BEA WebLogic JRockit™ SDK

Using the JRockit Runtime Analyzer

Version 8.1 Service Pack 2 (Internal)
December 2003

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

Using the JRockit Runtime Analyzer

Using the JRockit Runtime Analyzer

This product is provided "as-is," without any expressed or implied warranties or support by BEA Systems, Inc. This product, which may or may not become an officially supported product from BEA Systems, may contain errors and/or inaccuracies. Use of this product is left solely upon the discretion of the user without any endorsement from BEA Systems. The JRA functionality may or may not be available in future JRockit versions. Questions and problems may be reported via online WebLogic JRockit newsgroups at `http://newsgroups.bea.com`.

The JRockit Runtime Analyzer (JRA) is an internal tool used by the WebLogic JRockit development team to analyze runtime performance of WebLogic JRockit and Java applications running on WebLogic JRockit. JRA is now being made available to external WebLogic JRockit users to give them the same analytical capabilities as BEA's developers.

JRA provides a wealth of information on the internals of WebLogic JRockit that development teams will find helpful for understanding and optimizing runtime performance. Some of these metrics are interesting to Java developers using WebLogic JRockit as their runtime, as well.

The JRockit Runtime Analyzer consists of two parts:

- One part of the Analyzer runs inside WebLogic JRockit and records information about the JVM and the Java application currently running thereon. This part of JRA saves this information to a "recording" file which is then opened in the other part: the analyzer tool.
- This other part of the tool is a regular Java application that displays the information contained in the JRA recording file.

This document described how to use JRA. It includes information on the following subjects:

Compatibility

The JRockit Runtime Analyzer is compatible with WebLogic JRockit 8.1 JVM, service pack 1.

Help Us Improve WebLogic JRockit

BEA Systems is making JRA available to external users so that you can help us make WebLogic JRockit JVM a better virtual machine. JRA was created to provide an easy way to capture information about WebLogic JRockit when it is running in a deployment environment. JRA is designed to induce a minimal overhead when enabled, and no overhead at all when not enabled. The typical overhead we have seen when it is enabled is around 1-2%, which is in most cases negligible.

Because of this, JRA can be used to capture a true picture of the running system. The data that is recorded is actually what is happening and not a side effect of having the tool enabled. JRA is able to do this by being very tightly integrated into the JVM.

The results of these recordings are often used within the engineering team to guide future improvements to WebLogic JRockit. You can help us with this effort by sending the JRA recordings of your J2SE or J2EE application running with WebLogic JRockit, as described in this procedure:

1. Create an email and attach an `.xml` of the recording to an email. If this is a large file, you might need to zip it up.

2. In the email, include brief description of your application so that we know what we are looking at.
3. Send the email to **jrockit-improve@bea.com**

How Will BEA Use These Recordings

WebLogic JRockit is already the fastest server-side JVM available, and to stay that way BEA is committed to find ways to improve its performance. The JRA is one of those ways.

The recordings will be analyzed by the development team using the JRA. We will look at the profile of your application and try to find out how we can improve WebLogic JRockit in the future to be able to run your application faster and more efficiently.

Note: BEA Systems will not disclose, share, or discuss any information related to the recording files with anyone outside of BEA.

How Can You Use the Recordings?

You can download the JRA and use it yourself to analyze the recordings. To acquire the JRA, simply download the file (`jra81sp1.zip`) from the Download JRockit Runtime Analyzer Tool link on:

`http://dev2dev.bea.com/products/wljrockit81/analyzer.jsp`

Note: Please read the information on this page carefully before downloading the JRA. Pay particular attention to the Disclaimer.

This file contains the `jraRecordingStarter.jar` and `RuntimeAnalyzer.jar` files, along with two utility files needed by `RuntimeAnalyzer.jar`. After downloading the zip file, unzip it to a directory of your choosing and run it from there.

While a lot of the information provided by the tool is only useful to the JVM engineers, you might find some of the information useful for improving the performance of your Java application. You can also gain some insight into the complicated inner workings of a modern day state of the art JVM.

