Разработка высоконагруженного сервера на Java

Андрей Паньгин
Одноклассники, ведущий инженер

План доклада

- Архитектура ОК
- Работа с сетью
- Удалённые вызовы и сериализация
- Кеширование
- Паузы JVM, оптимизация GC

Факты об ОК

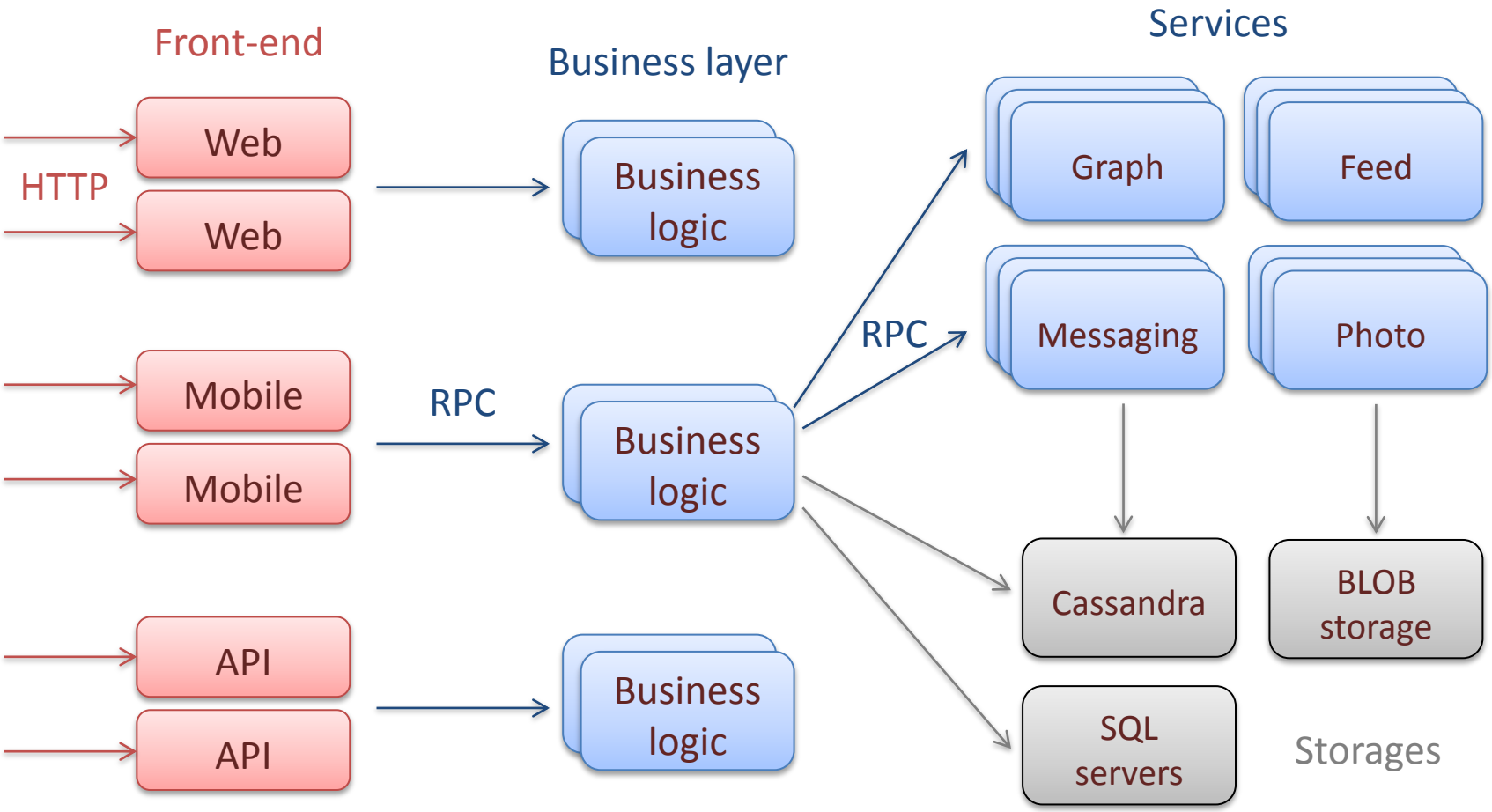
- 8000 серверов
- 48 млн. уникальных пользователей в день
- 8 млн. пользователей онлайн
- 300 000 web запросов в секунду
- 4 ПБ данных (без учёта дублирования)

Экстремальные нагрузки

- На что способен один сервер
  - 50,000 вызовов/с в сек (push delivery)
  - 40 Гбит/с (video download)
  - 100,000 пользователей онлайн (xmpp)
  - 384 ГБ в памяти (cache)

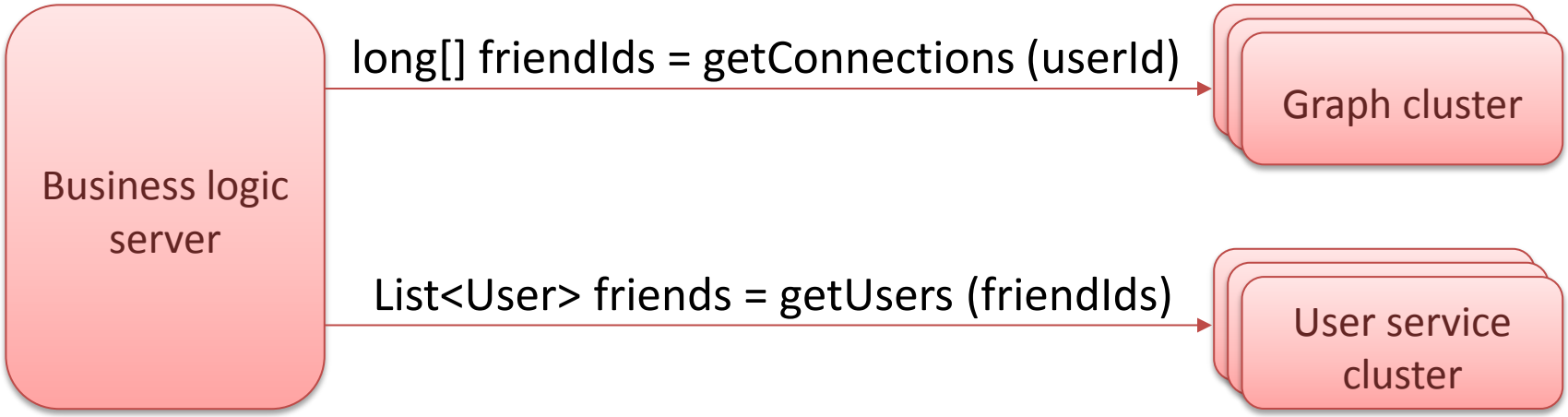
- Это всё Java!

Архитектура ОК



Внутренняя коммуникация

- Удалённый вызов методов (RPC)



План доклада

- Архитектура ОК
- Работа с сетью
- Удалённые вызовы и сериализация
- Кеширование
- Паузы JVM, оптимизация GC

java.net.Socket I/O

- Поток на каждое соединение

```
InputStream in = socket.getInputStream();
OutputStream out = socket.getOutputStream();

while (true) {
  int bytesRead = in.read(...);  // blocking call
  if (bytesRead > 0) {
    byte[] response = processRequest(...);
    out.write(response);         // blocking call
  }
}
```

Проблемы Socket I/O

- 10 тыс. соединений = 10 тыс. потоков
- Finalizers => утечка памяти
- byte[] массивы, копирование

NIO

- Selector + неблокирующие read/write

```
while (true) {
  if (selector.select() > 0) {  // blocking call
    for (SelectionKey key : selector.selectedKeys()) {
      if (key.isReadable()) {
        doRead(key);
      } else if (key.isWritable()) {
        doWrite(key);
      }
    }
    selector.selectedKeys().clear();
  }
}
```

NIO frameworks

- Apache MINA
  - http://mina.apache.org
- Netty
  - http://netty.io

- Основаны на NIO
- Event-driven модель

Проблемы NIO

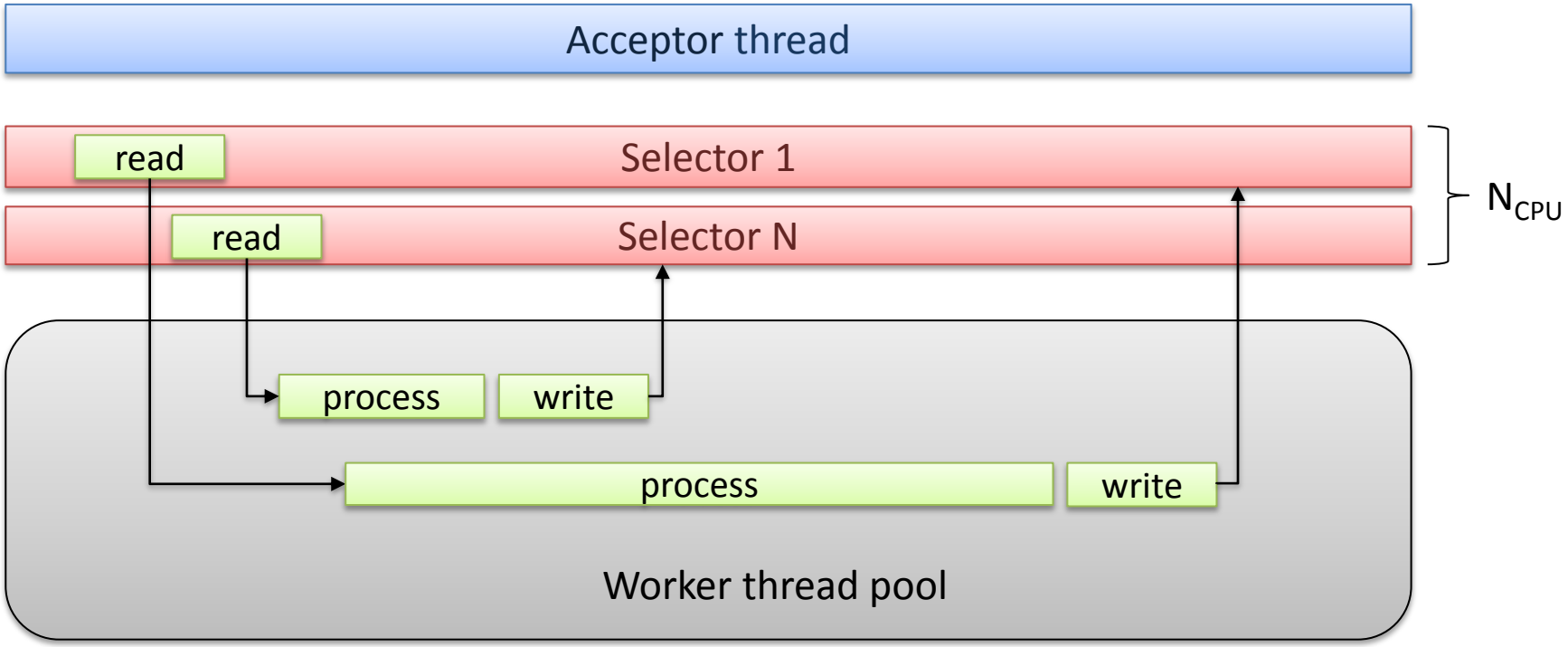
- Selector глючный, не потокобезопасный
- Нельзя делать select() на blocking сокетах
- Не работает setSoTimeout()
- Ограниченная поддержка SSL/TLS

- Не поддерживаются все возможности ОС
  - TCP опции: TCP_DEFER_ACCEPT, TCP_CORK
  - Флаги send/recv: MSG_MORE, MSG_PEEK

Решение

- JNI библиотека
  - Tomcat Native (APR connector)
  - one-nio
- Управление сокетами вручную
- Поддержка OpenSSL

Архитектура сервера



Пример сервера

```
public class MyHttpServer extends HttpServer {

  public MyHttpServer() throws IOException {
    super(new ConnectionString("https://0.0.0.0"));
  }

  @Path("/hello")
  public Response hello() {
    return Response.ok("Hello world");
  }

  public static void main(String[] args) throws Exception {
    new MyHttpServer().start();
  }
}
```

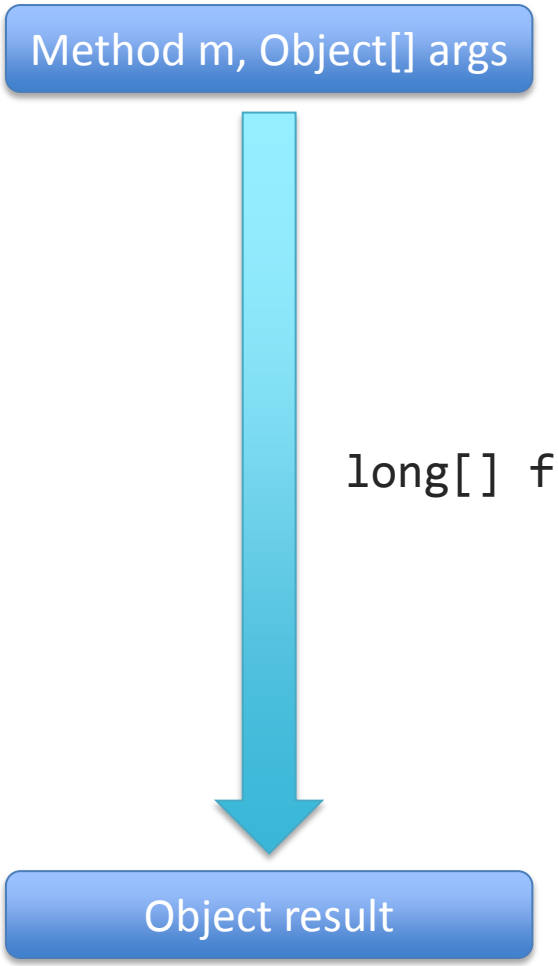
Пример клиента

```
public class MyHttpClient {

  public static void main(String[] args) throws Exception {
    HttpClient client = new HttpClient("https://localhost");

    Response response = client.get("/hello");
    System.out.println("Status: " + response.getStatus());
    System.out.println(response.toString());

    client.close();
  }
}
```

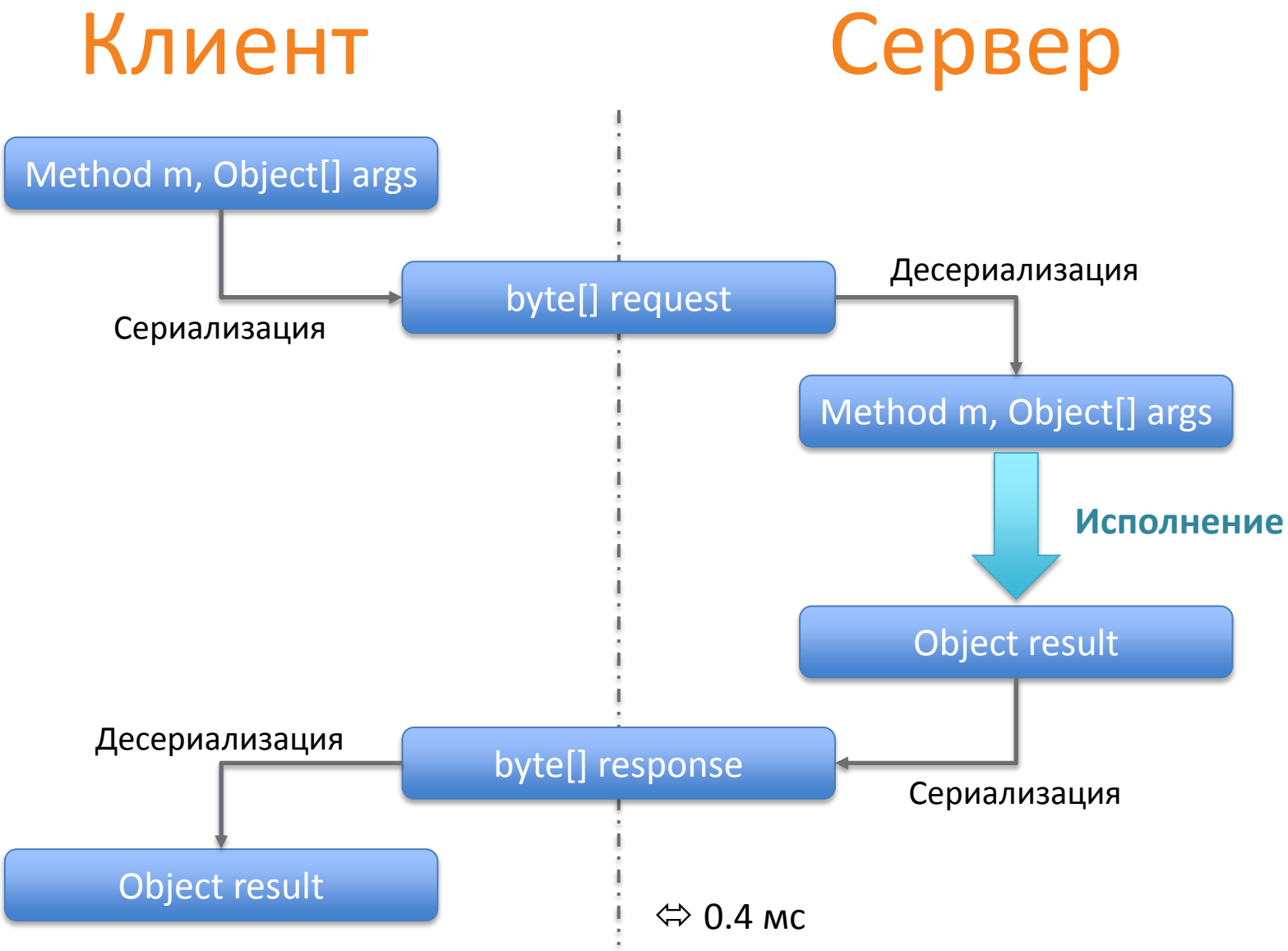
План доклада

- Архитектура ОК
- Работа с сетью
- Удалённые вызовы и сериализация
- Кеширование
- Паузы JVM, оптимизация GC

RPC сценарий




```
long[] friendIds = graph.getFriends(userId);
```


Клиент
Сервер
Method m, Object[] args
Сериализация
byte[] request
Десериализация
Method m, Object[] args
Исполнение
Object result
Сериализация
byte[] response
Десериализация
Object result
⇔ 0.4 мс

Сериализация

- Быстрая, компактная
- Минимум ручной работы
- Поддержка эволюции
  - ~~Java Serialization~~
  - ~~JBoss~~
  - ~~Thrift~~, ~~Avro~~, ~~Protobuf~~
  - ~~Kryo~~

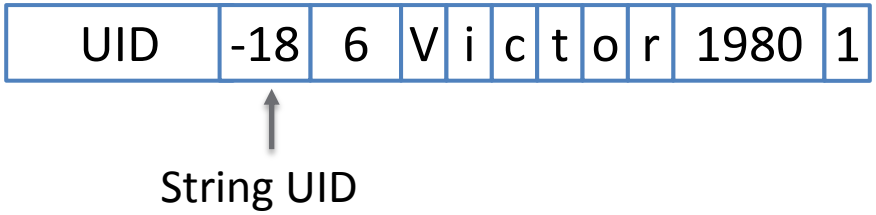
one.nio.serial

- Массивы и коллекции
  - count obj_1 … obj_N
- Мапы
  - count key_1 $value_1$ … key_N $value_N$
- Enum
  - short ordinal()
- Externalizable
  - readExternal / writeExternal
- Остальные Serializable
  - non-static non-transient fields

Схема сериализации

- Serializer
  - Long UID – хеш от имён, типов и порядка полей
- Repository
  - Map<Class, Serializer> для сериализации
  - Map<Long, Serializer> для десериализации

```
class Person {
    String name = "Victor";
    int yearOfBirth = 1980;
    boolean married = true;
}
```



Обмен схемами

1. Client → Server: request
2. Server throws SerializerNotFoundException ?
   - Client → Server: provideSerializer(serializer)
   - Goto 1
3. Deserialize response
4. Client throws SerializerNotFoundException ?
   - Client → Server: requestSerializer(uid)
   - Add to repository
   - Goto 3

Особенности реализации на Java

- Чтение и запись private полей
  - Reflection (медленно!)
  - sun.misc.Unsafe
- Создание экземпляров класса
  - sun.misc.Unsafe.allocateInstance()
  - Генерация байткода: http://asm.ow2.org
- Обход Java верификатора
  - Наследование sun.reflect.MagicAccessorImpl

План доклада

- Архитектура ОК
- Работа с сетью
- Удалённые вызовы и сериализация
- Кеширование
- Паузы JVM, оптимизация GC

Числа, которые нужно знать

L1 cache reference	0.5 ns
Main memory reference	100 ns
Compress 1K bytes w/ cheap algorithm	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Кеширование

- Что кешировать?
  - Данные из медленного хранилища (БД)
  - Результаты вычислений
- Где кешировать?
  - Java Heap (не подходит для объемов > 10 GB)
  - Off-heap memory

Как выйти за пределы Heap

- Native код (JNI обертки над malloc / free)
  - Платформозавимый
- ByteBuffer.allocateDirect
  - Размер буфера ≤ 2 GB
  - Освобождается автоматически при GC
  - Освобождение вручную:
    ((sun.nio.ch.DirectBuffer) buf).cleaner().clean();
- Unsafe.allocateMemory / freeMemory

Решения для off-heap кешей

- JSR 107: javax.cache
  - Ehcache, Coherence
- Apache DirectMemory
- MapDB
- Chronicle Map
- one-nio

Требования к кешам

- Ключи и значения произвольных типов
  - Long, String, byte[], сериализованные объекты
- Быстродействие
- Атомарные операции: read-modify-write
- До 384 GB RAM, до 100 млн. объектов
- Экспирация по времени
- Вытеснение LRU
- Персистентность

Персистентность

- Решает проблему холодного старта
- 2 уровня
  - Сохранение между перезапусками приложения
  - Снимки на диске (snapshots)
- Создание снимков
  - Stop-the-world
  - По сегментам
  - Copy-on-write (fork trick)

Shared Memory

- Механизм IPC
  - Linux: /dev/shm
  - Поддерживается sendfile()
- Создание объекта Shared Memory в Java
  - new RandomAccessFile("/dev/shm/cache", "rw");
- Отображение в адресное пространство:
  - FileChannel.map() → MappedByteBuffer
  - 2GB, no unmapping

Mapping > 2GB

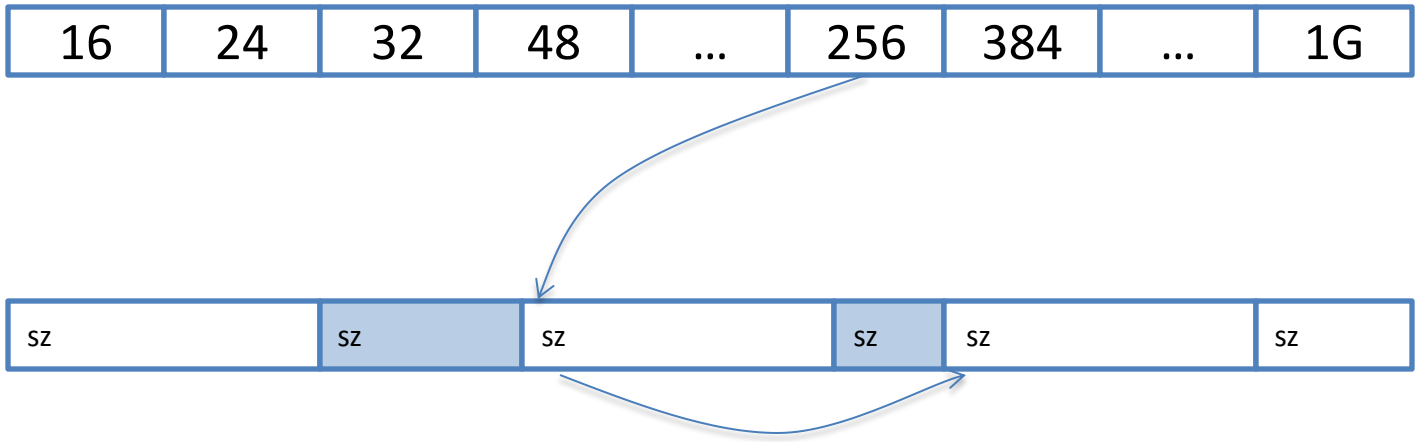
```
// Mapping
Method map0 = FileChannelImpl.class.getDeclaredMethod(
    "map0", int.class, long.class, long.class);
map0.setAccessible(true);
long addr = (Long) map0.invoke(f.getChannel(), 1, 0L, f.length());


// Unmapping
Method unmap0 = FileChannelImpl.class.getDeclaredMethod(
    "unmap0", long.class, long.class);
unmap0.setAccessible(true);
unmap0.invoke(null, addr, length);
```

Проблема абсолютных адресов

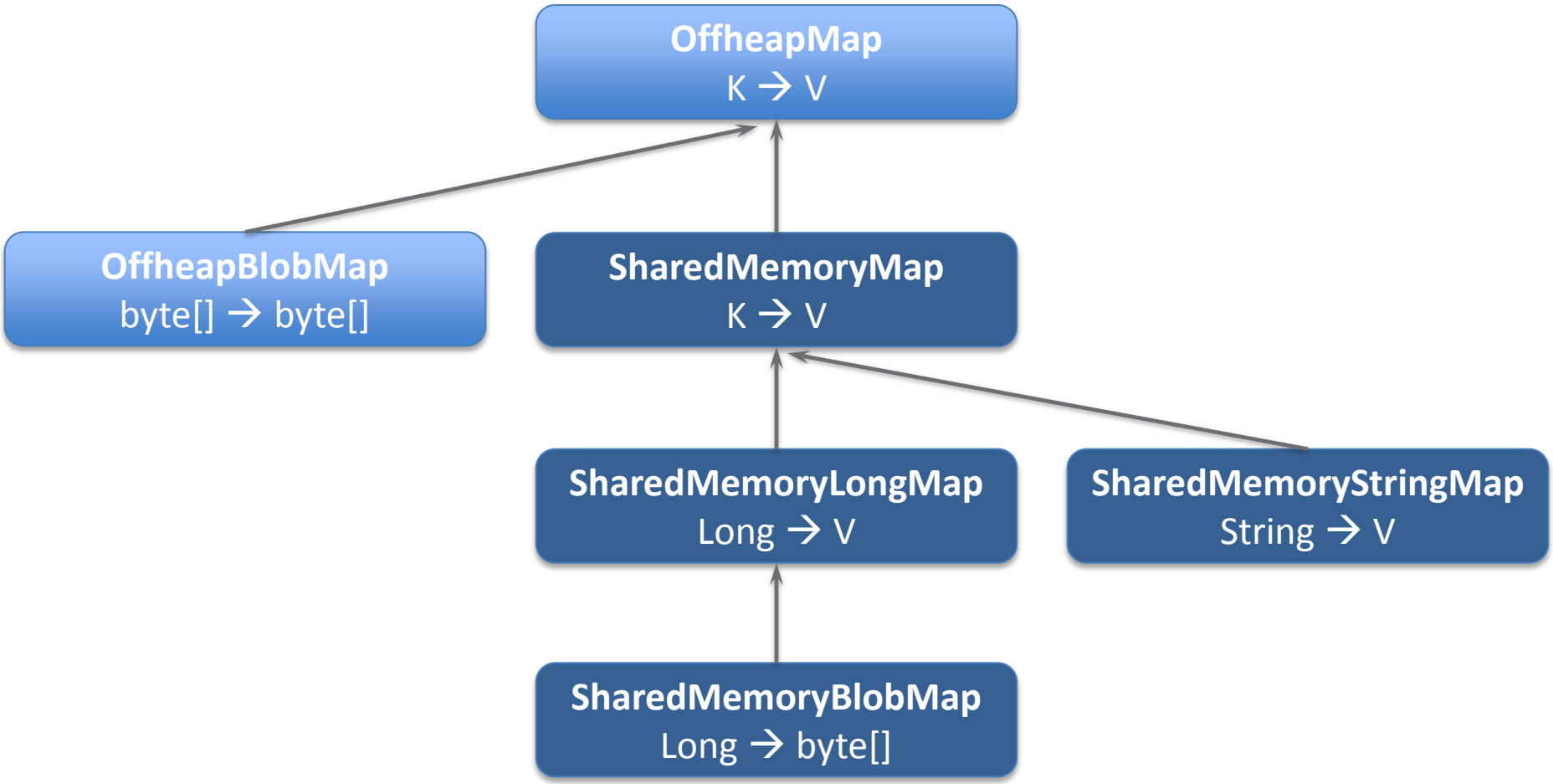
- Только относительная адресация
  - Хранение смещений вместо адресов
- Relocation
  - Сдвиг всех абсолютных адресов на старте

Распределение памяти

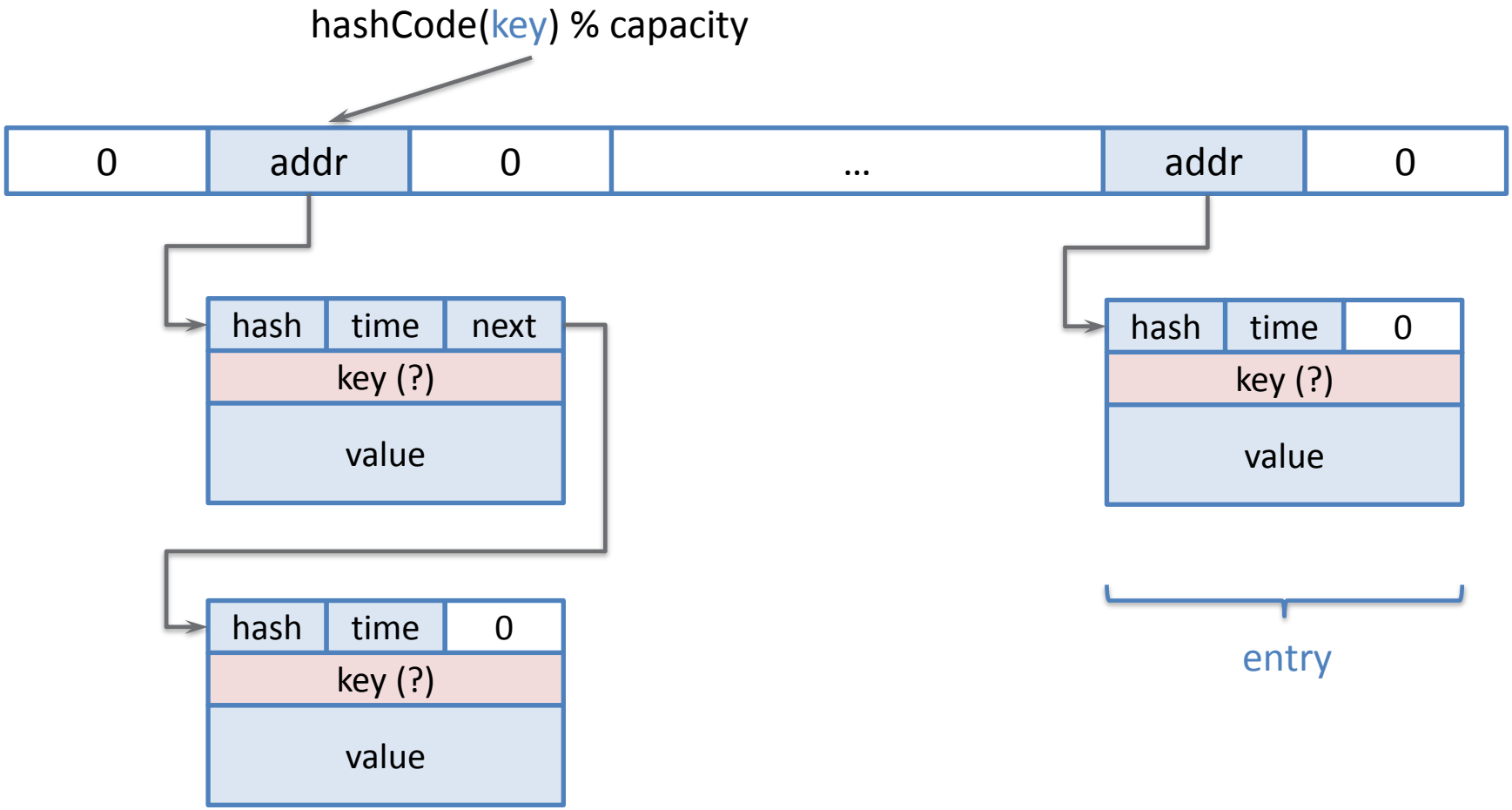
- Doug Lea’s malloc
- one.nio.mem.Malloc, MallocMT



Кеши в one-nio



Устройство кеша



Возможности кеша

- Потокобезопасность
  - get, put, remove, lockRecordForRead/Write
- Прямой доступ к off-heap, сериализация
- Поддержка shared memory
- Стратегии очистки
  - BasicCleanup (TTL), SamplingCleanup (LRU)

План доклада

- Архитектура ОК
- Работа с сетью
- Удалённые вызовы и сериализация
- Кеширование
- Паузы JVM, оптимизация GC

Паузы JVM

- Требования к latency
  - хорошо: 100 мс
  - плохо: 1 с
- GC
  - ParNew
  - CMS (initial mark, remark)
- G1 GC
  - -XX:MaxGCPauseMillis=200
  - Много улучшений в 8u40

Тюнинг GC

```
-XX:+PrintGCDetails
-XX:+PrintGCApplicationStoppedTime

-XX:+PrintClassHistogramBeforeFullGC
-XX:+PrintClassHistogramAfterFullGC
-XX:+PrintPromotionFailure
```

Тюнинг GC

-XX:+UseConcMarkSweepGC
-XX:+ExplicitGCInvokesConcurrent
-XX:+CMSClassUnloadingEnabled (default in 8)

-XX:+UnlockDiagnosticVMOptions
-XX:ParGCCardsPerStrideChunk=32768

Тюнинг GC

```
-XX:+UseCMSInitiatingOccupancyOnly
-XX:CMSInitiatingOccupancyFraction=75

-XX:CMSWaitDuration=10000
-XX:+CMSScavengeBeforeRemark

-XX:+ParallelRefProcEnabled
-XX:+CMSParallelInitialMarkEnabled (default in 8)
```

Паузы JVM

- VM operations
  - Thread dump, heap dump, debugging
  - Biased lock revocation (-XX:-UseBiasedLocking)
  - Deoptimization

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

Safepoint sync

- Непрерываемые операции
  - System.arraycopy(), clone(), ByteBuffer.get()
  - MappedByteBuffer I/O

```
-XX:+SafepointTimeout
-XX:SafepointTimeoutDelay=1000
```

GC-friendly структуры

```
class Entry {
  long key;
  Object value;
}

Entry[] entries;
```

```
long[] keys;
Object[] values;
```

```
class Blob {
  long offset;
  int length;
  byte[] sha1_hash;
}
```

```
class Blob {
  long offset;
  int length;
  int hash0, hash1, hash2,
      hash3, hash4;
}
```

GC-friendly структуры

```
class Props {
  Map<String, String> map;
}
```

```
class Props extends HashMap<String, String>
```

- String → char[] или byte[]
- ByteBuffer wrappers

GC-friendly структуры

- LinkedList → ArrayList
- HashMap → open-address hash table
- Коллекции примитивов (Trove)

- BitSet(100 000) → 100 x BitSet(1000)

Спасибо!

- Наш Open Source
  - https://github.com/odnoklassniki
- Блог
  - http://habrahabr.ru/company/odnoklassniki/blog/
- Контакты
  - andrey.pangin@corp.mail.ru
- Работа в ОК
  - http://v.ok.ru