BEA JRockit

Introduction to BEA JRockit JDK

Version 5.0 Service Pack 2
June 2005

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

Overview of the BEA JRockit JDK

Understanding the BEA JRockit JVM

What's in BEA JRockit JDK

New Features in BEA JRockit JDK

CHAPTER 1

Overview of the BEA JRockit JDK

The BEA JRockit JDK provides tools, utilities, and a complete runtime environment for developing and running applications using the Java programming language. The BEA JRockit JDK also includes the BEA JRockit Java Virtual Machine (JVM). The BEA JRockit JVM is developed and optimized for Intel architectures to ensure reliability, scalability, and manageability for Java applications.

This section contains information on the following subjects:

What is BEA JRockit?

The BEA JRockit JVM (see the Compatibility Between Releases for information on BEA JRockit versions) is a high performance JVM developed to ensure reliability, scalability, manageability, and flexibility for Java applications. The BEA JRockit JVM delivers a new level of performance for Java applications deployed on Intel 32-bit (Xeon) and 64-bit (Xeon and Itanium) architectures at significantly lower costs to the enterprise. Furthermore, it is the only enterprise-class JVM optimized for Intel architectures, providing seamless interoperability across multiple hardware and operating system configurations. BEA JRockit JVM makes it possible to gain optimal performance for your Java applications when running it on either the Windows or Linux operating system platforms on either 32-bit or 64-bit architectures.

For more information on JVMs in general, see the Introduction to the JVM specification at:

http://java.sun.com/docs/books/vmspec/2nd-edition/html/Introduction.doc.html#3057

About the JDK

The BEA JRockit JVM is one component of the BEA JRockit Java development kit (JDK). Along with the BEA JRockit JVM, this kit is comprised of the Java Runtime Environment (JRE), which contains the JVM and Java class libraries (as specified by the Java 2 Platform API Specification), along with a set of development tools, such a compiler, a debugger, and so on. For more information about the contents of the BEA JRockit JDK, please refer to What's in BEA JRockit JDK

Compatibility Information

For information on how BEA JRockit is updated, see Compatibility Between Releases at:

http://e-docs.bea.com/jrockit/jrdocs/suppPlat/prodsupp.html#999010

What Platforms Does BEA JRockit Support?

BEA JRockit JDK is certified to be compatible with J2SE 5.0. For a complete list of platforms that BEA JRockit supports, see BEA JRockit 5.0 Supported Configurations at:

http://e-docs.bea.com/jrockit/jrdocs/suppPlat/supp50.html

BEA JRockit JDK Support

You are entitled to support on the JVM if you have a support agreement with BEA.

CHAPTER 2

Understanding the BEA JRockit JVM

The BEA JRockit JVM has a number of important features that separate it from other JVMs on the market today. This section provides high-level descriptions of some of the more critical features to help you better understand what they can do. The BEA JRockit JVM consists of the following parts:

- The Java Development Kit
- The Java Runtime Environment
- The JRockit Management Console
- Code Generation and Optimization
- Memory Management (Garbage Collection)

The Java Development Kit

The BEA JRockit Java development kit (JDK) is very similar to the Sun JDK, except that it includes a new JRE with the BEA JRockit JVM and some changes to the implementation of the Java class libraries (however, all of the class libraries have the same behavior in the BEA JRockit JDK as in the Sun JDK).

The Java Runtime Environment

The BEA JRockit implementation of the Java 2 runtime environment (JRE) includes the BEA JRockit JVM, class libraries, and other files that support the execution of applications written in the Java programming language.

The JRockit Management Console

The Management Console connects to the BEA JRockit JVM and provides real-time information about server behavior and resource availability, such as memory usage and profiling information. This gives you a powerful way of retrieving real-time profile data about your application.

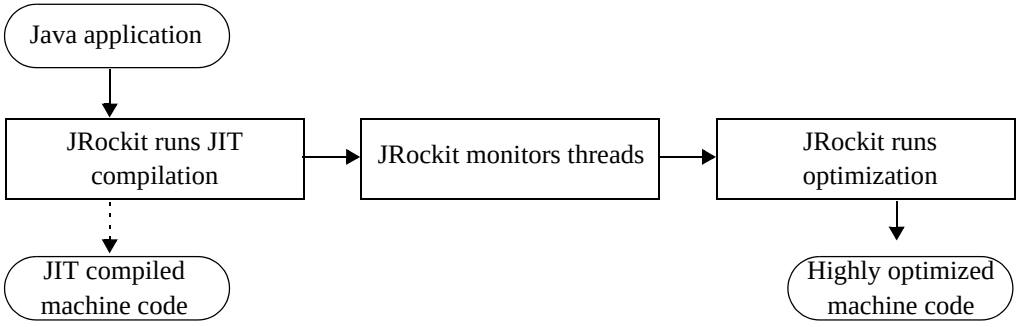
The Management Console provides a unique advantage when deploying a commercial Java solution because it gives you greater control of the complex set of interrelated variables that may affect your application in production. Administrators can monitor the BEA JRockit JVM operating characteristics and the Java application, and be automatically notified of changes in resource availability or operating characteristics as they occur. Based on this information, administrators can identify bottlenecks in performance and change operating and environmental parameters to optimize performance and availability.

