Graal, Value Types, Loom и другие ништяки

что это, и зачем нужно простому смертному

@olegchir

JUG.ru Group (конференции)

Сбербанк-Технологии: ППРБ
Минздрав: ЕГИСЗ, ИЭМК
Госуслуги
IUPAT
Starview Inc
Playtox
CodeOrchestra
Erlyvideo (Flussonic)


http://www.etsy.com/shop/sharpwriter
UEUSER 2012

The following is intended to outline our supposed general direction of growth and development as Java programmers. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release and timing of any features or functionality described for products or services remains at the sole discretion of corresponding companies. Opinions expressed are solely my own and do not express the views or opinions of my employer.

Информация предназначается чтобы обозначить наше общее направление роста и развития как Java-программистов. Она предоставляется только в целях ознакомления, и не может быть использована в контрактах или договорах любого вида. Эта информация не является попыткой предоставления какого-то материала, кода, или функциональности, и не должна быть использована в принятии коммерческих решений. Разработка, выпуск, календарные сроки любых объектов или функциональности, описанных в контексте продуктов или услуг различных компаний, остается на усмотрение соответствующих компаний. Описанные в докладе мнения не отражают позиции работодателя.

Java 9

Java 10

Java 11

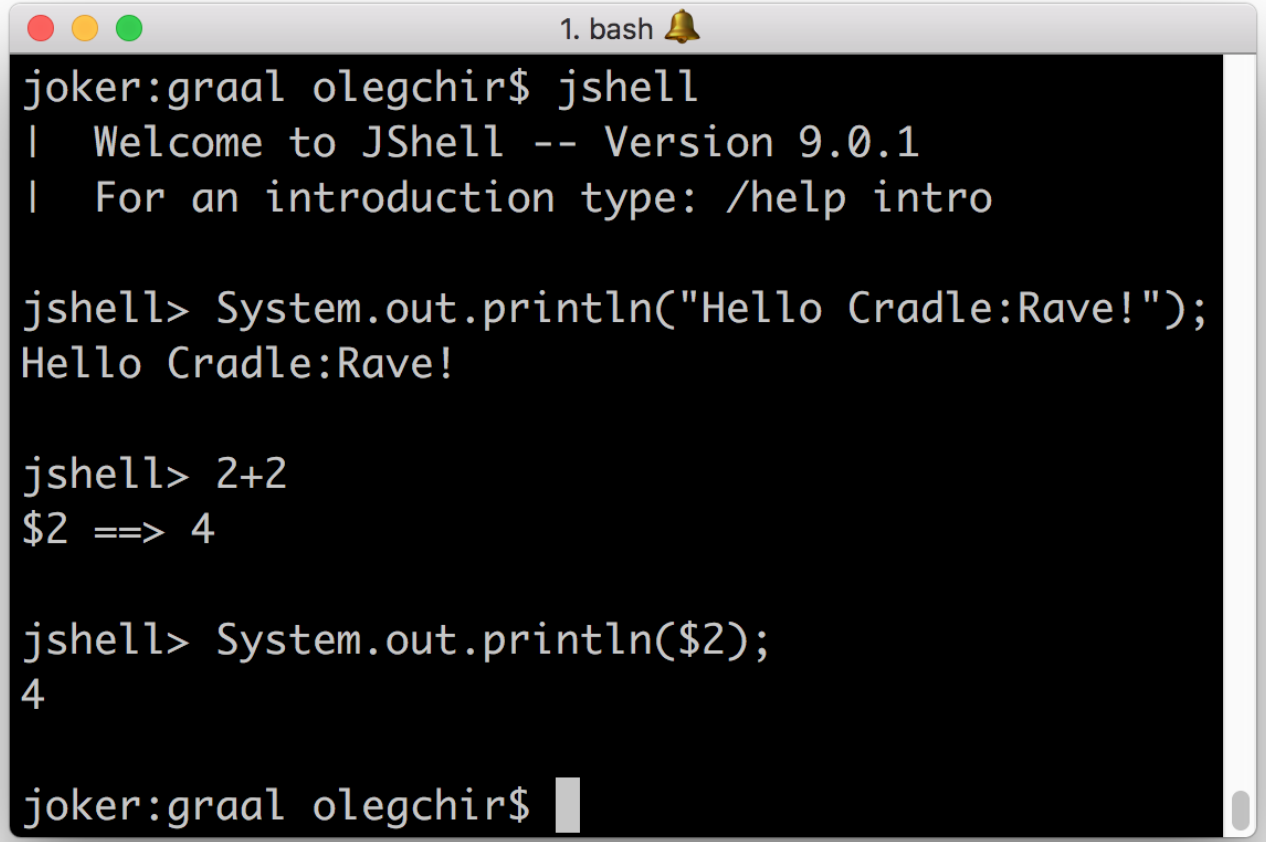
Process API updates
HTTP/2 Client
Improve Contended Locking
Unified JVM Logging
Compiler Control
Variable Handlers
Segmented Code Cache
Smart Java Compilation, Phase Two
The Modular JDK
Modular Source Code
Elide Deprecation Warnings on Import Statements
Resolve Lint and Doclint Warnings
Milling Project Coin
Remove GC Combinations Deprecated in JDK 8
Tiered Attribution for javac
Process Import Statements Correctly
Annotations Pipleine 2.0
Datagram Transport Layer Security (DTLS)
Modular Run-Time Images
Simplified Doclet API
jshell: The Java Shell (Read-Eval-Print Loop)
New Version-String Scheme
HTML5 Javadoc
Javadoc Search
UTF-8 Property Files
Unicode 7.0
Add More Diagnostic Commands
Create PKCS12 Keystores by Default
Remove Launch-Time JRE Version Selection
Improve Secure Application Perormance

