Thread Dump - Intelligence Report

File: *dump.txt*

Timestamp: 2024-04-06 12:53:36

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

WARNING 198 threads are WAITING on ***park()*** method in ***jdk.internal.misc.Unsafe*** file and they all have same stack trace. If multiple threads exhibit same stack trace, you might want to examine **their stack trace**.

Thread Count Summary

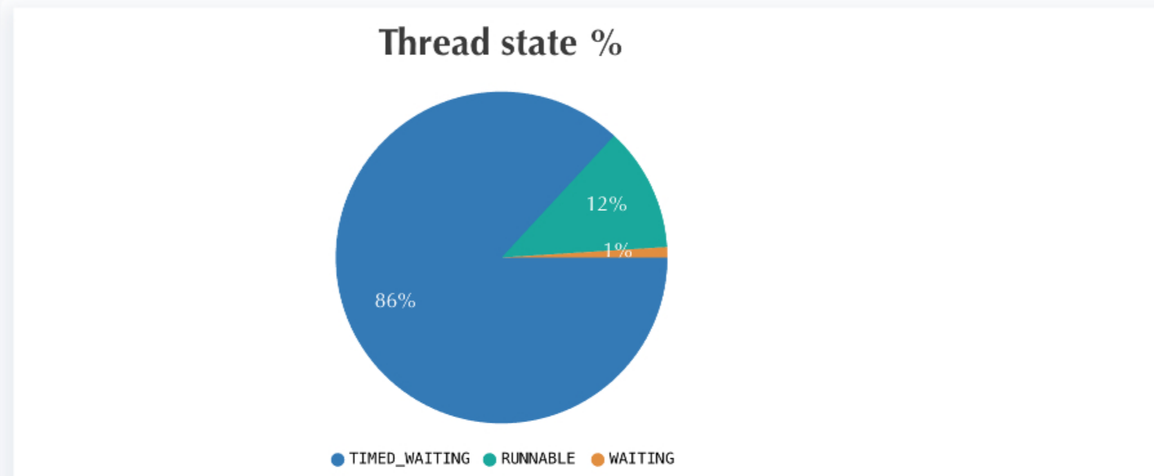
To learn about different thread states through real-life example, check out this video tutorial

Total Threads count: 242

209
TIMED_WAITING
View Details

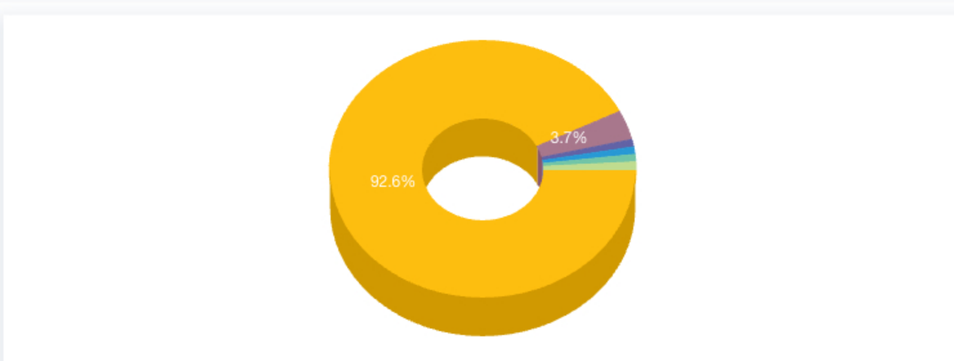
30
RUNNABLE
View Details

3
WAITING
View Details



Thread Pools

Threads with similar names are grouped in this section



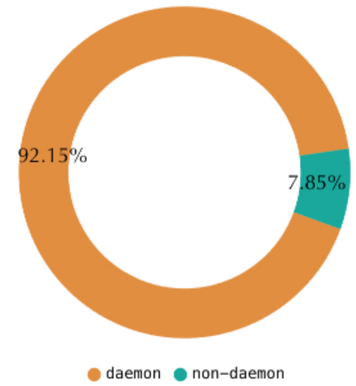
•	Thread Pool	Count	States
•	http-nio-8080-exec	**200** threads	AITING:198 RUNNABLE:2
•	GC Thread	**8** threads	RUNNABLE:8
•	G1 Conc	**2** threads	RUNNABLE:2
•	Catalina-utility	**2** threads	1 TIMED_W
•	http-nio-8080	**2** threads	RUNNABLE:2
•	HikariPool	**2** threads	TIMED_WAITING:2

