Java 1.4 Virtual Machine Options

Java > Unsupported

2005-04-29

Apple Inc.
© 2004, 2005 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Carbon, Mac, Mac OS, and Pages are trademarks of Apple Inc., registered in the United States and other countries.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction **Introduction to Java 1.4 Virtual Machine Options 5**

Chapter 1 **Option Flags and Settings 7**

General Options 7
Mac OS X-Specific 8
Heap Size 8
Garbage Collection: Memory Usage 9
Garbage Collection: General Settings 9
Compilation 10
Threading 10
Sharing 11

Document Revision History 13

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

Introduction to Java 1.4 Virtual Machine Options

The standard Java HotSpot VM options are available with the Mac OS X Java VM. In addition to the standard options, many nonstandard (`-X` and `-XX`) options are also available. These, and notable exceptions, are listed in "Option Flags and Settings" (page 7).

The Options are grouped with those of similar functionality.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

Option Flags and Settings

Note: Except where specified, any time *bytes* are specified as a parameter in an option, you may also designate kilobytes or megabytes by using the letter k or m respectivly. (You may use either uppercase or lowercase letters.) For example, the following would all be equivalent values for *bytes*:4194301, 4096k, 4096K, 4m, and 4M

General Options

`-server`

There is not a distinct server VM for Mac OS X. Although you may use the `-server` flag when invoking `java`, it does not start up a different VM, instead it starts the client VM that has been tuned for use in a server environment. This tuning includes:

- Using a different class list for the shared archive generation that does not include the GUI classes.
- Increasing the size of the Java heap.
- Increasing the size of the Eden generation.
- Turns on thread local Eden garbage collection.

`-X`

Displays a brief description of the nonstandard VM options.

`-Xbootclasspath:`*`path`*

Specifies a list of directories and JAR and ZIP archives to search for boot class files in. Seperate multiple entries with colons (:).

`-Xfuture`

Performs strict format-checking of class files. This option enforces a tighter conformance to the class file specification than the default, which is based on the standard in Java 1.1.x. You should test your code with this flag to ensure functionality in future versions of Java that may enforce stricter class file format-checking.

`-Xprof`

Sends detailed profiling data of the running program to standard output. This option should not be used in production code.

`-Xrs`

Same as `-XX:+ReduceSignalUsage`.

`-XX:+ MaxFDLimit`

Increases the file descriptor limit to the maximum.

`-XX:MaxDirectMemorySize=`*`size in megabytes`*

The maximum memory that can be allocated using NIO direct buffers. The default is 64 (64M).

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

`-XX:-PrintJavaStackAtFatalState`

By default Java backtraces are generated when a crash occurs in native code. Turn this option off if you are seeing crashes in your Java error reporting.

`-XX:+ReduceSignalUsage`

Normally, Java responds to `SIGHUP`, `SIGINT`, and `SIGTERM` signals. With this option, Java ignores these signals; you need to implement handlers for them in native code as appropriate. Also, implement any relevent shutdown procedure from `System.exit()`.

`-XX:ReservedCodeCacheSize=`*`size in megabytes`*

Sets the maximum code cache size. The default is 32 (32M).

Mac OS X-Specific

`-Xdock:icon=`*`pathToIconFile`*

Sets the icon displayed in the Dock. By default Mac OS X displays a generic Java icon unless you specify otherwise. You only need to use this for applications launched from the command line or from a JAR file. Double-clickable application bundles designate their icon in the `Info.plist` file.

`-Xdock:name=`*`applicationName`*

Sets the name to display in the Dock and in the menu bar. By default Mac OS X displays the fully qualified name of the main class unless you specify otherwise. You need to use this only for applications launched from the command line or from a JAR file. Double-clickable application bundles get the appropriate name from the `Info.plist` file.

`-XX:+UseFileLocking`

Off by default, this option enables Carbon file locking. If your Java application will be interracting with files that may be simultaneously acted on by Carbon applications, use this flag. It keeps the respective applications from modifying the file while the other is accessing it.

Heap Size

`-Xms`*`size in bytes`*

Sets the initial size of the Java heap. The default size is 2097152 (2MB). The values must be a multiple of, and greater than, 1024 bytes (1KB). (The `-server` flag increases the default size to 32M.)

`-Xmn`*`size in bytes`*

Sets the initial Java heap size for the Eden generation. The default value is 640K. (The `-server` flag increases the default size to 2M.)

`-Xmx`*`size in bytes`*

Sets the maximum size to which the Java heap can grow. The default size is 64M. (The `-server` flag increases the default size to 128M.) The maximum heap limit is about 2 GB (2048MB).

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

Garbage Collection: Memory Usage

Note: Many of the garbage collection flags are dependent on the settings for the heap size. Make sure that you have the appropriate sizes set for the heap before fine-tuning how garbage collection uses that memory space.

