

Асинхронный транспорт Cassandra

Вадим Цесько
Одноклассники



HighLoad⁺⁺
2022

Яндекс



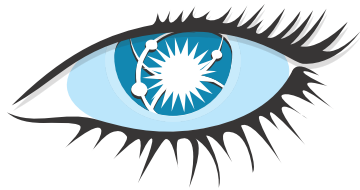
Одноклассники

- Техническая платформа VK^a
- > 12K серверов
- 7 ЦОДов
- > 4 Tb/c
- > 100K rps на ноду (4 ядра)
- p99 < 100 мс
- > 1 ЭБ

^aVK Видео, VK Звонки, RuStore и др.



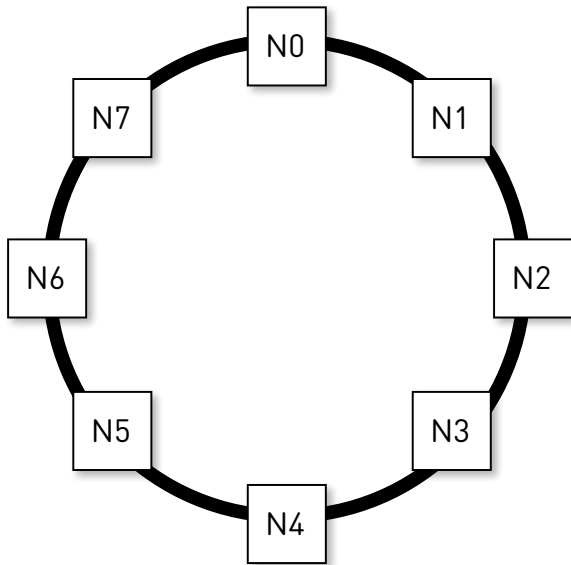
Cassandra¹ в ОК²

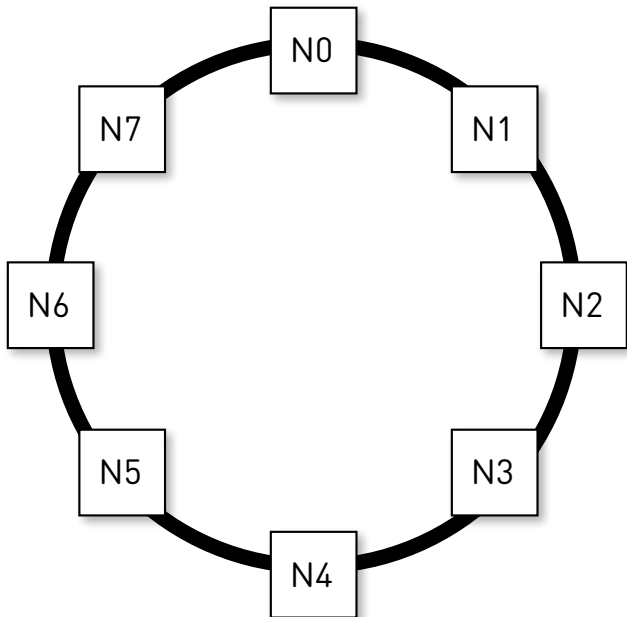


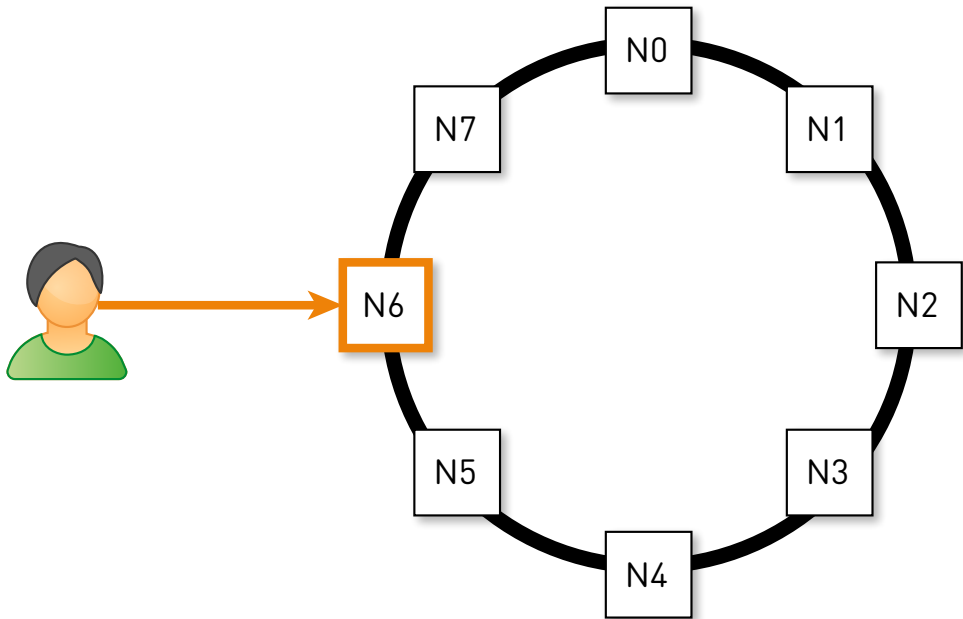
- **> 100 кластеров**
- **До 600 нод/кластер**
- **> 1K клиентов/кластер**
- **До 150K rps/ноду**
- **Спекулятивные запросы**

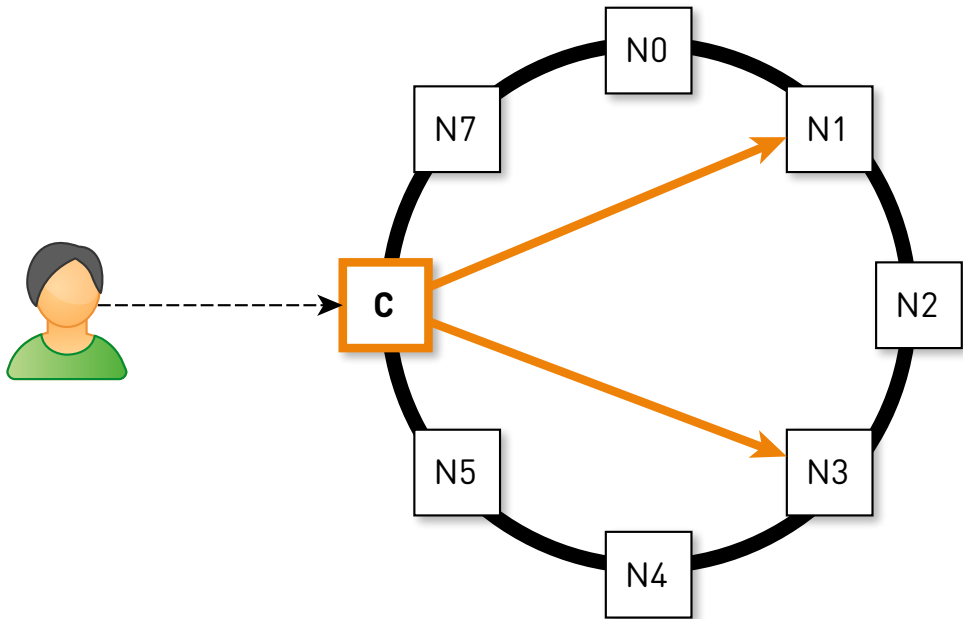
¹Олег Анастасьев. NewSQL = NoSQL+ACID

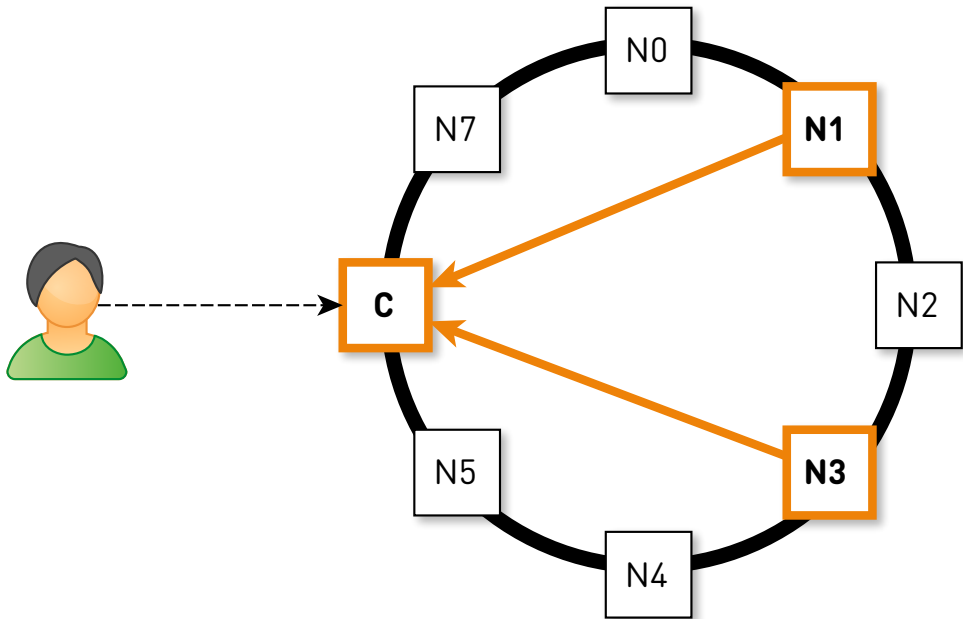
²Олег Анастасьев. Эффективные надежные микросервисы

