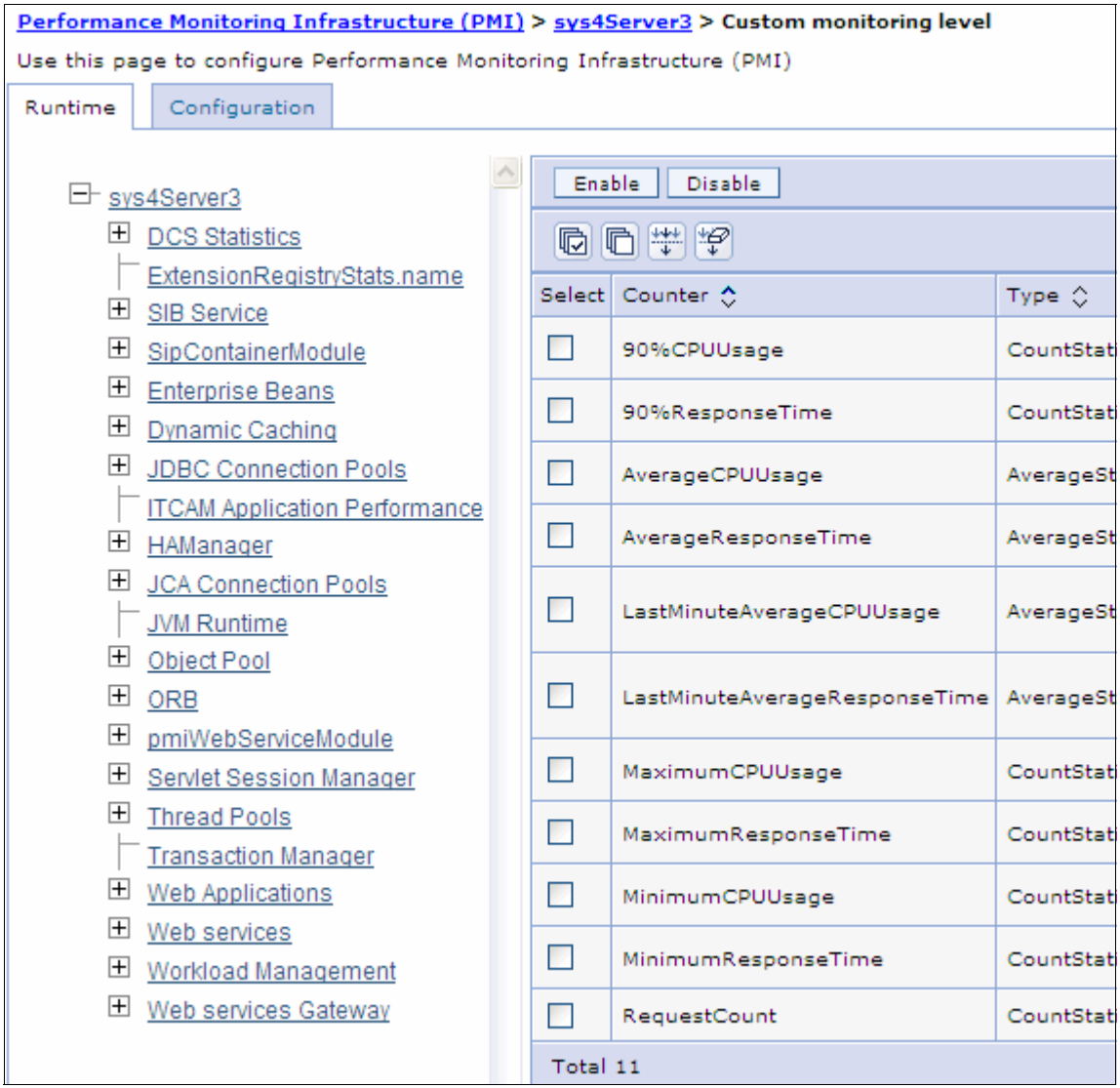
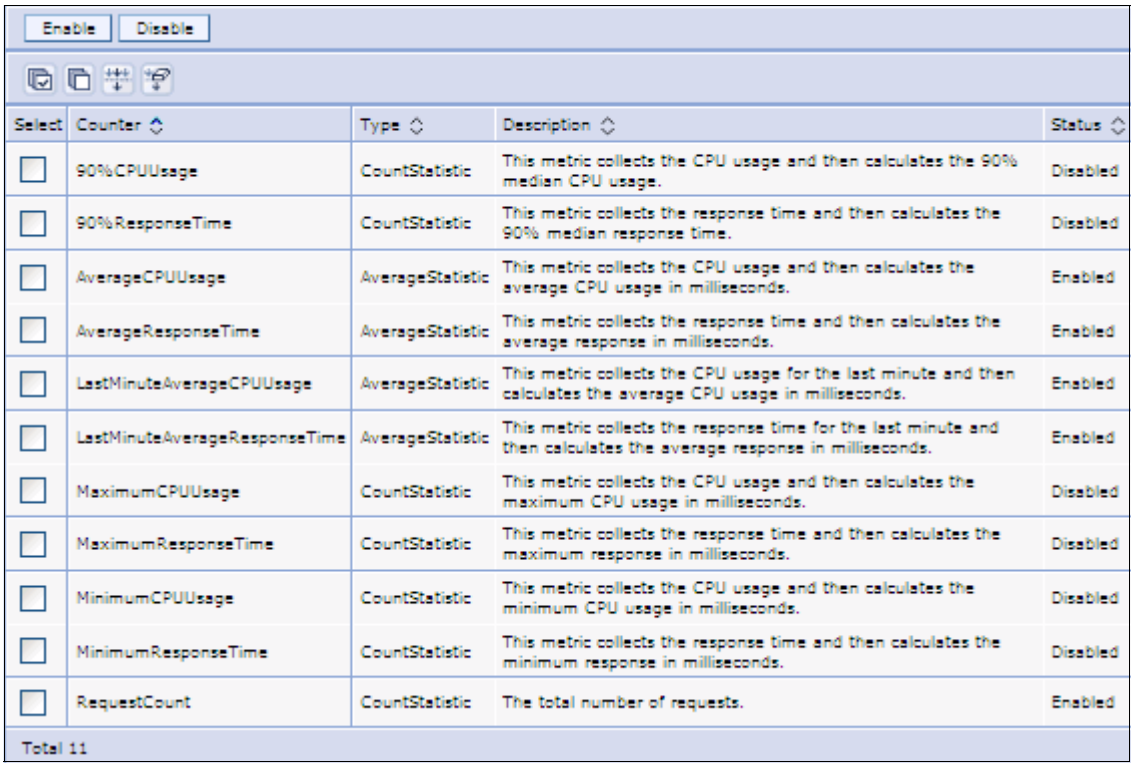


