Thread Dump - Intelligence Report

File: *Threaddumps.txt*

Timestamp: 2023-10-22 22:46:56

JVM Version: 64-Bit Server VM (11.0.15.1+2-LTS-10 mixed mode)

Our **machine learning (ML)** algorithms have detected problems in your application which can cause **application unresponsiveness**. Below are the problems detected by our ML algorithms:

FATAL The application is suffering from deadlock. Here are the threads that are causing the deadlock

Thread Count Summary

To learn about different thread states through real-life example, check out this video tutorial

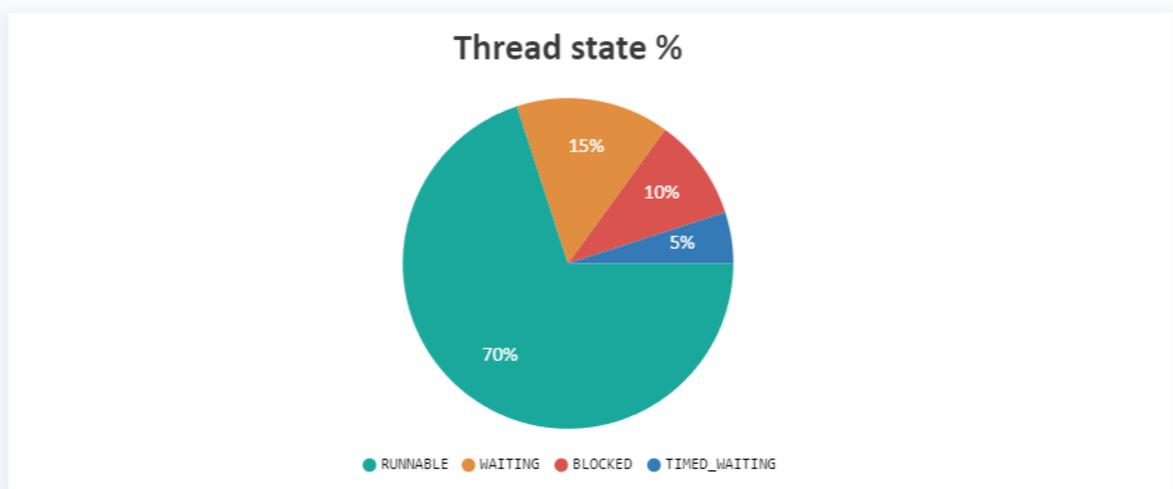
Total Threads count: 20

14
RUNNABLE
View Details

3
WAITING
View Details

2
BLOCKED
View Details

1
TIMED_WAITING
View Details

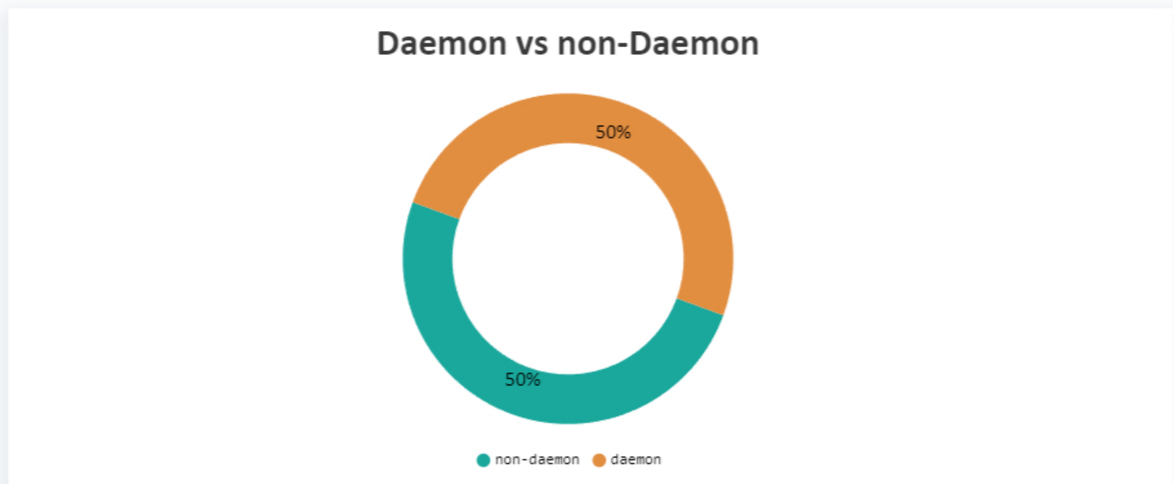


Daemon vs non-Daemon

Learn more about daemon and non-daemon (i.e. user threads)

