adaptTo()

APACHE SLING & FRIENDS TECH MEETUP
10-12 SEPTEMBER 2018

Thread dumps demystified
Miroslav Smiljanic, Adobe

adaptTo()

- In Adobe since 2011
- In Adobe Consulting since 2012
- Joined as AEM support engineer
- Before Adobe, 6 years of experience
- www.linkedin.com/in/miroslav-smiljanic

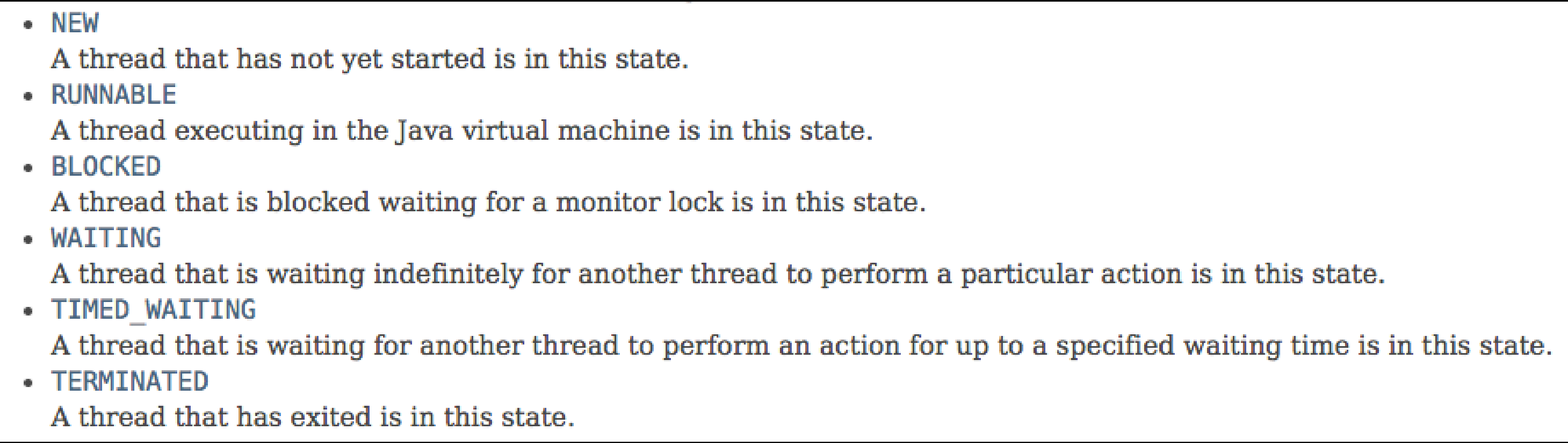
Why to bother with thread dumps

Motivation

- Solve concurrency problems
  - Thread dumps reports deadlock
  - Or with deeper analysis we can detect deadlock ourselves
- Detect processing bottlenecks
  - There is no deadlock but RUNNABLE thread blocks other threads
- Understand runtime profile of the application
- Improves your root cause analysis skills
- It is fun

Thread states

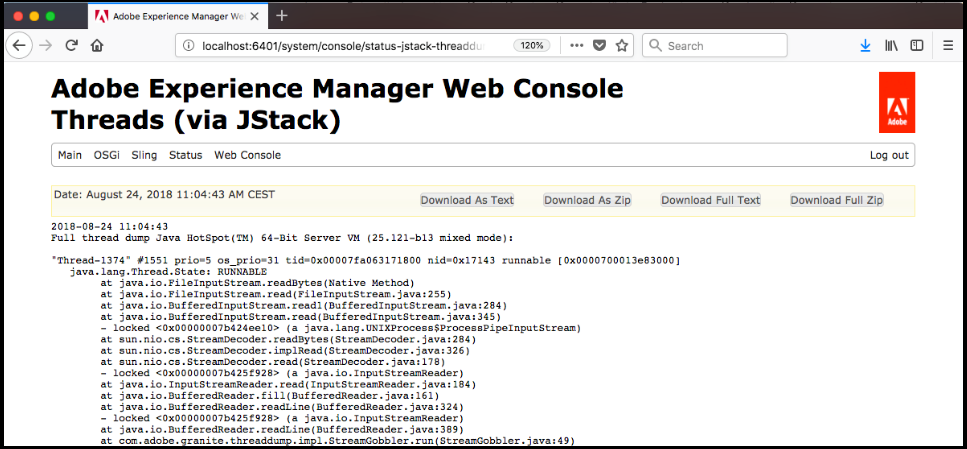
- https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html



- NEW
  A thread that has not yet started is in this state.
- RUNNABLE
  A thread executing in the Java virtual machine is in this state.
- BLOCKED
  A thread that is blocked waiting for a monitor lock is in this state.
- WAITING
  A thread that is waiting indefinitely for another thread to perform a particular action is in this state.
- TIMED_WAITING
  A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.
- TERMINATED
  A thread that has exited is in this state.

How to take thread dumps

- https://helpx.adobe.com/experience-manager/kb/TakeThreadDump.html
- Via OSGi console, /system/console/status-jstack-threaddump


Adobe Experience Manager We
localhost:6401/system/console/status-jstack-threaddu
120%
Search

Adobe Experience Manager Web Console
Threads (via JStack)

Main OSGi Sling Status Web Console Log out

Date: August 24, 2018 11:04:43 AM CEST

Download As Text | Download As Zip | Download Full Text | Download Full Zip

```
2018-08-24 11:04:43
Full thread dump Java HotSpot(TM) 64-Bit Server VM (25.121-b13 mixed mode):

"Thread-1374" #1551 prio=5 os_prio=31 tid=0x00007fa063171800 nid=0x17143 runnable [0x0000700013e83000]
   java.lang.Thread.State: RUNNABLE
        at java.io.FileInputStream.readBytes(Native Method)
        at java.io.FileInputStream.read(FileInputStream.java:255)
        at java.io.BufferedInputStream.read1(BufferedInputStream.java:284)
        at java.io.BufferedInputStream.read(BufferedInputStream.java:345)
        - locked <0x00000007b424ee10> (a java.lang.UNIXProcess$ProcessPipeInputStream)
        at sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:284)
        at sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:326)
        at sun.nio.cs.StreamDecoder.read(StreamDecoder.java:178)
        - locked <0x00000007b425f928> (a java.io.InputStreamReader)
        at java.io.InputStreamReader.read(InputStreamReader.java:184)
        at java.io.BufferedReader.fill(BufferedReader.java:161)
        at java.io.BufferedReader.readLine(BufferedReader.java:324)
        - locked <0x00000007b425f928> (a java.io.InputStreamReader)
        at java.io.BufferedReader.readLine(BufferedReader.java:389)
        at com.adobe.granite.threaddump.impl.StreamGobbler.run(StreamGobbler.java:49)
```

