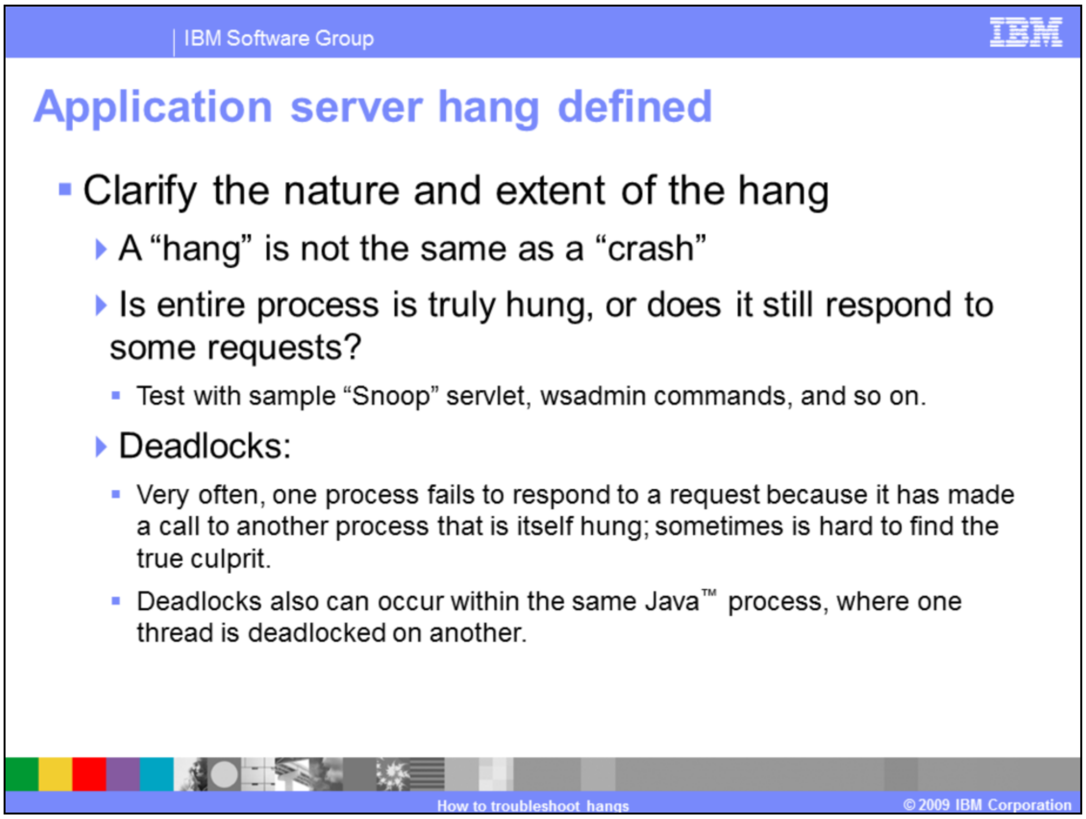
IBM Software Group

WebSphere Application Server 6.1 Problem determination guide

How to troubleshoot hangs

@business on demand.

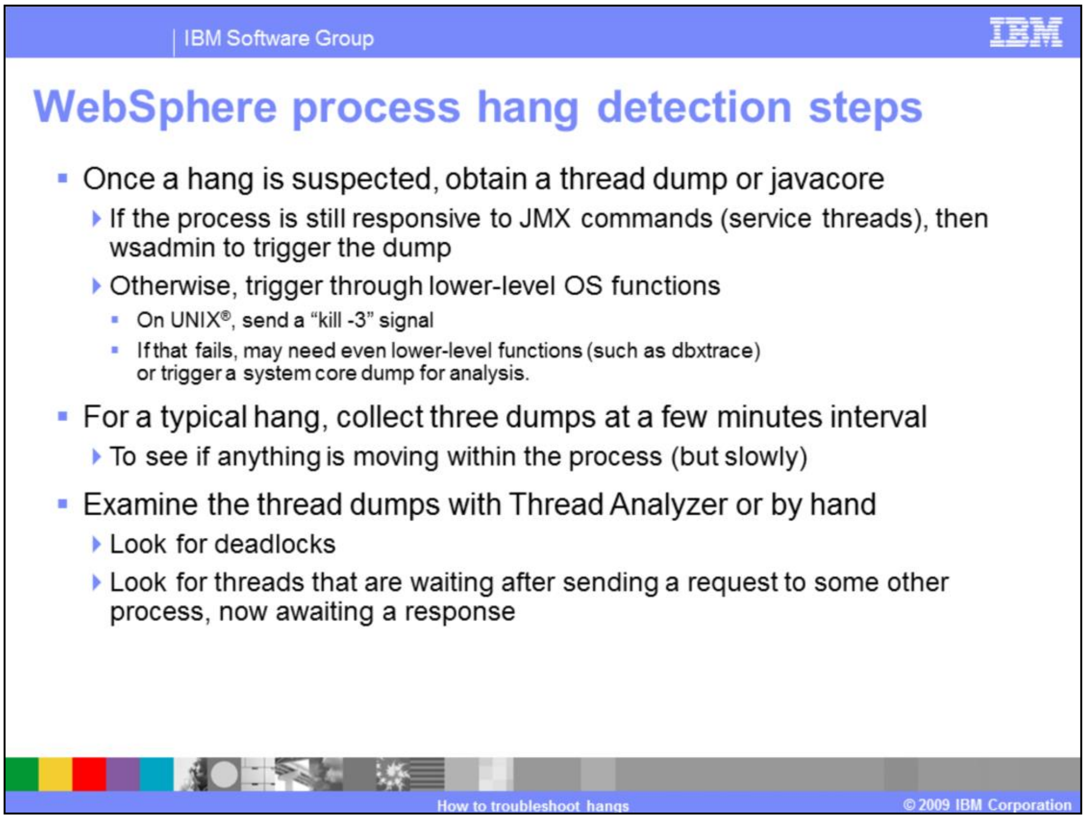
This presentation will act as an introduction to troubleshooting hangs when using WebSphere® Application Server version 6.1



Application server hang defined

- Clarify the nature and extent of the hang
  - A “hang” is not the same as a “crash”
  - Is entire process is truly hung, or does it still respond to some requests?
    - Test with sample “Snoop” servlet, wsadmin commands, and so on.
  - Deadlocks:
    - Very often, one process fails to respond to a request because it has made a call to another process that is itself hung; sometimes is hard to find the true culprit.
    - Deadlocks also can occur within the same Java™ process, where one thread is deadlocked on another.

A hang can be defined as a process or thread which has become unresponsive while still apparently alive. Contrast this with a crash, when a process abnormally ends with an error message.

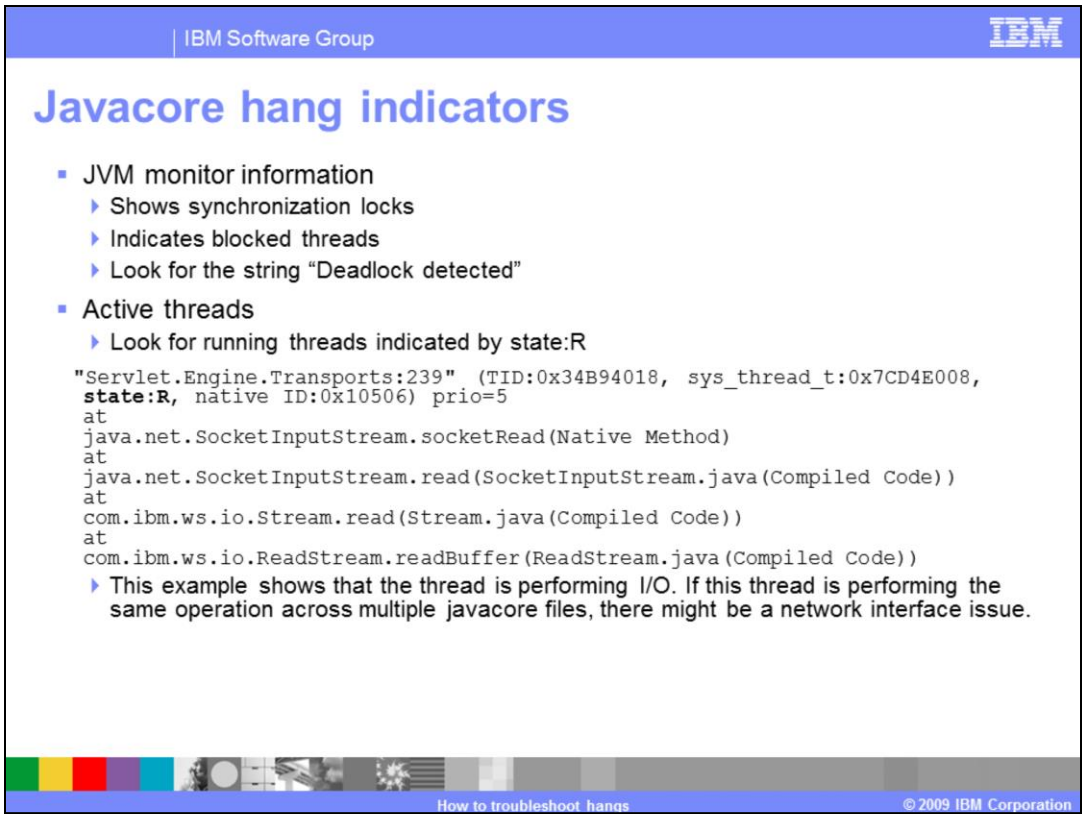

IBM Software Group

WebSphere process hang detection steps

- Once a hang is suspected, obtain a thread dump or javacore
  - If the process is still responsive to JMX commands (service threads), then wsadmin to trigger the dump
  - Otherwise, trigger through lower-level OS functions
    - On UNIX®, send a "kill -3" signal
    - If that fails, may need even lower-level functions (such as dbxtrace) or trigger a system core dump for analysis.
- For a typical hang, collect three dumps at a few minutes interval
  - To see if anything is moving within the process (but slowly)
- Examine the thread dumps with Thread Analyzer or by hand
  - Look for deadlocks
  - Look for threads that are waiting after sending a request to some other process, now awaiting a response

How to troubleshoot hangs © 2009 IBM Corporation

The basic problem determination method for hangs is to obtain one or, if possible, a series of thread dumps. If the process is still responsive to wsadmin commands, then the wsadmin command should be able to trigger the dump. Otherwise, depending upon your operating system, certain signals will trigger a thread dump. For a typical hang, collect three dumps at five minute intervals to determine if anything is moving within the process (albeit slowly). Examine the thread dumps with Thread Analyzer or by hand to look for deadlocks or to see if threads are awaiting responses from other processes. In newer JVMs, the javacore or thread dump will automatically perform deadlock detection and tell you if a deadlock has been detected. Look for the string "deadlock" in the javacore file.



