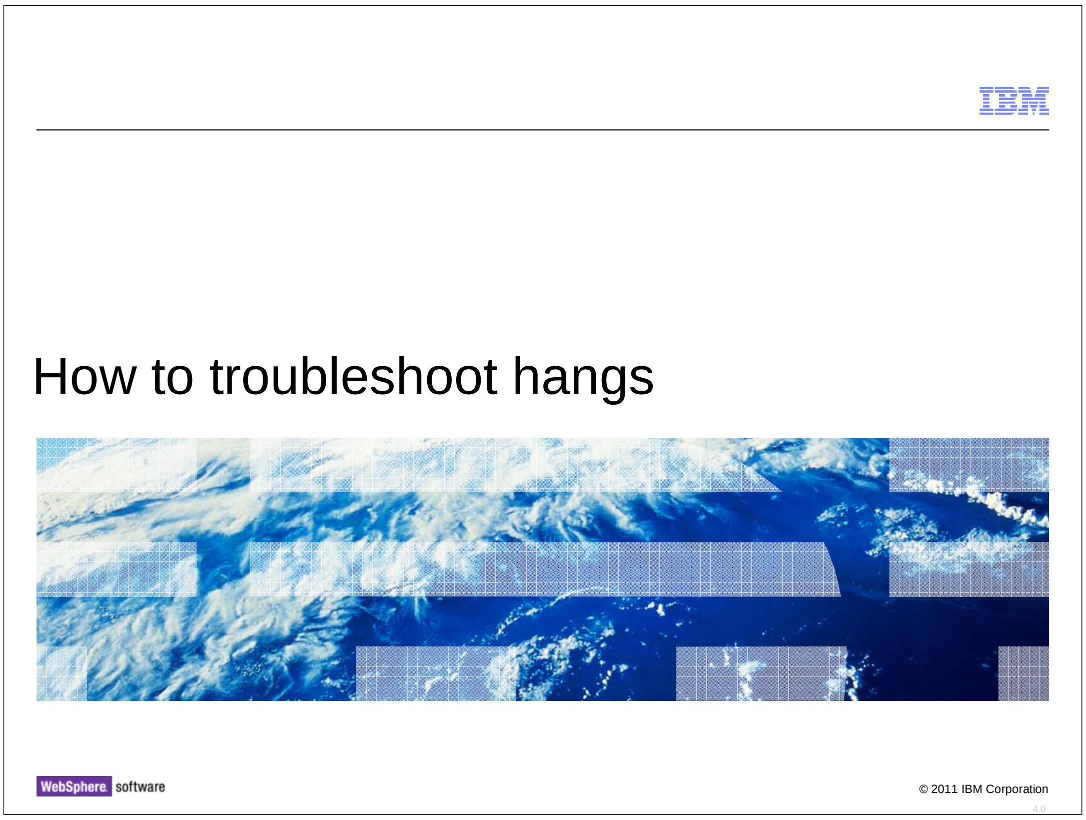
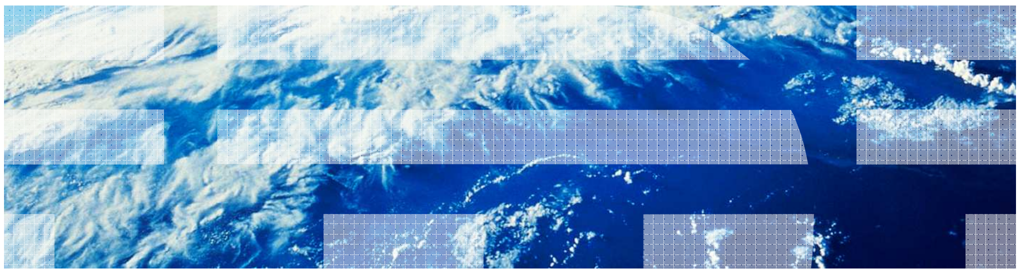
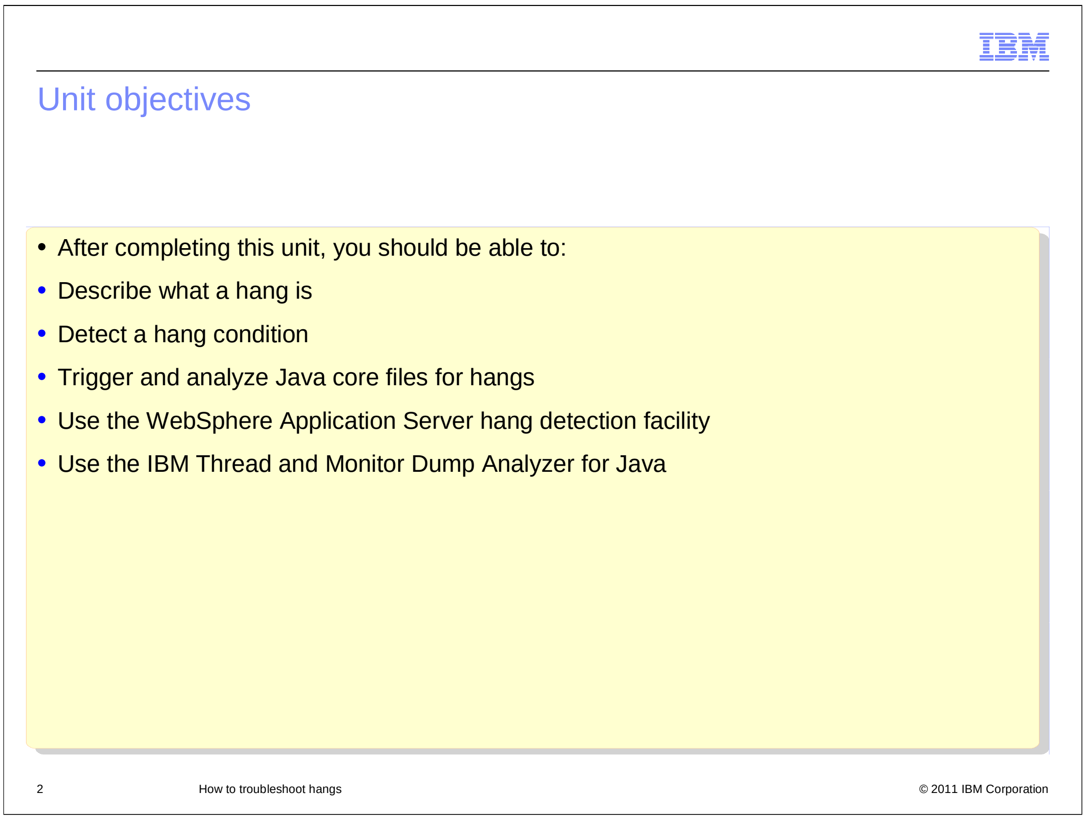


# How to troubleshoot hangs



This presentation will act as an introduction to troubleshooting hangs when using WebSphere Application Server version 7.0

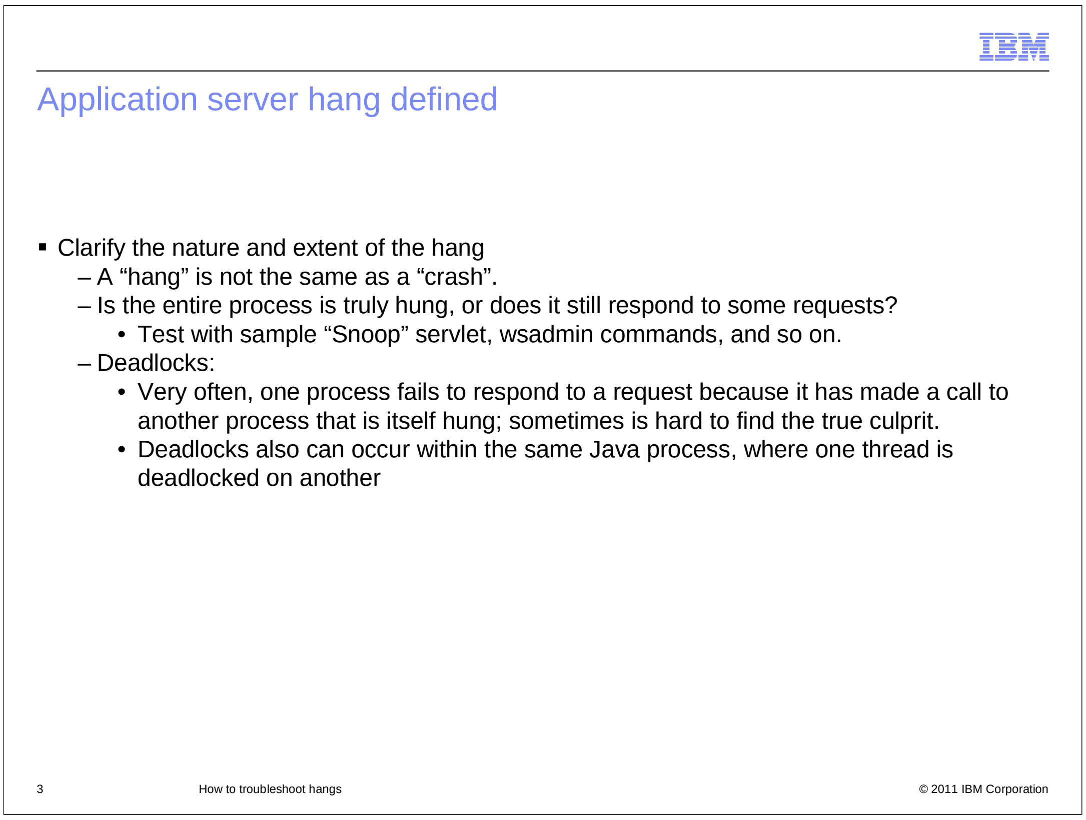


## Unit objectives

- After completing this unit, you should be able to:
- Describe what a hang is
- Detect a hang condition
- Trigger and analyze Java core files for hangs
- Use the WebSphere Application Server hang detection facility
- Use the IBM Thread and Monitor Dump Analyzer for Java

2 How to troubleshoot hangs © 2011 IBM Corporation

After completing his presentation you should be able to describe and detect a thread hang. Trigger a thread dump, and analyze it using IBM Thread and Monitor Dump Analyzer.



