

Performance Monitoring and Tuning in JRun 3.0

By: Patrick Brown and Keen Haynes

Senior Consultants

Allaire Corp.

Introduction

A vital attribute of any successful Web site is the ability to serve requests rapidly and efficiently under load. To this end, it is imperative that developers, quality assurance engineers, and Web administrators monitor performance metrics and tune server parameters to enhance server operation. This is most effectively done in a staging environment with controlled load testing prior to rolling out code in a production system. This article provides a brief description of some of the tools available to assist with the monitoring of server performance. References for third-party applications are also provided.

- [Performance vs. Scalability](#)
- [Performance Metrics](#)
- [JRun 3.0 Monitors](#)
- [Platform-Specific Tools](#)
- [Tuning Parameters](#)
- [Third-Party Tools](#)
- [Conclusion](#)

Performance vs. Scalability

It is important to understand that performance does not by default imply scalability. *Webster's New World Dictionary*, Second College Edition, defines performance as "the act of ... execution, accomplishment, fulfillment, etc.," whereas *scalable* is that which belongs to "a system of grouping or classifying in a series of steps or degrees according to a standard of relative size, amount, rank, etc." As such, there are times when scalability must suffer to maximize performance, and vice versa. Typically, a Web site must balance the two in such a way that the application both performs within stated specifications and is sufficiently robust to handle expected peak loads.

Performance Metrics

Engineers and administrators should be attentive to the following set of metrics that impact JRun Server performance:

- Database query time
- Web server response time
- Database connection time
- Page execution and queue time
- Web server connections

- Thread count
- CPU and memory usage
- End-user response time and experience

JRun 3.0 Monitors

JRun 3.0 provides two useful tools for monitoring server performance: Interceptor and Metrics logging. It should be emphasized that these tools should be implemented in a staging environment under load testing conditions. Using these tools in a production environment adds additional overhead requirements on the server.

The JRun Interceptor allows for the timing of distinct method calls within a servlet or JSP. In addition to the servlet execution time, the following statistics are output:

- Total time of the method call
- Number of times the servlet called the method
- Name of the calling servlet class or method
- Name of the called method
- Associated line number within the class where the call was made

The steps required for implementing the Interceptor are described in detail in Chapter 39 of the *Developing Applications with JRun* guide in the product documentation.

Enabling the Metrics logging within JRun provides the following output:

- Threads listening for new connections
- Threads waiting for new requests
- Threads waiting to run
- Threads currently running
- Total worker thread count
- Requests delayed as a result of concurrency
- Dropped requests
- Handled requests
- Time spent servicing requests
- Time spent in delay state
- Bytes read from request
- Bytes written for response
- Free memory in heap
- Total memory in heap
- Number of active sessions
- Number of sessions in memory

Metrics logging may be enabled using the JRun Management Console (JMC) or via the `global.properties` file (for all servers) or the `local.properties` files (for individual servers). From within the JMC, select "Log File Settings" within the desired server, then check the "Metrics" box under the "Logging Level" setting, as shown in the figure below.