IBM Software Group

IBM

Javacore hang indicators

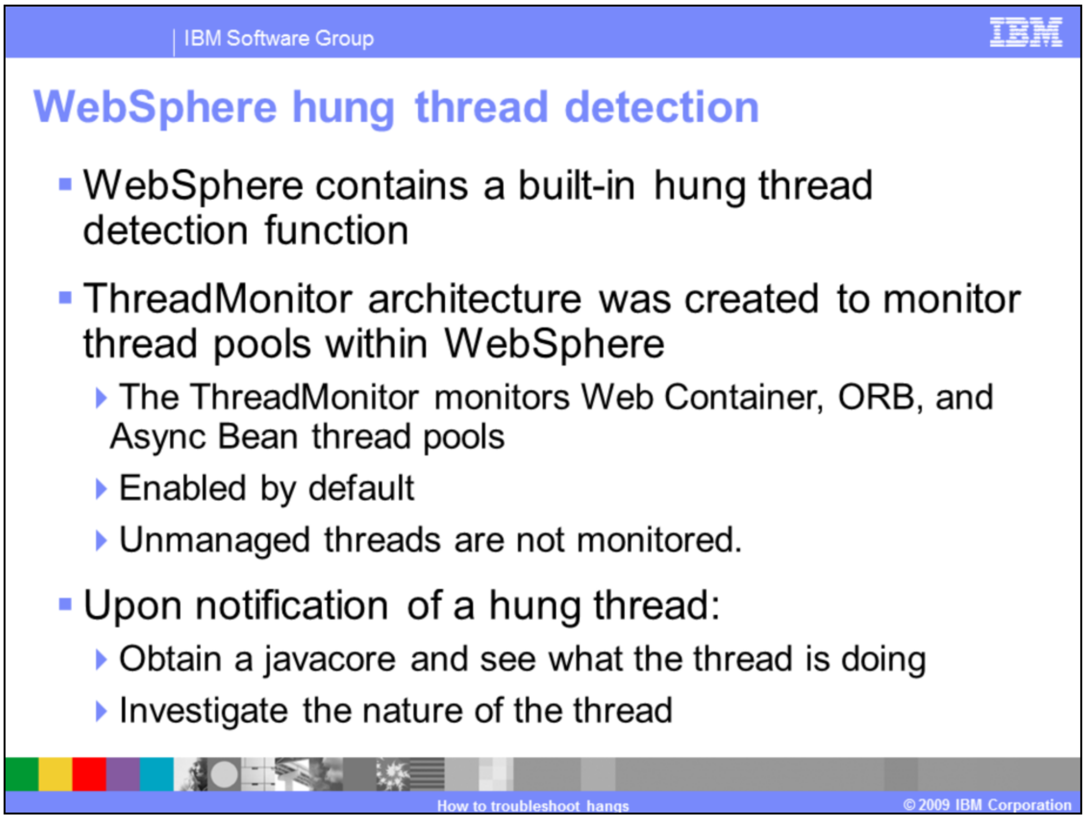
- JVM monitor information
  - Shows synchronization locks
  - Indicates blocked threads
  - Look for the string “Deadlock detected”
- Active threads
  - Look for running threads indicated by state:R

```
"Servlet.Engine.Transports:239"  (TID:0x34B94018, sys_thread_t:0x7CD4E008,
 state:R, native ID:0x10506) prio=5
 at
 java.net.SocketInputStream.socketRead(Native Method)
 at
 java.net.SocketInputStream.read(SocketInputStream.java(Compiled Code))
 at
 com.ibm.ws.io.Stream.read(Stream.java(Compiled Code))
 at
 com.ibm.ws.io.ReadStream.readBuffer(ReadStream.java(Compiled Code))
```

  - This example shows that the thread is performing I/O. If this thread is performing the same operation across multiple javacore files, there might be a network interface issue.

How to troubleshoot hangs © 2009 IBM Corporation

The monitor information in the javacore file shows what synchronization locks are held by which threads. It also shows which threads are blocked by monitors. This information is useful for determining the cause of a deadlocked or hung JVM.



IBM Software Group

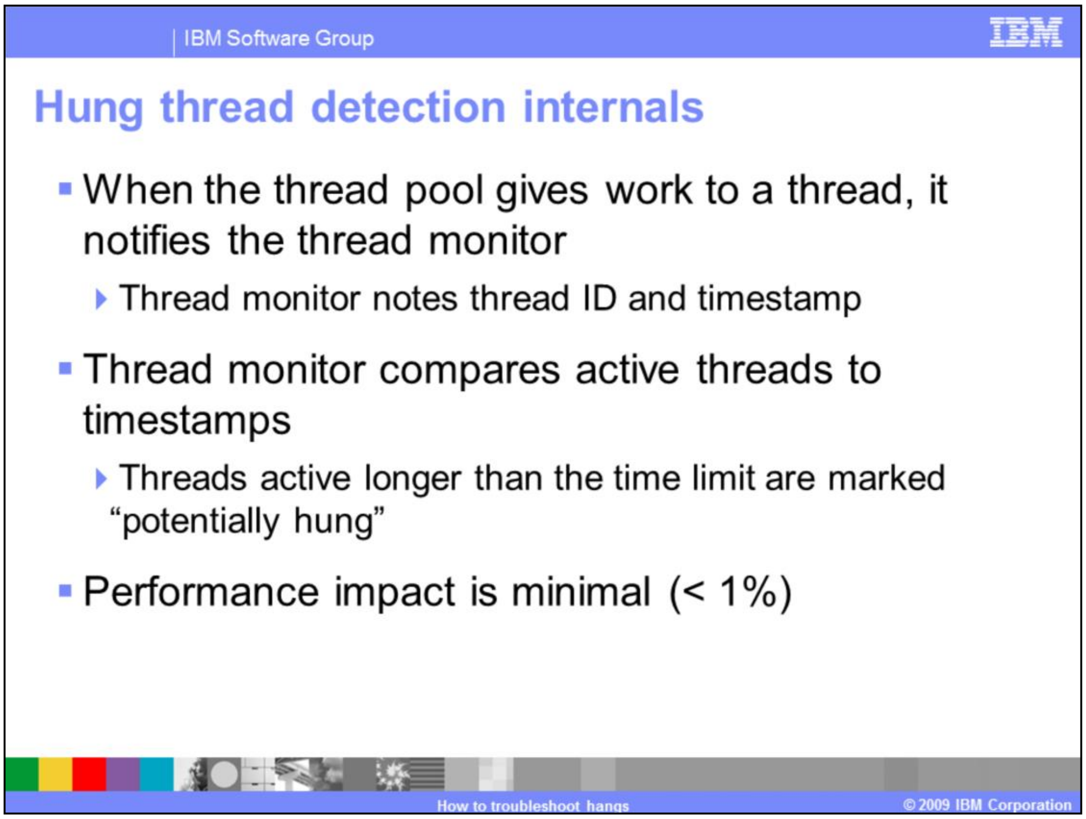
WebSphere hung thread detection

- WebSphere contains a built-in hung thread detection function
- ThreadMonitor architecture was created to monitor thread pools within WebSphere
  - The ThreadMonitor monitors Web Container, ORB, and Async Bean thread pools
  - Enabled by default
  - Unmanaged threads are not monitored.
- Upon notification of a hung thread:
  - Obtain a javacore and see what the thread is doing
  - Investigate the nature of the thread

How to troubleshoot hangs

© 2009 IBM Corporation

WebSphere Application Server contains a built-in hung thread detection function. It monitors the Web container, Object Requet Broker, and Asynchronous Bean thread pools, and is enabled by default. Note that unmanaged threads are not monitored. You can configure a hang detection policy to accommodate your applications and environment so that potential hangs can be reported, providing earlier detection of failing servers.