## Application server hang defined

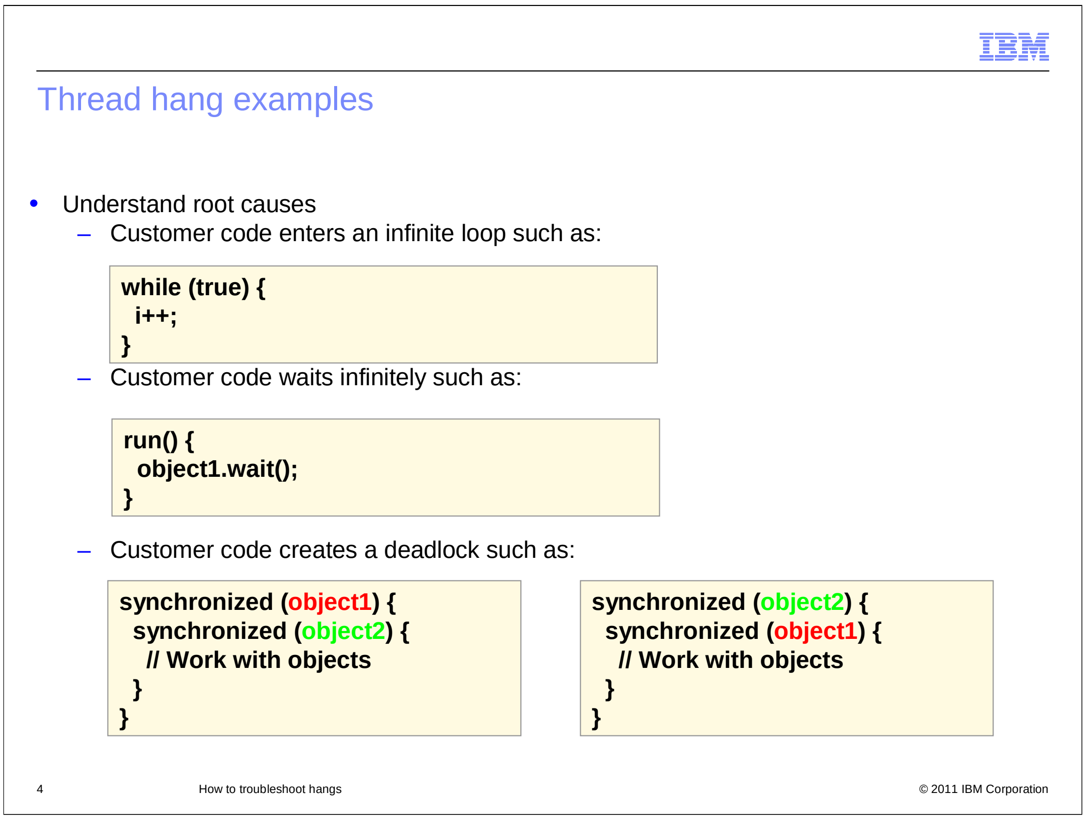
- Clarify the nature and extent of the hang
  - A “hang” is not the same as a “crash”.
  - Is the entire process is truly hung, or does it still respond to some requests?
    - Test with sample “Snoop” servlet, wsadmin commands, and so on.
  - Deadlocks:
    - Very often, one process fails to respond 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

3 How to troubleshoot hangs © 2011 IBM Corporation

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.

Deadlocks are often caused by one process or thread that is hung, which causes another thread to fail to respond to a request.

In Java, many deadlock conditions are a result of two  threads waiting on each other, requiring a lock on the object that the other already has obtained.



## Thread hang examples

- Understand root causes
  - Customer code enters an infinite loop such as:

```
while (true) {
  i++;
}
```

  - Customer code waits infinitely such as:

```
run() {
  object1.wait();
}
```

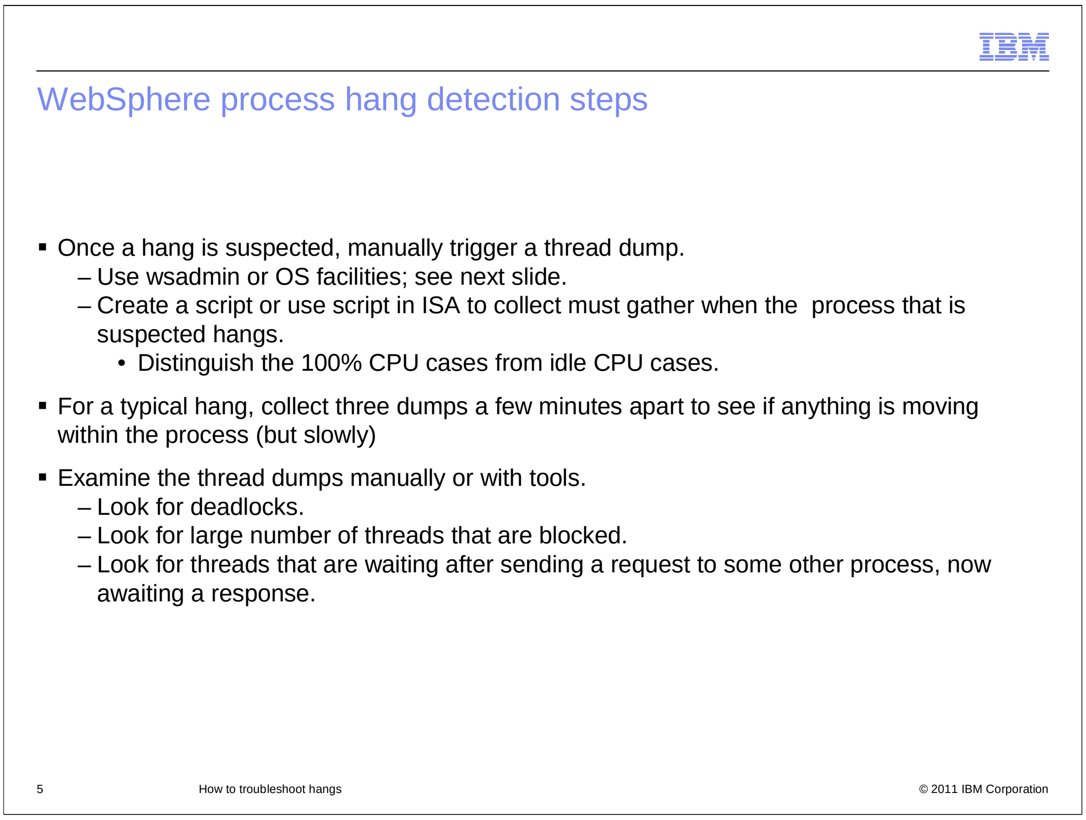
  - Customer code creates a deadlock such as:

```
synchronized (object1) {
  synchronized (object2) {
    // Work with objects
  }
}
```

```
synchronized (object2) {
  synchronized (object1) {
    // Work with objects
  }
}
```

4 How to troubleshoot hangs © 2011 IBM Corporation

This slide shows a common mistake that a development might make which can cause a thread hang or deadlock. In the top example, a simple logic error prevents the thread from doing further work due to an infinite loop condition. The middle example shows a thread being told to wait for outside notification that can or may not come. The last example on the slide demonstrates how a deadlock condition is formed.


## WebSphere process hang detection steps

- Once a hang is suspected, manually trigger a thread dump.
  - Use wsadmin or OS facilities; see next slide.
  - Create a script or use script in ISA to collect must gather when the  process that is suspected hangs.
    - Distinguish the 100% CPU cases from idle CPU cases.
- For a typical hang, collect three dumps a few minutes apart to see if anything is moving within the process (but slowly)
- Examine the thread dumps manually or with tools.
  - Look for deadlocks.
  - Look for large number of threads that are blocked.
  - Look for threads that are waiting after sending a request to some other process, now awaiting a response.

5 How to troubleshoot hangs © 2011 IBM Corporation

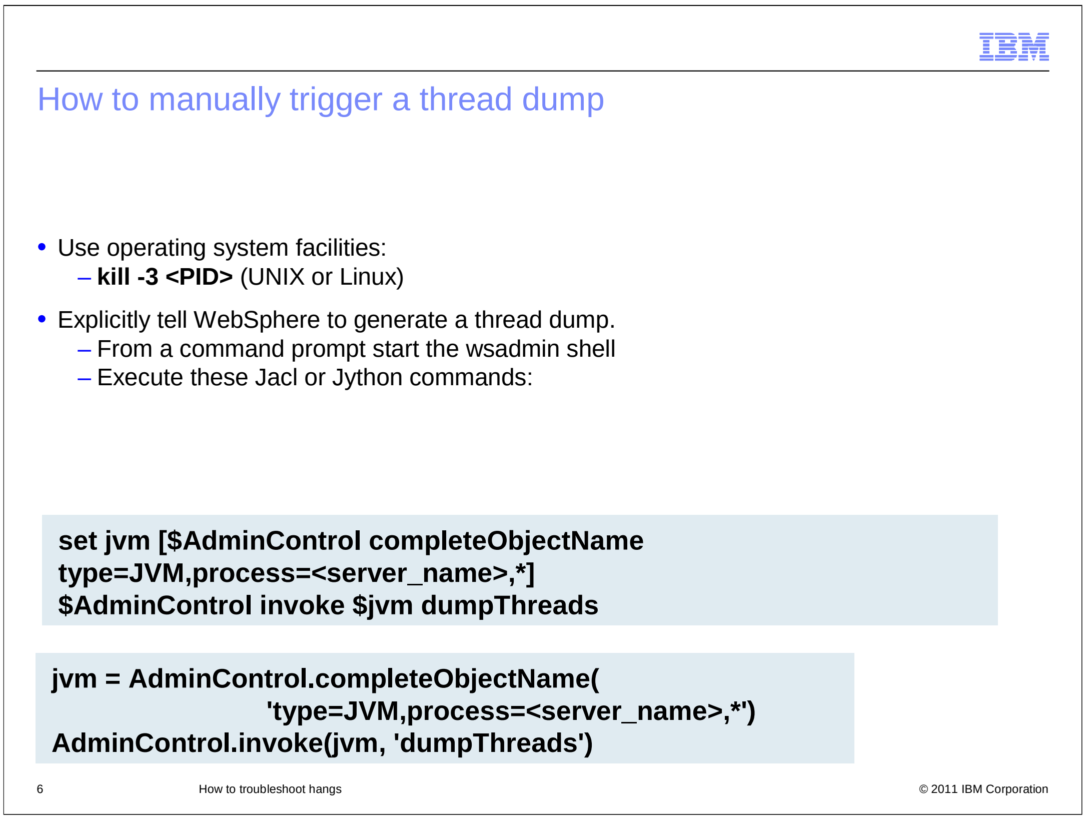
The basic problem determination method for hangs is to obtain one or, if possible, a series of thread dumps.

If the process is still responsive to wsadmin commands, then the wsadmin command should be able to trigger the dump.

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 to look for deadlocks or to see if threads are awaiting responses from other processes.

In newer IBM Virtual Machine for Java Platforms, the JVM will automatically perform deadlock detection and tell you if a deadlock has been detected. Look for the string "deadlock" in the thread dump.



