

Thread Dump - Intelligence Report

📄 File: *dump.txt*

🕒 Timestamp: 2024-04-06 15:52:56

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:

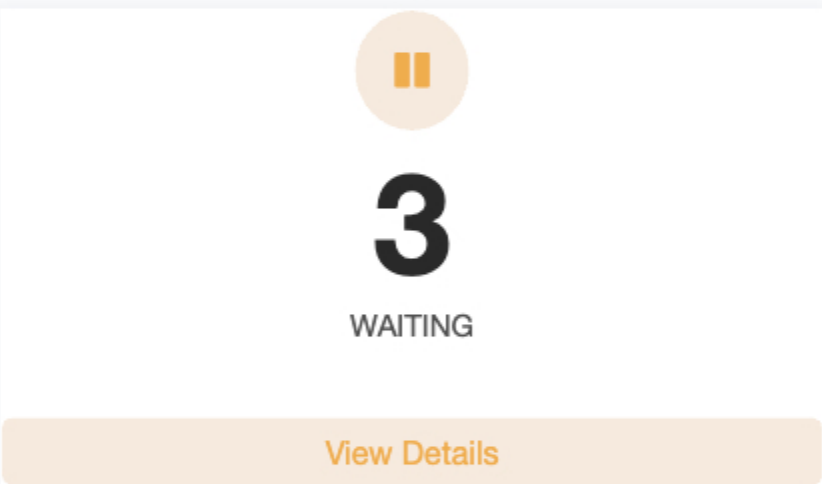
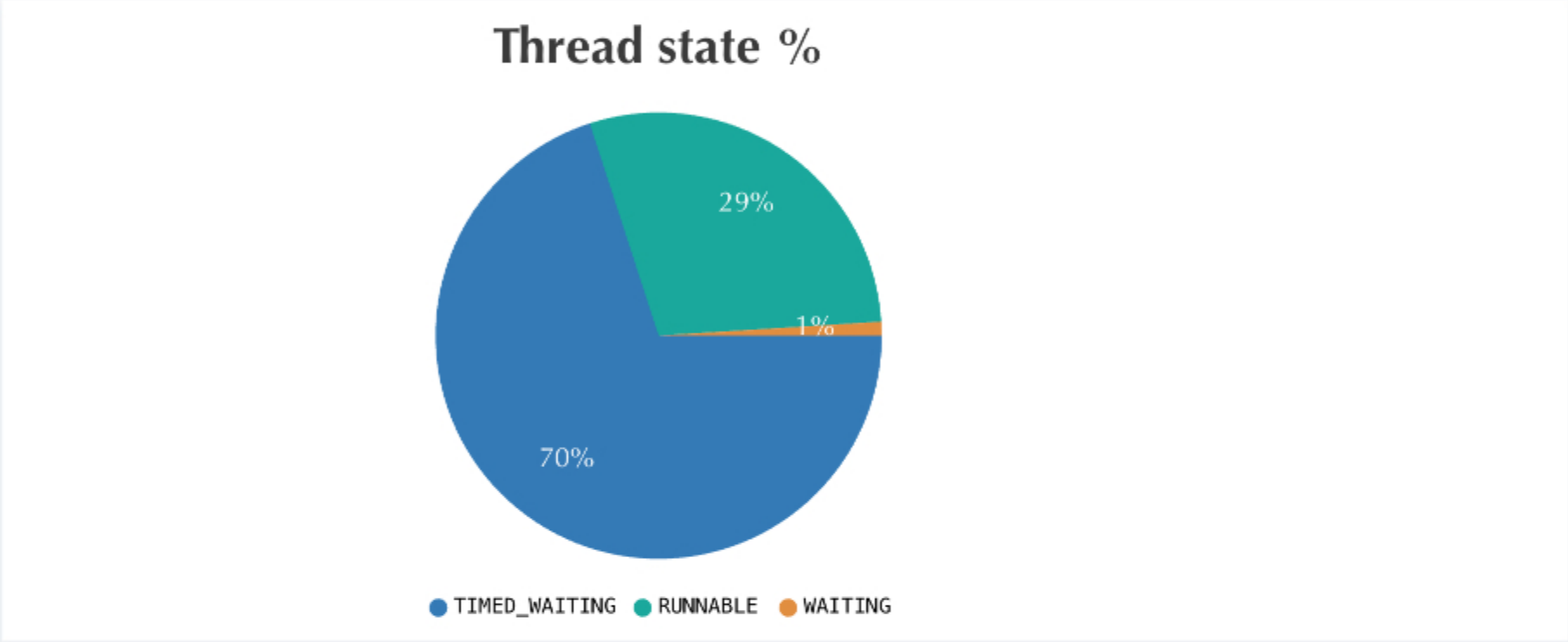
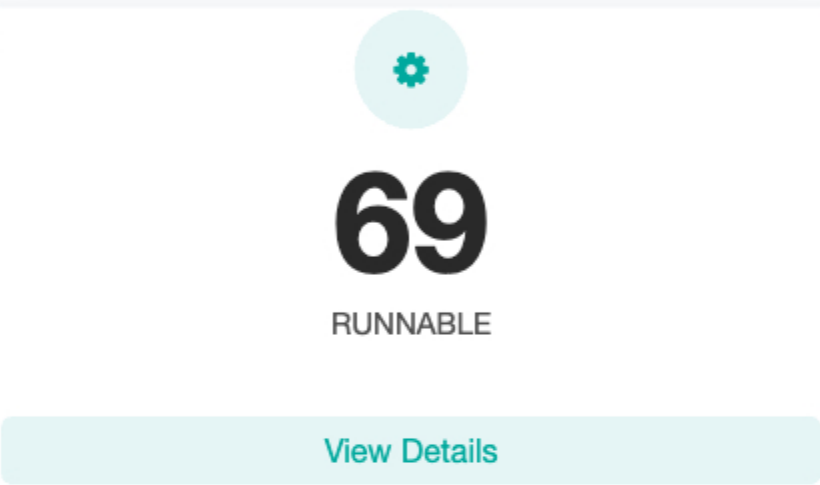
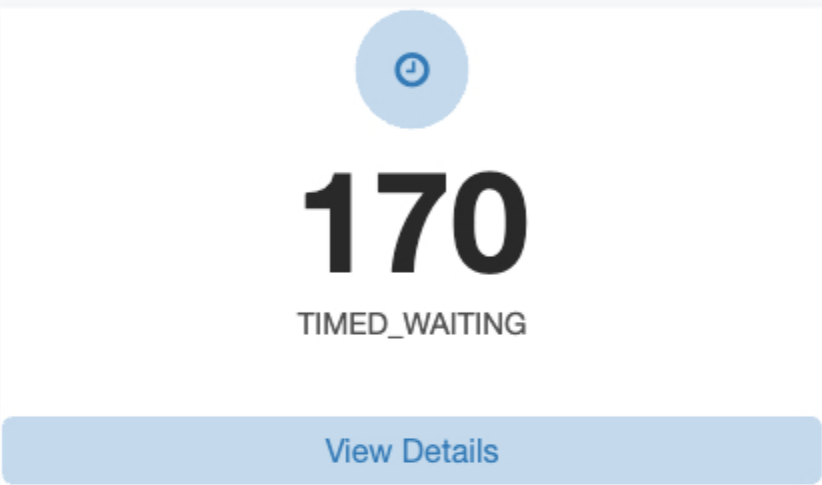


