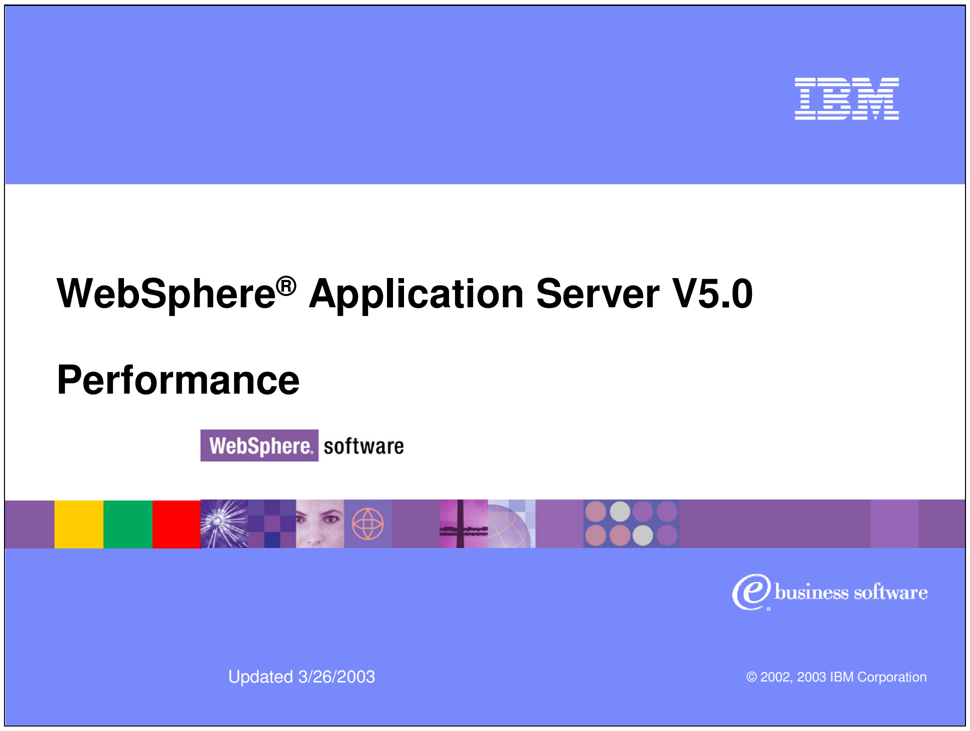


WebSphere® Application Server V5.0

Performance

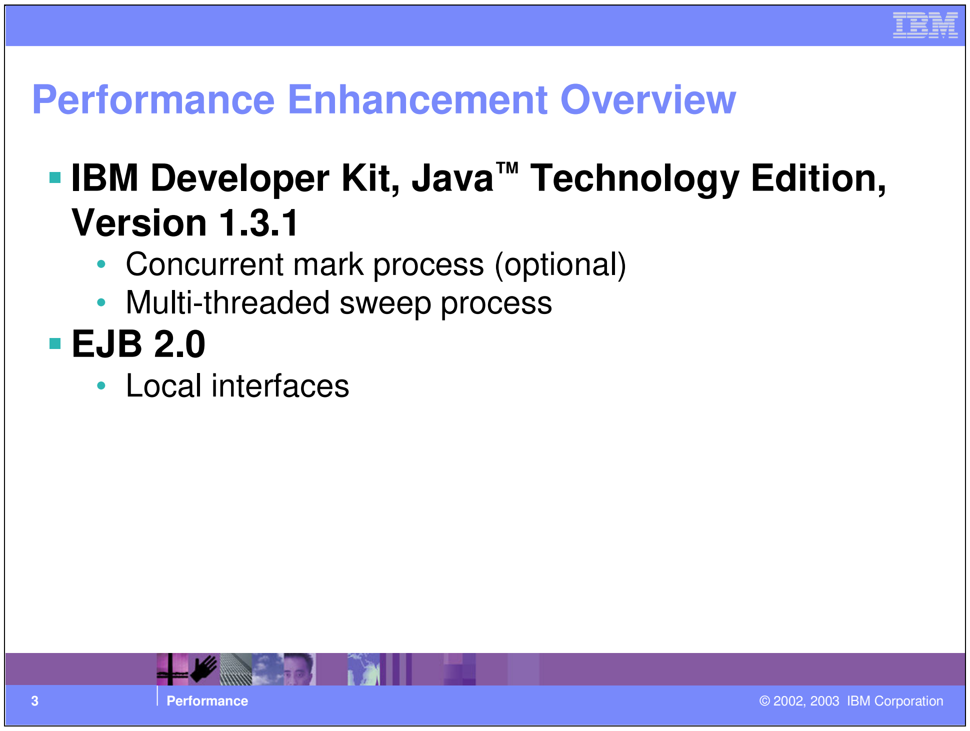
WebSphere. software

business software

Updated 3/26/2003

Objectives

- Highlight performance enhancements in WebSphere Application Server 5.0
- Dynamic caching improvements
- Discuss new features in PMI
- Introduce Tivoli Performance Viewer
- Discuss HTTP Session performance information
- Investigate queuing and pool management
- Introduce V5 Technology Previews



Performance Enhancement Overview

- **IBM Developer Kit, Java™ Technology Edition, Version 1.3.1**
  - Concurrent mark process (optional)
  - Multi-threaded sweep process
- **EJB 2.0**
  - Local interfaces

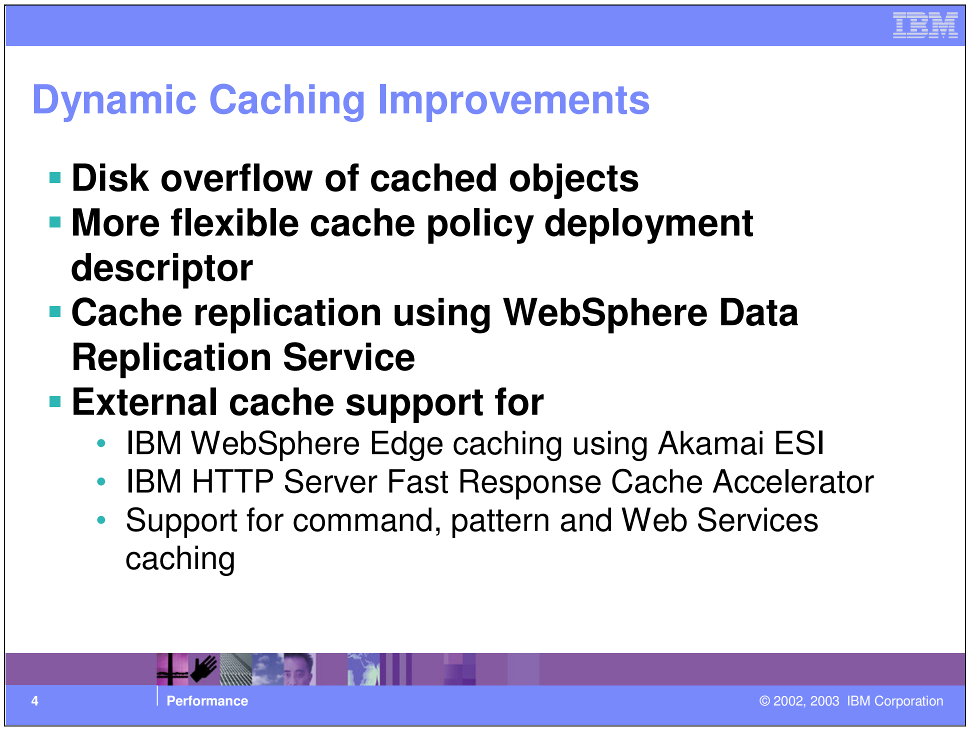
3 Performance © 2002, 2003 IBM Corporation

See the SPEC JBB2000 for comparisons of JDK 1.3.1 vs. JDK 1.3.0

One of the major differences with IBM Developer Kit is that the mark process can now be asynchronous. The sweep and compaction process, however, still remain synchronous and will stop all other work in the JVM when being executed.

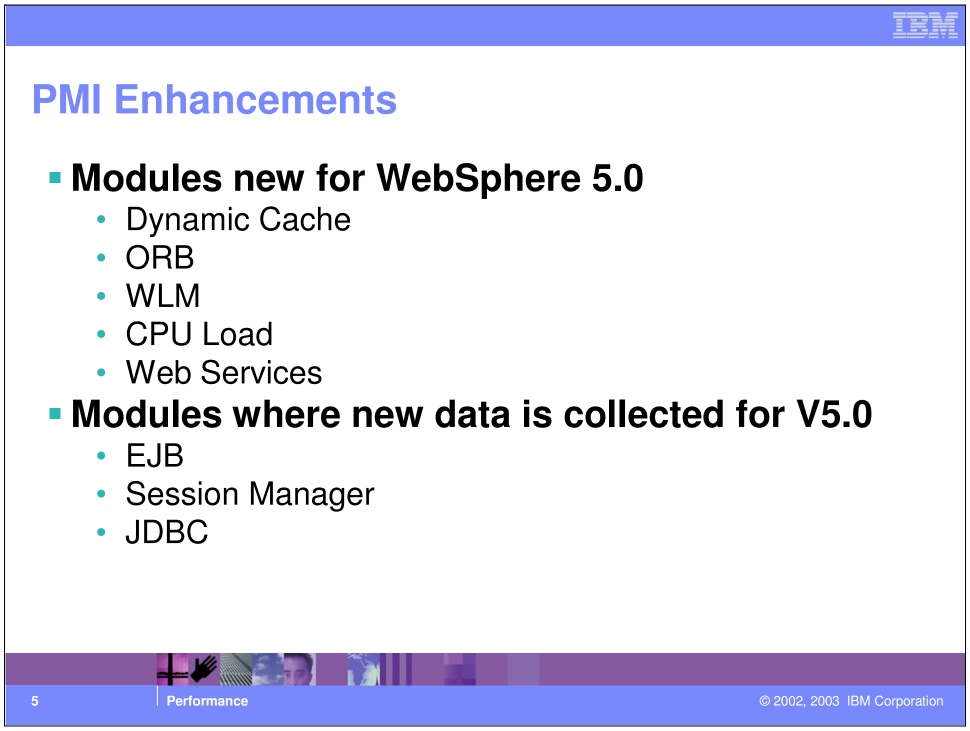
EJB 2.0

The benefits of local interfaces are covered in detail in another module.



