#	Source Code		Simplified Code	
	Original	Modified	Original	Modified

1	`LeafQueue leafQueue = ...;`
	`-synchronized (leafQueue) {`
	`  57 LOC`
	`}`
		`LeafQueue leafQueue = ...;`
		`+try { leafQueue.getReadLock().lock();`
		`  57 LOC`
		`+} finally {`
		`+  leafQueue.getReadLock().unlock();}`
			`synchronized (obj)`
			`{`
			`  ...`
			`}`
				`try {obj.lock();`
				`  ...`
				`} finally {`
				`  obj.unlock();`
				`}`

2	`-Lock readlock =`
	`-  classLoaderContainerMapLock.readLock();`
	`-try { readlock.lock();`
	`-  result = classLoaderContainerMap.get(tccl);`
	`-} finally {readlock.unlock();}`
	`-if (result == null) { Lock writelock =`
	`-  classLoaderContainerMapLock.writeLock();`
	`-  try { writeLock.lock();`
	`  result = classLoaderContainerMap.get(tccl);`
	`  if (result == null) {`
	`  result = new ServerContainerImpl();`
	`  classLoaderContainerMap.put(tccl,result);}`
	`-  } finally {writeLock.unlock();}`
		`+synchronized (classLoaderContainerMapLock) {`
		
		`  result = classLoaderContainerMap.get(tccl);`
		`  if (result == null) {`
		`  result = new ServerContainerImpl();`
		`  classLoaderContainerMap.put(tccl,result);}`
		`}`
			`try {`
			`  readLock.lock();`
			`  read operations`
			`} finally {`
			`  readLock.unlock();`
			`}`
			`try {`
			`  writeLock.lock();`
			`  write operations`
			`} finally {`
			`writeLock.unlock();`
			`}`
				
				`synchronized {`
				`  all operations`
				`}`

3	`static final Object lock = new Object();`
	`Map<...> count = new HashMap<>();`
	`-synchronized (count) {`
	`-  Pair<Job, String> key =`
	`-  new ImmutablePair<>(jobID, name);`
	
	`-  if (count.containsKey(key)) {`
	`-  count.put(key, count.get(key) + 1);`
	`-  } else {count.put(key, 1);}}`
		`static final Object lock = new Object();`
		`Map<...> count = new HashMap<>();`
		`+synchronized(lock)`
		`+  if (!jobCounts.containsKey(jobID)) {`
		`+  jobCounts.put(jobID, new HashMap<>());}`
		`+  Map<...> count = jobCounts.get(jobID);`
		`+  if (count.containsKey(name)) {`
		`+  count.put(name, count.get(name) + 1);`
		`+  } else {count.put(name, 1);}} `
			`synchronized (obj1)`
			`{`
			`  ...`
			`}`
				`synchronized`
				`(obj2) {`
				`  ...`
				`}`

4	`-public boolean isAccessed() {` `  return this.accessed;}`	`+public synchronized boolean isAccessed() {` `  return this.accessed;}`	`void foo() {...}`	`synchronized void` `foo() {...}`

5	`synchronized (buffers) { if (...) {`
	
	`-  if (spillWriter != null) {`
	`-  spillWriter.close();}`
	`  isFinished = true;}}`
		`synchronized (buffers) { if (...) {`
		`  isFinished = true;}}`
		`+if (spillWriter != null) {`
		`+  spillWriter.close();}`
			`synchronized(obj)`
			`{`
			`  statements1`
			`  statements2 }`
				`synchronized(obj)`
				`{`
				`  statements2 }`
				`Statements1`

6	`-synchronized void reset() {`
	
	`  map.clear();`
	`  members = EMPTY_MEMBERS;}`
		`+final Object membersLock = new Object();`
		`+void reset() { synchronized (membersLock) {`
		`  map.clear();`
		`  members = EMPTY_MEMBERS;}}`
			`synchronized void`
			`foo() {`
			`  ... }`
				`void foo() {`
				`  synchronized`
				`(obj) {`
				`  ... }}`