- https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jvisualvm.html

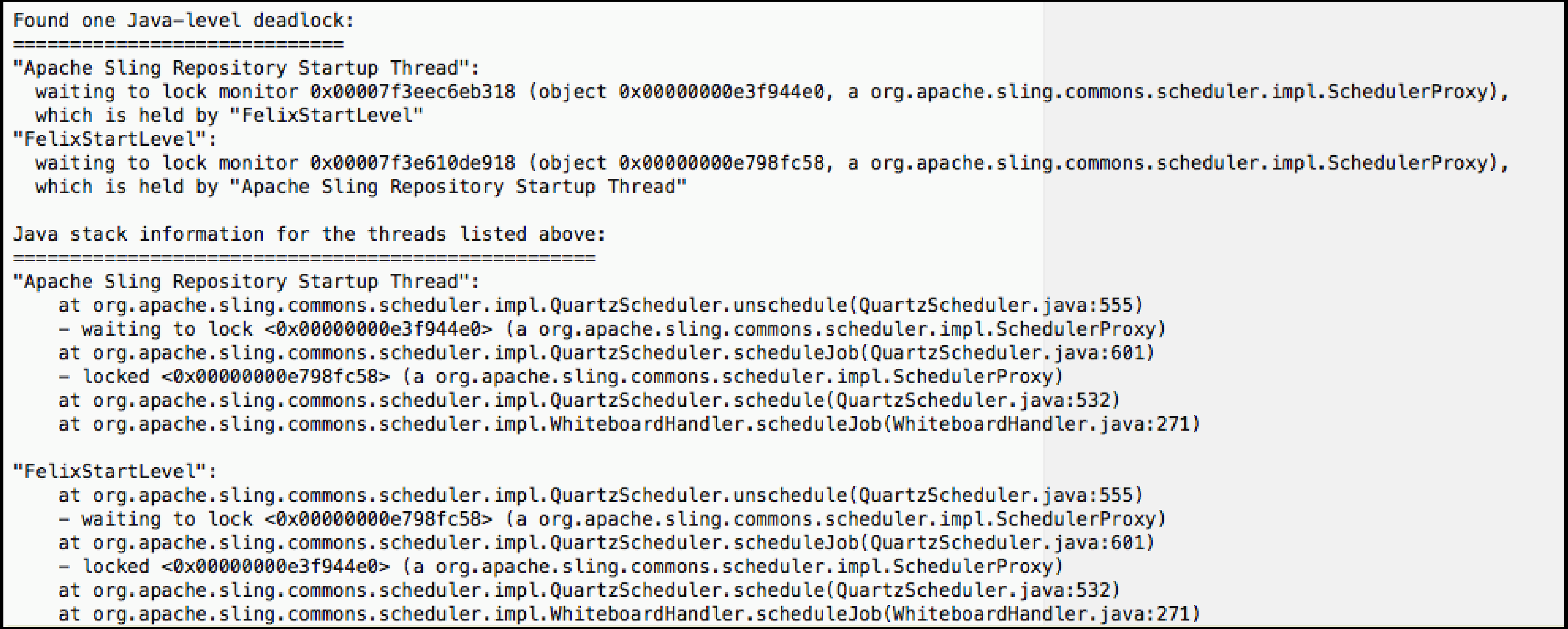
Deadlock detected in thread dump

Easy case

- SLING-7004: Deadlock at startup in Commons Scheduler
  - https://issues.apache.org/jira/browse/SLING-7004
- Thread dump detects deadlock
- Synchronization done with keyword **synchronized**

adaptTo()

Deadlock detected



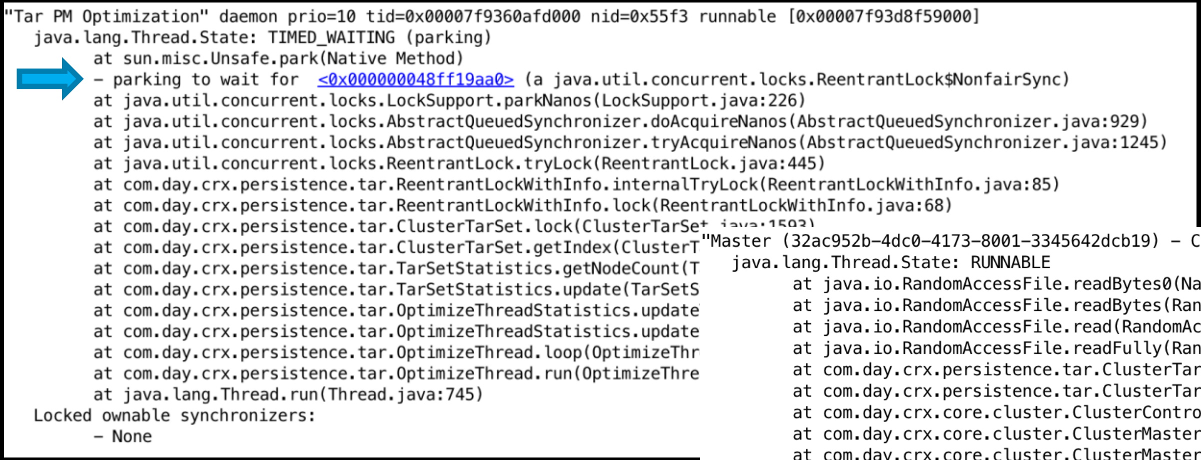
```
Found one Java-level deadlock:
=============================
"Apache Sling Repository Startup Thread":
  waiting to lock monitor 0x00007f3eec6eb318 (object 0x00000000e3f944e0, a org.apache.sling.commons.scheduler.impl.SchedulerProxy),
  which is held by "FelixStartLevel"
"FelixStartLevel":
  waiting to lock monitor 0x00007f3e610de918 (object 0x00000000e798fc58, a org.apache.sling.commons.scheduler.impl.SchedulerProxy),
  which is held by "Apache Sling Repository Startup Thread"

Java stack information for the threads listed above:
===================================================
"Apache Sling Repository Startup Thread":
    at org.apache.sling.commons.scheduler.impl.QuartzScheduler.unschedule(QuartzScheduler.java:555)
    - waiting to lock <0x00000000e3f944e0> (a org.apache.sling.commons.scheduler.impl.SchedulerProxy)
    at org.apache.sling.commons.scheduler.impl.QuartzScheduler.scheduleJob(QuartzScheduler.java:601)
    - locked <0x00000000e798fc58> (a org.apache.sling.commons.scheduler.impl.SchedulerProxy)
    at org.apache.sling.commons.scheduler.impl.QuartzScheduler.schedule(QuartzScheduler.java:532)
    at org.apache.sling.commons.scheduler.impl.WhiteboardHandler.scheduleJob(WhiteboardHandler.java:271)

"FelixStartLevel":
    at org.apache.sling.commons.scheduler.impl.QuartzScheduler.unschedule(QuartzScheduler.java:555)
    - waiting to lock <0x00000000e798fc58> (a org.apache.sling.commons.scheduler.impl.SchedulerProxy)
    at org.apache.sling.commons.scheduler.impl.QuartzScheduler.scheduleJob(QuartzScheduler.java:601)
    - locked <0x00000000e3f944e0> (a org.apache.sling.commons.scheduler.impl.SchedulerProxy)
    at org.apache.sling.commons.scheduler.impl.QuartzScheduler.schedule(QuartzScheduler.java:532)
    at org.apache.sling.commons.scheduler.impl.WhiteboardHandler.scheduleJob(WhiteboardHandler.java:271)
```