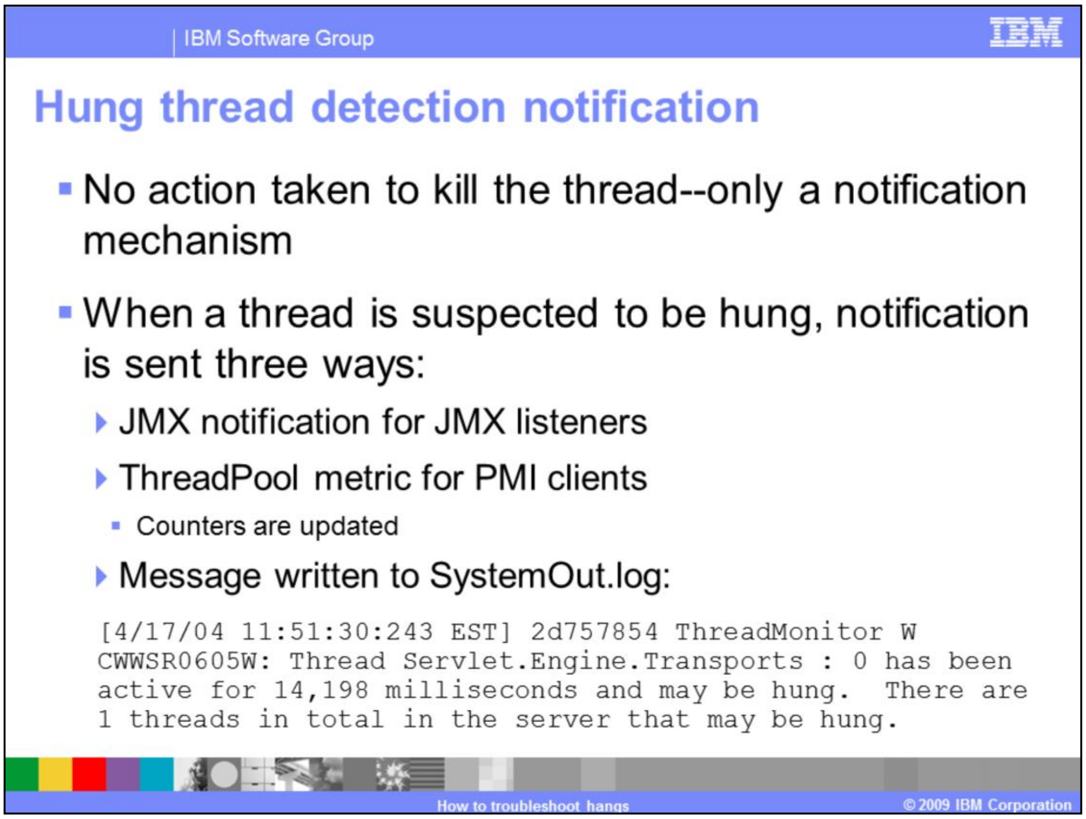
IBM Software Group

Hung thread detection internals

- When the thread pool gives work to a thread, it notifies the thread monitor
  - Thread monitor notes thread ID and timestamp
- Thread monitor compares active threads to timestamps
  - Threads active longer than the time limit are marked "potentially hung"
- Performance impact is minimal (< 1%)

How to troubleshoot hangs © 2009 IBM Corporation

When the thread pool issues work to a thread, it sends a notification to the thread monitor, which notes the thread identifier and the time in a list. At user-configurable intervals, the thread monitor looks at the active threads, and compares them to the list, to determine how long each thread has been active. If a thread has been active longer than the user-specified threshold, the thread is marked as "potentially hung", and notifications are sent.


IBM Software Group

IBM

Hung thread detection notification

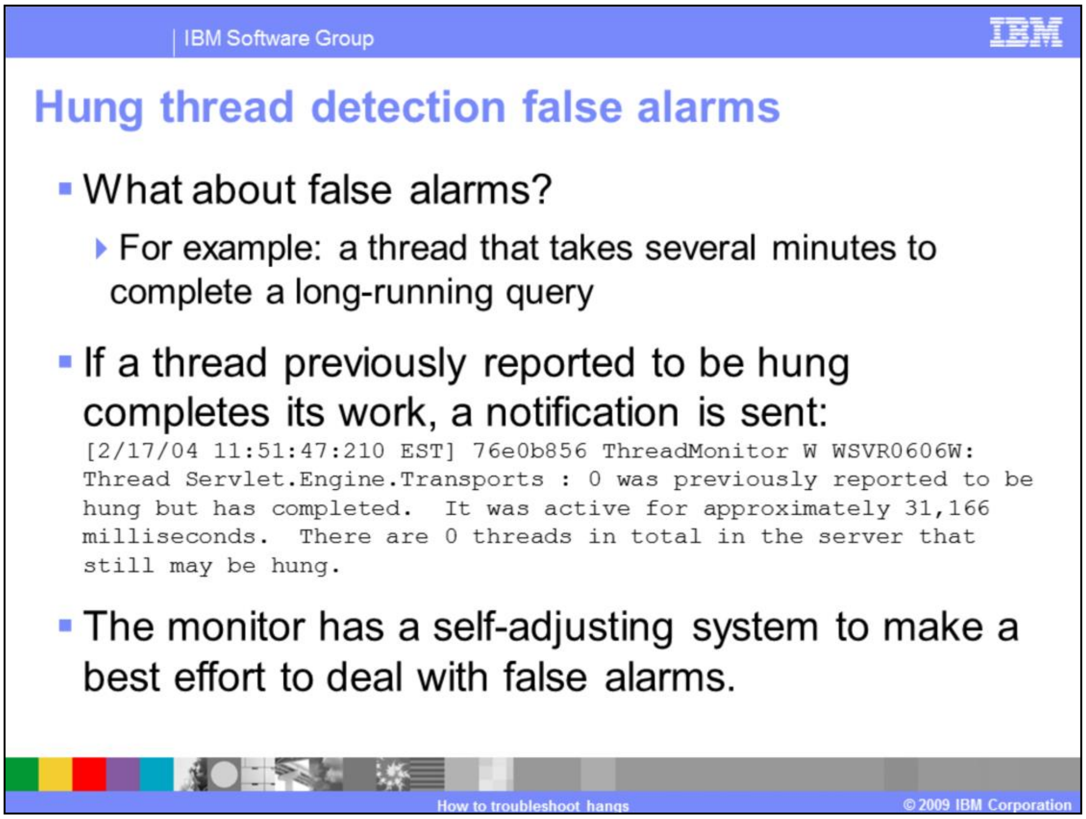
- No action taken to kill the thread--only a notification mechanism
- When a thread is suspected to be hung, notification is sent three ways:
  - JMX notification for JMX listeners
  - ThreadPool metric for PMI clients
    - Counters are updated
  - Message written to SystemOut.log:

```
[4/17/04 11:51:30:243 EST] 2d757854 ThreadMonitor W
CWWSR0605W: Thread Servlet.Engine.Transports : 0 has been
active for 14,198 milliseconds and may be hung.  There are
1 threads in total in the server that may be hung.
```

How to troubleshoot hangs

© 2009 IBM Corporation

The thread monitor doesn't try to deal with the hung threads, it just issues notifications, so that the administrator or developer can deal with the issues. The message written to the SystemOut log has a message identifier of WSVR0605W, and shows the thread name, the approximate time that the thread has been active, and the total number of threads which may be hung.



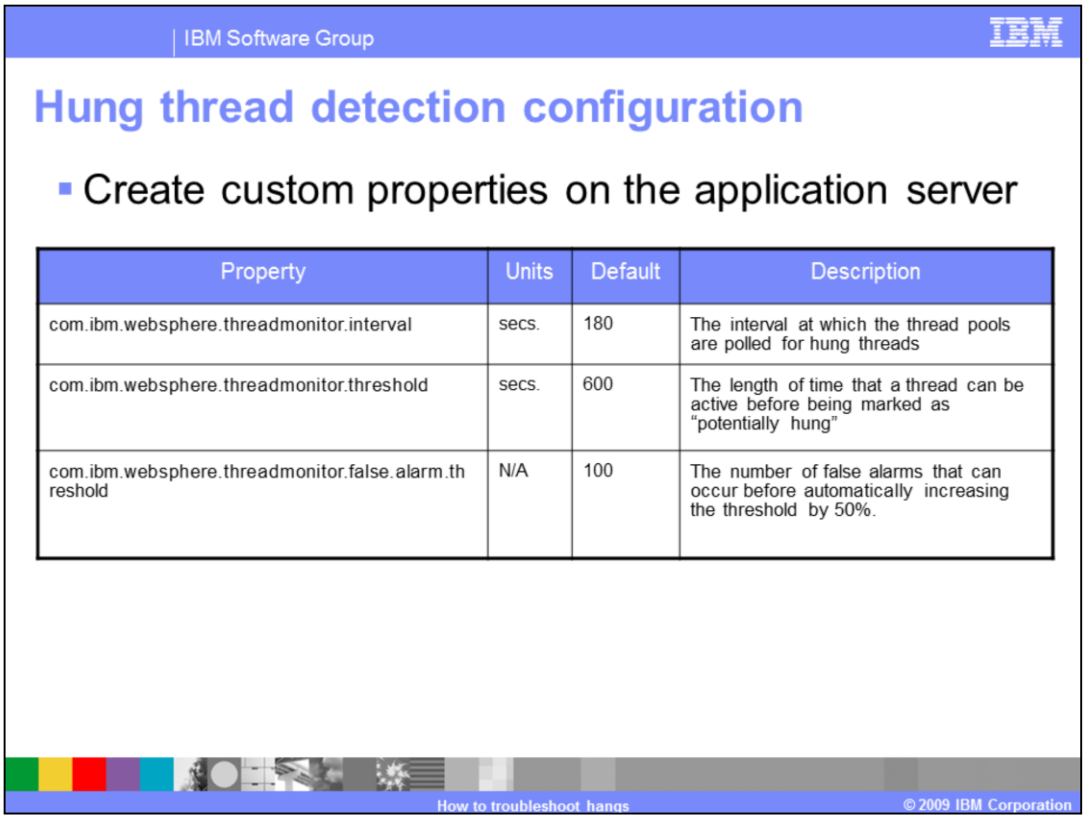
Hung thread detection false alarms

- What about false alarms?
  - For example: a thread that takes several minutes to complete a long-running query
- If a thread previously reported to be hung completes its work, a notification is sent:

```
[2/17/04 11:51:47:210 EST] 76e0b856 ThreadMonitor W WSVR0606W:
Thread Servlet.Engine.Transports : 0 was previously reported to be
hung but has completed.  It was active for approximately 31,166
milliseconds.  There are 0 threads in total in the server that
still may be hung.
```

- The monitor has a self-adjusting system to make a best effort to deal with false alarms.