7	`-synchronized void enqueue(final long seqno,`
	`  final boolean lastPacketInBlock,`
	`  final long offsetInBlock) {`
	`-  if (running) {`
	`  final Packet p = new Packet(...);`
	`  LOG.debug(...);`
	
	`  ackQueue.addLast(p); notifyAll();}}`
		`+void enqueue(final long seqno,`
		`  final boolean lastPacketInBlock,`
		`  final long offsetInBlock) {`
		
		`  final Packet p = new Packet(...);`
		`  LOG.debug(...);`
		`+  synchronized (this) { if (running) {`
		`  ackQueue.addLast(p); notifyAll();}}}`
			`synchronized void`
			`foo(...) {`
			`  statements1`
			`  statements2`
			`}`
				`statements1`
				`synchronized (obj)`
				`{`
				`  statements2`
				`}`

8	`-Membership membership = null;`
	`public boolean hasMembers() {`
	`  if (membership == null) setupMembership();`
	`  return membership.hasMembers();}`
	`synchronized void setupMembership() {`
	`  if ( membership == null ) {`
	`  membership = new Membership(...);}}`
		`+volatile Membership membership = null;`
		`public boolean hasMembers() {`
		`  if (membership == null) setupMembership();`
		`  return membership.hasMembers();}`
		`synchronized void setupMembership() {`
		`  if ( membership == null ) {`
		`  membership = new Membership(...);}}`
			`T foo;`	`volatile T foo;`

9	`-volatile int requestCount;`
	
	`-  requestCount++;`
		`+final AtomicInteger requestCount =`
		`+  new AtomicInteger(0);`
		`+  requestCount.incrementAndGet();`
			`volatile T foo;`	`TT foo;`

#	Source Code		Simplified Code	
	Original	Modified	Original	Modified

7	-synchronized void enqueue(final long seqno,
	  final boolean lastPacketInBlock,
	  final long offsetInBlock) {
	- if (running) {
	  final Packet p = new Packet(...);
	  LOG.debug(...);
	
	  ackQueue.addLast(p);
	  notifyAll();
	  }
	}
		+void enqueue(final long seqno,
		  final boolean lastPacketInBlock,
		  final long offsetInBlock) {
		
		  final Packet p = new Packet(...);
		  LOG.debug(...);
		+ synchronized (this)
		+ if (running) {
		+ ackQueue.addLast(p);
		+ notifyAll();
		  }
		  }
			synchronized void
			foo() {
			  statements1
			  statements2
			}
				statements1
				synchronized (obj)
				{
				  statements2
				}

8	synchronized (buffers) {
	  if (...) {
	- if (spillWriter != null) {
	- spillWriter.close();
	- }
	  isFinished = true;
	  }
	}
		synchronized (buffers) {
		  if (...) {
		  isFinished = true;
		  }
		}
		+if (spillWriter != null) {
		+ spillWriter.close();
		+}
			synchronized (obj)
			{
			  statements1
			  statements2
			}
				synchronized (obj)
				{
				  statements2
				}
				statements1

9	-Membership membership = null;
	public boolean hasMembers() {
	  if (membership == null) setupMembership();
	  return membership.hasMembers();
	}
	synchronized void setupMembership() {
	  if (membership == null) {
	  membership = new Membership(
	  super.getLocalMember(true));
	  }
	}
		+volatile Membership membership = null;
		public boolean hasMembers() {
		  if (membership == null) setupMembership();
		  return membership.hasMembers();
		}
		synchronized void setupMembership() {
		  if (membership == null) {
		  membership = new Membership(
		  super.getLocalMember(true));
		  }
		}
			T foo;	volatile T foo;

10	-private volatile int requestCount; -private volatile int errorCount; - requestCount++;	+private final AtomicInteger requestCount = + new AtomicInteger(0); +private final AtomicInteger errorCount = + new AtomicInteger(0); + requestCount.incrementAndGet();	volatile T foo;	TT foo;