160 threads are WAITING on *park()* method in *jdk.internal.misc.Unsafe* file and they all have same stack trace. If multiple threads exhibit same stack trace, you might want to examine [their stack trace](#).

Thread Count Summary

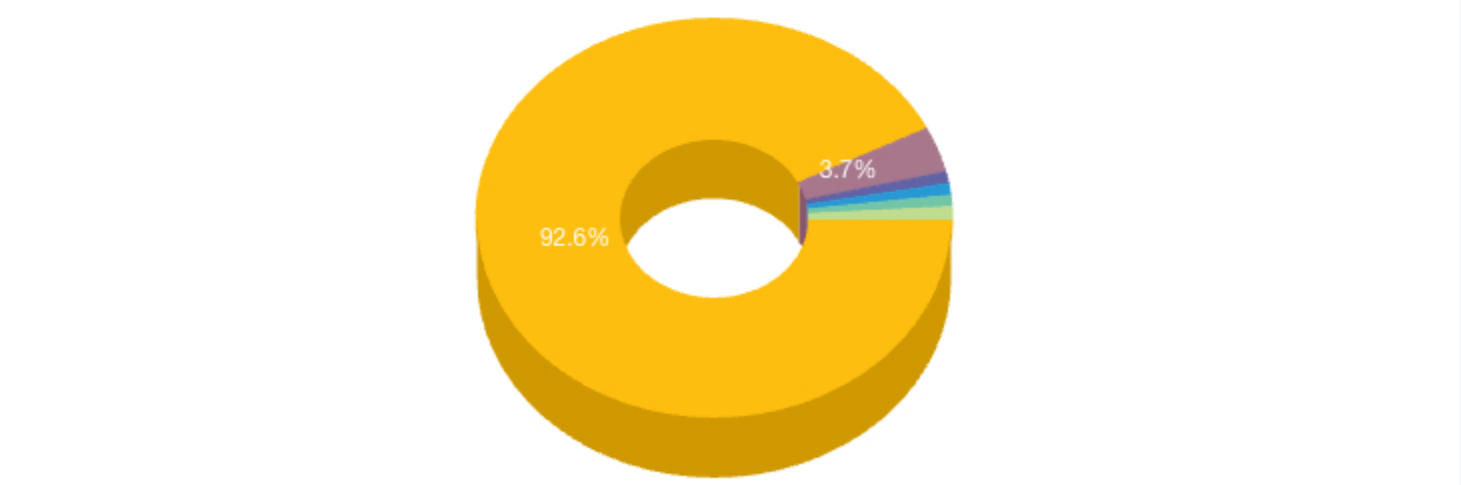
📖 To learn about different thread states through real-life example, check out this [video tutorial](#)

Total Threads count: 242



Thread Pools

Threads with similar names are grouped in this section

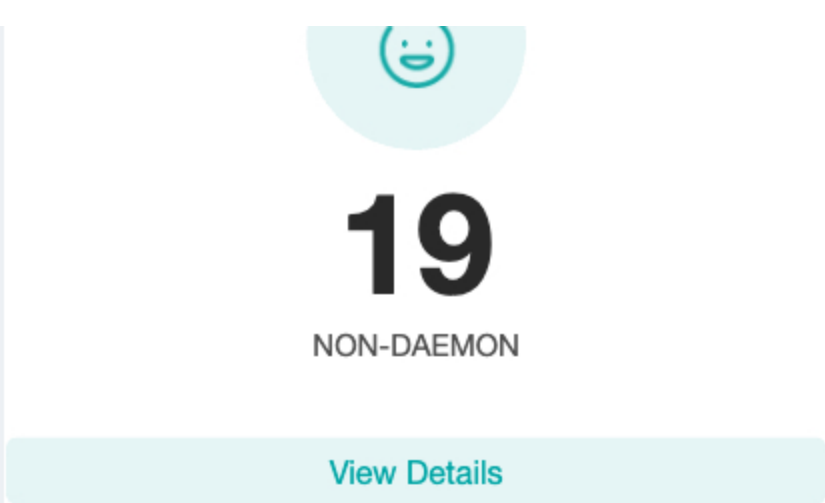


	Thread Pool	Count	States
●	http-nio-8080-exec	200 threads	TIMED_WAITING:160 RUNNABLE:40
●	GC Thread	8 threads	RUNNABLE:8
●	G1 Conc	2 threads	RUNNABLE:2
●	Catalina-utility	2 threads	1 TIMED_WAITING:1
●	http-nio-8080	2 threads	RUNNABLE:2
●	HikariPool	2 threads	TIMED_WAITING:2

Daemon vs non-Daemon

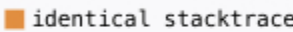
Learn more about [daemon and non-daemon \(i.e. user threads\)](#)