It's possible that a thread can run for longer than the specified threshold for legitimate reasons. When a thread that was previously marked as "potentially hung" completes its work and exits, a notification is sent. After a certain number of false alarms, the threshold is automatically increased by 50% to account for these long-running threads.

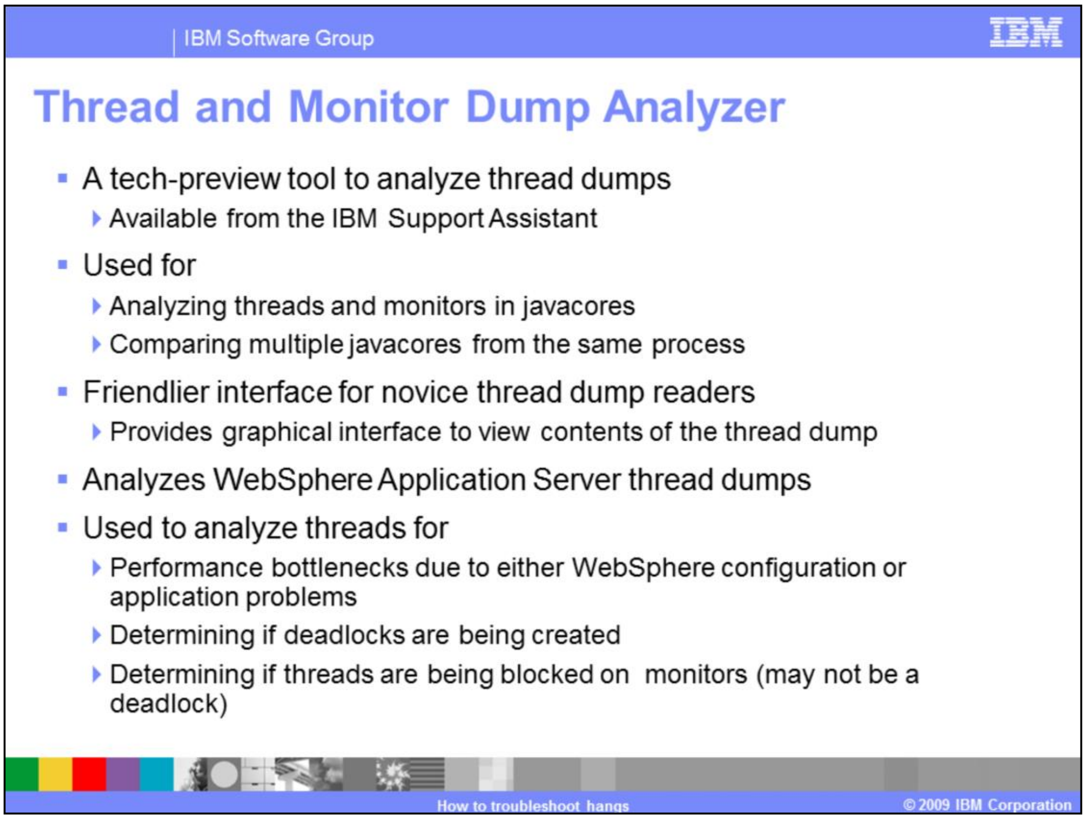


Hung thread detection configuration

- Create custom properties on the application server

Property	Units	Default	Description
com.ibm.websphere.threadmonitor.interval	secs.	180	The interval at which the thread pools are polled for hung threads
com.ibm.websphere.threadmonitor.threshold	secs.	600	The length of time that a thread can be active before being marked as "potentially hung"
com.ibm.websphere.threadmonitor.false.alarm.threshold	N/A	100	The number of false alarms that can occur before automatically increasing the threshold by 50%.

The hang detection policy can be configured by creating custom properties for the application server.



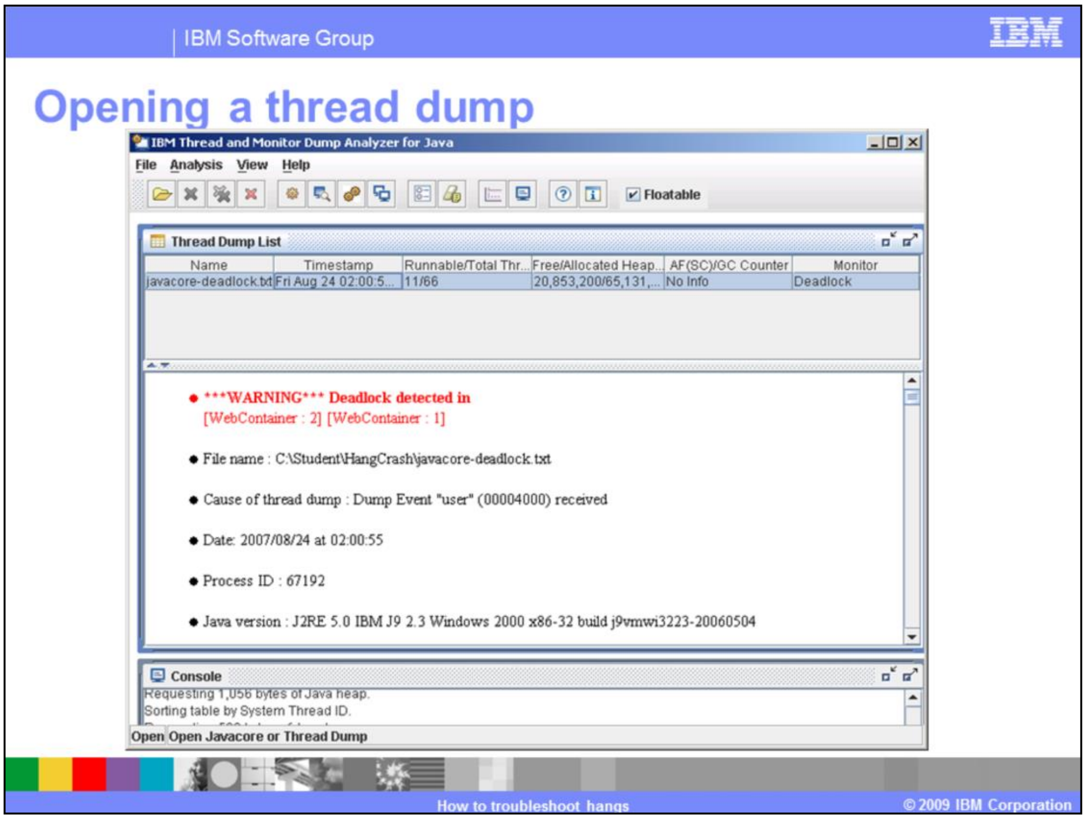
IBM Software Group

Thread and Monitor Dump Analyzer

- A tech-preview tool to analyze thread dumps
  - Available from the IBM Support Assistant
- Used for
  - Analyzing threads and monitors in javacores
  - Comparing multiple javacores from the same process
- Friendlier interface for novice thread dump readers
  - Provides graphical interface to view contents of the thread dump
- Analyzes WebSphere Application Server thread dumps
- Used to analyze threads for
  - Performance bottlenecks due to either WebSphere configuration or application problems
  - Determining if deadlocks are being created
  - Determining if threads are being blocked on monitors (may not be a deadlock)

How to troubleshoot hangs © 2009 IBM Corporation

The Thread and Monitor Dump Analyzer is an IBM Support tool designed to simplify the act of analyzing javacore files. It is designed so that novice troubleshooters and experts alike can use the tool to analyze thread dumps. The tool is available through the IBM Support Assistant workbench.


IBM Software Group

Opening a thread dump

IBM Thread and Monitor Dump Analyzer for Java

Name	Timestamp	Runnable/Total Thr...	Free/Allocated Heap...	AF(SC)/GC Counter	Monitor
javacore-deadlock.txt	Fri Aug 24 02:00:5...	11/66	20,853,200/65,131,...	No Info	Deadlock

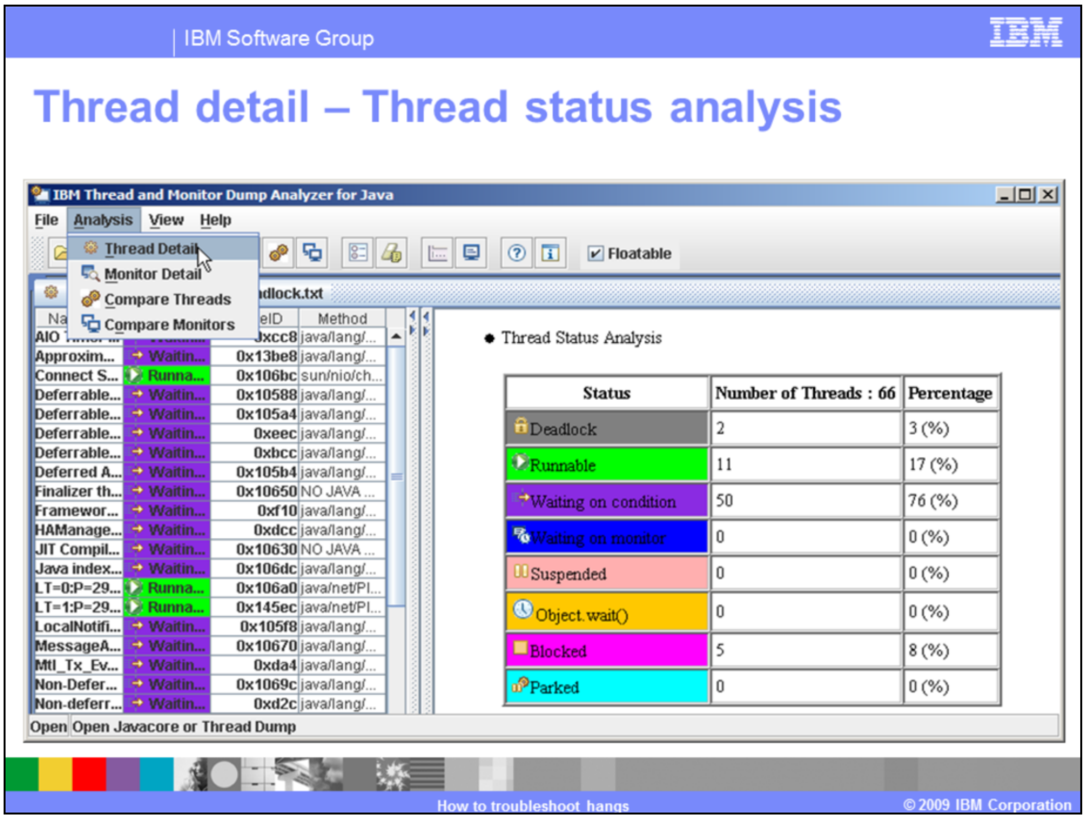
- ***WARNING*** Deadlock detected in [WebContainer : 2] [WebContainer : 1]
- File name : C:\Student\HangCrash\javacore-deadlock.txt
- Cause of thread dump : Dump Event "user" (00004000) received
- Date: 2007/08/24 at 02:00:55
- Process ID : 67192
- Java version : J2RE 5.0 IBM J9 2.3 Windows 2000 x86-32 build j9vmwi3223-20060504

```
Requesting 1,056 bytes of Java heap.
Sorting table by System Thread ID.
```