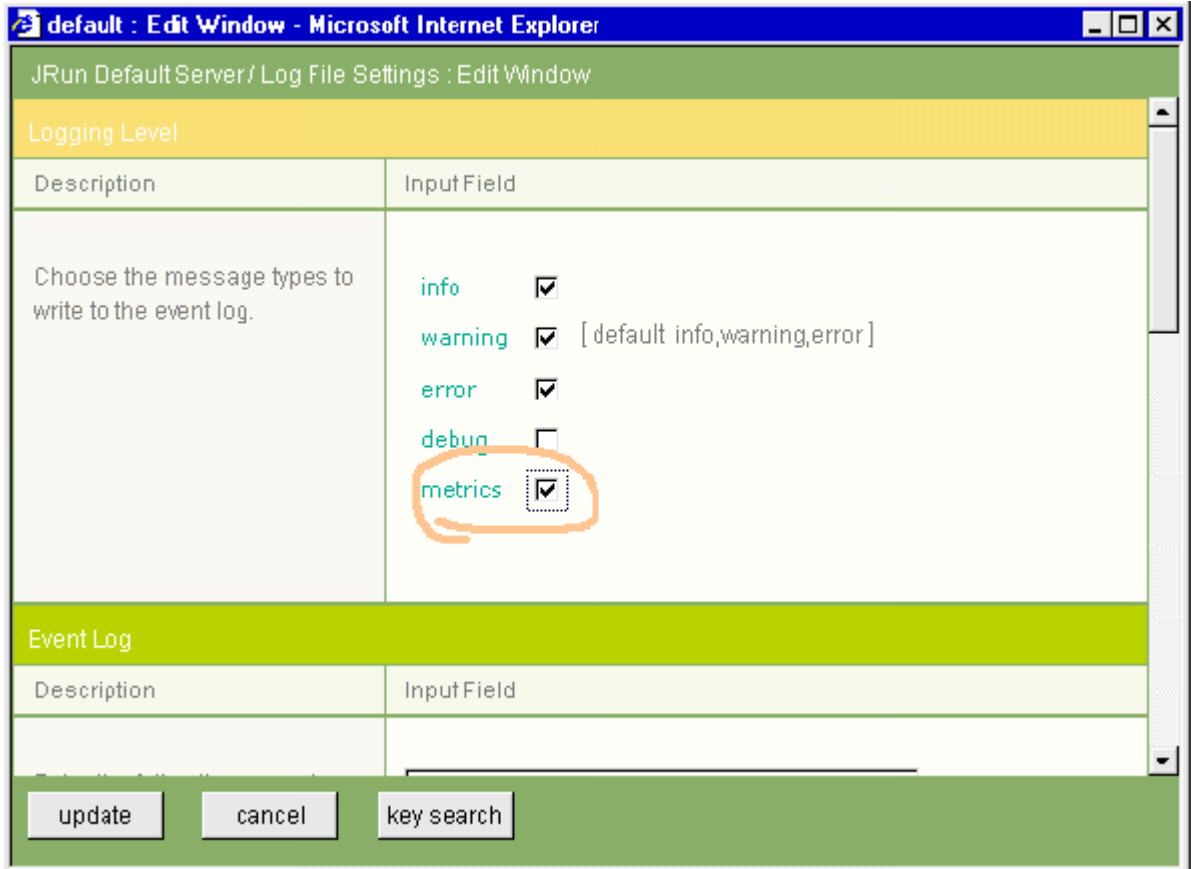


Figure 1: Enable metrics logging using the JMC.

The temporal sampling granularity as well as output formats and selection or deselection of individual metrics logged may be set via the `global.properties/local.properties` files. Refer to Chapter 36 in the *Developing Applications with JRun* guide for additional information.

Platform-Specific Tools

Both Solaris and Windows NT provide OS-level monitoring tools that may be used in measuring JRun server performance. In Solaris, the following commands will display useful information:

- `prtdiag -v` : system status information
- `uptime` : current load average
- `top` : dynamic load monitor
- `vmstat` : system memory, CPU utilization

- iostat : I/O statistics
- truss : system execution calls

Inspecting the output of each or all of these system commands gives substantial insight into memory use, CPU utilization, and system behavior. More information regarding the syntax and output information can be found in the product documentation.

In a Windows NT environment, common tools used in monitoring server and system performance include the following:

- Performance Monitor
- Task Manager
- Microsoft SQL Server Enterprise Manager
- Oracle Performance Monitor

Refer to the Windows Help and the respective Microsoft SQL and Oracle Help menus for further information.

Tuning Parameters

Perhaps the three most important tuning parameters within JRun are the following:

- jcp.endpoint.main.min.threads
- jcp.endpoint.main.active.threads
- jcp.endpoint.main.max.threads

These parameters are set in the global.properties (for all servers) and local.properties (for individual servers) files. They may also be configured via the JMC in the Web server setting within the individual servers, as shown below:

JRun Default Server > External Web Server

These settings allow this server to connect to an external Web server. JRun is commonly configured to enhance various third-party Web servers with Java Servlet & JavaServer Pages support. Supported third-party Web servers include:

- Microsoft Persona Web Server / Internet Information Server
- Netscape FastTrack / Enterprise / iPlanet Servers
- Apache Server
- O'Reilly WebSite Pro
- Zeus Web Server

Name	Value	Summary
External Web Server Address	*	Address of external Web servers that can connect to this server
Listening Address	127.0.0.1	Socket address used to listen for connections from external Web servers
Listening Port	5000	Socket port used to listen for connections from external Web servers
Idle Thread Timeout	300	Number of seconds threads remain idle before being destroyed
Minimum Thread Count	10	Minimum number of handler threads in pool
Maximum Active Requests	300	Number of concurrent requests accepted before new requests are queued
Maximum Concurrent Requests	1000	Number of concurrent requests accepted before new requests are denied
Connection Module	on	The current status of this connection module

edit | connector wizard

Figure 2: Configuring tuning parameters via the JMC.

The `jcp.endpoint.main.min.threads` setting determines the number of handler threads created at startup. There may be a slight performance improvement realized if this number is set to something greater than default (1) in cases where there are a number of threads required immediately in an application.

The `jcp.endpoint.main.active.threads` setting determines the maximum number of simultaneous worker threads. When the number of requests exceeds this value, they are placed into the queue. By default, this value is set to 100. Depending upon the application and the server hardware, this value may be set substantially higher or lower for optimal server behavior.

The `jcp.endpoint.main.max.threads` setting determines the maximum number of threads, both working and queue, allowed. The default value for this setting is 1000. Exceeding this value will result in the message "JRun is busy, please try again later" being sent to the client. There is less overhead required for managing queue threads than there is in managing worker threads.

Obtaining the optimal balance in the settings for `jcp.endpoint.main.active.threads` and `jcp.endpoint.main.max.threads` is best done via load testing in a staging environment.

Third-Party Tools

In addition to the performance-monitoring tools provided as part of the JRun Server application and the previously mentioned OS-specific monitoring tools, there are a number of third-party performance monitoring tools available, including the following:

HAT - <http://java.sun.com/people/billf/heap/>

JProbe - <http://www.klgroup.com/>

jProf - <http://starship.python.net/crew/garyp/jProf.html>

Optimizelt - <http://www.optimizeit.com/>

The inclusion of these tools in this article should not be considered an endorsement by the authors or Allaire Corporation.

Conclusion

This article has provided an overview of the tools available for gaining insight into the performance of a JRun-served application, both at the server and at the operating system levels. The *JRun Setup Guide* and *Developing Applications with JRun* product documentation provide detailed information regarding the implementation of Interceptor and Metrics logging. Further guidance on JRun 3.0 Performance and Scalability Tuning is provided in the articles "[JRun 3.0 Performance and Scalability Tuning Part 1](#)", and [JRun 3.0 Performance and Scalability Tuning Part 2](#) by JRun QA Engineer Scott Stirling. It is important that developers, quality assurance specialists, and Web administrators understand the importance of monitoring server performance during load testing of their applications to help to identify bottlenecks and to optimize their servers prior to going live in production.