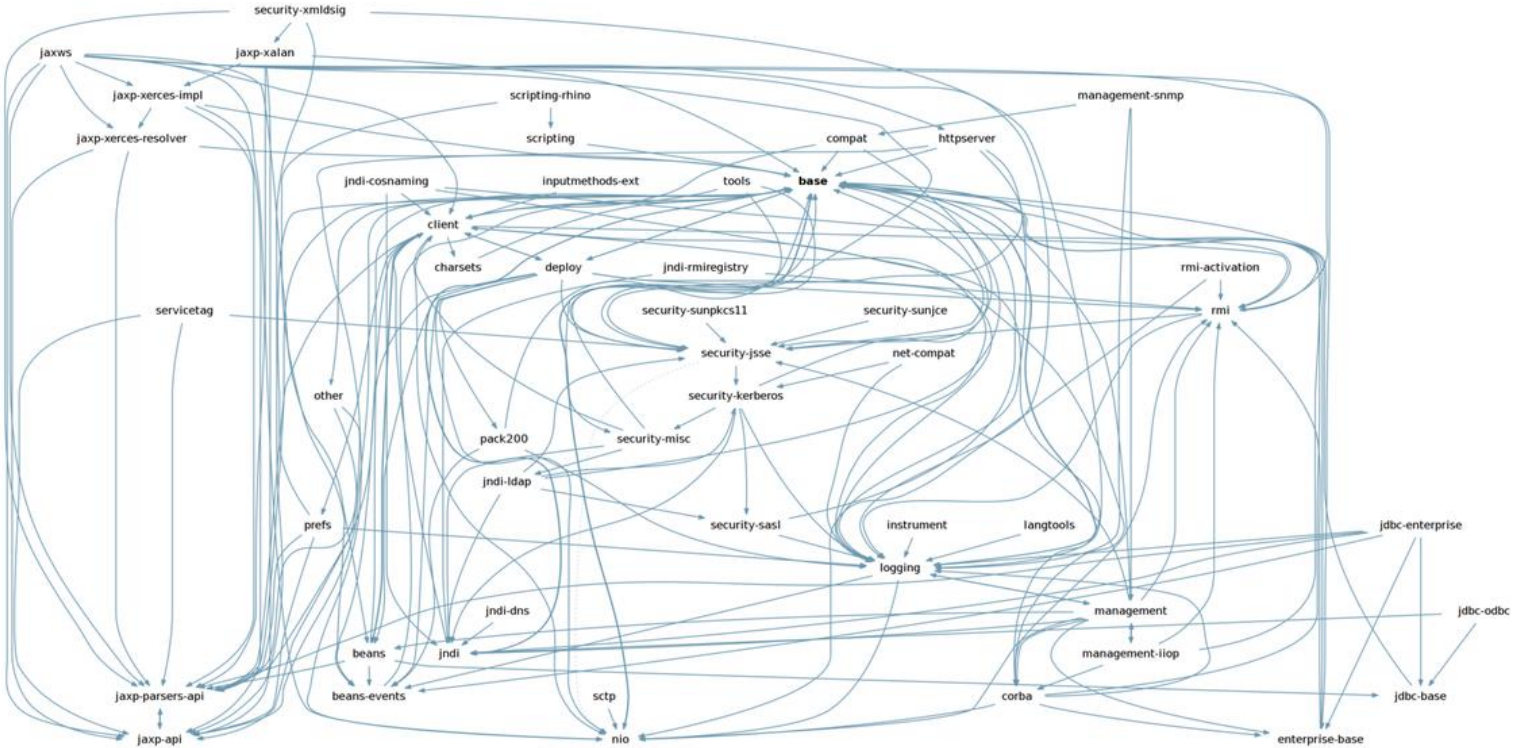
Generate Run-Time Compiler Tests
Test Class-File Attributes Generated by javac
Parser API for Nashorn
Linux/AArch64 Port
Multi-Release JAR Files
Remove the JVM TI hprof Agent
Remove the jhat Tool
Java-Level JVM Compiler Interface
TLS ALPN
Validate JVM Command-Line Flag Arguments
Leverage CPU Instructions for GHASH and RSA
Compile for Older Platform Versions
Make G1 the Default Garbage Collector
OCSP Stapling for TLS
Store Interned Strings in CDS Archives
Multi-Resolution Images
Use CLDR Locale Data by Default
Prepare JavaFX for Modularization
Compact Strings
Merge Selected Xerces Updates into JAXP
BeanInfo Annotations
Update GStreamer n JavaFX/Media
HarfBuzz Front-Layout Engine
Stack-Walking API
Encapsulate Most Internal APIs
Module System
TIFF Image I/O
HiDPI Graphics on Windows and Linux
Platform Logging API and Service
MArlin Graphics Renderer
More Concurrency Updates

Unicode 8.0
XML Catalogs
Convenience Factory Methods for Collections
Reserved Stack Areas for Critical Sections
Unified GC Logging
Platform-Specific Desktop Features
DRBG-Based SecureRandom Implementations
Enchanced MethodHandles
Modular Java Application Packaging
Dynamic Linking of Langage Defined Object Models
Enchanced Deprecation
Additional Tests for Humongous Objects in G1
Improve Test-Failure Troubl;eshooting
Indify String Concatenation
HotSpot C++ Unit-Test Framework
jlink: The Java Linker
Enable GTK 3 on Linux
New HotSpot Build System
Spin-Wait Hints
SHA-3 Hash Algorithms
Disable SHA-1 Certificates
Deprecate the Applet API
Filter Incoming Serialization Data
Deprecate the Concurrent Mark Sweep GC
Implement Selected ECMAScript 6 Features
Linux/s390x Port
Ahead-of-Time Compilation
Unified arm32/arm64 Port
Remove Demos and Samples
Reorganize Documentation

JS HELL

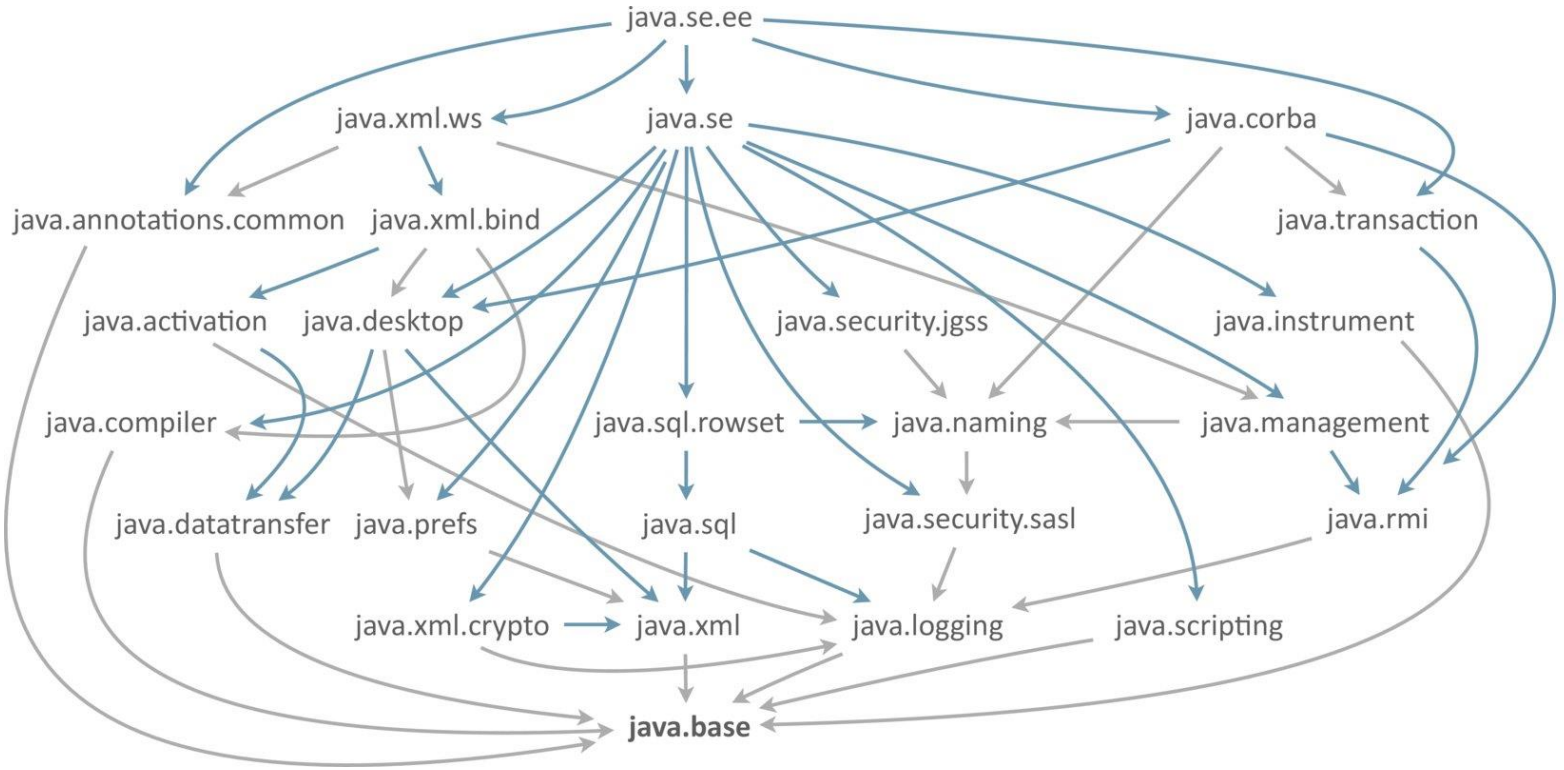
JSHELL


1. bash
joker:graal olegchir$ jshell
| Welcome to JShell -- Version 9.0.1
| For an introduction type: /help intro
jshell> System.out.println("Hello Cradle:Rave!");
Hello Cradle:Rave!
jshell> 2+2
$2 ==> 4
jshell> System.out.println($2);
4
joker:graal olegchir$



2010

https://twitter.com/mreinhold/status/882644292036026368?lang=en



2017

https://twitter.com/mreinhold/status/882644292036026368?lang=en

Twitter: #WorksFineOnJDK9

Akka
Ant
AAvatica
Bean Validation
Blynk Server
BootstrapFX
ByteBuddy
Calcite
Camel
cglib
Commons Lang
CruiseControl
Derby
Eclipse Collections
Eclipse Eclemma
Eclipse IDE
Elasticsearch
EqualsVerifier
Gluon Mobile
Gradle
Graph Hopper
HeapStats
Hibernate ORM
Hibernate Validator
install4j
instana
IntelliJ IDEA
Jackson
jaCoCo
javassist
JCTools
Jersey
JMH
JITWatch
Joda Beans
Joda Convert
jOOQ
JOSM
JProfiler
JRebel
JUnit
Kafka
Lettuce
Lilith
Log4j
logback
Lucene
LWGL
Maven
Mockito
Netty
Rabbit MQ Client
Rapidoid
RxJava
slf4j
Solr
Spotbugs
Spring Boot
Spring Framework
stagemonitor
TestNG
Thymeleaf
Wildfly
Woodstox

СВОБОДУ ВСЕМ!