Figure 46 ITCAM Application performance counters

7. The next step is to choose which ITCAM for WAS counters are needed. The counter panel provides additional description information and the current status of the counter.

   Choose the counters that are desired and click **Enable**.

   Note that some counters can only be activated by the system. This means that they will stay in disabled mode, even if you try to enable them. This does not mean the counters are broken. In Figure 47, you can see that some counters remain in the Disabled state, even though all the counters were selected to be enabled.



Select	Counter	Type	Description	Status
☐	90%CPUUsage	CountStatistic	This metric collects the CPU usage and then calculates the 90% median CPU usage.	Disabled
☐	90%ResponseTime	CountStatistic	This metric collects the response time and then calculates the 90% median response time.	Disabled
☐	AverageCPUUsage	AverageStatistic	This metric collects the CPU usage and then calculates the average CPU usage in milliseconds.	Enabled
☐	AverageResponseTime	AverageStatistic	This metric collects the response time and then calculates the average response in milliseconds.	Enabled
☐	LastMinuteAverageCPUUsage	AverageStatistic	This metric collects the CPU usage for the last minute and then calculates the average CPU usage in milliseconds.	Enabled
☐	LastMinuteAverageResponseTime	AverageStatistic	This metric collects the response time for the last minute and then calculates the average response in milliseconds.	Enabled
☐	MaximumCPUUsage	CountStatistic	This metric collects the CPU usage and then calculates the maximum CPU usage in milliseconds.	Disabled
☐	MaximumResponseTime	CountStatistic	This metric collects the response time and then calculates the maximum response in milliseconds.	Disabled
☐	MinimumCPUUsage	CountStatistic	This metric collects the CPU usage and then calculates the minimum CPU usage in milliseconds.	Disabled
☐	MinimumResponseTime	CountStatistic	This metric collects the response time and then calculates the minimum response in milliseconds.	Disabled
☐	RequestCount	CountStatistic	The total number of requests.	Enabled

Total 11

Figure 47 ITCAM counters enabled

8. After enabling the counters, navigate to the Current Activity Tivoli Performance Viewer panel for the server that is being monitored (Figure 48). Open the Performance Modules tree view and then select the **ITCAM Application Performance** menu item.