## How to manually trigger a thread dump

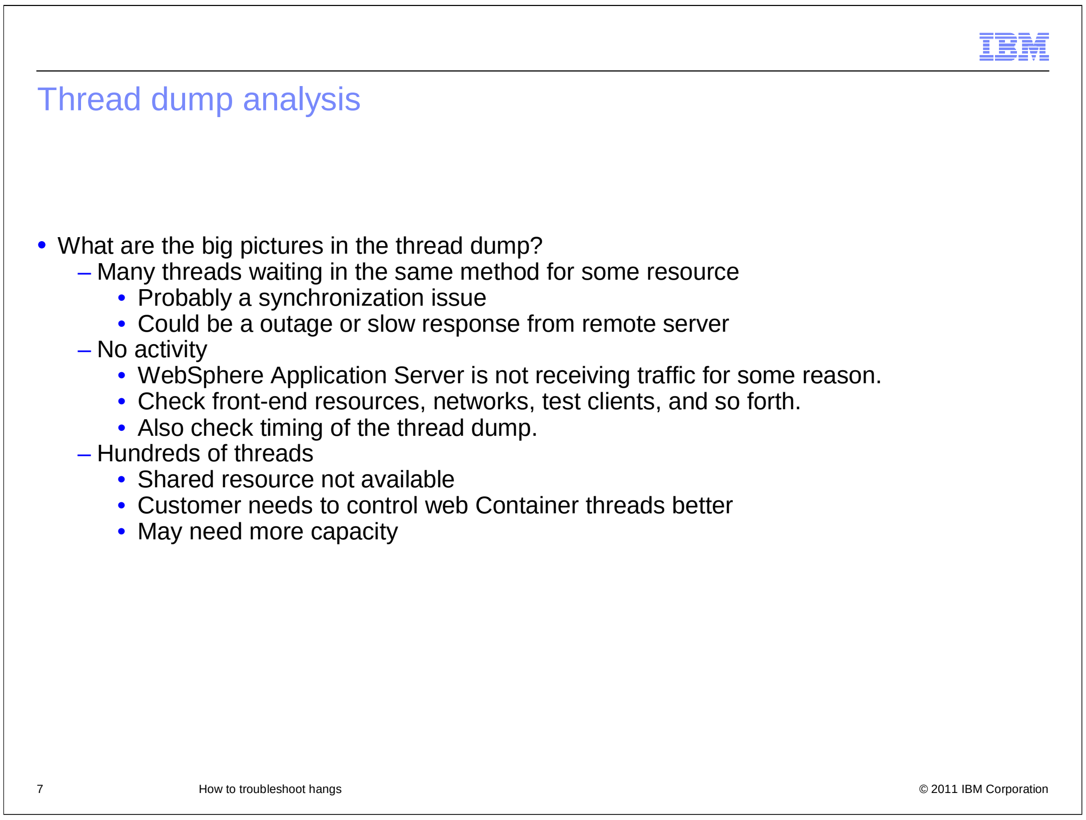
- Use operating system facilities:
  - **kill -3 <PID>** (UNIX or Linux)
- Explicitly tell WebSphere to generate a thread dump.
  - From a command prompt start the wsadmin shell
  - Execute these Jacl or Jython commands:

```
set jvm [$AdminControl completeObjectName
type=JVM,process=<server_name>,*]
$AdminControl invoke $jvm dumpThreads
```

```
jvm = AdminControl.completeObjectName(
                        'type=JVM,process=<server_name>,*')
AdminControl.invoke(jvm, 'dumpThreads')
```

6 How to troubleshoot hangs © 2011 IBM Corporation

This slide demonstrates several ways one can manually trigger a thread dump using the kill command or wsadmin scripting session.

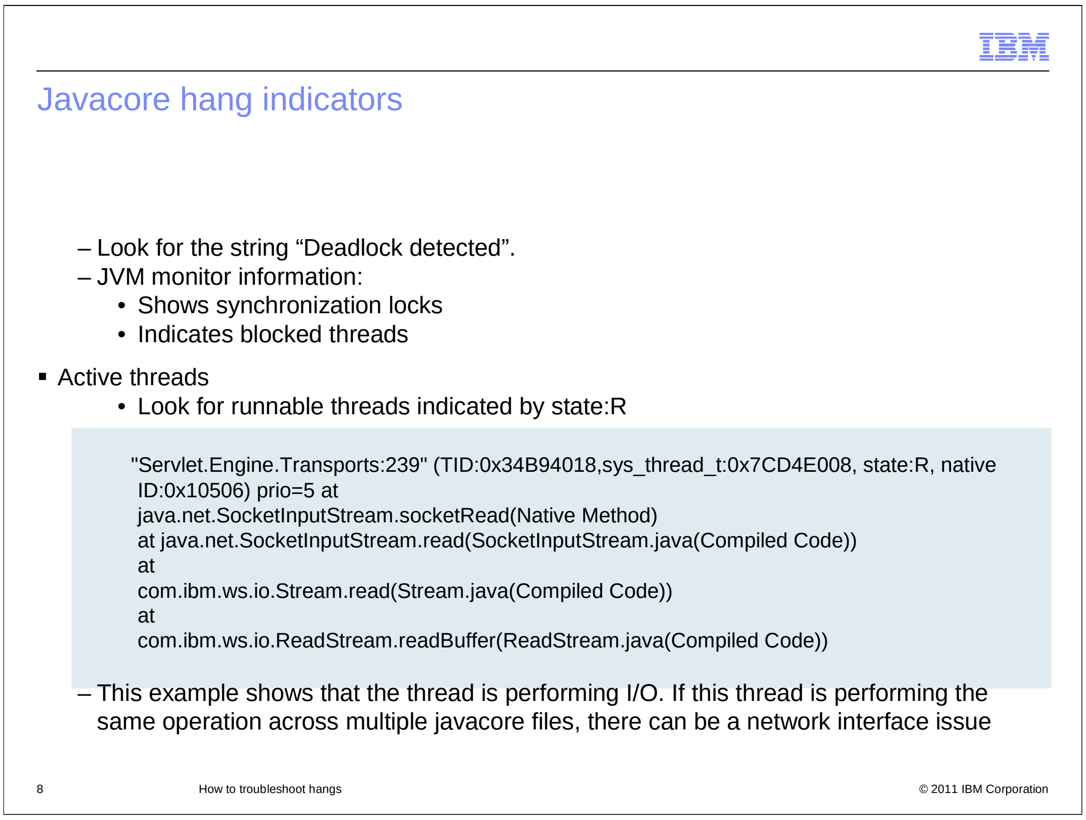


## Thread dump analysis

- What are the big pictures in the thread dump?
  - Many threads waiting in the same method for some resource
    - Probably a synchronization issue
    - Could be a outage or slow response from remote server
  - No activity
    - WebSphere Application Server is not receiving traffic for some reason.
    - Check front-end resources, networks, test clients, and so forth.
    - Also check timing of the thread dump.
  - Hundreds of threads
    - Shared resource not available
    - Customer needs to control web Container threads better
    - May need more capacity

7 How to troubleshoot hangs © 2011 IBM Corporation

After you have a few thread dumps to look at, look for conditions where many threads executing the same method call as this can indicate that the displayed method is delaying the threads. A lack of active web container threads can indicate that the server is not receiving any traffic. Additionally, too many threads in wait or blocked state can indicate a problem.



## Javacore hang indicators

- – Look for the string “Deadlock detected”.
- – JVM monitor information:
  - Shows synchronization locks
  - Indicates blocked threads