- JFR (Java Flight Recorder)
- JMC (Java Mission Control)
- ZGC (Z Garbage Collector)
- AppCDS (Application Class-Data Sharing, JEP 310)

JAVAEE → JAKARTA EE

DARTH JAKARTA V1.0

COMPACT

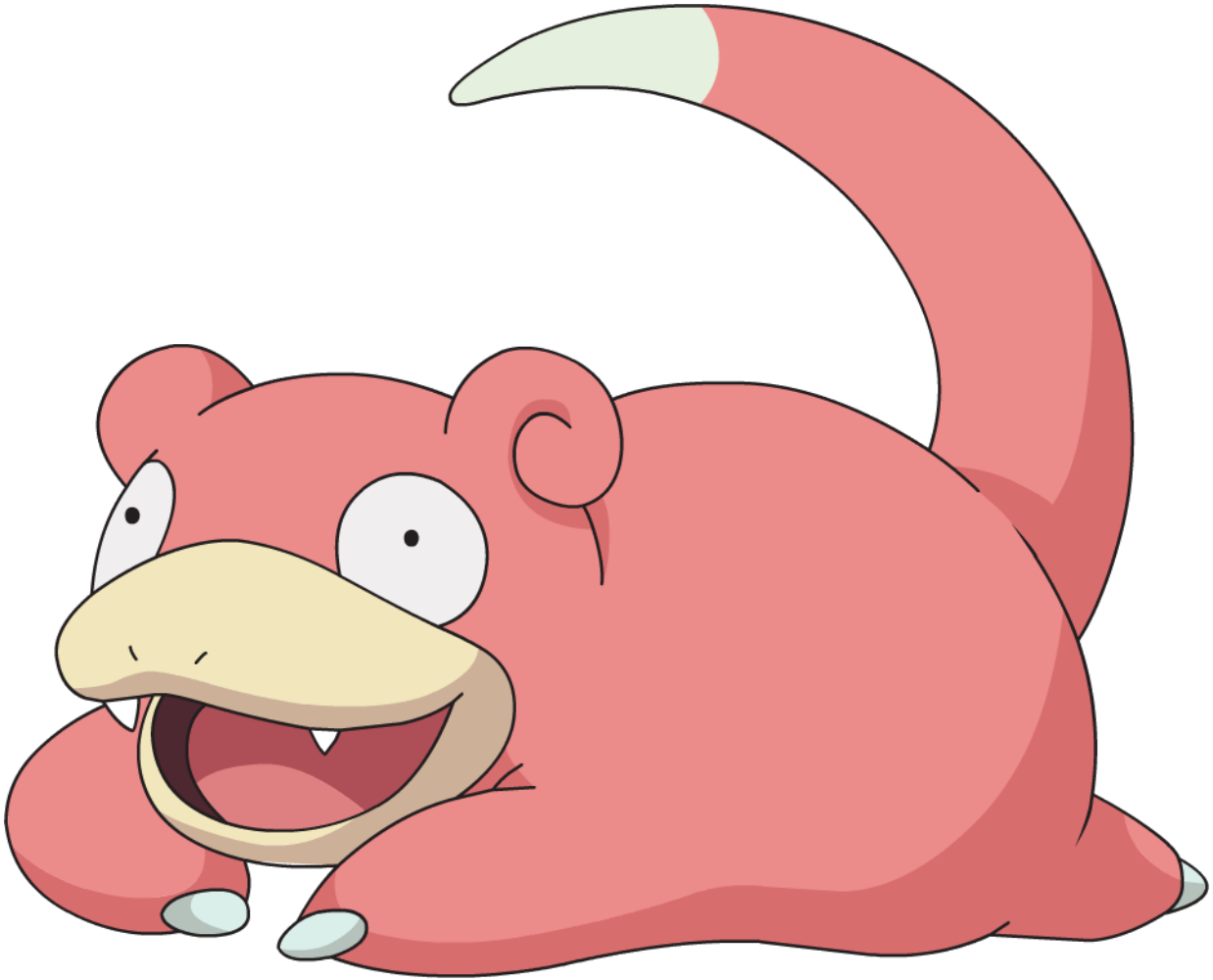
ULTRACOMPACT

DARTH JAKARTA V2.0

НОВЫЙ ПЛАН ВЫПУСКА РЕЛИЗОВ

Moving Java Forward Faster

(Mark Reinhold)