`-XX:MinHeapFreeRatio=`*`percentage as a whole number`*

Modifies the minimum percentage of the heap kept free after garbage collection. The default value of 40. If at least 40% of the heap is not freed after garbage collection, the heap size increases.

`-XX:MaxHeapFreeRatio=`*`percentage as a whole number`*

Changes the maximum percentage of heap kept free after garbage collection before the heap is shrunk. The default value is 70. This means that if a garbage collection results in more than 70% of the heap being freed, the heap size decreases.

`-XX:NewSize=`*`size in bytes`*

Sets the default size for the Eden generation of allocated objects. The default value is 640K. (The `-server` flag increases the default size to 2M.)

`-XX:MaxNewSize=`*`size in bytes`*

Allows you to change the upper limit of the young object space in which new objects are allocated. The default value is 640K. (The `-server` flag increases the default size to 2M.)

`-XX:NewRatio=`*`value`*

Changes the ratio of new to old space sizes from the default value of 8 where the new space is 1/8 the size of the old space.

`-XX:SurvivorRatio=`*`number`*

Modifies the ratio of size of the Eden to the survivor space from the default of 10 where Eden is survivorRatio+2 times larger than the survivor space.

`-XX:TargetSurvivorRatio=`*`percentage`*

Desired percentage of survivor space used after scavenge. Default is 50.

`-XX:MaxPermSize=`*`size in MB`*

Modifies the size of the permanent generation. The default is value is 32 (32MB).

`-XX:-CleanPagesOnUncommit`

Normally, when the garbage collection mechanism determines that the heap can be shrunk, i.e., a lot of memory that had been used is no longer needed by the program, the Java VM marks the memory as clean. The operating system is then able to reclaim pages for use by other processes without needing to page out the old memory content to disk. This flag turns off this behavior so that these pages are not marked as clean.

Garbage Collection: General Settings

`-Xincgc`

This flag is not supported in Mac OS X. The train garbage collector is not supported.

`-Xnoclassgc`

Disables garbage collection of classes.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

`-XX:+UseConcMarkSweepGC`
: Enables concurrent mark and sweep garbage collection. This option has an effect only on multiprocessor computers.

`-XX:+UseParallelGC`
: Enables parallel garbage collection. This option has an effect only on multiprocessor computers.

`-XX:-DisableExplicitGC`
: Ignores explicit calls to `System.gc()` in your code. The VM still performs garbage collection when it normally would. This option just disallows you from explictly forcing garbage collection in your code.

`-XX:+PrintTenuringDistribution`
: Prints tenuring age information for allocated objects in the young generation.

Compilation

`-Xint`
: Runs the VM in interperated mode only. With this option, none of the bytecodes are compiled.

`-XX:CompileThreshold=`*`value`*
: Changes the number of method invocations (branches) before compilation begins. The default is 1000.

`-XX:-InlineUnreachedCalls`
: By default, the VM performs method inlining on whatever code it can to facilitate optimazation by the compiler. Setting this flag causes less code to be compiled due to inlining. Code that would not normally be reached, like exceptions, is not inlined and therefore interpereted at runtime. Setting this flag may be detrimental to performance.

`-XX:+CITime`
: Displays how much time is spent in compiled code.

`-XX:+PrintCompilation`
: Prints a trace of the methods as they are compiled.

Threading

`-XX:NewSizeThreadIncrease=`*`size in KB`*
: Allows you to specify how much to increment the young object space size per active thread. This option may be useful in regulating an increased allocation rate due to increased threads. The default increment is 16 (16 kilobytes).

`-XX:ThreadStackSize=`*`size in KB`*
: Changes the thread stack size from the operating system's default size.

`-XX:+UseTLAB`
: Enables a thread-local allocation buffer. Using the buffer allows for more scalable allocation for heavily threaded applications, greatly increasing allocation performance. It is on by default on multiprocessor computers and in Mac OS X Server.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

Sharing

`-XX:+PrintSharedSpaces`
: Turns on verbose output about sharing.

`-XX:-UseSharedSpaces`
: Turns sharing off.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *Java 1.4 Virtual Machine Options*.

Date	Notes
2005-04-29	Added information on the MaxDirectMemorySize option.
2004-10-05	Fixed references to `-XX:+PrintSharedSpaces` and `-XX:-InlineUnreachedCalls`.
2004-02-26	Added reference to `-XX:-CleanPagesOnUncommit` flag.
	Added reference to `-XX:+ MaxFDLimit` flag.
2004-01-08	Changed title to Java 1.4 Virtual Machine Options.
	Added reference to `-Xdock:icon` flag.
2003-06-23	Released as a standalone document.
2003-05-15	Released as an appendix to *Java 1.4.1 Development for Mac OS X*.

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.

Document Revision History

2005-04-29 | © 2004, 2005 Apple Computer, Inc. All Rights Reserved.