Creating a Recording

To create a JRA recording with WebLogic JRockit, use the `JraRecordingStarter` utility and follow these simple steps:

1. Start your Java application with WebLogic JRockit; include the `-Xmanagement` option in the command line.

2. Initiate the recording by entering this command:

```
> java -jar JraRecordingStarter.jar <server> <port> <filename>
  <recording time>
```

The following same steps explain the recording process in greater detail:

1. Start your Java application with WebLogic JRockit as usual, including the command line option `-Xmanagement`. This tells WebLogic JRockit to open a port and listen to commands from a management client (read more about this in the Using the WebLogic JRockit Management Console).

   Note: The JRA recording functionality is only available in WebLogic JRockit 8.1SP1.

2. Make sure that your application is up and running and under load. You must always make sure that the application is under stress when making a recording, since the data captured from an idle application will not show where there is room for improvements.

3. Use the following command to initiate a recording:

```
> java -jar JraRecordingStarter.jar <server> <port> <filename>
  <recording time>
```

   Where the arguments are:

   – `<server>` is the host name of the machine where your application is running (usually `localhost`).

   – `<port>` is the port that WebLogic JRockit is listening to (usually 7090)

   – `<filename>` is the name of the file you want to save the recording to (for example jrarecording.xml). This is the name of the file on the host where the recording is done (`<server>` above). The file will be created in the current directory of the WebLogic JRockit process. The file will be overwritten if it exits.

   – `<recording time>` is the length of the recording in seconds (a good length is five minutes; this is, 300 seconds).

   For example:

```
> java -jar JraRecordingStarter.jar localhost 7090 jrarecording.xml 300
```

After the recording has commenced, you will see a message printed by WebLogic JRockit that the recording has started. You will see another message halfway through the recording and a final message when the recording is finished. After the final message is printed you can shut down the application.

Looking at a Recording

Use the JRA to take a look at the recording you just created.

1. Start the tool with `java -jar RuntimeAnalyzer.jar`. This will open a GUI application.
2. Open the file created above (jrarecording.xml) in the tool and use the different tabs to see information about the application you just ran.
   - The General tab provides a collection of general data such as thread, allocation, exception and memory usage statistics. It also displays the options used to start WebLogic JRockit.
   - The Methods tab has a list of the *top hot methods* during the recording. This data is collected by periodically sampling the running threads in the JVM and looking at which method they are executing. Methods prefixed by `jvm#` are native methods in the JVM. By clicking on a method you can see it's predecessors and successors to the right.
   - The GC tab has detailed information about each garbage collection event that has occurred. Graphs show the heap usage before and after each GC as well as pause times and number of `java.lang.ref.*` objects discovered. There is also a view showing which call trace execute the allocation that triggered a GC to happen.
   - Finally, the Optimizations tab displays the methods that were optimized by the adaptive optimization system in WebLogic JRockit. The methods are displayed in chronological order.

Submit Your Questions

Since this is an internal tool and not a supported product, we cannot make any guarantees about the correctness of the data we show, or the stability of the product when using JRA. Neither do we have the ability to provide support or answer questions about JRA. The tool is provided as is for you convenience and to help us improve WebLogic JRockit. The JRA functionality may or may not be available in future WebLogic JRockit versions. If you have any questions you are welcome to share them in the WebLogic JRockit newsgroups which are monitored by the engineering team.

Frequently Asked Questions

I am getting very few samples, is there anything I can change?

Try re-running WebLogic JRockit with the `-XXjranonativesamples` parameter and see if you get more samples now.

Is the overhead really as small as 1-2%?

Yes, for the applications that we have measured this has been the case. However the overhead can momentarily be larger when the `.xml` file is written to disk.

My server is under such heavy load that the JraRecordingStarter consistently fails to connect to it. Is there another way to start a recording?

Yes. There are command line parameters to WebLogic JRockit to start a recording after a specified time. These options are a bit more inflexible, but work better when the server is under heavy load:

- `-XXjradelay=<seconds>` - A JRA recording will start after the specified number of seconds after the JVM is started.
- `-XXjrarecordingtime=<seconds>` - Specifies how long the recording will be.
- `-XXjrafilename=<filename>` - Specify the filename for the recording file.