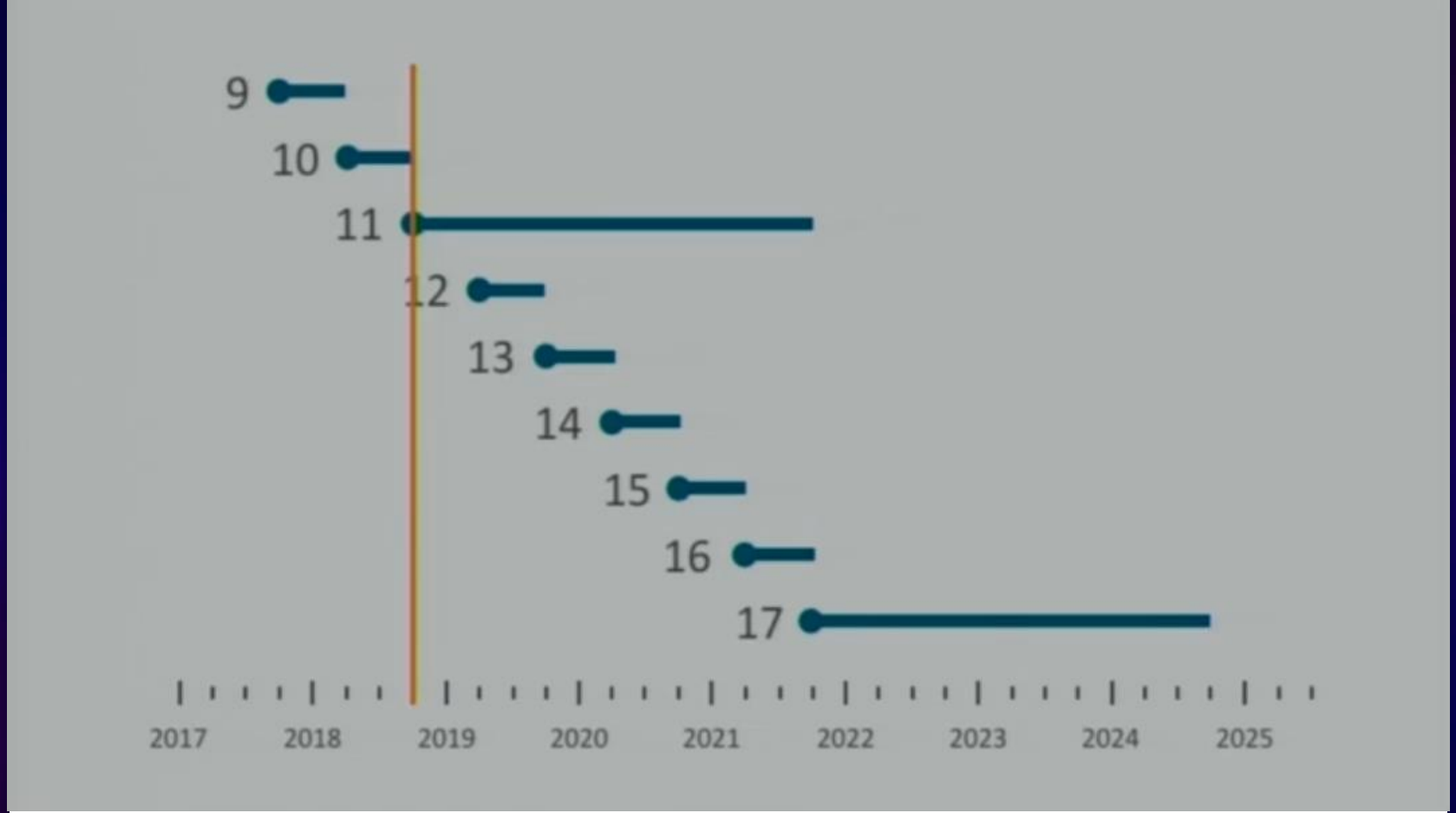
Mark Reinhold • 1st

Chief Architect, Java Platform Group at Oracle

Oracle • Massachusetts Institute of Technology

San Francisco Bay Area • 250

Message ···



https://fosdem.org/2018/schedule/event/state_openjdk/

ПЛАН РЕЛИЗОВ

(Сам план, описания фаз)

2017/12/14 Первая фаза замедления
2018/01/11 All Tests Run
2018/01/18 Вторая фаза замедления
2018/02/08 Первый Release Candidate
2018/02/22 Окончательный Release Candidate
2018/03/20 General Availability

---вы находитесь здесь---


12
9
3
6

Изображение: https://alexsportvn.com/cach-nhan-biet-giay-nike-chinh-hang

John Rose • 1st

Java Virtual Machine Architect

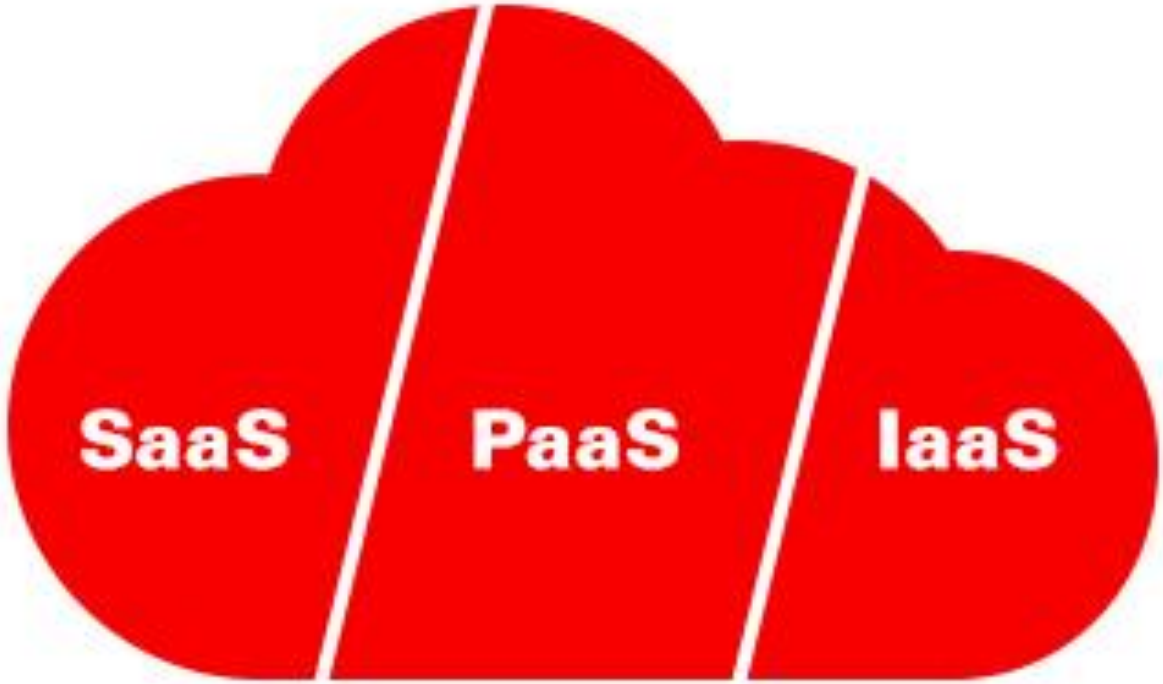
Oracle • UC Santa Barbara

San Francisco Bay Area • 423

Message ···

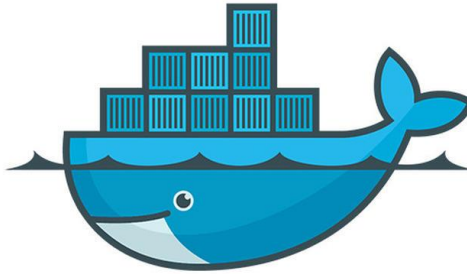
https://fosdem.org/2018/schedule/event/jam_futures/


CALIFORNIA
OUTATIME



Изображение: http://www.bankingtech.com/2017/03/oracle-earnings-look-promising-but-why-the-iaas-obsession/


Azure



docker

kubernetes

LXC

Язык Java и другие JVM языки

Платформа JVM

Язык Java

- Популярные конструкции
- Быстро усваивается (C++?)
- Целостность. Единая модель языка
  (объекты, массивы, значния, типы – ощущаются одинаково)
- Обратная совместимость
- Интероп с неуправляемыми языками
- Другие языки (JS, Ruby, Python)

Платформа JVM

- Эффективный расход памяти
- Мощные новые оптимизации (AOT, JIT, GC, RT)
- Гранулярная многопоточность
- Прямой доступ к железу (CPU, GPU)
- Переносимость между платформами

- Корутины и легкие треды
- Структуры как в C++
- Сборка мусора, которая не тормозит

Fiber

- **User mode thread**
  (управление не о ОС, а Java runtime/user code)
- Работает как thread
- Занимает мало памяти
- Дешевые переключения
- Можно использовать не тысячи, а миллиорны

Почему сейчас

- Сессии становятся дольше (realtime push,итп)
- Сервера ждут IO (ответ от базы и сервисов)
- Утилизация CPU 5-10%
- Железо пропадает зря!

Есть два стула

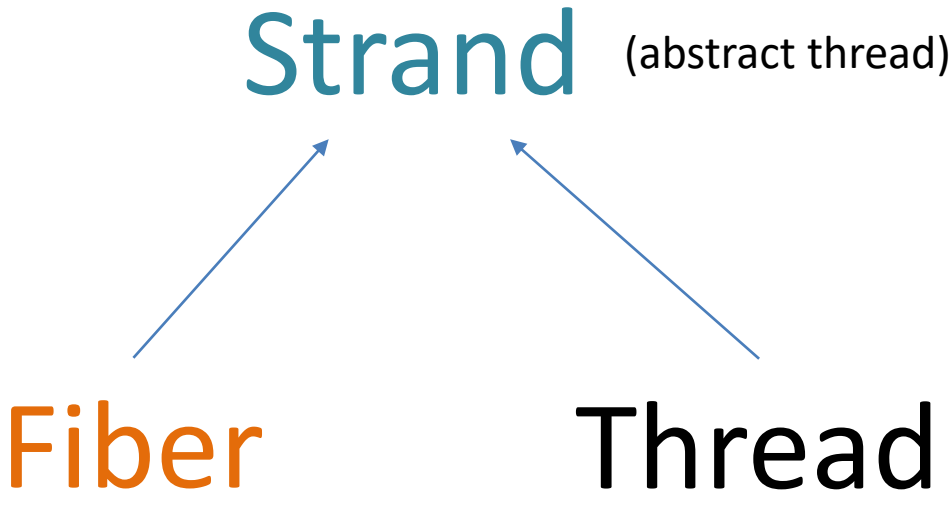
- Продолжать писать блокирующий код
- Писать асинхронный код
  - Плохо работает с легаси блокирующим
  - Плохо мастшатбируется (читать, отлаживать)