For more information on the Management Console, please refer to Using the JRockit JVM Management Console.

Code Generation and Optimization

The code generator in JRockit is running in the background during the entire run of your Java application to automatically adapt the code to run its best. The code generator works in three steps as described in Figure 2-1.

Figure 2-1 Overview of Road to Optimized Code for Your Java Application



1. The first step of code generation is the Just In Time (JIT) compilation. This compilation allows your Java application to get started and run while the code that is generated is not highly optimized for the platform.

Compilation time is part of application execution time, thus compiling all of the methods with all available optimizations will negatively impact application performance, that is why the first step is a JIT compilation, and therefore, JRockit does not fully optimize all methods at start-up.

2. During the second phase, JRockit monitors all threads, in run-time, to find out which methods are taking the longest to run. The monitoring is done in the background in a special sample thread, which means that the performance of your application is not disturbed.

   The sample thread identifies what each thread is executing and notes some of the execution history. This information is tracked for all the methods and when it is perceived that a method is experiencing heavy use—in other words, is “hot”—that method is marked for optimization. Usually, a flurry of such optimization opportunities occur in the application's early run stages, with the rate slowing down as execution continues.

3. During the third phase, JRockit runs an optimization round of the methods that JRockit has perceived as the most used—“hot”—methods. This optimization is run in the background and does not disturb the running application.

Memory Management (Garbage Collection)

Memory management relies on effective “garbage collection,” the process of clearing dead objects from the heap, thus releasing that space for new objects. BEA JRockit uses a dynamic garbage collector that is based upon one of two priorities that you set: application throughput or duration of the pause times caused by garbage collection. The dynamic garbage collector uses predefined heuristics to determine, in runtime, which garbage collection algorithm to use for each application.

In some instances, dynamic garbage collection might not be the most effective way to recycle memory. In those cases, BEA JRockit also provides a number of “static” garbage collectors that can be started by specifying the actual collector (`-Xgc:<collectorName>`) at startup.

For information on selecting and using garbage collectors, see Using the BEA JRockit Memory Management System.

CHAPTER 3

What's in BEA JRockit JDK

BEA JRockit JDK is very similar, in the file layout, to the Sun JDK, except that it includes a new JRE with the BEA JRockit JVM and some changes to the Java class libraries (however, all of the class libraries have the same behavior in BEA JRockit as in the Sun JDK). For a more detailed description of the differences between the two JDKs, please refer to File Differences Between BEA JRockit JDK and Sun HotSpot JDK.

This section describes the contents of the BEA JRockit JDK and compares a BEA JRockit JDK installation to a comparable Sun JDK installation. It includes information on the following subjects:

- JDK Contents
- File Differences Between BEA JRockit JDK and Sun HotSpot JDK
- BEA JRockit Command Line Options

JDK Contents

This section describes the various components that make up the BEA JRockit JDK. It also identifies the folder in which these components reside.

Development Tools

Found in: `/bin`

Development tools and utilities help you develop, execute, debug, and document programs written in the Java programming language. The BEA JRockit JDK includes the standard tools

commonly distributed with the typical Java JDKs. While most of these are standard JDK tools and are proven to work well with Java development projects, you are free to use any other third party tools, compilers, debuggers, IDEs, and so on that might work best in your situation. The tools included with BEA JRockit JDK are:

- `Javac` compiler
- `Jdb` debugger
- `Javadoc`, which is used to create an HTML documentation site for the JVM API

For more information on these tools, please refer to Sun Microsystem's Java™ 2 JDK at:

`http://java.sun.com/j2se/1.5.0/index.jsp`

Additional Libraries

Found in: `/lib`

Additional class libraries and support files required by the development tools.

C Header Files

Found in: `/include`

Header files that support native-code programming using the Java Native Interface and the Java Virtual Machine Tools Interface and other functionality of the Java 2 Platform.

The Management Console

Found in: `/console`

The BEA JRockit Management Console is used to monitor and control running instances of the JRockit JVM. It provides real-time information about the running application's characteristics, which can be used both during development-for example, to find where in an application's life cycle it consumes more memory-and in a deployed environment-for example, to monitor the system health of a running application server.

Java Runtime Environment (JRE)

Found in: `/jre`

The BEA JRockit implementation of the Java 2 runtime environment for use by the JDK. The runtime environment includes the BEA JRockit JVM, class libraries, and other files that support the execution of programs written in Java.

Java Virtual Machine

By definition, the JVM is BEA JRockit JVM, as described in this documentation set.

