This presentation will act as an introduction to JVM tuning when using WebSphere Application Server V6.

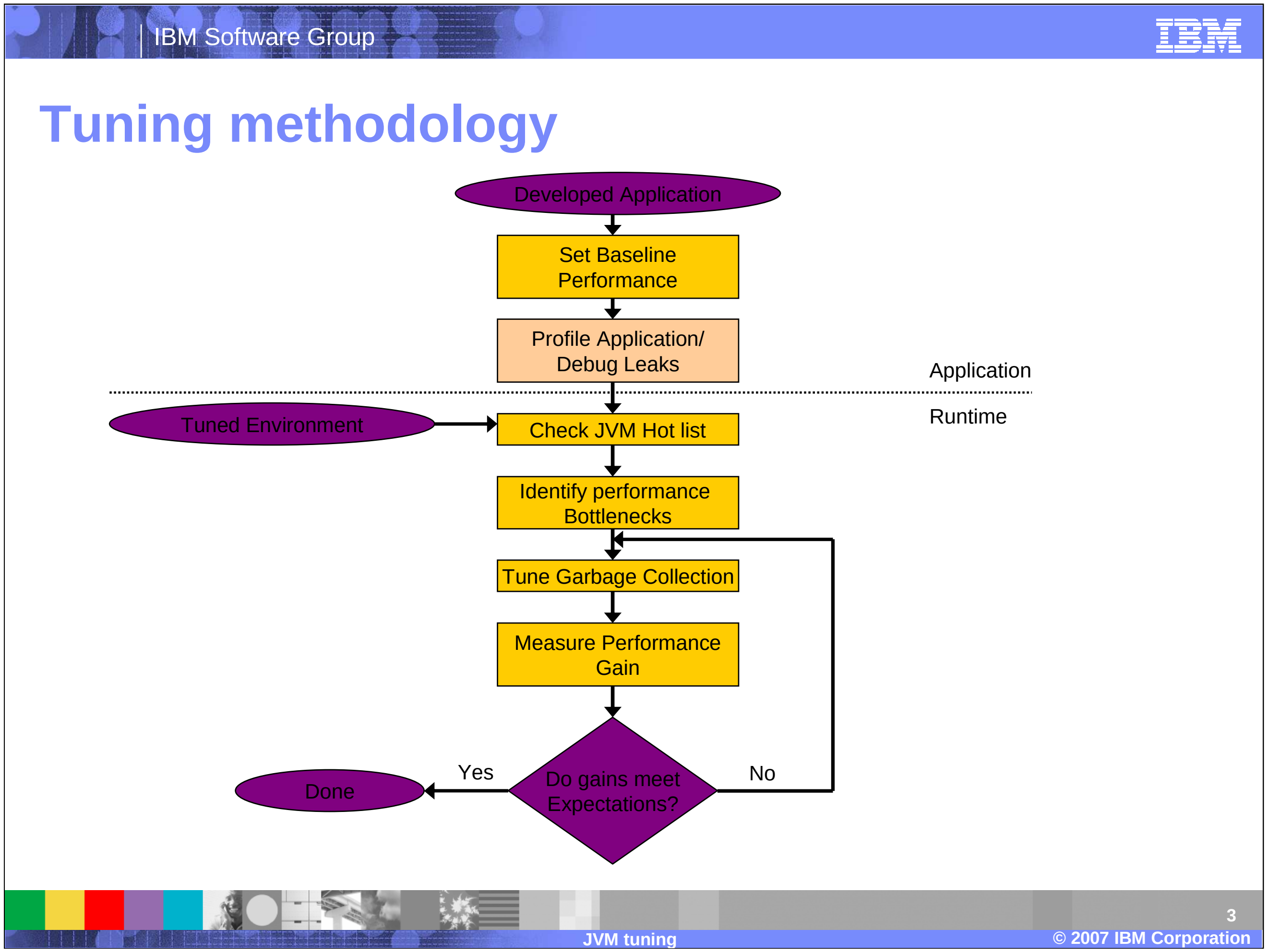
JVM tuning

After completing this topic, you should be able to:

- Present a tuning methodology
- Understand tuning considerations
- Describe special considerations for the Sun JDK

© 2007 IBM Corporation

After completing this topic, you should be able to present and understand a JVM™ tuning methodology.



JVM tuning is a key component in the tuning methodology of WebSphere Application Server and the applications it runs.

Heap tuning

- Proper JVM heap sizing
  - Often the only major area of WebSphere Application Server requiring tuning
- Too little heap
  - Causes the application to GC frequently
  - Consumes CPU
  - Extreme cases
    - Can lead to error conditions/server failure
- Too much heap
  - Wasteful
  - May cause longish response times, especially on Sun
- How do you know how the heap's performing
  - Process size != heap size
  - Use verbosegc to find GC characteristics

© 2007 IBM Corporation

Proper JVM heap sizing is critical to a healthy JVM and WebSphere Application Server. An insufficiently sized heap will force garbage collection to occur too frequently, which wastes cpu. Having a heap which is too large may waste valuable storage; can cause long response times, especially on Sun; and can lead to insufficient native heap, particularly on AIX. It is worth noting that with a min heap size different from the max heap size, the JVM is allowed to resize the heap over time to adjust to changing conditions. Even among the experts, opinions vary on whether that's a a good thing or a bad thing.

