Dealing with Stuck Threads in WebLogic

WEBLOGIC, WLDF

Dealing with Stuck Threads in WebLogic

Radu Dobrinescu June 10, 2014 4 Comments

Most WebLogic administrators have to deal with the *Stuck Threads problem* from time to time. Stuck threads are JVM threads that have been running for more than a certain configurable time (default 600 seconds).

These threads cannot be killed or cleared, but there are still things that an admin can do to alleviate the impact of such undesirable behavior. And the final solution should be finding the root cause of the stuck threads in order to avoid them completely, rather than react when this condition appears. But analysis can be tedious and can take a long time, especially when waiting for an application provider to fix their code.

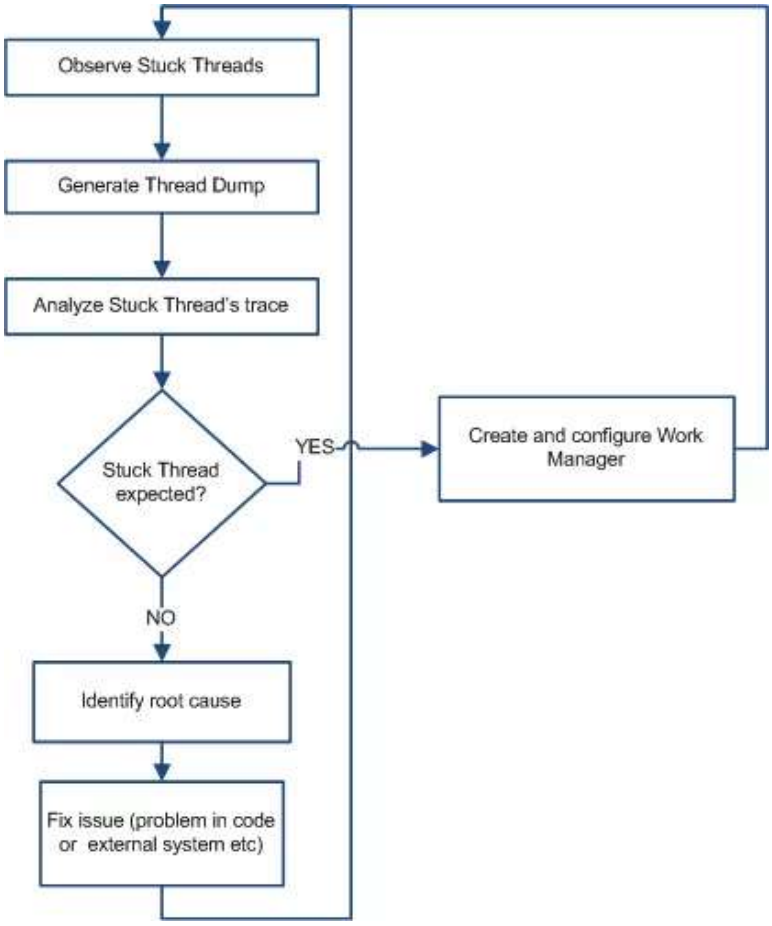
In the meantime, the admin can and should automate the tasks of collecting diagnostic data and recovering from a complete server hang eventually caused by an increasing number of Stuck Threads. This posts details how.

First thing to check when there are Stuck Threads, is to generate a Thread Dump and analyze the threads that are being stuck. It could be that the stuck threads are generated by a problem in the code, by the unavailability of an external system, or it can even be an expected behavior, in the case of some long lasting operations in the application.