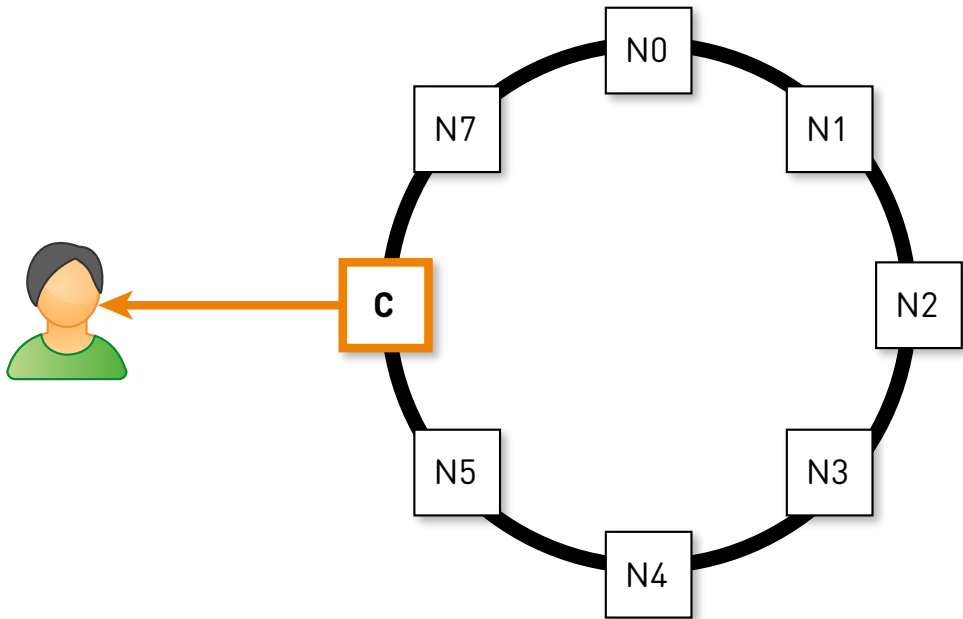


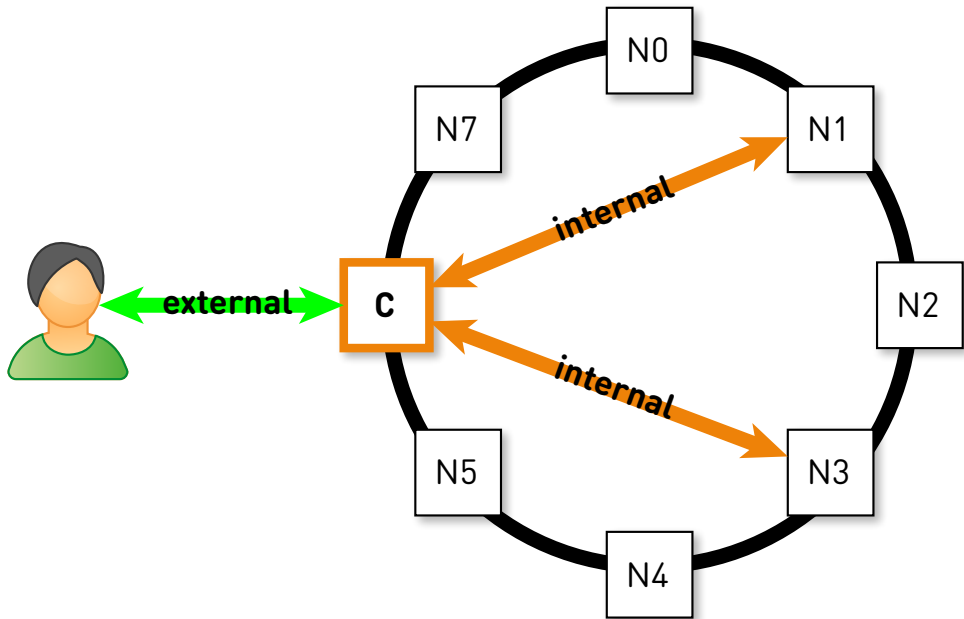


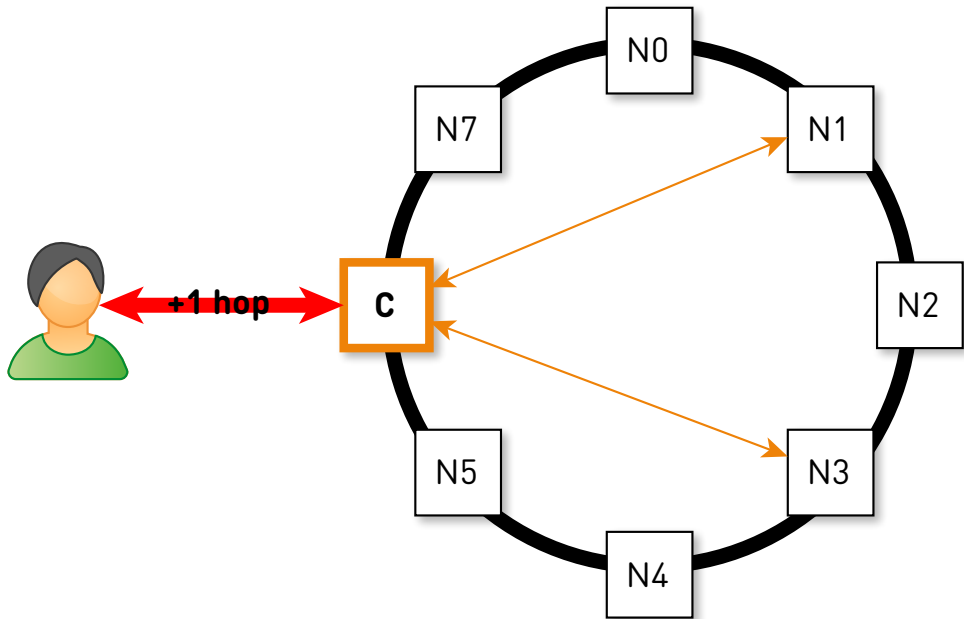


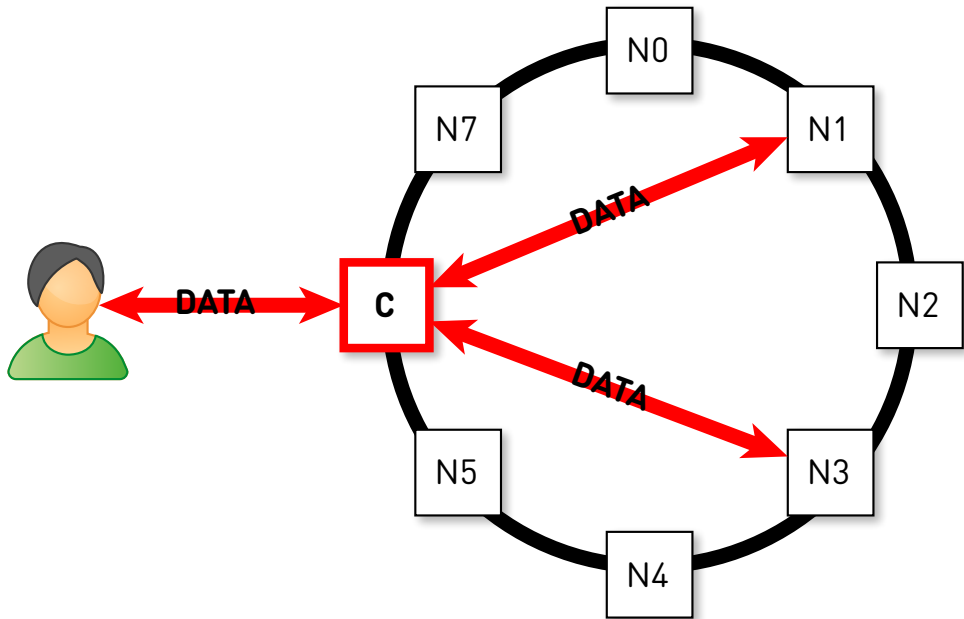


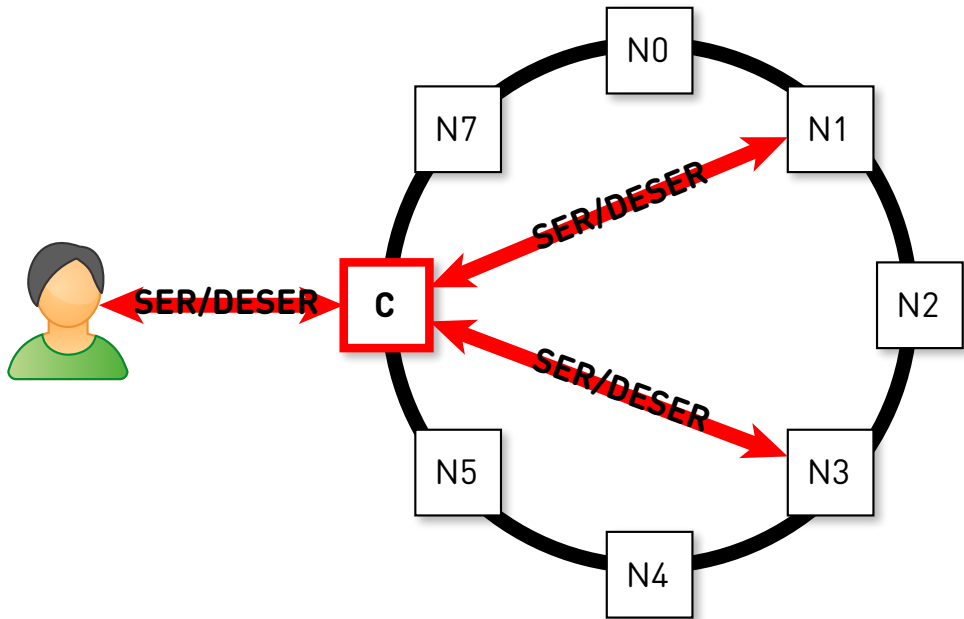


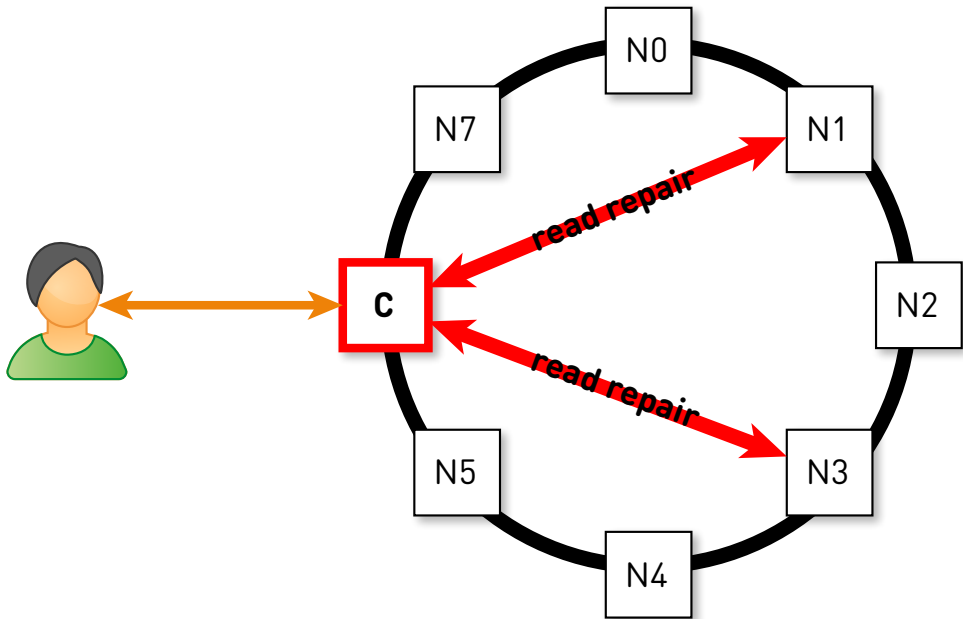


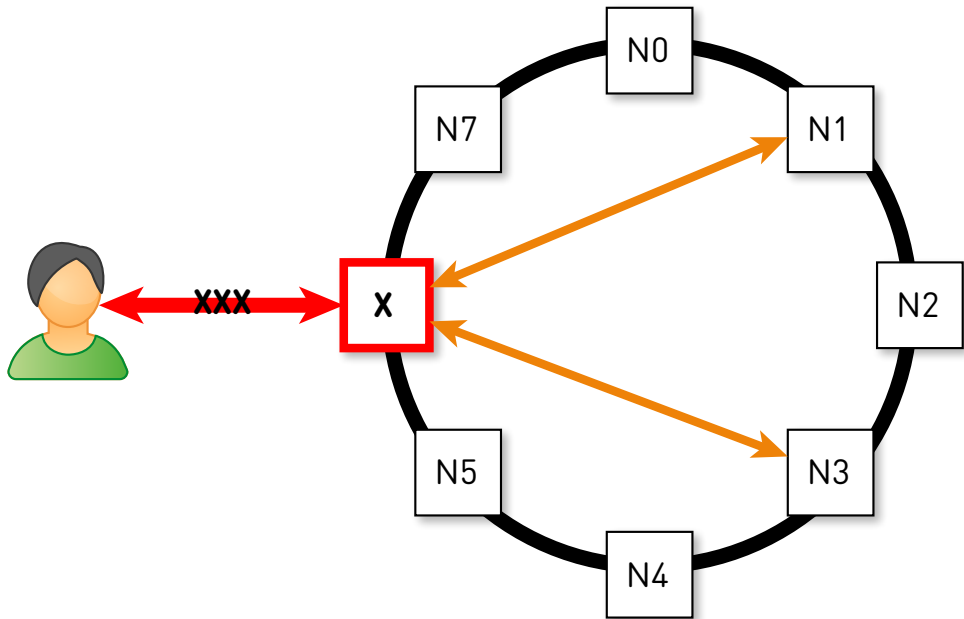


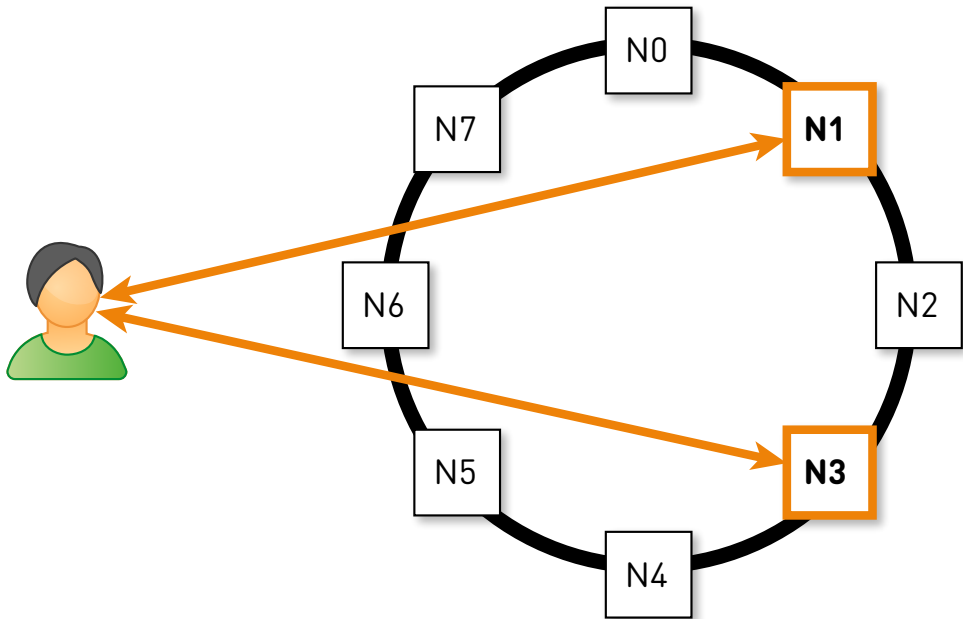


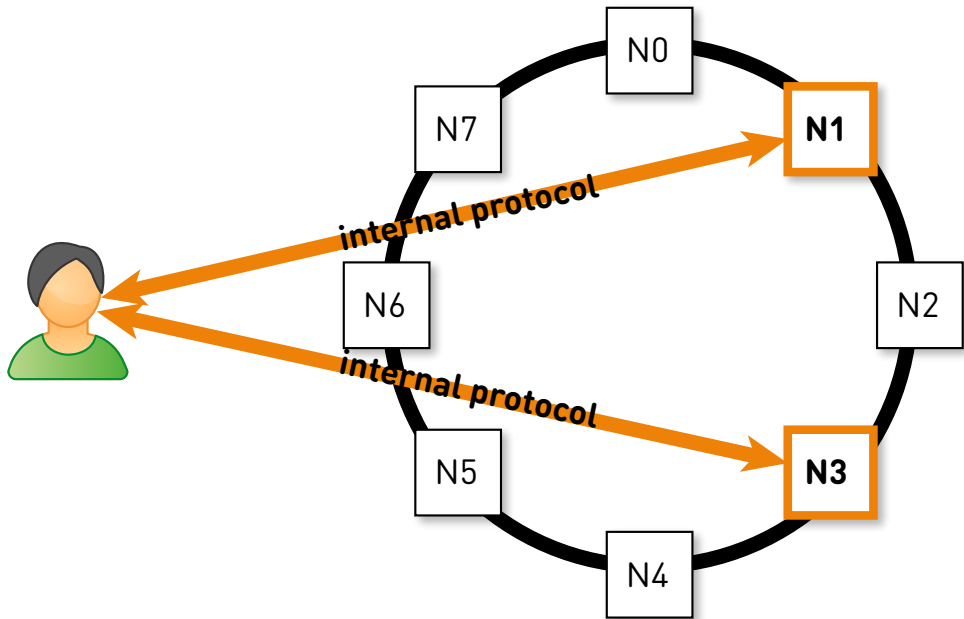


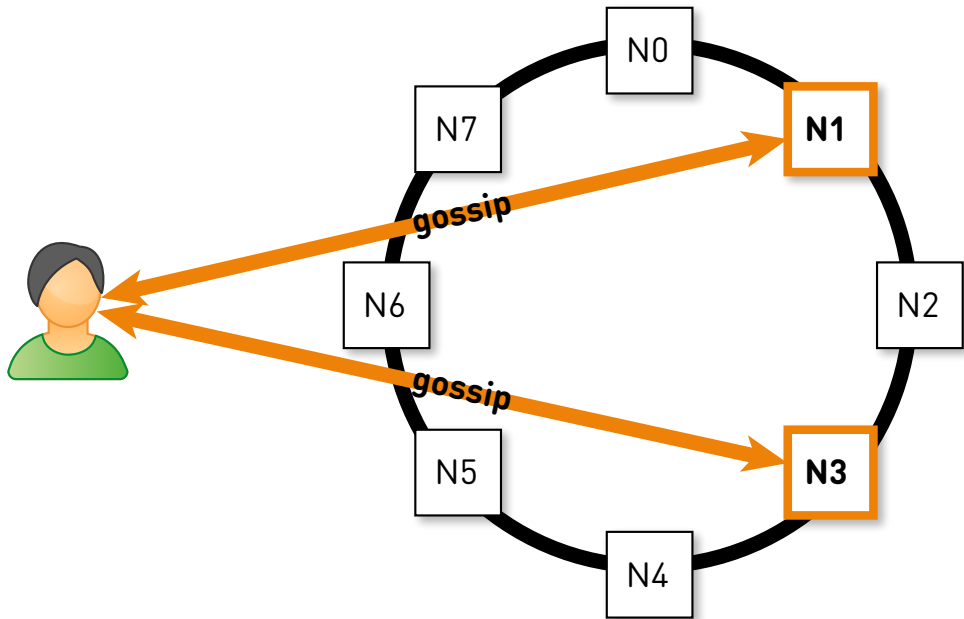


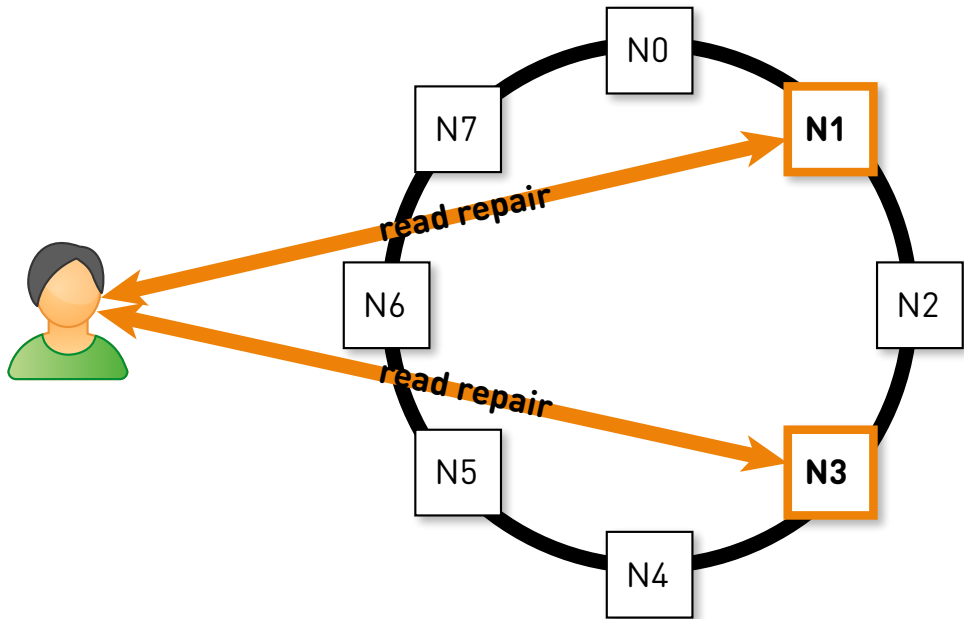


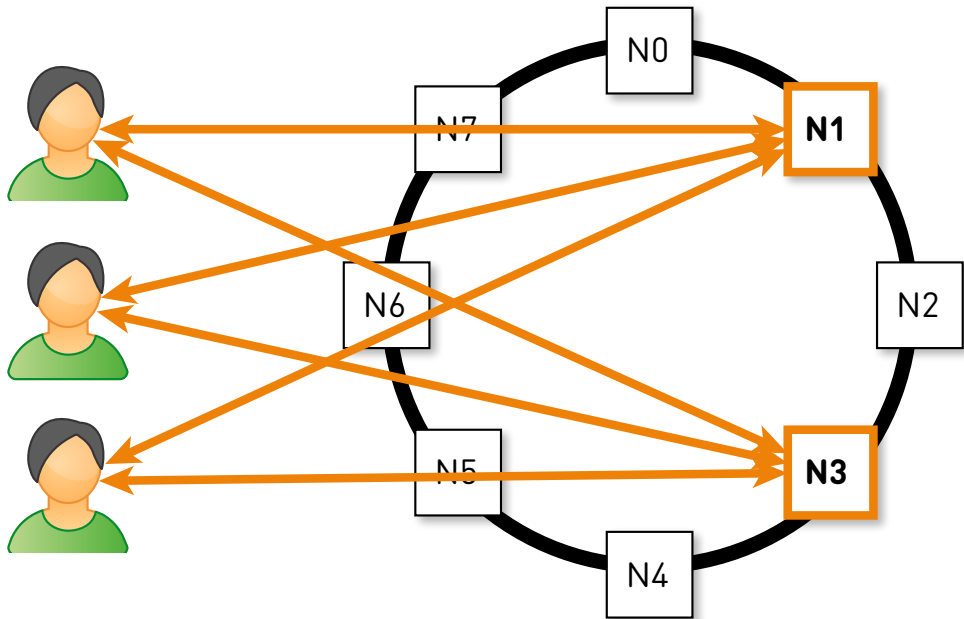








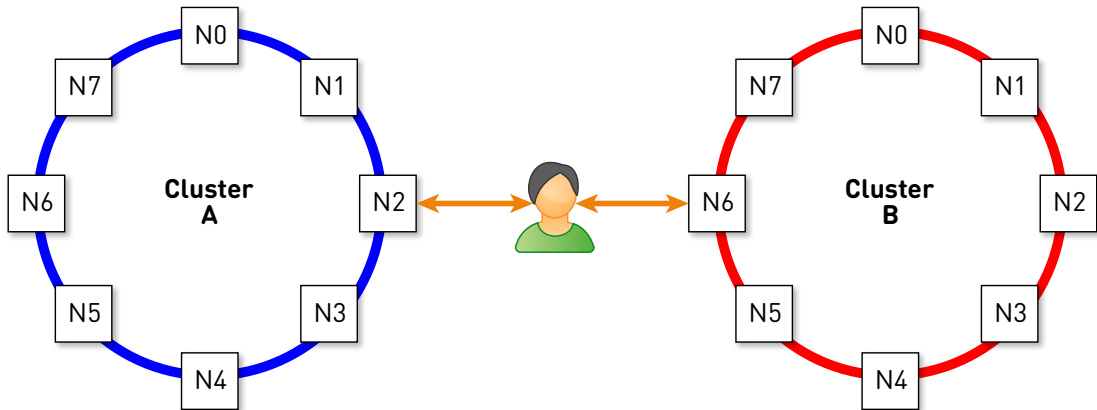




«Толстый» клиент

- Клиент — **участник** кластера
- Просто не хранит данные
- Общается по **внутреннему протоколу C***
- Участвует в **Gossip**
- **Координация** на клиентах
- ...
- Но есть нюансы





Singletons everywhere

```
1 private static class MSHandle {  
2     public static final IMessagingService instance =  
3         new MessagingService();  
4 }  
5  
6 public static IMessagingService instance() {  
7     return MSHandle.instance;  
8 }
```



Singletons everywhere

```
1 public static final Schema instance = new Schema();
```

```
1 public static final Gossiper instance = new Gossiper();
```

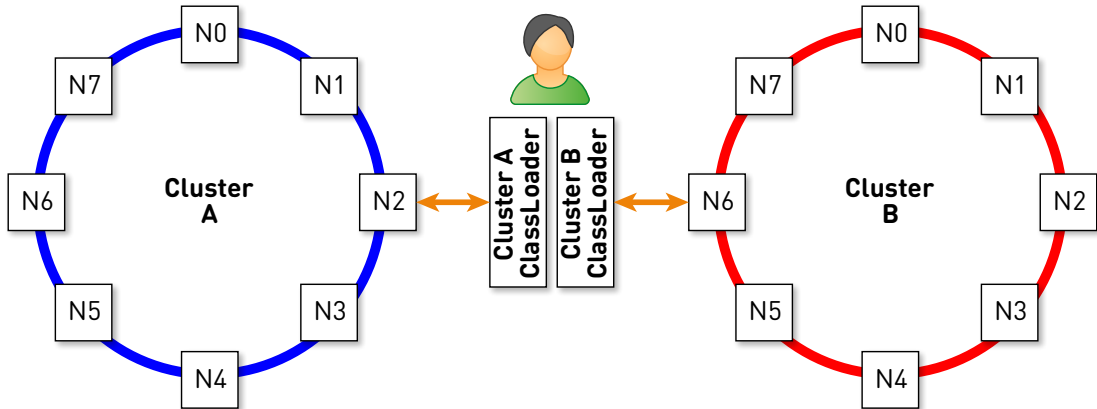
```
1 public static final IFailureDetector instance =  
2     new FailureDetector();
```

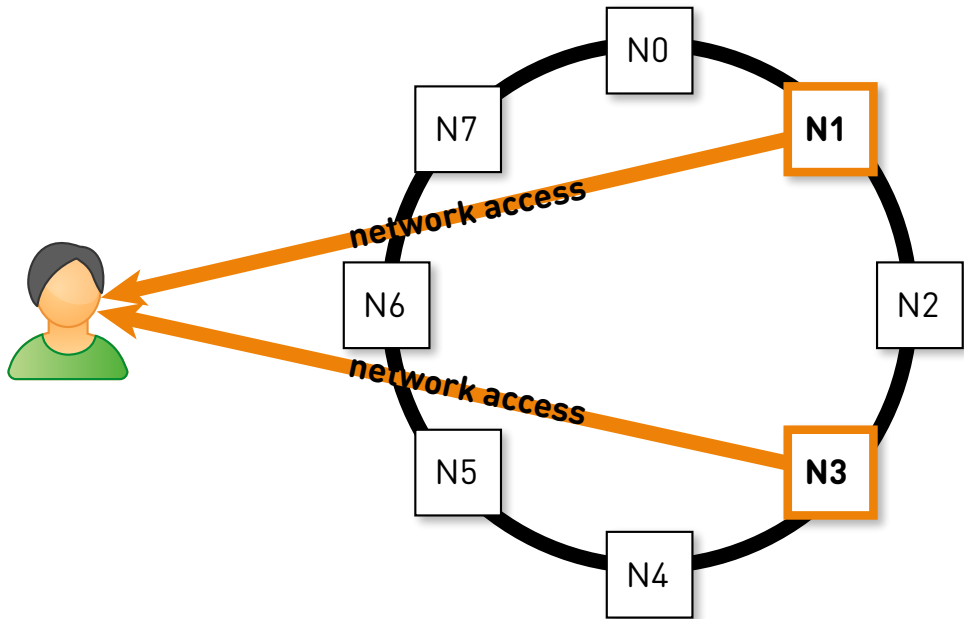
```
1 public static final StorageProxy instance =  
2     new StorageProxy();
```

```
1 public static final StorageService instance =  
2     new StorageService();
```