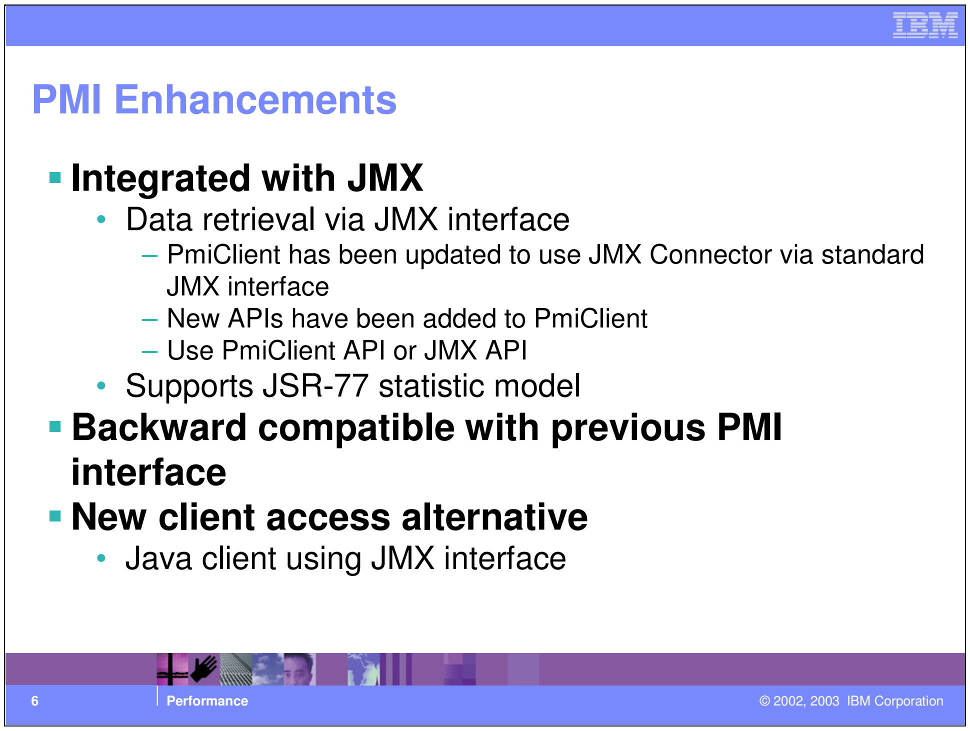
Dynamic Caching Improvements

- **Disk overflow of cached objects**
- **More flexible cache policy deployment descriptor**
- **Cache replication using WebSphere Data Replication Service**
- **External cache support for**
  - IBM WebSphere Edge caching using Akamai ESI
  - IBM HTTP Server Fast Response Cache Accelerator
  - Support for command, pattern and Web Services caching



PMI Enhancements

- **Modules new for WebSphere 5.0**
  - Dynamic Cache
  - ORB
  - WLM
  - CPU Load
  - Web Services
- **Modules where new data is collected for V5.0**
  - EJB
  - Session Manager
  - JDBC



PMI Enhancements

- **Integrated with JMX**
  - Data retrieval via JMX interface
    - PmiClient has been updated to use JMX Connector via standard JMX interface
    - New APIs have been added to PmiClient
    - Use PmiClient API or JMX API
  - Supports JSR-77 statistic model
- **Backward compatible with previous PMI interface**
- **New client access alternative**
  - Java client using JMX interface

6 Performance © 2002, 2003 IBM Corporation

The PMI infrastructure has been updated to support the additional performance data types that JMX provides (JSR-77). The performance data types that JMX defines are:

CountStatistic

TimeStatistic

BoundaryStatistic

RangeStatistic

BoundedRangeStatistic

PMI now provides support for all of the statistical providers in JSR-77 along with the following providers not included in the JSR:

ORB

JNDI

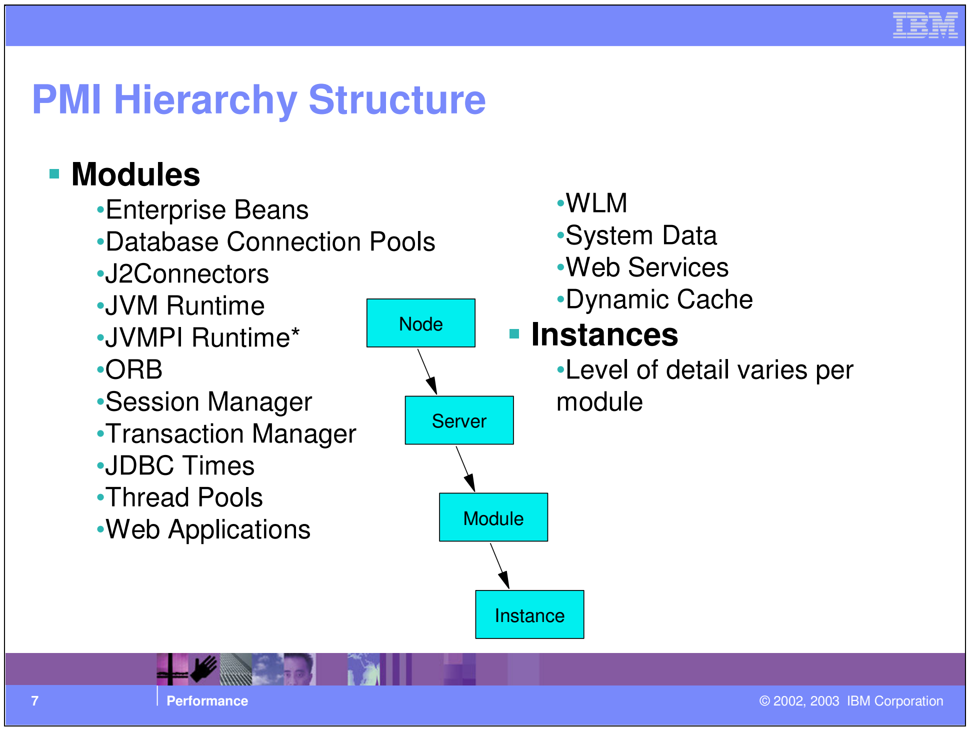
WLM

Security

The existing APIs in the PmiClient wrapper have been kept the same so that vendors/customers who have written to 4.0 PmiClient APIs do not have to change their existing applications.

The PmiClient has been updated to use the EJB JMX Connector to communicate with MBeans through standard JMX interface. It will also call the system management API to get a list of nodes and application servers in the domain.

Because the PMI infrastructure has been changed to utilize the JMX infrastructure, vendors and clients now have a new alternative way to access performance information.



PMI Hierarchy Structure

- **Modules**
  - •Enterprise Beans
  - •Database Connection Pools
  - •J2Connectors
  - •JVM Runtime
  - •JVMPI Runtime*
  - •ORB
  - •Session Manager
  - •Transaction Manager
  - •JDBC Times
  - •Thread Pools
  - •Web Applications
  - •WLM
  - •System Data
  - •Web Services
  - •Dynamic Cache
- **Instances**
  - •Level of detail varies per module

$$\begin{tikzcd}\text{Node} \arrow[d] \\ \text{Server} \arrow[d] \\ \text{Module} \arrow[d] \\ \text{Instance}\end{tikzcd}$$

7 Performance © 2002, 2003 IBM Corporation

The list shows all of the available modules that can be monitored through the use of the PMI. The data metrics can then be viewed with any tool that is designed to interface with the PMI.

PerformanceServlet

Tivoli Performance Viewer

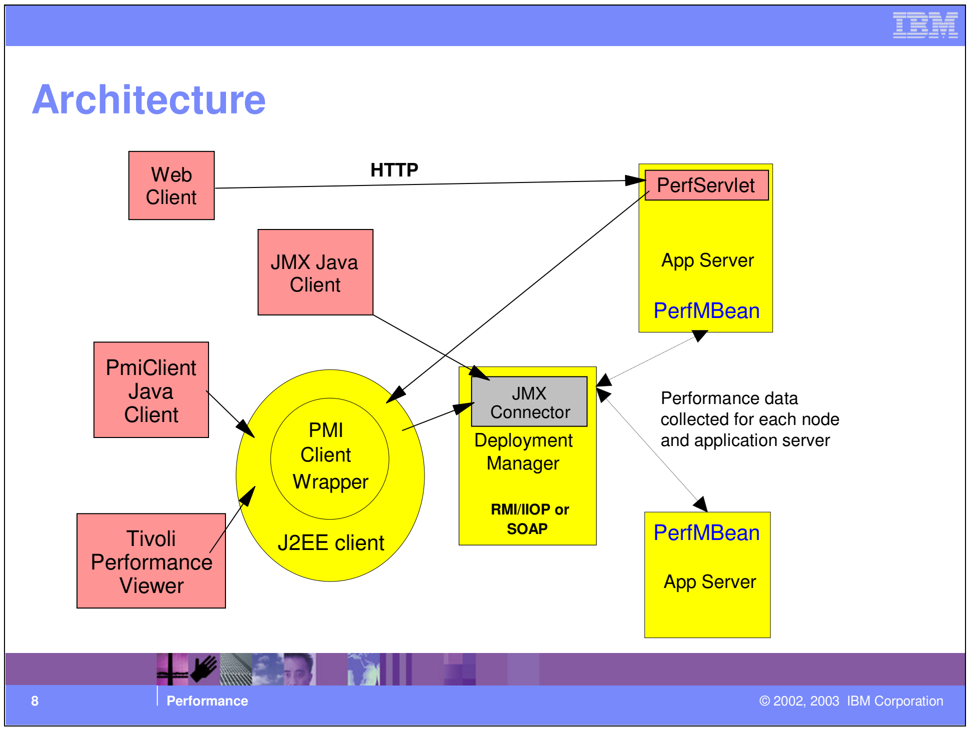
3rd Party Tools

Keep in mind that the details below each module will differ from one to the other and the InfoCenter should be consulted to get the exact list of all counters available for each module.