Thread synchronized with java.util.concurrent

Thread synchronized with java.util.concurrent

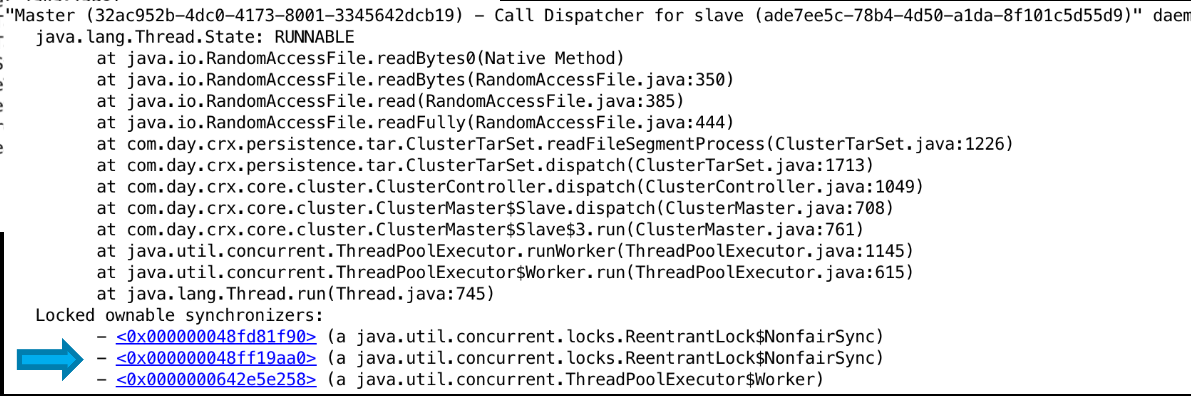
- Classes that can be used for synchronization
  - java.util.concurrent.Semaphore
  - java.util.concurrent.locks.ReentrantLock
  - java.util.concurrent.locks.ReentrantReadWriteLock.ReadLock
  - java.util.concurrent.locks.ReentrantReadWriteLock.WriteLock
- "a framework for locking and waiting for conditions that are distinct from built-in synchronization and monitors"
- If the Java VM flag **-XX:+PrintConcurrentLocks** is set then stack trace shows list of synchronizers (concurrent locks) owned by specific thread
- The same effect when using **jstack –l <pid>**

CQ5 share nothing clustering



```
"Tar PM Optimization" daemon prio=10 tid=0x00007f9360afd000 nid=0x55f3 runnable [0x00007f93d8f59000]
   java.lang.Thread.State: TIMED_WAITING (parking)
        at sun.misc.Unsafe.park(Native Method)
        - parking to wait for  <0x000000048ff19aa0> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
        at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:226)
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireNanos(AbstractQueuedSynchronizer.java:929)
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.tryAcquireNanos(AbstractQueuedSynchronizer.java:1245)
        at java.util.concurrent.locks.ReentrantLock.tryLock(ReentrantLock.java:445)
        at com.day.crx.persistence.tar.ReentrantLockWithInfo.internalTryLock(ReentrantLockWithInfo.java:85)
        at com.day.crx.persistence.tar.ReentrantLockWithInfo.lock(ReentrantLockWithInfo.java:68)
        at com.day.crx.persistence.tar.ClusterTarSet.lock(ClusterTarSe
        at com.day.crx.persistence.tar.ClusterTarSet.getIndex(ClusterT
        at com.day.crx.persistence.tar.TarSetStatistics.getNodeCount(T
        at com.day.crx.persistence.tar.TarSetStatistics.update(TarSetS
        at com.day.crx.persistence.tar.OptimizeThreadStatistics.update
        at com.day.crx.persistence.tar.OptimizeThreadStatistics.update
        at com.day.crx.persistence.tar.OptimizeThread.loop(OptimizeThr
        at com.day.crx.persistence.tar.OptimizeThread.run(OptimizeThre
        at java.lang.Thread.run(Thread.java:745)
   Locked ownable synchronizers:
        - None
```

- Blocked thread



```
"Master (32ac952b-4dc0-4173-8001-3345642dcb19) - Call Dispatcher for slave (ade7ee5c-78b4-4d50-a1da-8f101c5d55d9)" daem
   java.lang.Thread.State: RUNNABLE
        at java.io.RandomAccessFile.readBytes0(Native Method)
        at java.io.RandomAccessFile.readBytes(RandomAccessFile.java:350)
        at java.io.RandomAccessFile.read(RandomAccessFile.java:385)
        at java.io.RandomAccessFile.readFully(RandomAccessFile.java:444)
        at com.day.crx.persistence.tar.ClusterTarSet.readFileSegmentProcess(ClusterTarSet.java:1226)
        at com.day.crx.persistence.tar.ClusterTarSet.dispatch(ClusterTarSet.java:1713)
        at com.day.crx.core.cluster.ClusterController.dispatch(ClusterController.java:1049)
        at com.day.crx.core.cluster.ClusterMaster$Slave.dispatch(ClusterMaster.java:708)
        at com.day.crx.core.cluster.ClusterMaster$Slave$3.run(ClusterMaster.java:761)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:745)
   Locked ownable synchronizers:
        - <0x000000048fd81f90> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
        - <0x000000048ff19aa0> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
        - <0x0000000642e5e258> (a java.util.concurrent.ThreadPoolExecutor$Worker)
```