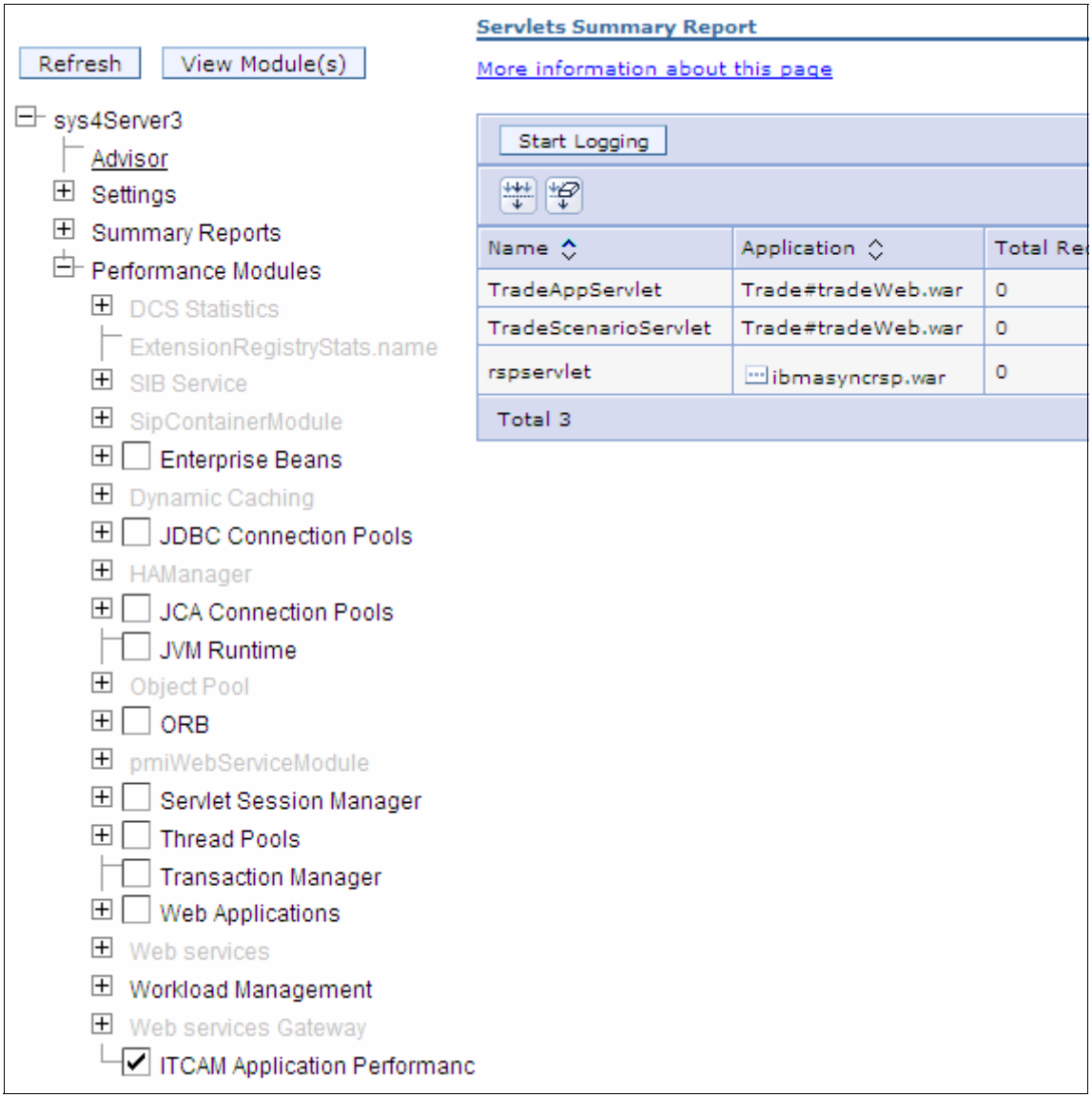


Figure 48 Select the ITCAM Application performance menu item

9. Click the **View Module(s)** button to view the data (Figure 49).

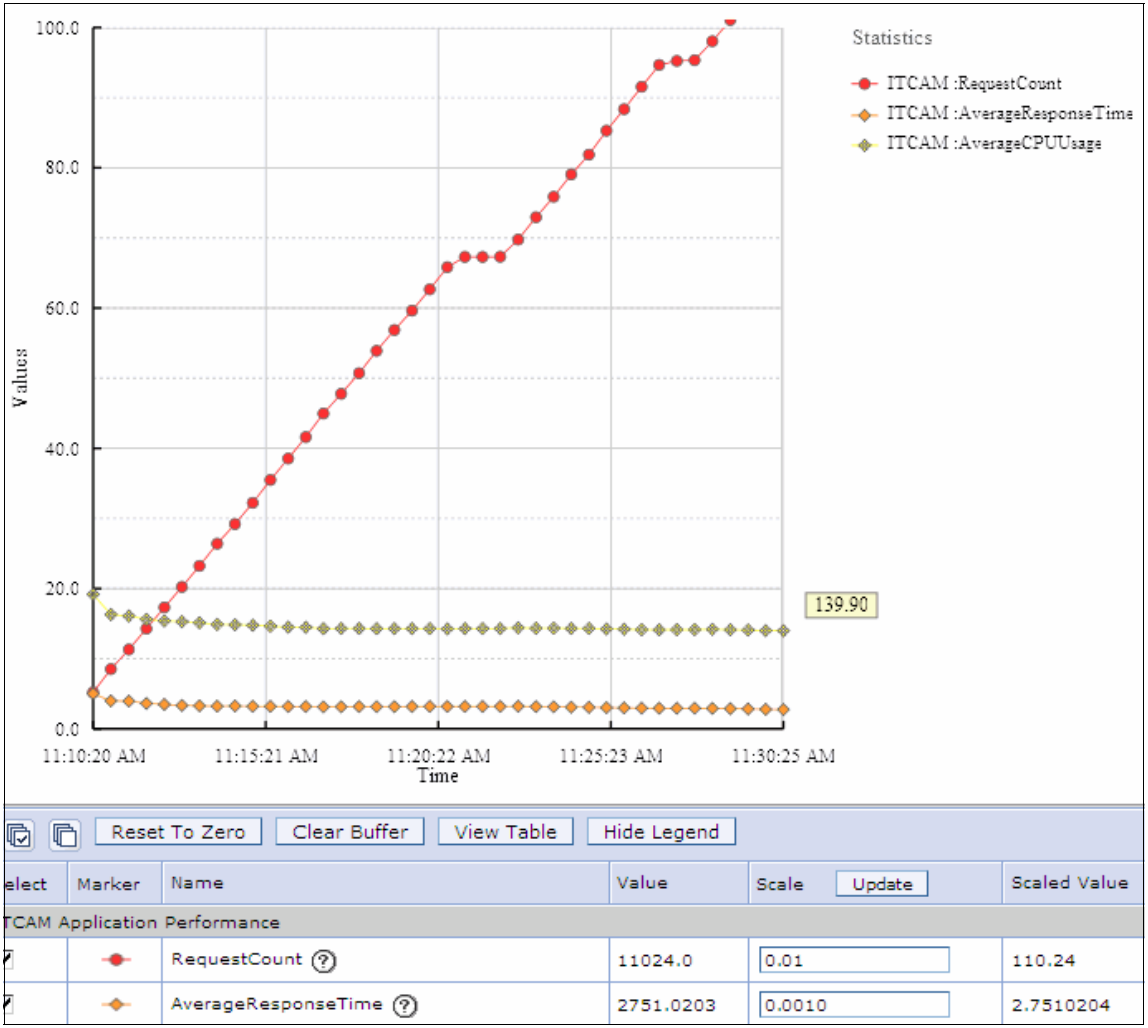


Figure 49 ITCAM counters graph

Monitoring considerations summary

In this section we provide some final thoughts and advice on monitoring.

Other tools and considerations

There are a number of other sources of information an administrator can use in relation to monitoring the health of an application server.

Such resources include logging, garbage collection, and operating system views:

- Logging:

  WebSphere Application Server provides a great many logging options and the most significant environment incidents are logged automatically. To support the analysis of logging activity, the IBM Support assistant can be used to launch the IBM Tivoli Log Analyzer. This tool supports the downloading of symptom catalogs of known issues that go into the logs. Further, application developers can build symptom catalogs and write logs from their applications to add to the manageability of their applications.