A suggested Stuck Thread analysis process looks like this:

TOP POSTS & PAGES

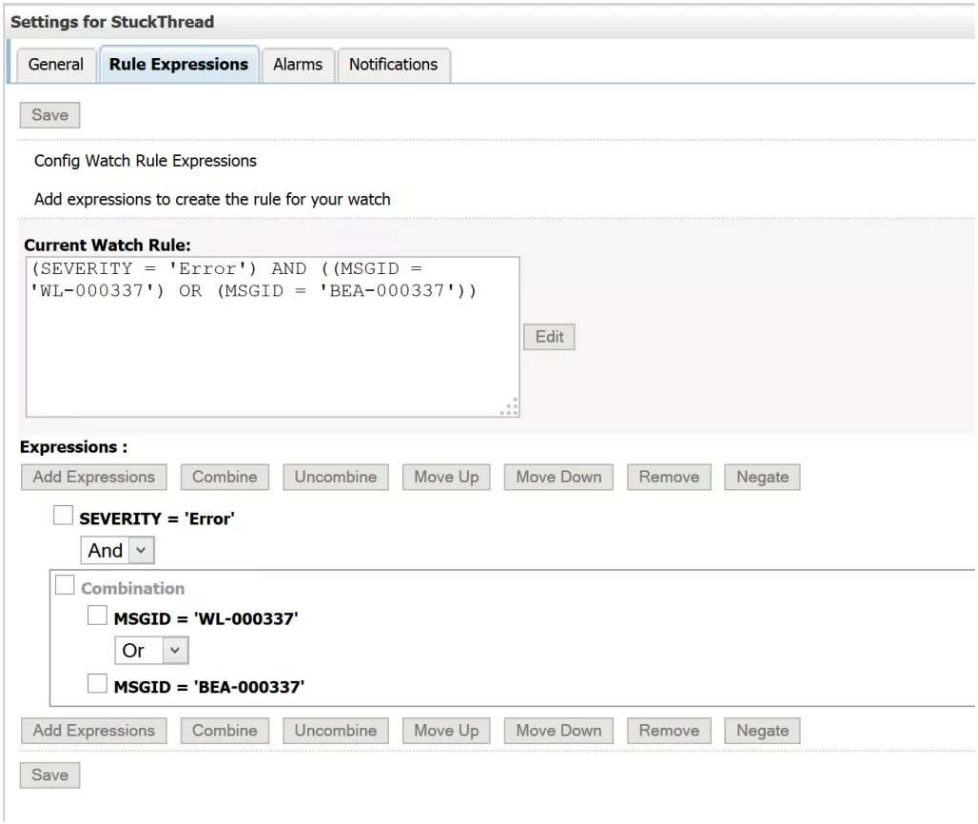
Unable to start WLST: "Could Not Find or Load Main Class weblogic.WLST"
Dealing with Stuck Threads in WebLogic
WebLogic AdminServer startup and shutdown scripts
How To Improve Weblogic Servers Startup Time
WebLogic 12c Enterprise Deployment Architecture in the Amazon Cloud
Cannot connect to Node Manager. : Access to domain for user 'weblogic' denied
Passing arguments to WLST scripts
Setting up SOA 11g XA Transaction Timeouts - General Rules and Recommendations
Weblogic Domain Startup and Status check WLST scripts
Using Work Managers to Limit the Impact of a Stuck Application in WebLogic Server