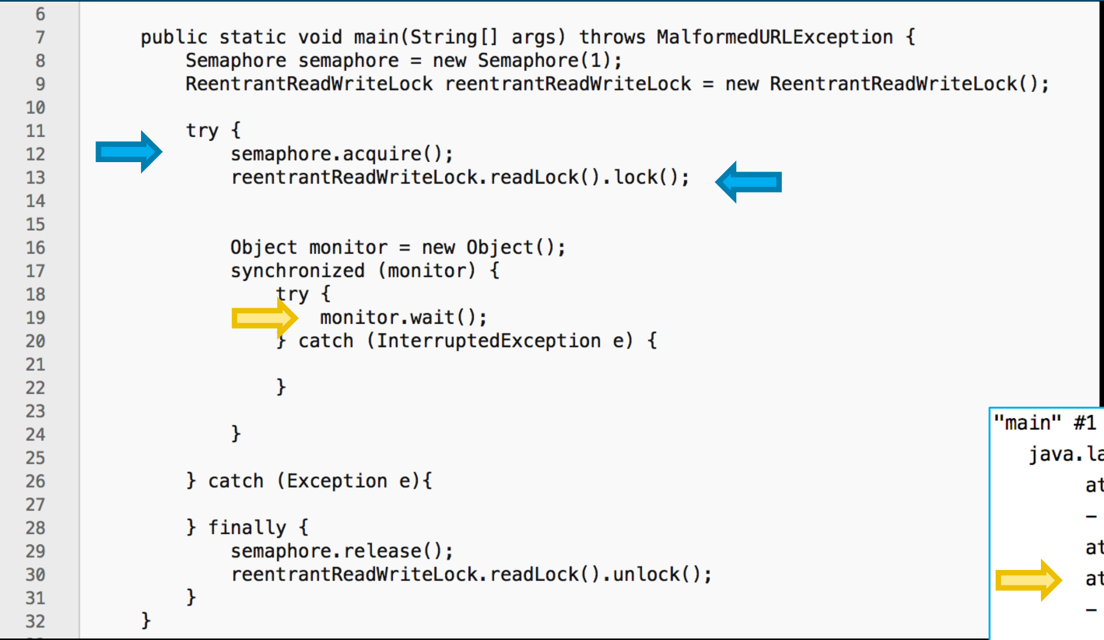
- Blocking thread

- Problem: slave node joined the cluster after tow days
  - During synchronization maser blocked repository access for other threads

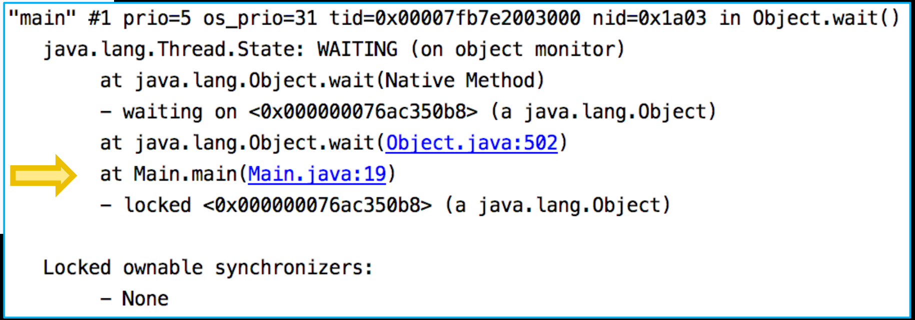
adaptTo()

Demo

But …



```
 6
 7     public static void main(String[] args) throws MalformedURLException {
 8         Semaphore semaphore = new Semaphore(1);
 9         ReentrantReadWriteLock reentrantReadWriteLock = new ReentrantReadWriteLock();
10
11         try {
12             semaphore.acquire();
13             reentrantReadWriteLock.readLock().lock();
14
15
16             Object monitor = new Object();
17             synchronized (monitor) {
18                 try {
19                     monitor.wait();
20                 } catch (InterruptedException e) {
21
22                 }
23
24             }
25
26         } catch (Exception e){
27
28         } finally {
29             semaphore.release();
30             reentrantReadWriteLock.readLock().unlock();
31         }
32     }
```

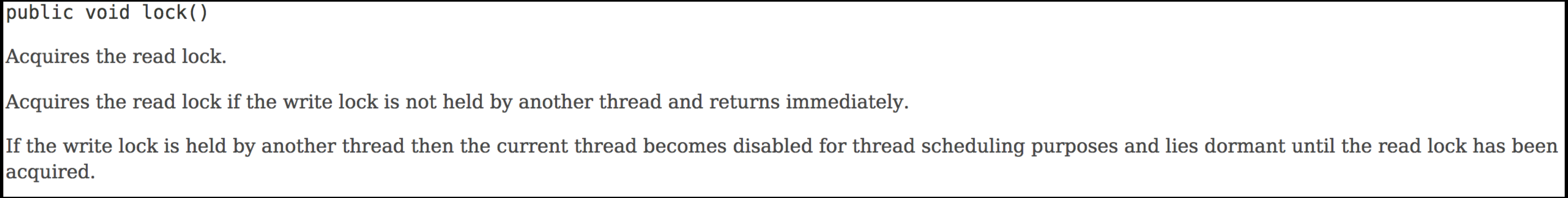


```
"main" #1 prio=5 os_prio=31 tid=0x00007fb7e2003000 nid=0x1a03 in Object.wait()
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x000000076ac350b8> (a java.lang.Object)
        at java.lang.Object.wait(Object.java:502)
        at Main.main(Main.java:19)
        - locked <0x000000076ac350b8> (a java.lang.Object)

   Locked ownable synchronizers:
        - None
```

- Locked synchronizers not detected!!
- That makes hard to analyze thread dumps

ReentrantReadWriteLock

- ReentrantReadWriteLock.ReadLock.lock()



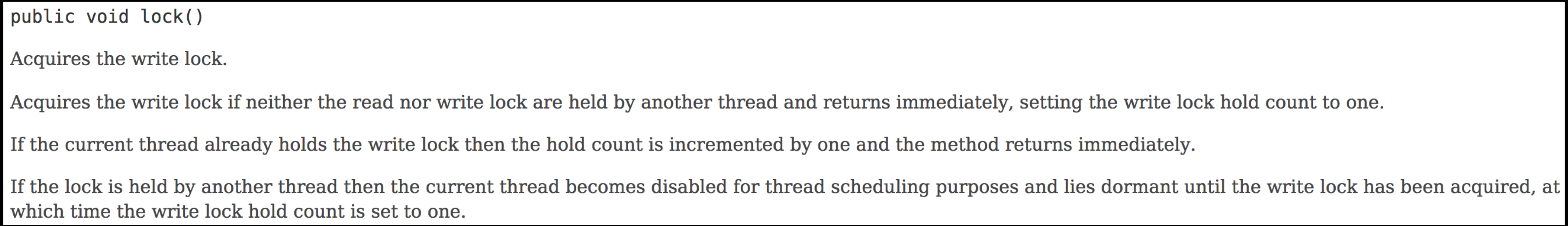
```
public void lock()
```

Acquires the read lock.

Acquires the read lock if the write lock is not held by another thread and returns immediately.

If the write lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the read lock has been acquired.