Standard J2SE JRE Features

In addition to JRE components specific to BEA JRockit JDK, the JRE also contains components found in the Sun implementation of the JRE. For a complete list of the standard J2SE JRE features, see the Sun documentation:

http://java.sun.com/j2se/1.5.0/docs/index.html

File Differences Between BEA JRockit JDK and Sun HotSpot JDK

Table 3-1 and Table 3-2 list the differences between BEA JRockit JDK and Sun Microsystems' HotSpot JDK. The tables list, by component and operating system (OS/arch), files that either exist in HotSpot or BEA JRockit

Note: $ARCH = i386 on Linux IA32, but $ARCH = x86_64 on Linux Itanium and mydir[/*] means mydir and mydir/*

Table 3-1 Files Used Only in Sun JDK; Not used by BEA JRockit JDK

Component	OS/arch	Files	Notes
Sun Hotspot VM support files	Windows IA32 Windows Itanium	jre/bin/client[/*] jre/bin/msvcrt.dll jre/bin/hpi.dll jre/bin/nio.dll jre/bin/server/* jre/bin/zip.dll jre/lib/classlist	Windows IA32 only Windows IA32 only
	Linux IA32	bin/jinfo bin/jmap bin/jsadebugd bin/jstack man/*/man1/jinfo.1 man/*/man1/jstack.1 man/*/man1/jsadebugd.1 man/*/man1/jmap.1	
	Linux IA32 Linux Itanium linux x64	jre/lib/i386/client/* jre/lib/$ARCH/libnio.so jre/lib/$ARCH/libzip.so jre/lib/$ARCH/server/* jre/lib/classlist	Linux IA32 only Linux IA32 only

Table 3-2 Files Only in BEA JRockit JDK; Not Used or Supported by Sun JDK

Component	O/S	Files	Notes
BEA JRockit Management Console	Linux IA32 Linux Itanium Windows IA32 Windows Itanium	bin/console[.exe] bin/jrcc[.exe] console/* jre/lib/managementserver.jar	
BEA JRockit Management API	Windows IA32 Windows Itanium Linux IA32 Linux Itanium	jre/lib/managementapi.jar	

Table 3-2 Files Only in BEA JRockit JDK; Not Used or Supported by Sun JDK

Component	O/S	Files	Notes

BEA JRockit JVM support files	Windows IA32
	Windows Itanium
	Linux IA32
	Linux Itanium
	linuxX86_64
		`bin/jrcmd[.exe]`
		`jre/bin/jrcmd[.exe]`
		`jre/jrockit.license`
		`jre/lib/jrockit.jar`	

	Windows IA32
	Windows Itanium
		`jre/bin/dbghelp.dll`
		`jre/binjripc.dll`
		`jre/bin/jrockit/*`
		`jre/bin/msvcr71.dll`
		`jre/bin/perfctrs.dll`
		`jre/bin/perfctrsinst.exe`
		`jre/lib/perfctrs.h`
		`jre/lib/perfctrs.ini`
			`Windows IA32 only`

	Linux IA32
	Linux Itanium
	linuxX86_64
		`jre/lib/<arch>/jrockit/*`
		`jre/lib/<arch>/libjripc.se`
		`jre/lib/ia64/librtm.so`
			`Linux Itanium only`

BEA JRockit Command Line Options

BEA JRockit configuration and tuning parameters are set by using specific command line options, which you can enter either along with the start-up command or include in a start-up script. These options are discussed in the relevant sections of the BEA JRockit documentation set; that is:

- Starting and Configuring BEA JRockit JVM, at:

  `http://edocs.bea.com/wljrockit/docs50/userguide/start.html#1015884`

- Using the BEA JRockit Memory Management System, at:

  `http://edocs.bea.com/wljrockit/docs50/userguide/memman.html#1013105`

- Tuning the JRockit JVM, at:

  `http://edocs.bea.com/wljrockit/docs50/tuning/index.html`

You can also find an alphabetical list of all BEA JRockit command line options at:

- BEA JRockit JDK Command Line Options by Name, at:

  `http://edocs.bea.com/wljrockit/docs50/options.html`

CHAPTER 4

New Features in BEA JRockit JDK

BEA JRockit JDK represents a incremental step up from previous version of this JDK. BEA JRockit contains many new features designed to improve performance and streamline operation. Note that some features provided in this version of BEA JRockit JDK are non-standard or "experimental." These features, while tested to work with this version of the JDK, are provided as-is and might change at any time.

This section includes information on the following subjects:

Performance Enhancements

- Support for Large Pages on x86.
  - On Linux it is called Huge Pages. Read documentation about huge pages in kernal source: Documentation/vm/hugetlbpage.txt
  - On Windows it is called to lock pages. Read more about lock pages here: http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/ServerHelp/e72dcdf6-fe17-49dd-a382-02baad31a137.mspx
- Improved register allocation with better result on optimized code and faster results on just-in-time compiled code, both for normal and for debug code.
- Improved code selection on x86 platforms.