Open Open Javacore or Thread Dump

How to troubleshoot hangs © 2009 IBM Corporation

You can search the local file system for one or more javacore files. Each file is loaded into the tool and analyzed. The tool will provide a warning if any deadlocked threads are found within the dumps. Additionally, the tool will display summary information from the javacore file such as file name, cause of the dump, data, process identifier, Java version, Java heap information, and much more.



Thread detail – Thread status analysis

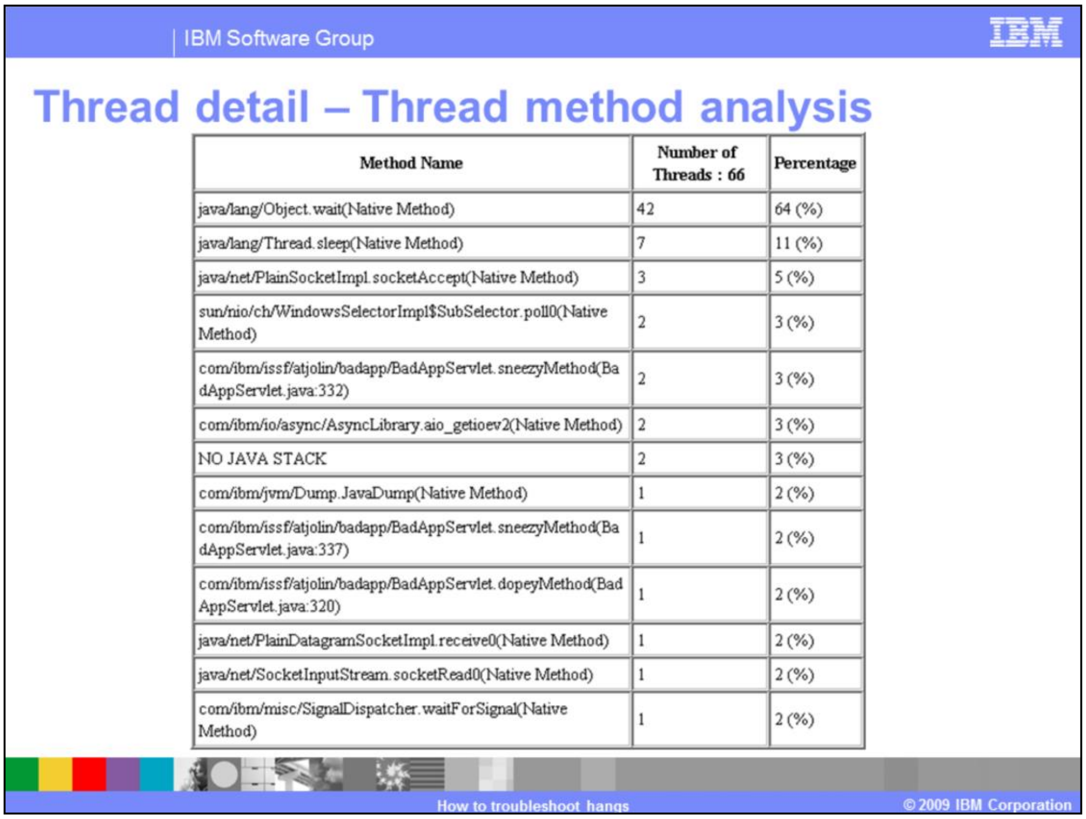
Na		eID	Method
AIO		xcc8	java/lang/...
Approxim...	Waitin...	0x13be8	java/lang/...
Connect S...	Runna...	0x106bc	sun/nio/ch...
Deferrable...	Waitin...	0x10588	java/lang/...
Deferrable...	Waitin...	0x105a4	java/lang/...
Deferrable...	Waitin...	0xeec	java/lang/...
Deferrable...	Waitin...	0xbcc	java/lang/...
Deferred A...	Waitin...	0x105b4	java/lang/...
Finalizer th...	Waitin...	0x10650	NO JAVA ...
Framewor...	Waitin...	0xf10	java/lang/...
HAManage...	Waitin...	0xdcc	java/lang/...
JIT Compil...	Waitin...	0x10630	NO JAVA ...
Java index...	Waitin...	0x106dc	java/lang/...
LT=0:P=29...	Runna...	0x106a0	java/net/PI...
LT=1:P=29...	Runna...	0x145ec	java/net/PI...
LocalNotifi...	Waitin...	0x105f8	java/lang/...
MessageA...	Waitin...	0x10670	java/lang/...
Mtl_Tx_Ev...	Waitin...	0xda4	java/lang/...
Non-Defer...	Waitin...	0x1069c	java/lang/...
Non-deferr...	Waitin...	0xd2c	java/lang/...

- Thread Status Analysis

Status	Number of Threads : 66	Percentage
Deadlock	2	3 (%)
Runnable	11	17 (%)
Waiting on condition	50	76 (%)
Waiting on monitor	0	0 (%)
Suspended	0	0 (%)
Object.wait()	0	0 (%)
Blocked	5	8 (%)
Parked	0	0 (%)

Open Open Javacore or Thread Dump

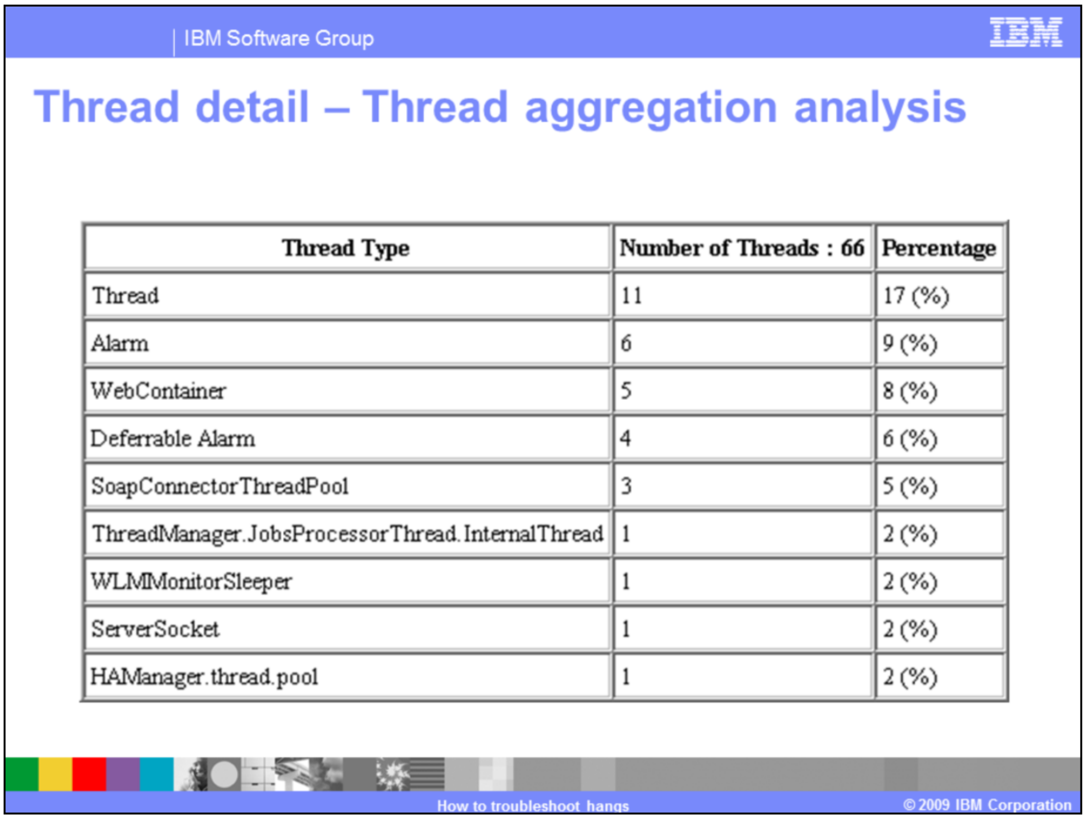
The Analysis menu allows you to display thread and monitor details for a single javacore. If you open multiple javacores, you can display a comparative thread or monitor analysis. The thread detail analysis displays thread status analysis, thread method analysis, thread aggregation analysis, memory segment analysis. The thread status analysis shows the number of threads in each state: Deadlocked, Runnable, Blocked, and so forth. Threads are sorted by thread name. Thread Detail View provides the thread name, the state of a thread, the method name, the Java stack trace, and the native stack trace.



