Мифы и факты о мееееедленной Java

Андрей Паньгин
Одноклассники

2016


Опять всё тормозит?!
Там профилируют
Тут оптимизируют
А код писать кто будет?

О нас

@AndreiPangin


Ведущий программист

Специалист


StackOverflow гик

Одноклассники

- Top 20 SimilarWeb
- 50M DAU
- 1000 Гб/с
- 8000 серверов
- 99% Java

Java в HighLoad

Netty
Jetty
Tomcat
Cassandra
Voldemort
ZooKeeper
Neo4j

Hadoop
HBase
Hive
Lucene
Solr
Elasticsearch
Ignite

В Java медленное выделение памяти

Миф #1

1

Выделение памяти

```
for (int i = 0; i < 100000; i++) {
    char* arr = new char[1024 * 1024];
    delete[] arr;
}
```

C++

```
for (int i = 0; i < 100000; i++) {
    byte[] arr = new byte[1024 * 1024];
}
```

Java

Выделение памяти

```cpp
for (int i = 0; i < 100000; i++) {
    char* arr = new char[1024 * 1024];
    delete[] arr;
}
```

C++

680 ± 50 мс

```java
for (int i = 0; i < 100000; i++) {
    byte[] arr = new byte[1024 * 1024];
}
```

Java

9 020 ± 80 мс

Выделение памяти

```C++
for (int i = 0; i < 100000; i++) {
    volatile char* arr = new char[1024 * 1024];
    arr[0] = 1;
    delete[] arr;
}
```

680 ± 50 мс

```Java
for (int i = 0; i < 100000; i++) {
    byte[] arr = new byte[1024 * 1024];
}
```

9 020 ± 80 мс

Выделение памяти

```C++
for (int i = 0; i < 100000; i++) {
    volatile char* arr = new char[1024 * 1024];
    arr[0] = 1;
    delete[] arr;
}
```

680 ± 50 мс

||

150 ГБ/с

Выделение памяти

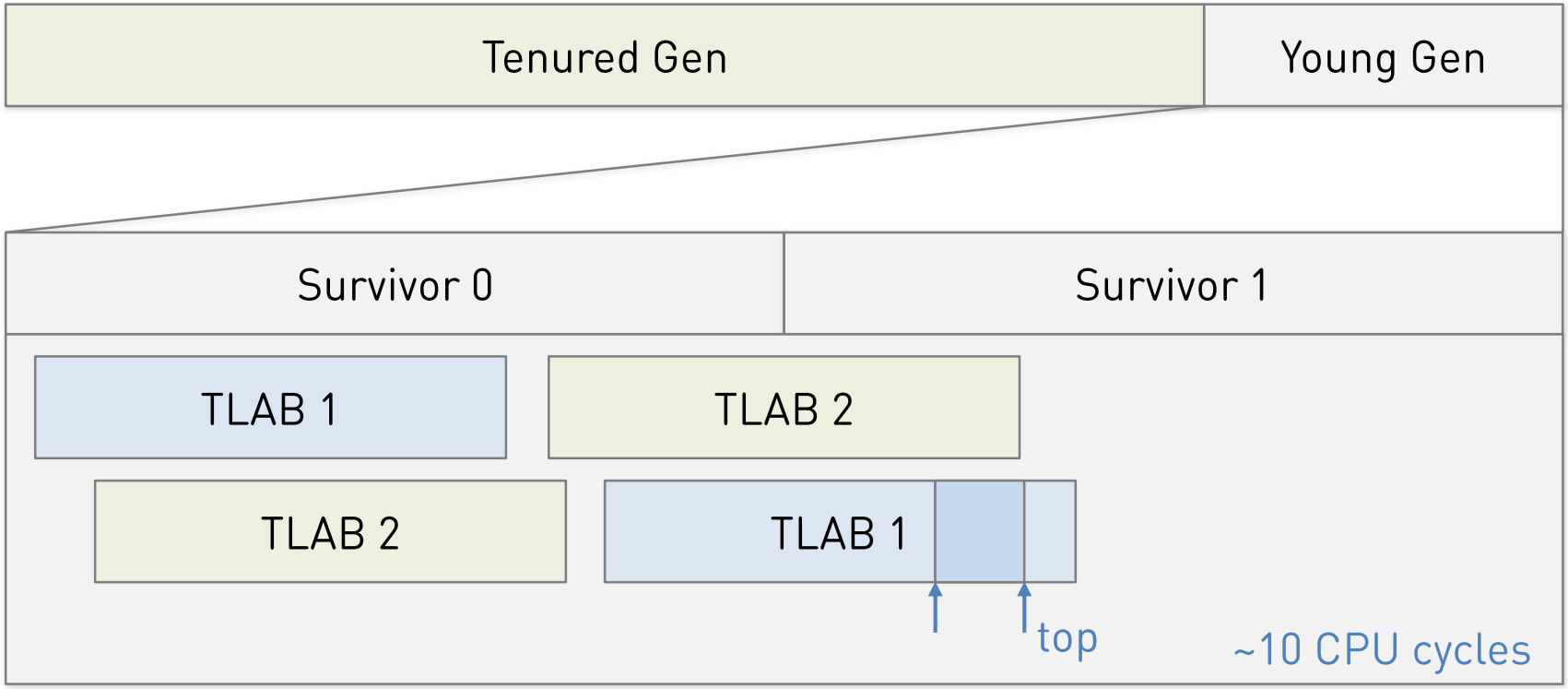
```C++
for (int i = 0; i < 100000; i++) {
    char* arr = new char[1024 * 1024];
    memset(arr, 0, 1024 * 1024);
    delete[] arr;
}
```

33 800 ± 600 мс

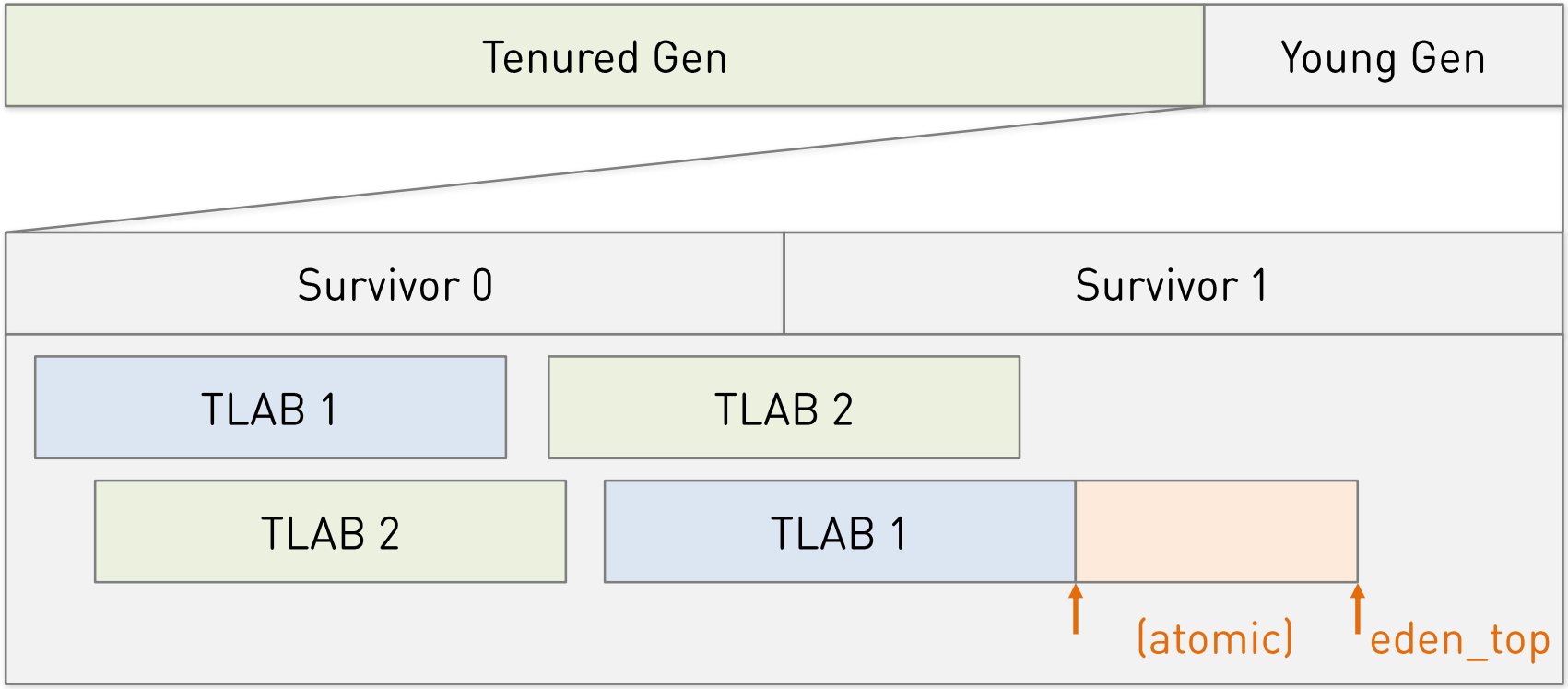
```Java
for (int i = 0; i < 100000; i++) {
    byte[] arr = new byte[1024 * 1024];
}
```

9 020 ± 80 мс

Структура Java Heap



Структура Java Heap



В Java медленная работа с файлами

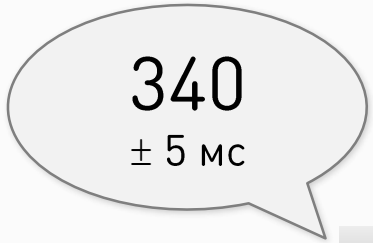
Миф #2

2

Работа с файлами

```c
int fd = open(fileName, O_RDONLY);
size_t size = fsize(fd);
char* buf = (char*)mmap(NULL, size, PROT_READ, MAP_SHARED, fd, 0);

int lines = 0;
for (size_t i = 0; i < size; i++) {
    if (buf[i] == '\n') {
        lines++;
    }
}
printf("Lines = %d\n", lines);
```



C

Работа с файлами

```java
try (RandomAccessFile raf = new RandomAccessFile(fileName, "r")) {

    ByteBuffer buf = raf.getChannel().map(READ_ONLY, 0, raf.length());

    int lines = 0;
    for (int i = 0; i < raf.length(); i++) {
        if (buf.get(i) == '\n') {
            lines++;
        }
    }
    return lines;
}
```

Java

Работа с файлами

```
try (RandomAccessFile raf = new RandomAccessFile(fileName, "r")) {

    ByteBuffer buf = raf.getChannel().map(READ_ONLY, 0, raf.length());

    int lines = 0;
    for (int i = 0; i < raf.length(); i++) {
        if (buf.get(i) == '\n') {
            lines++;
        }
    }
    return lines;
}
```

169 000
мс

Работа с файлами

```
try (RandomAccessFile raf = new RandomAccessFile(fileName, "r")) {

    ByteBuffer buf = raf.getChannel().map(READ_ONLY, 0, raf.length());

    int lines = 0;
    for (int i = 0; i < raf.length(); i++) {
        if (buf.get(i) == '\n') {
            lines++;
        }
    }
    return lines;
}
```

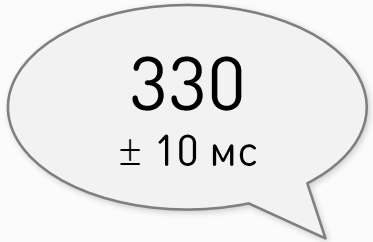
169 000
мс

Работа с файлами

```
try (RandomAccessFile raf = new RandomAccessFile(fileName, "r")) {

    ByteBuffer buf = raf.getChannel().map(READ_ONLY, 0, raf.length());

    int lines = 0;
    int size = (int) raf.length();
    for (int i = 0; i < size; i++) {
        if (buf.get(i) == '\n') {
            lines++;
        }
    }
    return lines;
}
```



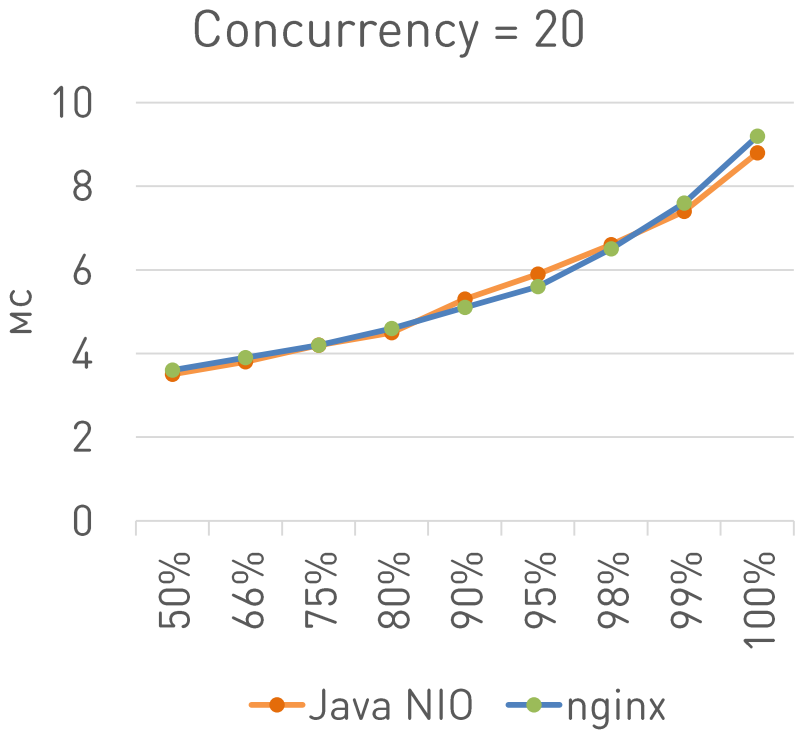
Java

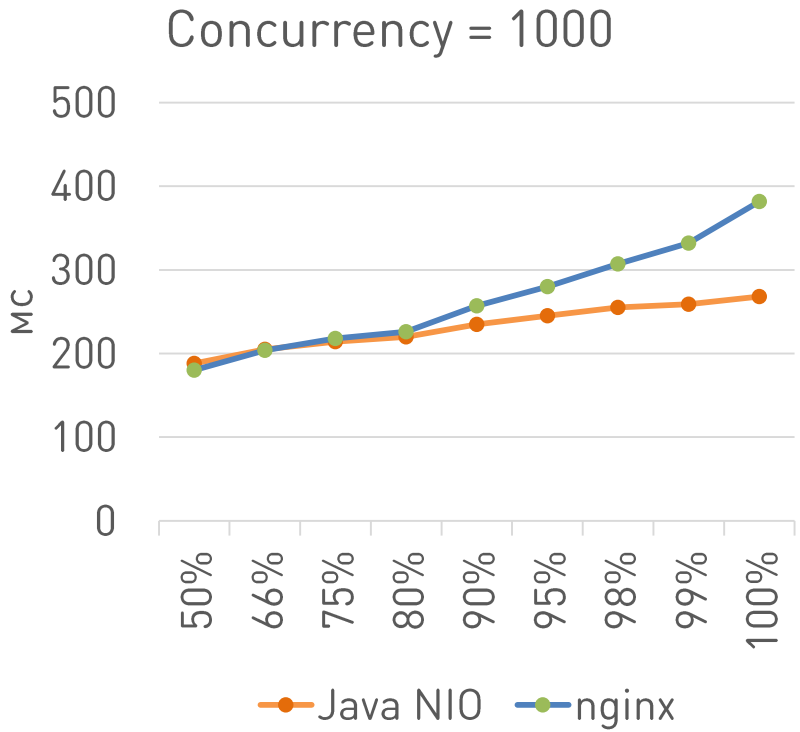
В Java медленная работа с сетью

Миф #3

3

HTTP сервер, 1 Gb/s





ab2 -k -n 10000 -c 1000 "http://server/20k.bin"

Коварный Keep-Alive

```
for (int i = 0; i < 100; i++) {
   Socket s = new Socket(host, 80);
   query(s);
   s.close();
}
```

```
Socket s = new Socket(host, 80);
for (int i = 0; i < 100; i++) {
   query(s);
}
s.close();
```

http://stackoverflow.com/questions/37265825

Коварный Keep-Alive

```
for (int i = 0; i < 100; i++) {
   Socket s = new Socket(host, 80);
   query(s);
   s.close();
}
```

33 мс

```
Socket s = new Socket(host, 80);
for (int i = 0; i < 100; i++) {
   query(s);
}
s.close();
```

60 мс

http://stackoverflow.com/questions/37265825

Коварный Keep-Alive

```
for (int i = 0; i < 100; i++) {
   Socket s = new Socket(host, 80);
   query(s);
   s.close();
}
```

33 мс

```
Socket s = new Socket(host, 80);
for (int i = 0; i < 100; i++) {
   query(s);
}
s.close();
```

60 мс

```
private void query(Socket s) throws IOException {
    DataOutputStream out = new DataOutputStream(s.getOutputStream());
    out.writeBytes("GET / HTTP/1.1 …");

    byte[] response = readResponse();
}
```

http://stackoverflow.com/questions/37265825

Коварный Keep-Alive

```
for (int i = 0; i < 100; i++) {
   Socket s = new Socket(host, 80);
   query(s);
   s.close();
}
```

33 мс

```
Socket s = new Socket(host, 80);
for (int i = 0; i < 100; i++) {
   query(s);
}
s.close();
```

60 мс

```
private void query(Socket s) throws IOException
    DataOutputStream out = new DataOutputStream
    out.writeBytes("GET / HTTP/1.1 …");

    byte[] response = readResponse();
}
```

http://stackoverflow.com/questions/37265825

Коварный Keep-Alive

```
for (int i = 0; i < 100; i++) {
    Socket s = new Socket(host, 80);
    s.setTcpNoDelay(true);
    query(s);
    s.close();
}
```

24 мс

```
Socket s = new Socket(host, 80);
s.setTcpNoDelay(true);
for (int i = 0; i < 100; i++) {
    query(s);
}
s.close();
```

12 мс

```
private void query(Socket s) throws IOException {
    OutputStream out = s.getOutputStream();
    out.write("GET / HTTP/1.1 …".getBytes());

    byte[] response = readResponse();
}
```

http://stackoverflow.com/questions/37265825

Когда Java NIO не хватает?

- 1 Gb/s – легко, как насчёт 40?
- Фичи ОС
  - TCP Fast Open, TCP_DEFER_ACCEPT
  - SO_REUSEPORT, MSG_MORE , MSG_PEEK
- Интеграция с OpenSSL
  - Tickets, OCSP stapling, ALPN

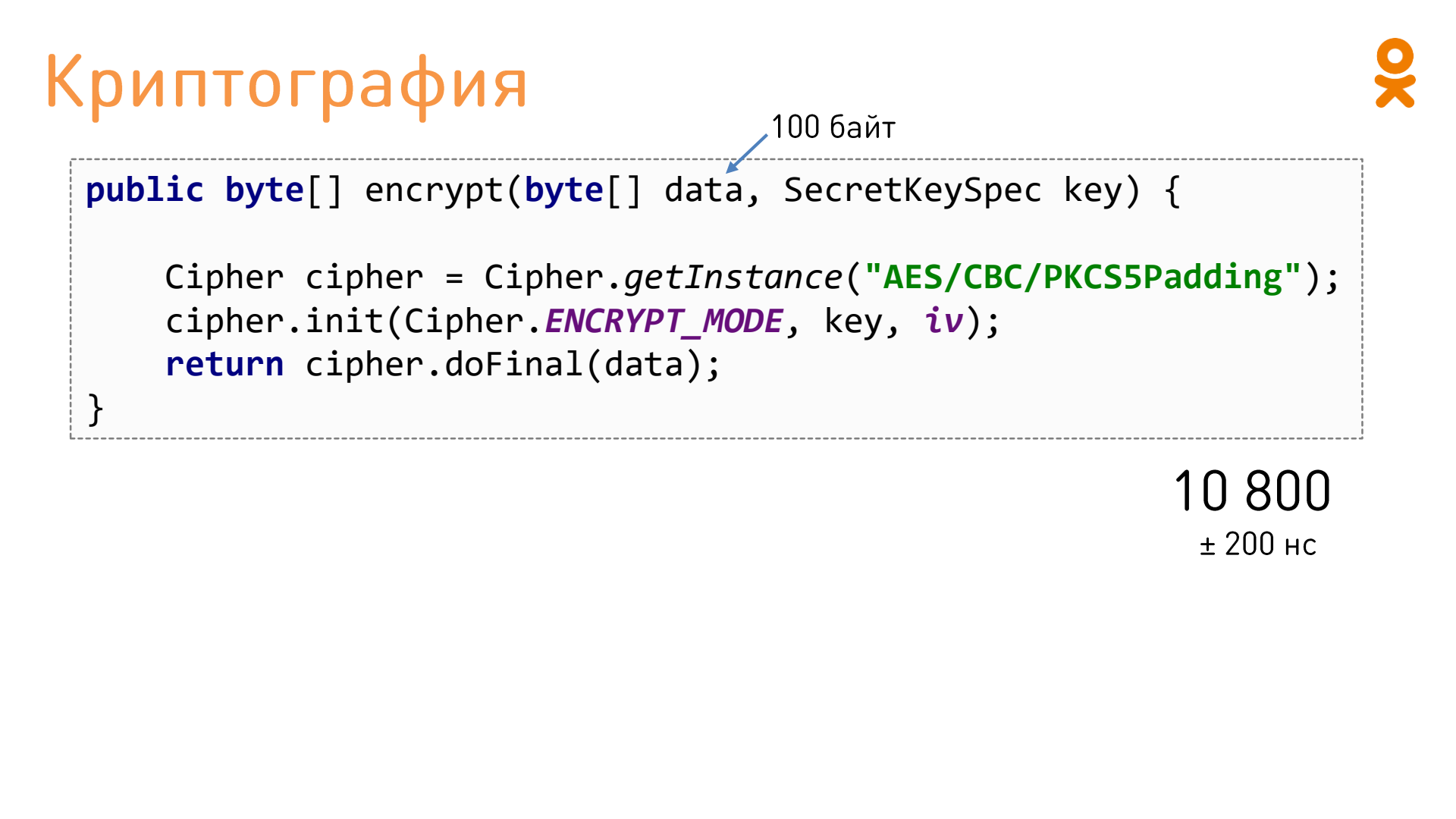
https://github.com/odnoklassniki/one-nio

В Java медленная криптография

Миф #4

4

Криптография




```
public byte[] encrypt(byte[] data, SecretKeySpec key) {

    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, key, iv);
    return cipher.doFinal(data);
}
```

10 800

± 200 нс

Криптография

```
static ThreadLocal<Cipher> CIPHER = …

public byte[] encrypt(byte[] data, SecretKeySpec key) {

    Cipher cipher = CIPHER.get();
    cipher.init(Cipher.ENCRYPT_MODE, key, iv);
    return cipher.doFinal(data);
}
```

10 800 ± 200 нс → 2 400 ± 80 нс

Криптография

```
static ThreadLocal<Cipher> CIPHER = …

public byte[] encrypt(byte[] data, SecretKeySpec key) {

    Cipher cipher = CIPHER.get();
    cipher.init(Cipher.ENCRYPT_MODE, key, iv);
    return cipher.doFinal(data);
}
```

10 800 ± 200 нс → 2 400 ± 80 нс → 380 ± 20 нс

Криптография

- Hardware support
  - AES-NI

- -XX:+UseAESIntrinsics
  - AES/ECB
  - AES/CBC

В Java неэффективные потоки

Миф #5

5

Многопоточность

```
double[] arr = new Random().doubles(100_000_000)
        .map(Math::tanh)
        .toArray();
```

14,1 ± 0,1 с

Многопоточность

```
double[] arr = new Random().doubles(100_000_000)
        .parallel()
        .map(Math::tanh)
        .toArray();
```

14,1 ± 0,1 с

13,0 ± 0,2 с

Многопоточность

```
double[] arr = new Random().doubles(100_000_000)
        .parallel()
        .map(Math::tanh)
        .toArray();
```

```
Random.next(int bits)
```

```
    AtomicLong seed = this.seed;
    do {
        oldseed = seed.get();
        nextseed = (oldseed * multiplier + addend) & mask;
    } while (!seed.compareAndSet(oldseed, nextseed));
```

Многопоточность

```
double[] arr = ThreadLocalRandom.current().doubles(100_000_000)
        .parallel()
        .map(Math::tanh)
        .toArray();
```

14,1 ± 0,1 с

13,0 ± 0,2 с

3,9 ± 0,2 с

Java потоки vs. Native потоки

- 1 ⇔ 1
- -XX:ThreadPriorityPolicy=1
  - требует root
- CPU Affinity


BUSTED

Как устроен компилятор в JVM

6

Когда узнал, что в HotSpot нет JIT

Когда узнал, что в HotSpot нет JIT

- DAC (Dynamic Adaptive Compiler)
  - Interpreter
  - Background compilation
  - Profiling
  - OSR
  - Tiered Compilation
  - Deoptimization
  - Recompilation

Client vs. Server

	C1	C2
Скорость компиляции	быстро	умеренно
Оптимизации	простые	средние

- jre/lib/amd64/jvm.cfg

```
-server KNOWN
-client IGNORE
```

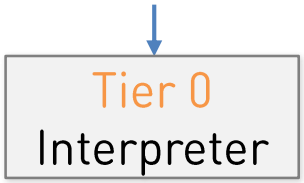
Tiered Compilation

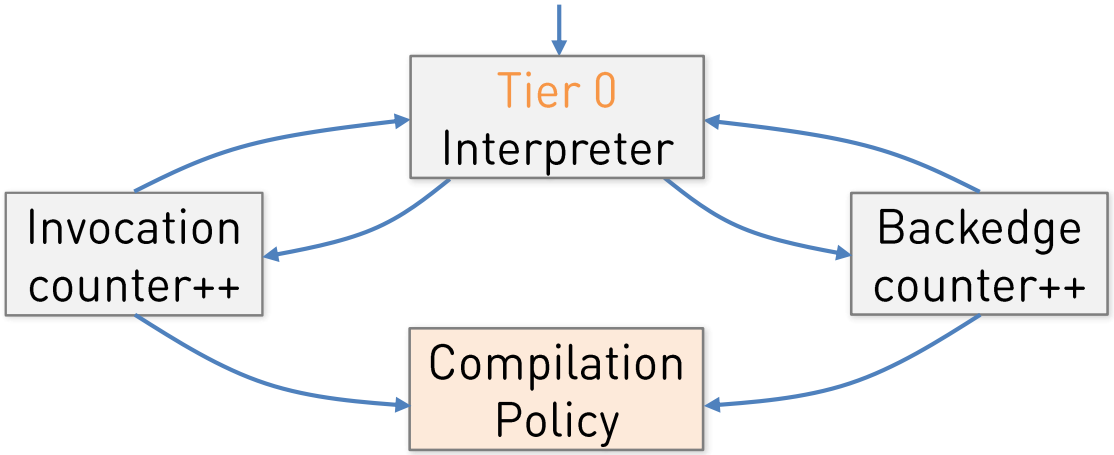
- По умолчанию в JDK 8
- Одна JVM – два компилятора

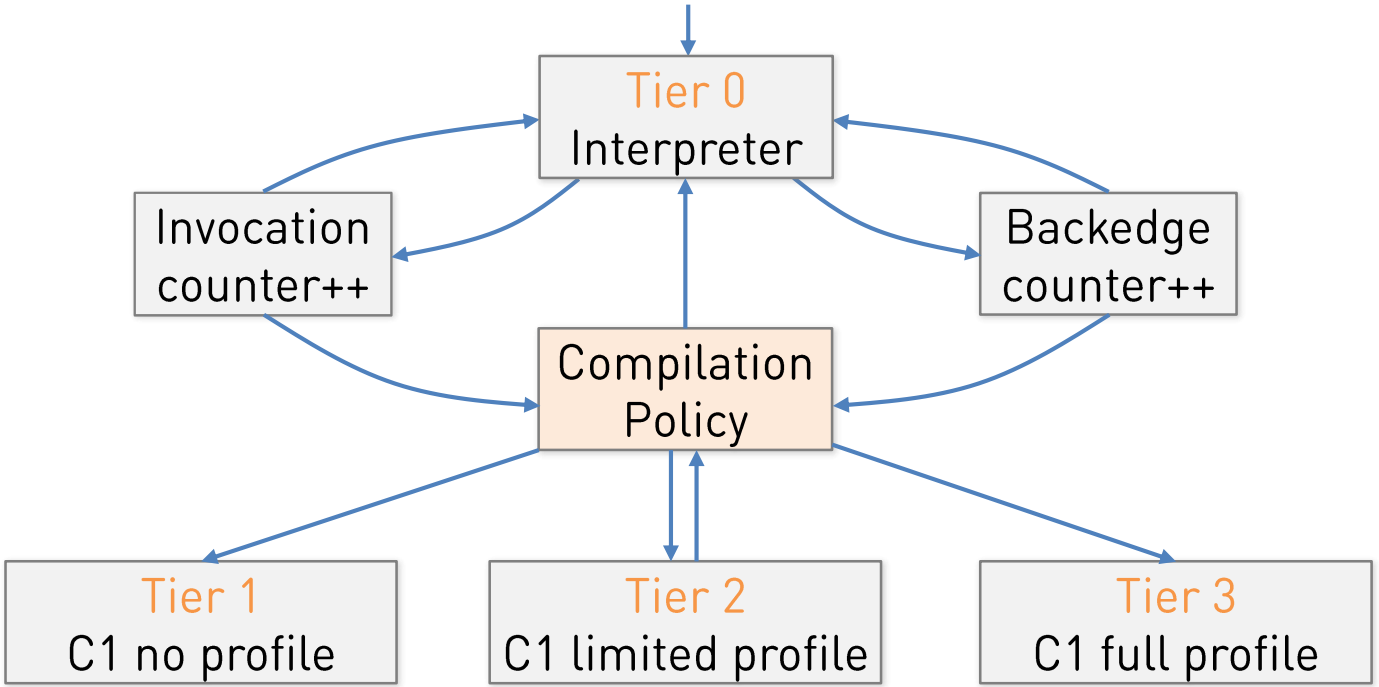
- Только C1: -XX:TieredStopAtLevel=1

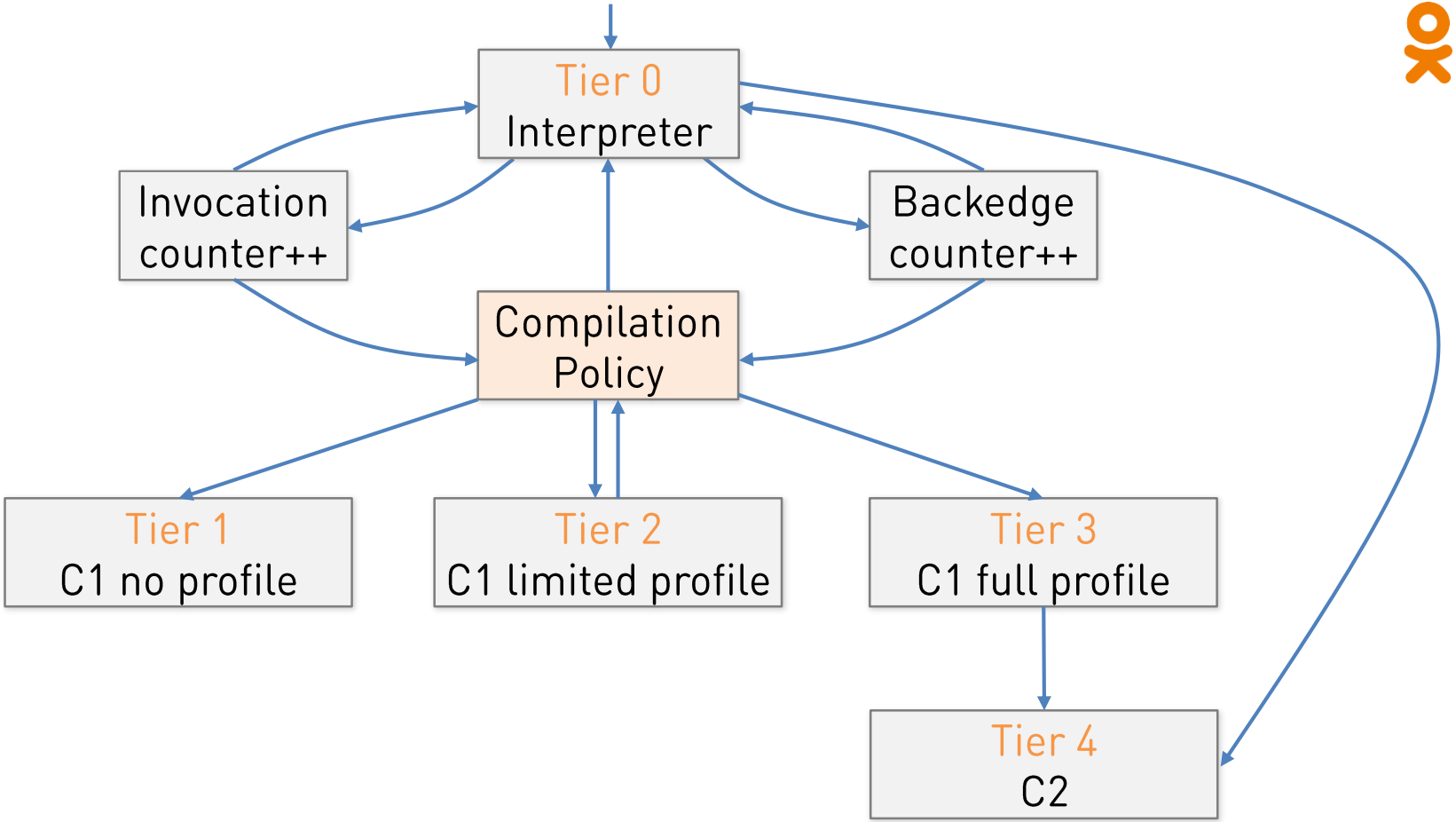
- Только C2: -XX:-TieredCompilation
  - ReservedCodeCacheSize меньше в 5 раз

Выбор горячих методов


Tier 0
Interpreter


Tier 0
Interpreter
Invocation
counter++
Backedge
counter++
Compilation
Policy


Tier 0
Interpreter
Invocation
counter++
Backedge
counter++
Compilation
Policy
Tier 1
C1 no profile
Tier 2
C1 limited profile
Tier 3
C1 full profile


Tier 0
Interpreter
Invocation
counter++
Backedge
counter++
Compilation
Policy
Tier 1
C1 no profile
Tier 2
C1 limited profile
Tier 3
C1 full profile
Tier 4
C2

Компиляторные оптимизации

7

Оптимизации

- compiler tactics
  - delayed compilation
  - tiered compilation
  - on-stack replacement
  - delayed reoptimization
  - program dependence graph representation
  - static single assignment representation
- proof-based techniques
  - exact type inference
  - memory value inference
  - memory value tracking
  - constant folding
  - reassociation
  - operator strength reduction
  - null check elimination
  - type test strength reduction
  - type test elimination
  - algebraic simplification
  - common subexpression elimination
  - integer range typing
- flow-sensitive rewrites
  - conditional constant propagation
  - dominating test detection
  - flow-carried type narrowing
  - dead code elimination
- language-specific techniques
  - class hierarchy analysis
  - devirtualization
  - symbolic constant propagation
  - autobox elimination
  - escape analysis
  - lock elision
  - lock fusion
  - de-reflection
- speculative (profile-based) techniques
  - optimistic nullness assertions
  - optimistic type assertions
  - optimistic type strengthening
  - optimistic array length strengthening
  - untaken branch pruning
  - optimistic N-morphic inlining
  - branch frequency prediction
  - call frequency prediction
- memory and placement transformation
  - expression hoisting
  - expression sinking
  - redundant store elimination
  - adjacent store fusion
  - card-mark elimination
  - merge-point splitting
- loop transformations
  - loop unrolling
  - loop peeling
  - safepoint elimination
  - iteration range splitting
  - range check elimination
  - loop vectorization
- global code shaping
  - inlining (graph integration)
  - global code motion
  - heat-based code layout
  - switch balancing
  - throw inlining
- control flow graph transformation
  - local code scheduling
  - local code bundling
  - delay slot filling
  - graph-coloring register allocation
  - linear scan register allocation
  - live range splitting
  - copy coalescing
  - constant splitting
  - copy removal
  - address mode matching
  - instruction peepholing
  - DFA-based code generator

Inlining

```
byte value = buffer.get(2);
```

```
public byte get(int i) {
    return hb[ix(checkIndex(i))];
}
```

Inlining

```
byte value = buffer.hb[buffer.ix(buffer.checkIndex(2))];
```

```
final int checkIndex(int i) {
    if ((i < 0) || (i >= limit))
        throw new IndexOutOfBoundsException();
    return i;
}
```

Inlining

```
if ((2 < 0) || (2 >= buffer.limit))
    throw new IndexOutOfBoundsException();
byte value = buffer.hb[buffer.ix(2)];
```

```
protected int ix(int i) {
    return i + offset;
}
```

Inlining

```
if ((2 < 0) || (2 >= buffer.limit))
    throw new IndexOutOfBoundsException();
byte value = buffer.hb[2 + buffer.offset];
```

Inlining

```
-XX:MaxInlineLevel=9
-XX:MaxInlineSize=35
-XX:FreqInlineSize=325
-XX:MaxTrivialSize=6
-XX:MinInliningThreshold=250
-XX:LiveNodeCountInliningCutoff=40000
```

Loop peeling & invariant hoisting

```
for (Listener listener : listeners) {
    listener.fire((Event) arguments.get(0));
}
```

```
int length = listeners.length;
if (length != 0) {
    Event event = (Event) arguments.get(0);
    listeners[0].apply(event);

    for (int i = 1; i < length; i++) {
        listeners[i].apply(event);
    }
}
```

Loop Unrolling

```
for (int i = 0; i < A.length; i++) {
    for (int j = i; j > 0 && A[j-1] > A[j]; j--) {
        int tmp = A[j-1];
        A[j-1] = A[j];
        A[j] = tmp;
    }
}
```

4420 ± 20 мс


```
int tmp = A[j];
A[j] = A[j-1];
A[j-1] = tmp;
```

1930 ± 20 мс

Loop Unrolling

```
for (int j = i; j > 3; j -= 4) {

    if (!(A[j-1] > A[j])) break;
    int tmp = A[j];
    A[j] = A[j-1];
    A[j-1] = tmp;

    if (!(A[j-2] > A[j-1])) break;
    tmp = A[j-1];
    A[j-1] = A[j-2];
    A[j-2] = tmp;
    ...
}
```

Loop Unrolling

```
for (int j = i; j > 3; j -= 4) {

    if (!(A[j-1] > A[j])) break;
    int tmp = A[j];
    A[j] = A[j-1];
    A[j-1] = tmp;

    if (!(A[j-2] > A[j-1])) break;
    tmp = A[j-1];
    A[j-1] = A[j-2];
    A[j-2] = tmp;
    ...
}
```

Loop Unrolling

```
for (int j = i; j > 3; j -= 4) {

    if (!(A[j-1] > A[j])) break;
    int tmp = A[j];
    A[j] = A[j-1];
    A[j-1] = tmp;

    if (!(A[j-2] > tmp)) break;
    tmp = A[j-1];
    A[j-1] = A[j-2];
    A[j-2] = tmp;
    ...
}
```

Loop Unrolling

```
for (int j = i; j > 3; j -= 4) {

    if (!(A[j-1] > A[j])) break;
    int tmp = A[j-1];
    A[j-1] = A[j];
    A[j] = tmp;

    if (!(A[j-2] > A[j-1])) break;
    tmp = A[j-2] ;
    A[j-2] = A[j-1];
    A[j-1] = tmp;
    ...
}
```

-XX:LoopUnrollLimit=0

Allocation elimination

```
static final Rectangle RECT = new Rectangle(10, 20, 30, 40);

@Benchmark
public boolean newRect() {
    return new Rectangle(10, 20, 30, 40).contains(25, 25);
}

@Benchmark
public boolean staticRect() {
    return RECT.contains(25, 25);
}
```

2,85 ± 0,1 нс

5,44 ± 0,1 нс

Allocation elimination

```
@Benchmark
public boolean newRect() {
    Rectangle r = new Rectangle();
    r.x = 10;
    r.y = 20;
    r.width = 30;
    r.height = 40;
    return r.contains(25, 25);
}
```

Scalar Replacement

```
@Benchmark
public boolean newRect() {
    Rectangle r = new Rectangle();
    int x = 10;
    int y = 20;
    int width = 30;
    int height = 40;
    return 25 >= x && 25 >= y &&
           25 < (x + width) && 25 < (y + height);
}
```

Constant Folding

```
@Benchmark
public boolean newRect() {
    int x = 10;
    int y = 20;
    int width = 30;
    int height = 40;
    return 25 >= x && 25 >= y &&
           25 < (x + width) && 25 < (y + height);
    return true;
}
```

Allocation elimination

```
public Point getPoint() {
    ...
    return new Point(x, y);
}

public void print() {
    Point p = getPoint();
    System.out.println(p.getX());
    System.out.println(p.getY());
}
```

Allocation elimination

```
public Point getPoint() {
    ...
    return new Point(x, y);
}

public void print() {
    Point p = new Point();
    int x = ...
    int y = ...
    System.out.println(x);
    System.out.println(y);
}
```

-XX:+EliminateAllocations

VarArg elimination

```
public void log(Object... args) {
    for (Object arg : args) {
        print(arg);
    }
}

public void operation() {
    ...
    log("operation", 100, true);
}
```

VarArg elimination

```
public void log(Object... args) {
  if (args.length > 0) print(args[0]);
  if (args.length > 1) print(args[1]);
  if (args.length > 2) print(args[2]);
  if (args.length > 3) print(args[3]);
}

public void operation() {
    ...
    log("operation", 100, true);
}
```

Lock optimizations

```
@Benchmark
public int list() {
    return Collections.binarySearch(list, 555);
}

@Benchmark
public int vector() {
    return Collections.binarySearch(vector, 555);
}
```

32,3
± 0,1 нс

250,8
± 3 нс

Lock optimizations

```
@Benchmark
public int list() {
    return Collections.binarySearch(list, 555);
}

@Benchmark
public int vector() {
    return Collections.binarySearch(vector, 555);
}
```

32,3
± 0,1 нс

48,0
± 0,8 нс

-XX:BiasedLockingStartupDelay=0

Lock optimizations

```
@Benchmark
public int list() {
    return Collections.binarySearch(list, 555);
}

@Benchmark
public int vector() {
    Vector<Integer> v = vector;
    synchronized (v) {
        return Collections.binarySearch(v, 555);
    }
}
```

32,3
± 0,1 нс

35,6
± 0,5 нс

String optimizations

```
@Benchmark
public int concat() {
    return "Point[x=" + p.x + ",y=" + p.y + "]";
}

@Benchmark
public int stringBuilder() {
    StringBuilder sb = new StringBuilder(32);
    sb.append("Point[x=");
    sb.append(p.x);
    sb.append(",y=");
    sb.append(p.y);
    sb.append("]");
    return sb.toString();
}
```

String optimizations

```java
@Benchmark
public int concat() {
    return "Point[x=" + p.x + ",y=" + p.y + "]";
}

@Benchmark
public int stringBuilder() {
    StringBuilder sb = new StringBuilder(32);
    sb.append("Point[x=");
    sb.append(p.x);
    sb.append(",y=");
    sb.append(p.y);
    sb.append("]");
    return sb.toString();
}
```

38,1
± 1 нс

60,8
± 2 нс

Strength reduction

```
n / 8
```

```
n >> 3
```

```
n * 5
```

```
(n << 2) + n
```

```
Math.pow(x, 2)
```

?

```
x * x
```

Intrinsics

Math	min, max, pow, sqrt, sin, cos
Object	getClass, hashCode, clone
Class	getSuperclass, isInstance, isPrimitive
Integer	reverseBytes, bitCount, numberOfLeadingZeros
System	arrayCopy, currentTimeMillis, nanoTime
Thread	currentThread, isInterrupted
String	equals, indexOf, compareTo
Unsafe	getX, putX, compareAndSwapX, copyMemory

http://hg.openjdk.java.net/jdk8u/jdk8u/hotspot/file/tip/src/share/vm/classfile/vmSymbols.hpp

Хочу больше оптимизаций!

- -XX:+AggressiveOpts

- -XX:+UseFastAccessorMethods

Хочу больше оптимизаций!

- -XX:+AggressiveOpts
  - JDK-8150552
    Remove -XX:+AggressiveOpts

- -XX:+UseFastAccessorMethods
  - JDK-6385687
    UFAM considered harmful

Как не надо писать бенчмарки

8

java -version

Client vs. Server VM

32-bit vs. 64-bit

Разрешение таймера

System.currentTimeMillis

System.nanoTime

Все яйца в одной корзине

```
t0 = System.nanoTime();
for (int i = 0; i < 1000_000; i++) {
    algo1();
}
t1 = System.nanoTime();

t0 = System.nanoTime();
for (int i = 0; i < 1000_000; i++) {
    algo2();
}
t1 = System.nanoTime();
```

OSR заглушка

```
public static void main(String[] args) {

    long startTime = System.nanoTime();
    for (double d = 0; d < 1000_000; d++) {
        // loop
    }
    long endTime = System.nanoTime();
```

3,6 ± 0,2 мс

```
    DoubleWrapper w = new DoubleWrapper();
    for (w.d = 0; w.d < 1000_000; w.d++) {
        // loop
    }
```

1,4 ± 0,2 мс

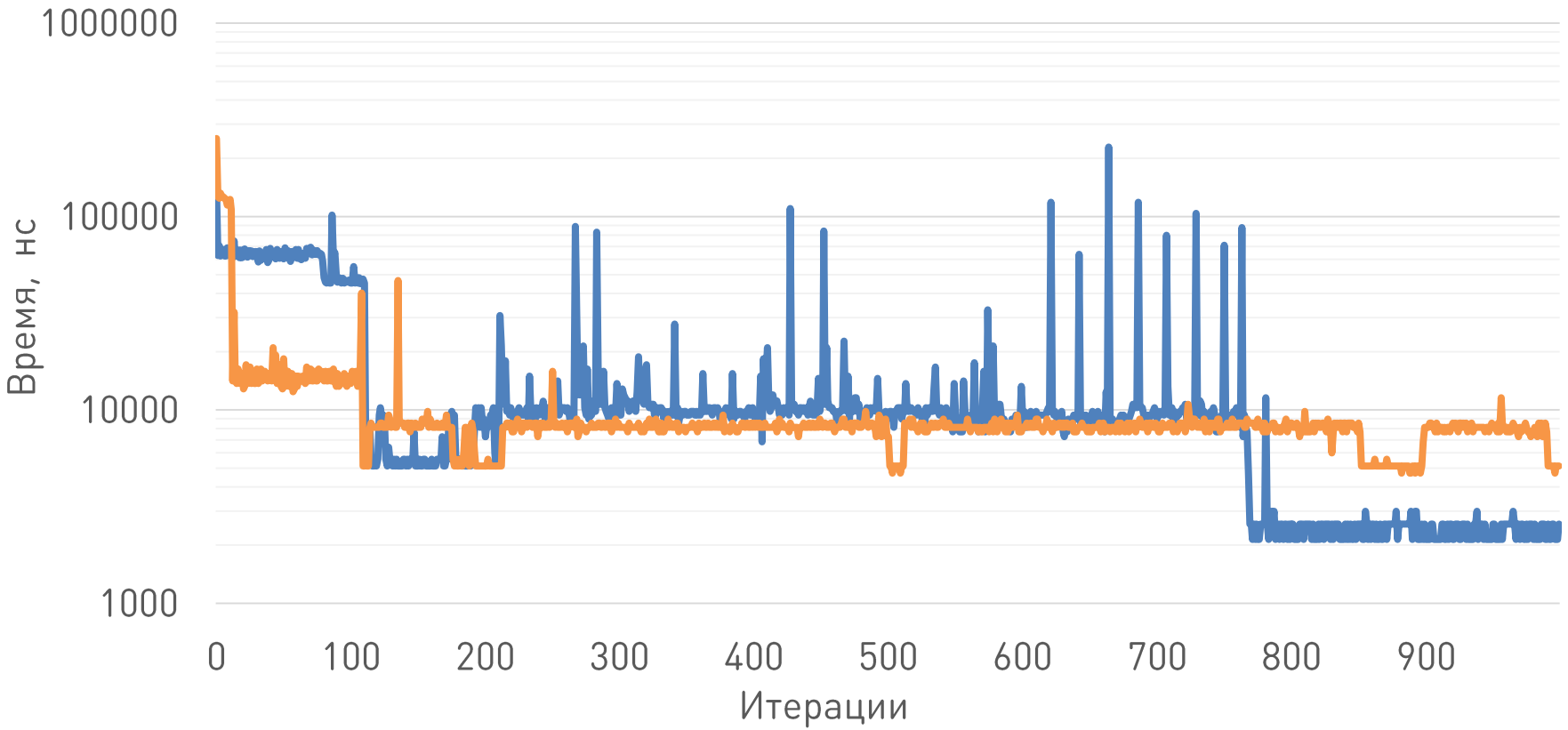
http://stackoverflow.com/questions/33842116

OSR заглушка

```
    for (double d = 0; d < 1000_000; d++) {
        // calculations
    }
```

- В отдельный метод!
- Вызывать несколько раз

Устойчивость



Profile Pollution

```
@Param({"500", "520"})
int iterations;

@Setup
public void init() {
    for (int i = 0; i < iterations; i++) {
        stream().collect(...);
    }
}

@Benchmark
public long loop() {
    return stream().count();
}
```

(500) 33 ± 1 нс
(520) 82 ± 3 нс

http://stackoverflow.com/questions/33842116

Side effects

```
@Benchmark
public void testMethodRef() {
    Arrays.stream(array).forEach(System.out::println);
}
```

Анализ байткода

```
@Benchmark
public double sqrtD() {
    return Math.sqrt(x);
}

@Benchmark
public float sqrtF() {
    return (float) Math.sqrt((float) x);
}
```

Анализ байткода

double sqrtD()

```
0: aload_0
1: getfield      x:D
4: invokestatic  sqrt:(D)D
7: dreturn
```

6,8
± 0,2 нс

float sqrtF()

```
 0: aload_0
 1: getfield      x:D
 4: d2f
 5: f2d
 4: invokestatic  sqrt:(D)D
 9: d2f
10: freturn
```

4,4
± 0,1 нс

Анализ байткода

double sqrtD()

```
0: aload_0
1: getfield      x:D
4: invokestatic  sqrt:(D)D
7: dreturn
```

6,8
± 0,2 нс

float sqrtF()

```
 0: aload_0
 1: getfield      x:D
 4: d2f
 5: f2d
 4: invokestatic  sqrt:(D)D
 9: d2f
10: freturn
```

4,4
± 0,1 нс

f2d → Math.sqrt() → d2f
“sqrtss”

Не все байткоды одинаково полезны

- Анализируйте скомпилированный код
  - -XX:+UnlockDiagnosticVMOptions
    -XX:+PrintAssembly
  - jmh –prof perfasm

И как теперь с этим жить?

- Пишите простой, понятный код
  - Он нравится не только людям, но и JVM
- Пользуйтесь правильными инструментами
  - http://openjdk.java.net/projects/code-tools/jmh/
- Не верьте цифрам! Подключайте голову.

Вопросы?