The amount of detail that can be captured through the PMI has been enhanced yet again in this release. You can now capture metrics down to a specific method, whereas before all methods within an EJB were lumped together. This will provided one more step of granularity when attempting to find performance bottlenecks.

* JVMPI Runtime is the only metric listed that requires additional setup in the application server to capture the metrics. There is an advanced JVM argument that needs to be set before the application server is started. The reason for this is because of the tremendous performance impact on the application when JVMPI metrics are being captured.

Approximately 175 different metrics can be captured by the PMI, in order to give a holistic view of the application performance and behavior during runtime.



Architecture

Clients using either a Java client to connect to the PmiClient or the JMX interface to gather performance information now have the choice of using either RMI/IIOP or SOAP.

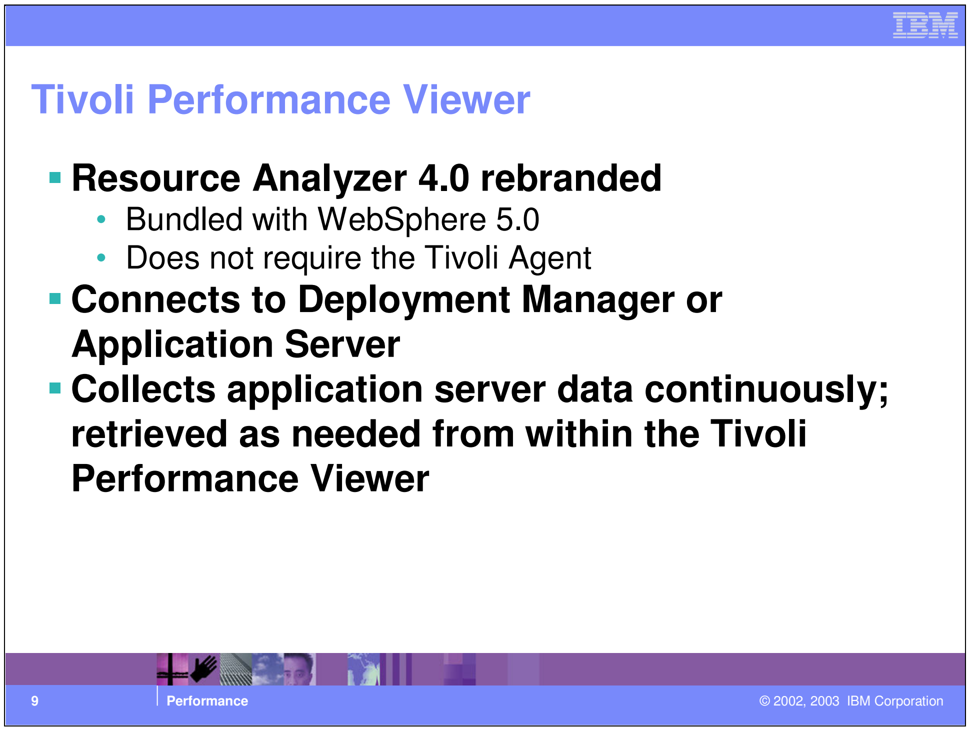
Clients access performance data through the PmiClient in the following manner:

The JMX interface is used to send the request for performance info to the appropriate MBean on the App Server where the information needs to be collected from.

Upon receiving the performance request, the MBean will delegate the request to the PmiCollaborator interface, which parses the parameters, converts the MBean ID to a PMI data path, finds the requested data, and returns the data to the MBean.

The performance info is then passed back through the JMX system to the client making the request.

The items in pink indicate the various methods of accessing performance data in WebSphere 5.0.



Tivoli Performance Viewer

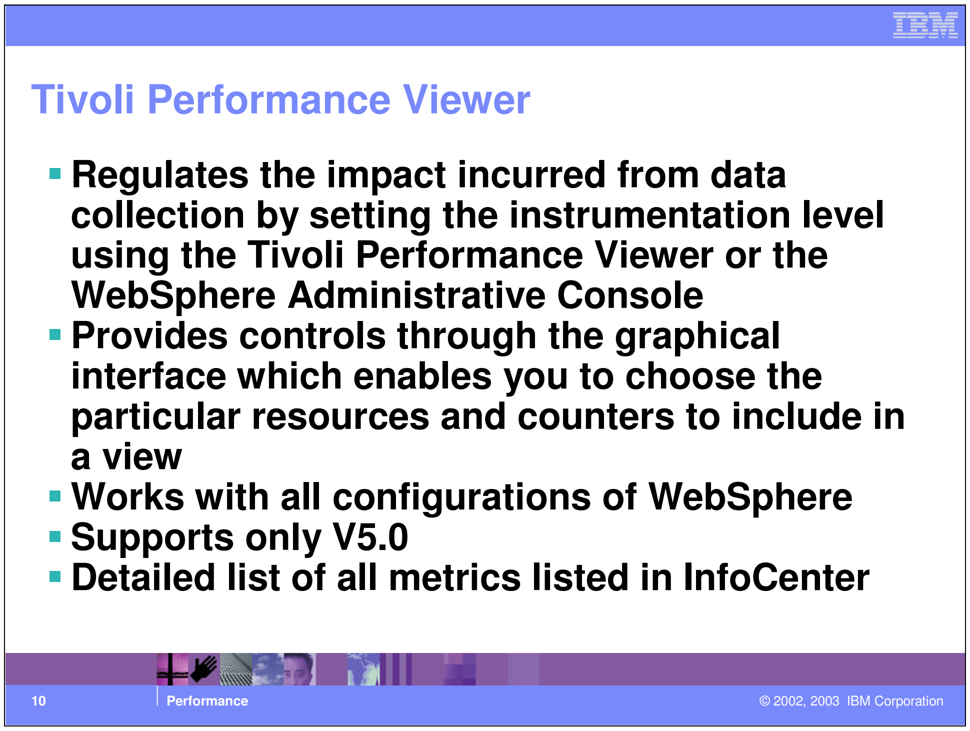
- **Resource Analyzer 4.0 rebranded**
  - Bundled with WebSphere 5.0
  - Does not require the Tivoli Agent
- **Connects to Deployment Manager or Application Server**
- **Collects application server data continuously; retrieved as needed from within the Tivoli Performance Viewer**

9 Performance © 2002, 2003 IBM Corporation

Tivoli Performance Viewer connects directly to the Application Server when the Base configuration is installed

Tivoli Performance Viewer connects to the Deployment Manager when the Network Deployment configuration and higher are installed.

Tivoli Performance Viewer is Resource Analyzer, rebranded to fit into IBM's marketing scheme.



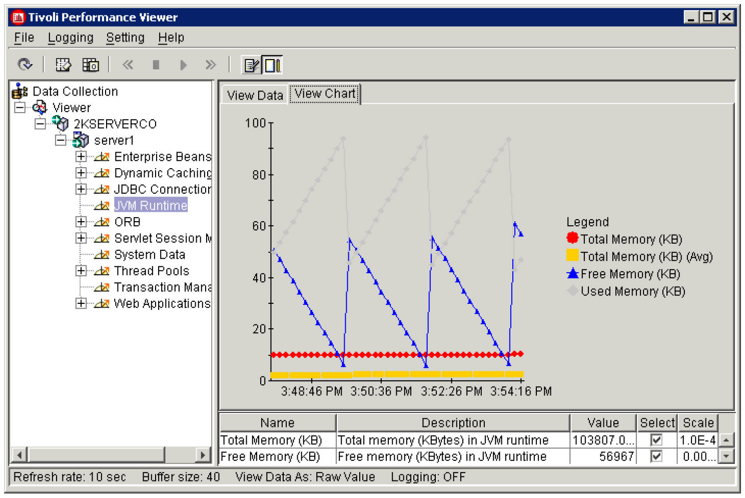
Tivoli Performance Viewer

- Regulates the impact incurred from data collection by setting the instrumentation level using the Tivoli Performance Viewer or the WebSphere Administrative Console
- Provides controls through the graphical interface which enables you to choose the particular resources and counters to include in a view
- Works with all configurations of WebSphere
- Supports only V5.0
- Detailed list of all metrics listed in InfoCenter

10 | Performance © 2002, 2003 IBM Corporation

Tivoli Performance Viewer is not backwards compatible with previous versions of WebSphere. It will only work WebSphere 5.0.