Thread detail – Thread method analysis

Method Name	Number of Threads : 66	Percentage
java/lang/Object.wait(Native Method)	42	64 (%)
java/lang/Thread.sleep(Native Method)	7	11 (%)
java/net/PlainSocketImpl.socketAccept(Native Method)	3	5 (%)
sun/nio/ch/WindowsSelectorImpl$SubSelector.poll0(Native Method)	2	3 (%)
com/ibm/issf/atjolin/badapp/BadAppServlet.sneezyMethod(BadAppServlet.java:332)	2	3 (%)
com/ibm/io/async/AsyncLibrary.aio_getioev2(Native Method)	2	3 (%)
NO JAVA STACK	2	3 (%)
com/ibm/jvm/Dump.JavaDump(Native Method)	1	2 (%)
com/ibm/issf/atjolin/badapp/BadAppServlet.sneezyMethod(BadAppServlet.java:337)	1	2 (%)
com/ibm/issf/atjolin/badapp/BadAppServlet.dopeyMethod(BadAppServlet.java:320)	1	2 (%)
java/net/PlainDatagramSocketImpl.receive0(Native Method)	1	2 (%)
java/net/SocketInputStream.socketRead0(Native Method)	1	2 (%)
com/ibm/misc/SignalDispatcher.waitForSignal(Native Method)	1	2 (%)

The thread method analysis view provides a summary of what all of the threads in the JVM were doing at the time the dump was taken.

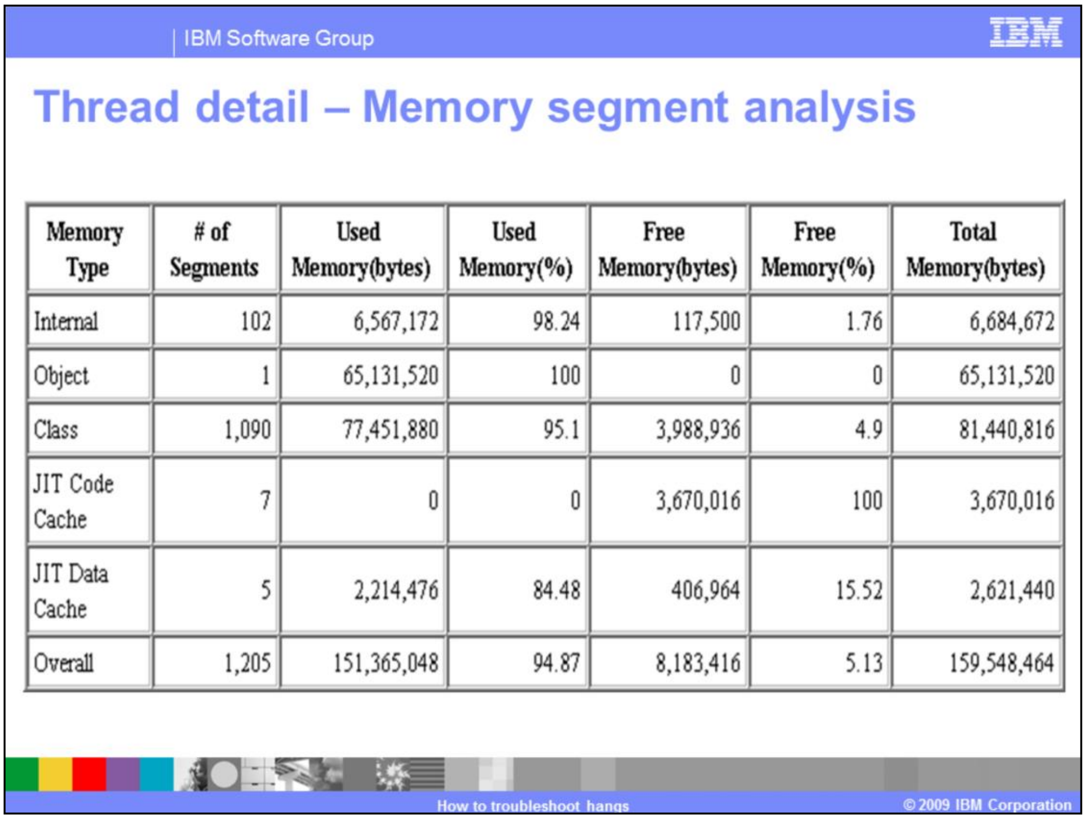

IBM Software Group

Thread detail – Thread aggregation analysis

Thread Type	Number of Threads : 66	Percentage
Thread	11	17 (%)
Alarm	6	9 (%)
WebContainer	5	8 (%)
Deferrable Alarm	4	6 (%)
SoapConnectorThreadPool	3	5 (%)
ThreadManager.JobsProcessorThread.InternalThread	1	2 (%)
WLMMonitorSleeper	1	2 (%)
ServerSocket	1	2 (%)
HAManager.thread.pool	1	2 (%)

How to troubleshoot hangs

Thread aggregation analysis details the types of threads that were seen in the dump.


IBM Software Group

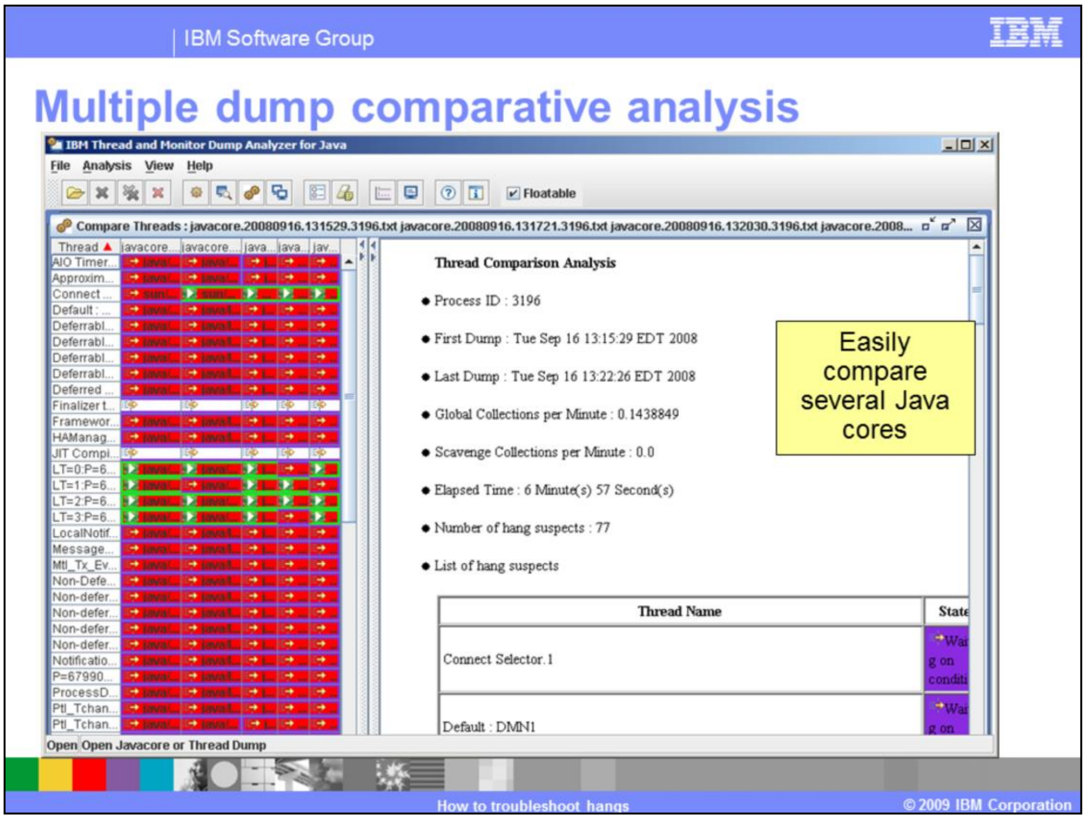
IBM

Thread detail – Memory segment analysis

Memory Type	# of Segments	Used Memory(bytes)	Used Memory(%)	Free Memory(bytes)	Free Memory(%)	Total Memory(bytes)
Internal	102	6,567,172	98.24	117,500	1.76	6,684,672
Object	1	65,131,520	100	0	0	65,131,520
Class	1,090	77,451,880	95.1	3,988,936	4.9	81,440,816
JIT Code Cache	7	0	0	3,670,016	100	3,670,016
JIT Data Cache	5	2,214,476	84.48	406,964	15.52	2,621,440
Overall	1,205	151,365,048	94.87	8,183,416	5.13	159,548,464

How to troubleshoot hangs © 2009 IBM Corporation

This slide shows sample out for the memory segment analysis view. This view provide information regarding the amount of memory allocated and the number of memory segments used by the server from which this dump was taken.



Multiple dump comparative analysis

IBM Thread and Monitor Dump Analyzer for Java

Compare Threads : javacore.20080916.131529.3196.txt javacore.20080916.131721.3196.txt javacore.20080916.132030.3196.txt javacore.2008...