Daemon vs non-Daemon

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

223
DAEMON
View Details

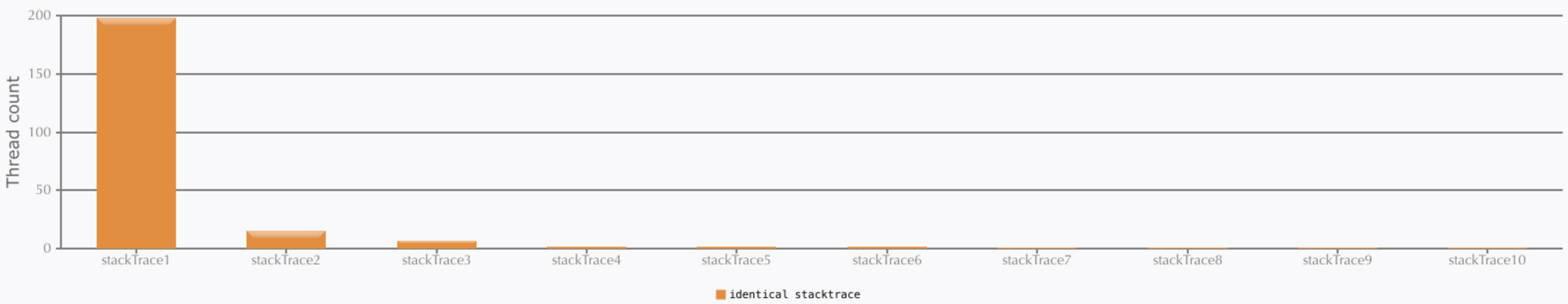
19
NON-DAEMON
View Details



Threads with identical stack trace

Become Performance Expert! Training from FastThread Architect!

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
198 TIMED_WAITING threads	java.lang.Thread.State: TIMED_WAITING (parking) at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fd2208d0> (a java.util.concurrent.SynchronousQueue$Transferer) at java.util.concurrent.locks.LockSupport.parkNanos(java.base@21.0.2/LockSupport.java:410) at java.util.concurrent.LinkedTransferQueue$DualNode.await(java.base@21.0.2/LinkedTransferQueue.java:452) ... See complete stacktrace. WARNING 198 threads are WAITING on ***park()*** method in ***jdk.internal.misc.Unsafe*** file and they all have same stack trace. If multiple threads exhibit same stack trace, you might want to examine their stack trace. (Note: If your application is unresponsive or poorly respon ding, it might be caused because these threads).
15 RUNNABLE threads	stacktrace See complete stacktrace.
7 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.
2 TIMED_WAITING threads	java.lang.Thread.State: TIMED_WAITING (parking) at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fc801110> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject) at java.util.concurrent.locks.LockSupport.parkNanos(java.base@21.0.2/LockSupport.java:269) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(java.base@21.0.2/AbstractQueuedSynchronizer.java:1847) ... See complete stacktrace.
2 TIMED_WAITING threads	java.lang.Thread.State: TIMED_WAITING (parking) at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fd2e3280> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject) at java.util.concurrent.locks.LockSupport.parkNanos(java.base@21.0.2/LockSupport.java:269) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(java.base@21.0.2/AbstractQueuedSynchronizer.java:1758) ... See complete stacktrace.
1 TIMED_WAITING threads	java.lang.Thread.State: TIMED_WAITING (parking) at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fd6064b8> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject) at java.util.concurrent.locks.LockSupport.parkNanos(java.base@21.0.2/LockSupport.java:269) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(java.base@21.0.2/AbstractQueuedSynchronizer.java:1758) ... See complete stacktrace.
1 WAITING	java.lang.Thread.State: WAITING (parking)

threads	at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fd6064b8> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject) at java.util.concurrent.locks.LockSupport.park(java.base@21.0.2/LockSupport.java:371) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionNode.block(java.base@21.0.2/AbstractQueuedSynchronizer.java:519) ... See complete stacktrace.
1 TIMED_WAITING threads	java.lang.Thread.State: TIMED_WAITING (on object monitor) at java.lang.Object.wait0(java.base@21.0.2/Native Method) - waiting on <0x00000007fd9593e8> (a jdk.jfr.internal.JVM$ChunkRotationMonitor) at java.lang.Object.wait(java.base@21.0.2/Object.java:366) at jdk.jfr.internal.PlatformRecorder.takeNap(jdk.jfr@21.0.2/PlatformRecorder.java:559) ... See complete stacktrace.
1 RUNNABLE threads	java.lang.Thread.State: RUNNABLE at sun.nio.ch.KQueue.poll(java.base@21.0.2/Native Method) at sun.nio.ch.KQueueSelectorImpl.doSelect(java.base@21.0.2/KQueueSelectorImpl.java:125) at sun.nio.ch.SelectorImpl.lockAndDoSelect(java.base@21.0.2/SelectorImpl.java:130) - locked <0x00000007fd963c20> (a sun.nio.ch.Util$2) ... 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
207 threads	jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) To see stack trace click here.	86%
3 threads	java.lang.Object.wait0(java.base@21.0.2/Native Method) To see stack trace click here.	1%
2 threads	sun.nio.ch.Net.poll(java.base@21.0.2/Native Method) To see stack trace click here.	1%
2 threads	sun.nio.ch.Net.accept(java.base@21.0.2/Native Method) To see stack trace click here.	1%
1 threads	java.lang.Thread.sleep0(java.base@21.0.2/Native Method) To see stack trace click here.	0%