- ReentrantReadWriteLock.WriteLock.lock()



```
public void lock()
```

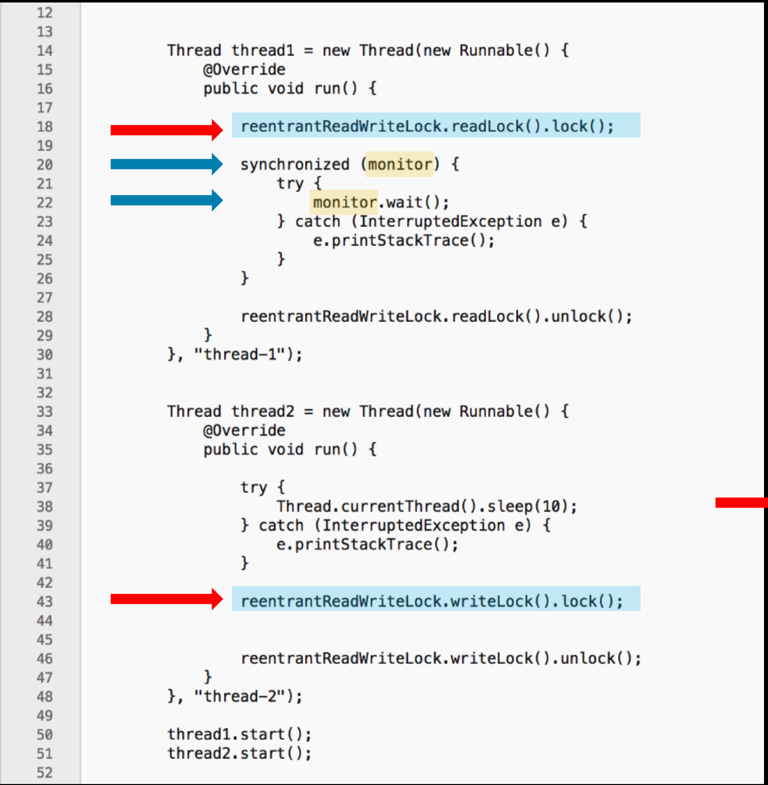
Acquires the write lock.

Acquires the write lock if neither the read nor write lock are held by another thread and returns immediately, setting the write lock hold count to one.

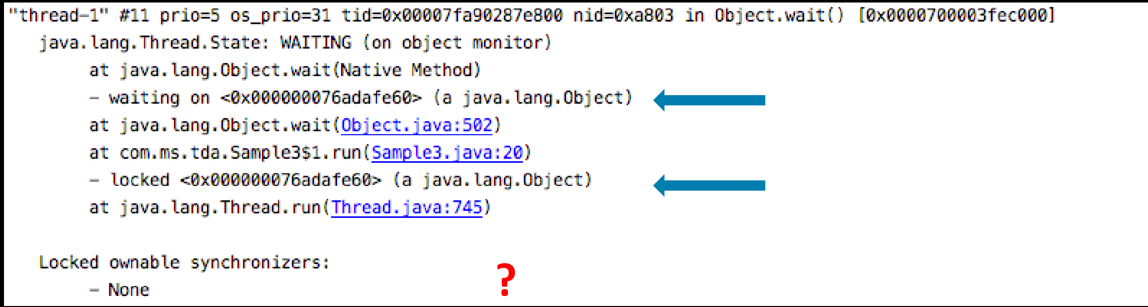
If the current thread already holds the write lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the write lock has been acquired, at which time the write lock hold count is set to one.

Two threads example

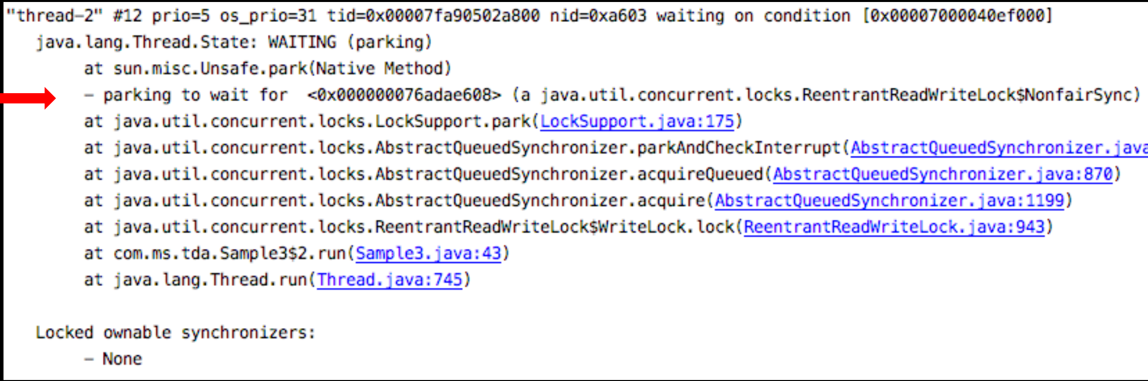


```
12
13
14      Thread thread1 = new Thread(new Runnable() {
15          @Override
16          public void run() {
17
18              reentrantReadWriteLock.readLock().lock();
19
20              synchronized (monitor) {
21                  try {
22                      monitor.wait();
23                  } catch (InterruptedException e) {
24                      e.printStackTrace();
25                  }
26              }
27
28              reentrantReadWriteLock.readLock().unlock();
29          }
30      }, "thread-1");
31
32
33      Thread thread2 = new Thread(new Runnable() {
34          @Override
35          public void run() {
36
37              try {
38                  Thread.currentThread().sleep(10);
39              } catch (InterruptedException e) {
40                  e.printStackTrace();
41              }
42
43              reentrantReadWriteLock.writeLock().lock();
44
45
46              reentrantReadWriteLock.writeLock().unlock();
47          }
48      }, "thread-2");
49
50      thread1.start();
51      thread2.start();
52
```



```
"thread-1" #11 prio=5 os_prio=31 tid=0x00007fa90287e800 nid=0xa803 in Object.wait() [0x0000700003fec000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x000000076adafe60> (a java.lang.Object)
        at java.lang.Object.wait(Object.java:502)
        at com.ms.tda.Sample3$1.run(Sample3.java:20)
        - locked <0x000000076adafe60> (a java.lang.Object)
        at java.lang.Thread.run(Thread.java:745)

   Locked ownable synchronizers:
        - None
```

?