Thread ▲	javacore...	javacore...	java...	java...	jav...
AIO Timer...	java/...	java/l...	j...	...	...
Approxim...	java/...	java/l...	j...	...	...
Connect ...	sun/...	sun/...	...	...	...
Default : ...	java/...	java/l...	j...	...	...
Deferrabl...	java/...	java/l...	j...	...	...
Deferrabl...	java/...	java/l...	j...	...	...
Deferrabl...	java/...	java/l...	j...	...	...
Deferrabl...	java/...	java/l...	j...	...	...
Deferred ...	java/...	java/l...	j...	...	...
Finalizer t...					
Framewor...	java/...	java/l...	j...	...	...
HAManag...	java/...	java/l...	j...	...	...
JIT Compi...					
LT=0:P=6...	java/...	java/...	j...	...	...
LT=1:P=6...	java/...	java/...	j...	...	...
LT=2:P=6...	java/...	java/...	j...	...	...
LT=3:P=6...	java/...	java/...	j...	...	...
LocalNotif...	java/...	java/l...	j...	...	...
Message...	java/...	java/l...	j...	...	...
Mtl_Tx_Ev...	java/...	java/l...	j...	...	...
Non-Defe...	java/...	java/l...	j...	...	...
Non-defer...	java/...	java/l...	j...	...	...
Non-defer...	java/...	java/l...	j...	...	...
Non-defer...	java/...	java/l...	j...	...	...
Non-defer...	java/...	java/l...	j...	...	...
Notificatio...	java/...	java/l...	j...	...	...
P=67990...	java/...	java/l...	j...	...	...
ProcessD...	java/...	java/l...	j...	...	...
Ptl_Tchan...	java/...	java/l...	j...	...	...
Ptl_Tchan...	java/...	java/...	j...	...	...

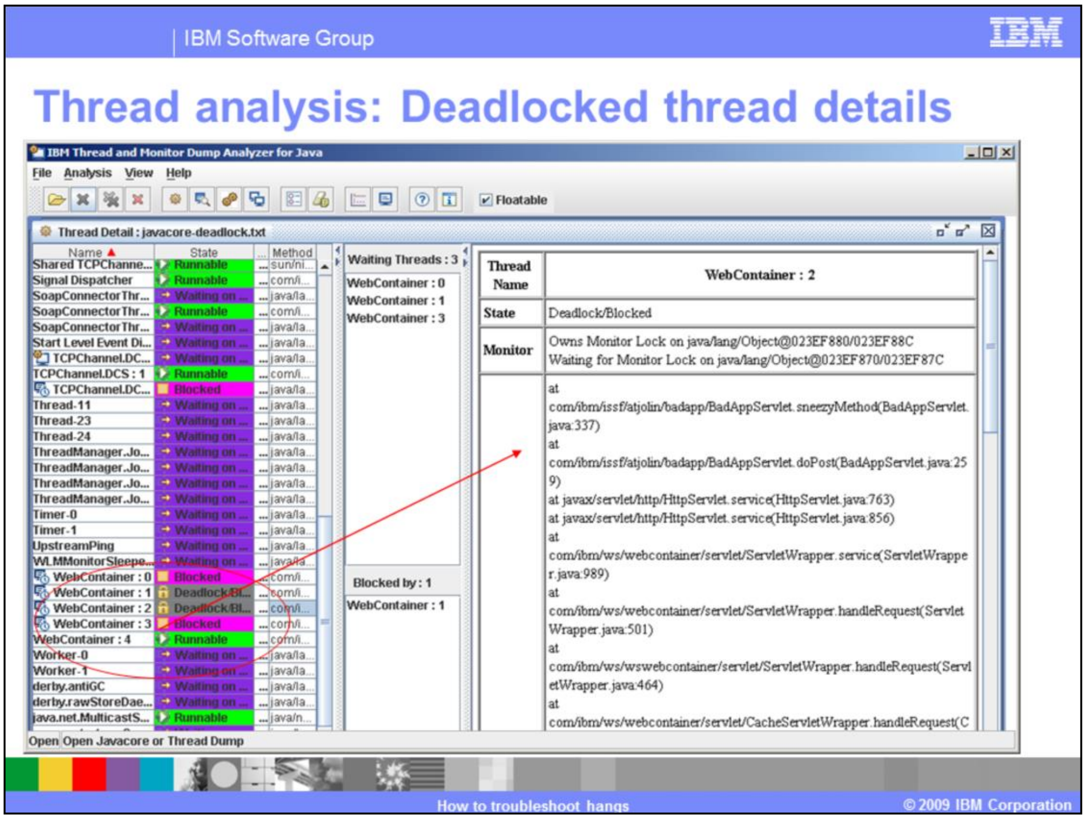
Thread Comparison Analysis

- Process ID : 3196
- First Dump : Tue Sep 16 13:15:29 EDT 2008
- Last Dump : Tue Sep 16 13:22:26 EDT 2008
- Global Collections per Minute : 0.1438849
- Scavenge Collections per Minute : 0.0
- Elapsed Time : 6 Minute(s) 57 Second(s)
- Number of hang suspects : 77
- List of hang suspects

Thread Name	State
Connect Selector.1	Wai g on conditi
Default : DMN1	Wai g on

Open Open Javacore or Thread Dump

The Thread and Monitor Dump Analyzer tool can provide comparative analysis between one or more thread dumps taken from the same server. This is useful for determining if threads are truly hung or are just moving very slowly. The tool provides color highlighting to easily identify threads states.



Thread analysis: Deadlocked thread details

IBM Thread and Monitor Dump Analyzer for Java

File Analysis View Help

Floatable

Thread Detail : javacore-deadlock.txt

Name	State	Method
Shared TCPChanne...	Runnable	sun/ni...
Signal Dispatcher	Runnable	com/i...
SoapConnectorThr...	Waiting on ...	java/la...
SoapConnectorThr...	Runnable	com/i...
SoapConnectorThr...	Waiting on ...	java/la...
Start Level Event Di...	Waiting on ...	java/la...
TCPChannel.DC...	Waiting on ...	java/la...
TCPChannel.DCS : 1	Runnable	com/i...
TCPChannel.DC...	Blocked	java/la...
Thread-11	Waiting on ...	java/la...
Thread-23	Waiting on ...	java/la...
Thread-24	Waiting on ...	java/la...
ThreadManager.Jo...	Waiting on ...	java/la...
ThreadManager.Jo...	Waiting on ...	java/la...
ThreadManager.Jo...	Waiting on ...	java/la...
ThreadManager.Jo...	Waiting on ...	java/la...
Timer-0	Waiting on ...	java/la...
Timer-1	Waiting on ...	java/la...
UpstreamPing	Waiting on ...	java/la...
WLMMonitorSleepe...	Waiting on ...	java/la...
WebContainer : 0	Blocked	com/i...
WebContainer : 1	Deadlock/Bl...	com/i...
WebContainer : 2	Deadlock/Bl...	com/i...
WebContainer : 3	Blocked	com/i...
WebContainer : 4	Runnable	com/i...
Worker-0	Waiting on ...	java/la...
Worker-1	Waiting on ...	java/la...
derby.antiGC	Waiting on ...	java/la...
derby.rawStoreDae...	Waiting on ...	java/la...
java.net.MulticastS...	Runnable	java/n...

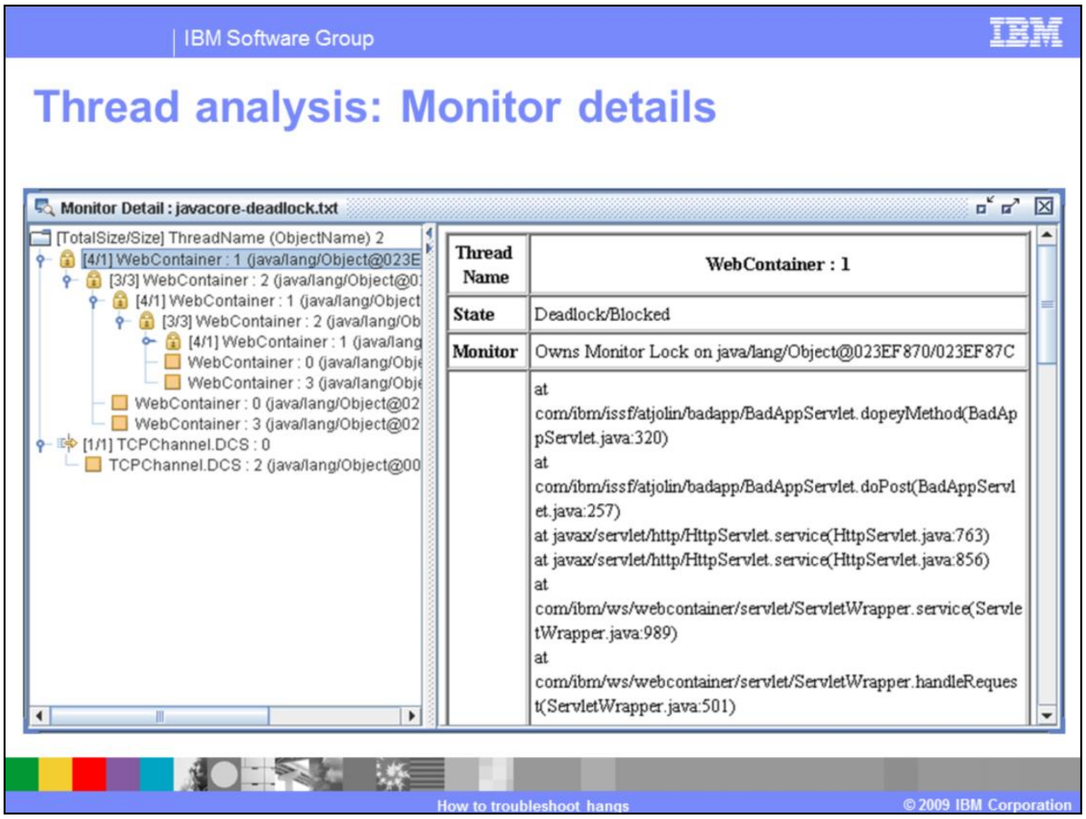
Waiting Threads : 3
WebContainer : 0
WebContainer : 1
WebContainer : 3

Blocked by : 1
WebContainer : 1

Thread Name	WebContainer : 2
State	Deadlock/Blocked
Monitor	Owns Monitor Lock on java/lang/Object@023EF880/023EF88C Waiting for Monitor Lock on java/lang/Object@023EF870/023EF87C
	at com/ibm/issf/atjolin/badapp/BadAppServlet.sneezyMethod(BadAppServlet.java:337) at com/ibm/issf/atjolin/badapp/BadAppServlet.doPost(BadAppServlet.java:259) at javax/servlet/http/HttpServlet.service(HttpServlet.java:763) at javax/servlet/http/HttpServlet.service(HttpServlet.java:856) at com/ibm/ws/webcontainer/servlet/ServletWrapper.service(ServletWrapper.java:989) at com/ibm/ws/webcontainer/servlet/ServletWrapper.handleRequest(ServletWrapper.java:501) at com/ibm/ws/wswebcontainer/servlet/ServletWrapper.handleRequest(ServletWrapper.java:464) at com/ibm/ws/webcontainer/servlet/CacheServletWrapper.handleRequest(C

Open Open Javacore or Thread Dump

In the left pane, each thread name can be selected and the details of the thread are displayed in the right pane. Deadlocked threads appear in the thread listing with a state of Deadlock or Blocked. They are also highlighted with a gray color and have a padlock icon on them for easy identification. By clicking on the thread in the left pane, one can see the thread waiting on this thread and the thread that is blocking the selected thread.