Show all methods >>

CPU consuming threads

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

Thread	CPU consuming thread's stacktrace
http-nio-8080-exec-6 nativeId: 219203	java.lang.Thread.State: RUNNABLE at jdk.proxy2.$Proxy99.hashCode(jdk.proxy2/Unknown Source) at java.util.HashMap.hash(java.base@21.0.2/HashMap.java:338) at java.util.HashMap.getNode(java.base@21.0.2/HashMap.java:576) at java.util.HashMap.get(java.base@21.0.2/HashMap.java:564) ... See complete stacktrace.

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

GC Thread type	Count
Concurrent GC	2

10

GC threads

View Details

GC Worker Thread	8
Total	10

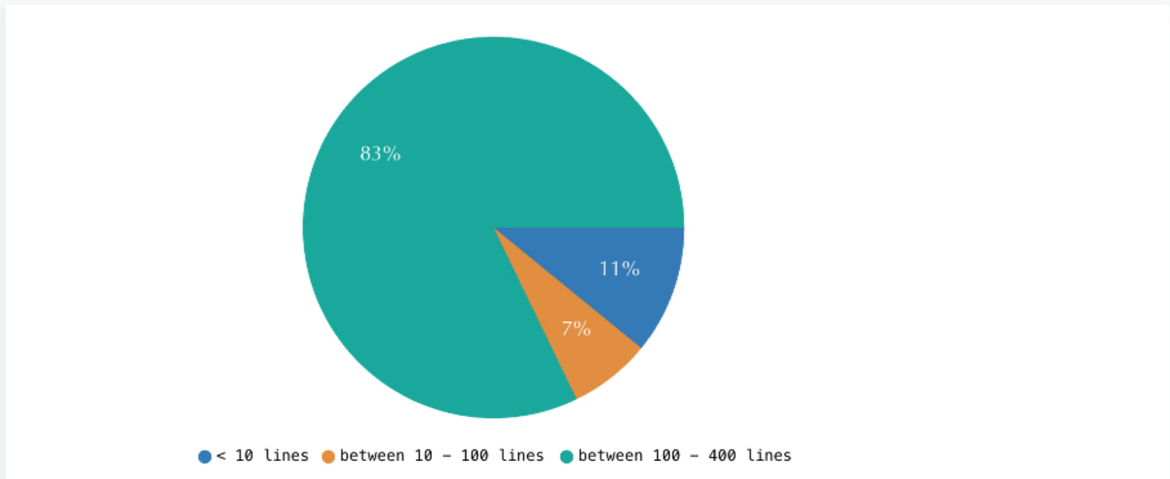
GC thread count is normal

Threads Stack Length

Lengthy stacks can cause StackOverflowError. Learn more

No Problem in Stack trace length.

Stack Length	Thread count
< 10 lines	26
between 10 - 100 lines	16
between 100 - 400 lines	200



Complex DeadLocks

Learn more about Complex Deadlock

No Complex Deadlocks found

Dead Lock

Learn more about Deadlock

No Deadlock 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.

- (220) root
  - (213) java.lang.Thread.run(java.base@21.0.2/Thread.java:1583)
    - (213) java.lang.Thread.runWith(java.base@21.0.2/Thread.java:1596)
  - (2) No compile task
  - (2) jdk.internal.misc.InnocuousThread.run(java.base@21.0.2/InnocuousThread.java:186)
    - (2) java.lang.Thread.run(java.base@21.0.2/Thread.java:1583)
  - (1) java.lang.ref.Reference$ReferenceHandler.run(java.base@21.0.2/Reference.java:208)
    - (1) java.lang.ref.Reference.processPendingReferences(java.base@21.0.2/Reference.java:246)
  - (1) java.lang.ref.Finalizer$FinalizerThread.run(java.base@21.0.2/Finalizer.java:173)
    - (1) java.lang.ref.NativeReferenceQueue.remove(java.base@21.0.2/NativeReferenceQueue.java:89)
  - (1) org.springframework.boot.web.embedded.tomcat.TomcatWebServer$1.run(TomcatWebServer.java:217)
    - (1) org.apache.catalina.core.StandardServer.await(StandardServer.java:564)

My Patterns

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