В чем профит

- Обычный код становится дешевле (+легаси)
- Не нужно выбирать между стульями
- Можно писать «современным» образом

Loom

Thread (Runnable, ThreadSheduler)


Strand
(abstract thread)
Fiber
Thread

```
ServerSocketChannel s = ServerSocketChannel.open().bind(new InetSocketAddress(8080));
new Fiber(() -> {
    try {
        while (true) {
            final SocketChannel ch = s.accept();
            new Fiber(() -> {
                try {
                    var buf = ByteBuffer.allocateDirect(1024);
                    var n = ch.read(buf);
                    String response = "HTTP/1.0 200 OK\r\n";
                    n = ch.write(encoder.encode(CharBuffer.wrap(response)));
                } catch (IOException e) {}
            }).start();
        }
    } catch (IOException e) {}
}).start();
```

Demo

Demo

Ron Pressler

Quasar | Parallel Universe

Sponsor

hotspot	HotSpot Group

Lead

rpressler	Ron Pressler

Committers

acorn	Karen Kinnear
alanb	Alan Bateman
briangoetz	Brian Goetz
dcubed	Daniel D. Daugherty
dholmes	David Holmes
ehelin	Erik Helin
eosterlund	Erik Österlund
fparain	Frederic Parain
jrose	John R Rose
kvn	Vladimir Kozlov
mchung	Mandy Chung
mikael	Mikael Vidstedt
mr	Mark Reinhold
psandoz	Paul Sandoz
rriggs	Roger Riggs
vlivanov	Vladimir Ivanov

Project Loom: Fibers and Continuations for the Java

Project Loom: Fibers and Continuations for the JVM (Proposal)

Project Loom: Fibers and Continuations for the Java

Project Loom: Fibers and Continuations for the JVM (Proposal)

Корутины в Kotlin — Роман Елизаров, JetBrains

Value Types

```
final class Point {
    public final int x;
    public final int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

```
var points = new ArrayList<Point>();
points.add(new Point(1,2));
points.add(new Point(3,4));
```

```
ClassLayout.parseInstance(points).toPrintable()
ClassLayout.parseInstance(points.get(i)).toPrintable()
```

java.util.ArrayList object internals:

OFFSET	SIZE	TYPE DESCRIPTION
0	4	(object header)
4	4	(object header)
8	4	(object header)
12	**4**	**int AbstractList.modCount**
16	**4**	**int ArrayList.size**
20	4	java.lang.Object[] ArrayList.elementData

[(object), (object), null, null, null,
null, null, null, null, null]

com.olegchir.Point object internals:

OFFSET	SIZE	TYPE	DESCRIPTION	
0	4		(object header)	
4	4		(object header)	
8	4		(object header)	
12	**4**	**int**	**Point.x**	**1**
16	**4**	**int**	**Point.y**	**2**
20	4		(loss due to the next object alignment)	

Instance size: 24 bytes

Space losses: 0 bytes internal + 4 bytes external = 4 bytes

```
@ValueCapableClass
final class Point {
    public final int x;
    public final short y;
    public final short z;

    Point(int x, short y, short z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }
    ...
}
```

```
@Override
 public boolean equals(Object o) {
    if (o instanceof Point) {
       Point that = (Point)o;
       return that.x == x &&
              that.y == y &&
              that.z == z;
    } else {
       return false;
    }
}