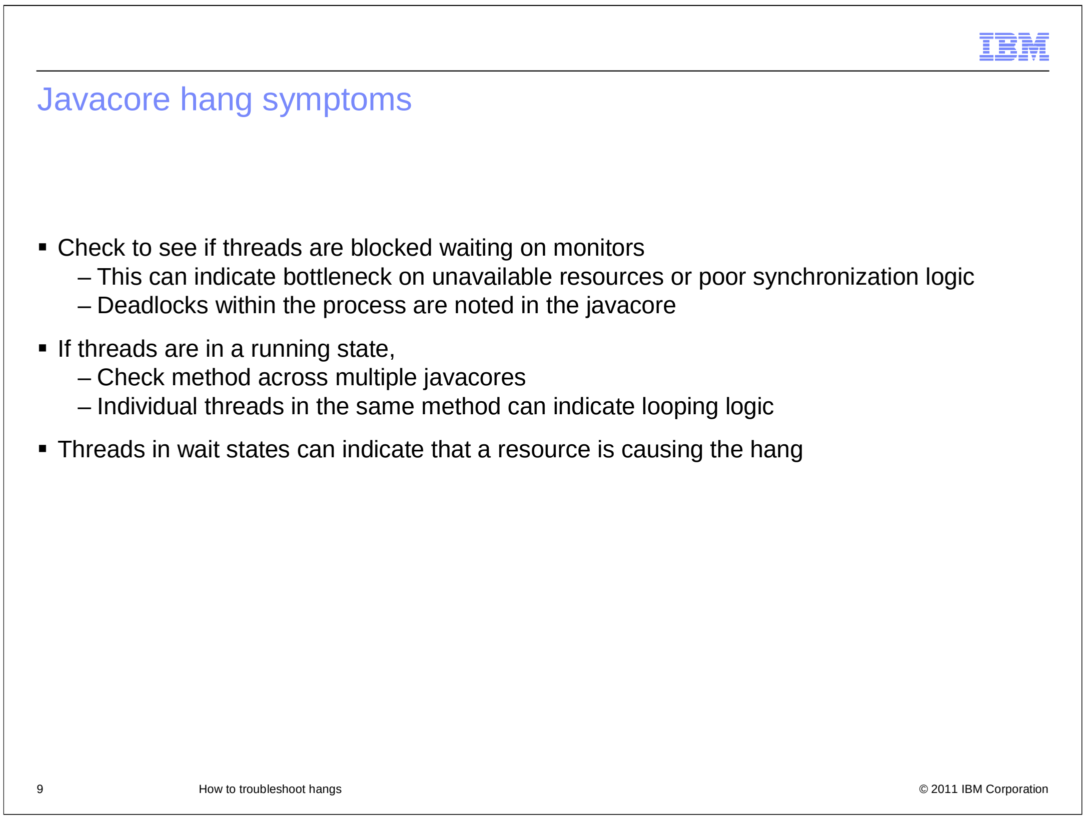
- Active threads
  - Look for runnable threads indicated by state:R

```
"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 can be a network interface issue

8 How to troubleshoot hangs © 2011 IBM Corporation

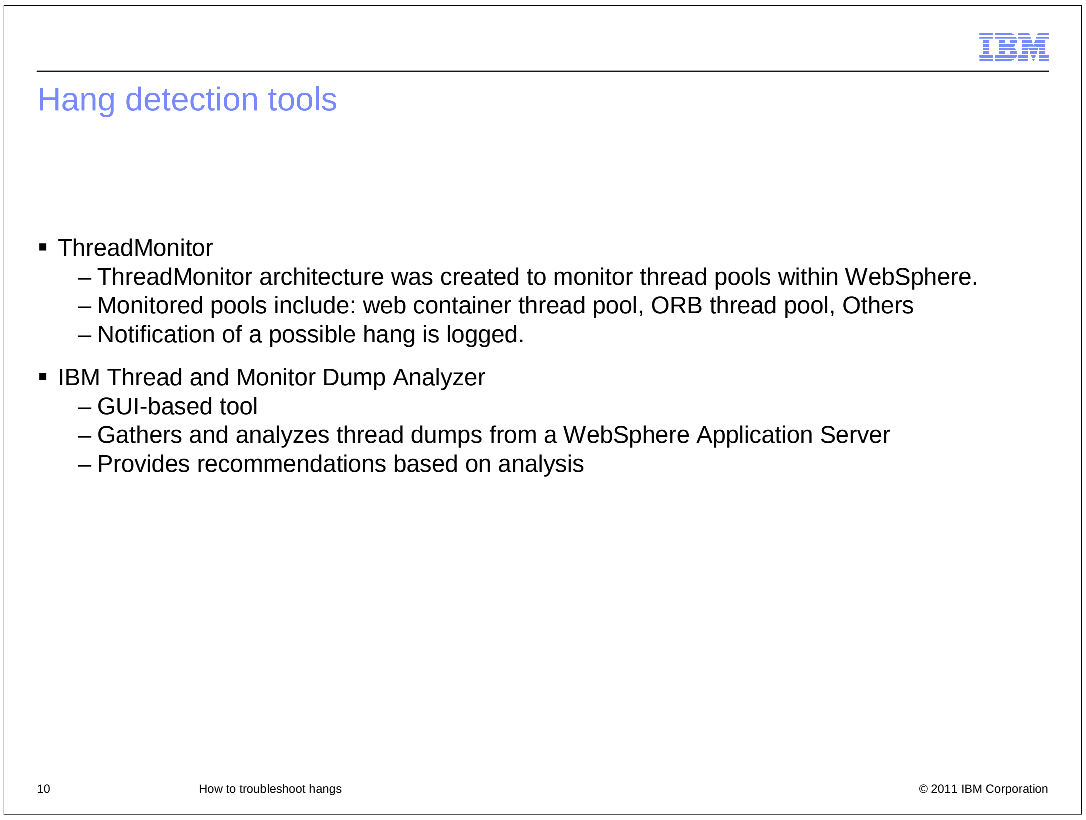
The monitor information in the thread dump shows what synchronization locks are held by which threads. A “Deadlock detected” message in the dump provides a clear indication of a deadlock condition in the JVM. The monitor information also shows which threads are blocked by monitors. This information is useful for determining the cause of a deadlocked or hung JVM.



## Javacore hang symptoms

- Check to see if threads are blocked waiting on monitors
  - This can indicate bottleneck on unavailable resources or poor synchronization logic
  - Deadlocks within the process are noted in the javacore
- If threads are in a running state,
  - Check method across multiple javacores
  - Individual threads in the same method can indicate looping logic
- Threads in wait states can indicate that a resource is causing the hang

There are some fairly easy-to-spot symptoms that present themselves in a thread dump that can be used to identify the source of the hang or performance degradation. For example, if you see a single thread holding up several other threads, you have a good place to start looking for the root cause. In other cases, you can see a thread running the same code across all of the thread dumps you have to review. This might indicate that the running code is having a problem that is causing the hung thread.

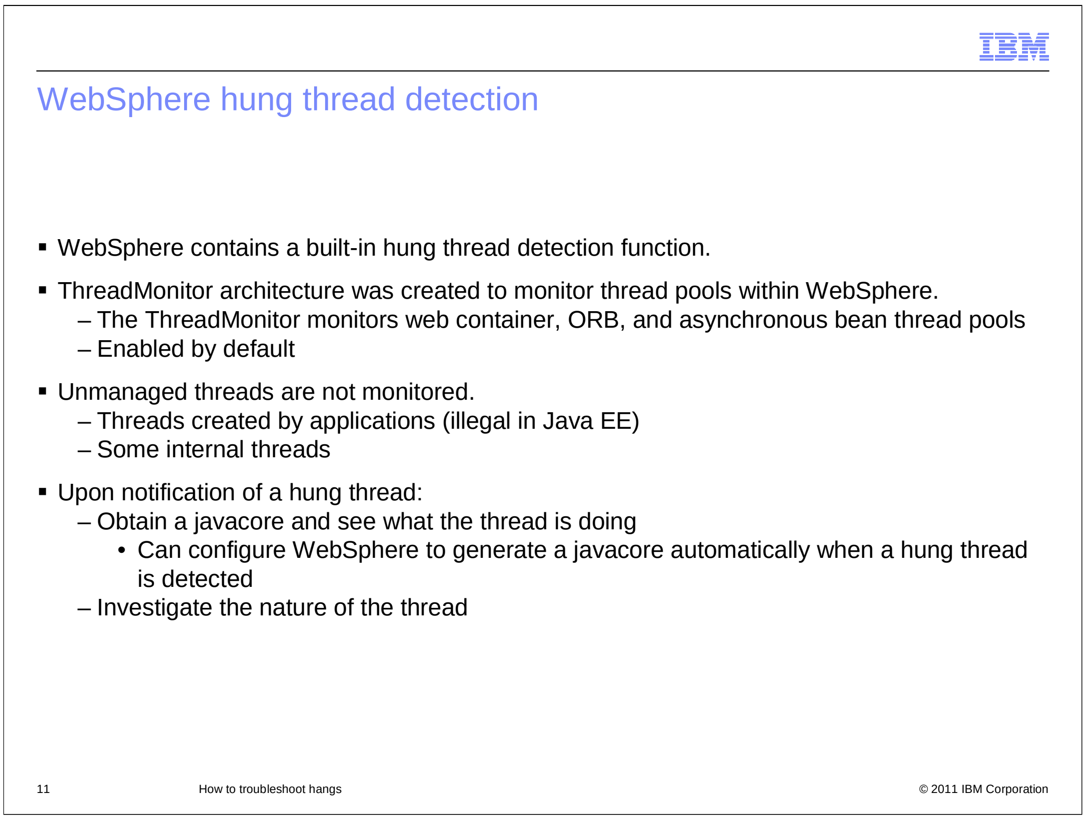


## Hang detection tools

- ThreadMonitor
  - ThreadMonitor architecture was created to monitor thread pools within WebSphere.
  - Monitored pools include: web container thread pool, ORB thread pool, Others
  - Notification of a possible hang is logged.
- IBM Thread and Monitor Dump Analyzer
  - GUI-based tool
  - Gathers and analyzes thread dumps from a WebSphere Application Server
  - Provides recommendations based on analysis

10 How to troubleshoot hangs © 2011 IBM Corporation

IBM has several tools available to help you analyze thread dumps. The ThreadMonitor component within WebSphere Application Server will provide notifications in the logs when a possible hung thread is detected. For post mortem analysis, the IBM Thread and Monitor Dump Analyzer (TMDA) tool can be used. TMDA is a GUI-based tool that is used to analyze thread dumps from both IBM and Oracle Hotspot JVMs and provide possible recommendations on the cause.

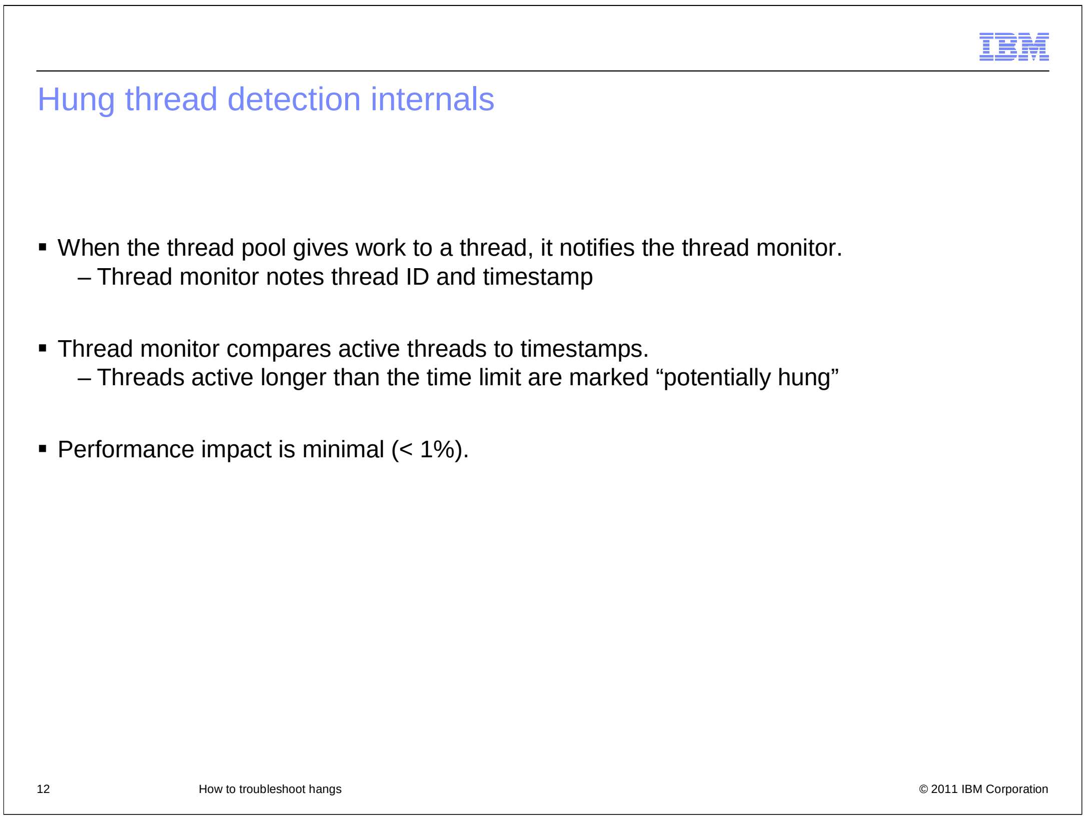


## WebSphere hung thread detection

- 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 asynchronous bean thread pools
  - Enabled by default
- Unmanaged threads are not monitored.
  - Threads created by applications (illegal in Java EE)
  - Some internal threads
- Upon notification of a hung thread:
  - Obtain a javacore and see what the thread is doing
    - Can configure WebSphere to generate a javacore automatically when a hung thread is detected
  - Investigate the nature of the thread

WebSphere Application Server contains a built-in hung thread detection function called the Thread Monitor. It monitors the web container, Object Request Broker, and Asynchronous Bean thread pools, and is enabled by default. Threads not created from WebSphere-managed thread pools 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.



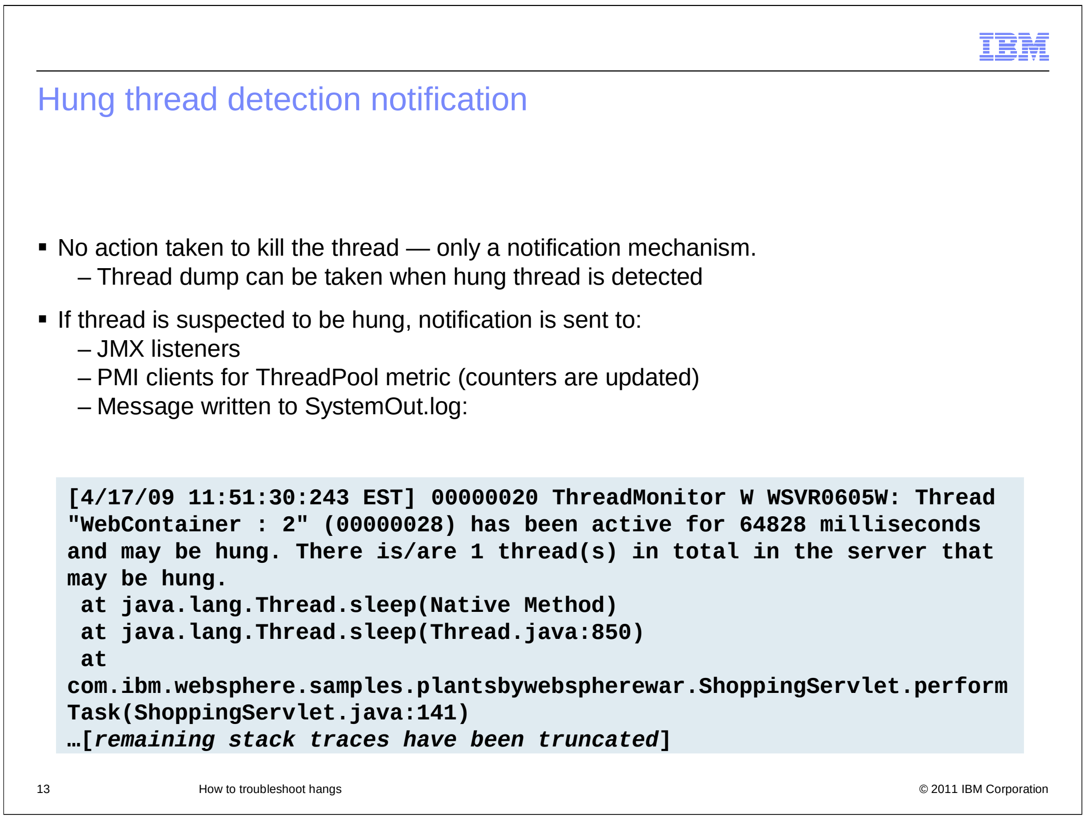
## Hung thread detection internals

- 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%).

12 How to troubleshoot hangs © 2011 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.



## Hung thread detection notification

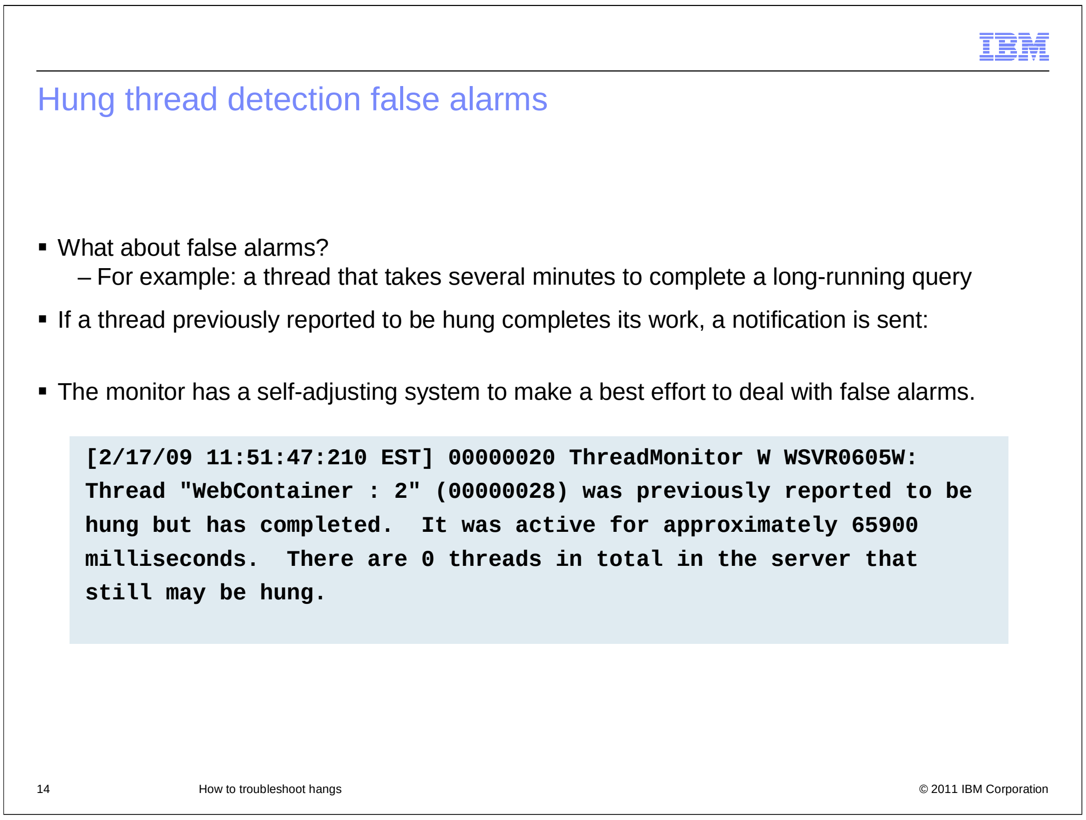
- No action taken to kill the thread — only a notification mechanism.
  - Thread dump can be taken when hung thread is detected
- If thread is suspected to be hung, notification is sent to:
  - JMX listeners
  - PMI clients for ThreadPool metric (counters are updated)
  - Message written to SystemOut.log:

```
[4/17/09 11:51:30:243 EST] 00000020 ThreadMonitor W WSVR0605W: Thread
"WebContainer : 2" (00000028) has been active for 64828 milliseconds
and may be hung. There is/are 1 thread(s) in total in the server that
may be hung.
 at java.lang.Thread.sleep(Native Method)
 at java.lang.Thread.sleep(Thread.java:850)
 at
com.ibm.websphere.samples.plantsbywebspherewar.ShoppingServlet.perform
Task(ShoppingServlet.java:141)
…[remaining stack traces have been truncated]
```

13 How to troubleshoot hangs © 2011 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, and has a message identifier of WSVR0605W, also shows the thread name, the approximate time that the thread has been active, and the total number of threads which can be hung.



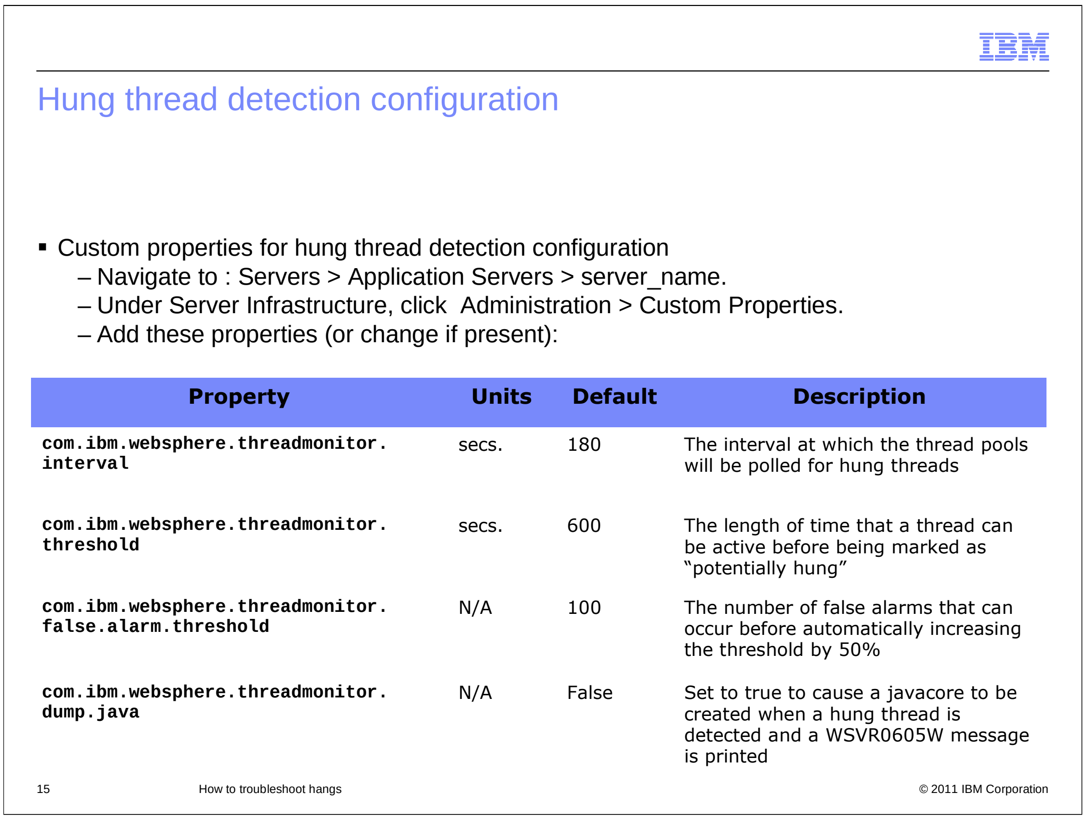
## Hung thread detection false alarms

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