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
160 TIMED_WAITING threads	<div>java.lang.Thread.State: TIMED_WAITING (parking) at java.lang.Thread.park0(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fceb0458> (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.</div> <div><div> WARNING</div><div>160 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 responding, it might be caused because these threads).</div></div>
31 RUNNABLE threads	<div>java.lang.Thread.State: RUNNABLE at sun.nio.ch.Net.poll(java.base@21.0.2/Native Method) at sun.nio.ch.NioSocketImpl._poll(java.base@21.0.2/NioSocketImpl.java:191) at sun.nio.ch.NioSocketImpl._poll(java.base@21.0.2/NioSocketImpl.java:201) at sun.nio.ch.NioSocketImpl.implRead(java.base@21.0.2/NioSocketImpl.java:309) ... See complete stacktrace.</div>
15 RUNNABLE threads	<div>stacktrace See complete stacktrace.</div>
7 RUNNABLE threads	<div>java.lang.Thread.State: RUNNABLE at sun.nio.ch.Net.poll(java.base@21.0.2/Native Method) at sun.nio.ch.NioSocketImpl._poll(java.base@21.0.2/NioSocketImpl.java:191) at sun.nio.ch.NioSocketImpl._poll(java.base@21.0.2/NioSocketImpl.java:201) at sun.nio.ch.NioSocketImpl.implRead(java.base@21.0.2/NioSocketImpl.java:309) ... See complete stacktrace.</div>
7 RUNNABLE threads	<div>java.lang.Thread.State: RUNNABLE Locked ownable synchronizers: - None See complete stacktrace.</div>
2 RUNNABLE threads	<div>java.lang.Thread.State: RUNNABLE No compile task Locked ownable synchronizers: - None See complete stacktrace.</div>
2 TIMED_WAITING threads	<div>java.lang.Thread.State: TIMED_WAITING (parking) at java.lang.Thread.park0(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fc800928> (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.</div>
2 TIMED_WAITING threads	<div>java.lang.Thread.State: TIMED_WAITING (parking)</div>

threads	<pre>java.lang.Thread.State: TIMED_WAITING (parking) at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fd2dc088> (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.</pre>
2 RUNNABLE threads	<pre>java.lang.Thread.State: RUNNABLE at sun.nio.ch.Net.poll(java.base@21.0.2/Native Method) at sun.nio.ch.NioSocketImpl.park(java.base@21.0.2/NioSocketImpl.java:191) at sun.nio.ch.NioSocketImpl.timedRead(java.base@21.0.2/NioSocketImpl.java:280) at sun.nio.ch.NioSocketImpl.implRead(java.base@21.0.2/NioSocketImpl.java:304) ... See complete stacktrace.</pre>
1 TIMED_WAITING threads	<pre>java.lang.Thread.State: TIMED_WAITING (parking) at jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) - parking to wait for <0x00000007fd029928> (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.</pre>

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
168 threads	<pre>jdk.internal.misc.Unsafe.park(java.base@21.0.2/Native Method) To see stack trace click here.</pre>	69% <div><div></div></div>
42 threads	<pre>sun.nio.ch.Net.poll(java.base@21.0.2/Native Method) To see stack trace click here.</pre>	17% <div><div></div></div>
3 threads	<pre>java.lang.Object.wait0(java.base@21.0.2/Native Method) To see stack trace click here.</pre>	1% <div><div></div></div>
2 threads	<pre>sun.nio.ch.Net.accept(java.base@21.0.2/Native Method) To see stack trace click here.</pre>	1% <div><div></div></div>
1 threads	<pre>java.lang.Thread.sleep0(java.base@21.0.2/Native Method) To see stack trace click here.</pre>	0%

[Show all methods >>](#)

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](#)



10

GC threads

[View Details](#)