1. Observing the Stuck Threads

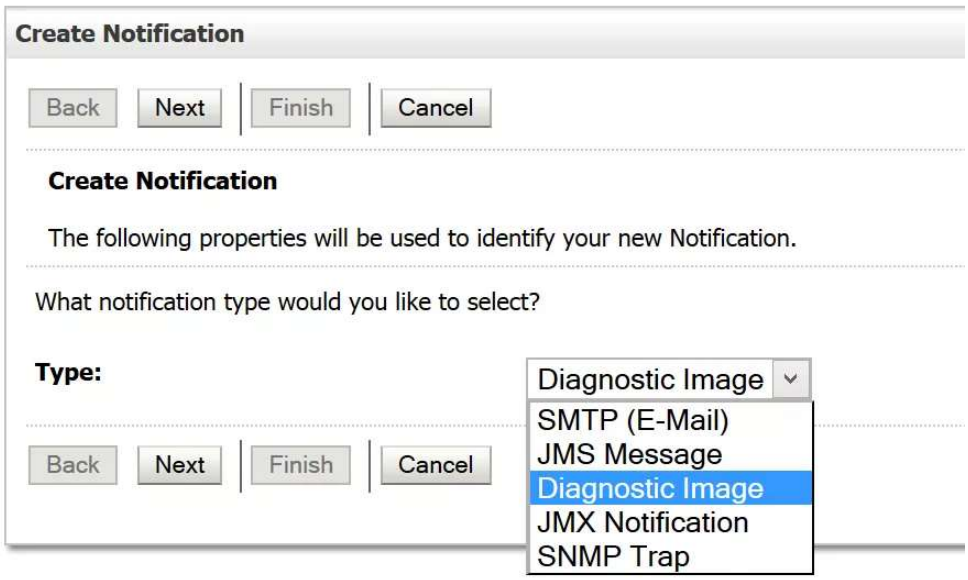
When a WebLogic server instance's thread pool has stuck threads, the instance will be marked as running with status "Warning" in the Administration console. Of course, it is more convenient to have a watch that triggers when there are stuck threads detected. In latest FMW installations, there is a diagnostic module created by default which contains a Server Log type watch, trigerring when the "Stuck Threads" message is detected in the servers logs. If you do not already have this watch named "StuckThreads" in your domain, then you can create a new one by these instructions and use the following Watch Rule for it:

((MSGID = 'BEA-000337') OR (MSGID = 'WL-000337')) AND (SEVERITY = 'Error')

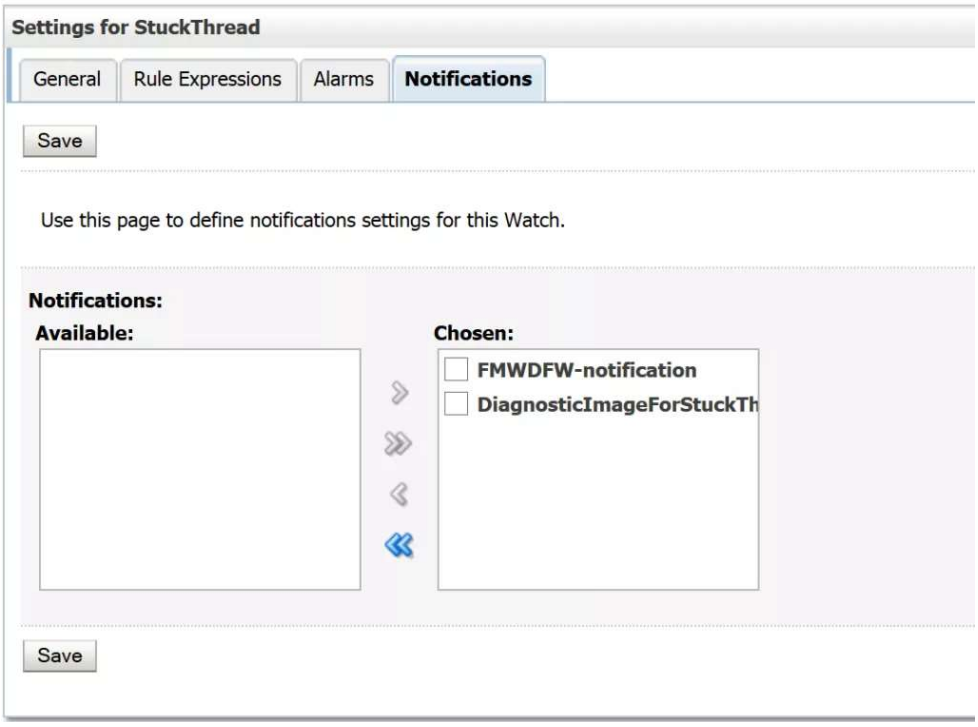


2. Generating a Thread Dump

There are many ways to generate thread dumps from Java processes, and probably the most used one is to issue "kill -3" on the PID. However, WebLogic and WLDF allow to automate the collection of thread dumps when stuck threads are detected. In order to do that, simply create a new Notification as per the same instructions as in this post. Choose the type of the Notification to be "Diagnostic Image".



and attach this Notification to the previously created Watch.