Tivoli Performance Viewer - Scaling



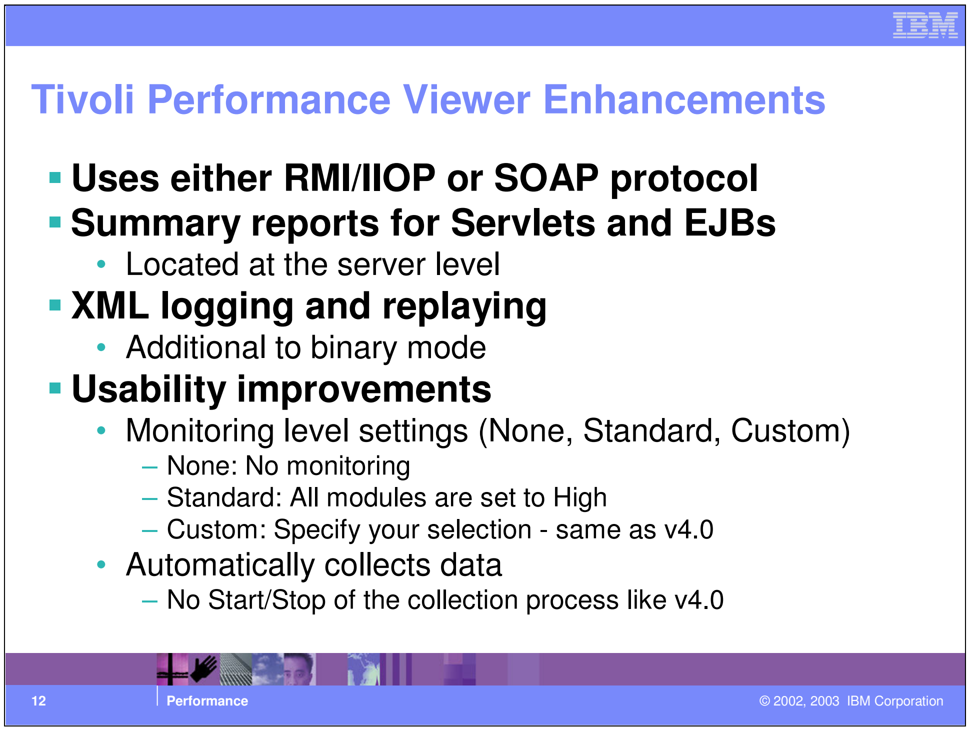
Name	Description	Value	Select	Scale
Total Memory (KB)	Total memory (KBytes) in JVM runtime	103807.0...	✓	1.0E-4
Free Memory (KB)	Free memory (KBytes) in JVM runtime	56967	✓	0.00...

Refresh rate: 10 sec Buffer size: 40 View Data As: Raw Value Logging: OFF

11 Performance © 2002, 2003 IBM Corporation

The graphical view of the Tivoli Performance Viewer has an auto-scale feature, which can cause confusion sometimes when looking at the data on the graph. This can be changed manually by the user in the scale column of the data.

The select column only indicates which data is being graphed. All the metrics in the list will actually be captured and can be displayed in the chart at any time.



Tivoli Performance Viewer Enhancements

- **Uses either RMI/IIOP or SOAP protocol**
- **Summary reports for Servlets and EJBs**
  - Located at the server level
- **XML logging and replaying**
  - Additional to binary mode
- **Usability improvements**
  - Monitoring level settings (None, Standard, Custom)
    - None: No monitoring
    - Standard: All modules are set to High
    - Custom: Specify your selection - same as v4.0
  - Automatically collects data
    - No Start/Stop of the collection process like v4.0

12 | Performance © 2002, 2003 IBM Corporation

A couple of things to keep in mind when using XML log files are:

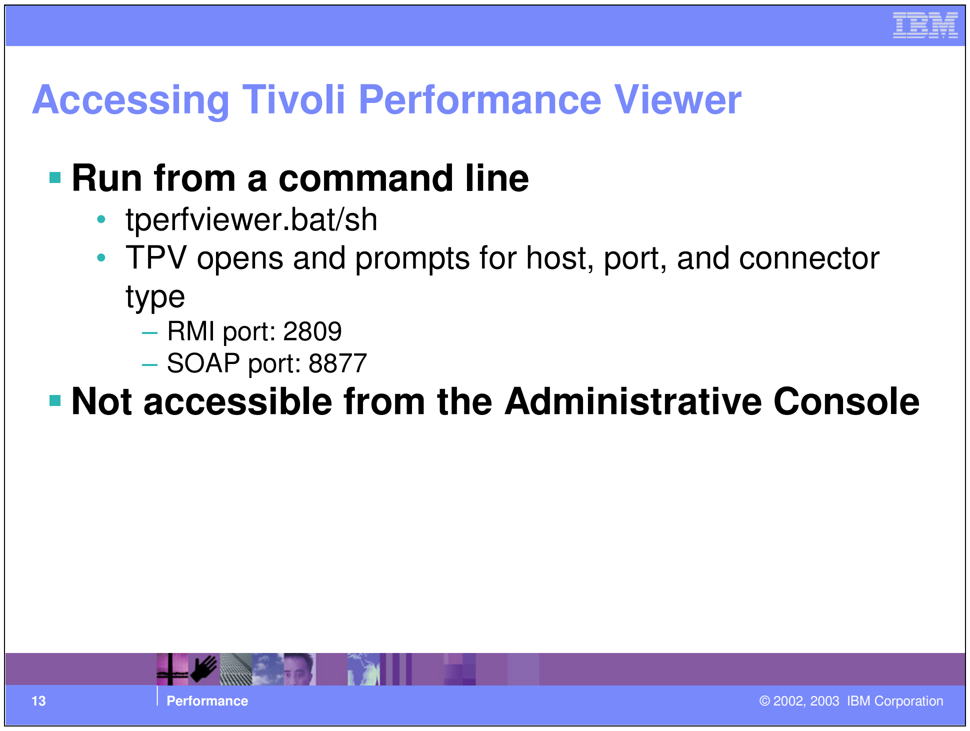
The XML logs will be larger than the binary logs, so if space is a concern for your system you should be aware of this.

The XML log will be able to be replayed with the TPV just like the binary files, but will also allow you to read the data into other tools if you have the need to do further analysis.

The monitoring levels map back to similar levels in V4.

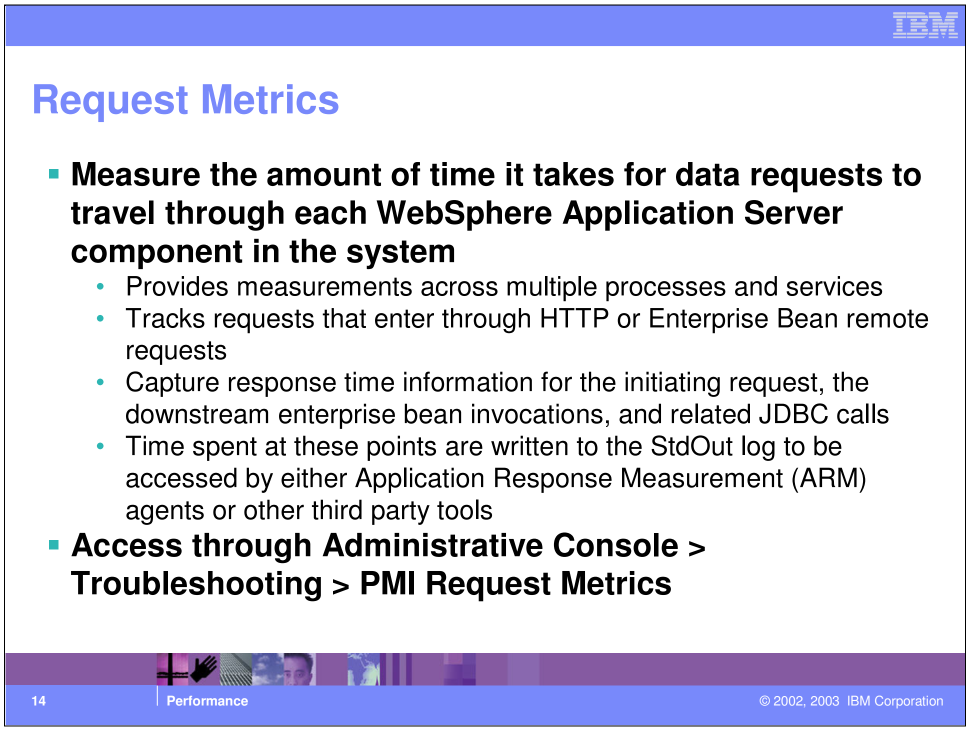
Standard - Set all the modules to the high level. The performance impact for this will be between 1 - 3 percent.

Custom - Allows you to specify any level you desire, including the maximum metric capturing that can result in a 5 percent impact on performance.



Accessing Tivoli Performance Viewer

- **Run from a command line**
  - tperfviewer.bat/sh
  - TPV opens and prompts for host, port, and connector type
    - RMI port: 2809
    - SOAP port: 8877
- **Not accessible from the Administrative Console**



Request Metrics

- **Measure the amount of time it takes for data requests to travel through each WebSphere Application Server component in the system**
  - Provides measurements across multiple processes and services
  - Tracks requests that enter through HTTP or Enterprise Bean remote requests
  - Capture response time information for the initiating request, the downstream enterprise bean invocations, and related JDBC calls
  - Time spent at these points are written to the StdOut log to be accessed by either Application Response Measurement (ARM) agents or other third party tools
- **Access through Administrative Console > Troubleshooting > PMI Request Metrics**

14 | Performance © 2002, 2003 IBM Corporation

Performance Monitoring Infrastructure (PMI) Request Metrics helps identify run time and application performance problems by capturing process hop response times in multi-tiered applications and recording the data in system logs. For requests that start from either an HTTP or enterprise bean remote requests, Request Metrics captures response times for the initiating request and any related downstream enterprise bean invocations and Java Database Connectivity (JDBC) calls. Request Metrics also provides the same information on process hop response time through the Application Response Measurement (ARM) interface.

Application Response Measurement (ARM) - Provides a mechanism whereby applications can provide response time measurement to a centralized reporting and management facility.

WebSphere Application Server does not provide an ARM agent, but can be used with agents conforming to the ARM 2.0 standard. Current support for Tivoli ARM.