Tuning considerations (1 of 2)

- “Throughput” (the JVM definition)
  - Measure of productive time excluding time spent in GC.
- Pauses
  - Measure of time when application execution pauses during GC
- Turn on verbosegc for more information on GC operation
- Avoid calling System.gc() from the application
  - Causes the least efficient, slowest GC to take place
    - Always triggers a compaction
    - System.gc() does not always trigger an immediate GC
  - Allow the JVM to determine optimum time to GC
  - Can disable in IBM JDKs using –Xdisableexplicitgc

© 2007 IBM Corporation

There are two primary measures of garbage collection performance. *Throughput* is the percentage of total time not spent in garbage collection, considered over long periods of time. *Pauses* are the times when an application appears unresponsive because garbage collection is occurring.

Tuning considerations (2 of 2)

- Start with a reasonable maximum heap (-Xmx) size
  - 512M for WebSphere Application Server
- Test different maximum heap sizes to find optimal setting
  - Opposing forces
    - The larger the heap, typically the longer the GC cycle
    - The smaller the heap, the more frequently GC is required
  - Size maximum heap somewhat larger than steady state to allow for peak load
- Setting minimum heap size (-Xms) can improve server startup time
  - If using default, may need to be resized several times during startup
- But setting the minimum too large may impact runtime performance
  - Larger value means more memory space requiring garbage collection
  - The JVM can not compensate if you make a poor choice

© 2007 IBM Corporation

JVM heap tuning is an iterative process, and you should always start with a reasonable maximum heap size. While monitoring verbosegc output, test with different maximum heap sizes to find an optimal setting through load and stress testing.

Tuning considerations native heap

- BEWARE: The GC does not look into the native heap !!
- Ensure JVM is not being swapped out of memory to disk
  - Total Memory Used by JVM > Maximum Heap Size
    - Native objects (ex: Database Connections)
  - Physical Memory Available > Total Memory Used by JVM
    - JVMs do not page effectively due to garbage collection

© 2007 IBM Corporation

Remember that verbose garbage collection does not look into the native heap. Large numbers of thread or database connections can increase the memory used by the JVM outside of the Java heap. Any paging activity will seriously degrade JVM performance.

Sun JDK garbage collection

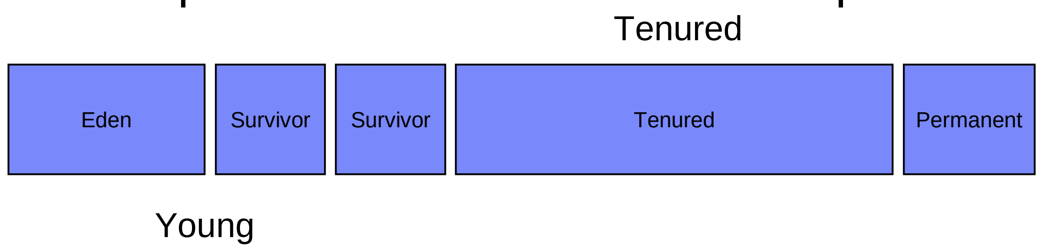
- Heap is divided into multiple generations
  - Young generation – objects are created here.
  - Older generations – objects are promoted to older generation.
- Shorter GC pauses since not all generations are collected at once.
- Each generation is garbage collected independently
  - Uses different collection strategy for different generations.
- Minor collection
  - Collects youngest generation.
- Full collection
  - If minor collection is not able to free enough memory to fulfill the allocation request, older generations are collected.

© 2007 IBM Corporation

In the Sun JDK, the heap is divided into multiple generations in an effort to reduce garbage collection pauses. The idea here is that as most objects have short lifetimes, it is useful to restrict garbage collection to the most recently allocated objects. Different algorithms are used in each generation. For young generations compaction or by copy algorithms are used. For older tenured generations the normal mark and sweep algorithms are used.

Sun JDK Generational Heap sizes: Young and Tenured ratio

- Consider sizing the ratio between Young and Tenured Generational Heaps (-XX:NewRatio)
  - Depends on lifetime of objects allocated by the application
  - Increasing the young generation becomes counterproductive at half the total heap or less



© 2007 IBM Corporation

There is no one right way to size generations. The best choice is determined by user requirements and the way the application uses memory. For example, a very large *young* generation may maximize throughput, but does so at the expense of footprint, promptness, and pause times. *young* generation pauses can be minimized by using a small *young* generation at the expense of throughput.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject= Feedback about SW5706G14B_JVM_Tuning.ppt

© 2007 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

JDK, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.