 @Override
 public int hashCode() {
    return Objects.hash(x, y, z);
}
```

- Класс должен быть помечен как final
- Класс должен быть правильным классом (не интерфейсом)
- Суперкласс обязательно Object
- Все нестатические поля помечены как final
- Свои методы: equals, hashCode, toString
- Нельзя переопределять: clone, finalize

- Immutable
- Нельзя использовать == (нужно equals)
- Эквивалентные объекты нельзя заменять

Тесты

Сергей Куксенко

http://cr.openjdk.java.net/~skuksenko/valhalla/benchmarks/

Сборка

hg clone http://hg.openjdk.java.net/valhalla/valhalla valhalla-mvt

cd valhalla-mvt

hg update -r mvt // старая стабильная демка осени прошлого года
hg update -r lword //новая линия Партии

bash ./configure
--enable-debug
--with-target-bits=64
--disable-warnings-as-errors

make images

Shady Values (John Rose)

https://habrahabr.ru/company/jugru/blog/336378/

http://cr.openjdk.java.net/~jrose/values/shady-values.html

JVM Language Summit

Minimal Values Under the Hood with Bjorn Vardal @bvaardal and Frédéric Parain

Programming with Minimal Values with Maurizio Cimadamore

SHENANDOAH

JEP 189 (JDK-8046179): An Ultra-Low-Pause-Time Garbage Collector
Authors Christine H. Flood, Roman Kennke, **reviewed by Aleksey Shipilev**

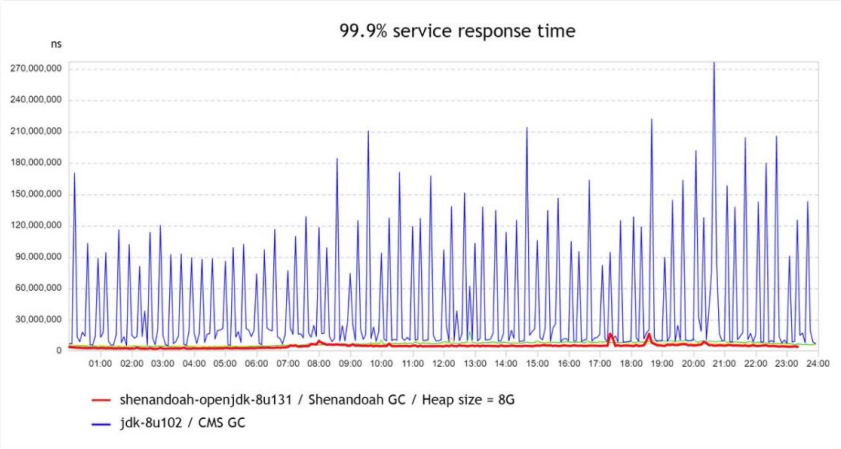
SHENANDOAH

1. Ultra-Low-Pause-Time Garbage Collector
2. Not the one GC to rule them all
3. Отзывчивость и предсказуемые короткие паузы
4. Вне зависимости от размера heap
5. Решает проблему SLA 10-500ms
6. Платим за короткие паузы CPU и RAM
7. Marking, compacting – многопоточные
8. Ограничено сканированием thread stacks, чтобы построить рутсет графа объектов

Andrei Pangin
@AndreiPangin

Following

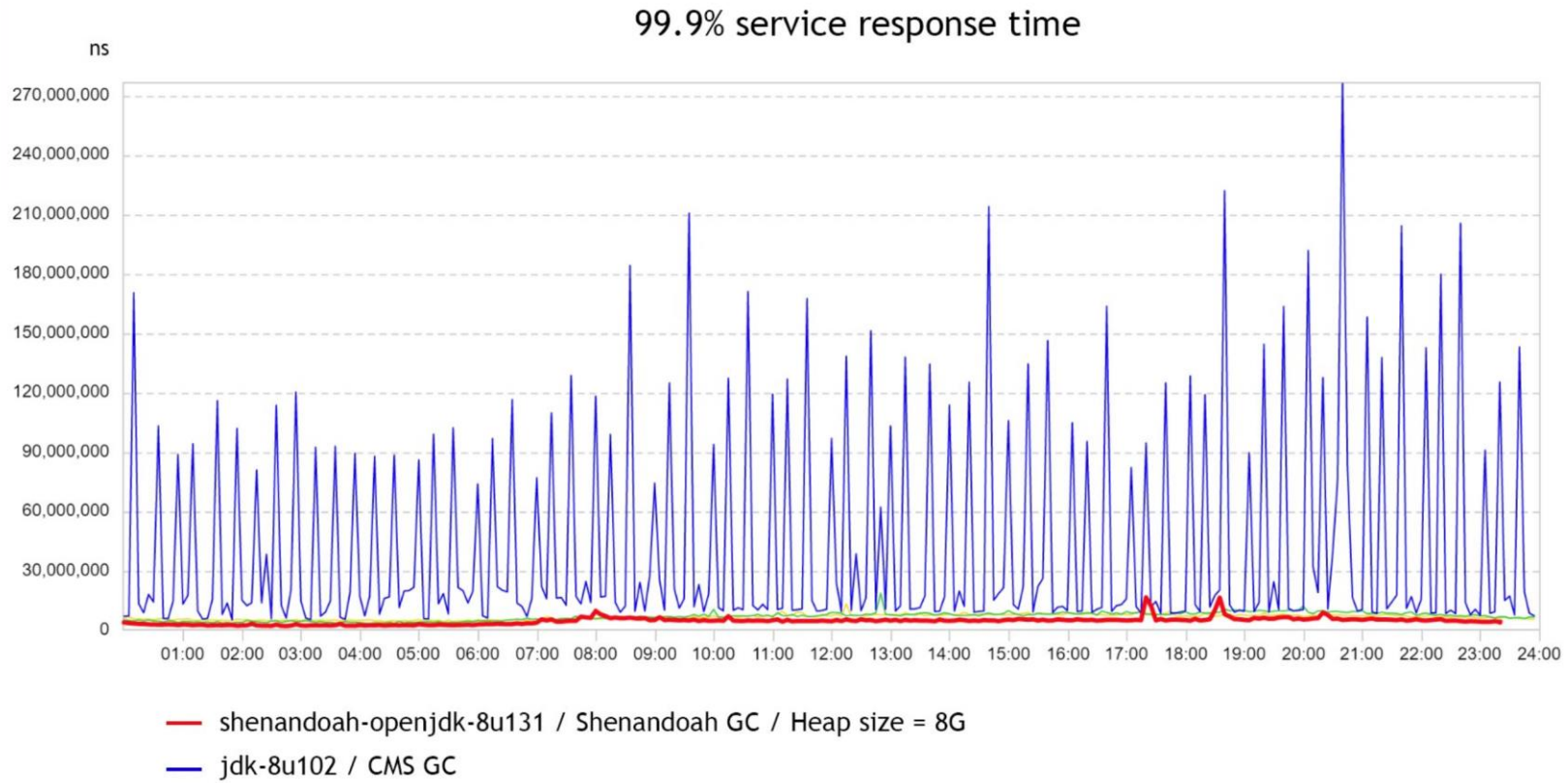
@shipilev Migration to #ShenandoahGC dramatically reduced response latency for one of #odnoklassniki applications. Haven't yet succeeded with other applications though.



4:19 AM - 2 Feb 2018

27 Retweets **66** Likes

5 27 66



https://gist.github.com/apangin/3fb525aed1e06457a0a408b0e6125e0e

https://twitter.com/AndreiPangin/status/959174428318928898

SHENANDOAH: ССЫЛКИ

1. Статья «An open-source concurrent compacting garbage collector for OpenJDK»
2. JEP 189 (JDK-8046179)
3. Shenandoah Wiki

Доклады Алексея Шипилёва

4. «Сборщик мусора, который смог»

   Первая часть: доклад на JPoint и JBreak 2017
   (доступна прямо сейчас)

   Вторая часть: доклад на Joker 2017
   (скорей всего, будет в публичном доступе через несколько месяцев)

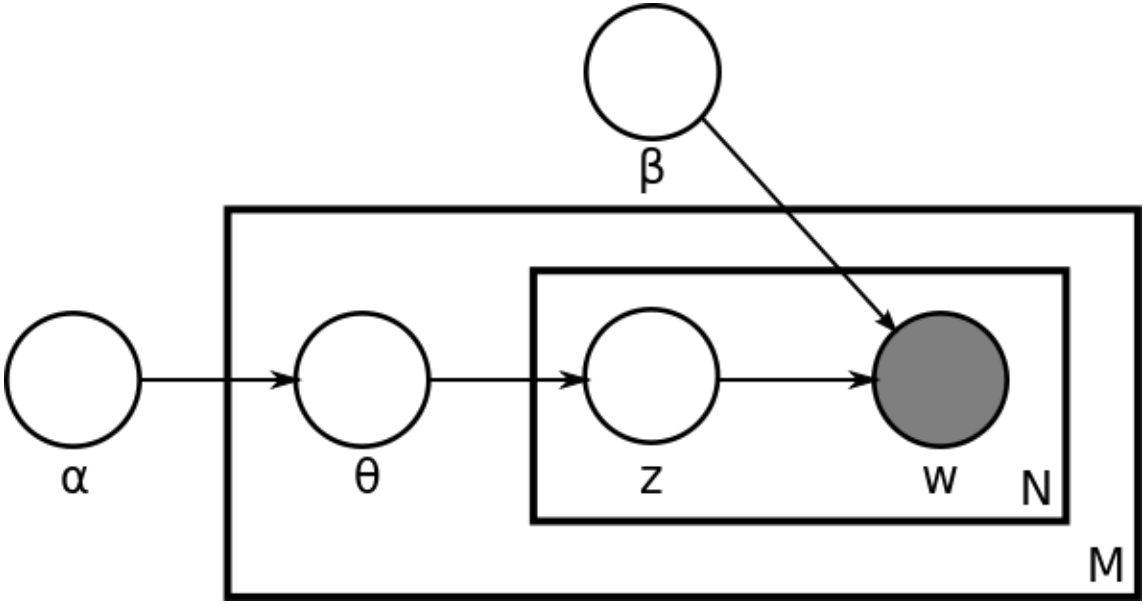
5. Devoxx Belgium
6. Слайды с JFokus VM Tech Summit

Проблемы

- Data Science
- Современные динамические языки
- Много мусора

Латентное размещение Дирихле (LDA)

сходство частей данных по группам

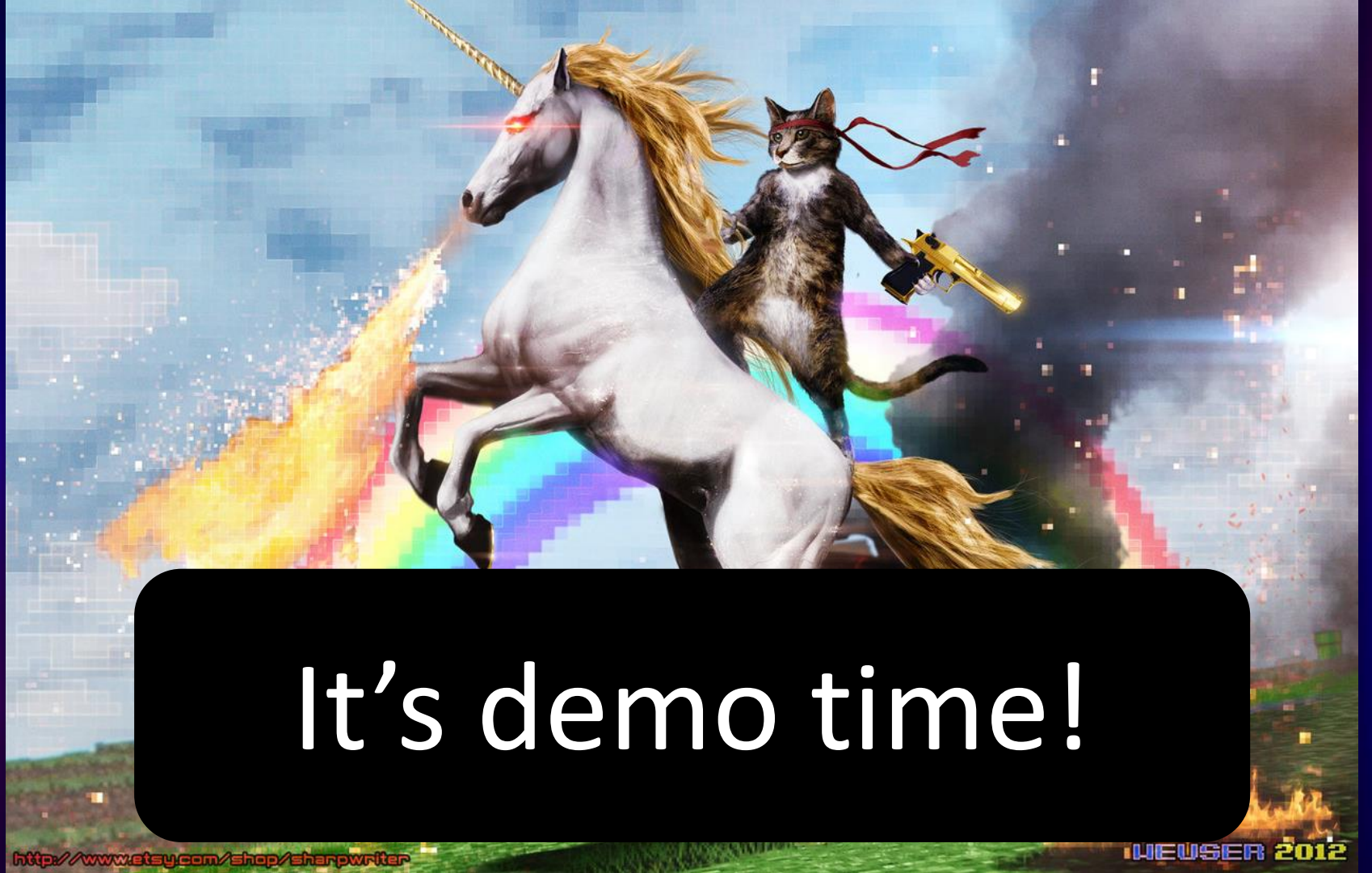


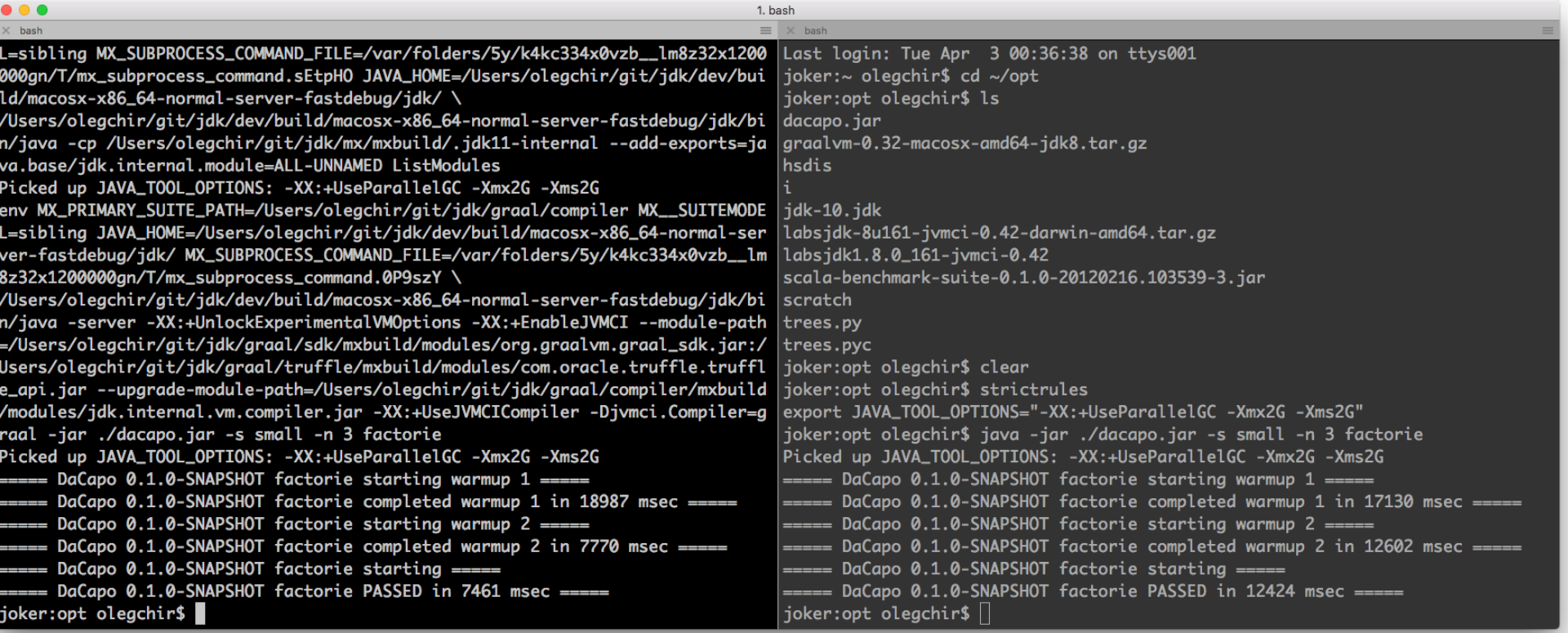
Scala DaCapo

factorie

http://www.scalabench.org/

java -jar ./dacapo.jar -s small -n 3 factorie


It’s demo time!
http://www.etsy.com/shop/sharpwriter
NEUSER 2012



7461 msec

12424 msec