  Figure 50 shows the IBM Tivoli Log analyzer.

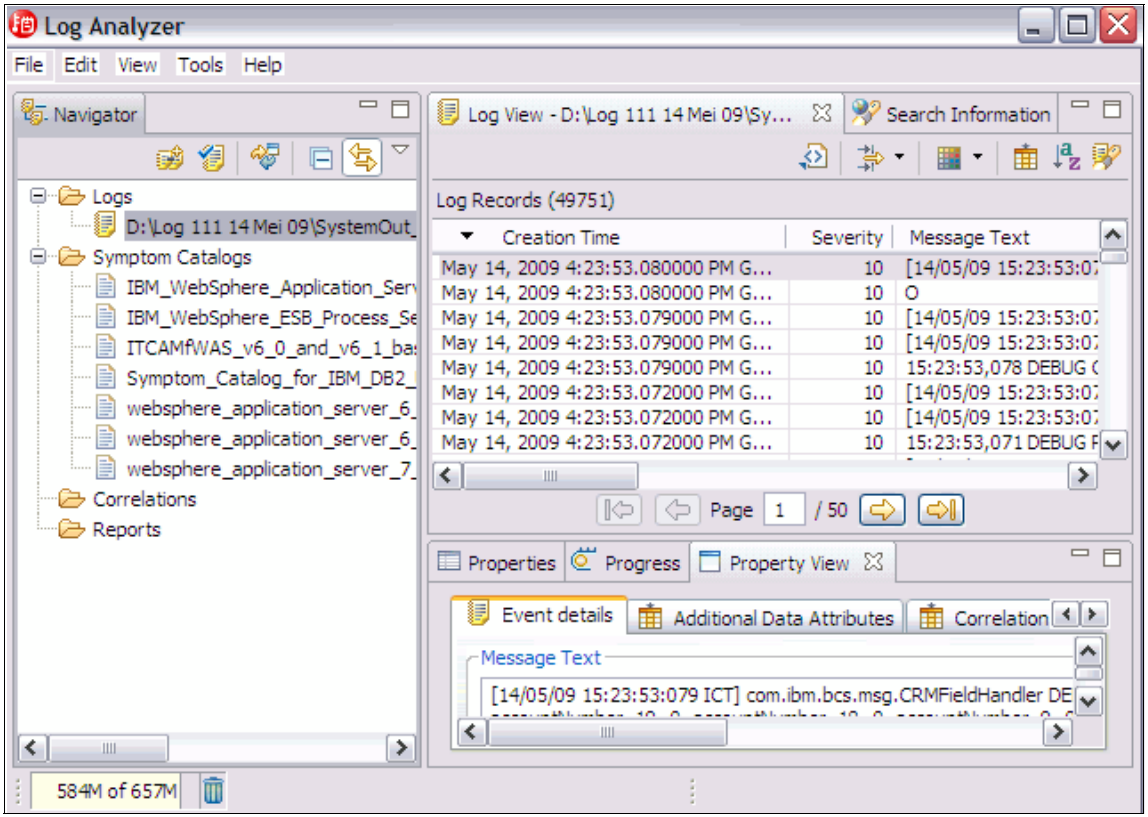


Figure 50 IBM Tivoli log analyzer

- Verbose garbage collection:

  Verbose garbage collection was mentioned earlier in the chapter. Again, we recommend that, assuming there is appropriate disk space, a monitoring strategy would include verbose garbage collection. To enable verbose garbage collection, navigate to the process definition for the JVM of the server (Figure 51) and check the **Verbose garbage collection** box.