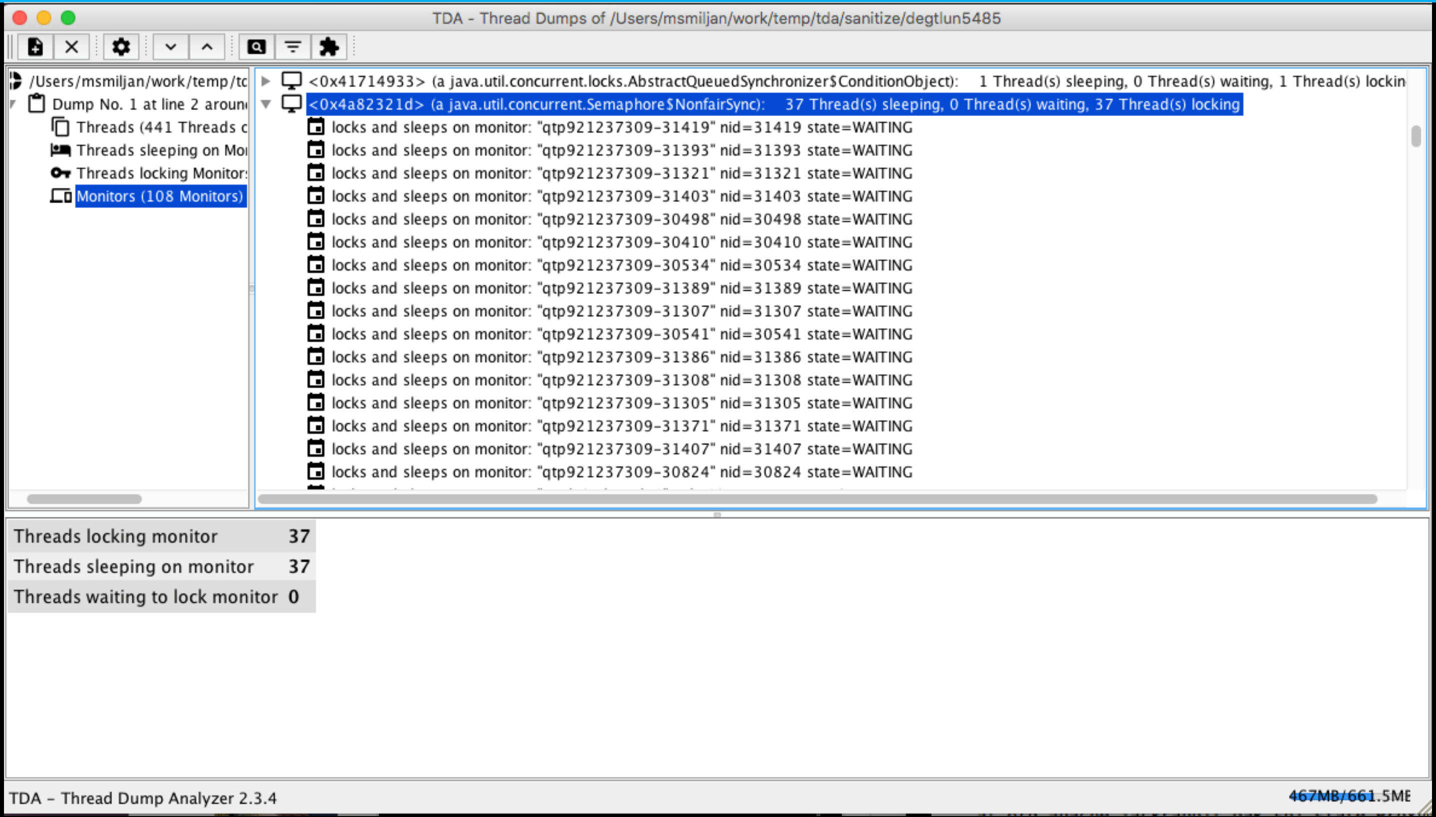


```
"thread-2" #12 prio=5 os_prio=31 tid=0x00007fa90502a800 nid=0xa603 waiting on condition [0x00007000040ef000]
   java.lang.Thread.State: WAITING (parking)
        at sun.misc.Unsafe.park(Native Method)
        - parking to wait for  <0x000000076adae608> (a java.util.concurrent.locks.ReentrantReadWriteLock$NonfairSync)
        at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(AbstractQueuedSynchronizer.java
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireQueued(AbstractQueuedSynchronizer.java:870)
        at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquire(AbstractQueuedSynchronizer.java:1199)
        at java.util.concurrent.locks.ReentrantReadWriteLock$WriteLock.lock(ReentrantReadWriteLock.java:943)
        at com.ms.tda.Sample3$2.run(Sample3.java:43)
        at java.lang.Thread.run(Thread.java:745)

   Locked ownable synchronizers:
        - None
```

Known “issue”

- Locked synchronizers not detected when using
  - java.util.concurrent.Semaphore
  - java.util.concurrent.locks.ReentrantReadWriteLock.ReadLock
- The situation is the same when using 3rd party libraries
- For example the class that was used in Jackrabbit
  - EDU.oswego.cs.dl.util.concurrent.WriterPreferenceReadWriteLock
- “Bug” has been reported
  - https://bugs.java.com/bugdatabase/view_bug.do?bug_id=6207928
  - Fixed or “works as designed”?

Blocking thread not detected



TDA - Thread Dumps of /Users/msmiljan/work/temp/tda/sanitize/degtlun5485

```
/Users/msmiljan/work/temp/tc
  Dump No. 1 at line 2 aroun
    Threads (441 Threads c
    Threads sleeping on Mo
    Threads locking Monitor:
    Monitors (108 Monitors)
```

```
▶ <0x41714933> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject):   1 Thread(s) sleeping, 0 Thread(s) waiting, 1 Thread(s) lockin
▼ <0x4a82321d> (a java.util.concurrent.Semaphore$NonfairSync):   37 Thread(s) sleeping, 0 Thread(s) waiting, 37 Thread(s) locking
    locks and sleeps on monitor: "qtp921237309-31419" nid=31419 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31393" nid=31393 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31321" nid=31321 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31403" nid=31403 state=WAITING
    locks and sleeps on monitor: "qtp921237309-30498" nid=30498 state=WAITING
    locks and sleeps on monitor: "qtp921237309-30410" nid=30410 state=WAITING
    locks and sleeps on monitor: "qtp921237309-30534" nid=30534 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31389" nid=31389 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31307" nid=31307 state=WAITING
    locks and sleeps on monitor: "qtp921237309-30541" nid=30541 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31386" nid=31386 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31308" nid=31308 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31305" nid=31305 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31371" nid=31371 state=WAITING
    locks and sleeps on monitor: "qtp921237309-31407" nid=31407 state=WAITING
    locks and sleeps on monitor: "qtp921237309-30824" nid=30824 state=WAITING
```

Threads locking monitor	37
Threads sleeping on monitor	37
Threads waiting to lock monitor	0