10
NON-DAEMON
View Details

10
DAEMON
View Details



Dead Lock

Learn more about Deadlock

FATAL The application is suffering from deadlock. Below are the threads that are causing the deadlock

Thread **Thread-0** is in deadlock with thread **Thread-1**

Thread-0

PRIORITY : 5

NATIVE ID (DECIMAL) : 13160

THREAD ID : 0X000001B576D9C800

STATE : BLOCKED

NATIVE ID : 0X3368

stackTrace:
java.lang.Thread.State: BLOCKED (on object monitor)
at org.example.Threaddeadlock.lambda$main$0(Threaddeadlock.java:13)
- waiting to lock **<0x00000007110ee670>** (a java.lang.Object)
- locked **<0x00000007110ee660>** (a java.lang.Object)
at org.example.Threaddeadlock$$Lambda$14/0x0000000800066840.run(Unknown Source)
at java.lang.Thread.run(java.base@**11.0.15.1/Thread.java:834**)
Locked ownable synchronizers:
- None

Thread-1

PRIORITY : 5

NATIVE ID (DECIMAL) : 11400

THREAD ID : 0X000001B576D5D800

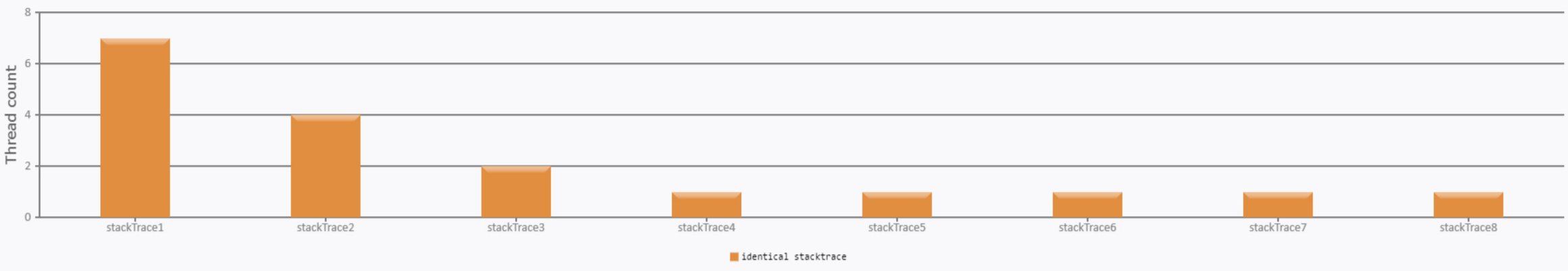
STATE : BLOCKED

NATIVE ID : 0X2C88

stackTrace:
java.lang.Thread.State: BLOCKED (on object monitor)
at org.example.Threaddeadlock.lambda$main$1(Threaddeadlock.java:24)
- waiting to lock **<0x00000007110ee660>** (a java.lang.Object)
- locked **<0x00000007110ee670>** (a java.lang.Object)
at org.example.Threaddeadlock$$Lambda$15/0x0000000800066c40.run(Unknown Source)
at java.lang.Thread.run(java.base@**11.0.15.1/Thread.java:834**)
Locked ownable synchronizers:
- None

Threads with identical stack trace

Threads with identical stack traces are grouped here. If lot of threads start to exhibit identical stack trace it might be a concern, learn RSI Pattern



Thread Count	Identical Stack trace
7 RUNNABLE threads	stacktrace See complete stacktrace.
4 RUNNABLE threads	java.lang.Thread.State: RUNNABLE Locked ownable synchronizers: - None See complete stacktrace.
2 RUNNABLE threads	java.lang.Thread.State: RUNNABLE No compile task Locked ownable synchronizers: - None See complete stacktrace.