```
[2/17/09 11:51:47:210 EST] 00000020 ThreadMonitor W WSVR0605W:
Thread "WebContainer : 2" (00000028) was previously reported to be
hung but has completed.  It was active for approximately 65900
milliseconds.  There are 0 threads in total in the server that
still may be hung.
```

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.

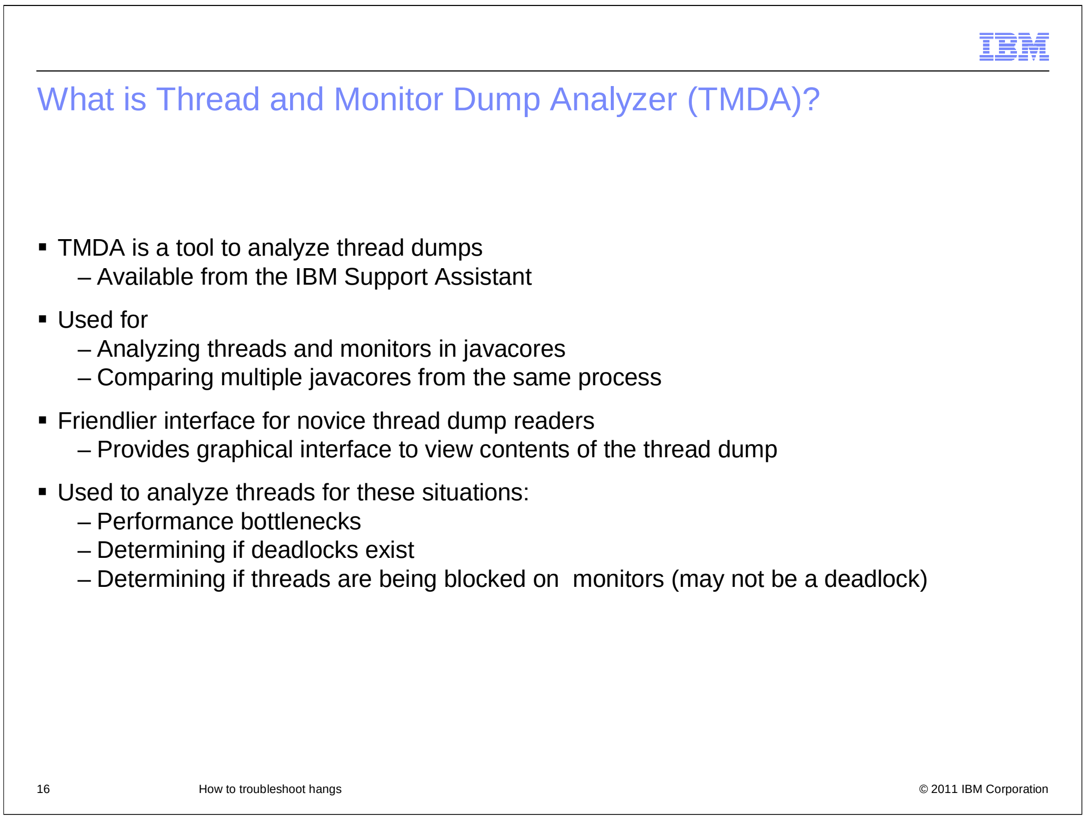


## Hung thread detection configuration

- Custom properties for hung thread detection configuration
  - Navigate to : Servers > Application Servers > server_name.
  - Under Server Infrastructure, click  Administration > Custom Properties.
  - Add these properties (or change if present):

| Property | Units | Default | Description |
|---|---|---|---|
| `com.ibm.websphere.threadmonitor.interval` | secs. | 180 | The interval at which the thread pools will be 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% |
| `com.ibm.websphere.threadmonitor.dump.java` | N/A | False | Set to true to cause a javacore to be created when a hung thread is detected and a WSVR0605W message is printed |

The hang detection policy can be configured by creating custom properties for the application server from the AdminConsole. This slide shows the default settings of the custom properties unless specified otherwise.



## What is Thread and Monitor Dump Analyzer (TMDA)?

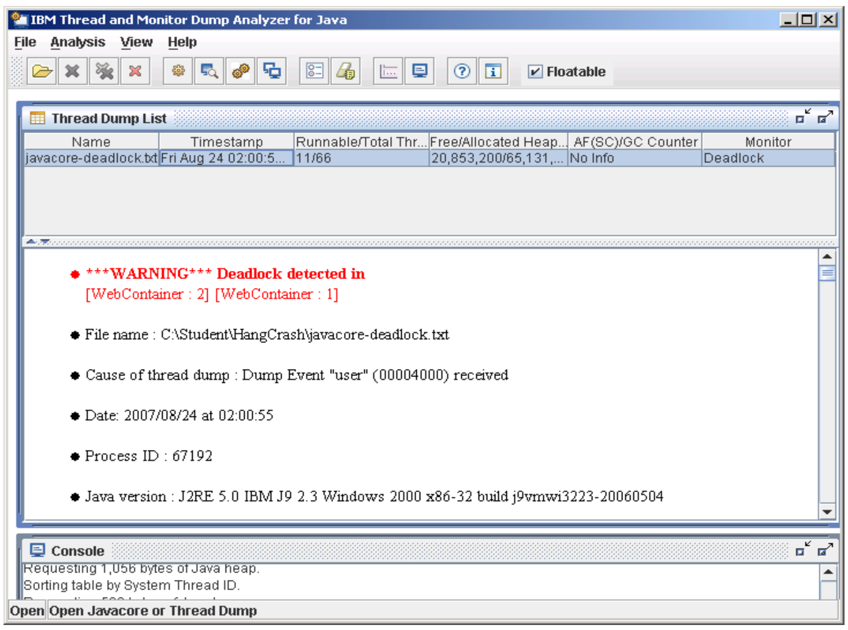
- TMDA is a tool to analyze thread dumps
  - Available from the IBM Support Assistant
- Used for
  - Analyzing threads and monitors in javacores
  - Comparing multiple javacores from the same process
- Friendlier interface for novice thread dump readers
  - Provides graphical interface to view contents of the thread dump
- Used to analyze threads for these situations:
  - Performance bottlenecks
  - Determining if deadlocks exist
  - Determining if threads are being blocked on  monitors (may not be a deadlock)

16 How to troubleshoot hangs © 2011 IBM Corporation

The Thread and Monitor Dump Analyzer is an IBM Support tool available through the IBM Support Assistant workbench.

It is designed to simplify the act of analyzing thread dumps and is designed so that novice troubleshooters and experts alike can use the tool to analyze thread dumps.

## TMDA: open a thread dump


IBM Thread and Monitor Dump Analyzer for Java
File Analysis View Help
Floatable
Thread Dump List
Console
Open
Open Javacore or Thread Dump

| 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 |

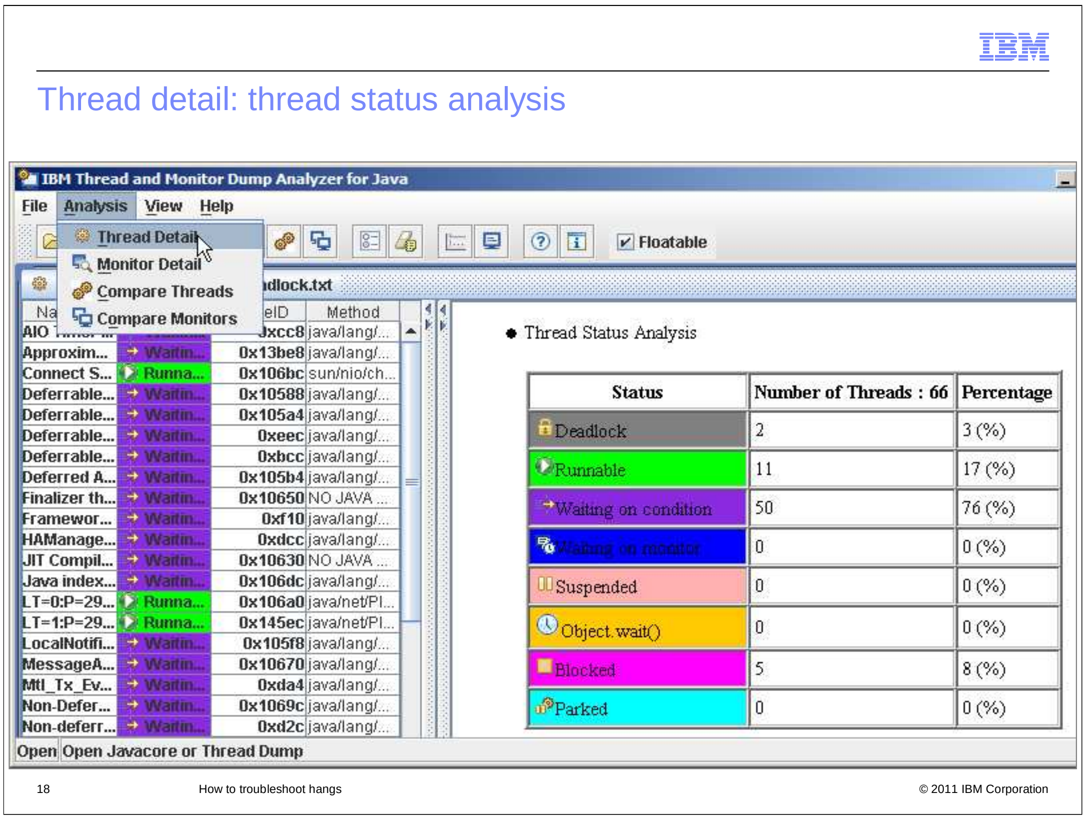
- ***WARNING*** Deadlock detected in [WebContainer : 2] [WebContainer : 1]
- File name : C:\Student\HangCrash\javacore-deadlock.txt
- Cause of thread dump : Dump Event "user" (00004000) received
- Date: 2007/08/24 at 02:00:55
- Process ID : 67192
- Java version : J2RE 5.0 IBM J9 2.3 Windows 2000 x86-32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

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

17 How to troubleshoot hangs © 2011 IBM Corporation

You can search the local file system for one or more thread dump 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 thread dump file such as file name, cause of the dump, data, process identifier, Java version, Java heap information, and much more.



IBM

## Thread detail: thread status analysis

IBM Thread and Monitor Dump Analyzer for Java

| 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 (%) |

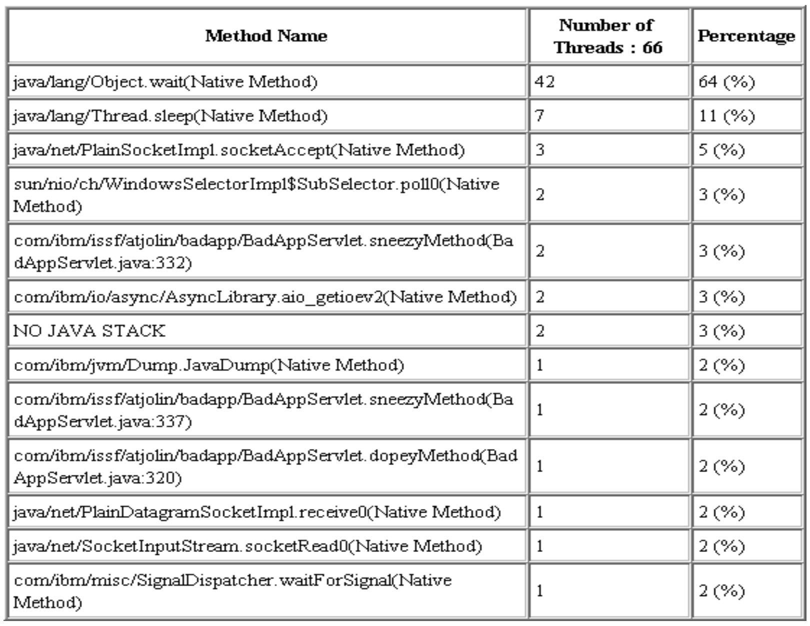
18 How to troubleshoot hangs © 2011 IBM Corporation

The Analysis menu allows you to display thread and monitor details for a single thread dump. If you open multiple thread dumps, 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.