```
===== DaCapo 0.1.0-SNAPSHOT factorie starting =====
[36,157s][info][gc] GC(61) Pause Young (Allocation Failure) 673M->46M(2041M) 356,700m
[36,643s][info][gc] GC(62) Pause Young (Allocation Failure) 715M->47M(2014M) 66,255ms
[37,051s][info][gc] GC(63) Pause Young (Allocation Failure) 688M->47M(2027M) 5,716ms
[37,461s][info][gc] GC(64) Pause Young (Allocation Failure) 688M->47M(2029M) 6,684ms
[37,868s][info][gc] GC(65) Pause Young (Allocation Failure) 691M->47M(2028M) 6,521ms
[38,271s][info][gc] GC(66) Pause Young (Allocation Failure) 691M->47M(2031M) 6,484ms
[38,692s][info][gc] GC(67) Pause Young (Allocation Failure) 694M->49M(2030M) 8,001ms
[39,103s][info][gc] GC(68) Pause Young (Allocation Failure) 697M->49M(2033M) 7,676ms
[39,534s][info][gc] GC(69) Pause Young (Allocation Failure) 701M->49M(2032M) 7,481ms
[39,950s][info][gc] GC(70) Pause Young (Allocation Failure) 701M->49M(2034M) 7,103ms
[40,364s][info][gc] GC(71) Pause Young (Allocation Failure) 704M->49M(2034M) 7,488ms
[40,777s][info][gc] GC(72) Pause Young (Allocation Failure) 704M->49M(2036M) 8,089ms
[41,190s][info][gc] GC(73) Pause Young (Allocation Failure) 707M->49M(2035M) 6,592ms
[41,625s][info][gc] GC(74) Pause Young (Allocation Failure) 707M->49M(2037M) 6,708ms
[42,038s][info][gc] GC(75) Pause Young (Allocation Failure) 710M->49M(2037M) 6,392ms
[42,478s][info][gc] GC(76) Pause Young (Allocation Failure) 710M->49M(2039M) 6,541ms
[42,911s][info][gc] GC(77) Pause Young (Allocation Failure) 713M->49M(2038M) 6,161ms
[43,334s][info][gc] GC(78) Pause Young (Allocation Failure) 713M->49M(2040M) 5,936ms
[43,756s][info][gc] GC(79) Pause Young (Allocation Failure) 715M->49M(2039M) 6,013ms
[44,192s][info][gc] GC(80) Pause Young (Allocation Failure) 715M->49M(2041M) 6,149ms
[44,624s][info][gc] GC(81) Pause Young (Allocation Failure) 717M->49M(2040M) 5,569ms
[45,041s][info][gc] GC(82) Pause Young (Allocation Failure) 717M->49M(2042M) 5,357ms
[45,476s][info][gc] GC(83) Pause Young (Allocation Failure) 719M->49M(2041M) 5,246ms
[45,915s][info][gc] GC(84) Pause Young (Allocation Failure) 719M->49M(2043M) 5,569ms
[46,344s][info][gc] GC(85) Pause Young (Allocation Failure) 721M->49M(2042M) 5,609ms
===== DaCapo 0.1.0-SNAPSHOT factorie PASSED in 12633 msec =====
```