1 TIMED_WAITING threads	java.lang.Thread.State: TIMED_WAITING (on object monitor) at java.lang.Object.wait(java.base@11.0.15.1/Native Method) - waiting on <0x00000007110064d0> (a java.lang.ref.ReferenceQueue$Lock) at java.lang.ref.ReferenceQueue.remove(java.base@11.0.15.1/ReferenceQueue.java:155) - waiting to re-lock in wait() <0x00000007110064d0> (a java.lang.ref.ReferenceQueue$Lock) ... See complete stacktrace.
1 RUNNABLE threads	java.lang.Thread.State: RUNNABLE at java.net.SocketInputStream.socketRead0(java.base@11.0.15.1/Native Method) at java.net.SocketInputStream.socketRead(java.base@11.0.15.1/SocketInputStream.java:115) at java.net.SocketInputStream.read(java.base@11.0.15.1/SocketInputStream.java:168) at java.net.SocketInputStream.read(java.base@11.0.15.1/SocketInputStream.java:140) ... See complete stacktrace.
1 BLOCKED threads	java.lang.Thread.State: BLOCKED (on object monitor) at org.example.Threaddeadlock.lambda$main$1(Threaddeadlock.java:24) - waiting to lock <0x00000007110ee660> (a java.lang.Object) - locked <0x00000007110ee670> (a java.lang.Object) at org.example.Threaddeadlock$$Lambda$15/0x0000000800066c40.run(Unknown Source) ... See complete stacktrace.
1 RUNNABLE threads	java.lang.Thread.State: RUNNABLE at java.lang.ref.Reference.waitForReferencePendingList(java.base@11.0.15.1/Native Method) at java.lang.ref.Reference.processPendingReferences(java.base@11.0.15.1/Reference.java:241) at java.lang.ref.Reference$ReferenceHandler.run(java.base@11.0.15.1/Reference.java:213) Locked ownable synchronizers: ... See complete stacktrace.
1 BLOCKED threads	java.lang.Thread.State: BLOCKED (on object monitor) at org.example.Threaddeadlock.lambda$main$0(Threaddeadlock.java:13) - waiting to lock <0x00000007110ee670> (a java.lang.Object) - locked <0x00000007110ee660> (a java.lang.Object) at org.example.Threaddeadlock$$Lambda$14/0x0000000800066840.run(Unknown Source) ... See complete stacktrace.

Last executed methods

Methods that threads were executing when thread dump was captured is reported. Learn All roads lead to Rome pattern

Thread Count	Method	Percentage
3 threads	java.lang.Object.wait(java.base@11.0.15.1/Native Method) To see stack trace click here.	15%
1 threads	org.example.Threaddeadlock.lambda$main$0(Threaddeadlock.java:13) To see stack trace click here.	5%
1 threads	java.lang.ref.Reference.waitForReferencePendingList(java.base@11.0.15.1/Native Method) To see stack trace click here.	5%
1 threads	java.net.SocketInputStream.socketRead0(java.base@11.0.15.1/Native Method) To see stack trace click here.	5%
1 threads	org.example.Threaddeadlock.lambda$main$1(Threaddeadlock.java:24) To see stack trace click here.	5%

CPU consuming threads

If application is consuming high CPU, investigate below threads. Learn Athlete pattern

Not reported!

Need help diagnosing high CPU consumption? Learn our Effective Tips

Blocking Threads - Transitive Graph

Threads that block other threads are displayed here. Blocking threads makes application unresponsive, learn Traffic Jam pattern