TDA – Thread Dump Analyzer 2.3.4

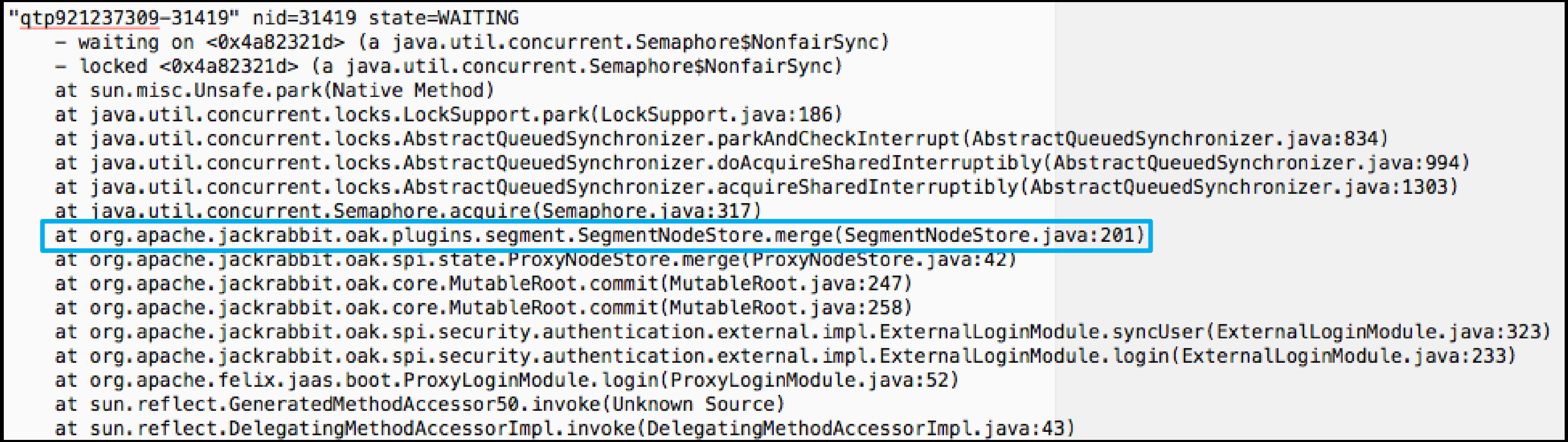
467MB/661.5MB

adaptTo()

How to proceed?

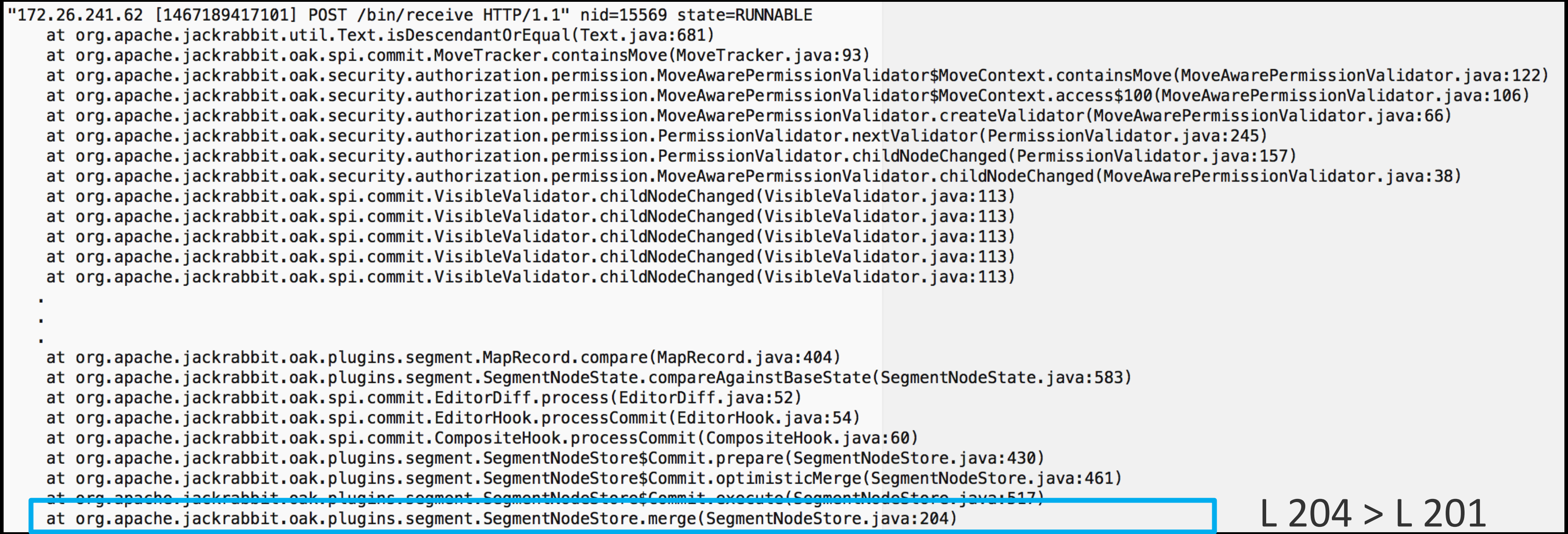
- How to identify thread that owns the lock?
- Looking at the all threads blocked on the monitor do folowing
- Identify the class that is using java.util.concurent
- Remember the class, method and line number
  - For example **at org.examlpe.MyClass.method1(MyClass.java:100)**
- Some threads can have different method from the same class
  - For example **at org.examlpe.MyClass.method2(MyClass.java:200)**
- Locking thread candidates (thread that potentially owns the lock)
  - Really good one: has the same class and method(s) in stack but greater line number
  - Has the same class **org.examlpe.MyClass** but different method

One of the blocked threads



```
"qtp921237309-31419" nid=31419 state=WAITING
    - waiting on <0x4a82321d> (a java.util.concurrent.Semaphore$NonfairSync)
    - locked <0x4a82321d> (a java.util.concurrent.Semaphore$NonfairSync)
    at sun.misc.Unsafe.park(Native Method)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:186)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(AbstractQueuedSynchronizer.java:834)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireSharedInterruptibly(AbstractQueuedSynchronizer.java:994)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireSharedInterruptibly(AbstractQueuedSynchronizer.java:1303)
    at java.util.concurrent.Semaphore.acquire(Semaphore.java:317)
    at org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStore.merge(SegmentNodeStore.java:201)
    at org.apache.jackrabbit.oak.spi.state.ProxyNodeStore.merge(ProxyNodeStore.java:42)
    at org.apache.jackrabbit.oak.core.MutableRoot.commit(MutableRoot.java:247)
    at org.apache.jackrabbit.oak.core.MutableRoot.commit(MutableRoot.java:258)
    at org.apache.jackrabbit.oak.spi.security.authentication.external.impl.ExternalLoginModule.syncUser(ExternalLoginModule.java:323)
    at org.apache.jackrabbit.oak.spi.security.authentication.external.impl.ExternalLoginModule.login(ExternalLoginModule.java:233)
    at org.apache.felix.jaas.boot.ProxyLoginModule.login(ProxyLoginModule.java:52)
    at sun.reflect.GeneratedMethodAccessor50.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
```