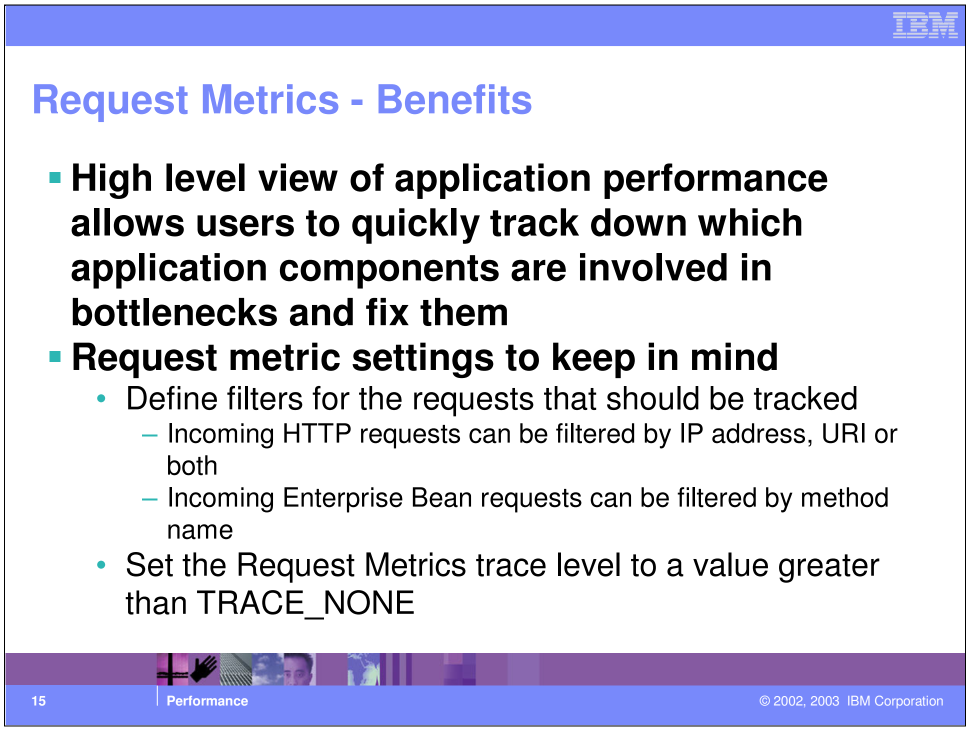
When active, Request Metrics compares each incoming request to a set of known filters. If the request matches any filter with a trace level greater than TRACE_NONE, trace records are generated for that request.

This trace information is recorded to the StdOut system log (located in the <WAS_ROOT_INSTALL_DIR>\logs\<SERVER_NAME> directory) for the application server that the request metrics are being monitored on. The request metric trace information will have the following syntax in the StdOut system log (for more information on Request Metrics see the WebSphere V5 InfoCenter):

```
PMRM0003I: parent:ver=n,ip=n.n.n.n,time=nnnnnnnnnn,pid=nnnn,reqid=nnnnnn,event=nnnn
- current:ver=n,ip=n.n.n.n,time=nnnnnnnnnn,pid=nnnn,reqid=nnnnnn,event=nnnn
     type=TTT detail=some_detail_information elapsed=nnnn
```

Typically, requests enter the system and create processes that fan out across several nodes within a distributed system. Each process can further fan out and call other processes. When the processes fan out, trace records are generated for each process. Then, these trace records can be correlated together to build a sequence diagram of the response times for the request. The processes are only recorded if they are generated through a remote enterprise bean call.

One way in which request metrics can be used to capture the current performance of your application is to use them, along with a load generator, to drive synthetic transactions against your server. The user can then set request metric filters, so that only the HTTP requests from the client generating the load are monitored. The trace information can then be gathered and analyzed to give a real-world measurement of the performance of the application. This method could be used in both a development or a production environment..



Request Metrics - Benefits

- **High level view of application performance allows users to quickly track down which application components are involved in bottlenecks and fix them**
- **Request metric settings to keep in mind**
  - Define filters for the requests that should be tracked
    - Incoming HTTP requests can be filtered by IP address, URI or both
    - Incoming Enterprise Bean requests can be filtered by method name
  - Set the Request Metrics trace level to a value greater than TRACE_NONE

15 | Performance © 2002, 2003 IBM Corporation

Request metrics filters

Incoming HTTP requests

Client IP Address Filters: Use client IP address filters in the data center to filter known addresses and provide a mechanism to make a request or set of requests while the system is under normal load.

URI Filters: URI filtering provides a mechanism to filter, based on the URI of the incoming HTTP request.

Incoming enterprise bean requests

Enterprise bean method name filters are specified with the fully package name qualified enterprise bean method name.

To enable request metrics in WebSphere do the following:

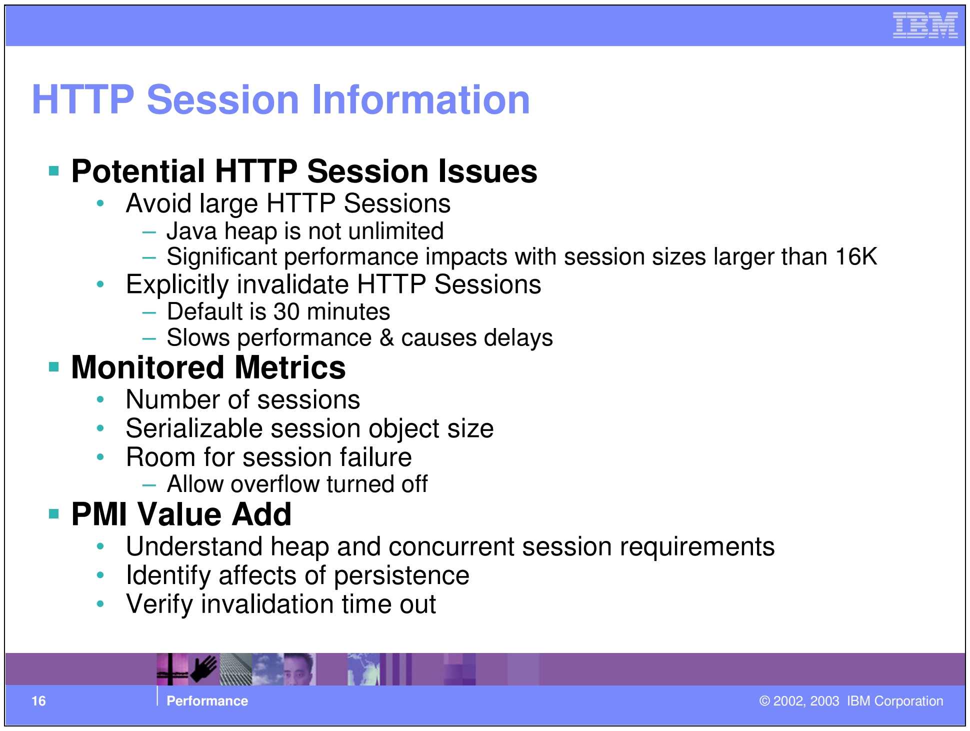
Click Troubleshooting > PMI Request Metrics in the administrative console navigation tree.

Select the check box in the enable field under the Configuration tab.

Click Apply or OK.

Save the changes.

The performance of the application server may be significantly impacted if no filters are set for the request metrics (meaning that the information will be collected on all of the available activity on the server).



HTTP Session Information

- **Potential HTTP Session Issues**
  - Avoid large HTTP Sessions
    - Java heap is not unlimited
    - Significant performance impacts with session sizes larger than 16K
  - Explicitly invalidate HTTP Sessions
    - Default is 30 minutes
    - Slows performance & causes delays
- **Monitored Metrics**
  - Number of sessions
  - Serializable session object size
  - Room for session failure
    - Allow overflow turned off
- **PMI Value Add**
  - Understand heap and concurrent session requirements
  - Identify affects of persistence
  - Verify invalidation time out

16 Performance © 2002, 2003 IBM Corporation

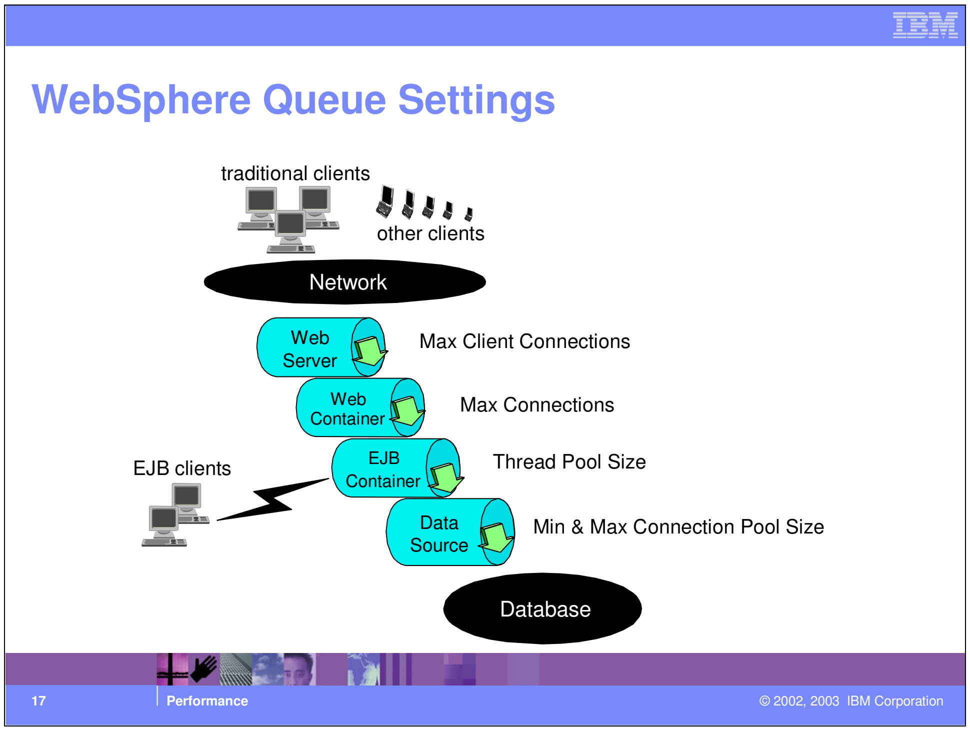
It is still common to see performance problems with HTTP Sessions. Two of the major issues are the size of the session and how long the session is being held in memory.