No transitive blocks found

GC Threads

Garbage collection threads count reported. Learn Scavengers pattern

2

GC threads

View Details

GC Thread type	Count
Concurrent GC	1
GC Worker Thread	1
Total	2

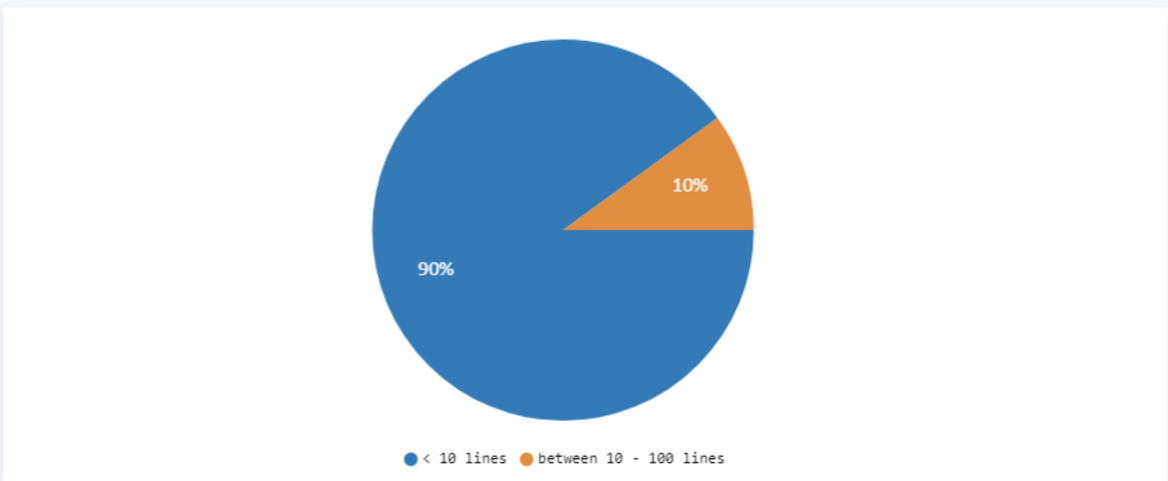
GC thread count is normal

Threads Stack Length

Lengthy stacks can cause StackOverflowError. Learn more

No Problem in Stack trace length.

Stack Length	Thread count
< 10 lines	18
between 10 - 100 lines	2



Complex DeadLocks

Learn more about Complex Deadlock

No Complex Deadlocks found

Finalizer Thread

If finalizer thread is BLOCKED or WAITING for a prolonged period, it can result in OutOfMemoryError, to learn more visit Leprechaun Trap pattern

No problem with Finalizer Thread.

Exception

Threads throwing commonly known Exceptions/Errors are reported here. Learn more

No known exceptions are reported.

Bottom up Call Stack Tree

Reverse Call stack

All threads stacktrace are combined in to one single tree. Learn it's benefits.

(9) root

(2) No compile task

- (2) java.lang.Thread.run(java.base@11.0.15.1/Thread.java:834)
  - (1) org.example.Threaddeadlock$$Lambda$14/0x0000000800066840.run(Unknown Source)
  - (1) org.example.Threaddeadlock$$Lambda$15/0x0000000800066c40.run(Unknown Source)
- (1) org.example.Threaddeadlock.main(Threaddeadlock.java:33)
  - (1) java.lang.Thread.join(java.base@11.0.15.1/Thread.java:1379)
- (1) java.lang.ref.Reference$ReferenceHandler.run(java.base@11.0.15.1/Reference.java:213)
  - (1) java.lang.ref.Reference.processPendingReferences(java.base@11.0.15.1/Reference.java:241)
- (1) java.lang.ref.Finalizer$FinalizerThread.run(java.base@11.0.15.1/Finalizer.java:170)
  - (1) java.lang.ref.ReferenceQueue.remove(java.base@11.0.15.1/ReferenceQueue.java:176)
- (1) jdk.internal.misc.InnocuousThread.run(java.base@11.0.15.1/InnocuousThread.java:134)
  - (1) java.lang.Thread.run(java.base@11.0.15.1/Thread.java:834)
- (1) com.intellij.rt.execution.application.AppMainV2$1.run(AppMainV2.java:53)
  - (1) java.io.BufferedReader.readLine(java.base@11.0.15.1/BufferedReader.java:392)

My Patterns(Beta)

Manage my Patterns

Create, apply, and visualize custom domain patterns for precise analysis. Learn More

No Pattern Found

Divide & Conquer Panel

View More

Search Panel

View More