Now everytime there are Stuck Threads detected, there are also automatic Diagnostic Images generated in the configured directory. This includes information on the state of the server at a certain time, and a JVM Thread Dump that can be used in further analysis.

3. Thread Dump Analysis.

The stack trace includes calls of Java methods executed in that thread, ordered from the bottom up. Perhaps with the involvement of the developers, a WebLogic admin should be able to determine if a stuck thread is caused by an issue in the application, or by simply waiting for a slow external system. If the delay is expected, then the admin can either create a new Work Manager for this application and ignore the Stuck Threads altogether or configure a higher value for the Stuck Thread Max Time parameter (default 600 seconds) at server level.

This is a very good post regarding WebLogic thread dump analysis, showing what is normal and expected, and what you want to look out for:

http://allthingsmdw.blogspot.de/2012/02/analyzing-thread-dumps-in-middleware_08.html

4. Recovering from a stuck server

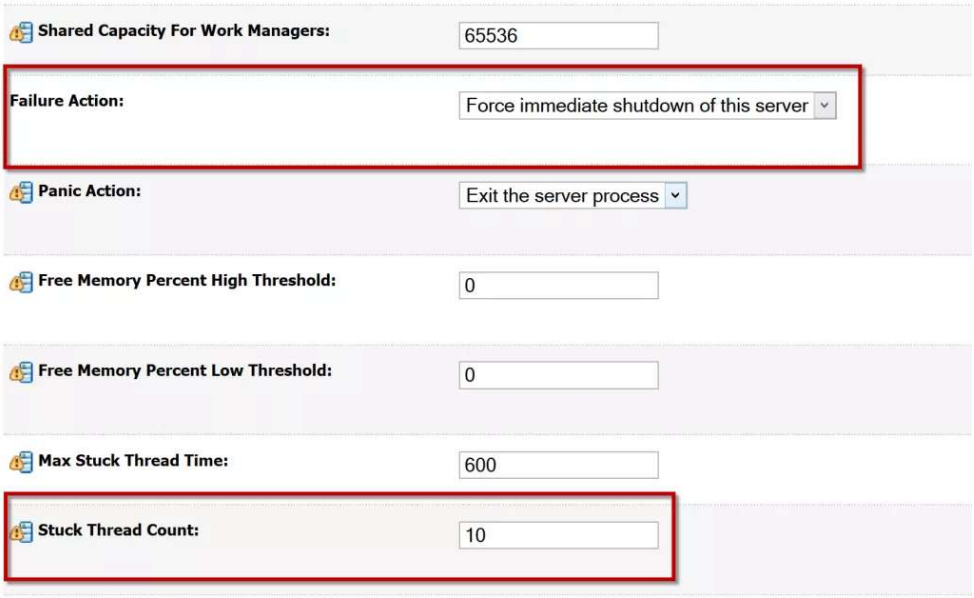
Many times, the number of stuck threads increases until the point when the WebLogic server itself comes to a halt, and all applications are simply hanging. Since the thread dump analysis together with a root cause analysis and a solution implementation can take a long time, it is a good idea to set up an automatic restart of the server once a critical number of stuck threads is reached, which will ensure an automatic recovery of the services. This is especially recommended if that application is deployed in a clustered environment, where other instances can take over the load during the affected instance's restart.

This mechanism requires the Node Manager to be configured, and can be enabled by setting up the following parameters in the "Overload" section of the server configuration:

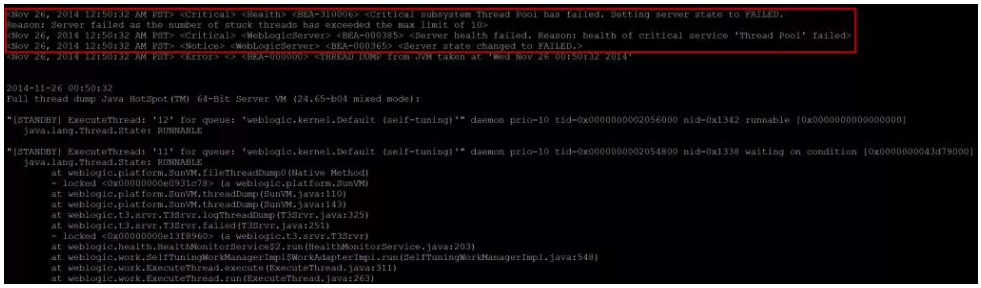
Stuck Thread Count: the number of stuck threads that cause the server to hang. This can be determined by closely monitoring the server during load, as some stuck threads might be able

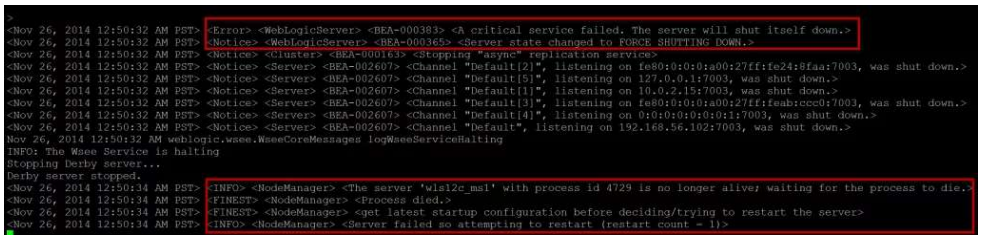
to recover and could be expected during normal operations, so you should not restart the server too soon. Once this specific number of Stuck Threads is reached, the server is marked as Failed.

Failure Action: this is the action that the server should take once it is marked as "Failed".



Once this is set up, the server will be automatically restarted by the NodeManager when the number of stuck threads reaches 10. Here are the entries in the log file:





Share this:

Share

Stuck Threads ThreadPool weblogic

← Passing arguments to WLST scripts

WebCenter Spaces and Coherence: Configuration, Best Practice, Performance Gain (Part 3/3) →

4 thoughts on “Dealing with Stuck Threads in WebLogic”

1. 

   Mamta
   August 31, 2015 at 7:02 am

   Facing thread issue in weblogic 11g, Please help .

   <[STUCK] ExecuteThread: '0' for queue: 'weblogic.kernel.Default (self-tuning)' has been busy for "1,269" seconds working on the request "weblogic.work.SelfTuningWorkManagerImpl$WorkAdapterImpl@134f5b5d", which is more than the configured time (StuckThreadMaxTime) of "1,200" seconds. Stack trace:
   java.net.SocketInputStream.socketRead0(Native Method)
   java.net.SocketInputStream.read(SocketInputStream.java:129)
   java.io.BufferedInputStream.fill(BufferedInputStream.java:218)
   java.io.BufferedInputStream.read1(BufferedInputStream.java:258)
   java.io.BufferedInputStream.read(BufferedInputStream.java:317)
   weblogic.net.http.MessageHeader.isHTTP(MessageHeader.java:224)
   weblogic.net.http.MessageHeader.parseHeader(MessageHeader.java:148)
   weblogic.net.http.HttpClient.parseHTTP(HttpClient.java:468)
   weblogic.net.http.HttpURLConnection.getInputStream(HttpURLConnection.java:377)
   weblogic.net.http.SOAPHttpURLConnection.getInputStream(SOAPHttpURLConnection.java:37)
   com.nucleus.finnone.communication.sms.utils.SMSSender.httpPostSMSURL(SMSSender.java:244)
   com.nucleus.finnone.communication.sms.utils.SMSSender.smsUrlSender(SMSSender.java:149)
   com.nucleus.finnone.communication.sms.utils.SendSMS.executeGdoForGeneration(SendSMS.java:686)
   com.nucleus.finnone.communication.sms.utils.SendSMS.smsGenerationPostStorage(SendSMS.java:355)
   com.nucleus.finnone.communication.sms.utils.SendSMS.dataStorageForGeneration(SendSMS.java:261)
   com.nucleus.finnone.communication.sms.ejb.mdbBean.SMSMDBBean.sendCommunication(Unknown Source)
   com.nucleus.finnone.communication.sms.ejb.mdbBean.SMSMDBBean.onMessage(Unknown Source)
   weblogic.ejb.container.internal.MDListener.execute(MDListener.java:574)
   weblogic.ejb.container.internal.MDListener.transactionalOnMessage(MDListener.java:477)
   weblogic.ejb.container.internal.MDListener.onMessage(MDListener.java:379)
   weblogic.jms.client.JMSSession.onMessage(JMSSession.java:4659)
   weblogic.jms.client.JMSSession.execute(JMSSession.java:4345)
   weblogic.jms.client.JMSSession.executeMessage(JMSSession.java:3821)
   weblogic.jms.client.JMSSession.access$000(JMSSession.java:115)
   weblogic.jms.client.JMSSession$UseForRunnable.run(JMSSession.java:5170)
   weblogic.work.SelfTuningWorkManagerImpl$WorkAdapterImpl.run(SelfTuningWorkManagerImpl.java:528)
   weblogic.work.ExecuteThread.execute(ExecuteThread.java:209)
   weblogic.work.ExecuteThread.run(ExecuteThread.java:178)