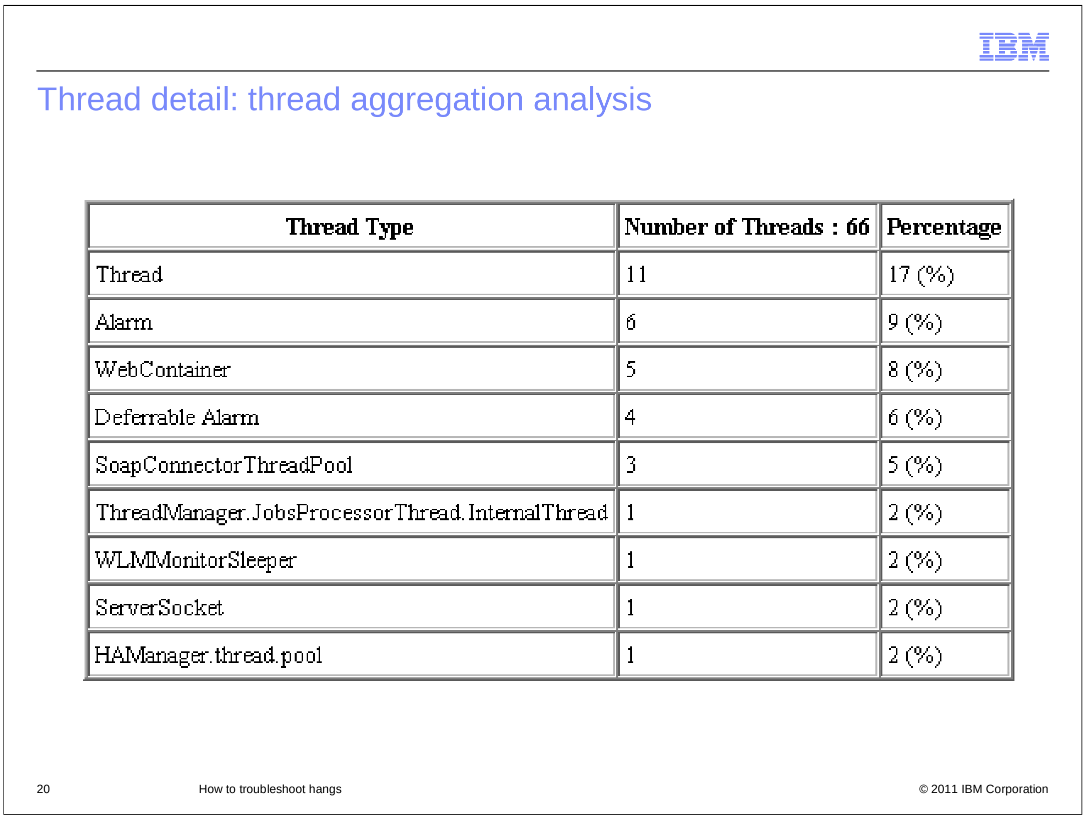
The thread status analysis shows the number of threads in each state: Deadlocked, Runnable, Blocked, and so forth. Threads are sorted by thread name. Thread Detail View provides the thread name, the state of a thread, the method name, the Java stack trace, and the native stack trace;

## 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.

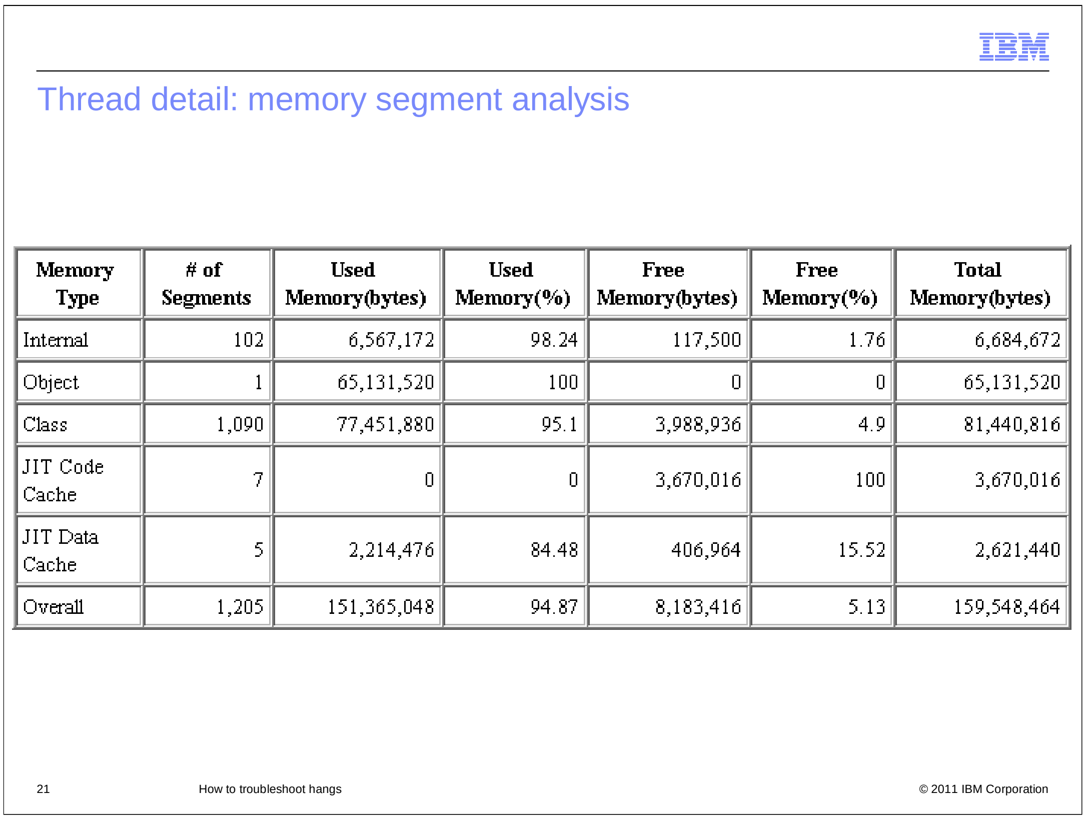


## Thread detail: thread aggregation analysis

| 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 (%) |

20 How to troubleshoot hangs © 2011 IBM Corporation

The thread aggregation analysis view details the types of threads that were seen in the dump.



## Thread detail: memory segment analysis

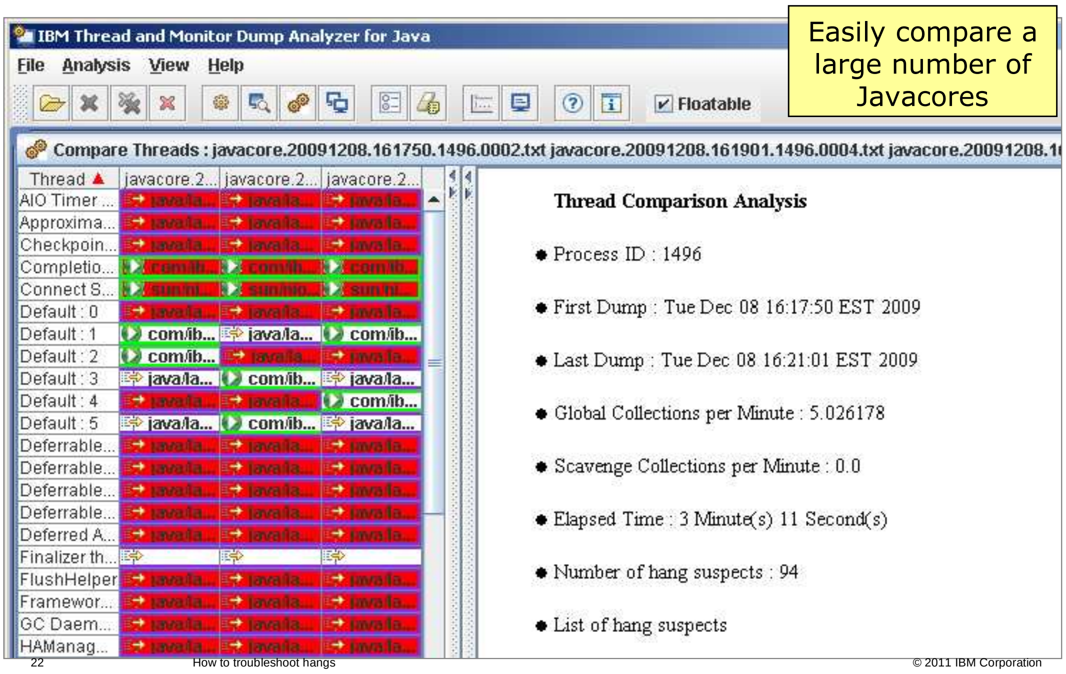
| Memory Type | # of Segments | Used Memory(bytes) | Used Memory(%) | Free Memory(bytes) | Free Memory(%) | Total Memory(bytes) |
|---|---|---|---|---|---|---|
| Internal | 102 | 6,567,172 | 98.24 | 117,500 | 1.76 | 6,684,672 |
| Object | 1 | 65,131,520 | 100 | 0 | 0 | 65,131,520 |
| Class | 1,090 | 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 |

21 How to troubleshoot hangs © 2011 IBM Corporation

This slide shows sample output 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

Easily compare a large number of Javacores



Compare Threads : javacore.20091208.161750.1496.0002.txt javacore.20091208.161901.1496.0004.txt javacore.20091208.1

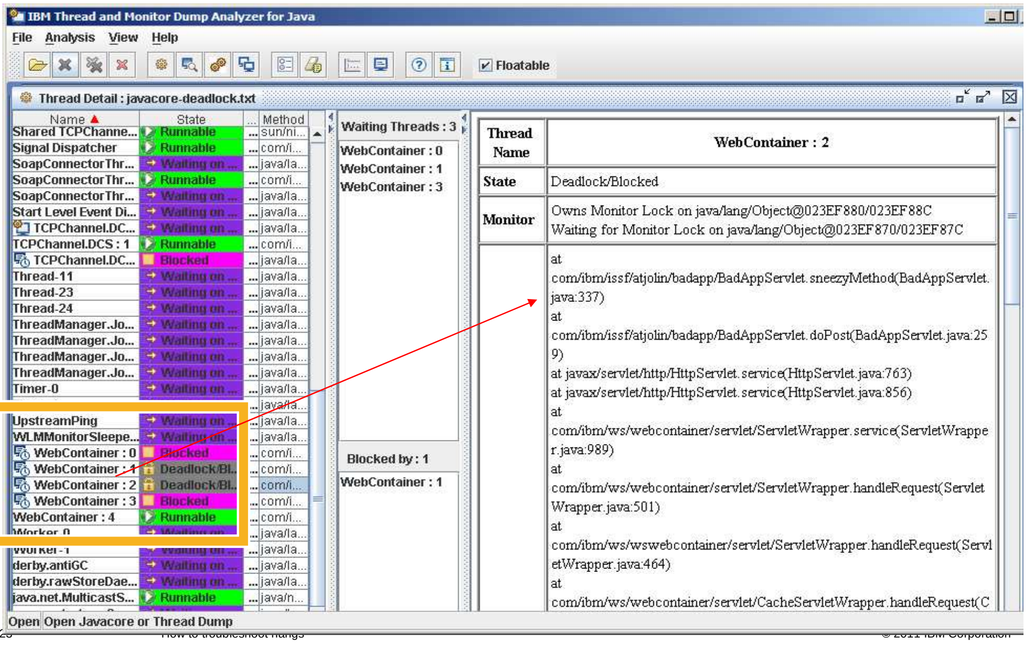
| Thread ▲ | javacore.2... | javacore.2... | javacore.2... |
|---|---|---|---|
| AIO Timer ... | java/la... | java/la... | java/la... |
| Approxima... | java/la... | java/la... | java/la... |
| Checkpoin... | java/la... | java/la... | java/la... |
| Completio... | com/ib... | com/ib... | com/ib... |
| Connect S... | sun/ni... | sun/nio... | sun/ni... |
| Default : 0 | java/la... | java/la... | java/la... |
| Default : 1 | com/ib... | java/la... | com/ib... |
| Default : 2 | com/ib... | java/la... | java/la... |
| Default : 3 | java/la... | com/ib... | java/la... |
| Default : 4 | java/la... | java/la... | com/ib... |
| Default : 5 | java/la... | com/ib... | java/la... |
| Deferrable... | java/la... | java/la... | java/la... |
| Deferrable... | java/la... | java/la... | java/la... |
| Deferrable... | java/la... | java/la... | java/la... |
| Deferrable... | java/la... | java/la... | java/la... |
| Deferred A... | java/la... | java/la... | java/la... |
| Finalizer th... | | | |
| FlushHelper | java/la... | java/la... | java/la... |
| Framewor... | java/la... | java/la... | java/la... |
| GC Daem... | java/la... | java/la... | java/la... |
| HAManag... | java/la... | java/la... | java/la... |

**Thread Comparison Analysis**

- Process ID : 1496
- First Dump : Tue Dec 08 16:17:50 EST 2009
- Last Dump : Tue Dec 08 16:21:01 EST 2009
- Global Collections per Minute : 5.026178
- Scavenge Collections per Minute : 0.0
- Elapsed Time : 3 Minute(s) 11 Second(s)
- Number of hang suspects : 94
- List of hang suspects

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
Open
Open Javacore or Thread Dump

| 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... |
| | | ... | 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 |

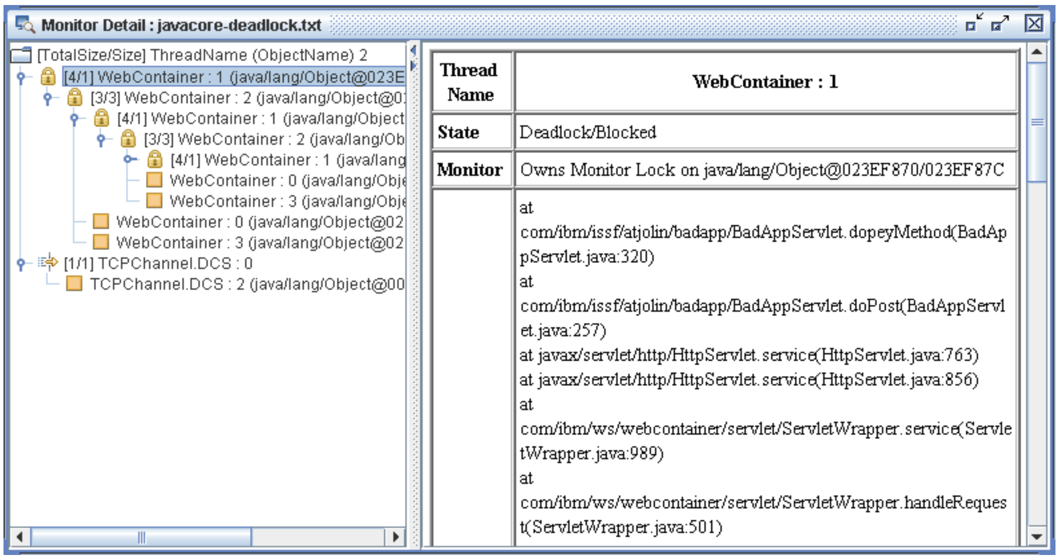
| Blocked by : 1 |
|---|
| WebContainer : 1 |

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

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.

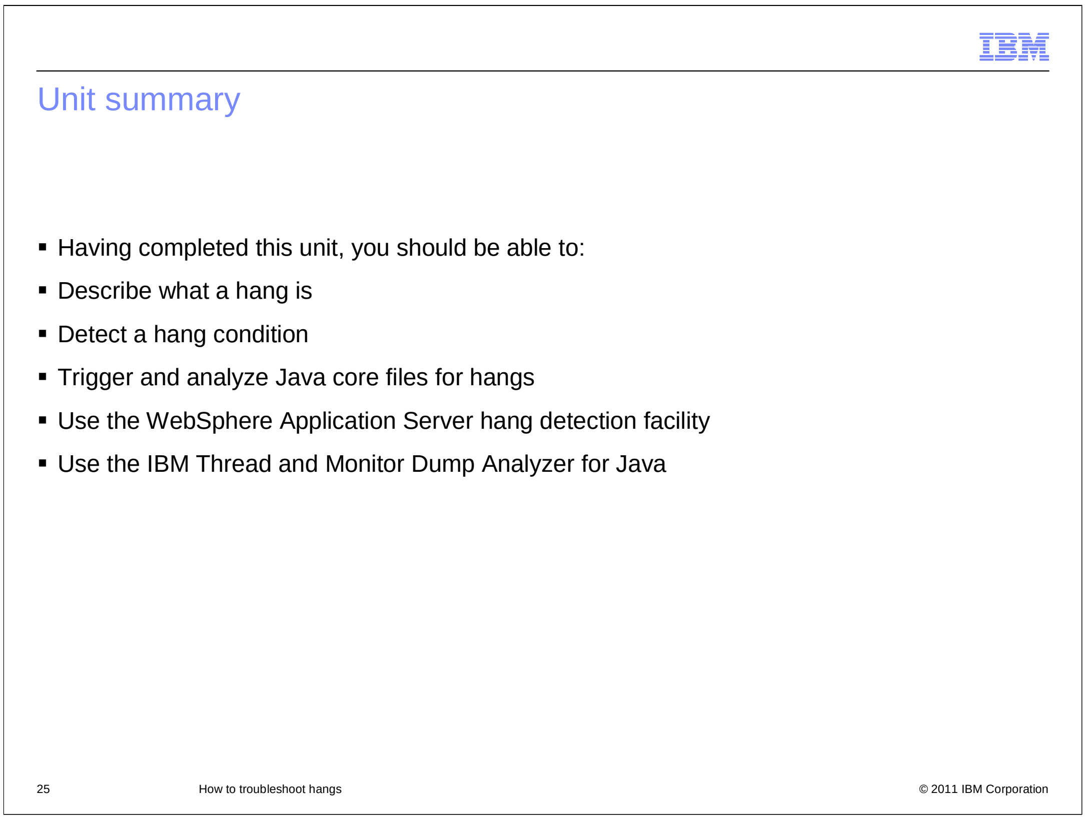
## Thread analysis: monitor details