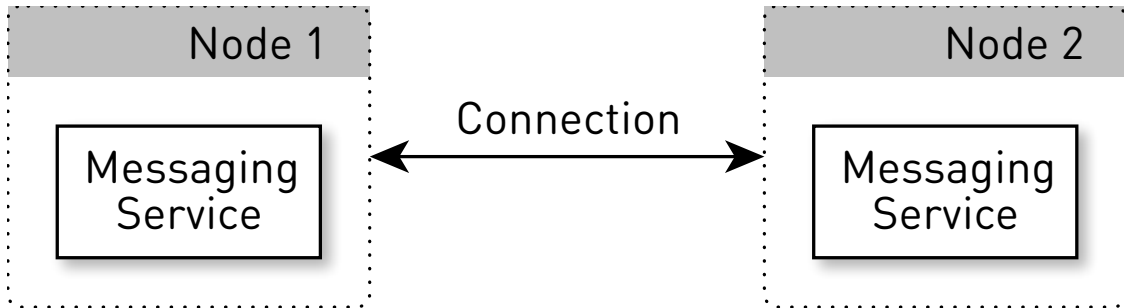


ClassLoader per cluster

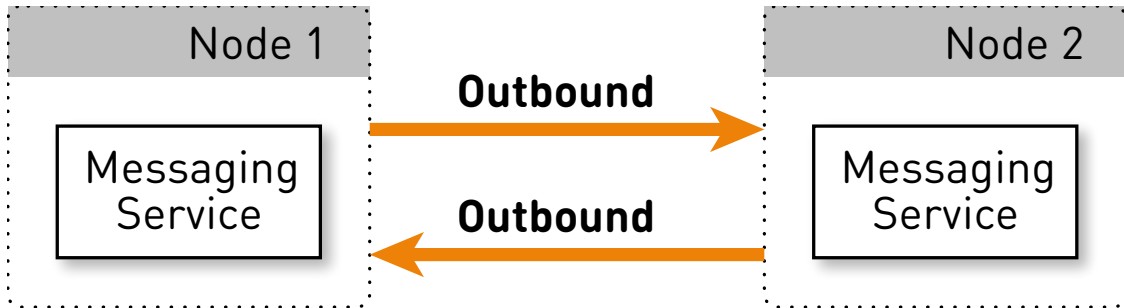




Внутренний транспорт



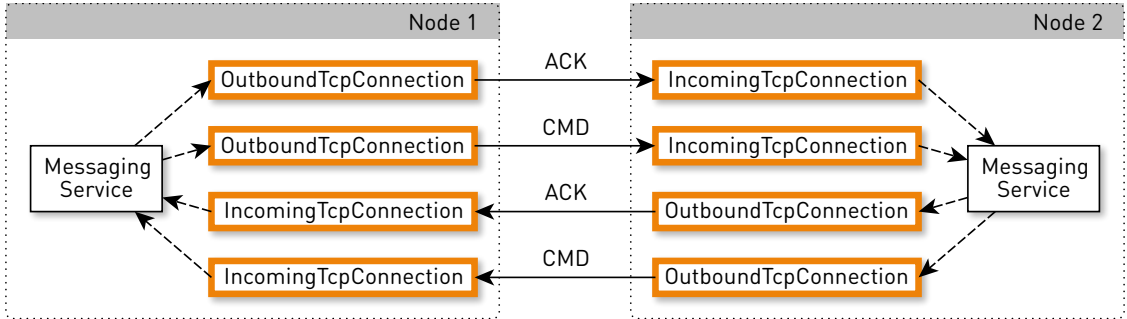
Односторонние соединения



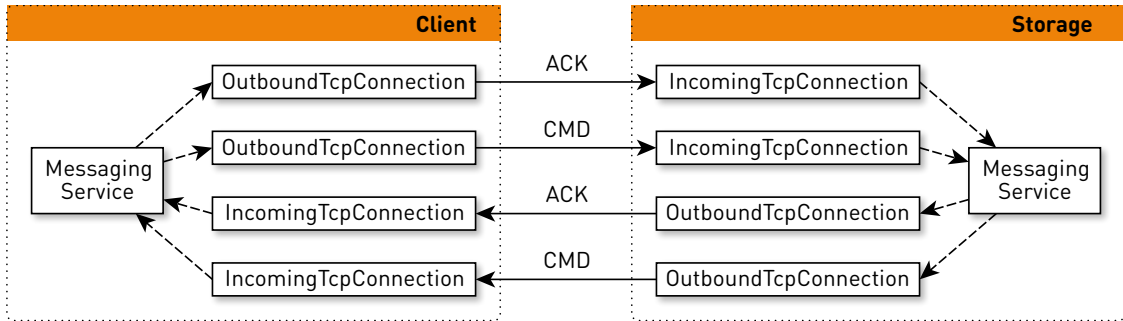
ACK и CMD



Thread per connection



Клиент — тоже участник кластера



Итого

- Каждый **клиент**: по **4 потока** для **каждой ноды**
- Каждая **нода**: по **4 потока** для **каждого клиента** и **каждой ноды**
- **Тысячи** соединений/**потоков** в каждом процессе
- Все соединения/потоки **активны**
- И сетевой доступ в **обе стороны**



Цели

- Эксплуатация
 - Соединения `client` $\rightarrow$ `node` и `node` $\leftrightarrow$ `node`
 - Поддержать **клиентов** за NAT/VPN
- Оптимизация
 - Обслуживать **небольшим** пулом потоков



Задачи

- Перейти на **Async IO**
- Использовать **входящие соединения**
- Передавать данные **в обе стороны**



MessagingService

```
1 void listen(InetAddress localEp);  
2  
3 void receive(MessageIn message);  
4  
5 void sendOneWay(MessageOut message, InetAddress to);  
6  
7 void sendReply(MessageIn request, MessageOut response);  
8  
9 void shutdown();
```



MessagingService

```
1 void listen(InetAddress localEp);  
2  
3 void receive(MessageIn message);  
4  
5 void sendOneWay(MessageOut message, InetAddress to);  
6  
7 void sendReply(MessageIn request, MessageOut response);  
8  
9 void shutdown();
```



MessagingService

```
1 void listen(InetAddress localEp);  
2  
3 void receive(MessageIn message);  
4  
5 void sendOneWay(MessageOut message, InetAddress to);  
6  
7 void sendReply(MessageIn request, MessageOut response);  
8  
9 void shutdown();
```



MessagingService

```
1 void listen(InetAddress localEp);  
2  
3 void receive(MessageIn message);  
4  
5 void sendOneWay(MessageOut message, InetAddress to);  
6  
7 void sendReply(MessageIn request, MessageOut response);  
8  
9 void shutdown();
```



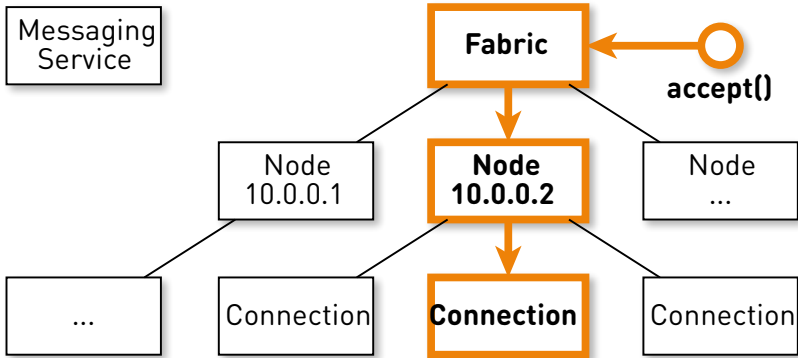


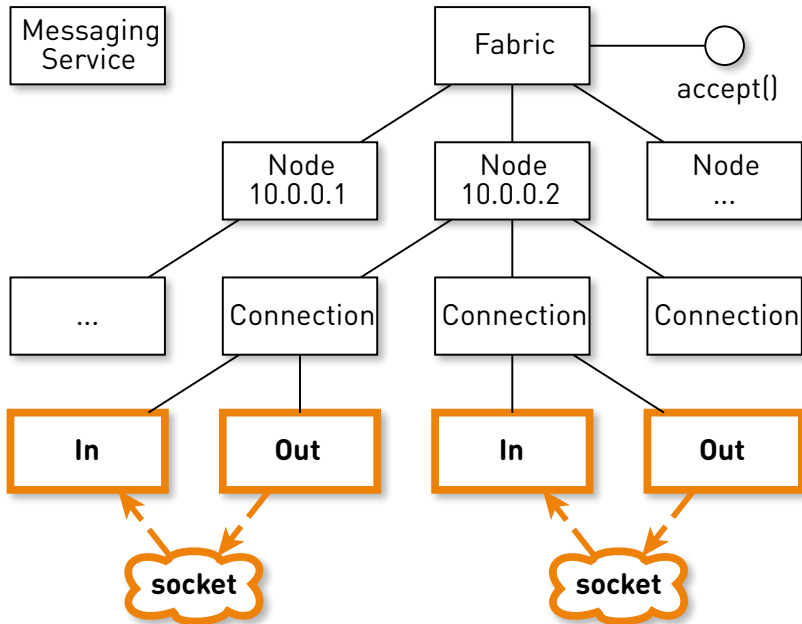
Messaging
Service

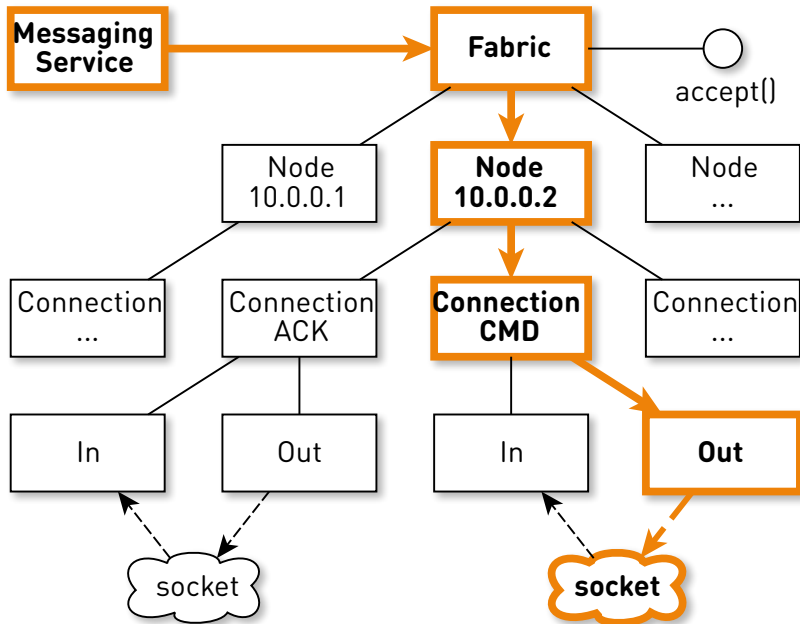
Fabric

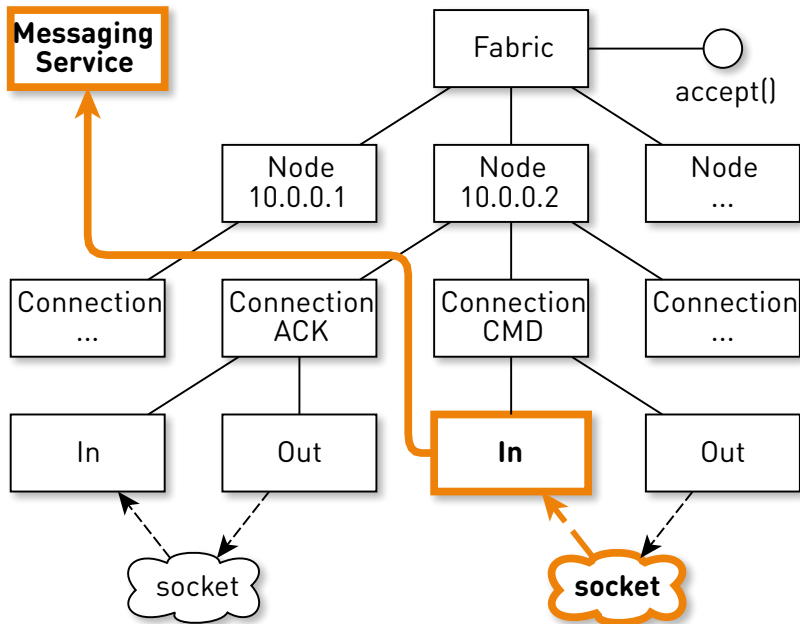
accept()











AsynchronousSocketChannel

```
1 public abstract <A> void connect(  
2     SocketAddress remote,  
3     A attachment,  
4     CompletionHandler<Void, ? super A> handler);  
5  
6 public abstract <A> void read/write(  
7     ByteBuffer dst/src,  
8     long timeout,  
9     TimeUnit unit,  
10    A attachment,  
11    CompletionHandler<Integer, ? super A> handler);
```



CompletionHandler

```
1 void completed(V result, A attachment);  
2  
3 void failed(Throwable exc, A attachment);
```



completed(V result, A attachment)

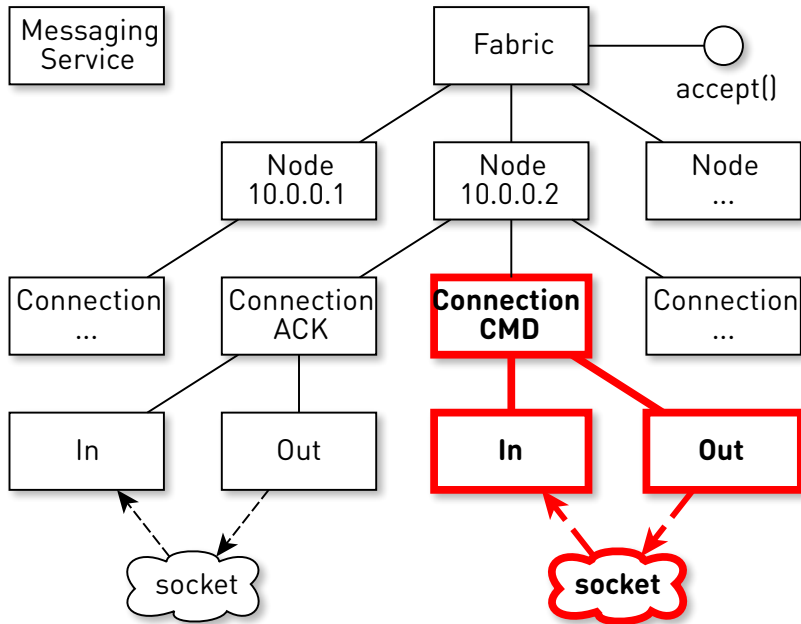
- Завершилась ли предыдущая запись/чтение?
- Полностью ли записали/прочитали?
- Можно начинать парсить/отправлять следующее сообщение?
- В каком потоке вызвался callback?
- Что и от кого нужно защищать?
- ...

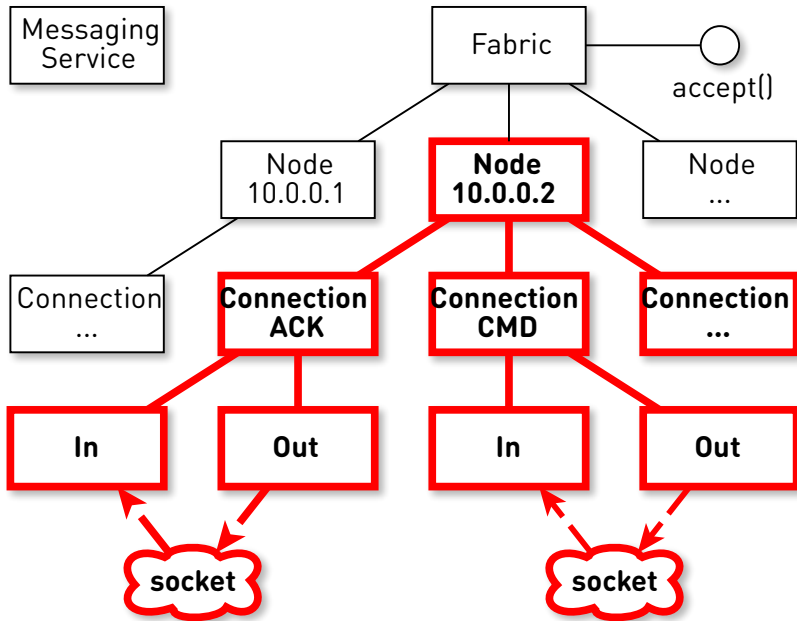


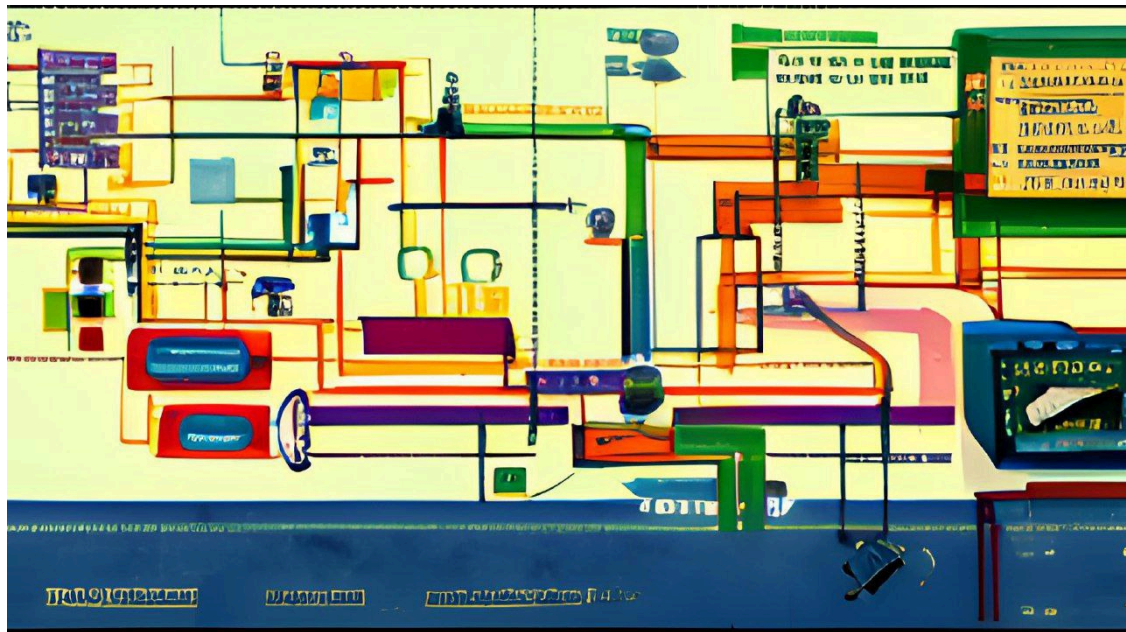
CompletionHandler

```
1 void completed(V result, A attachment);  
2  
3 void failed(Throwable exc, A attachment);
```









Что мы хотим?

- **Простой последовательный** код
- Обслуживать **тысячи соединений**
- На **небольшом пуле** потоков



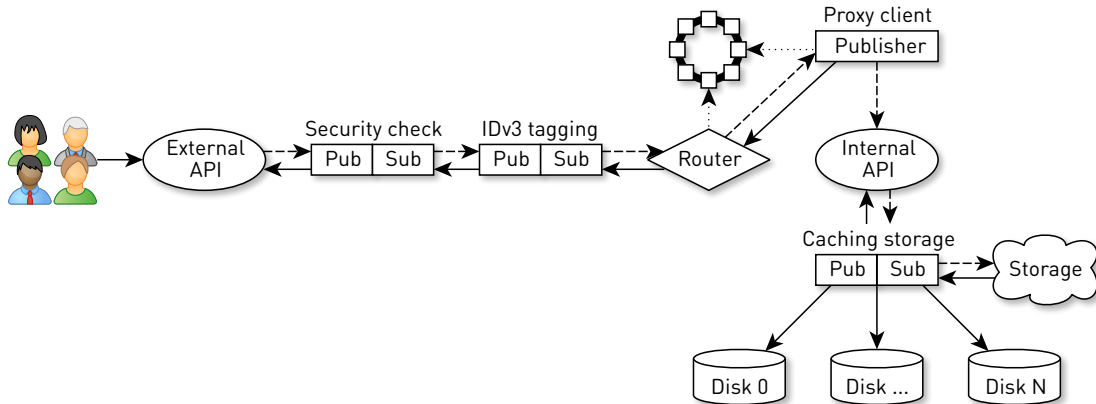
Joker<?> 2018

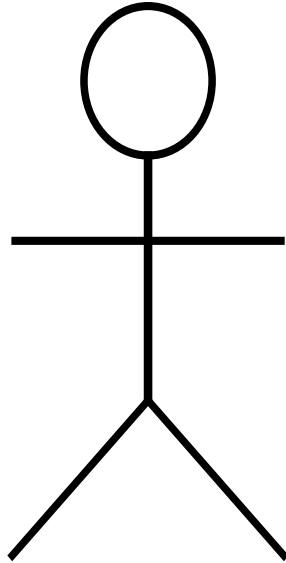
Вадим Цесько
Одноклассники

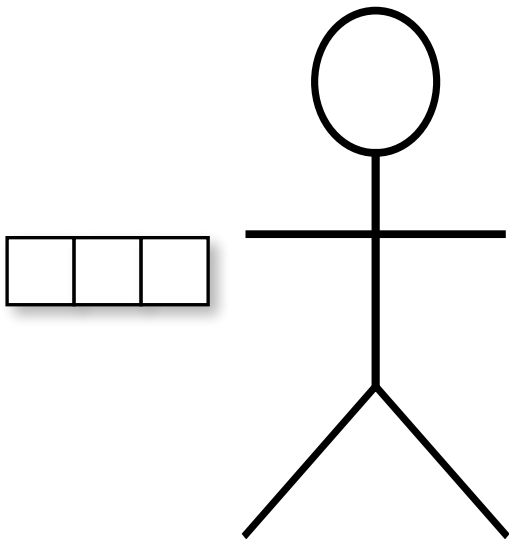
Реактивный раздатчик
ok.ru/music

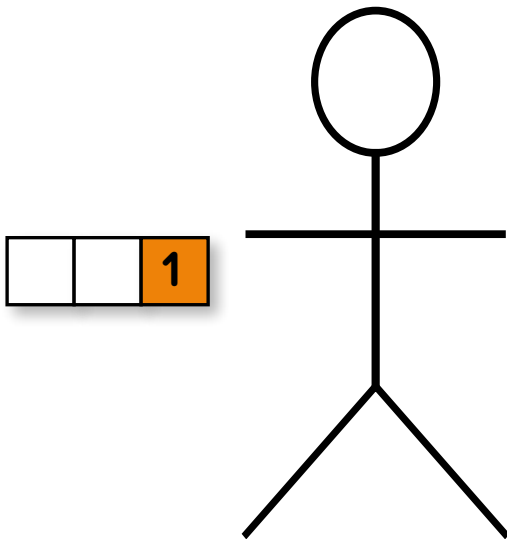


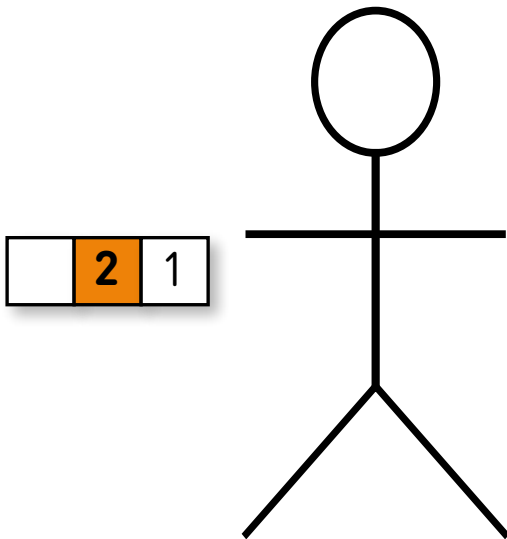
Actor Model в помощь

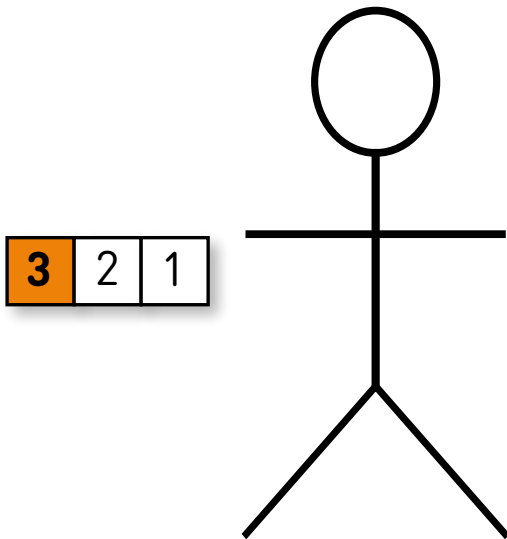


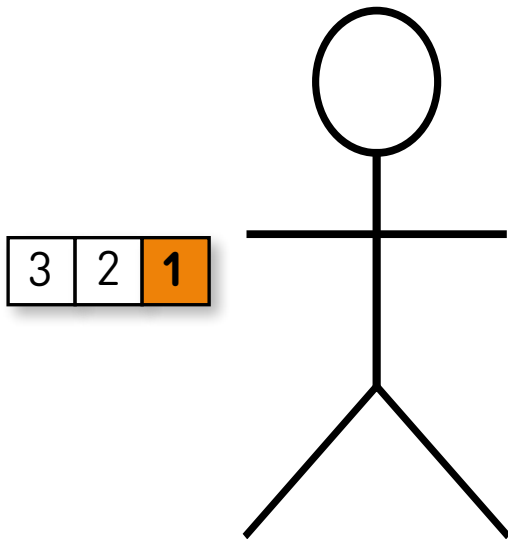


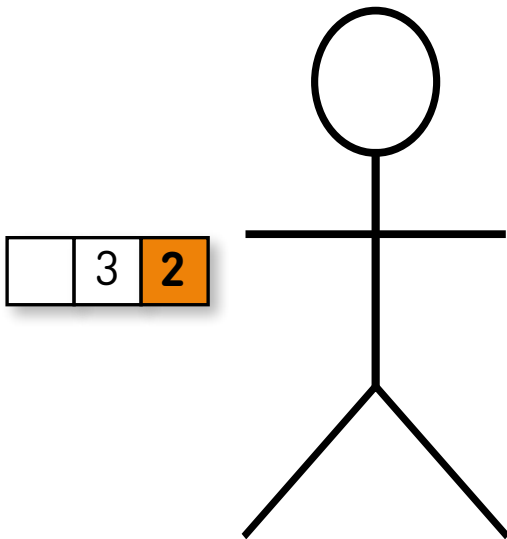


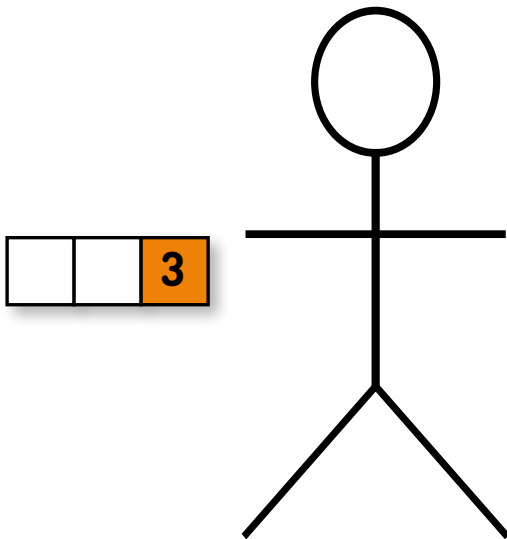


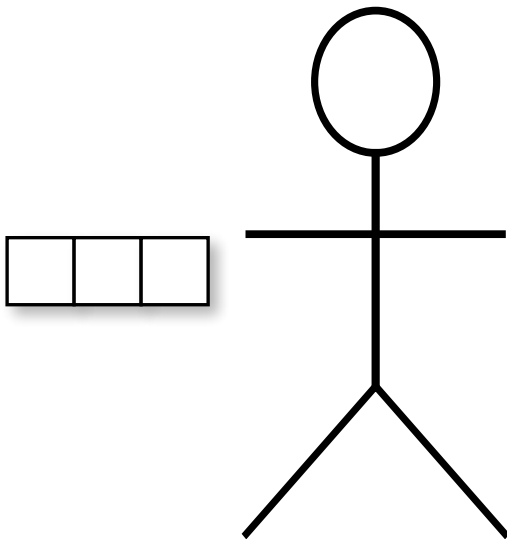


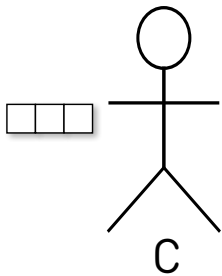
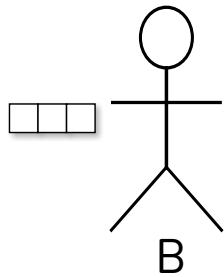
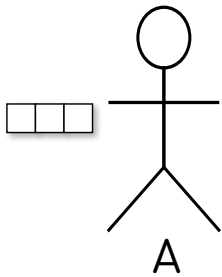


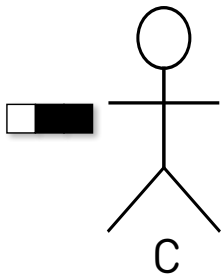
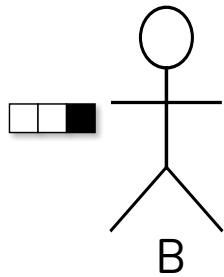
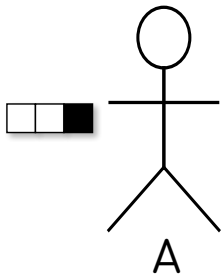


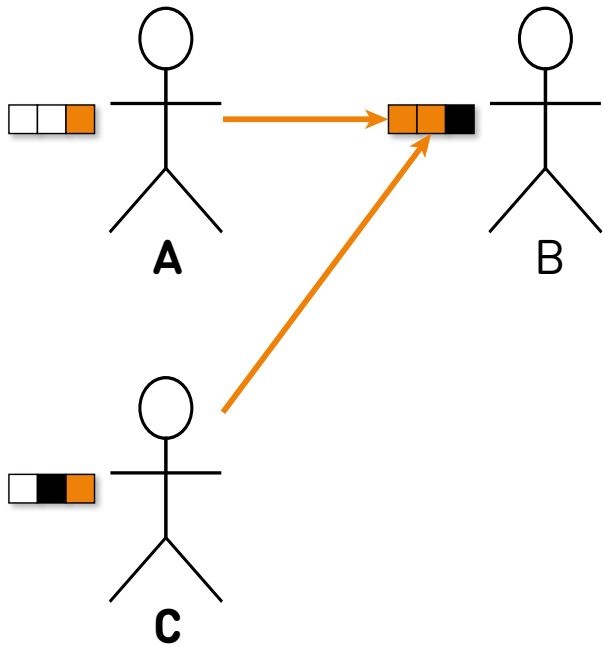


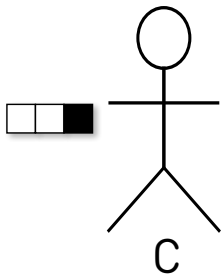
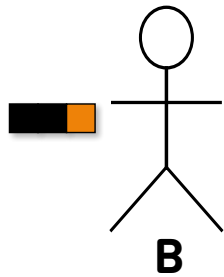
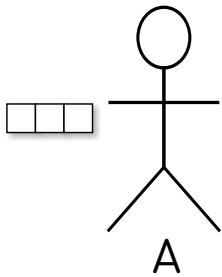


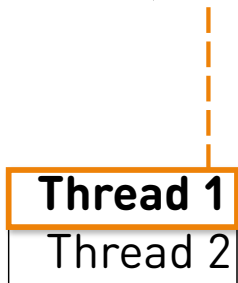
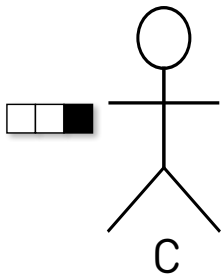
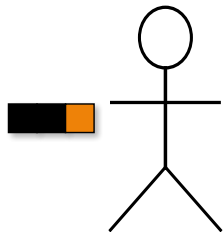
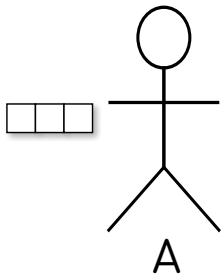


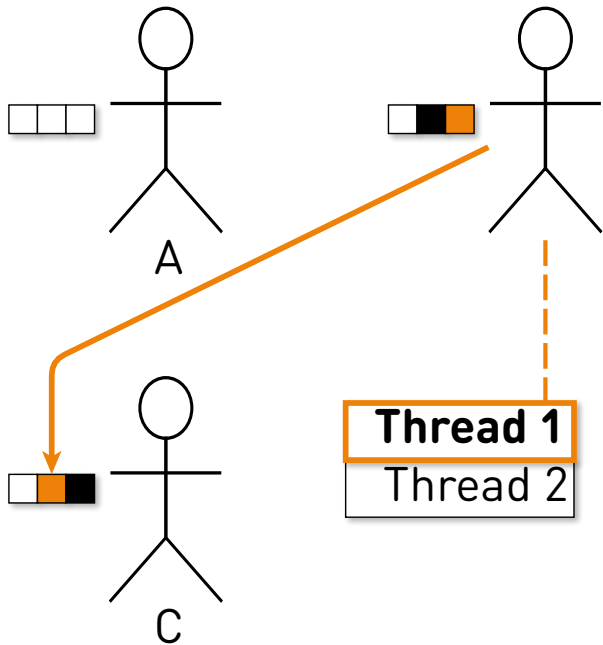


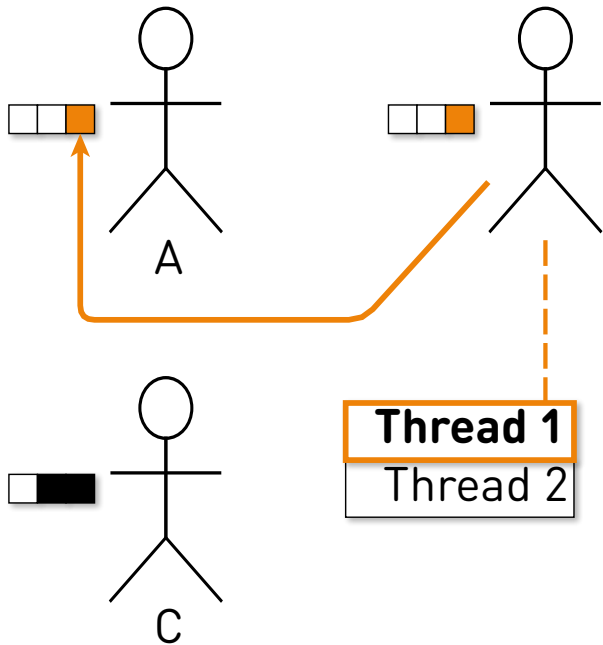


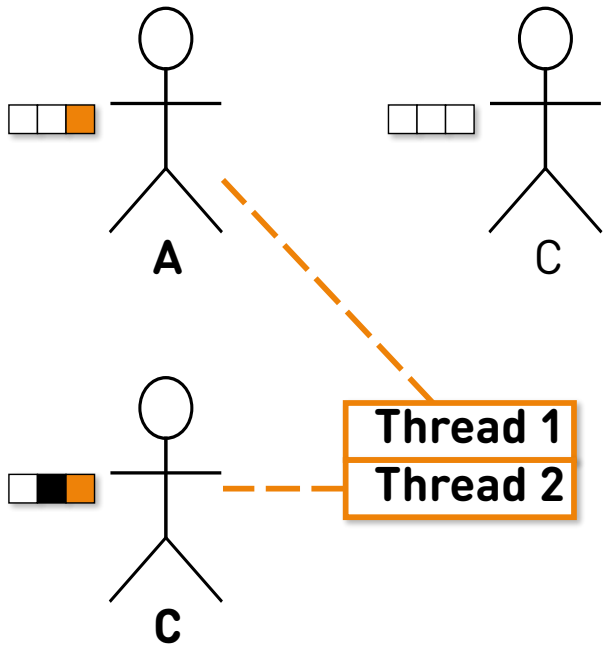












Ссылки

- **Потоковая обработка данных с помощью Actor Model @ ADD 2012**
- **Акка в Яндекс @ JPoint 2014**
- **Реактивный раздатчик ok.ru/music @ Joker 2018**

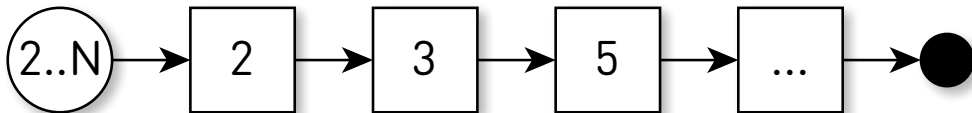


one-actor

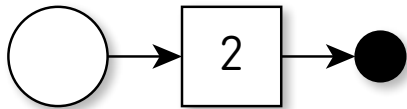
```
1 /**
2  * Implements sieve of Eratosthenes.
3  * <p>
4  * Allocates a child {@link Divisor} when a new prime is found.
5  * Sifts candidates to the child if can't divide them.
6  * <p>
7  * Supports recursive {@link #collect(Collection)} request which
8  * {@link #stop()}s all the children in the chain starting from
9  * the last item.
10 *
11 * @author incubos
12 */
13 class Divisor extends Actor<Divisor.Message> {
```



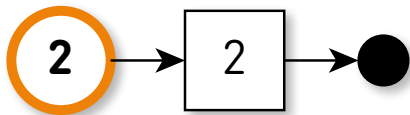
Решето Эратосфена



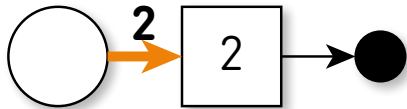
Инициализация



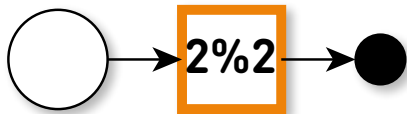
Просеиваем 2



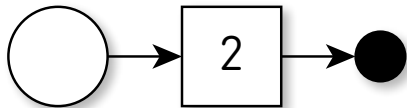
Просеиваем 2



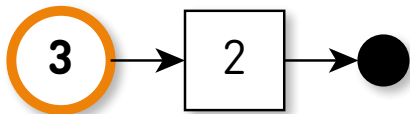
Просеиваем 2



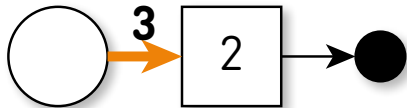
Отсеяли



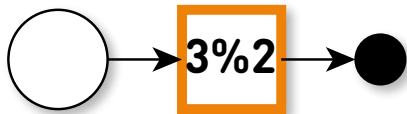
Просеиваем 3



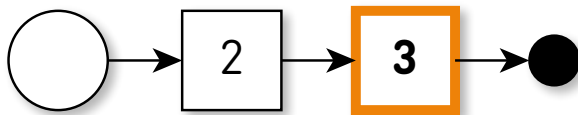
Просеиваем 3



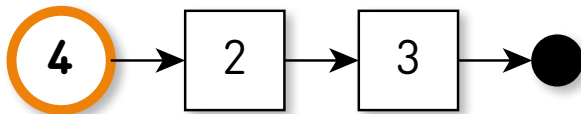
Просеиваем 3



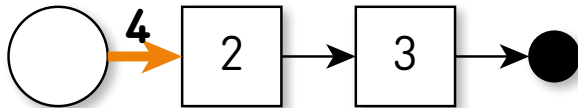
Просеяли



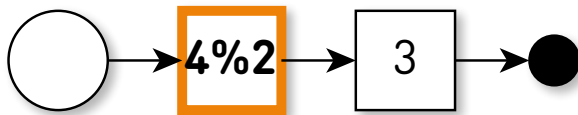
Просеиваем 4



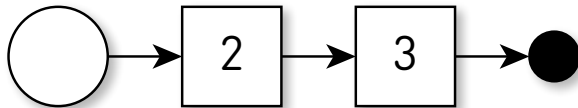
Просеиваем 4



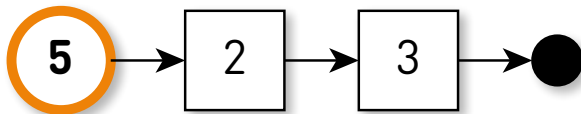
Просеиваем 4



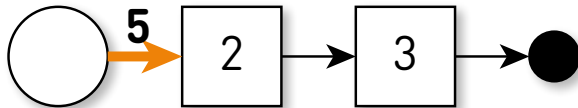
Отсеяли



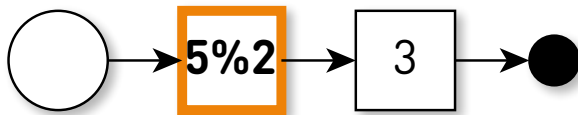
Просеиваем 5



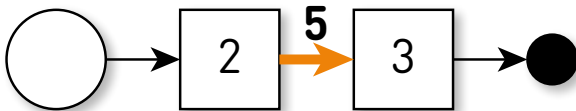
Просеиваем 5



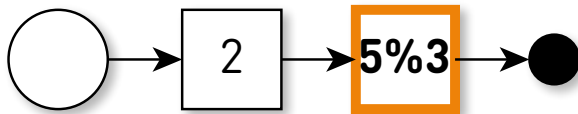
Просеиваем 5



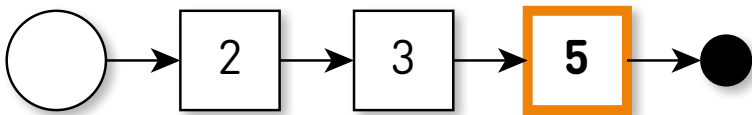
Просеиваем 5



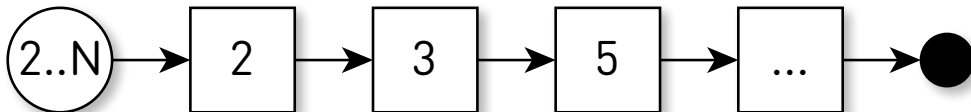
Просеиваем 5



Просеяли



И так далее



Состояние

```
1 class Divisor extends Actor<Divisor.Message> {  
2     final long prime;  
3     Divisor next = null;
```



Состояние

```
1 class Divisor extends Actor<Divisor.Message> {  
2     final long prime;  
3     Divisor next = null;
```



Состояние

```
1 class Divisor extends Actor<Divisor.Message> {  
2     final long prime;  
3     Divisor next = null;
```



Конструирование

```
1 Divisor(  
2     Supervisor parent,  
3     long prime,  
4     ExecutorService executor) {  
5     super(parent, prime, executor);  
6  
7     this.prime = prime;  
8 }
```



Конструирование

```
1 Divisor(  
2     Supervisor parent,  
3     long prime,  
4     ExecutorService executor) {  
5     super(parent, prime, executor);  
6  
7     this.prime = prime;  
8 }
```



Конструирование

```
1 Divisor(  
2     Supervisor parent,  
3     long prime,  
4     ExecutorService executor) {  
5     super(parent, prime, executor);  
6  
7     this.prime = prime;  
8 }
```



Конструирование

```
1 Divisor(  
2     Supervisor parent,  
3     long prime,  
4     ExecutorService executor) {  
5     super(parent, prime, executor);  
6  
7     this.prime = prime;  
8 }
```



Supervisor

```
1 void onChildError(Actor<?> child, Exception e);  
2  
3 void onChildStop(Actor<?> child);
```



Supervisor

```
1 void onError(Actor<?> child, Exception e);  
2  
3 void onStop(Actor<?> child);
```



Supervisor

```
1 void onError(Actor<?> child, Exception e);  
2  
3 void onStop(Actor<?> child);
```



Death Pact

```
1 @Override
2 public void onChildStop(Actor<?> child) {
3     assert child == next;
4
5     stop();
6 }
```



Protocol

```
1 static final class Check extends Message {  
2     final long candidate;  
3  
4     Check(long candidate) { ... }  
5 }
```

```
1 static final class Collect extends Message {  
2     final Collection<Long> to;  
3  
4     Collect(Collection<Long> to) { ... }  
5 }
```



Matching

```
1 @Override
2 protected void receive(Message message) {
3     switch (message) {
4         case Check check -> doCheck(check.candidate);
5         case Collect collect -> doCollect(collect.to);
6         default -> throw new IllegalArgumentException();
7     }
8 }
```



```
1 void doCheck(long candidate) {
2     if (candidate % prime == 0) {
3         // Not prime
4         return;
5     }
6
7     if (next == null) {
8         // New prime discovered
9         next = new Divisor(this, candidate, dispatcher);
10    } else {
11        // Pass through
12        next.check(candidate);
13    }
14 }
```



```
1 void doCheck(long candidate) {
2     if (candidate % prime == 0) {
3         // Not prime
4         return;
5     }
6
7     if (next == null) {
8         // New prime discovered
9         next = new Divisor(this, candidate, dispatcher);
10    } else {
11        // Pass through
12        next.check(candidate);
13    }
14 }
```



```
1 void doCheck(long candidate) {  
2     if (candidate % prime == 0) {  
3         // Not prime  
4         return;  
5     }  
6  
7     if (next == null) {  
8         // New prime discovered  
9         next = new Divisor(this, candidate, dispatcher);  
10    } else {  
11        // Pass through  
12        next.check(candidate);  
13    }  
14 }
```



```
1 void doCheck(long candidate) {
2     if (candidate % prime == 0) {
3         // Not prime
4         return;
5     }
6
7     if (next == null) {
8         // New prime discovered
9         next = new Divisor(this, candidate, dispatcher);
10    } else {
11        // Pass through
12        next.check(candidate);
13    }
14 }
```



```
1 void doCollect(Collection<Long> to) {  
2     to.add(prime);  
3     if (next == null) {  
4         stop();  
5     } else {  
6         next.collect(to);  
7     }  
8 }
```




```
1 void doCollect(Collection<Long> to) {  
2     to.add(prime);  
3     if (next == null) {  
4         stop();  
5     } else {  
6         next.collect(to);  
7     }  
8 }
```



```
1 void doCollect(Collection<Long> to) {  
2     to.add(prime);  
3     if (next == null) {  
4         stop();  
5     } else {  
6         next.collect(to);  
7     }  
8 }
```



```
1 void doCollect(Collection<Long> to) {  
2     to.add(prime);  
3     if (next == null) {  
4         stop();  
5     } else {  
6         next.collect(to);  
7     }  
8 }
```



```
1 var dispatcher = new Dispatcher("eratosthenes");
2 var stopped = new CountDownLatch(1);
3 var supervisor = new Supervisor() {
4     @Override
5     public void onChildError(Actor<?> child, Exception e) {
6         onChildStop(child);
7     }
8
9     @Override
10    public void onChildStop(Actor<?> child) {
11        stopped.countDown();
12    }
13 };
14 var sieve = new Divisor(supervisor, 2, dispatcher);
```



```
1 var dispatcher = new Dispatcher("eratosthenes");
2 var stopped = new CountDownLatch(1);
3 var supervisor = new Supervisor() {
4     @Override
5     public void onChildError(Actor<?> child, Exception e) {
6         onChildStop(child);
7     }
8
9     @Override
10    public void onChildStop(Actor<?> child) {
11        stopped.countDown();
12    }
13 };
14 var sieve = new Divisor(supervisor, 2, dispatcher);
```



```
1 var dispatcher = new Dispatcher("eratosthenes");
2 var stopped = new CountDownLatch(1);
3 var supervisor = new Supervisor() {
4     @Override
5     public void onChildError(Actor<?> child, Exception e) {
6         onChildStop(child);
7     }
8
9     @Override
10    public void onChildStop(Actor<?> child) {
11        stopped.countDown();
12    }
13 };
14 var sieve = new Divisor(supervisor, 2, dispatcher);
```



```
1 var dispatcher = new Dispatcher("eratosthenes");
2 var stopped = new CountDownLatch(1);
3 var supervisor = new Supervisor() {
4     @Override
5     public void onChildError(Actor<?> child, Exception e) {
6         onChildStop(child);
7     }
8
9     @Override
10    public void onChildStop(Actor<?> child) {
11        stopped.countDown();
12    }
13 };
14 var sieve = new Divisor(supervisor, 2, dispatcher);
```



Вычисление

```
1 for (long i = 3; i <= 1000; i++) {  
2     sieve.enqueue(new Check(i));  
3 }  
4  
5 Deque<Long> primes = new LinkedBlockingDeque<>();  
6 sieve.enqueue(new Collect(primes));  
7  
8 stopped.await();  
9  
10 assertEquals(997, primes.getLast().longValue());
```



Вычисление

```
1 for (long i = 3; i <= 1000; i++) {  
2     sieve.enqueue(new Check(i));  
3 }  
4  
5 Deque<Long> primes = new LinkedBlockingDeque<>();  
6 sieve.enqueue(new Collect(primes));  
7  
8 stopped.await();  
9  
10 assertEquals(997, primes.getLast().longValue());
```



Вычисление

```
1 for (long i = 3; i <= 1000; i++) {  
2     sieve.enqueue(new Check(i));  
3 }  
4  
5 Deque<Long> primes = new LinkedBlockingDeque<>();  
6 sieve.enqueue(new Collect(primes));  
7  
8 stopped.await();  
9  
10 assertEquals(997, primes.getLast().longValue());
```



Вычисление

```
1 for (long i = 3; i <= 1000; i++) {  
2     sieve.enqueue(new Check(i));  
3 }  
4  
5 Deque<Long> primes = new LinkedBlockingDeque<>();  
6 sieve.enqueue(new Collect(primes));  
7  
8 stopped.await();  
9  
10 assertEquals(997, primes.getLast().longValue());
```



Вычисление

```
1 for (long i = 3; i <= 1000; i++) {  
2     sieve.enqueue(new Check(i));  
3 }  
4  
5 Deque<Long> primes = new LinkedBlockingDeque<>();  
6 sieve.enqueue(new Collect(primes));  
7  
8 stopped.await();  
9  
10 assertEquals(997, primes.getLast().longValue());
```



one-actor

400 LOC без зависимостей.

[all classes]

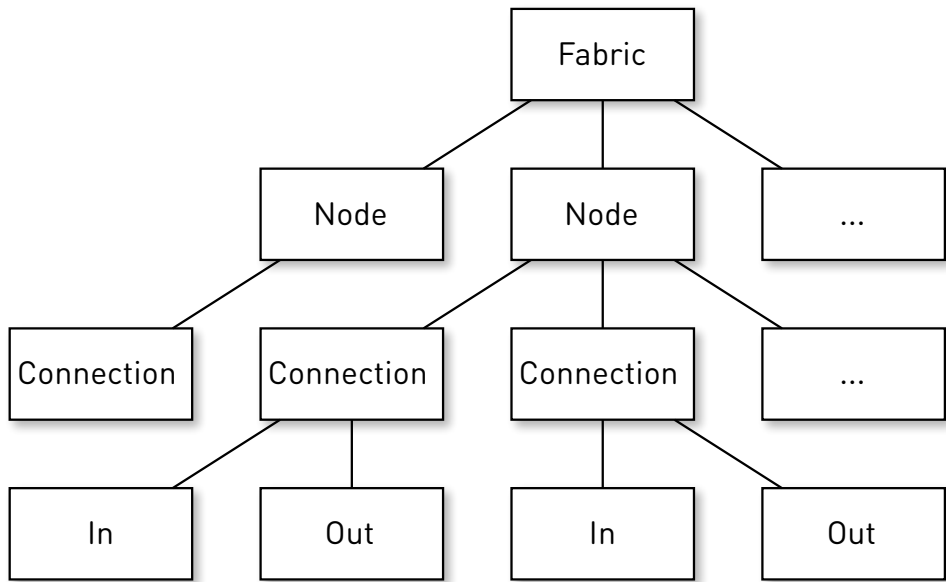
Overall Coverage Summary

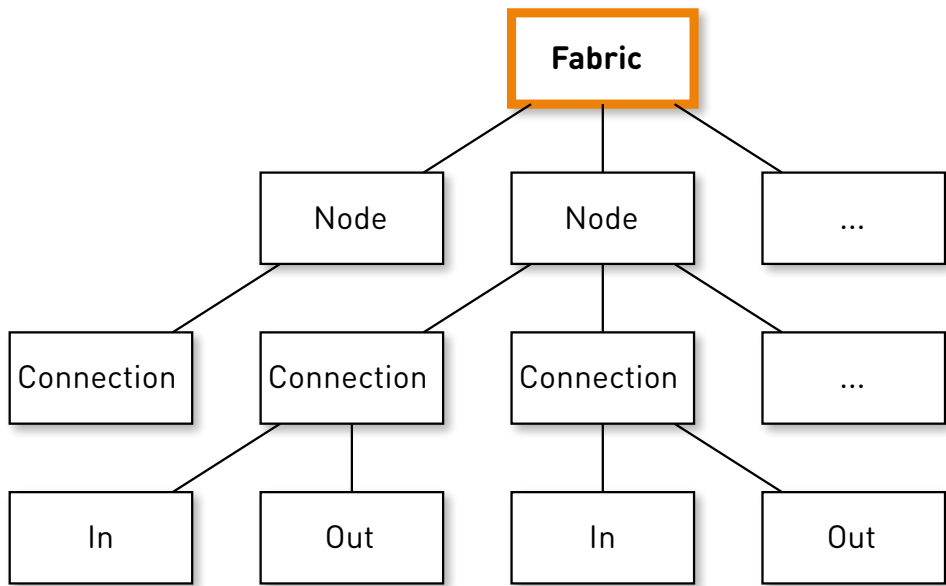
Package	Class, %	Method, %	Line, %
all classes	100% (7/ 7)	100% (31/ 31)	100% (133/ 133)

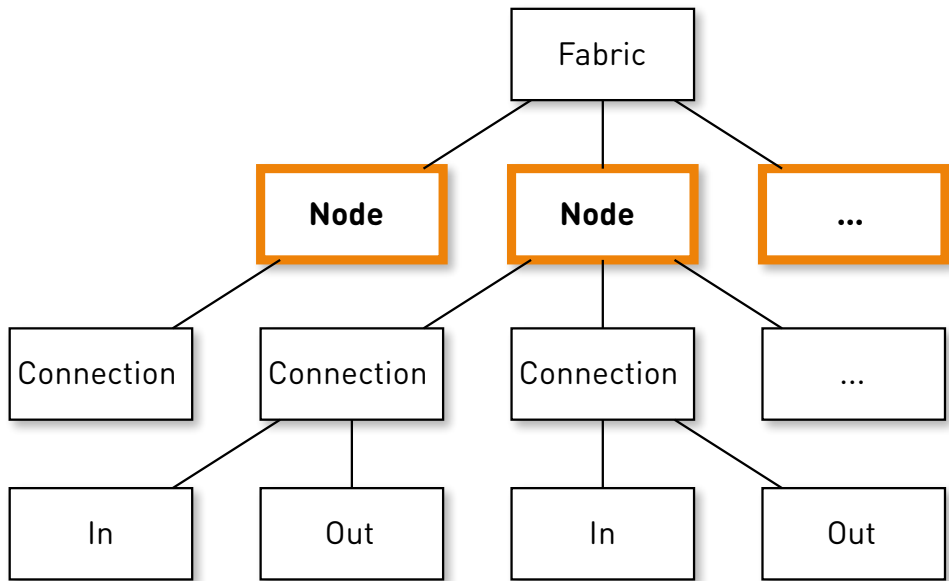
Coverage Breakdown

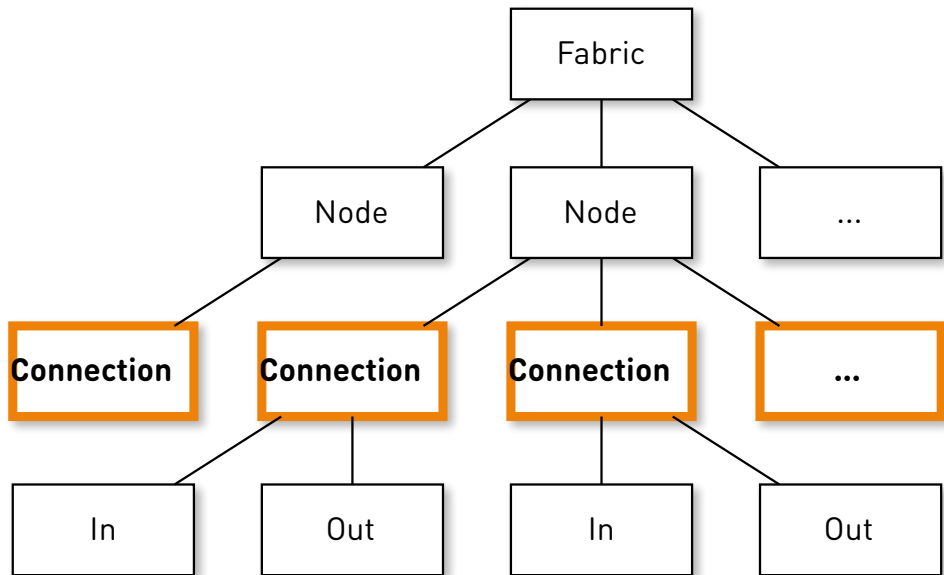
Package ▲	Class, %	Method, %	Line, %
one.actor	100% (7/ 7)	100% (31/ 31)	100% (133/ 133)

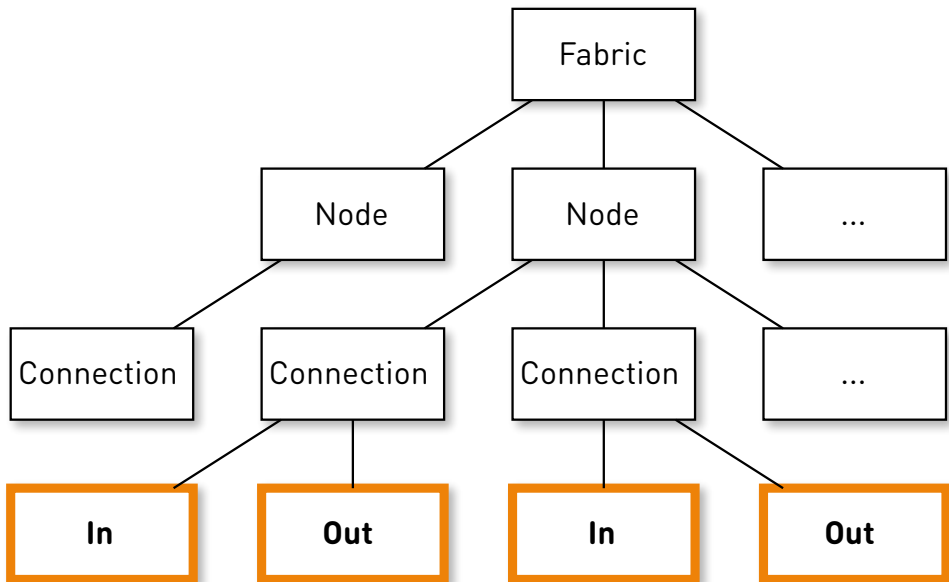












```
1 out.write(...,  
2     new CompletionHandler<Integer, ByteBuffer>() {  
3         @Override  
4         public void completed(...) {  
5             ...  
6             enqueue(Ready.INSTANCE);  
7         }  
8  
9         @Override  
10        public void failed(...) {  
11            ...  
12            enqueue(WriteFailure.INSTANCE);  
13        }  
14    });
```

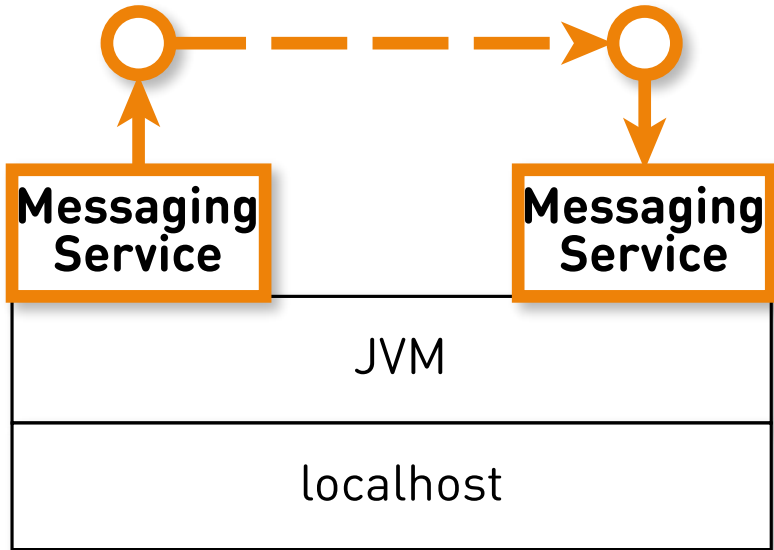


```
1 out.write(...,  
2     new CompletionHandler<Integer, ByteBuffer>() {  
3         @Override  
4         public void completed(...) {  
5             ...  
6             enqueue(Ready.INSTANCE);  
7         }  
8  
9         @Override  
10        public void failed(...) {  
11            ...  
12            enqueue(WriteFailure.INSTANCE);  
13        }  
14    });
```



```
1 out.write(...,  
2     new CompletionHandler<Integer, ByteBuffer>() {  
3         @Override  
4         public void completed(...) {  
5             ...  
6             enqueue(Ready.INSTANCE);  
7         }  
8  
9         @Override  
10        public void failed(...) {  
11            ...  
12            enqueue(WriteFailure.INSTANCE);  
13        }  
14    });
```



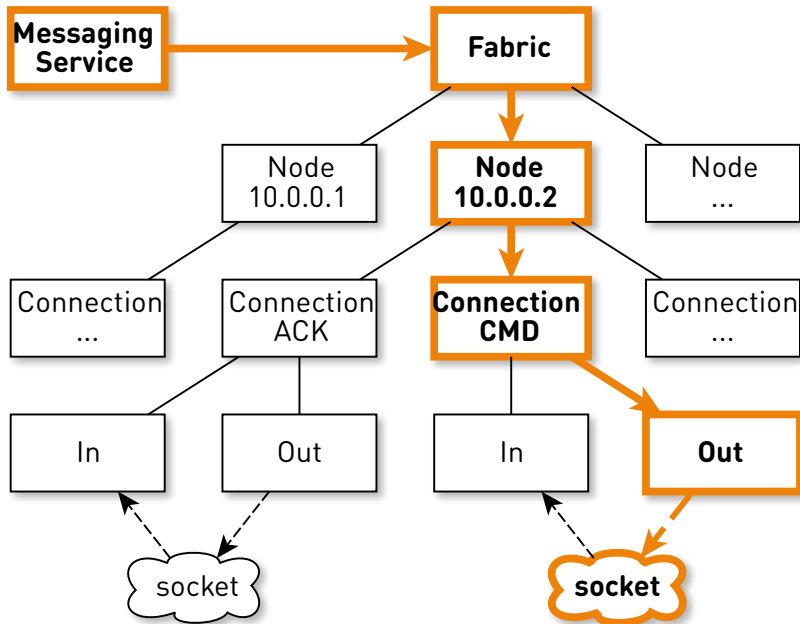


SendingBenchmark

- **10 М** сообщений по **1 КБ** в одну сторону
- Отправка **пачками по 100 К** сообщений
- **Не больше двух пачек** в полёте

7 μ s
на сообщение

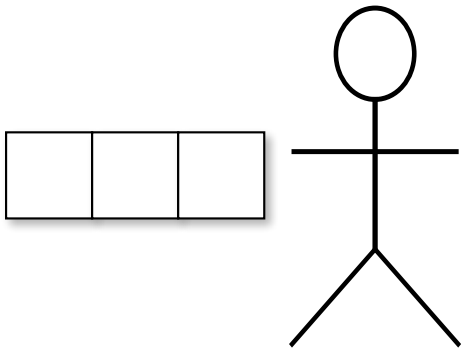




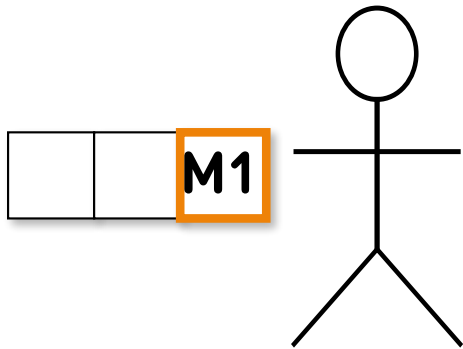
Асинхронная запись

```
1 /**
2  * @throws WritePendingException
3  *      If a write operation is already in progress on
4  *      this channel
5  */
6 public abstract <A> void write(
7     ByteBuffer src,
8     long timeout,
9     TimeUnit unit,
10    A attachment,
11    CompletionHandler<Integer,? super A> handler);
```

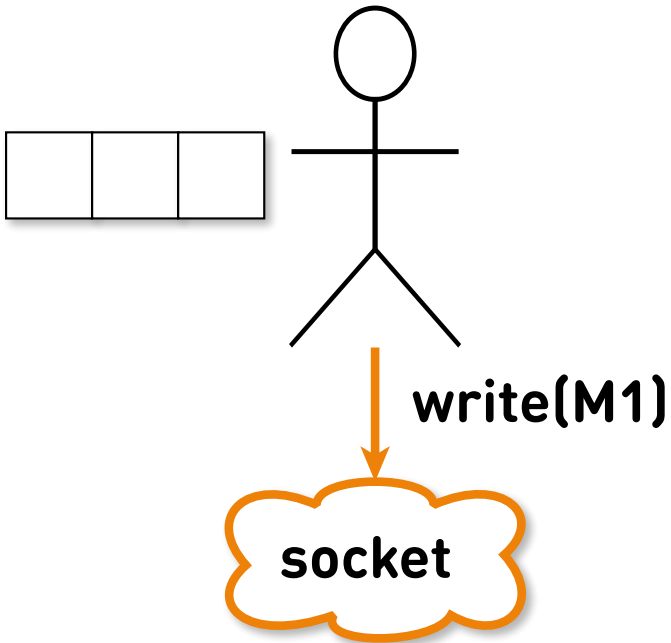


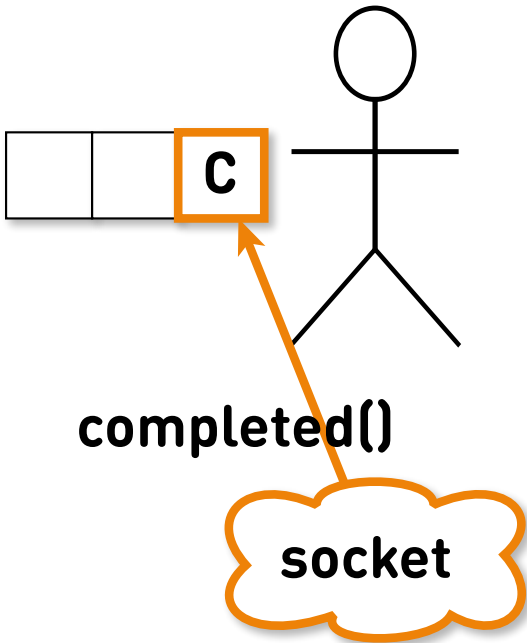


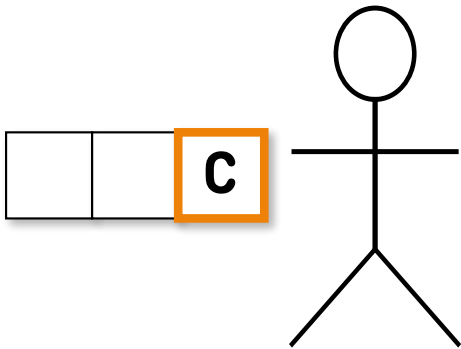
socket



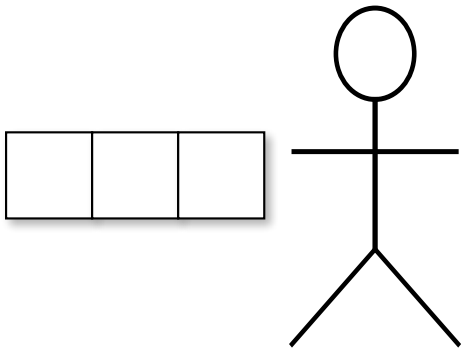
socket



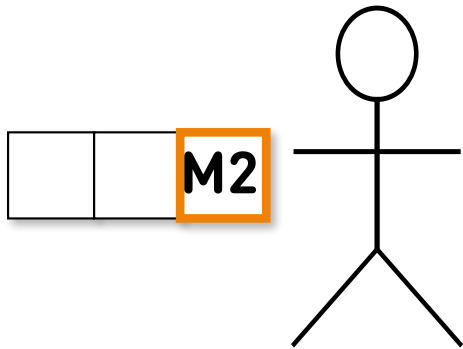




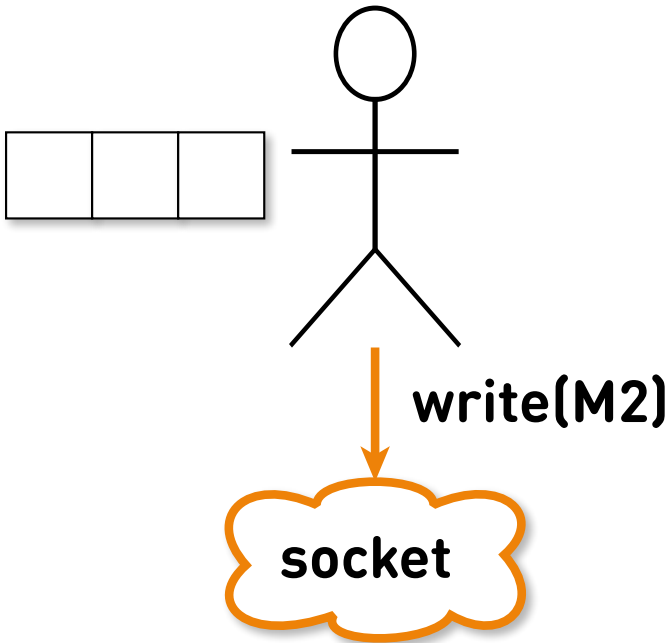
socket

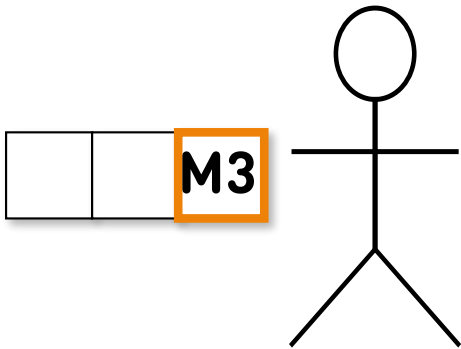


socket

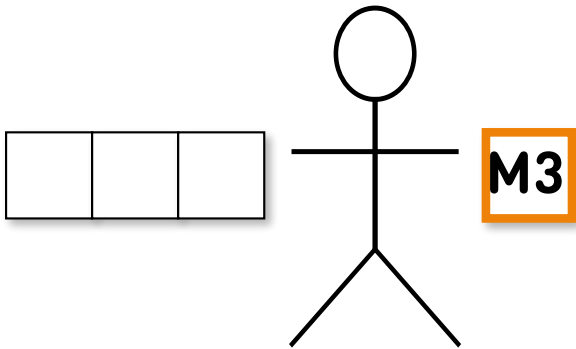


socket

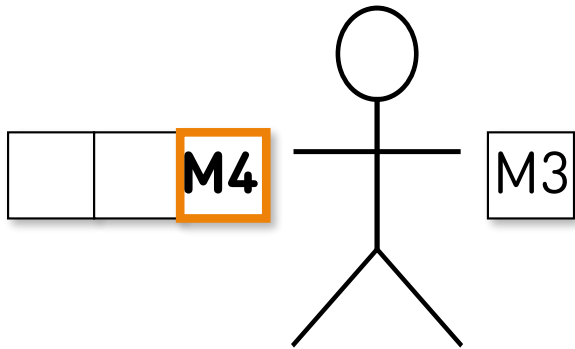




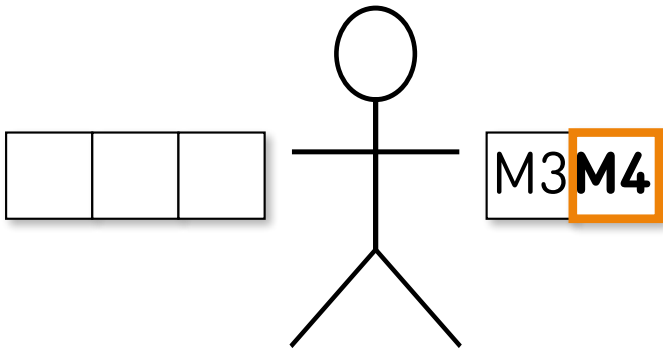
socket



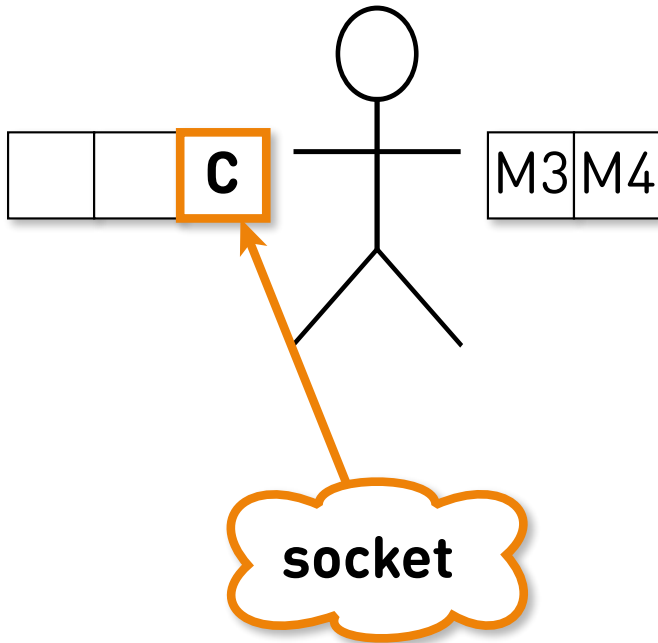
socket

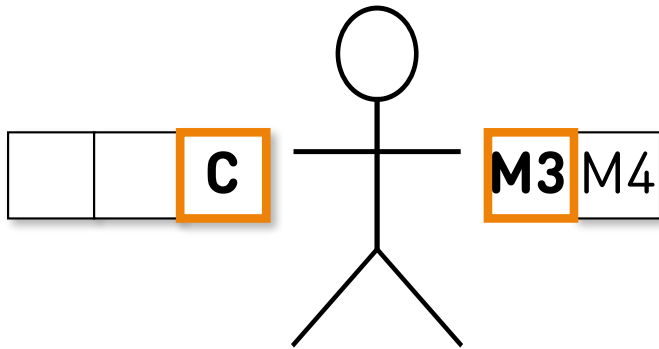


socket

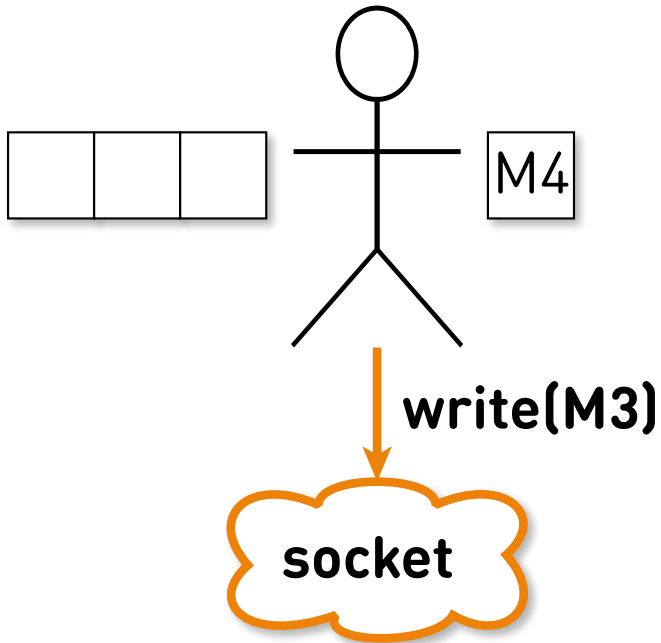


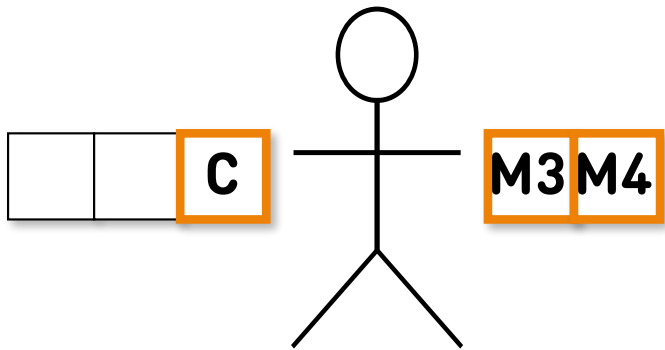
socket





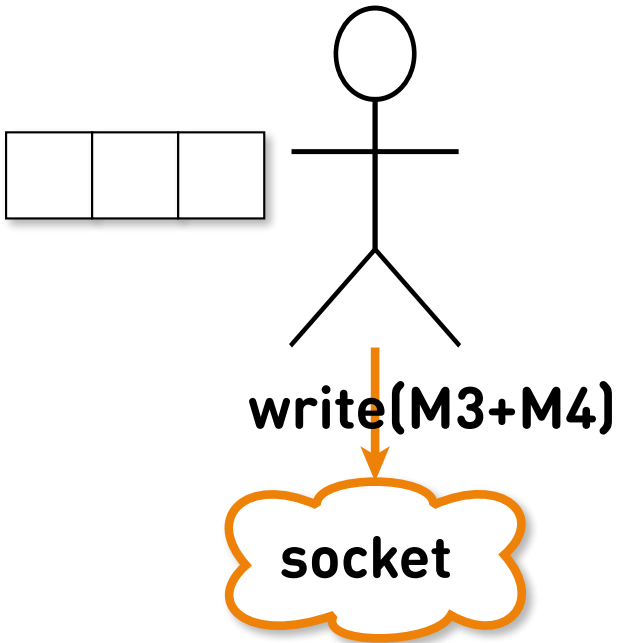
socket





socket





SendingBenchmark

- **7 $\mu\text{s}/\text{msg}$** , если по одному
- **1 $\mu\text{s}/\text{msg}$** , если батчами по **128 КБ**
- Читаем **тоже батчами** через offheap в In



PingPongBenchmark

- **1 М** сообщений
- Каждое **туда и обратно**
- **Последовательно**

54 μ s
RTT



Let's go deeper

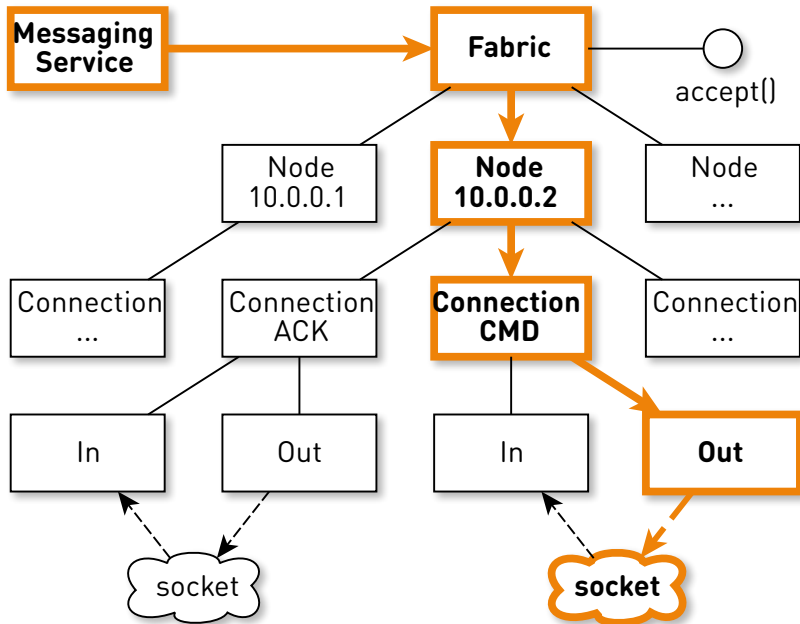
```
1 void enqueue(M message) {  
2     messages.offer(message);  
3     tryScheduleToExecute();  
4 }
```

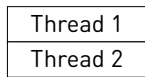
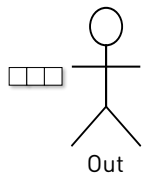
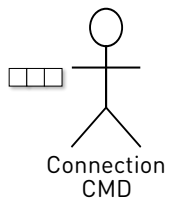
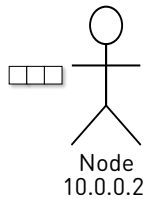


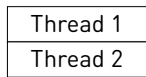
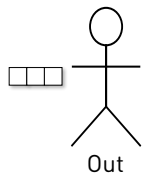
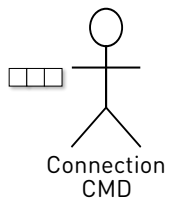
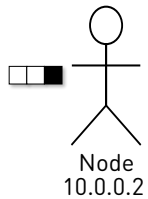
Even deeper

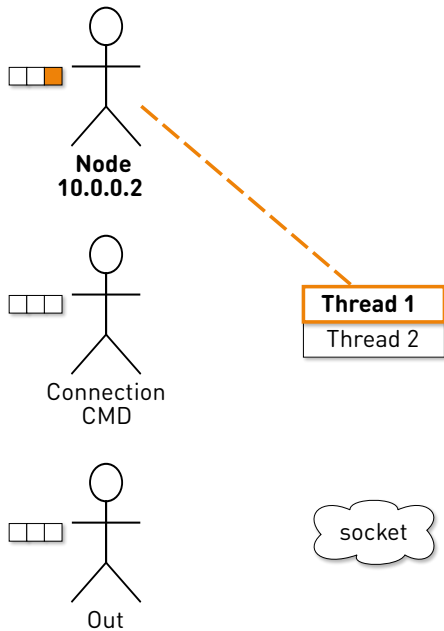
```
1 void tryScheduleToExecute() {  
2     if (on.compareAndSet(false, true)) {  
3         try {  
4             dispatcher.execute(this);  
5         } catch (Exception e) {  
6             ...  
7         }  
8     }  
9 }
```

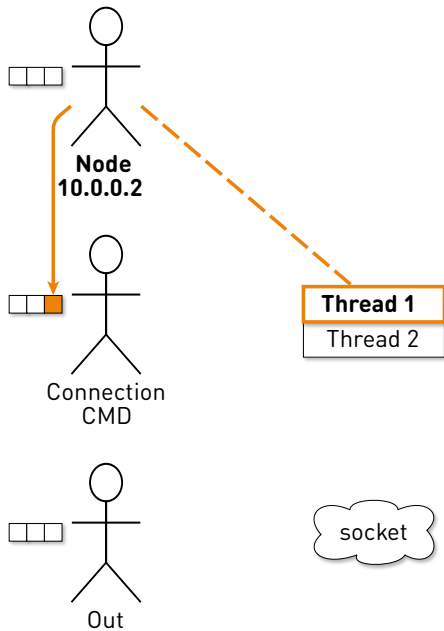


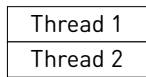
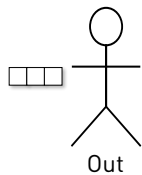
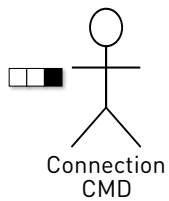
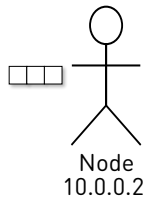


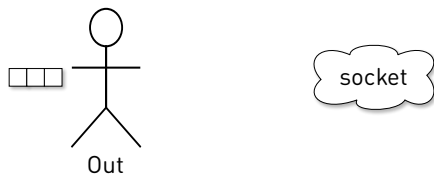
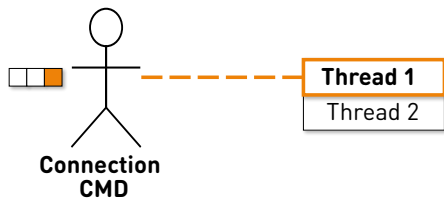
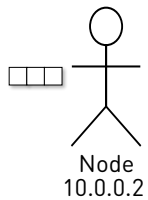


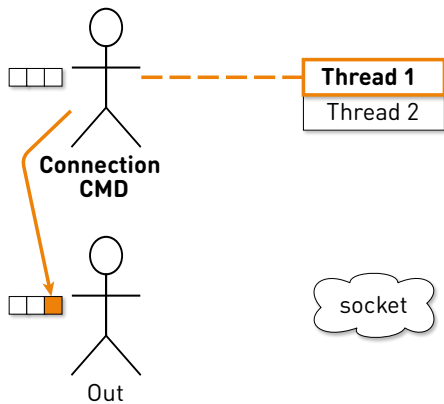
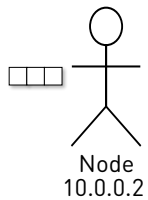


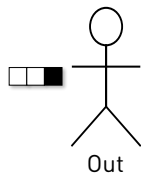
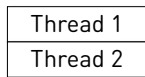
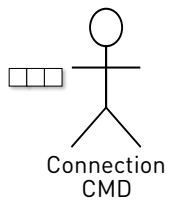
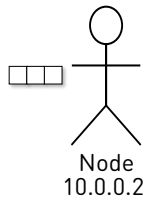


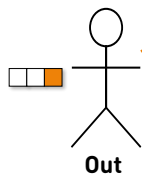
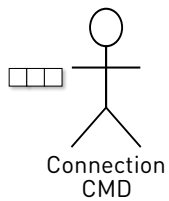
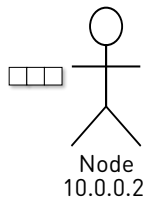


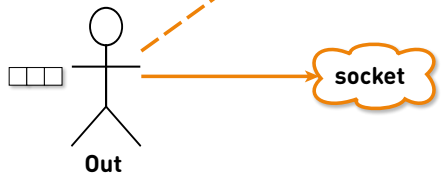
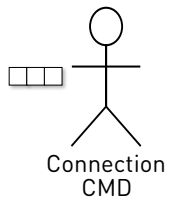
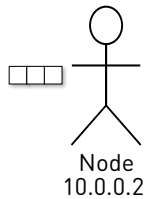


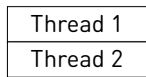
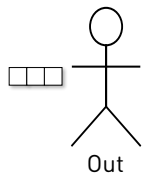
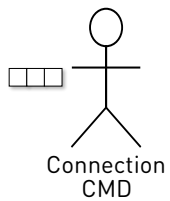
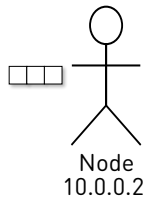


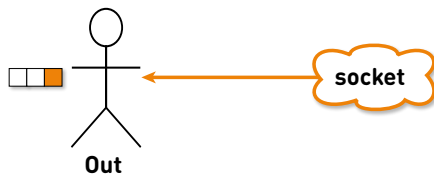
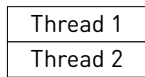
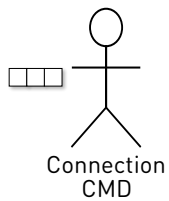
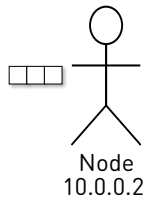


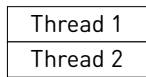
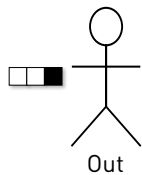
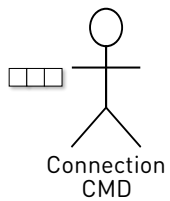
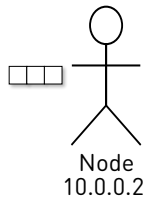












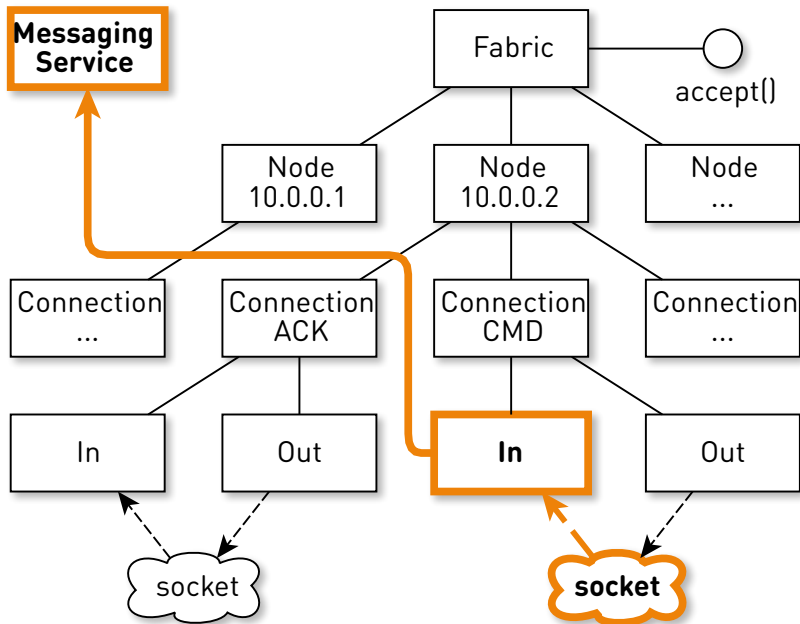
Call stack VMecto context switch?

```
1 @Override
2 public void execute(Runnable command) {
3     Thread current = Thread.currentThread();
4     if (current instanceof DispatcherThread) {
5         // Use the call stack instead of context switch
6         command.run();
7     } else {
8         super.execute(command);
9     }
```



```
1 10:28:17,508 org.apache.cassandra.service.CassandraDaemon$2 Exception in thread Thread[Fabric:17007:51,5,main]
2 java.lang.StackOverflowError
3     at sun.nio.ch.AsynchronousSocketChannelImpl.read(AsynchronousSocketChannelImpl.java:295)
4     at org.apache.cassandra.net.async.connection.In$1.completed(In.java:166)
5     at org.apache.cassandra.net.async.connection.In$1.completed(In.java:154)
6     at sun.nio.ch.Invoker.invokeUnchecked(Invoker.java:126)
7     at sun.nio.ch.Invoker.invokeDirect(Invoker.java:157)
8     at sun.nio.ch.UnixAsynchronousSocketChannelImpl.implRead(UnixAsynchronousSocketChannelImpl.java:553)
9     at sun.nio.ch.AsynchronousSocketChannelImpl.read(AsynchronousSocketChannelImpl.java:276)
10    at sun.nio.ch.AsynchronousSocketChannelImpl.read(AsynchronousSocketChannelImpl.java:297)
11    at org.apache.cassandra.net.async.connection.In$1.completed(In.java:166)
12    at org.apache.cassandra.net.async.connection.In$1.completed(In.java:154)
13    at sun.nio.ch.Invoker.invokeUnchecked(Invoker.java:126)
14    at sun.nio.ch.Invoker.invokeDirect(Invoker.java:157)
15    at sun.nio.ch.UnixAsynchronousSocketChannelImpl.implRead(UnixAsynchronousSocketChannelImpl.java:553)
16    ...
```





sun.nio.ch.Invoker

```
1 /**
2  * Returns true if the current thread is in
3  * the given channel's thread pool and
4  * we haven't exceeded the maximum number of
5  * handler frames on the stack.
6  */
7 static boolean mayInvokeDirect(
8     GroupAndInvokeCount myGroupAndInvokeCount,
9     AsynchronousChannelGroupImpl group) {
```



Ограничиваем call stack

```
1 @Override
2 public void execute(Runnable command) {
3     Thread current = Thread.currentThread();
4     if (current instanceof DepthCountingThread) {
5         DepthCountingThread thread = (DepthCountingThread) current;
6         int depth = thread.depth;
7         if (depth == throughput) {
8             super.execute(command);
9         } else {
10            thread.depth = depth + 1;
11            try {
12                // Use the call stack instead of context switch
13                command.run();
14            } finally {
15                thread.depth = depth;
16            }
17        }
18    }
```



```
1 class DepthCountingThread extends Thread {  
2     int depth = 0;  
3  
4     DepthCountingThread(  
5         Runnable target,  
6         String name) {  
7         super(target, name);  
8     }  
9 }
```



PingPongBenchmark RTT

- 54 μ s было
- 47 μ s при throughput == 1
- 38 μ s при throughput == 4



Throughput

- **64 КБ send:** $> 20 \text{ Gbps}$ и $> 40 \text{ Kmps}$
- **64 Б send:** $> 1 \text{ Gbps}$ и $> 1 \text{ Mmps}$
- **64 КБ ping-pong:** $> 14 \text{ Gbps}$ и $> 15 \text{ Kmps}$
- **64 Б ping-pong:** $> 80 \text{ Mbps}$ и $> 30 \text{ Kmps}$

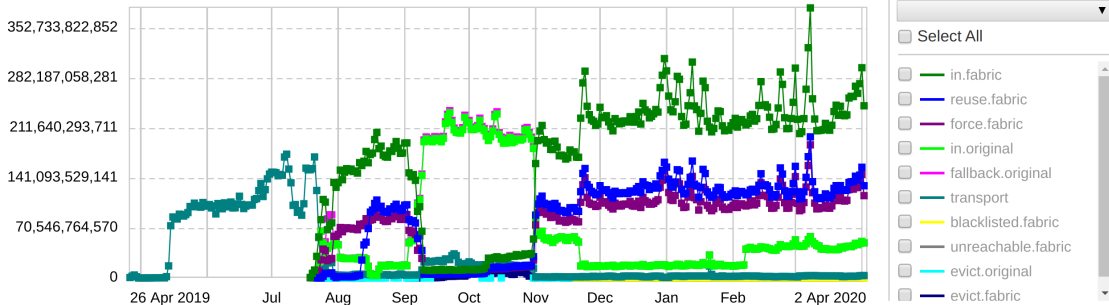


Раскатка в production

Transports

Original Reports / Одноклассники / Logs / Daily / Daily_NewSqlOperations

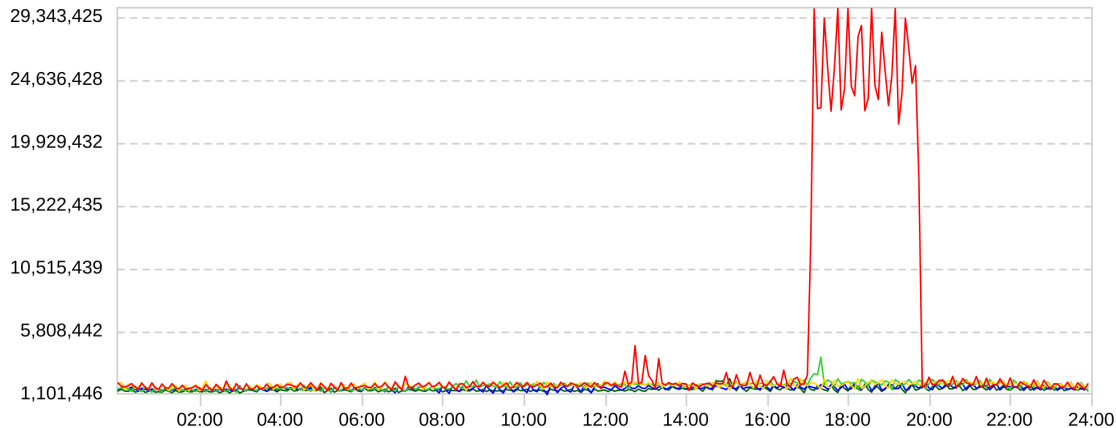
[Link](#) [Copy](#) [Short link](#) [Refresh](#) [Table](#) [Edit](#) [Download](#)



Video DAO Duration Avg.

.Dashboards / Одноклассники / Video / Video Monitoring / DAO / Video DAO Duration Avg

[Link](#) [Copy](#) [Short link](#) [Refresh](#) [Grid](#)



```
1 enum Checkpoint {
2     INIT,          // NEVER USE OR DELETE
3     DEBUG,         // Used only for unit tests
4     ACCEPTED,      // Accepted for sending
5     ENQUEUED,      // Enqueued for sending
6     BATCHED,       // Became part of a batch to be sent
7     SENDING,       // Just before sending
8     SENT,          // Just having sent
9     RECEIVED,      // At the destination
10    DELIVERING,     // Before delivering to MessagingService
11    RESPONDING;     // MessagingService responds to request
```



Трассировка

- Начальный timestamp
- Список (Checkpoint, Δt)
- Разрешение Δt **1 мс, max 16 с**
- Δt как VarUint
- **Всегда сериализована** в `byte[]`
- **1-3 байта** на Checkpoint
- **Несколько десятков байт** на трассу



Нежданчик

- Периодически **200 мс** от SENDING до RECEIVED

Нужен TCP_NODELAY после `ассерт()`

Входящие соединения теперь используются для **исходящих сообщений**.

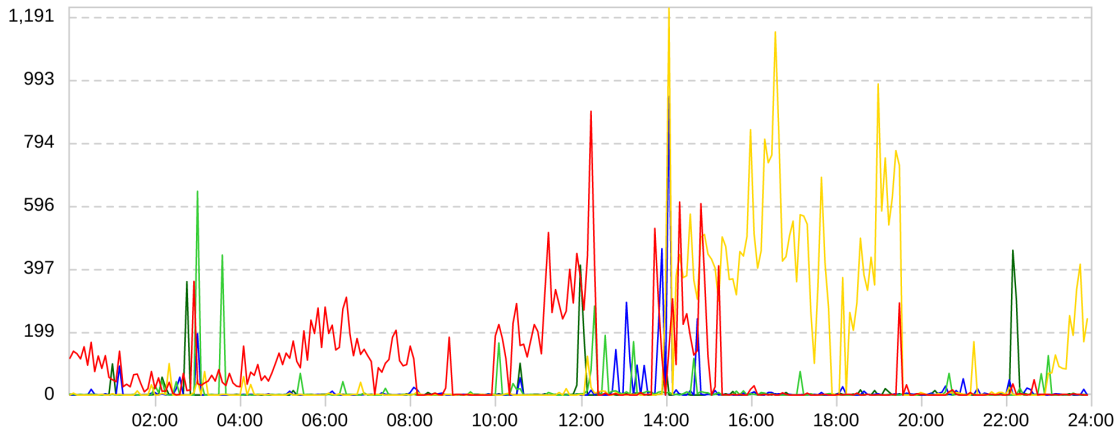


Флапы координаторов каждые 1/5 мин

Query processing errors

[🔗](#) [📎](#) [🔗](#) Short link [↻](#) [☰](#)

[Dashboards](#) / [Одноклассники](#) / [Photos](#) / [NewSQL Photos](#) / [Query processing errors](#)



Включили логи safepoint

- `-XX:+PrintGCApplicationStoppedTime`
- `-XX:+PrintSafepointStatistics`
- `-XX:PrintSafepointStatisticsCount=1`

Achtung!

- Пачки событий `(Bulk)RevokeBias`
- **Десятки пауз по десятку миллисекунд**

Eliminating Synchronization-Related Atomic Operations with Biased Locking and Bulk Rebiasing

Kenneth Russell

Sun Microsystems, Inc.
kenneth.russell@sun.com

David Detlefs *

david.detlefs@alum.mit.edu

Abstract

The Java™ programming language contains built-in synchronization primitives for use in constructing multithreaded programs. Efficient implementation of these synchronization primitives is necessary in order to achieve high performance.

Recent research [9, 12, 10, 3, 7] has focused on the run-time elimination of the atomic operations required to implement object monitor synchronization primitives. This paper describes a novel technique called *store-free biased locking* which eliminates all synchronization-related atomic operations on uncontended object monitors. The technique supports the bulk transfer of object ownership from one thread to another, and the selective disabling of the optimization where unprofitable, using epoch-based bulk rebiasing and revocation. It has been implemented in the production version of the Java HotSpot™ VM and has yielded significant performance improvements on a range of benchmarks and applications. The technique is applicable to any virtual machine-based programming language implementation with mostly block-structured locking primitives.

upon monitor entry, and sometimes upon exit, to ensure correct synchronization. These techniques fall back to using OS mutexes and condition variables when contention occurs.

A related class of optimizations which can be termed *biased locking* [3, 7, 9] rely on the further property that not only are most monitors uncontended, they are only entered and exited by one thread during the lifetime of the monitor. Such monitors may be profitably *biased* toward the owning thread, allowing that thread to enter and exit the monitor without using atomic operations. If another thread attempts to enter a biased monitor, even if no contention occurs, a relatively expensive *bias revocation* operation must be performed. The profitability of such an optimization relies on the benefit of the elimination of atomic operations being higher than the penalty of revocation.

Current refinements of biased locking techniques [12, 10] decrease or eliminate the penalty of bias revocation, but do not optimize certain synchronization patterns which occur in practice, and also impact peak performance of the algorithm.

Multiprocessor systems are increasingly prevalent; so much so that uniprocessors are now the exception rather than the norm.



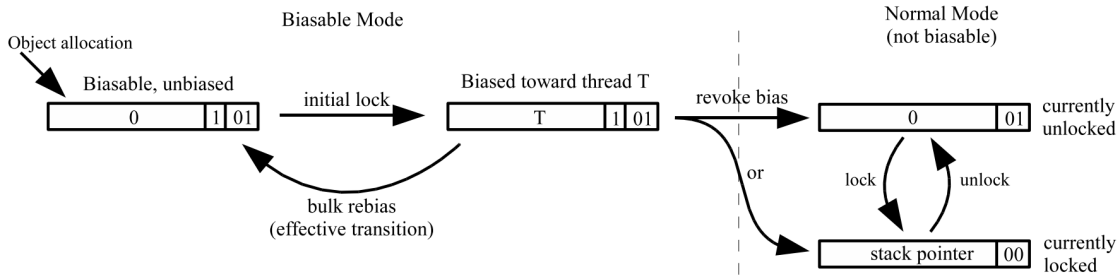


Figure 2. State transitions of an object's mark word under biased locking.

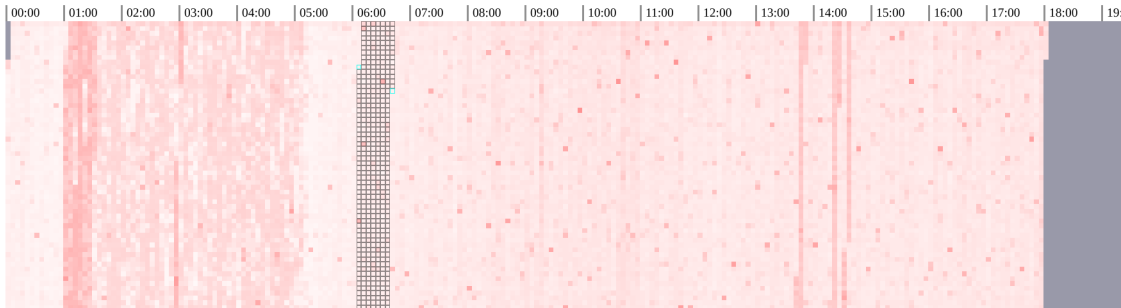
Отпрофилировали³ STW-операции

-e VMThread::execute --cstack dwarf

```
BiasedLocking::revoke_and_rebias(Handle, bool, Thread*)
ObjectSynchronizer::fast_enter(Handle, BasicLock*, bool, Thread*)
SharedRuntime::complete_monitor_locking_C(oopDesc*, BasicLock*, JavaThread*)
javax/management/ObjectName._getKeyPropertyList
javax/management/ObjectName.getKeyProperty
com/sun/jmx/mbeanserver/Repository$ObjectNamePattern.matchKeys
com/sun/jmx/mbeanserver/Repository.addAllMatching
com/sun/jmx/mbeanserver/Repository.query
com/sun/jmx/interceptor/DefaultMBeanServerInterceptor.queryNamesImpl
com/sun/jmx/interceptor/DefaultMBeanServerInterceptor.queryNames
com/sun/jmx/mbeanserver/JmxMBeanServer.queryNames
com/sun/jmx/remote/security/MBeanServerAccessController.queryNames
```

³<https://github.com/jvm-profiling-tools/async-profiler>

Сейчас ещё проще⁴



[Flame Graph](#)

Produced by [async-profiler](#)

all											
SCHED_BA.. S.. SCHED_OTHER											
org..	s..	o..	java/lang/Thread.run:833						org/apache/cassandra/conc..	st..	
jav..	t..	j..	java/util/concurrent/ThreadPoolExecutor\$Worker.run:635			ok/hyd..		su..	sun/ni..	java/lang/Thread.run:833	th..
org..	T..	j..	java/util/concurrent/ThreadPoo..	java/util/concurrent/ThreadPoolExecutor.runWorker:1..		org/a..		s..	sun/ni..	java/util/concurrent/Thre..	Th..
o..		j..	java/util/concurrent/ThreadPoo..	java/uti..	one/actor/Actor.run:82	sun/nio/ch/..		o..		s..	sun/ni..
j..			java/util/concurr..	java/ut..	java/uti..	org/apache/..	org/apach..	sun/nio/ch/..		e..	su..
									s..	java/uti..	org/apache/c..
											G1..

⁴Непрерывное профилирование в облаке с помощью eBPF @ JPoint 2022

-XX:-UseBiasedLocking

- ① JMX-монитор ходит в сервис **раз в минуту**
- ② one-log отправляет статус **каждые 5 мин**
- ③ one-log «трогает» biased JMX ObjectName
- ④ Паузы на **сотни миллисекунд** из-за RevokeBias
- ⑤ Координатор считают **мёртвым**



JEP 374: Deprecate and Disable Biased Locking

<i>Owner</i>	Patricio Chilano Mateo
<i>Type</i>	Feature
<i>Scope</i>	JDK
<i>Status</i>	Closed / Delivered
<i>Release</i>	15
<i>Component</i>	hotspot / runtime
<i>Discussion</i>	hotspot dash runtime dash dev at openjdk dot java dot net
<i>Effort</i>	XS
<i>Duration</i>	XS
<i>Reviewed by</i>	Coleen Phillimore, David Holmes, Mikael Vidstedt
<i>Endorsed by</i>	Mikael Vidstedt
<i>Created</i>	2019/12/03 14:24
<i>Updated</i>	2021/08/28 00:39
<i>Issue</i>	8235256

Summary

Disable biased locking by default, and deprecate all related command-line options.



Частые Full GC на стораджах

```
1 List<ExpiringMessage> batch = queue.extractBatch();
2 ...
3 out.write (...
4     new CompletionHandler<Integer, ByteBuffer>() {
5         @Override
6         public void completed(...) {
7             ...
8             // Tracing
9             fillTraceState (batch);
10            ...
```





JDK / JDK-8202252

(aio) Closed AsynchronousSocketChannel keeps completion handler alive

▼ Details

Type:	 Bug	Status:	RESOLVED
Priority:	 P3	Resolution:	Fixed
Affects Version/s:	7, 8, 10, 11	Fix Version/s:	12
Component/s:	core-libs		
Labels:	additional-information-received azul-interest dcs-psy jdk11u-fix-request jdk11u-fix-yes jdk8u-fix-request jdk8u-fix-yes oracle-bp redhat-interest reproducer-no webbug		
Subcomponent:	java.nio		
Resolved In Build:	b01		
CPU:	x86		
OS:	os_x		



```

1      // invoke handler and set result
2      CompletionHandler<Void,Object> handler = connectHandler;
3 +      connectHandler = null;
4      Object att = connectAttachment;
5      PendingFuture<Void,Object> future = connectFuture;
6      if (handler == null) {
7 @@ -405,6 +406,7 @@
8          this.readBuffer = null;
9          this.readBuffers = null;
10         this.readAttachment = null;
11 +        this.readHandler = null;
12
13         // allow another read to be initiated
14         enableReading();
15 @@ -600,6 +602,7 @@
16         this.writeBuffer = null;
17         this.writeBuffers = null;
18         this.writeAttachment = null;
19 +        this.writeHandler = null;

```



```
1 List<ExpiringMessage> batch = queue.extractBatch();
2 ...
3 out.write (...
4     new CompletionHandler<Integer, ByteBuffer>() {
5         @Override
6         public void completed(...) {
7             ...
8             // Tracing
9             fillTraceState (batch);
10            ...
```



```
1 List<ExpiringMessage> batch = queue.extractBatch();
2 ...
3 AtomicReference batchRef = new AtomicReference<>(batch);
4 out.write (...
5     new CompletionHandler<Integer, ByteBuffer>() {
6         @Override
7         public void completed(...) {
8             ...
9             // Tracing
10            fillTraceState (batchRef.getAndSet(null));
11            ...
```





Andrei Pangin @AndreiPangin · Jul 16, 2019

Found a severe memory leak in production with @incubos. Turns out to be a bug in JDK Async IO. Ironically the bug is marked as P5 - the lowest priority "cosmetic" problem. No fix until JDK 12. bugs.openjdk.java.net/browse/JDK-820...



1



6



23



Vladimir Ivanov @iwan0www · Jul 16, 2019

1. It's perfectly fine to raise bug priority even after the fix arrived.
2. Aleksey @shipilev might be interested in putting it on 8u & 11u backport list anyway ;-)



1



1



8



Aleksey Shipilëv
@shipilev

Replying to @iwan0www @AndreiPangin and @incubos

In backporting queue for 8u and 11u now.

11:55 PM · Jul 16, 2019 · [TweetDeck](#)



■ ● [Aleksey Shipilev](#) added a comment - 2019-07-16 12:13

Users report this as the "severe memory leak in production" here:

<https://twitter.com/AndreiPangin/status/1151174582856011776> -- raising priority.

■ ● [Aleksey Shipilev](#) added a comment - 2019-07-16 13:51

Fix Request (8u, 11u)

Backporting this fix resolves the serious memory leak in aio. Patch applies cleanly to 11u, and requires usual reshufflings in 8u. New test fails without the product patch (has unusual failure mode: timeouts on failure, when tracked reference is not GC-ed) in both 8u and 11u, and passes with it. Additionally, tier1 passes in 11u and 8u.

■ ● [Andrei Pangin](#) added a comment - 2019-07-16 14:32

In our case the symptoms are a bit different than in the bug description, but the same fix works fine.

The problem is caused by the writeHandler holding a reference to a CompletionHandler long after the write operation completes. Since we had ~3000 async channels, and each CompletionHandler retained several MB of data, the application exhausted 10 GB heap rather quickly. Setting writeHandler=null in finishWrite() solves the issue.

■ ● [Andrei Pangin](#) added a comment - 2019-07-21 06:55

Added another test for this bug (AsyncChannelLeak.java), which demonstrates the above scenario.



Соединения не работают и не рвутся

Дропаем пакеты:

```
$ iptables -A INPUT -p tcp --dport 65000 -j DROP  
$ iptables -A INPUT -p tcp --sport 65000 -j DROP
```

Свежие входящие соединения

Вместо старых **потенциально подвисших**.

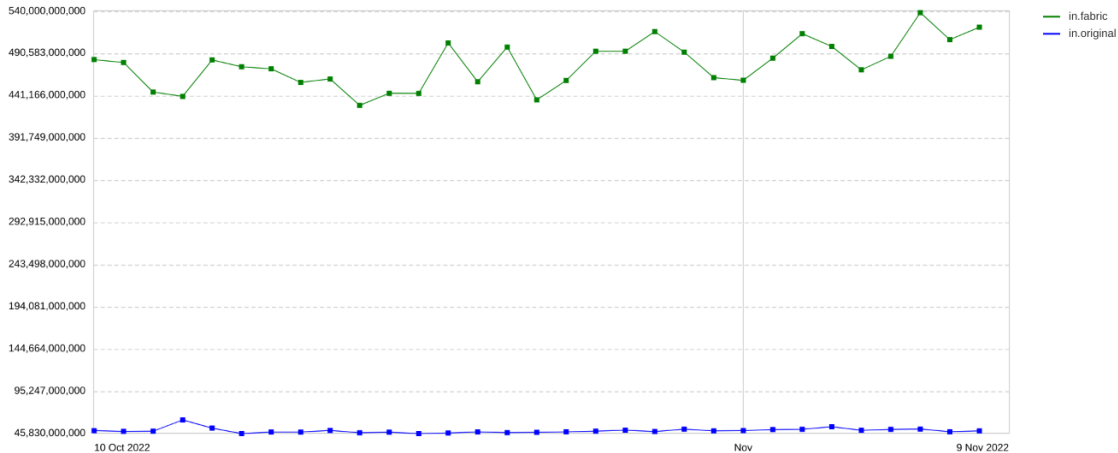
И многое другое

- Gossip только `client` $\rightarrow$ `node` и `node` $\leftrightarrow$ `node`
- Выпилили `EchoMessage`
- В(ы)ключение транспорта без downtime
- ...

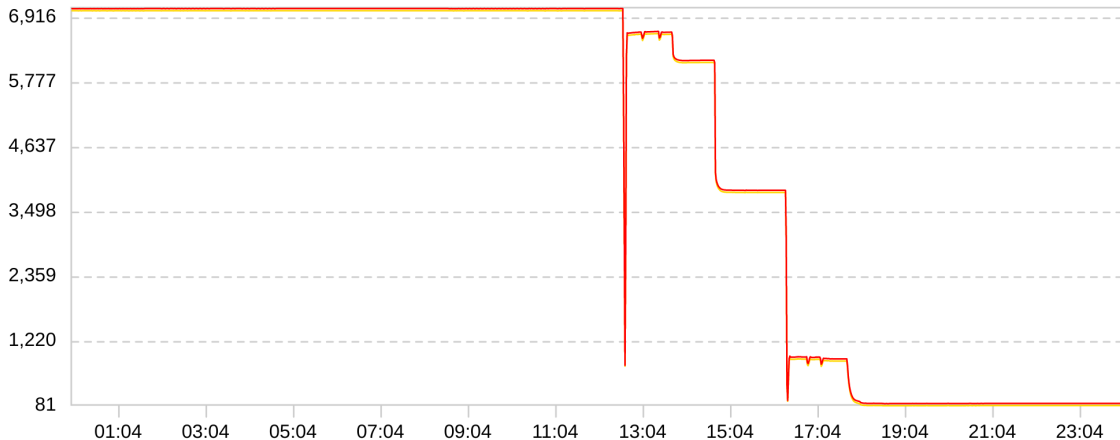


Production

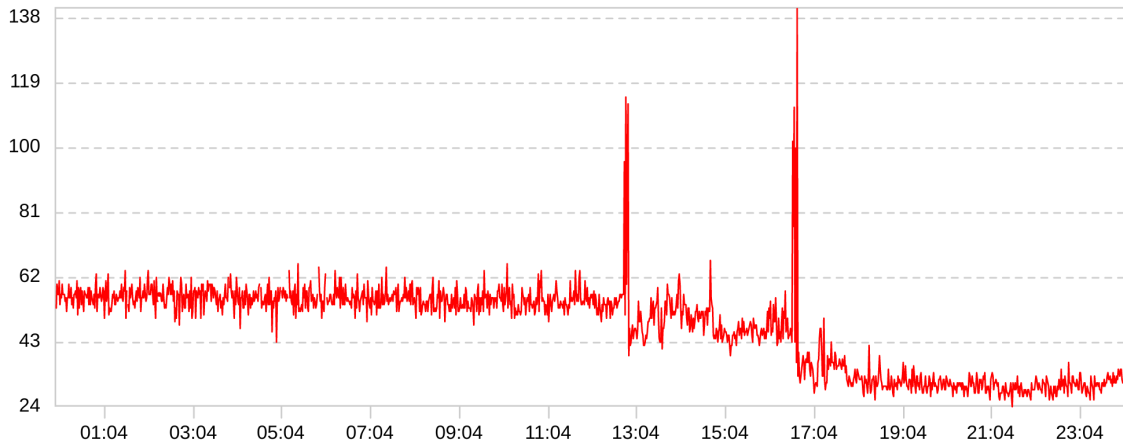
Входящие сообщения оригинального/асинхронного транспорта



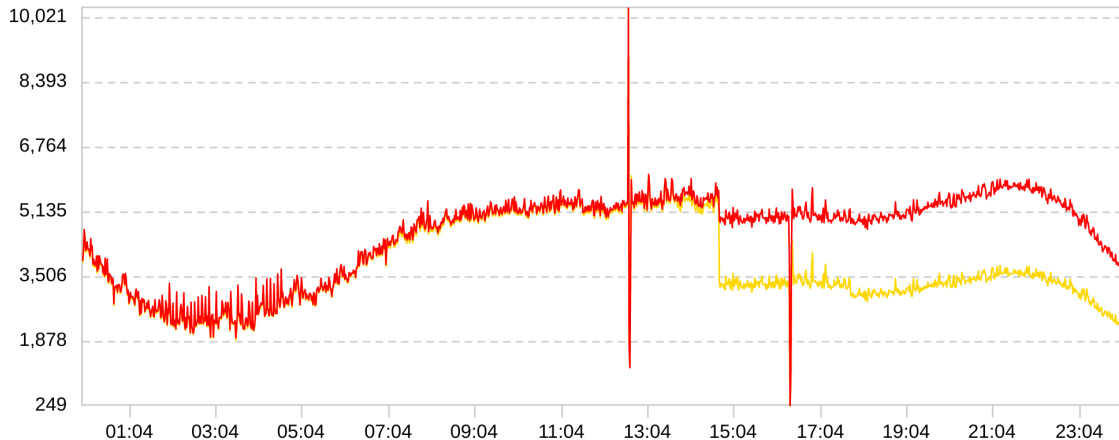
Потоки на ноде



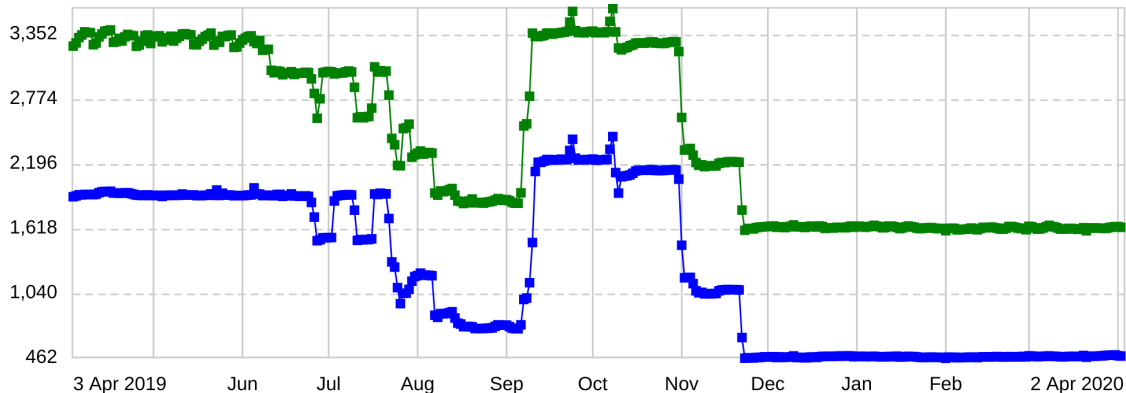
Время GC, мс



Сетевые пакеты



Потоки на клиенте



В заключение

- Цели **достигнуты**
 - Меньше потоков
 - Нет соединений к клиентам
 - Улучшили метрики GC
- Хочется **большей производительности** —
полезайте в **корные компоненты**
- **Знайте и любите** свой стек — грабли везде





Improvements to Internode Messaging

▼ Details

Type:	↑ Improvement	Status:	RESOLVED
Priority:	≡ High	Resolution:	Fixed
Component/s:	Messaging/Internode	Fix Version/s:	4.0-alpha1, 4.0
Labels:	None		
Change Category:	Quality Assurance		
Complexity:	Challenging		

Async IO
+
Actor Model
=


Благодарности

- Александр Христофоров
- Андрей Паньгин
- Олег Анастасьев



Спасибо!

Вадим Цесько
Одноклассники

incubos.org/contacts
oktech.ru



HighLoad⁺⁺
2022

Яндекс



Обратная связь
и комментарии
по докладу ----->

Слайды:
<http://bit.ly/3glLhRa>



HighLoad⁺⁺
2022

Яндекс