   Reply ↓

A. 

Radu Dobrinescu Post author
September 1, 2015 at 11:47 am

Hi Mamta,

I am not familiar with the FinnOne software, but it seems that WebLogic is waiting for some sort of response from the SMS service provider. You should check the connection between WebLogic and the SMS service provider, or if there are any unusual delays on that side. You could also verify your timeout implementation to make sure the thread pool doesn't become depleted.

Regards,
Radu

Reply ↓

2. Pingback: Weblogic State and HealthState Monitoring with Email Notification | Oracle Identity & Access Management

3. 

Anonymous
February 6, 2017 at 7:45 am

Hi ,
In the production server i got the health check issue. When server admin team check then they found some stuck thread message. So any one please let me know how can we find which class is responsible for stuck thread.

INFO Feb-02 09:02:50,010 Request[14625] [Action: get-giftcard-details, Class: GiftCardDetailsAction, Path: /fr/get-giftcard-details.do, SessionId: MWj9YT8LYvVlvp3XVhLVfDG1gQpYdXVknXcyvljWtnhX46GJyWyk!315867547!1486044168553] [[ACTIVE] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] (struts2.dispatcher.DispatcherExt)
INFO Feb-02 09:02:50,015 position:BEGIN; systemName:GetGiftCardDetails; methodName:GiftCardDetailsAction; 1486019471738:1486019471738 [[ACTIVE] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] (vmc.phone.GiftCardDetailsAction)
AUDIT Feb-02 09:02:50,015 position:BEGIN; systemName:PsWs; methodName:getGiftCardDetails; transId:1486019471738 [[ACTIVE] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'] (psws.bridge.PeoplesoftClientPsWsVMBridge)
INFO Feb-02 09:02:50,019 Before borrow: [active=1] [[ACTIVE] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'](vmc.wsutils.PoolingInvocationHandler)
INFO Feb-02 09:02:50,067 Before borrow: [active=0] [[STUCK] ExecuteThread: '4' for queue: 'weblogic.kernel.Default (self-tuning)'](vmc.wsutils.PoolingInvocationHandler)
INFO Feb-02 09:02:50,366 After release: [active=1](valid) [[ACTIVE] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'](vmc.wsutils.PoolingInvocationHandler)
INFO Feb-02 09:02:50,367 PSWSResponseStatus:SUCCESS [[ACTIVE] ExecuteThread: '11' for queue: 'weblogic.kernel.Default (self-tuning)'](psws.bridge.ResponseMapper)

Reply ↓

Leave a Reply

Your email address will not be published.

Comment

Name

Email

Website

POST COMMENT

Notify me of follow-up comments by email.

Notify me of new posts by email.