According to the WebSphere Performance Team, performance drops off considerably when your sessions are larger than 16k. The decline is even more dramatic if you are using persistent sessions.

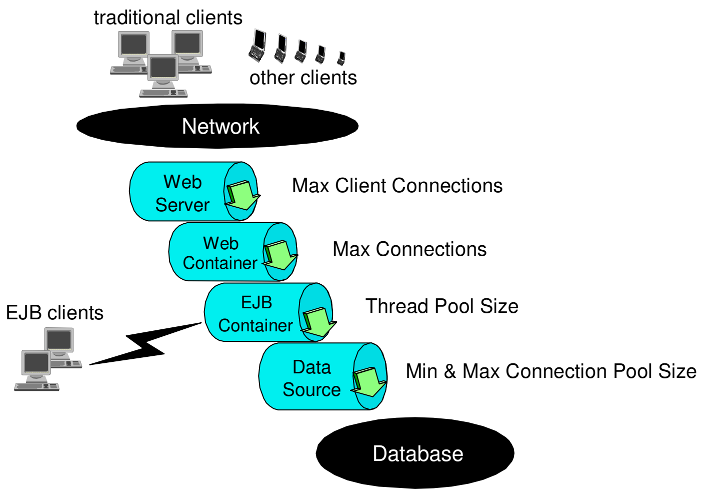
The length of time the session is held in memory is also a key factor in how much memory is being consumed by your application. If sessions in your application are hanging around for 30 minutes after a session is completed you could possibly create a memory constraint for your server.
30 minutes tends to be longer than most customers require and is the default value in WebSphere Application Server. Be sure to base the value that you set on your application's requirements.

The "Room for session failure" metric is used to store the number of sessions that are denied by the application server. Results in this column can come from one of two things: either you have disabled the "Allow for overflow" setting in Administrative console or the system is out of the physical resources to handle any more new sessions.
The "Number of sessions" metrics was part of PMI for v4. The "Serializable session object size" metric is new for V5.

The number of metrics being monitored through the PMI has been greatly increased with this version of WebSphere Application Server, therefore giving system administrators a much better view of what is happening with the application sessions.



WebSphere Queue Settings



17 Performance © 2002, 2003 IBM Corporation

You have the ability to set a maximum threshold for the following areas.

Web Server Connections

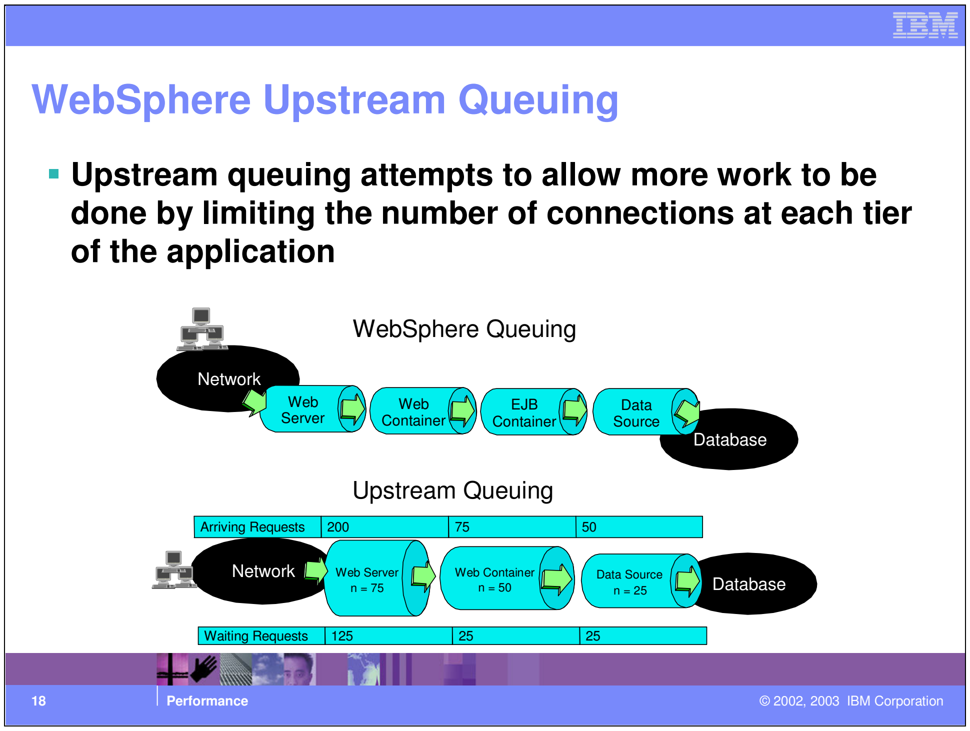
Web Container

Data Source Connection Pool

Database Agents

It is not recommended to use the Database agents as a funneling or limiting area. Your Database agents should always be greater then the total number of data sources in your environment.

If you have a Workload Management enabled environment, your Database agent number should be larger than all the data sources across all the application servers.



WebSphere Upstream Queuing

- **Upstream queuing attempts to allow more work to be done by limiting the number of connections at each tier of the application**

18 | Performance © 2002, 2003 IBM Corporation

There are several different levels to set up a funneling affect of the work being done by WebSphere.

Design the pooling system to allow as much work on the database and application server as possible without over-stressing each area.

ORB Thread pool size in the EJB container is a soft limit, whereas all other connections can be implemented as a hard limit.

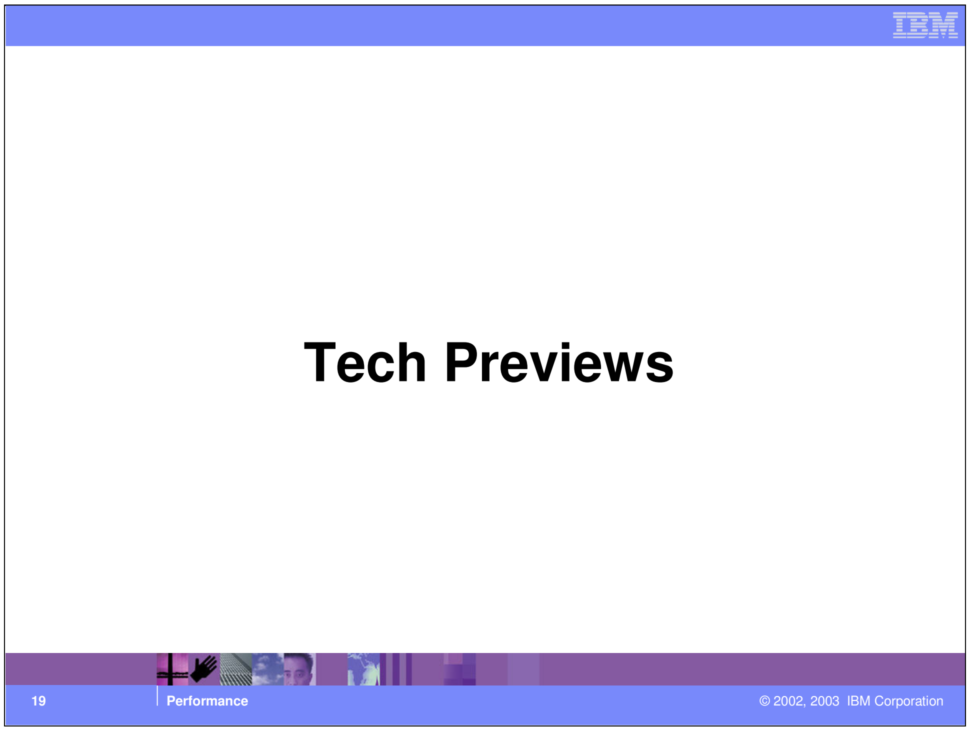
The use of the different pools can be used to funnel the work in order to allow the resources to be used by the proper processes.

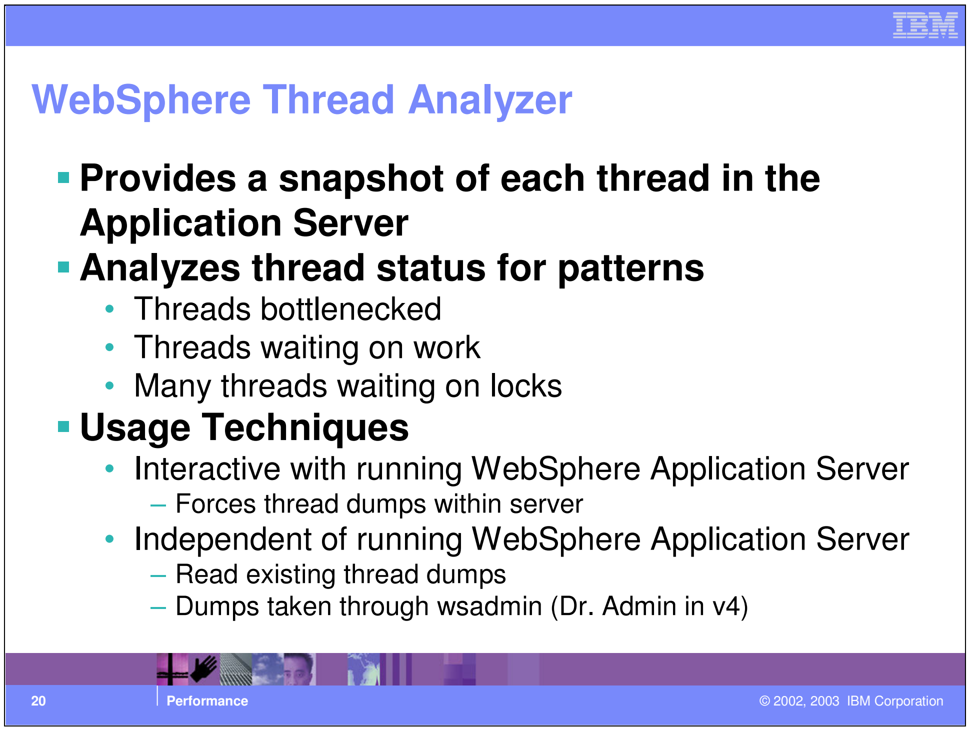
It is important not to overload any one tier because then you will have poor performance due to waiting threads. This is one of the most difficult things to do during performance

For instance, if a database is being overworked and causing slow response time, you may need to lower the number of connections. This allows those active connections the resources needed to perform the needed queries.