GC Thread type	Count
Concurrent GC	2
GC Worker Thread	8
Total	10

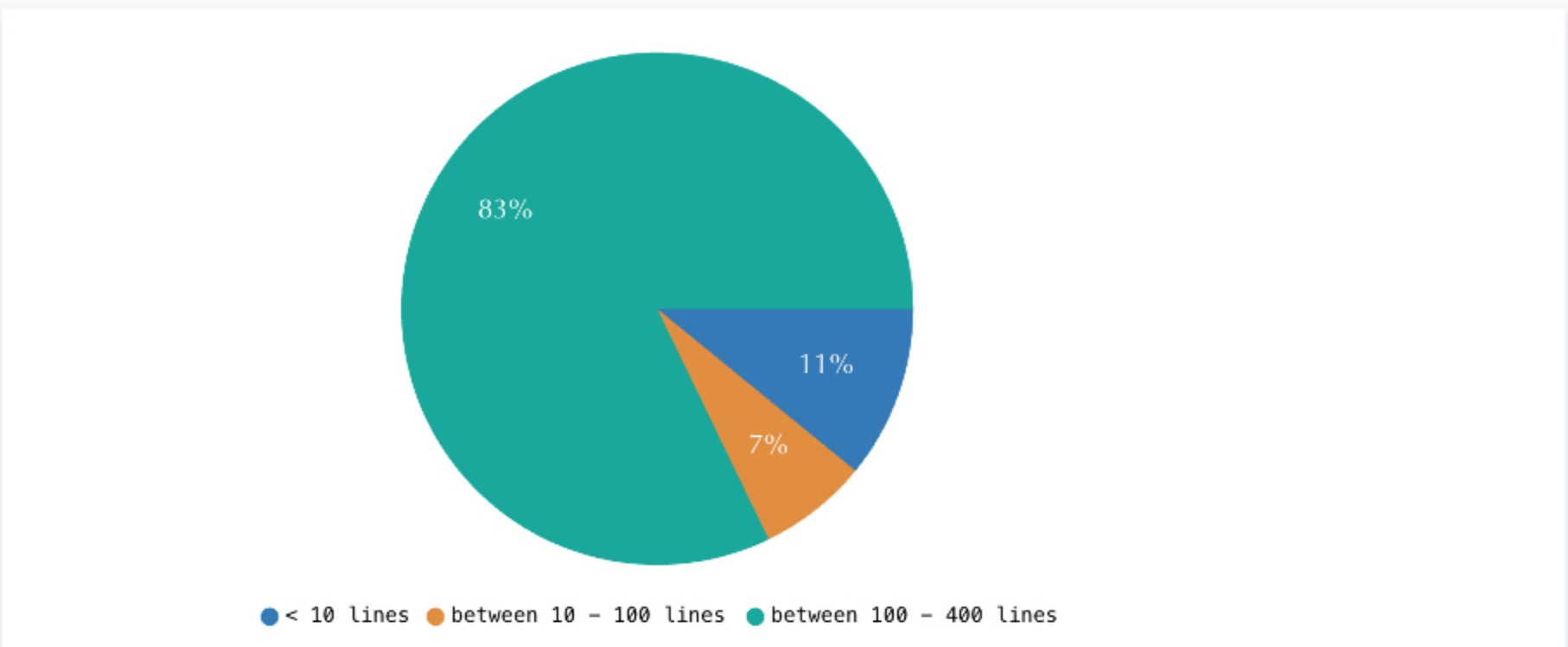
☒ GC thread count is normal

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

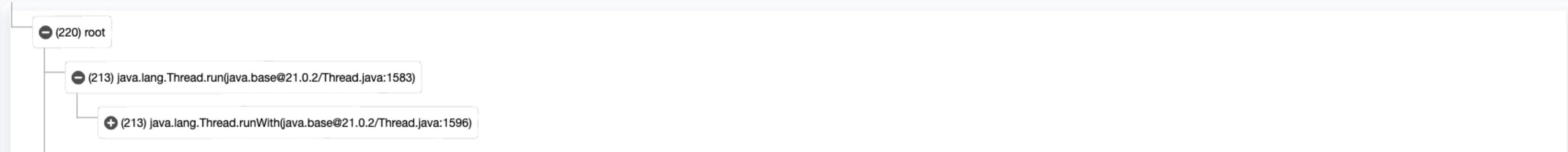
Threads throwing commonly known Exceptions/Errors are reported here. [Learn more](#)

☒ 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](#).





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](#)