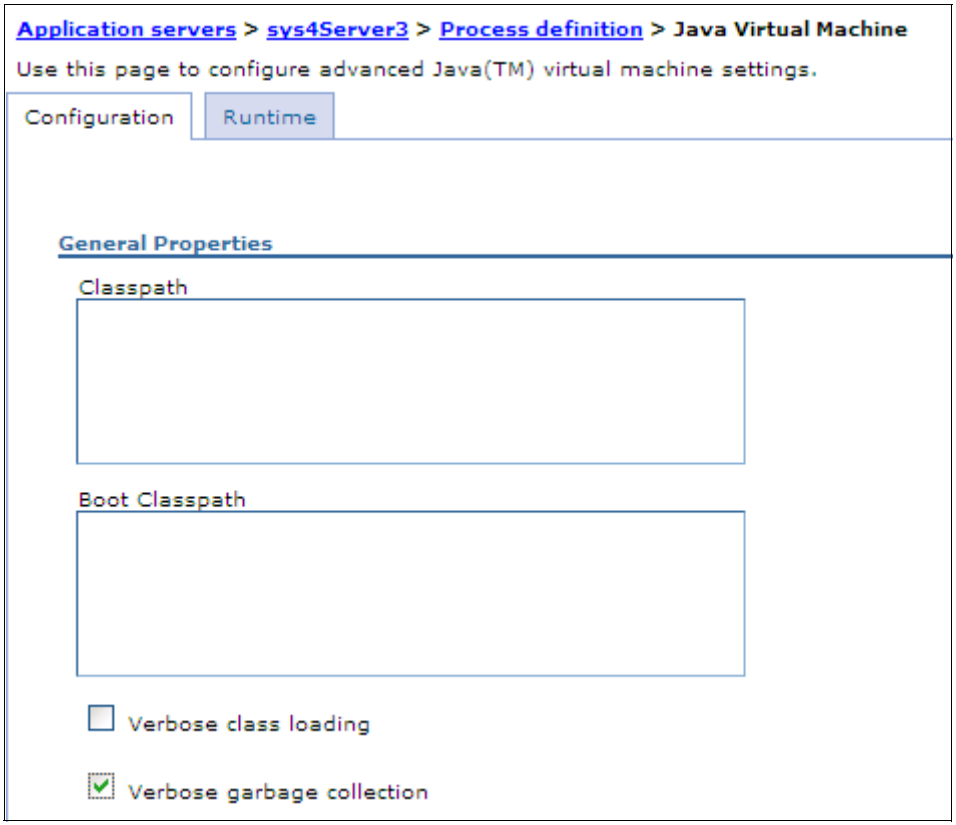


Figure 51 Enable verbose garbage collection

Verbose garbage collection can also be enabled at runtime (Figure 52).

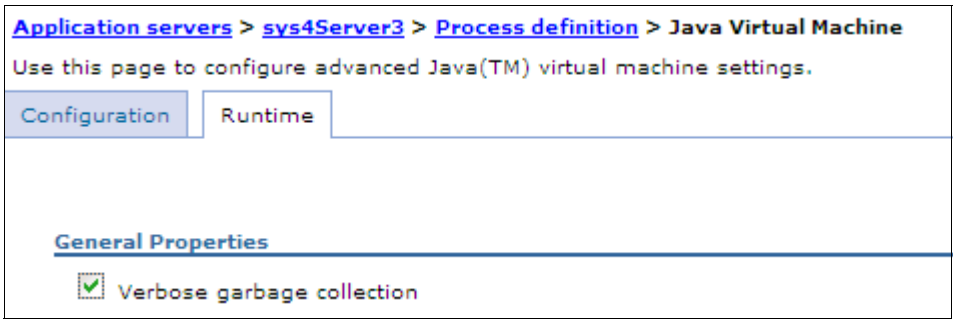


Figure 52 Enable verbose garbage collection at runtime

- Operating system views:

  Most operating systems have tools for the monitoring of memory, CPU utilization, and disk I/O. These should be monitored and controlled.

- The monitoring tools available with WebSphere Application Server are useful in any size enterprise. However, in larger enterprises, a more complete system monitoring solution (such as those provided in the advanced ITCAM product suites) are more suitable for improved centralized monitoring. Tools that can monitor systems end-to-end are very beneficial.

Summary of monitoring tips

The following summary lists a simple set of recommendations on how to establish a useful monitoring environment.

- Take time and understand the applications that are being deployed into the environment. Use this knowledge to plan for the kinds of metrics that will be beneficial in understanding application performance.
- Activate monitoring at the planned level in all testing environments especially when benchmarking. While vital to you ability to understand an environment, monitoring is not free and uses CPU, memory and other resources. Monitoring can impact capacity planning.
- Utilize monitoring to understand normal operations and gain an appreciation of what is normal for your systems.
- Check for differences in your systems after all release changes, especially if full performance testing and benchmarks have not been re-established.
- Use the basic metrics of PMI as a good starting set of metrics, but customize them to your needs.
- Monitoring WebSphere Application Server alone is not enough. Application server monitoring needs to be part of a overall monitoring strategy.

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

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

This document REDP-4579-00 was created or updated on October 13, 2009.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, 2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks