Внутрь JVM сквозь замочную скважину hashCode

https://github.com/vladimirdolzhenko/hashCodeLegend

Владимир Долженко, IHS Markit

@dolzhenko
vladimir.dolzhenko @ gmail.com

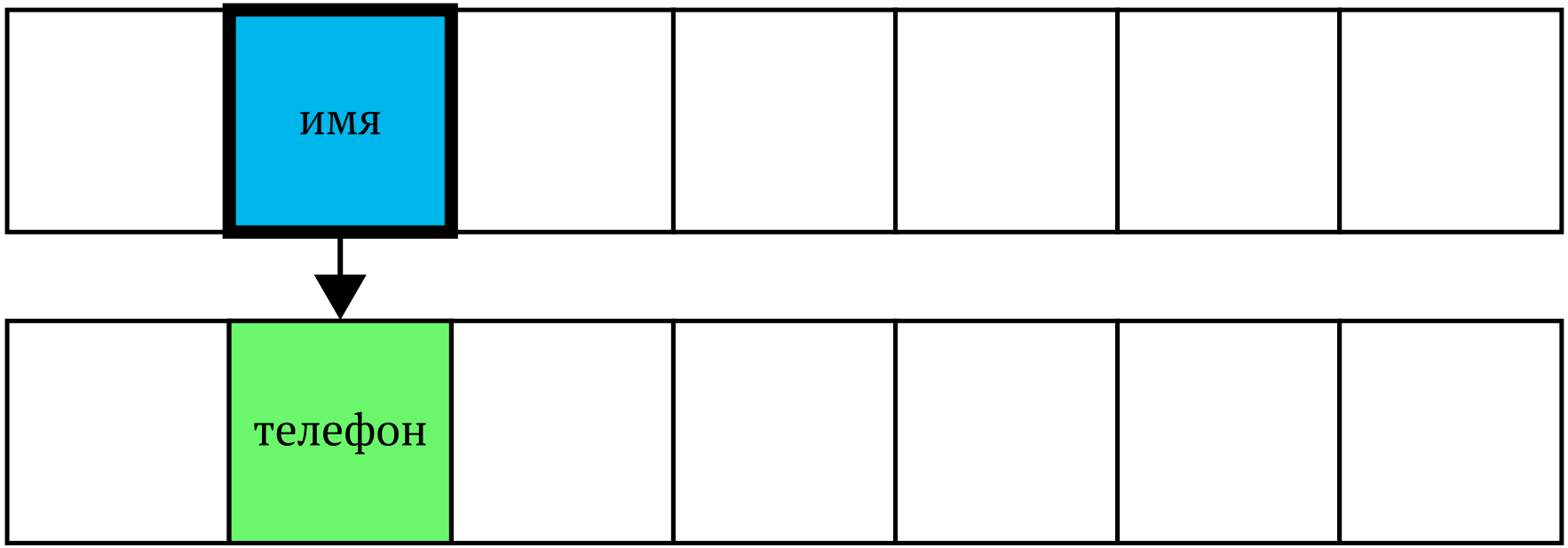
2017-04-10

no warranty

План

- Немного теории
- Правила вычисления hashCode
  - DoS-атака с примененимем hashCode
- HashCode как адрес объекта - миф или реальность
  - JVM потроха: GC и аллокации
  - Немного баттлов про hashCode
- Правильные инструменты: JMH и JOL
- Удешевление блокировок

Ассоциативный массив / Dictionary / Map




```
int hash =  hash( key );
int index = Math.abs( hash )
          % keys.length;
```

javadoc:

java.lang.Object

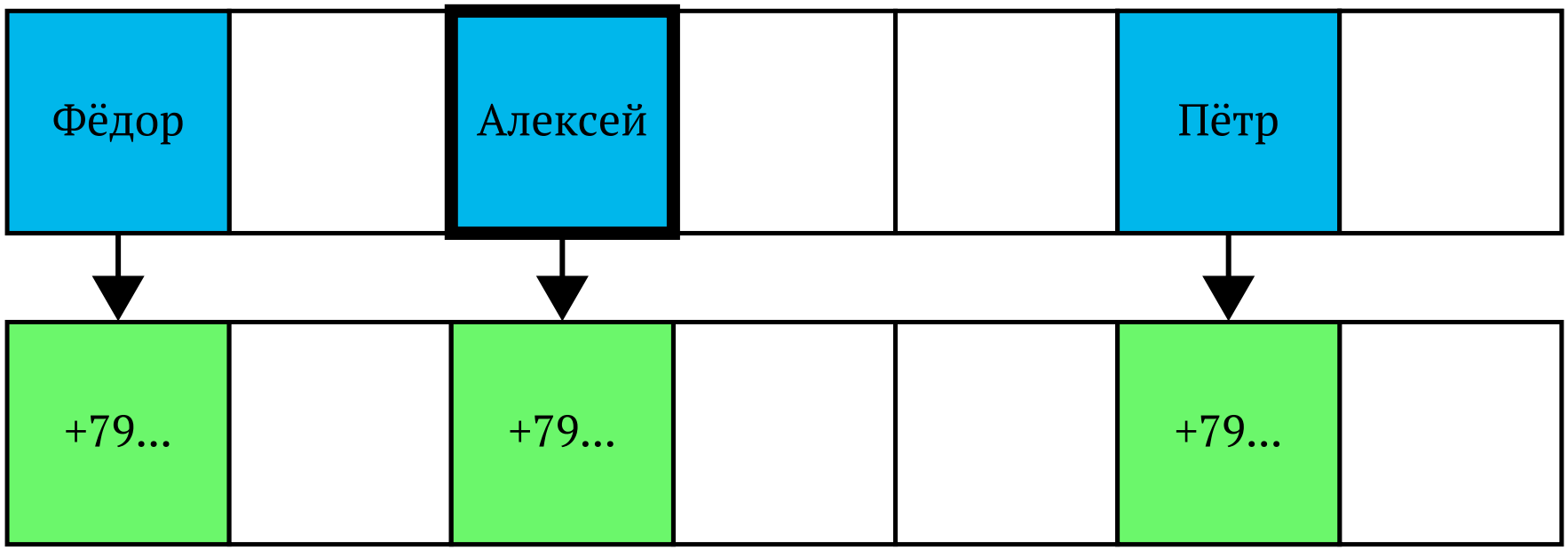
`public int hashCode()`

Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by **HashMap**.

https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

Записать (Алексей, +79…)

hashCode(Алексей) = 23 index = 23 % array.length = 2



Сложность поиска / вставки

```
String key = "Алексей";

int hash = key.hashCode();
int index = Math.abs( hash ) % keys.length

if ( key.equals( keys[index] ) ) {
    return values[index];
}
return null;
```

сложность → **O(1)**

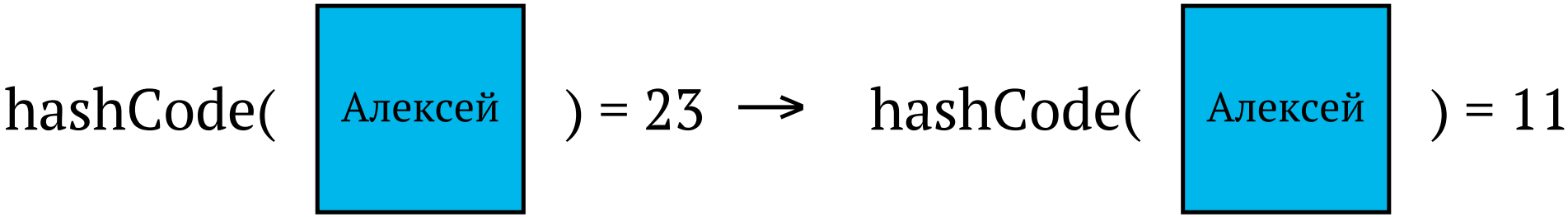
Контракт hashCode

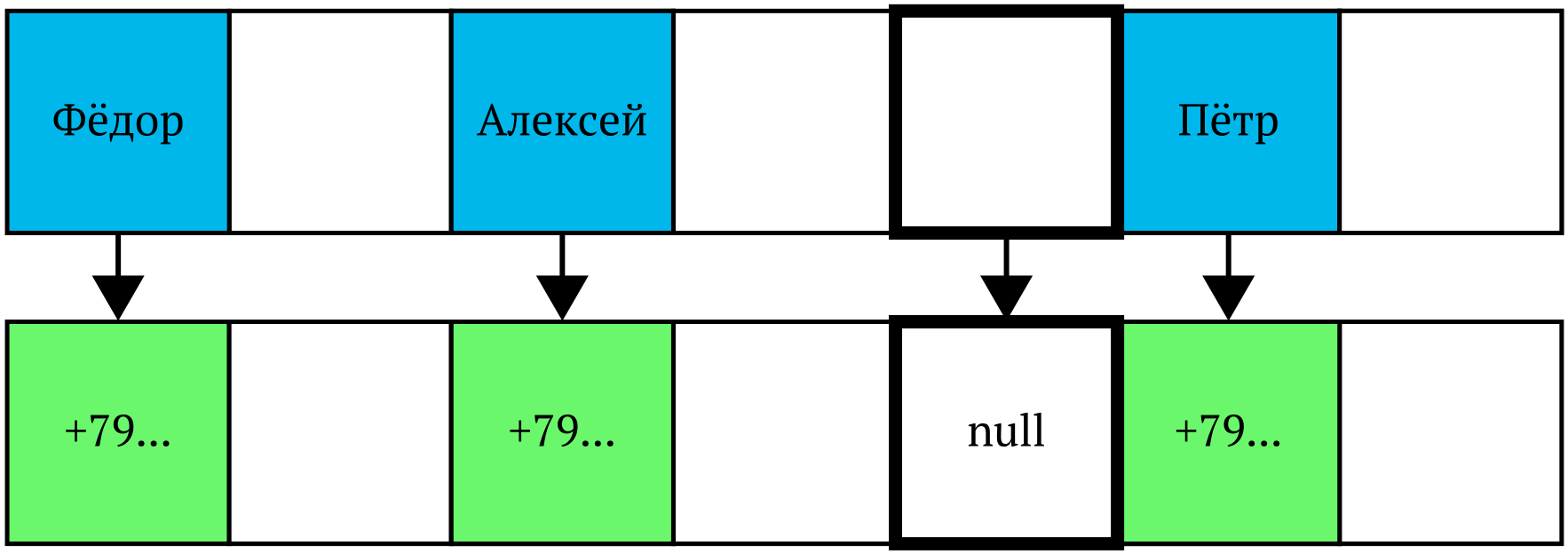
- неизменчивость и постоянство

Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method **must *consistently*** return the **same integer**, provided no information used in equals comparisons on the object is modified.

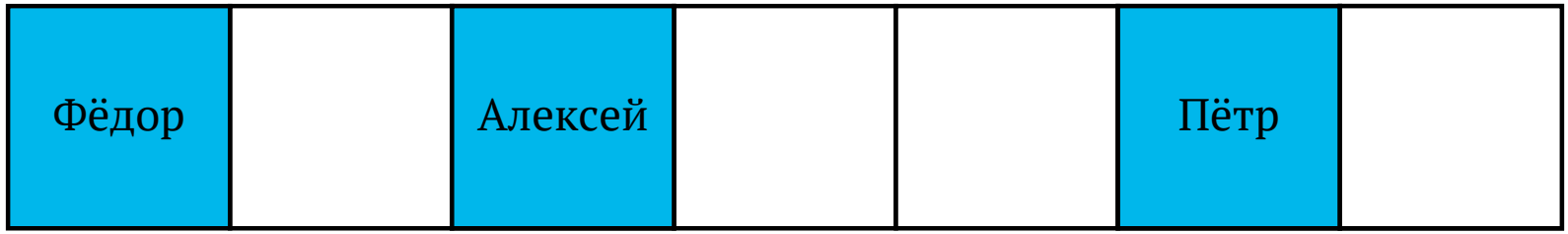
https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

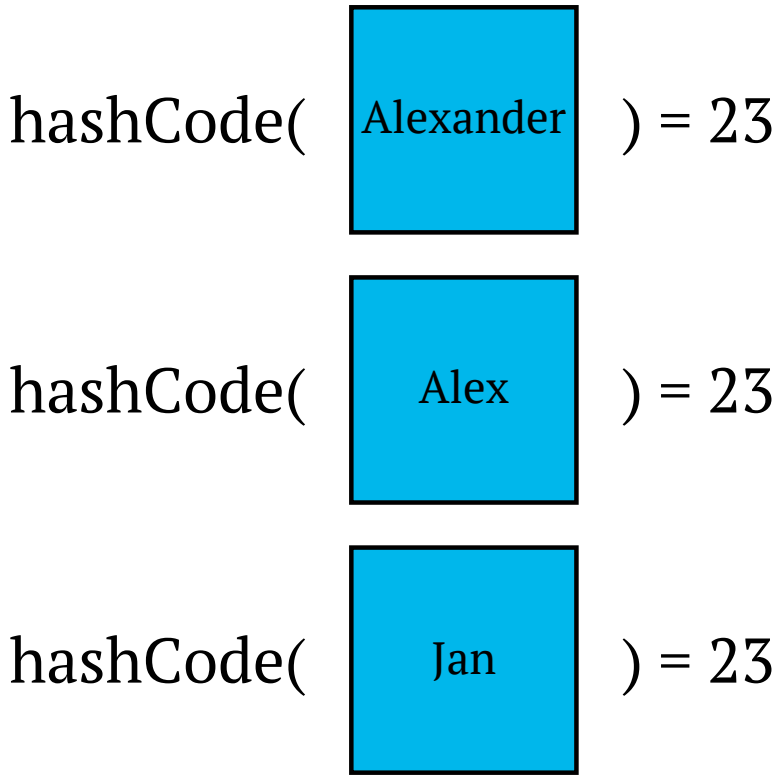
Нарушение контракта hashCode



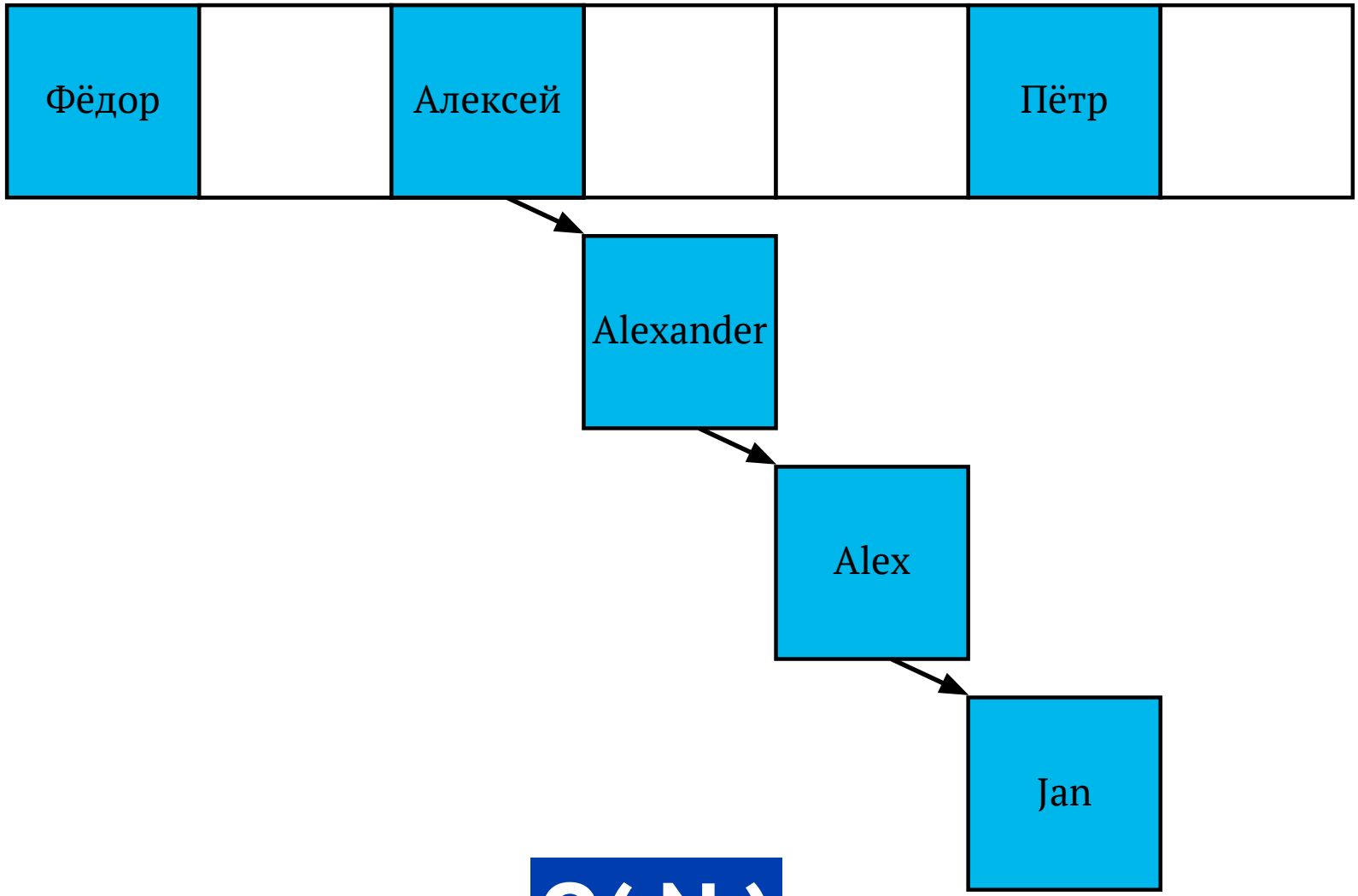


Коллизии


Фёдор
Алексей
Пётр

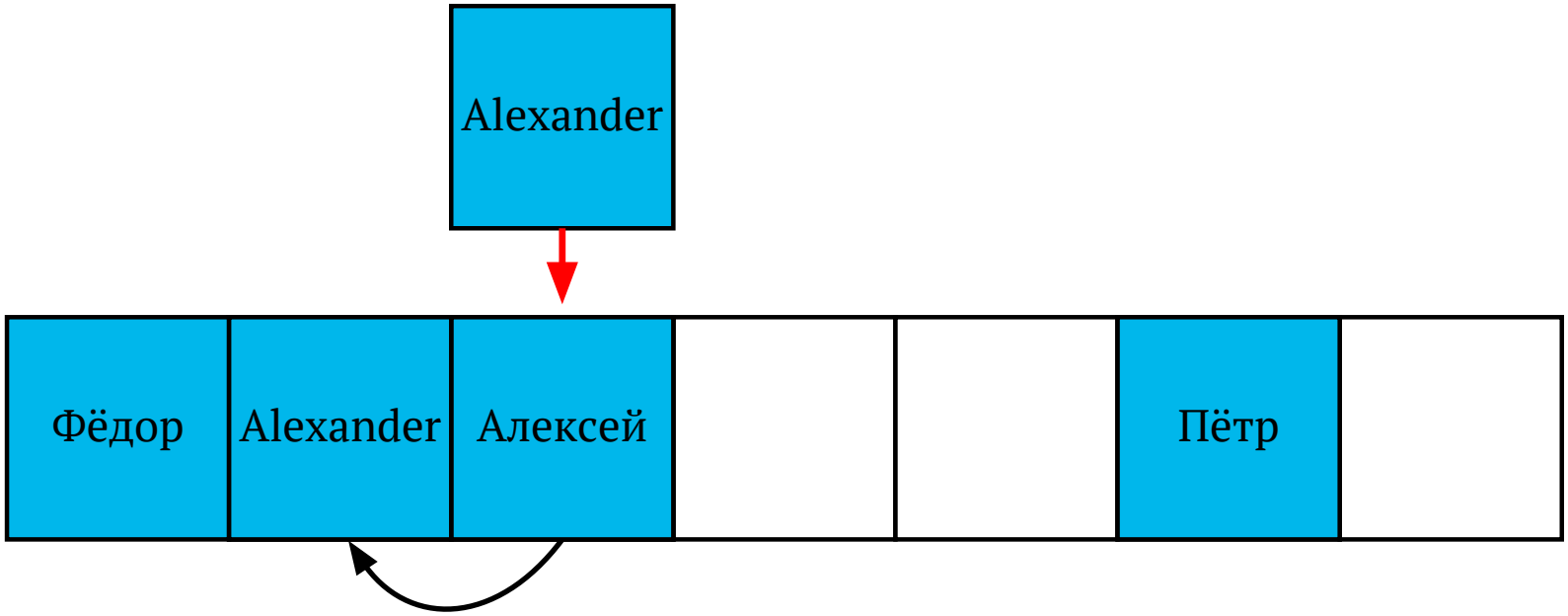

hashCode(Alexander) = 23
hashCode(Alex) = 23
hashCode(Jan) = 23

Chaining

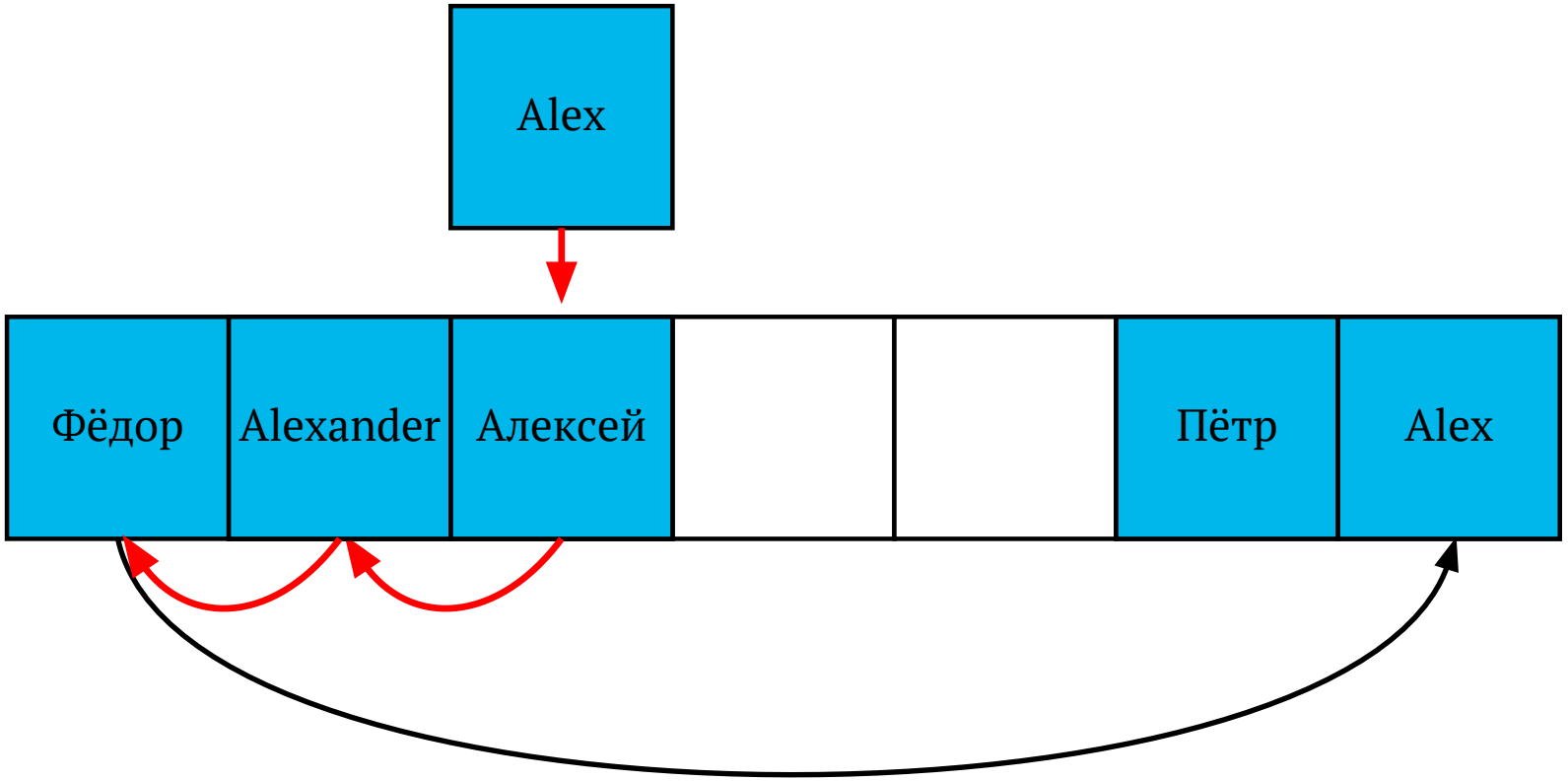


СЛОЖНОСТЬ → **O(N)**

Открытая адресация



Открытая адресация



сложность → **O(N)**

hashCode - функция от содержимого объекта

```
String s1 = new String("java");
String s2 = new String("java");

assert s1.hashCode() == s2.hashCode(); // true

Integer i1 = new Integer(42);
Integer i2 = new Integer(42);

assert i1.hashCode() == i2.hashCode(); // true
```

Полиномиальный hashCode

$$hashCode = \sum_{k=0}^{n} 31^{n-k} \cdot property_k$$

$$equals \Rightarrow$$
$$this.property_k == that.property_k,$$
$$\forall k \in [0, n]$$

String.hashCode()

```
public final class String {
    private final char value[];
    private int hash;

    public int hashCode() {
        int h = hash;
        if (h == 0 && value.length > 0) {
            char val[] = value;

            for (int i = 0; i < value.length; i++)
                h = 31 * h + val[i];

            hash = h;
        }
        return h;
    }
}
```

String.hashCode - часть public API

```
public final class String {
   /**
    * Returns a hash code for this string. The hash code
    * for a String object is computed as
    *
    * s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]
    *
    * using int arithmetic, where s[i]
    * is the ith character of the string,
    * n is the length of the string,
    * and ^ indicates exponentiation.
    * (The hash value of the empty string is zero.)
    *
    * @return  a hash code value for this object.
    */
     public int hashCode() {          ...       }
}
```

java v.1.1.1 - String.hashCode()

```
public int hashCode() {
    int h = 0;
    int off = offset;
    char val[] = value;
    int len = count;

    if (len < 16) {
        for (int i = len ; i > 0; i--) {
            h = (h * 37) + val[off++];
        }
    } else {
        // only sample some characters
        int skip = len / 8;
        for (int i = len ; i > 0; i -= skip, off += skip) {
            h = (h * 39) + val[off];
        }
    }
    return h;
}
```

31 : Детективная история

- **1997 год (!!!)**: java bug #4045622
  - Все слова и словоформы в словаре Merriam-Webster 2 ред. (311_141 строк, ср. длина 10 символов).
  - Все строки в `/bin/*`, `/usr/bin/*`, `/usr/lib/*`, `/usr/ucb/*` и `/usr/openwin/bin/*` (66_304 строк, ср. длина 21 символов).
  - Список URL, собранных web-пауком за несколько часов (28_372 строк, ср. длина 49 символов).

полиномы **31**, **33**, **37** → мин. средняя длина коллизий

Но ведь можно подобрать...

```
assert "Aa".hashCode()
    == "BB".hashCode();
```

$$31 \cdot c_0 + c_1 =$$
$$= 31 \cdot (c_0 - 1) + (c_1 + 31)$$

username $\Longrightarrow$ **499_331**

вариантов коллизий

Java Microbenchmark Harness

http://openjdk.java.net/projects/code-tools/jmh/

JMH :: "username" коллизии :: benchmark

```
@State(Scope.Benchmark)
public class MapPerfTest {

   Map map;
   String[] keys;

   @Setup
   public void setup() { ..... }

   @Benchmark @Threads( 1 )
   public Map fillMap() {
     for (String key : keys)
       map.put(key, key);

     return map;
   }

}
```

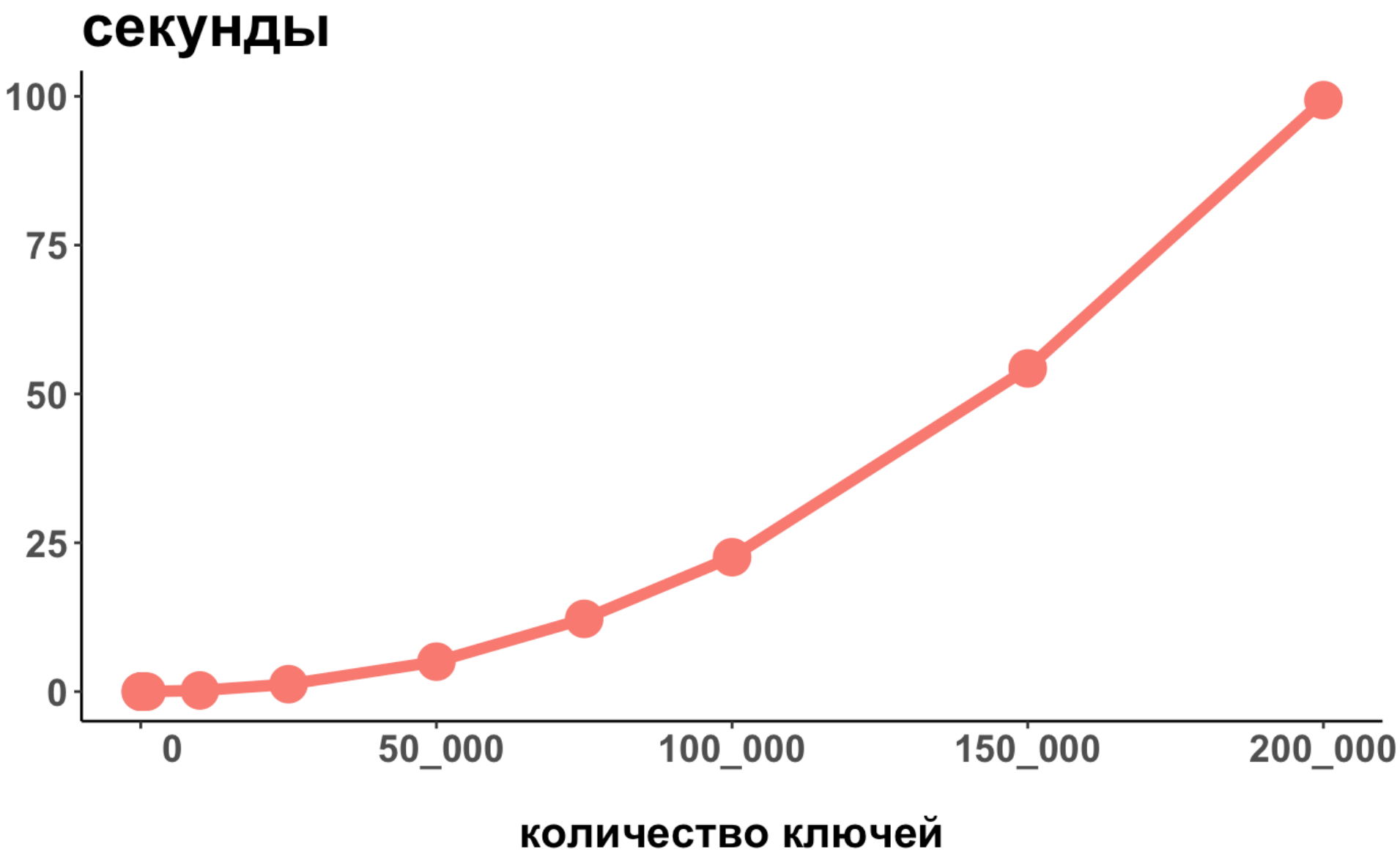
JMH :: "username" коллизии :: benchmark

```
@BenchmarkMode(Mode.AverageTime)
@Warmup(iterations = 5, time = 5, timeUnit = SECONDS)
@Measurement(iterations = 5, time = 5, timeUnit = SECONDS)
@State(Scope.Benchmark)
public class MapPerfTest {

   Map map;
   String[] keys;

   @Setup
   public void setup() { ... }

   @Benchmark @Threads( 1 )
   public Map fillMap() {
      // ...
   }
}
```

JMH :: "username" коллизии :: benchmark

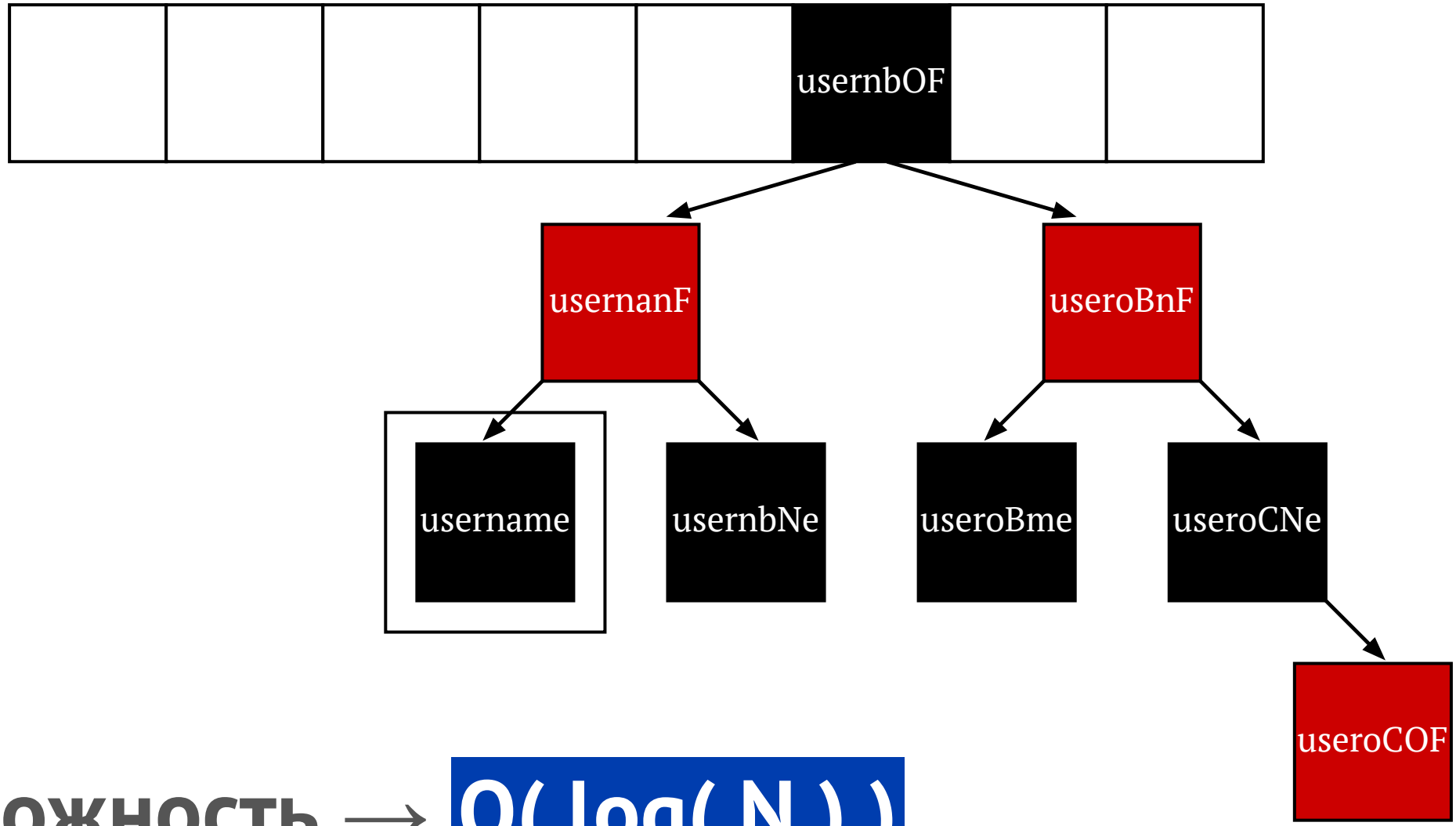
```
@State(Scope.Benchmark)
public class MapPerfTest {

   @Param( {"1", "1000", "10000", "100000", "200000"})
   int size;
   Map map;
   String[] keys;

   @Setup
   public void setup() {
     map = new HashMap<String, String>( size );
     keys = loadUsernameCollisionsFromFile( size );
   }

   @Benchmark @Threads( 1 )
   public Map fillMap() { ... }

}
```

"username" коллизии



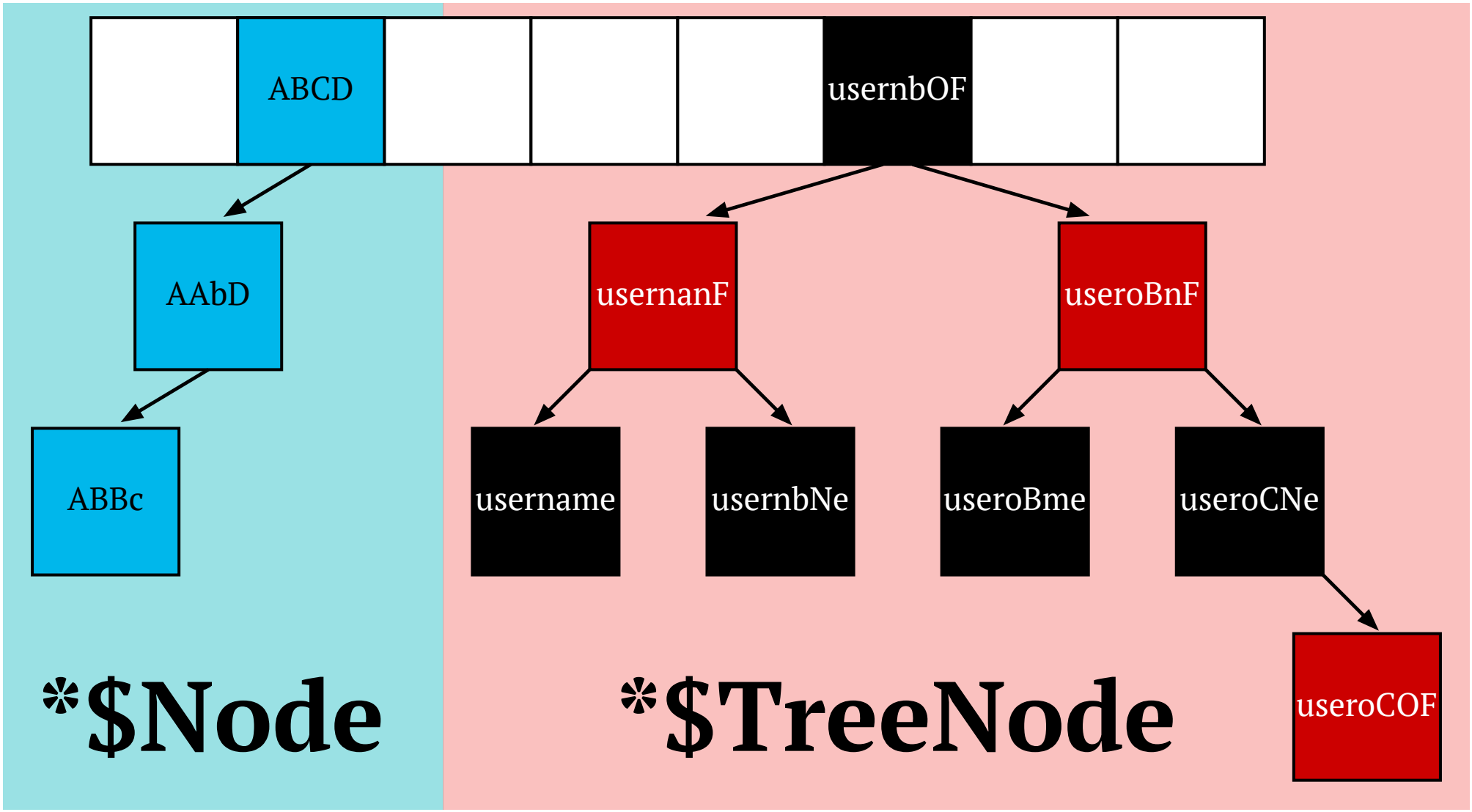
ещё одна функция нужна

Дополнительная функция: compareTo

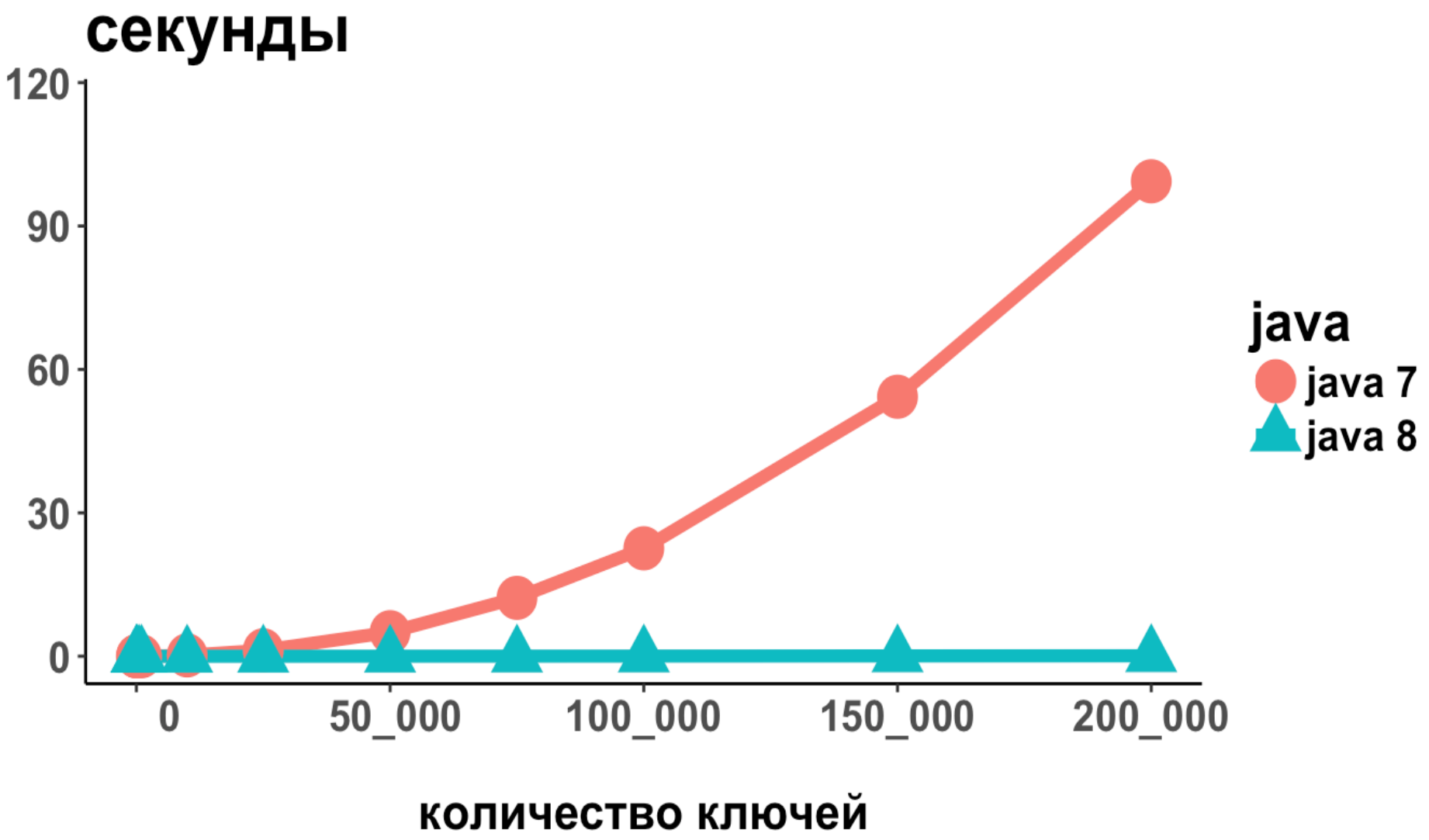


сложность → **O(log(N))**

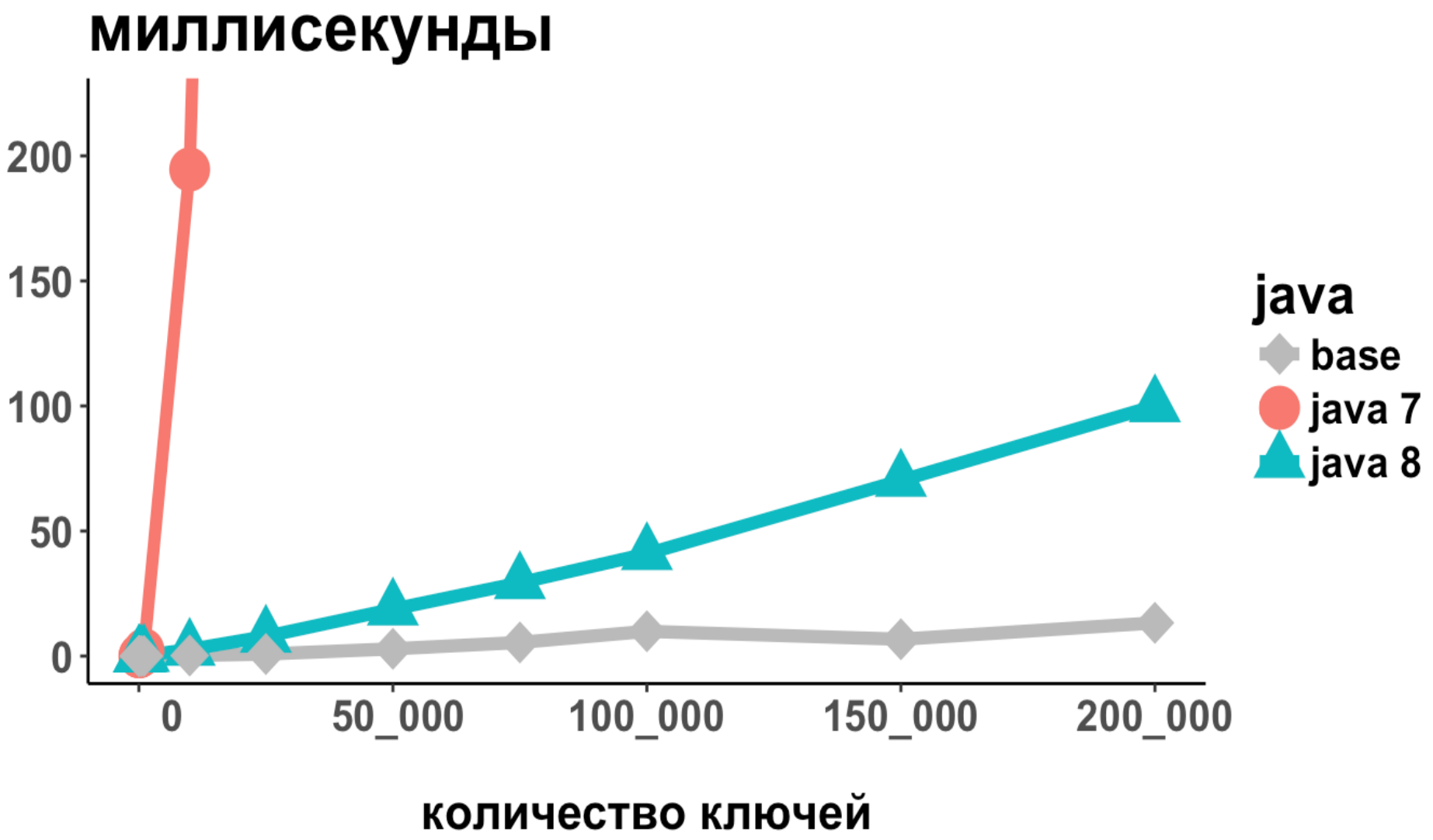
Chaining и Красно-Чёрное-Дерево



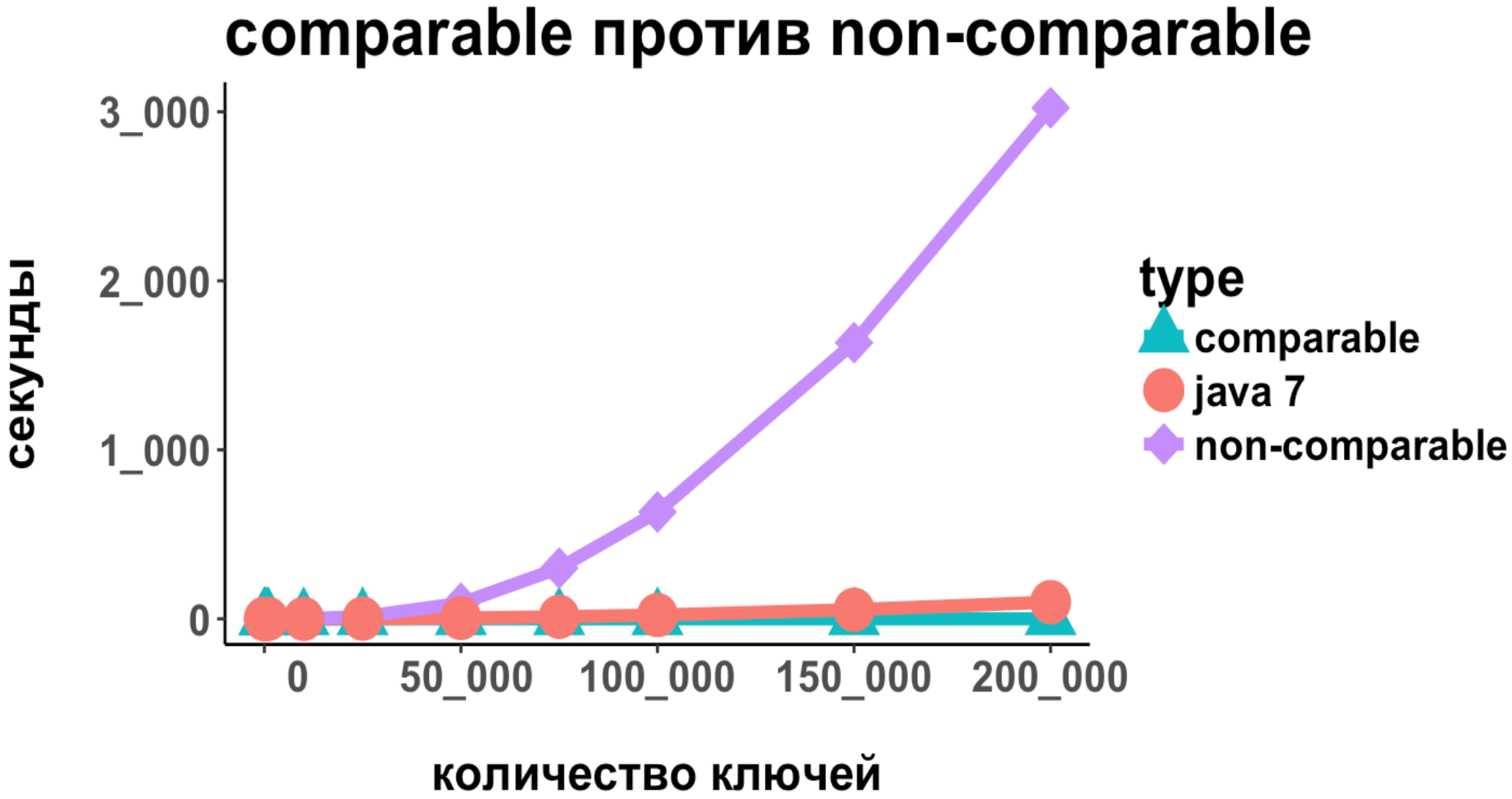
"username" коллизии



"username" коллизии :: 500x zoom-in



"username" коллизии :: Comparable



Object.hashCode
утёкшая абстракция

```
public int hashCode(){
   return &this;
}
```

Urban Legend

Urban Legend: Первоисточник

java.lang.Object

```
public int hashCode()
```

This is typically implemented by converting the **internal address of the object** into an integer.

https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

Адрес объекта ?

```
package java.lang;

public class Object {

    //  ..............

    public native int hashCode();
}
```

native method


ОПЯТЬ
UNSAFE ?

Java Object Layout

```
Object theObject = new Object();

VirtualMachine vm = VM.current();

long address = vm.addressOf( theObject );

long size = vm.sizeOf( theObject );
```

http://openjdk.java.net/projects/code-tools/jol/

JOL : sizeOf(new Pair())

```
class Pair<F, S> {
   F first;
   S second;
}
```

```
class Pair2<F, S> {
   F first;
   S second;
   int hashCode;
}
```

-XX:+UseCompressedOops и heap < 32Gb

-XX:+UseCompressedOops

24 b **24 b**

```
class Pair<F, S> {
  F first;
  S second;
}
```

```
class Pair2<F, S> {
  F first;
  S second;
  int hashCode;
}
```

-XX:+UseCompressedOops и heap < 32Gb

-XX:+UseCompressedOops

24 b **24 b**

```
class Pair<F, S> {
  F first;
  S second;
}
```

```
class Pair2<F, S> {
  F first;
  S second;
  int hashCode;
}
```

Pair object internals:

```
 OFFSET  SIZE   TYPE DESCRIPTION
      0    12        (object header)
     12     4 Object Pair.first
     16     4 Object Pair.second
     20     4        (loss due to the next
                      object alignment)
```

-XX:-UseCompressedOops или heap ⩾ 32Gb

-XX:-UseCompressedOops

32 b **40 b**

```
class Pair<F, S> {
  F first;
  S second;
}
```

```
class Pair2<F, S> {
  F first;
  S second;
  int hashCode;
}
```

-XX:-UseCompressedOops или heap ⩾ 32Gb

-XX:-UseCompressedOops

32 b **40 b**

```
class Pair<F, S> {
  F first;
  S second;
}
```

```
class Pair2<F, S> {
  F first;
  S second;
  int hashCode;
}
```

Pair2 object internals:

```
 OFFSET  SIZE   TYPE DESCRIPTION
      0    16        (object header)
     16     4    int Pair2.hashCode
     20     4        (alignment/padding gap)
     24     8 Object Pair2.first
     32     8 Object Pair2.second
```

Адрес и hashCode

class	`java.lang.Object`
address	`0x 07 6B A3 6D B8`
hashCode	`0x 6B A3 6D B8`
size	`16`

```
-XX:[redacted]
```

Следим за адресом объекта

```
final Object theObject = new Object();
final long initialAddress = getAddress( theObject );

List gcKeeper = new ArrayList();
gcKeeper.add( theObject );

long currentAddress = initialAddress;
while (initialAddress == currentAddress) {
    Object o = new Object();

    gcKeeper.add( o );

    currentAddress = getAddress( theObject );
}
```

```
-Xms256m -Xmx256m -XX:+UseSerialGC
```

Следим за адресом объекта

```
final Object theObject = new Object();
final long initialAddress = getAddress( theObject );

List gcKeeper = new ArrayList();
gcKeeper.add( theObject );

long currentAddress = initialAddress;
while (initialAddress == currentAddress) {
    Object o = new Object();

    gcKeeper.add( o );

    currentAddress = getAddress( theObject );
}
```

```
-Xms256m -Xmx256m -XX:+UseSerialGC
```

Следим за адресом объекта

```
final Object theObject = new Object();
final long initialAddress = getAddress( theObject );

List gcKeeper = new ArrayList();
gcKeeper.add( theObject );

long currentAddress = initialAddress;
while (initialAddress == currentAddress) {
    Object o = new Object();

    gcKeeper.add( o );

    currentAddress = getAddress( theObject );
}
```

```
-Xms256m -Xmx256m -XX:+UseSerialGC
```

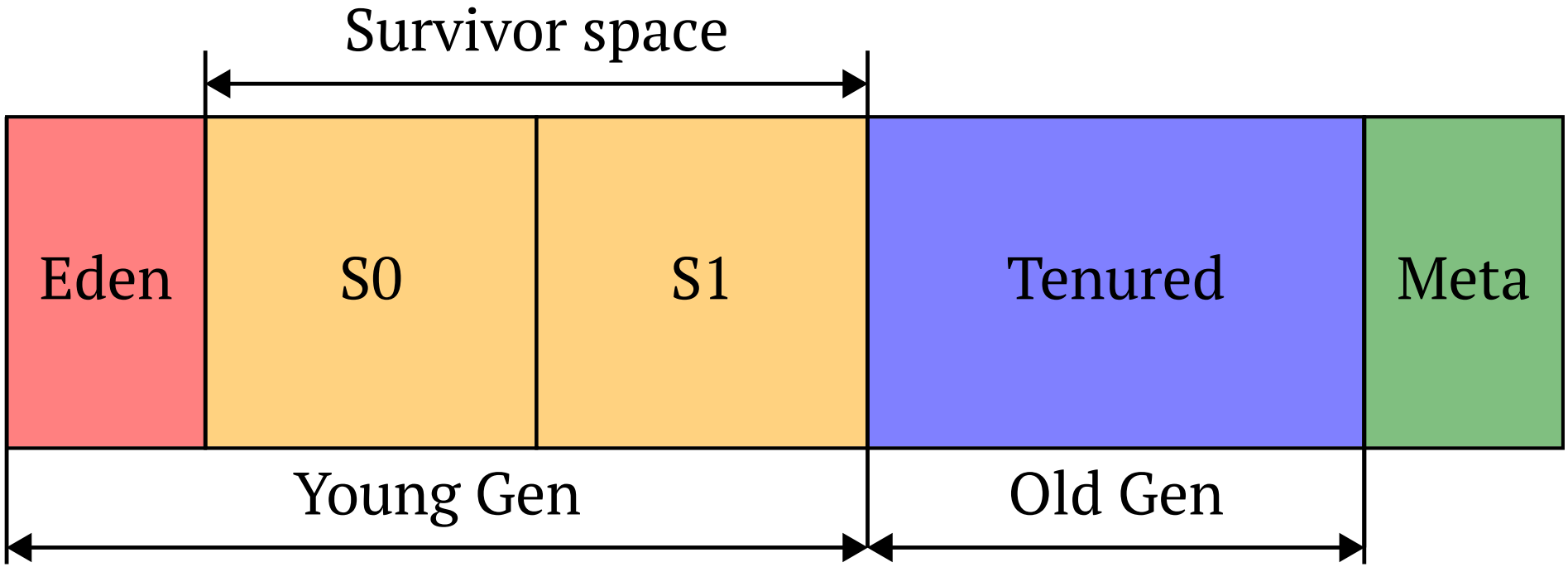
Следим за адресом объекта

```
final Object theObject = new Object();
final long initialAddress = getAddress( theObject );

List gcKeeper = new ArrayList();
gcKeeper.add( theObject );

long currentAddress = initialAddress;
while (initialAddress == currentAddress) {
    Object o = new Object();

    gcKeeper.add(o);

    currentAddress = getAddress( theObject );
}
```

```
-Xms256m -Xmx256m -XX:+UseSerialGC
```

demo

GC: поколения



GC: Serial / Parallel / CMS

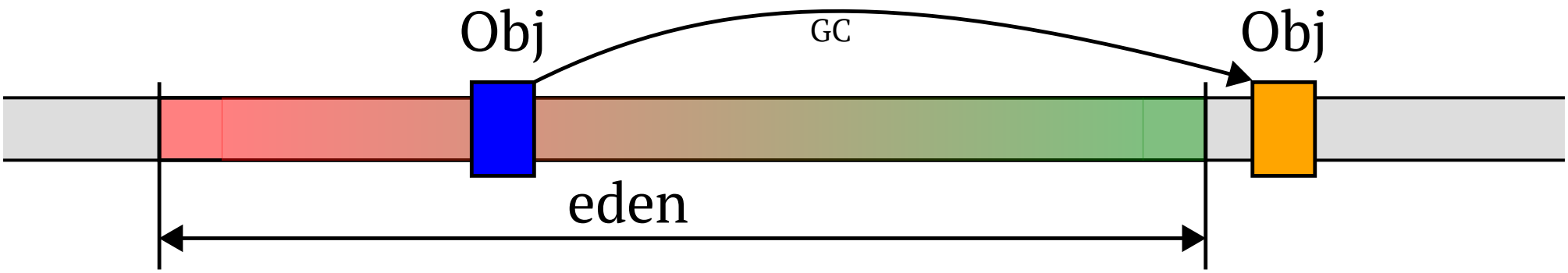
Следим за адресом объекта вместе с GC

```
final Object theObject = new Object();
final long initialAddress = getAddress( theObject );

List gcKeeper = new ArrayList();
gcKeeper.add( theObject );

long currentAddress = initialAddress;
while (initialAddress == currentAddress) {
    Object o = new Object();

    gcKeeper.add(o);

    currentAddress = getAddress( theObject );
}
```

-XX:+PrintGCDetails -Xmx256m -XX:+UseSerialGC

GC - не только сборка мусора

```
eden space 65536K, 2% used
[0x000000012a700000,0x000000012a8db400,0x000000012e700000)
Объект перемещён: с 0x12A8109F0 -> 0x12E7200F8
```



0x12A700000	0x12A8109F0	0x12E700000	0x12E7200F8
начало Eden	старый адрес	конец Eden	новый адрес

Следим за hashcode

```
final Object theObject = new Object();
final long initialHashCode = theObject.hashCode();

List gcKeeper = new ArrayList();
gcKeeper.add(theObject);

long currentHashCode = initialHashCode;
while (initialHashCode == currentHashCode) {
    Object o = new Object();

    gcKeeper.add(o);

    currentHashCode = theObject.hashCode();
}
```

```
-Xms256m -Xmx256m -XX:+UseSerialGC
```

demo

Скрытое свойство

```
package java.lang;

public class Object {

    // другие методы

    public native int hashCode();

}
```

Дамп объекта

class	`java.lang.Object`
address	`0x 07 6B A3 6D B8`
hashCode	`0x 6B A3 6D B8`
size	`16`
dump	`01 B8 6D A3 6B 00 00 00`
	`E5 01 00 F8 00 00 00 00`

Intel X86: Little Endian

Сколько влезет в кучу ?

`-Xmx256m`

```
long freeMemory =
   Runtime.getRuntime().freeMemory();
```

= **239_942_568**

```
freeMemory / vm.sizeOf( new Object() )
```

= **14_996_410**

Граница коллизий

```
eden space 65536K, 2% used
[0x000000012a700000, 0x000000012a8db400, 0x00000001
Объект перемещён: с 0x12A8109F0 -> 0x12E7200F8
```

```
65536K / vm.sizeOf( new Object() )
```

```
= 4_194_304
```

пространство hashCode

32bit → 22bit

= **4_194_304**

уникальных значений

hashCode → address →
→ memory allocation

Memory allocation

```
public interface Allocator {
    long malloc(long size);
}

class SimpleAllocator implements Allocator {
    private long memoryPointer;

    @Override
    public long malloc(long size) {
        long old = memoryPointer;
        memoryPointer += size;
        return old;
    }
}
```

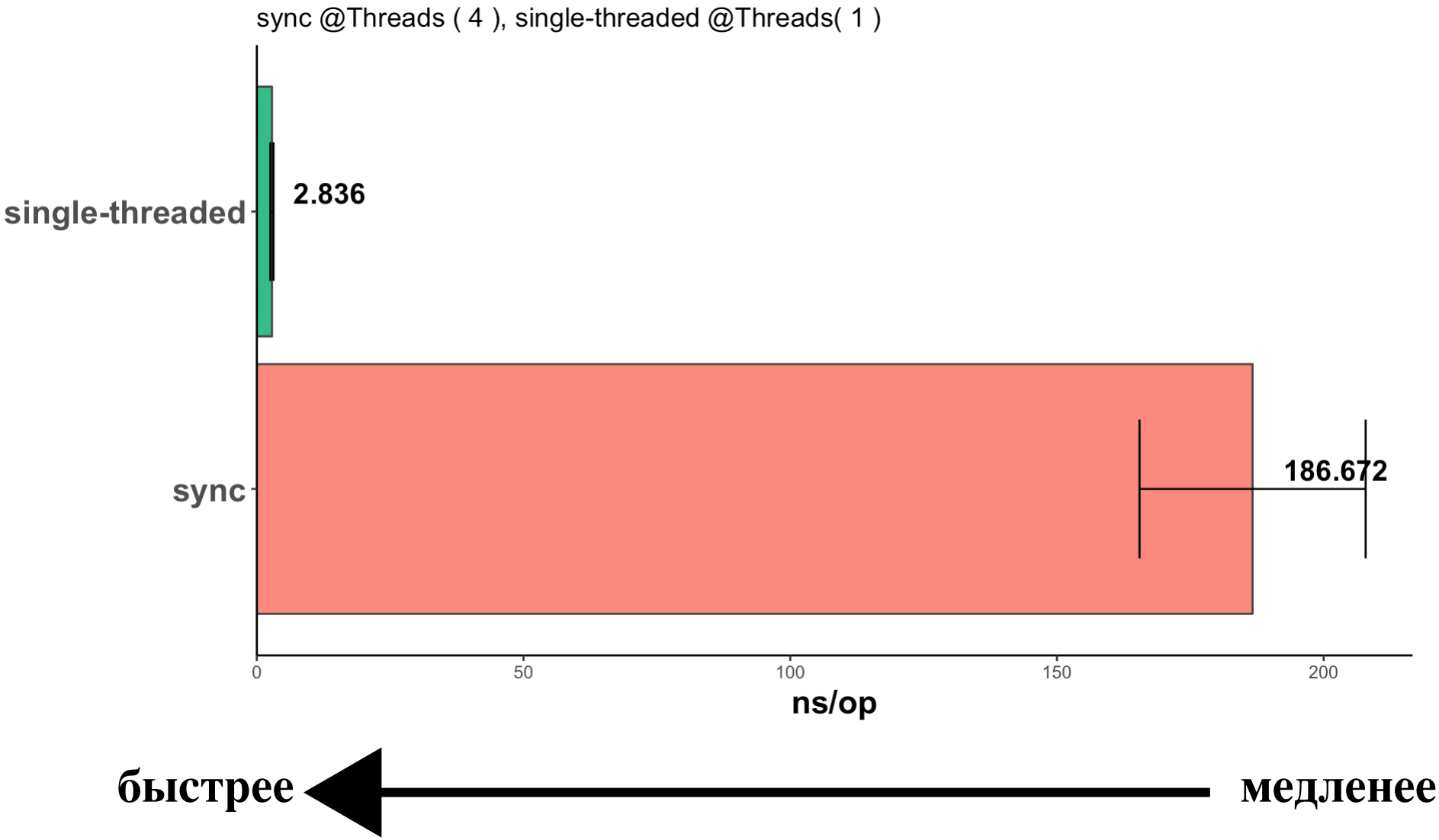
Исправим malloc: SyncAllocator

```
class SyncAllocator implements Allocator {
    private long memoryPointer;

    @Override
    public synchronized long malloc(long size) {
        long old = memoryPointer;
        memoryPointer += size;
        return old;
    }
}
```

SyncAllocator Performance Benchmark

```
@Benchmark @Threads( 4 )
public long syncAllocator() {
    return syncAllocator.malloc( 16 );
}
```

SyncAllocator Performance Benchmark



Можно ли лучше ?

Compare-and-Set

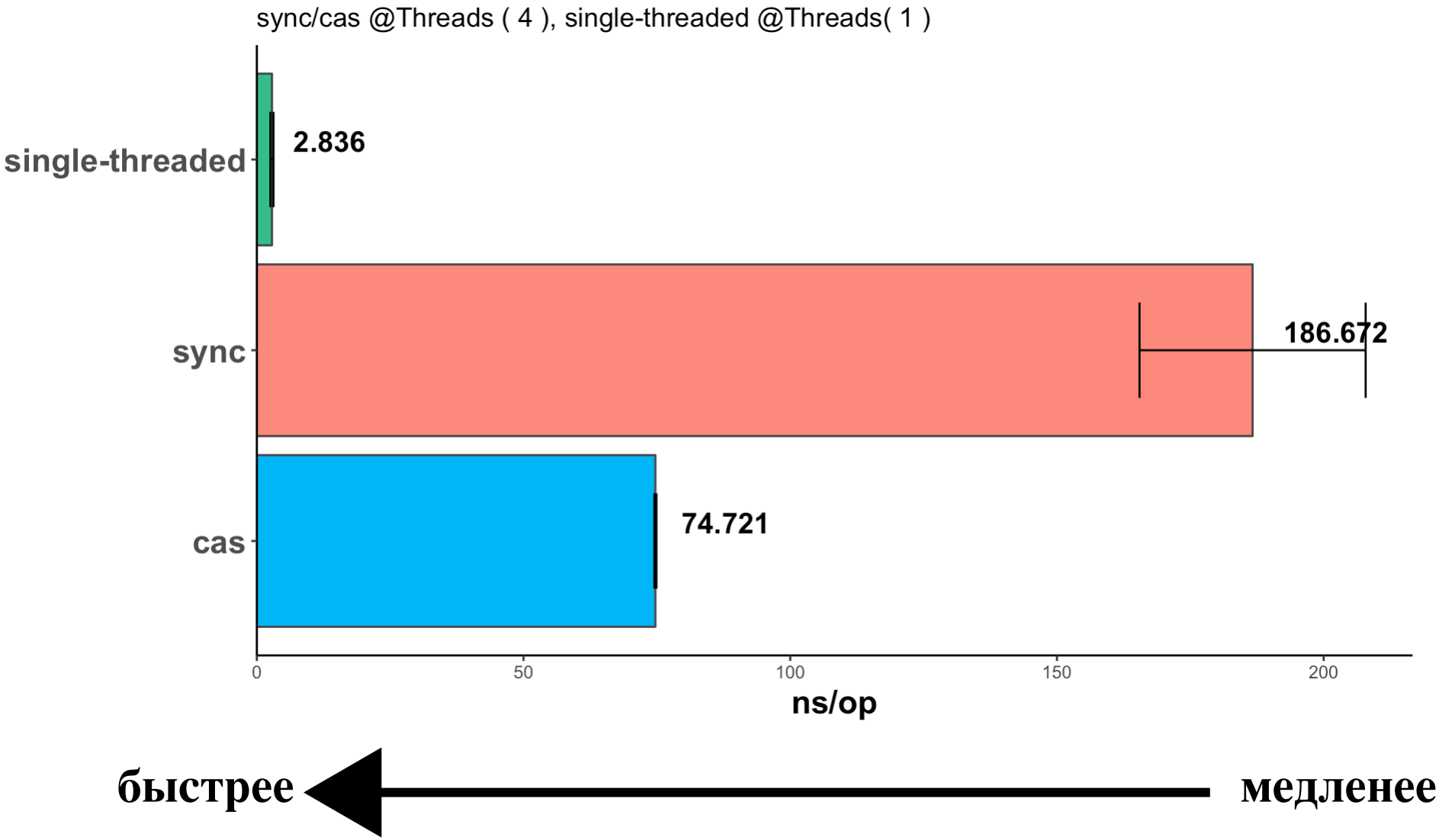
CAS Allocator

```
class CASAllocator implements Allocator {
    private final AtomicLong memoryPointer =
        new AtomicLong();

    @Override
    public long malloc(long size) {
        return memoryPointer.getAndAdd( size );
    }
}
```

Allocators Performance Benchmark

```
@Benchmark @Threads( 4 )
public long casAllocator() {
    return casAllocator.malloc( 16 );
}
```

Allocators Performance Benchmark

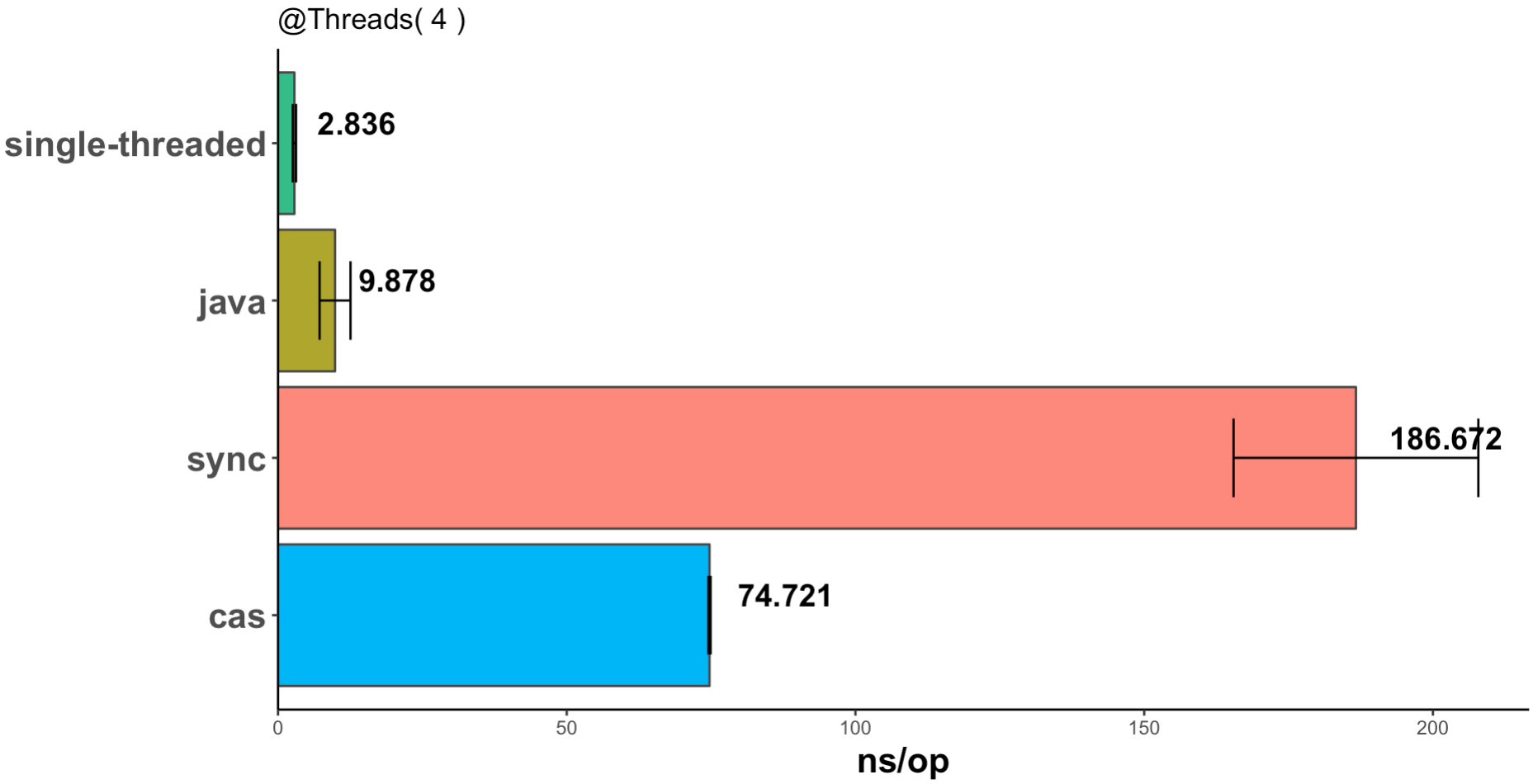


быстрее ← **медленее**

Java Allocation

```
@Benchmark @Threads( 4 )
public Object javaAllocation() {
    return new Object();
}
```

Java Allocation

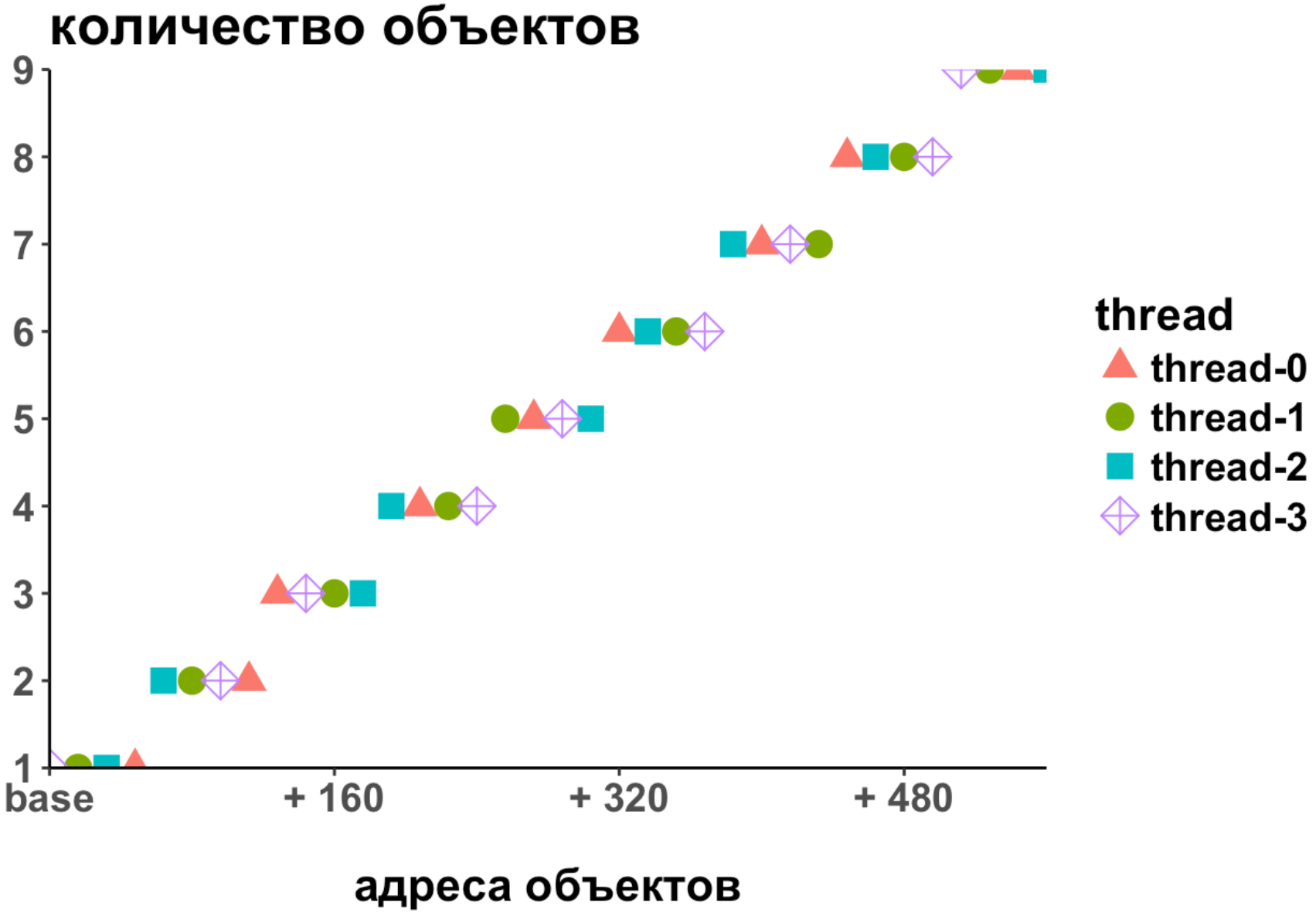


быстрее ← **медленее**

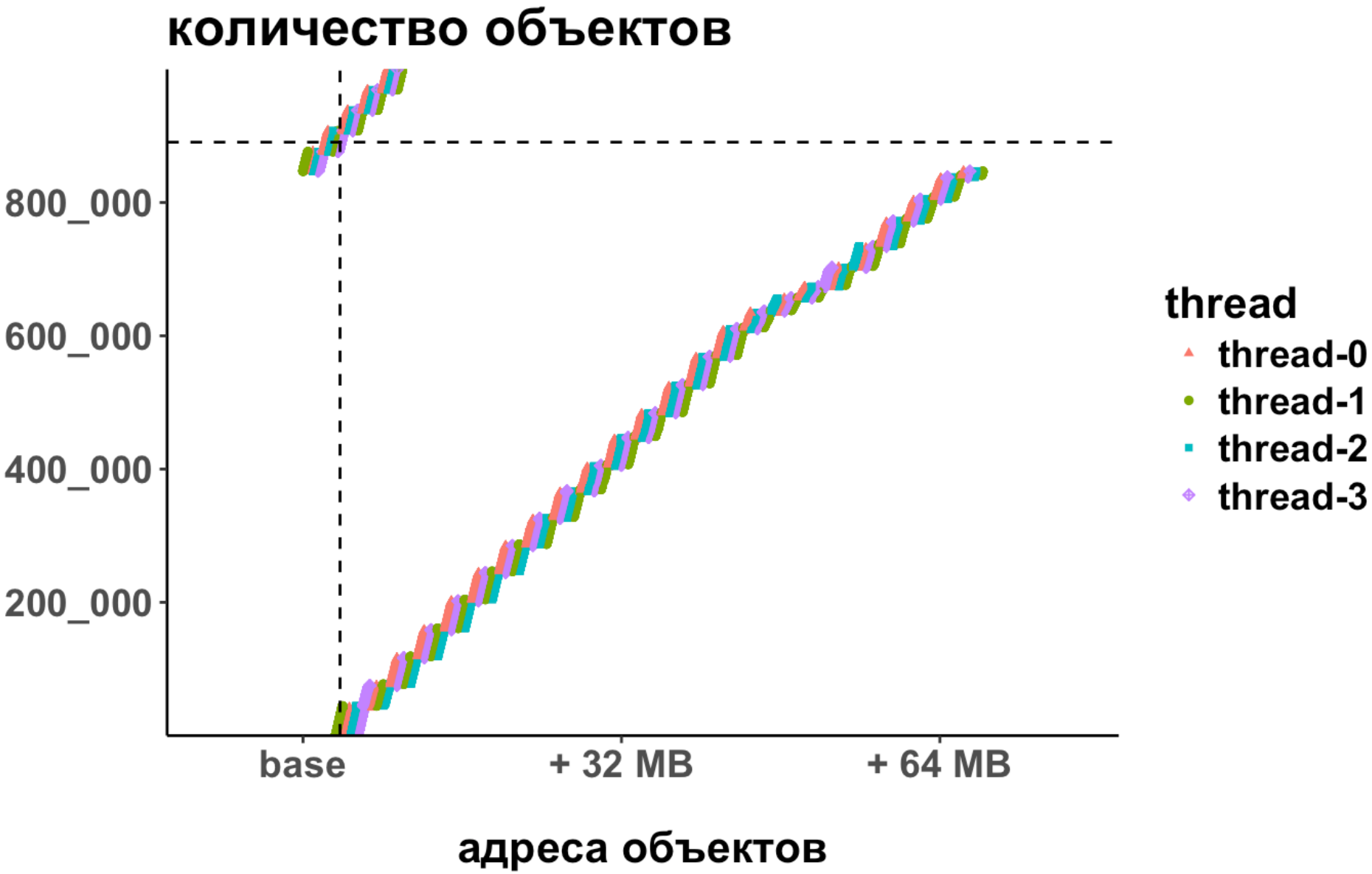
Можно ли ещё лучше ?

Распеделение hashCode по нитям

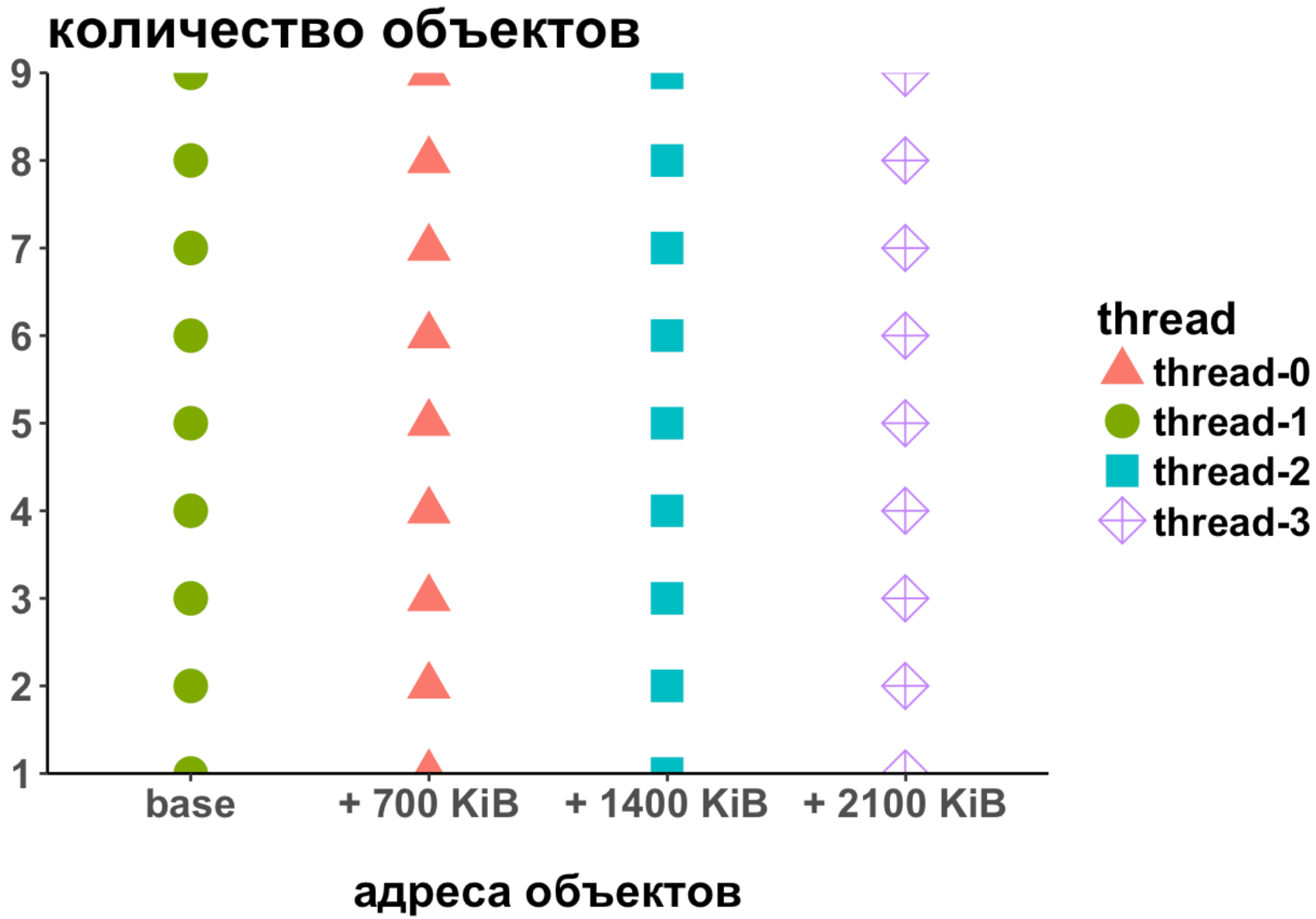
Распеделение hashCode : Ожидания



Распределение hashCode по 4 нитям



Хорошая новость



Даёшь БОЛЬШИЕ куски памяти !

```
public class TLABLikeAllocator implements Allocator {
  private static final long SIZE = 1024L * 1024L;

  private final AtomicLong memoryPointer = new AtomicLong();
  private final ThreadLocal<AddressHolder> threadLocal =
       ThreadLocal.withInitial(() -> new AddressHolder());

  public long malloc( long size ) {
    AddressHolder addressHolder = threadLocal.get();
    while( true ) {
      if (addressHolder.value + size <= addressHolder.maxValue) {
          long old = addressHolder.value;
          addressHolder.value += size;
          return old;
      }

      long value = memoryPointer.getAndAdd( SIZE );
      addressHolder.value = value;
      addressHolder.maxValue = value + SIZE;
    }
  }
}
```

Даёшь БОЛЬШИЕ куски памяти !

```
public class TLABLikeAllocator implements Allocator {
  private static final long SIZE = 1024L * 1024L;

  private final AtomicLong memoryPointer = new AtomicLong();
  private final ThreadLocal<AddressHolder> threadLocal =
       ThreadLocal.withInitial(() -> new AddressHolder());

  public long malloc( long size ) {
    AddressHolder addressHolder = threadLocal.get();
    while( true ) {
      if (addressHolder.value + size <= addressHolder.maxValue) {
          long old = addressHolder.value;
          addressHolder.value += size;
          return old;
      }

      long value = memoryPointer.getAndAdd( SIZE );
      addressHolder.value = value;
      addressHolder.maxValue = value + SIZE;
    }
  }
}
```

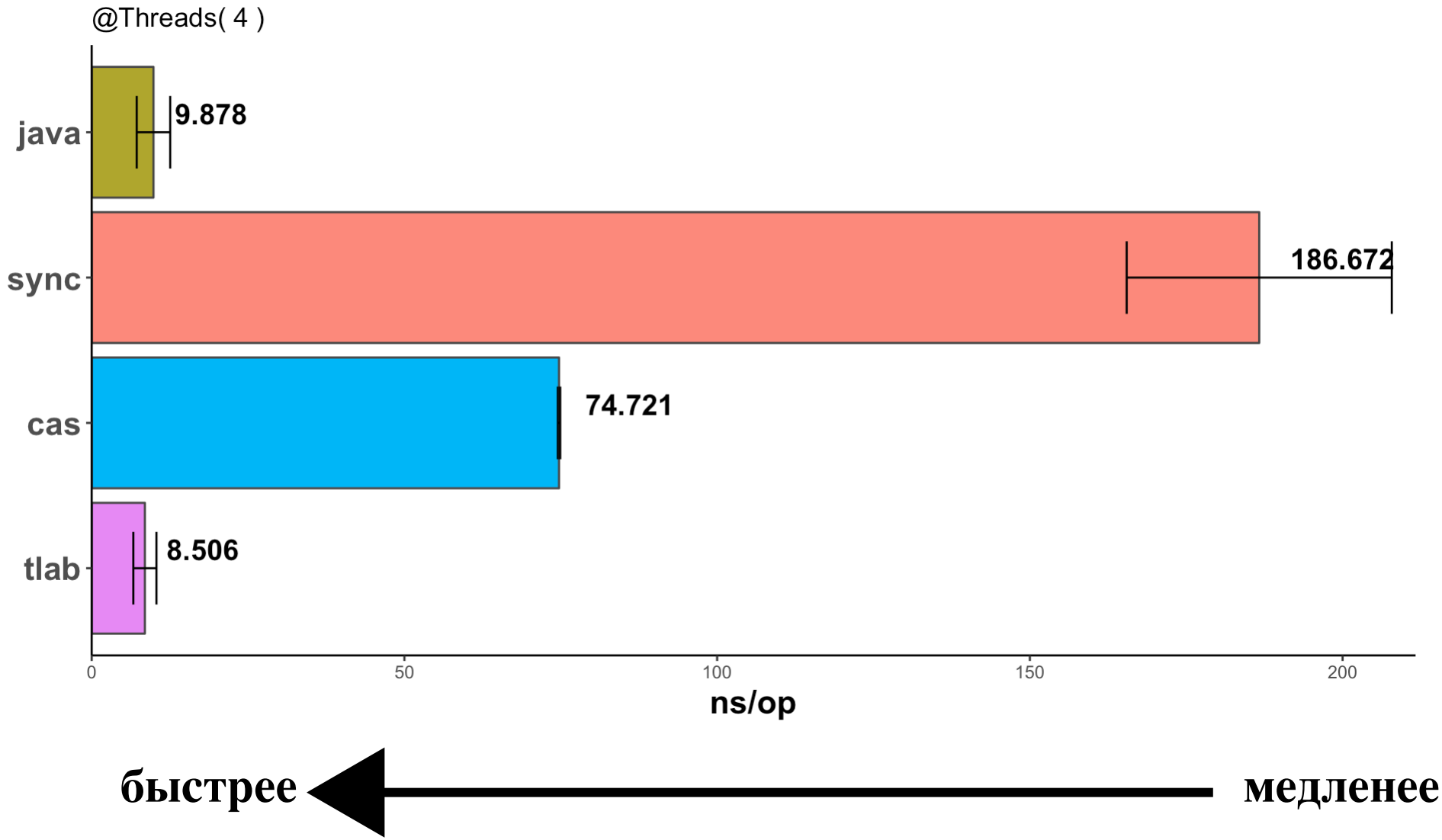
Даёшь БОЛЬШИЕ куски памяти !

```
public class TLABLikeAllocator implements Allocator {
  private static final long SIZE = 1024L * 1024L;

  private final AtomicLong memoryPointer = new AtomicLong();
  private final ThreadLocal<AddressHolder> threadLocal =
        ThreadLocal.withInitial(() -> new AddressHolder());

  public long malloc( long size ) {
    AddressHolder addressHolder = threadLocal.get();
    while( true ) {
      if (addressHolder.value + size <= addressHolder.maxValue) {
        long old = addressHolder.value;
        addressHolder.value += size;
        return old;
      }

      long value = memoryPointer.getAndAdd( SIZE );
      addressHolder.value = value;
      addressHolder.maxValue = value + SIZE;
    }
  }
}
```

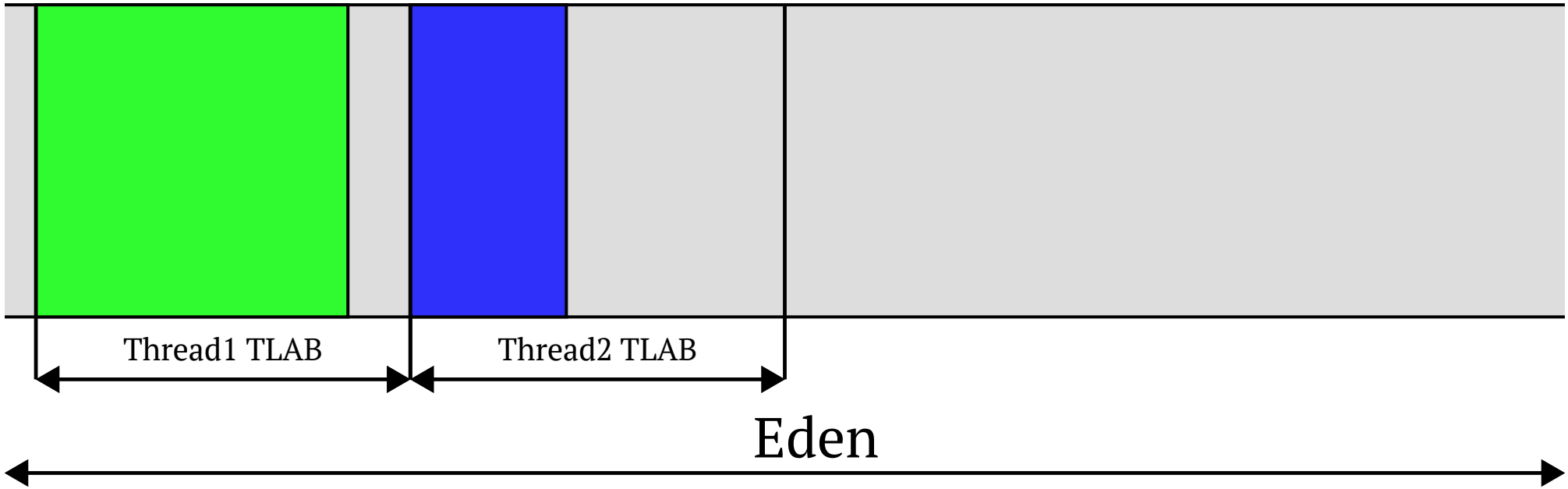
Allocators Performance Benchmark

```
@Benchmark @Threads( 4 )
public long tlabAllocator() {
    return tlabAllocator.malloc( 16 );
}
```

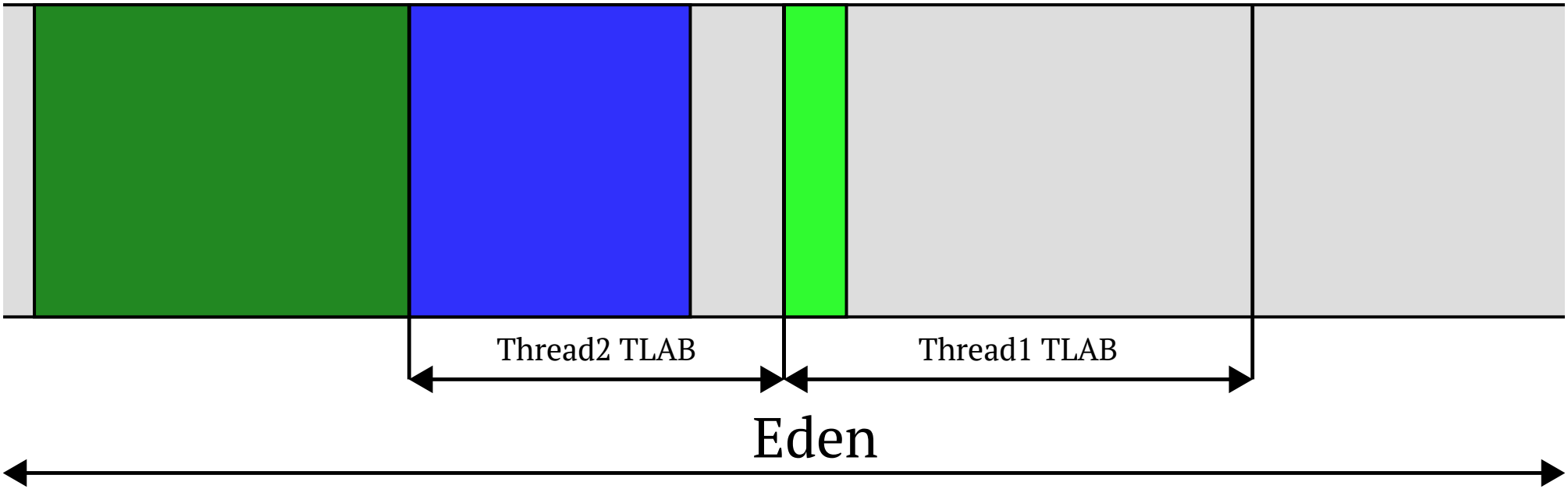
Allocators Performance Benchmark



Thread Local Allocation Buffer



Thread Local Allocation Buffer

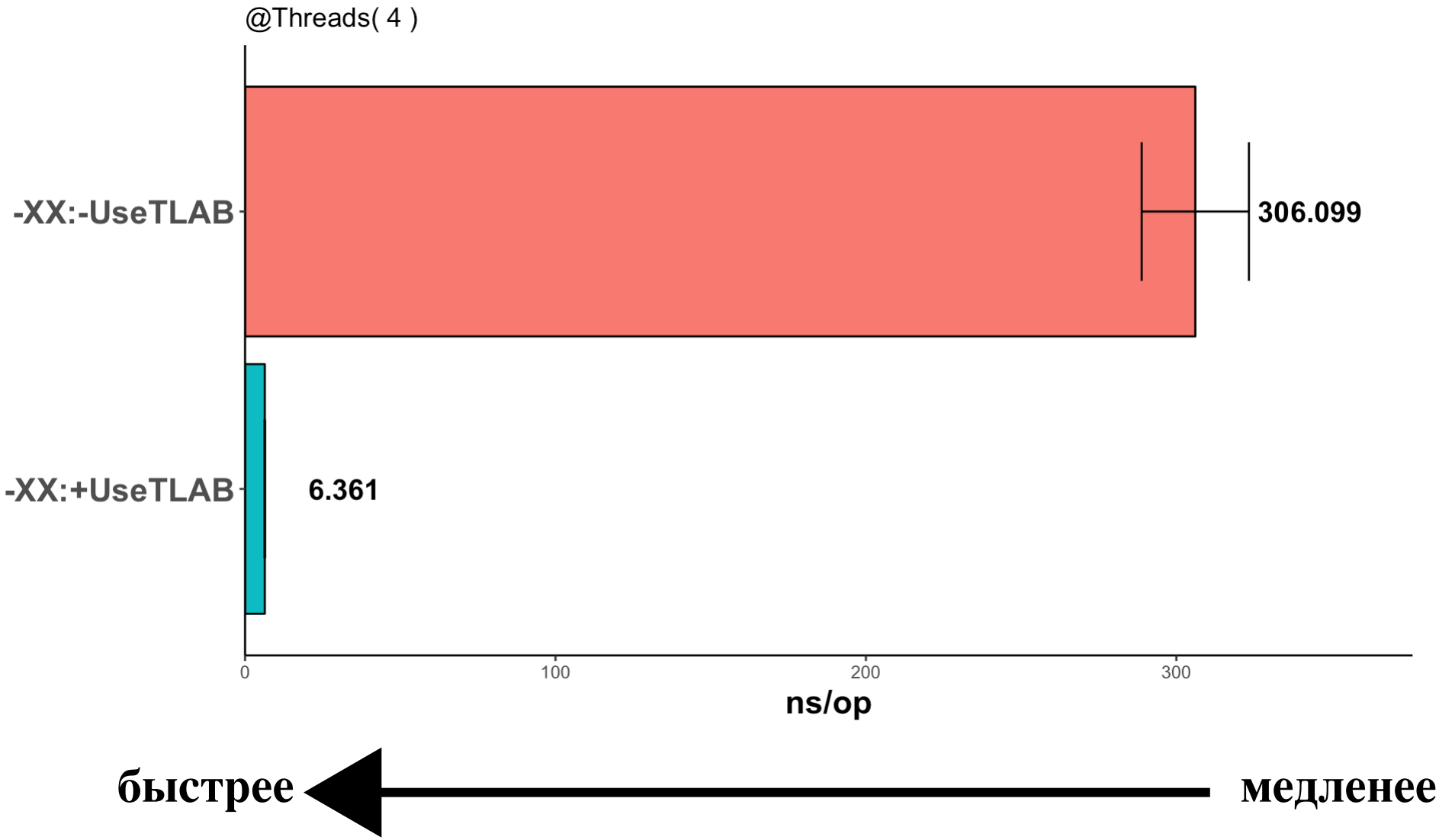


Стоимость TLAB

```
@Benchmark @Threads( 4 )
public Object allocate() {
    return new Object();
}
```

-XX:+UseTLAB vs **-XX:-UseTLAB**

Стоимость TLAB



32bit → 20bit

Может быть Random ?

Парадокс дней рождения

$$p_{uniq}(n) = \left(1 - \frac{1}{d}\right) \cdot \left(1 - \frac{2}{d}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{d}\right) =$$

$$= \frac{d}{d^n \cdot (d-n)!}$$

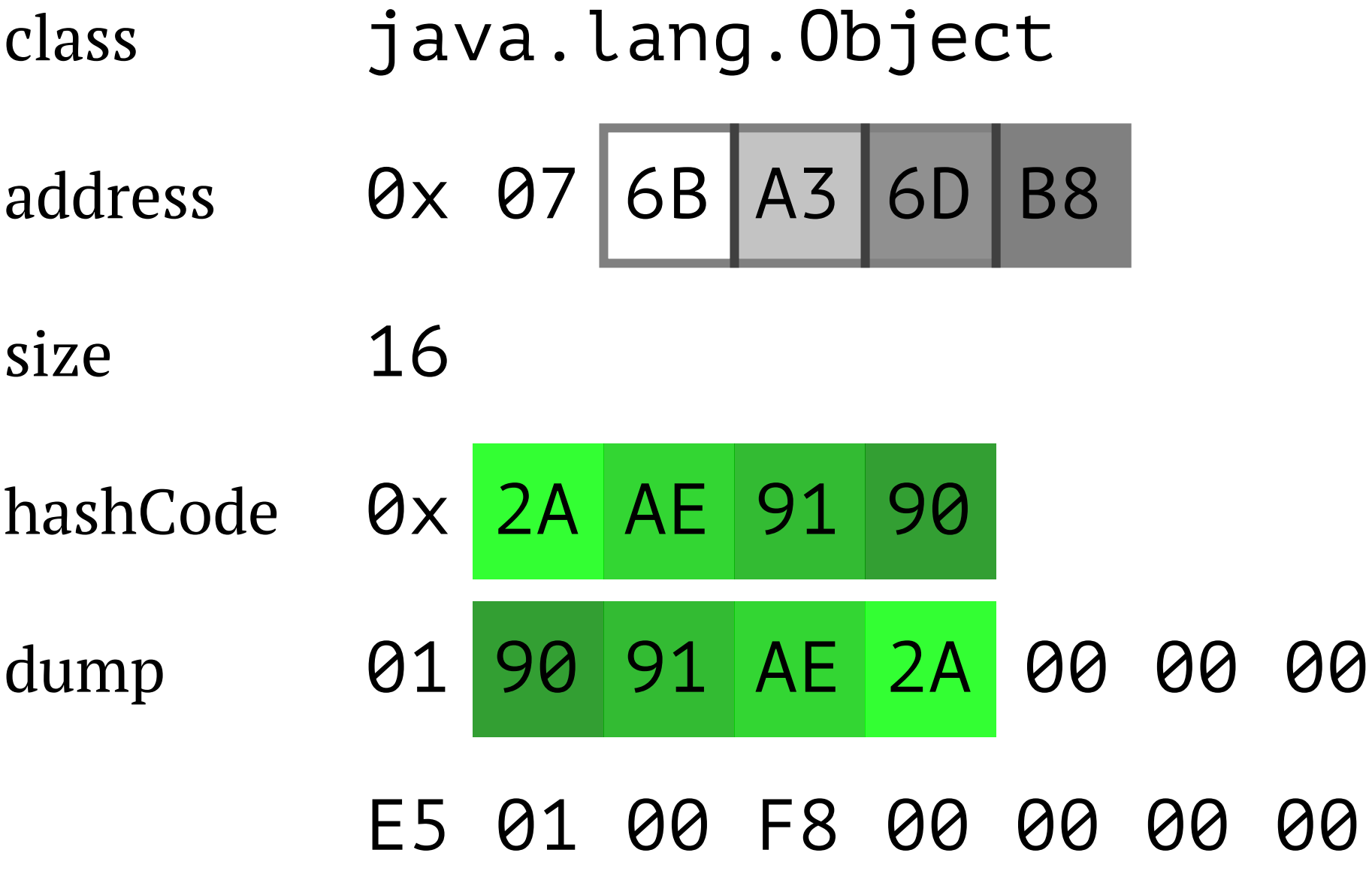
$$n \approx \sqrt{2d \cdot ln(\frac{1}{1 - p_{uniq}})}$$

при $d = 2^{32},\ p = 0.5 \Rightarrow n \approx 77162$

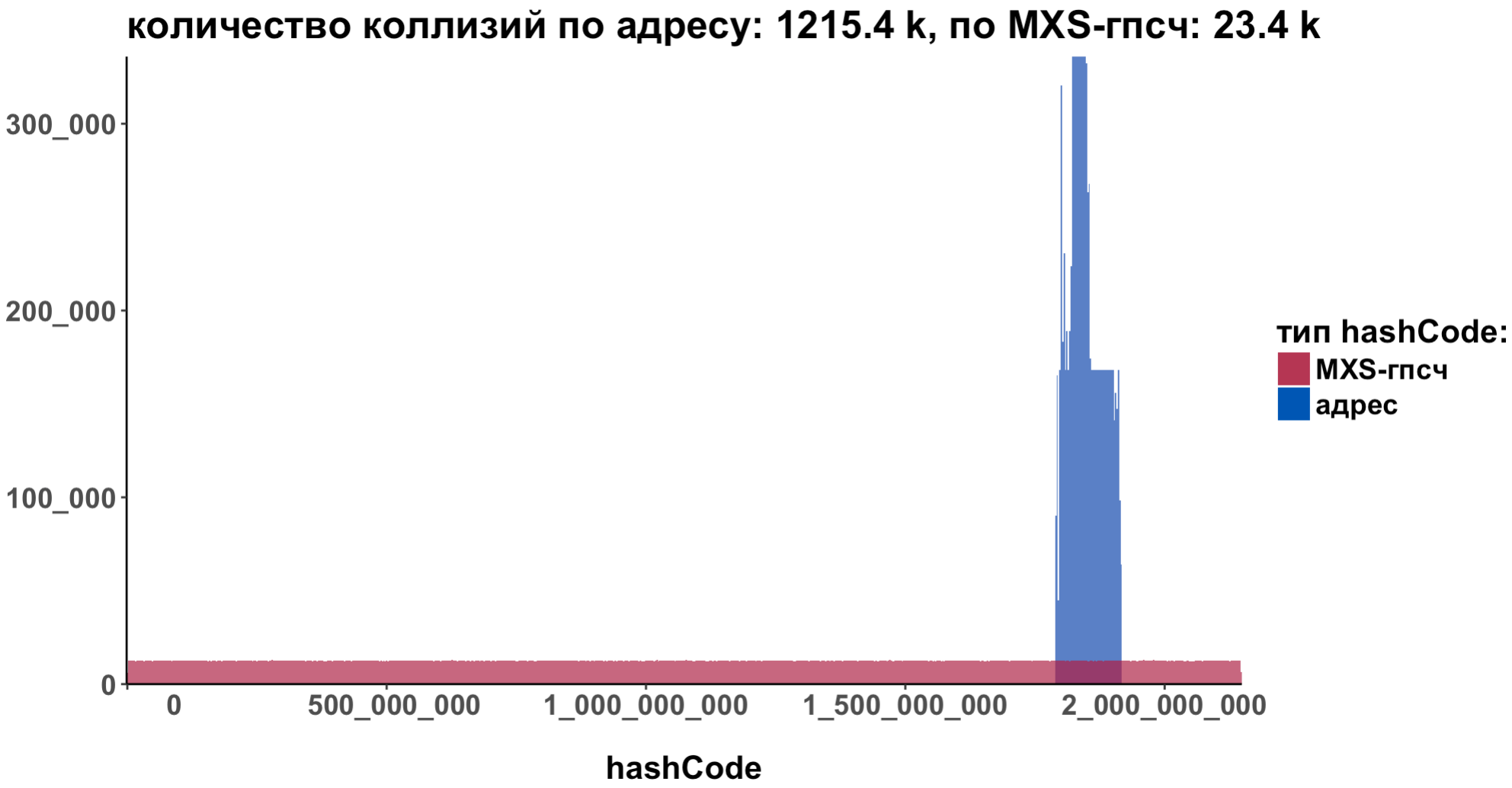
-XX:hashCode

-XX:hashCode=k	Тип
0	Park-Miller ГПСЧ
1	**fn(адрес объекта, глобальное состояние)**
2	**константа 1**
3	**последовательный счетчик**
4	**адрес объекта**
5	Marsaglia xor-shift ГПСЧ ***по умолчанию*** *в java 8*

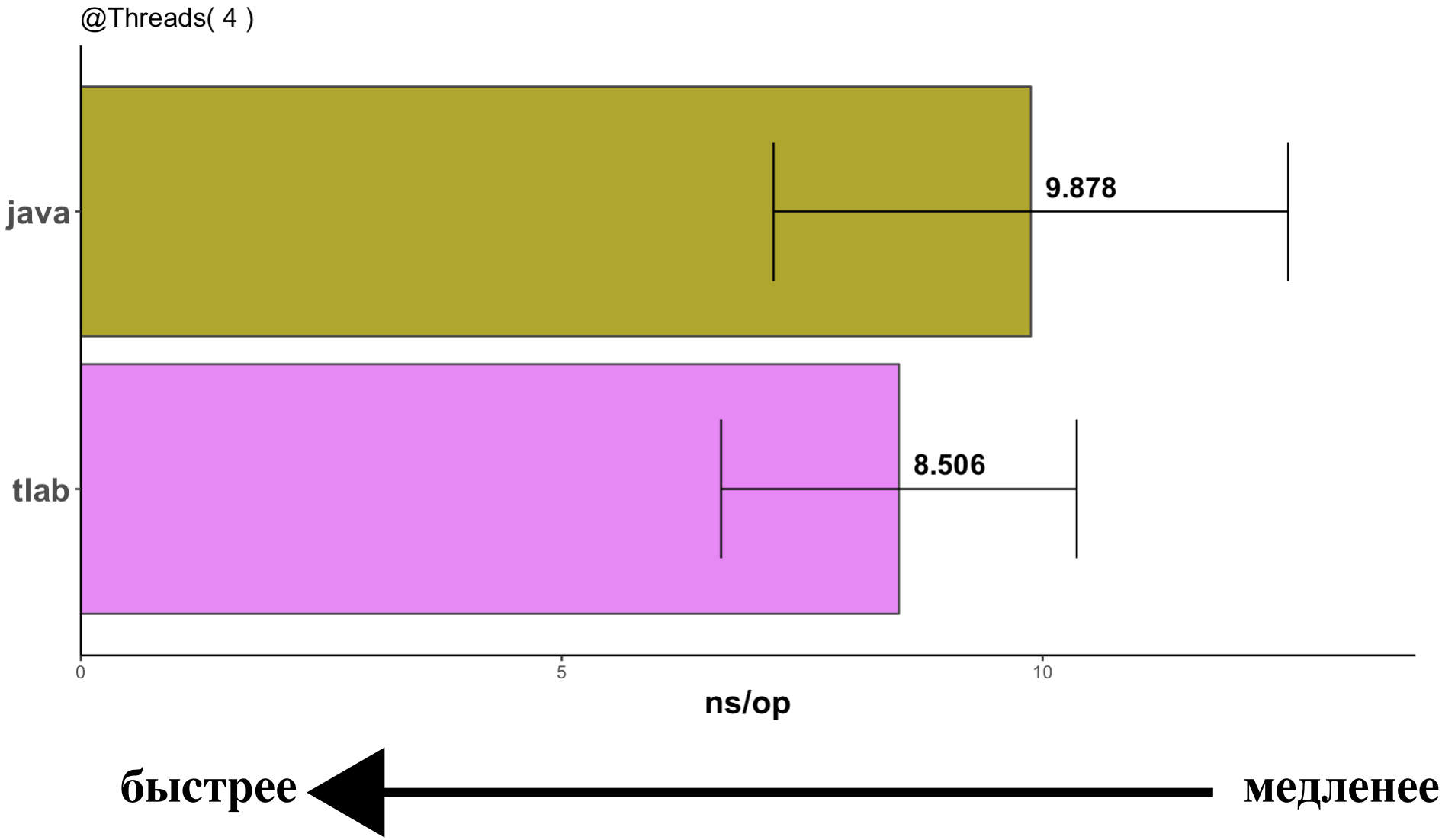
Дамп объекта при -XX:hashCode=5



Распределение hashCode 10 млн об-в в 10 нитях



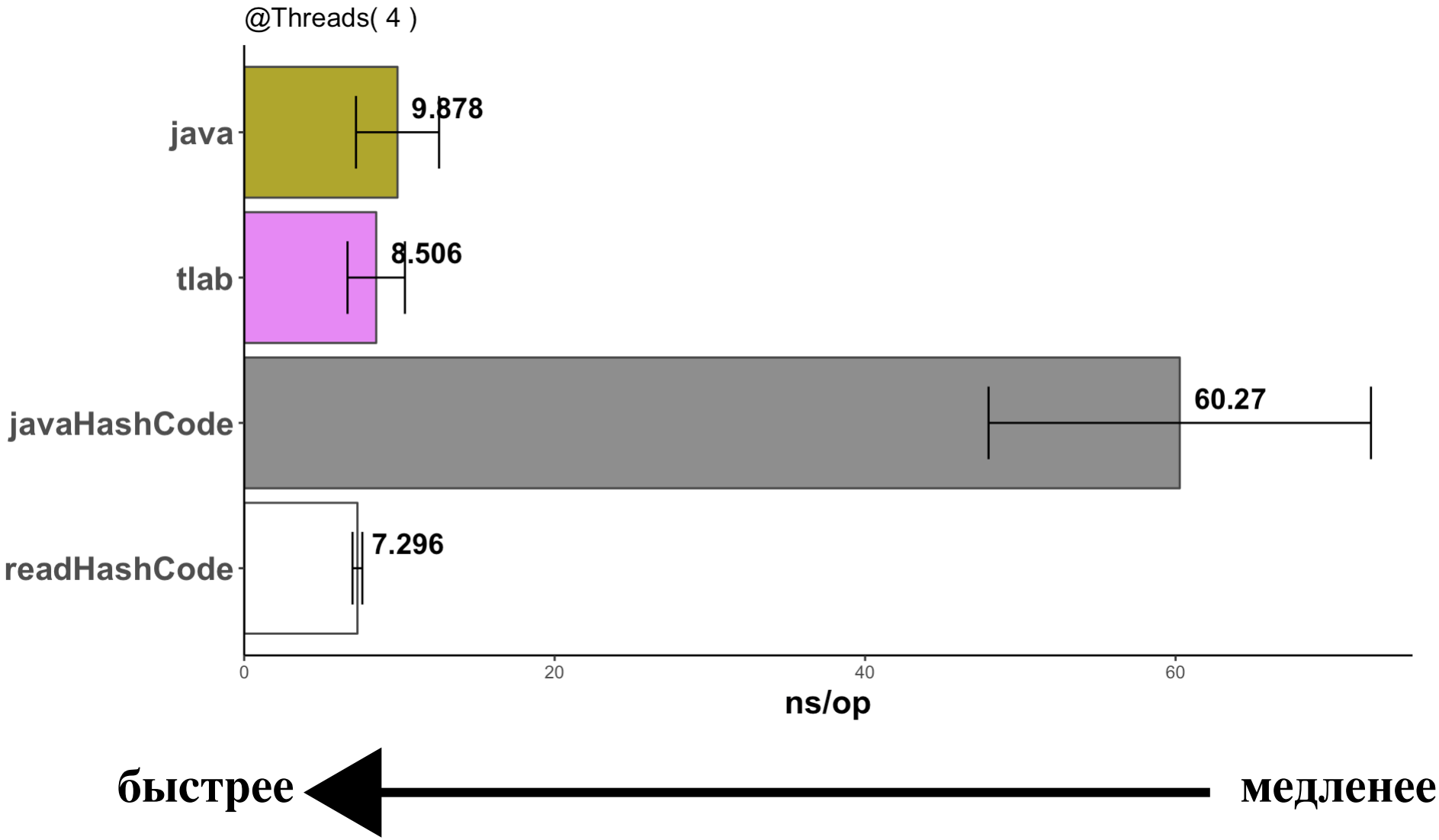
Вычислялся ли hashCode ?



Object.hashCode() benchmark

```
Object theObject = new Object();

@Benchmark @Threads( 4 )
public Object javaHashCode() {
    Object object = new Object();
    object.hashCode();
    return object;
}

@Benchmark @Threads( 4 )
public Object readHashCode() {
    return theObject.hashCode();
}
```

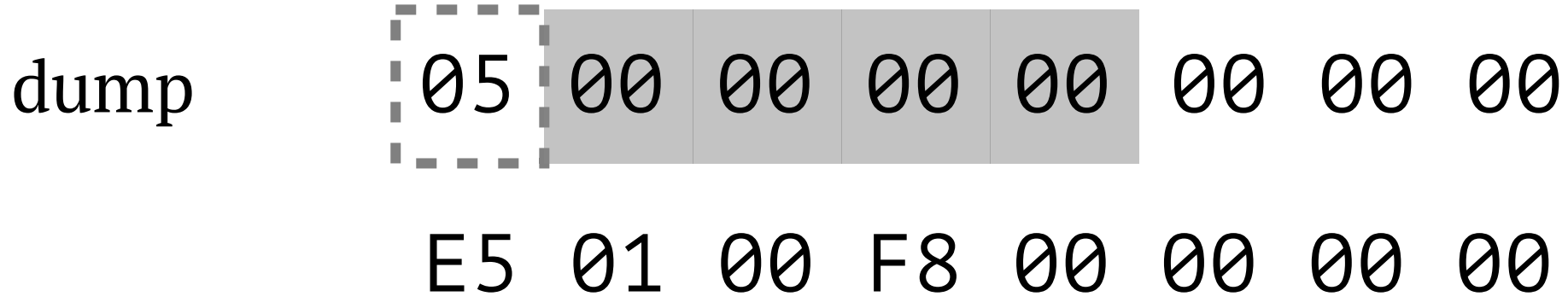
Object.hashCode() benchmark



быстрее ← **медленее**

Дамп объекта, сразу после создания

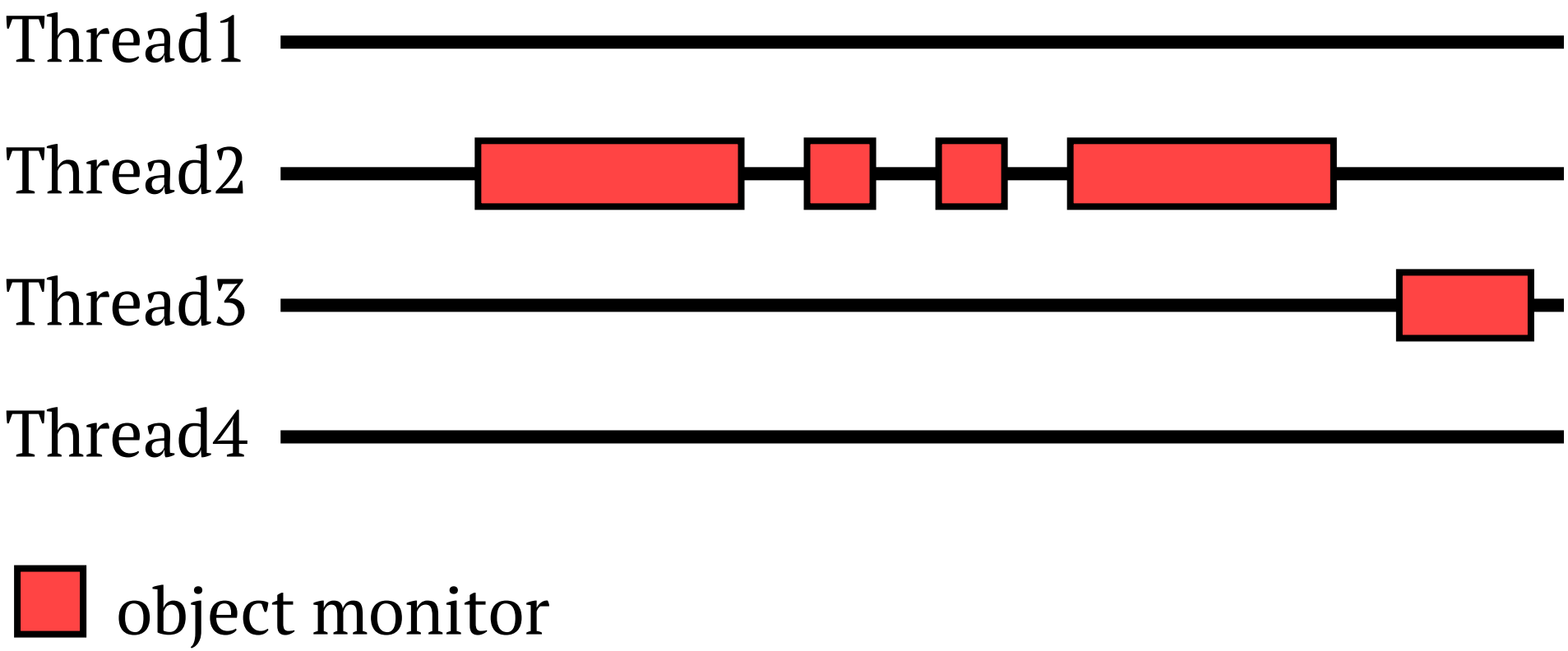
```
Object object = new Object();

dump( object );
```



```
public class Object {
   //// другие методы
   public final native void notify();

   public final native void wait(long timeout)
       throws InterruptedException;
}
```