Most of these settings can be monitored by the Tivoli Performance Viewer and changed through the Administrative Console.

The objective is to achieve maximum throughput within the environment or solution.


Tech Previews



WebSphere Thread Analyzer

- **Provides a snapshot of each thread in the Application Server**
- **Analyzes thread status for patterns**
  - Threads bottlenecked
  - Threads waiting on work
  - Many threads waiting on locks
- **Usage Techniques**
  - Interactive with running WebSphere Application Server
    - Forces thread dumps within server
  - Independent of running WebSphere Application Server
    - Read existing thread dumps
    - Dumps taken through wsadmin (Dr. Admin in v4)

20 Performance © 2002, 2003 IBM Corporation

Thread Analyzer takes thread dumps from the JVM and analyzes them to help determine the cause of slowdowns or hangs in WebSphere

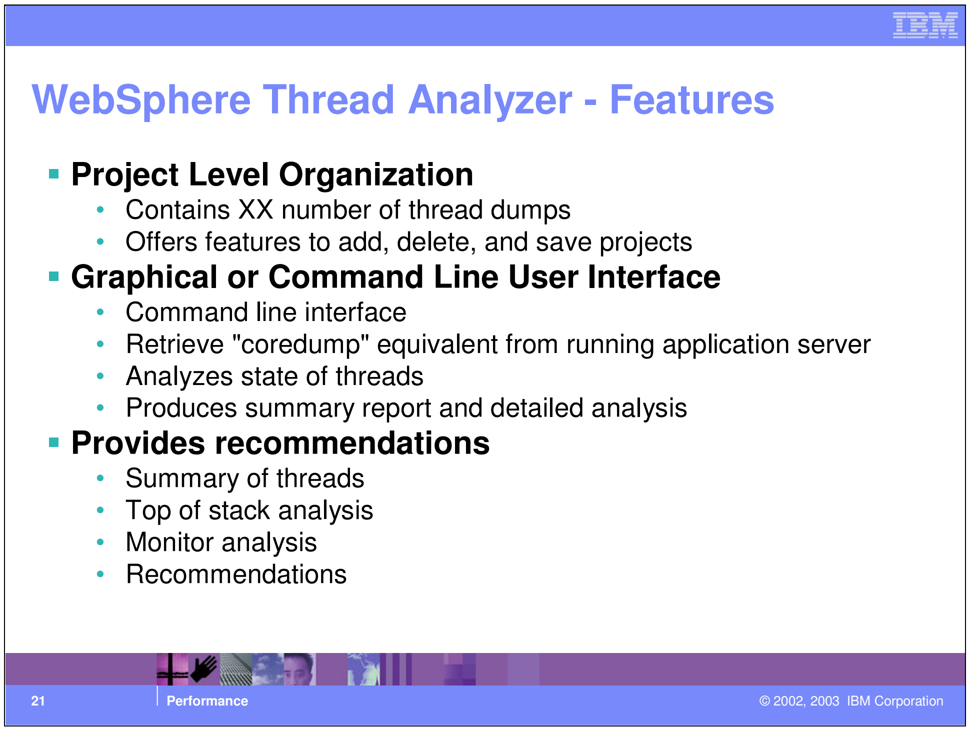
Only provides a snapshot of a single point in time, and doesn't provide application context flow

This was done manually before, but now is an automated process with a GUI interface.

Thread Analyzer can be used in both the production or run-time environment. The thread dump puts only a small performance hit on the Java process and does not bring it completely down.

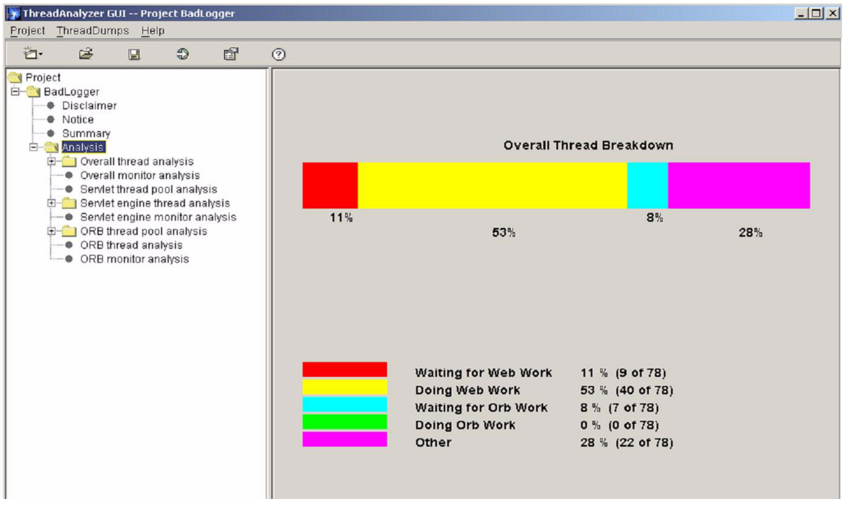
Should only be used with WebSphere - highly specialized tool with WebSphere specific knowledge.

Thread Analyzer can be useful even if it isn't installed on the deployment system, by utilizing it to analyze manually forced thread dumps from any production or test application server JVM.


WebSphere Thread Analyzer - Features

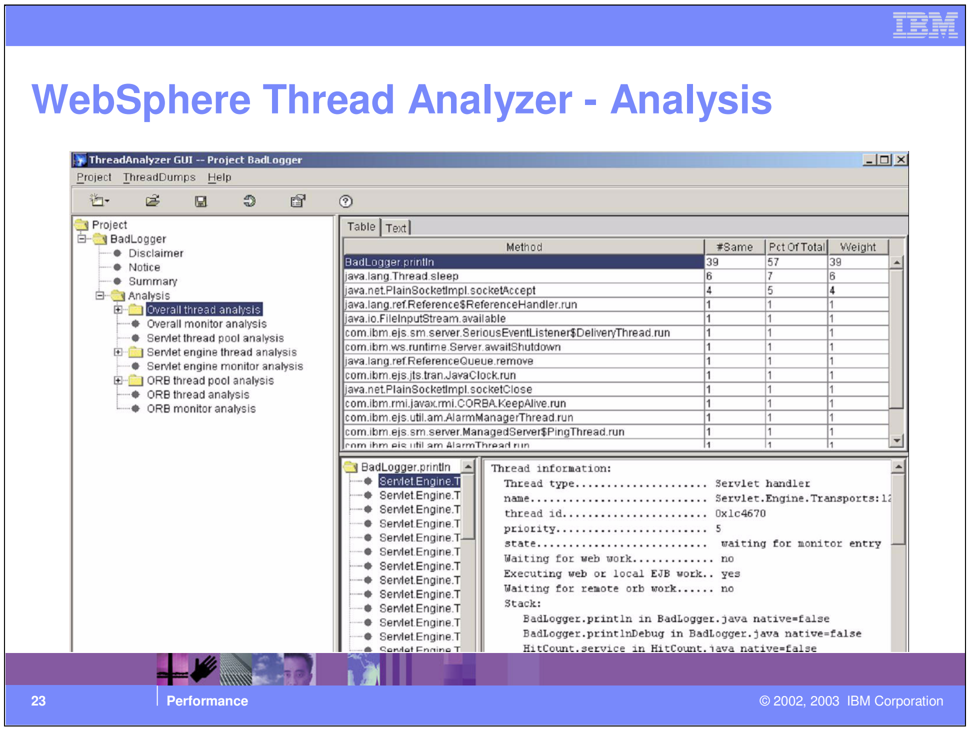
- **Project Level Organization**
  - Contains XX number of thread dumps
  - Offers features to add, delete, and save projects
- **Graphical or Command Line User Interface**
  - Command line interface
  - Retrieve "coredump" equivalent from running application server
  - Analyzes state of threads
  - Produces summary report and detailed analysis
- **Provides recommendations**
  - Summary of threads
  - Top of stack analysis
  - Monitor analysis
  - Recommendations

WebSphere Thread Analyzer - Breakdown



🟥	Waiting for Web Work	11 % (9 of 78)
🟨	Doing Web Work	53 % (40 of 78)
🟦	Waiting for Orb Work	8 % (7 of 78)
🟩	Doing Orb Work	0 % (0 of 78)
🟪	Other	28 % (22 of 78)

This screen shot shows the overall breakdown graphic of the threads in the snapshot.