IBM Software Group

IBM

Thread analysis: Monitor details

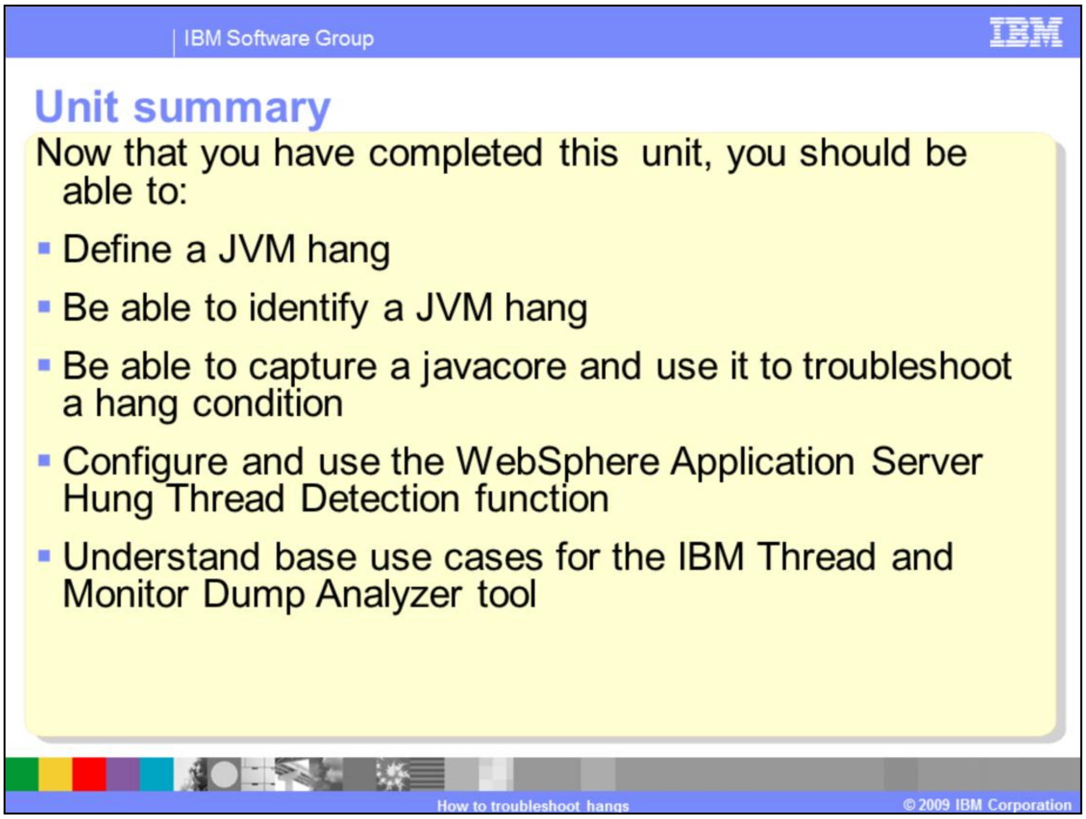
Monitor Detail : javacore-deadlock.txt

```
[TotalSize/Size] ThreadName (ObjectName) 2
  [4/1] WebContainer : 1 (java/lang/Object@023E
    [3/3] WebContainer : 2 (java/lang/Object@0
      [4/1] WebContainer : 1 (java/lang/Object
        [3/3] WebContainer : 2 (java/lang/Ob
          [4/1] WebContainer : 1 (java/lang
            WebContainer : 0 (java/lang/Obj
            WebContainer : 3 (java/lang/Obj
      WebContainer : 0 (java/lang/Object@02
      WebContainer : 3 (java/lang/Object@02
  [1/1] TCPChannel.DCS : 0
    TCPChannel.DCS : 2 (java/lang/Object@00
```

Thread Name	WebContainer : 1
State	Deadlock/Blocked
Monitor	Owns Monitor Lock on java/lang/Object@023EF870/023EF87C
	at com/ibm/issf/atjolin/badapp/BadAppServlet.dopeyMethod(BadAppServlet.java:320) at com/ibm/issf/atjolin/badapp/BadAppServlet.doPost(BadAppServlet.java:257) at javax/servlet/http/HttpServlet.service(HttpServlet.java:763) at javax/servlet/http/HttpServlet.service(HttpServlet.java:856) at com/ibm/ws/webcontainer/servlet/ServletWrapper.service(ServletWrapper.java:989) at com/ibm/ws/webcontainer/servlet/ServletWrapper.handleRequest(ServletWrapper.java:501)

How to troubleshoot hangs © 2009 IBM Corporation

The Monitor Detail view provides a hierarchical tree of the threads. By clicking each thread in the hierarchy you can see information about the monitor locks held by the thread and any monitor locks the thread is waiting for.



IBM Software Group

IBM

Unit summary

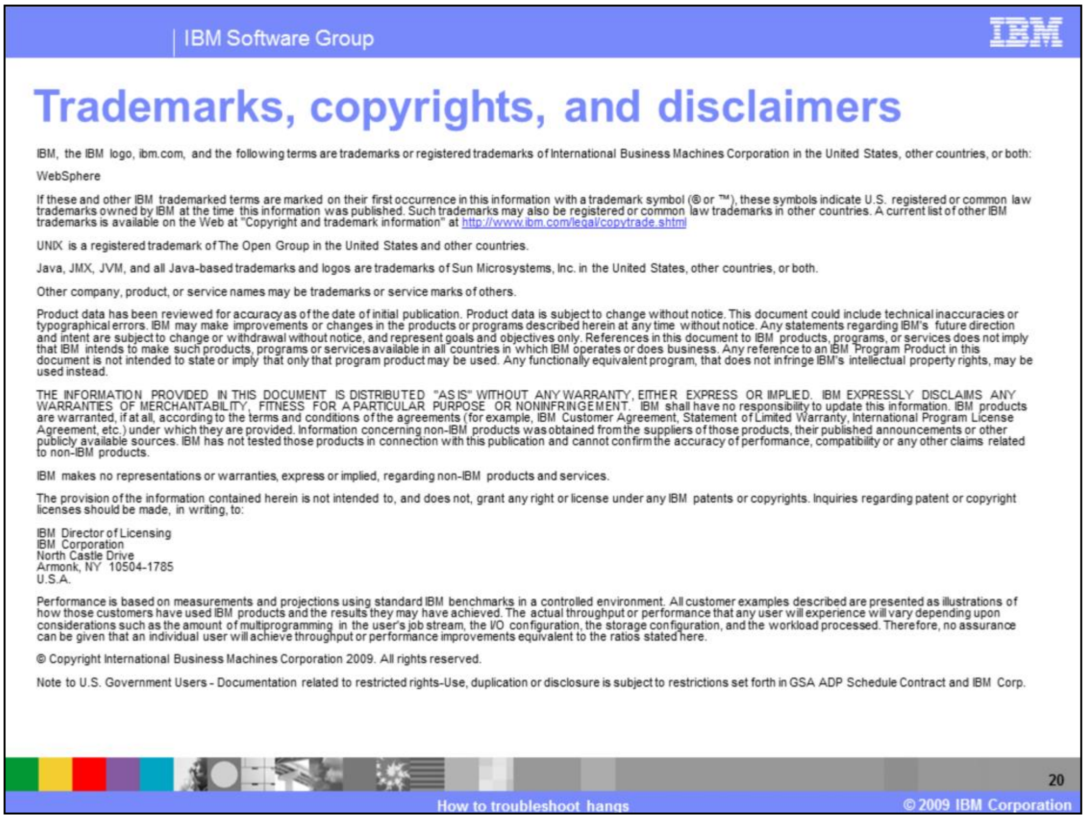
Now that you have completed this unit, you should be able to:

- Define a JVM hang
- Be able to identify a JVM hang
- Be able to capture a javacore and use it to troubleshoot a hang condition
- Configure and use the WebSphere Application Server Hung Thread Detection function
- Understand base use cases for the IBM Thread and Monitor Dump Analyzer tool

How to troubleshoot hangs

© 2009 IBM Corporation

Now that you have completed this unit, you should be able to define and identify a JVM hang, be able to capture a Java core and use it to troubleshoot a hang condition, configure and use the WebSphere Application Server Hung Thread Detection function, and understand base use cases for the IBM Thread and Monitor Dump Analyzer tool.



Trademarks, copyrights, and disclaimers