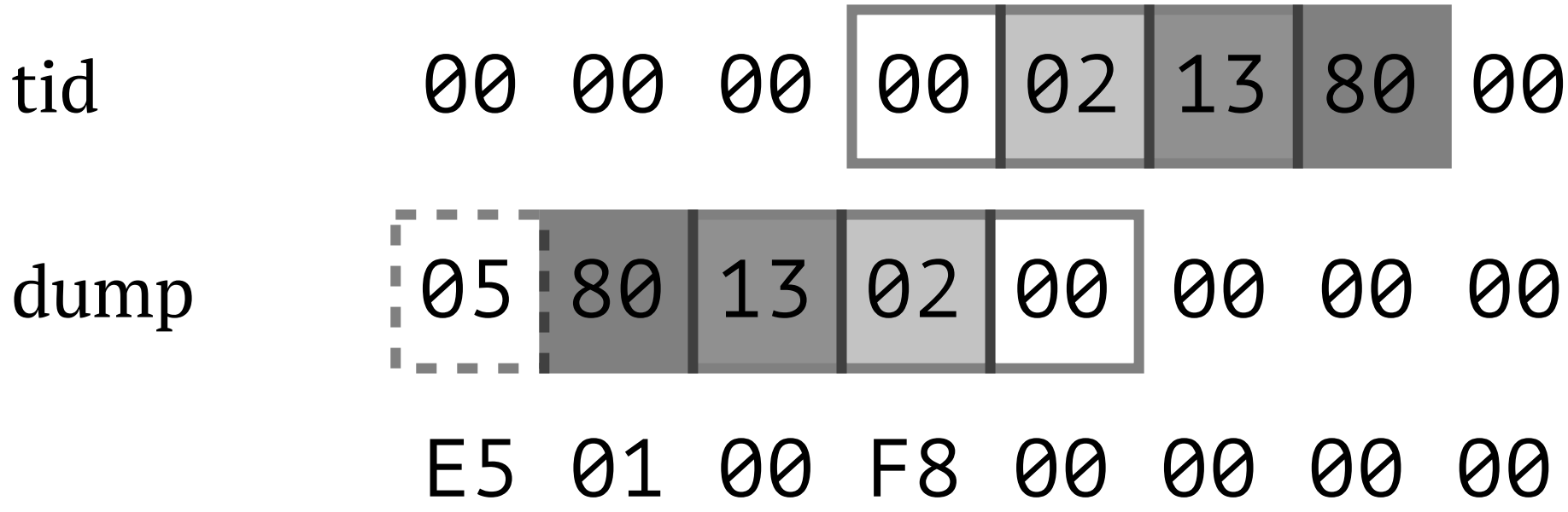
monitor

Biased Locking



Biased Locking demo:

```
synchronized (object) {
    object.notifyAll();
}
dump( object );
```



StringBufferPerfTest

```
StringBuffer buf = new StringBuffer();


@Benchmark @Thread(1)
public String bufferToString() {
   return buf.toString();
}
```

`-XX:+UseBiasedLocking` VS `-XX:-UseBiasedLocking`

```
-XX:BiasedLockingStartupDelay=0
```

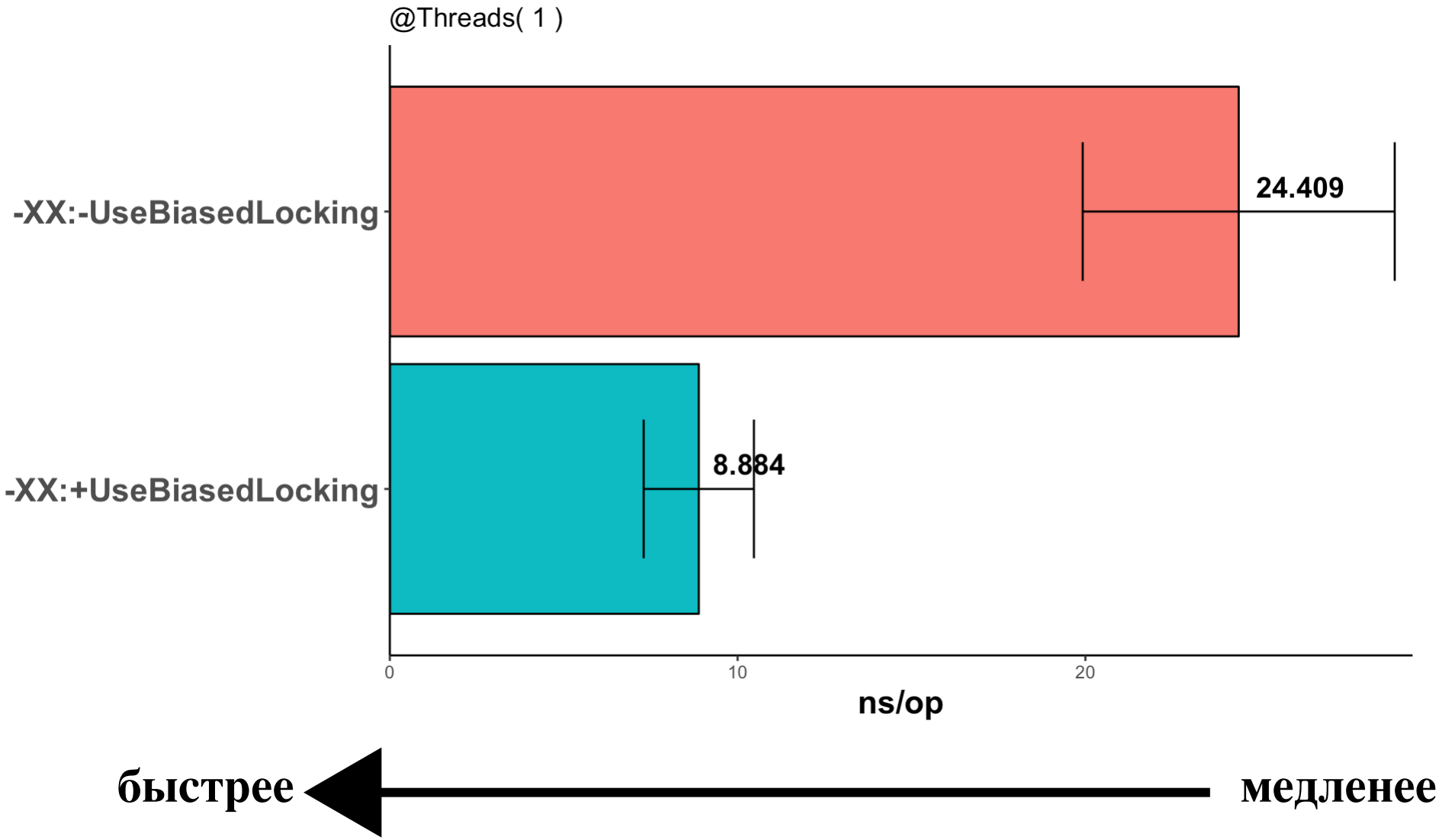
StringBufferPerfTest

```
StringBuffer buf = new StringBuffer();


@Benchmark @Thread(1)
public String bufferToString() {
   return buf.toString();
}
```

`-XX:+UseBiasedLocking` VS `-XX:-UseBiasedLocking`

`-XX:BiasedLockingStartupDelay=0`

StringBufferPerfTest результаты



быстрее ◀ ——— **медленее**

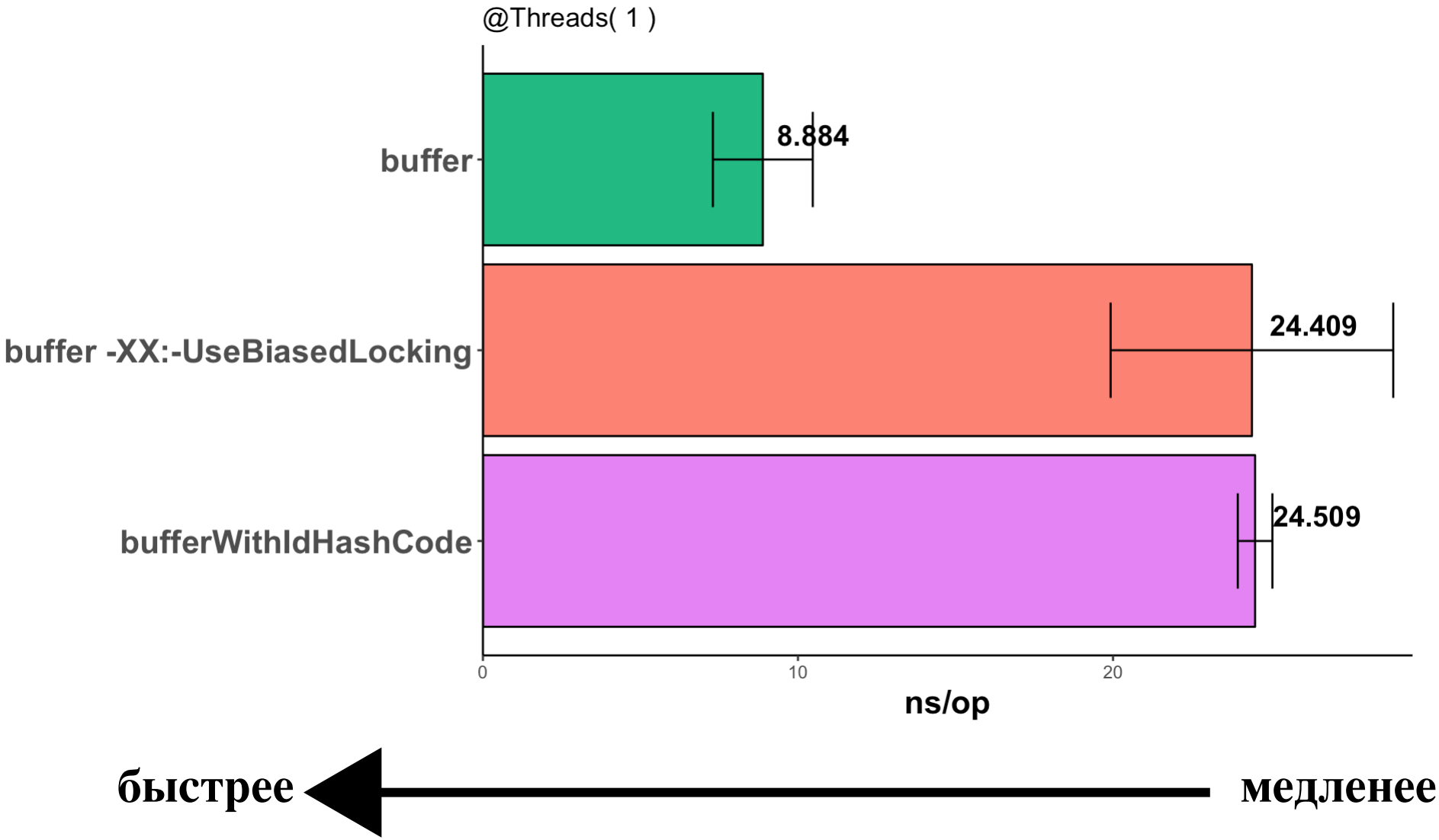
identityHashCode

```
StringBuffer buf = new StringBuffer();
StringBuffer bufferWithIdHashCode =
    new StringBuffer();

@Setup
public void setup(Blackhole bh) {
    bh.consume( System.identityHashCode(
        bufferWithIdHashCode ) );
}

@Benchmark  @Thread(1)
public String bufferWithIdHashCode() {
  return bufferWithIdHashCode.toString();
}

@Benchmark @Thread(1)
public String buffer() {
  return buf.toString();
}
```

identityHashCode

```
StringBuffer buf = new StringBuffer();
StringBuffer bufferWithIdHashCode =
    new StringBuffer();

@Setup
public void setup(Blackhole bh) {
    bh.consume( System.identityHashCode(
          bufferWithIdHashCode ) );
}

@Benchmark  @Thread(1)
public String bufferWithIdHashCode() {
  return bufferWithIdHashCode.toString();
}

@Benchmark  @Thread(1)
public String buffer() {
  return buf.toString();
}
```

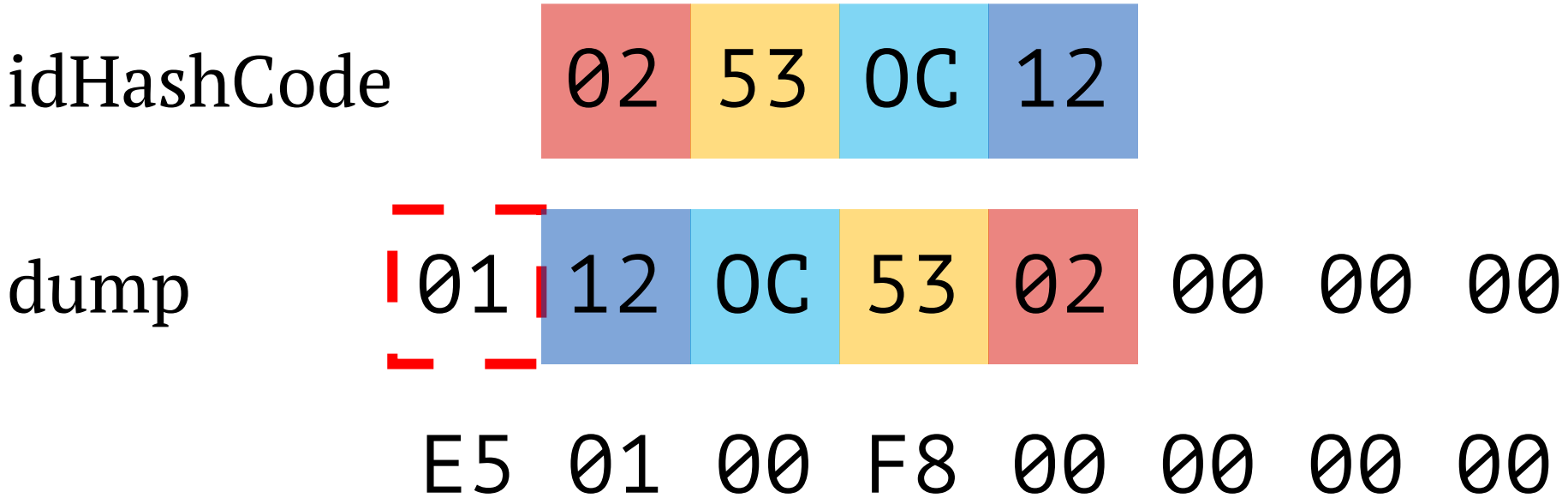
identityHashCode результаты



быстрее ← **медленее**

Revoke Biased Locking:

```
int idHashCode
   = System.identityHashCode( object );

dump( object );
```



Заключение

- **Hash-структуры** - быстро
- **Переопределяйте `hashCode` и `equals`**
  - и ещё лучше **определяйте `compareTo`**
  - исследуйте свои hash-функции
- hashCode - не **адрес**
  - **GC**, **TLAB**, не злоупотребляйте **`identityHashCode`**
- **Meten is weten** (голл.) «Измерение - знание»

слайды и примеры: github.com/vladimirdolzhenko/hashCodeLegend

email: vladimir.dolzhenko@gmail.com

twitter: @dolzhenko

Контакты

спасибо: @VladimirSitnikv @dj_begemot @AndreiPangin @gvsmirnov @i_sopov @relgames