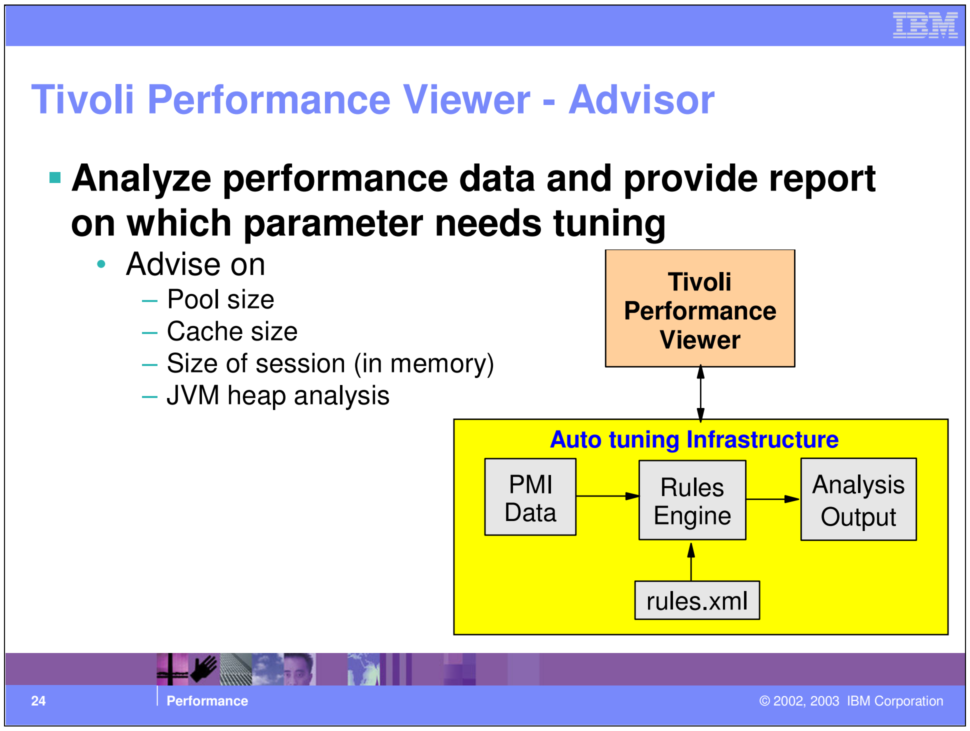
WebSphere Thread Analyzer - Analysis

Method	#Same	Pct Of Total	Weight
BadLogger.println	39	57	39
java.lang.Thread.sleep	6	7	6
java.net.PlainSocketImpl.socketAccept	4	5	4
java.lang.ref.Reference$ReferenceHandler.run	1	1	1
java.io.FileInputStream.available	1	1	1
com.ibm.ejs.sm.server.SeriousEventListener$DeliveryThread.run	1	1	1
com.ibm.ws.runtime.Server.awaitShutdown	1	1	1
java.lang.ref.ReferenceQueue.remove	1	1	1
com.ibm.ejs.jts.tran.JavaClock.run	1	1	1
java.net.PlainSocketImpl.socketClose	1	1	1
com.ibm.rmi.javax.rmi.CORBA.KeepAlive.run	1	1	1
com.ibm.ejs.util.am.AlarmManagerThread.run	1	1	1
com.ibm.ejs.sm.server.ManagedServer$PingThread.run	1	1	1
com.ibm.ejs.util.am.AlarmThread.run	1	1	1

```
Thread information:
  Thread type.................... Servlet handler
  name........................... Servlet.Engine.Transports:12
  thread id...................... 0x1c4670
  priority....................... 5
  state..........................  waiting for monitor entry
  Waiting for web work............ no
  Executing web or local EJB work.. yes
  Waiting for remote orb work...... no
  Stack:
     BadLogger.println in BadLogger.java native=false
     BadLogger.printlnDebug in BadLogger.java native=false
     HitCount.service in HitCount.java native=false
```

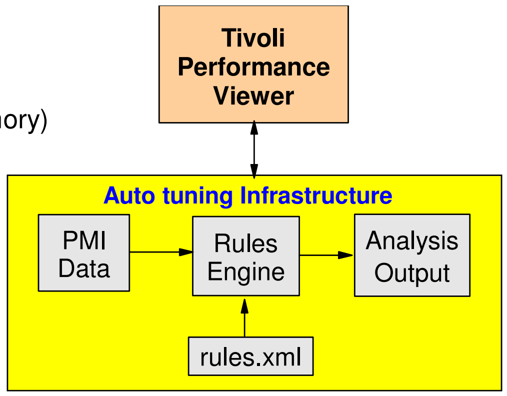
23 Performance © 2002, 2003 IBM Corporation

In this snapshot you can see that a majority of the threads are currently waiting to execute against the database, while only a few are actually doing commit work against the database. You can also see that by selecting a row in the table at the top, more detailed information is displayed in the bottom left panel. You can then select one of the thread in this panel and additional information is displayed in the panel on the bottom right.



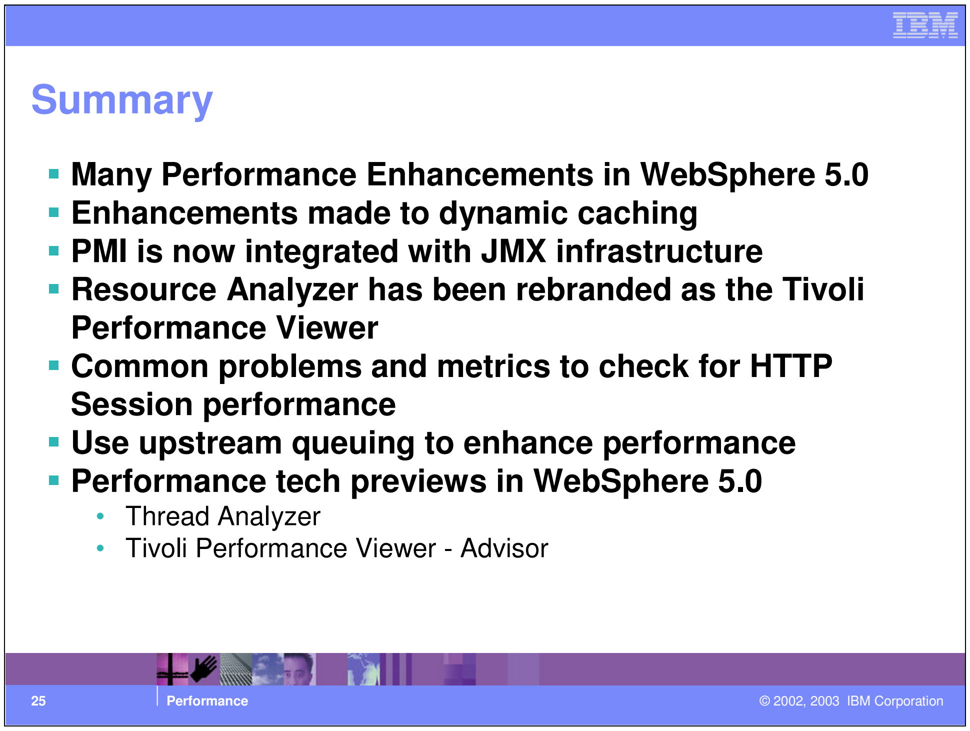
Tivoli Performance Viewer - Advisor

- Analyze performance data and provide report on which parameter needs tuning
  - Advise on
    - Pool size
    - Cache size
    - Size of session (in memory)
    - JVM heap analysis



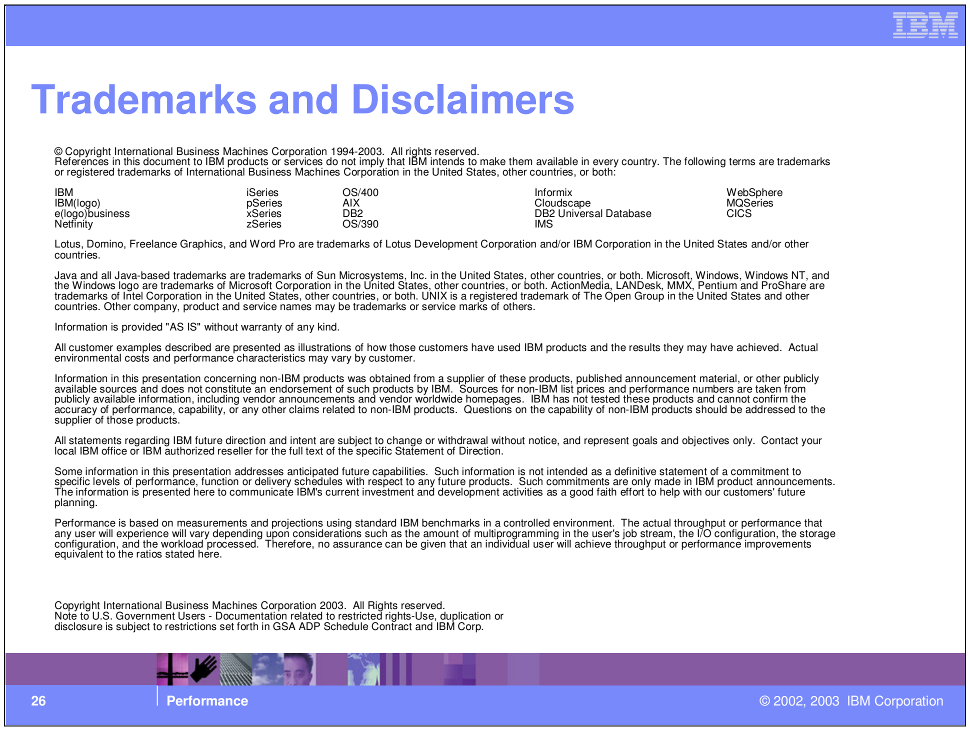
24 Performance © 2002, 2003 IBM Corporation

The rules.xml file will be determined by IBM and will provide the guidelines by which the Advisor will behave. In the future, additional measures may be taken to allow customers to define their own rules.xml file for the Rules Engine.



Summary

- **Many Performance Enhancements in WebSphere 5.0**
- **Enhancements made to dynamic caching**
- **PMI is now integrated with JMX infrastructure**
- **Resource Analyzer has been rebranded as the Tivoli Performance Viewer**
- **Common problems and metrics to check for HTTP Session performance**
- **Use upstream queuing to enhance performance**
- **Performance tech previews in WebSphere 5.0**
  - Thread Analyzer
  - Tivoli Performance Viewer - Advisor



Trademarks and Disclaimers