```
[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)<br>at com/ibm/issf/atjolin/badapp/BadAppServlet.doPost(BadAppServlet.java:257)<br>at javax/servlet/http/HttpServlet.service(HttpServlet.java:763)<br>at javax/servlet/http/HttpServlet.service(HttpServlet.java:856)<br>at com/ibm/ws/webcontainer/servlet/ServletWrapper.service(ServletWrapper.java:989)<br>at com/ibm/ws/webcontainer/servlet/ServletWrapper.handleRequest(ServletWrapper.java:501) |

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.

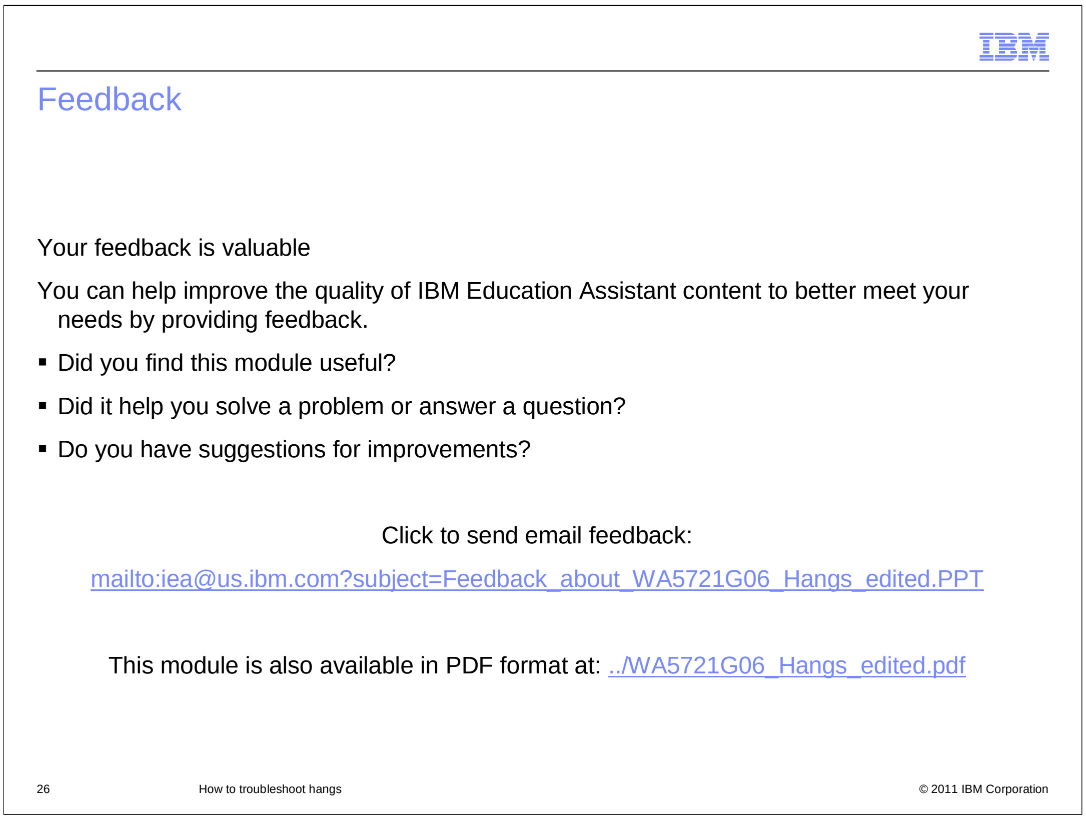


## Unit summary

- Having completed this unit, you should be able to:
- Describe what a hang is
- Detect a hang condition
- Trigger and analyze Java core files for hangs
- Use the WebSphere Application Server hang detection facility
- Use the IBM Thread and Monitor Dump Analyzer for Java

25 How to troubleshoot hangs © 2011 IBM Corporation

Having completed this presentation, you should be able to define and identify a JVM hang, be able to capture a thread dump 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.



## 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?

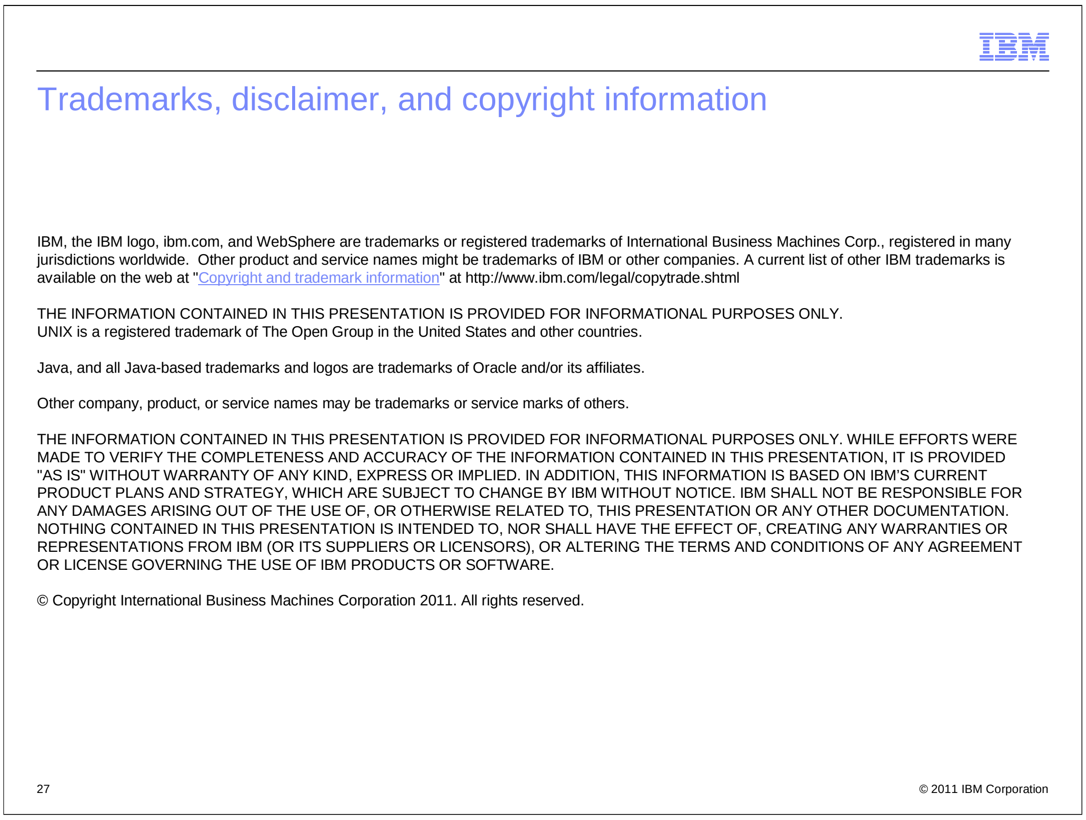
Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WA5721G06_Hangs_edited.PPT

This module is also available in PDF format at: ../WA5721G06_Hangs_edited.pdf

26 How to troubleshoot hangs © 2011 IBM Corporation

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



## Trademarks, disclaimer, and copyright information