Blocking thread

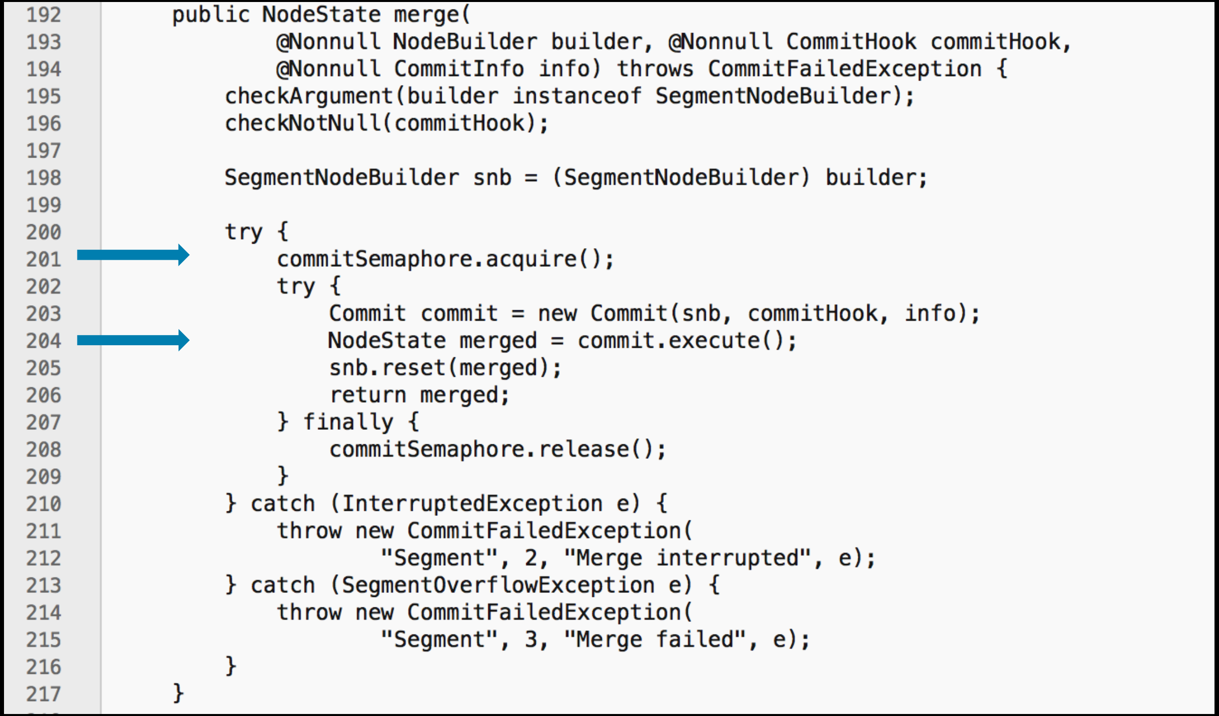


```
"172.26.241.62 [1467189417101] POST /bin/receive HTTP/1.1" nid=15569 state=RUNNABLE
    at org.apache.jackrabbit.util.Text.isDescendantOrEqual(Text.java:681)
    at org.apache.jackrabbit.oak.spi.commit.MoveTracker.containsMove(MoveTracker.java:93)
    at org.apache.jackrabbit.oak.security.authorization.permission.MoveAwarePermissionValidator$MoveContext.containsMove(MoveAwarePermissionValidator.java:122)
    at org.apache.jackrabbit.oak.security.authorization.permission.MoveAwarePermissionValidator$MoveContext.access$100(MoveAwarePermissionValidator.java:106)
    at org.apache.jackrabbit.oak.security.authorization.permission.MoveAwarePermissionValidator.createValidator(MoveAwarePermissionValidator.java:66)
    at org.apache.jackrabbit.oak.security.authorization.permission.PermissionValidator.nextValidator(PermissionValidator.java:245)
    at org.apache.jackrabbit.oak.security.authorization.permission.PermissionValidator.childNodeChanged(PermissionValidator.java:157)
    at org.apache.jackrabbit.oak.security.authorization.permission.MoveAwarePermissionValidator.childNodeChanged(MoveAwarePermissionValidator.java:38)
    at org.apache.jackrabbit.oak.spi.commit.VisibleValidator.childNodeChanged(VisibleValidator.java:113)
    at org.apache.jackrabbit.oak.spi.commit.VisibleValidator.childNodeChanged(VisibleValidator.java:113)
    at org.apache.jackrabbit.oak.spi.commit.VisibleValidator.childNodeChanged(VisibleValidator.java:113)
    at org.apache.jackrabbit.oak.spi.commit.VisibleValidator.childNodeChanged(VisibleValidator.java:113)
    at org.apache.jackrabbit.oak.spi.commit.VisibleValidator.childNodeChanged(VisibleValidator.java:113)
   .
   .
   .
    at org.apache.jackrabbit.oak.plugins.segment.MapRecord.compare(MapRecord.java:404)
    at org.apache.jackrabbit.oak.plugins.segment.SegmentNodeState.compareAgainstBaseState(SegmentNodeState.java:583)
    at org.apache.jackrabbit.oak.spi.commit.EditorDiff.process(EditorDiff.java:52)
    at org.apache.jackrabbit.oak.spi.commit.EditorHook.processCommit(EditorHook.java:54)
    at org.apache.jackrabbit.oak.spi.commit.CompositeHook.processCommit(CompositeHook.java:60)
    at org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStore$Commit.prepare(SegmentNodeStore.java:430)
    at org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStore$Commit.optimisticMerge(SegmentNodeStore.java:461)
    at org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStore$Commit.execute(SegmentNodeStore.java:517)
    at org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStore.merge(SegmentNodeStore.java:204)
```

L 204 > L 201

- Replication process was blocking all other threads

SegmentNodeStore.java



```
    public NodeState merge(
            @Nonnull NodeBuilder builder, @Nonnull CommitHook commitHook,
            @Nonnull CommitInfo info) throws CommitFailedException {
        checkArgument(builder instanceof SegmentNodeBuilder);
        checkNotNull(commitHook);

        SegmentNodeBuilder snb = (SegmentNodeBuilder) builder;

        try {
            commitSemaphore.acquire();
            try {
                Commit commit = new Commit(snb, commitHook, info);
                NodeState merged = commit.execute();
                snb.reset(merged);
                return merged;
            } finally {
                commitSemaphore.release();
            }
        } catch (InterruptedException e) {
            throw new CommitFailedException(
                    "Segment", 2, "Merge interrupted", e);
        } catch (SegmentOverflowException e) {
            throw new CommitFailedException(
                    "Segment", 3, "Merge failed", e);
        }
    }
```

- 37 threads are waiting to acquire the monitor
- POST /bin/receive holds the lock

Demo

Long running thread

- If possible it is good to have more than one thread
- It gives possibility to discover long lasting threads
- From there it is possible to continue investigation by increasing logging level for modules that appear in stack trace

Tools

adaptTo()

Tools

- TDA – Thread Dump Analyzer
  - https://github.com/irockel/tda
- IBM Thread and Monitor Dump Analyzer
- http://fastthread.io/index.jsp

Links

- http://javaeesupportpatterns.blogspot.com/p/thread-dump-analysis.html
- https://helpx.adobe.com/experience-manager/kb/thread-dump-analysis.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/hangloop002.html