```
===== DaCapo 0.1.0-SNAPSHOT factorie starting =====
[36,195s][info][gc] GC(40) Pause Young (Allocation Failure) 635M->56M(2017M) 371,059ms
[36,933s][info][gc] GC(41) Pause Young (Allocation Failure) 678M->56M(2002M) 295,813ms
[37,378s][info][gc] GC(42) Pause Young (Allocation Failure) 661M->56M(2009M) 12,692ms
[37,837s][info][gc] GC(43) Pause Young (Allocation Failure) 661M->58M(2014M) 13,305ms
[38,307s][info][gc] GC(44) Pause Young (Allocation Failure) 670M->58M(2011M) 13,814ms
[38,754s][info][gc] GC(45) Pause Young (Allocation Failure) 670M->58M(2017M) 12,420ms
[39,222s][info][gc] GC(46) Pause Young (Allocation Failure) 678M->58M(2015M) 12,816ms
[39,677s][info][gc] GC(47) Pause Young (Allocation Failure) 678M->58M(2021M) 12,522ms
[40,127s][info][gc] GC(48) Pause Young (Allocation Failure) 685M->58M(2019M) 11,827ms
[40,606s][info][gc] GC(49) Pause Young (Allocation Failure) 685M->58M(2024M) 12,949ms
[41,074s][info][gc] GC(50) Pause Young (Allocation Failure) 692M->58M(2023M) 12,944ms
[41,530s][info][gc] GC(51) Pause Young (Allocation Failure) 692M->58M(2027M) 11,225ms
===== DaCapo 0.1.0-SNAPSHOT factorie PASSED in 7943 msec =====
```

GRAAL JIT COMPILER

```
-XX:+UnlockExperimentalVMOptions
-XX:+UseJVMCICompiler
-XX:+EnableJVMCI
-Djvmci.Compiler=graal
```

Christian Thalinger

JBreak 2018:
Graal in Real Life

Joker 2017:
Twitter's quest for a Wholly Graal runtime

USER CPU TIME - RATIO

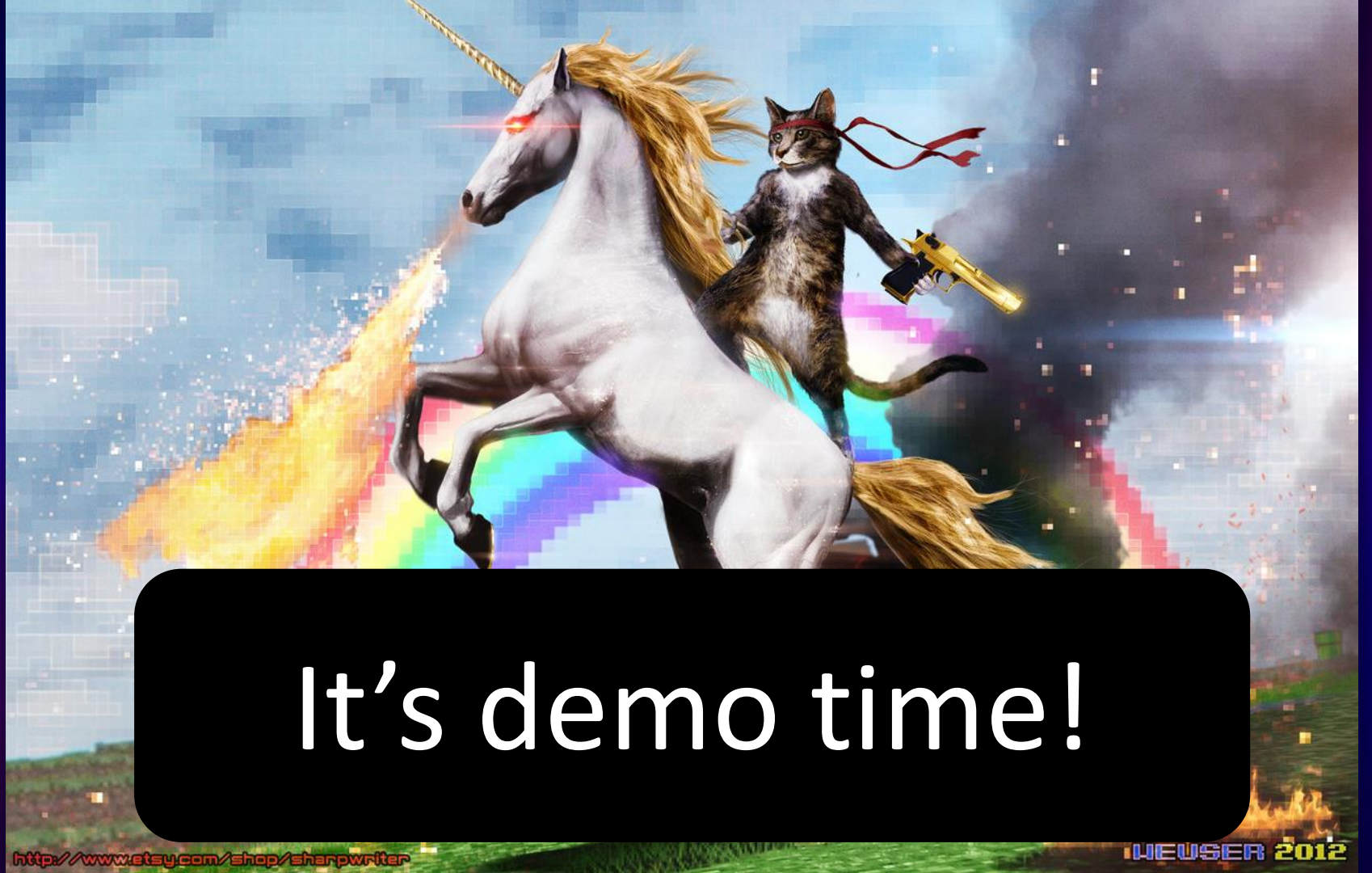
C2 Graal C2 JDK9 Graal JDK9 Ex @CHRISTHALINGER | #TWITTERVMTEAM

Joker 2017: [Twitter's quest for a Wholly Graal runtime]

Владимир Долженко

https://habrahabr.ru/post/351996/

```
private static void checkArgument(
                        boolean isValid,
                        String message,
                        Object ... arguments)
{
    if (!isValid){
            (String.format(message, arguments));
    }
}
```


It’s demo time!
http://www.etsy.com/shop/sharpwriter
NEUSER 2012

Benchmark	(value)	Mode	Cnt	Score	Error	Units
MyBenchmark.allocate	-1	thrpt	3	696308.213 ±	2793894.954	ops/s
MyBenchmark.allocate:·gc.alloc.rate	-1	thrpt	3	476.799 ±	1915.358	MB/sec
MyBenchmark.allocate:·gc.alloc.rate.norm	-1	thrpt	3	1079.954 ±	1.480	B/op
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'	-1	thrpt	3	0.663 ±	0.032	MB/sec
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'.norm	-1	thrpt	3	1.559 ±	7.081	B/op
MyBenchmark.allocate:·gc.churn.G1_Old_Gen	-1	thrpt	3	435.328 ±	1619.685	MB/sec
MyBenchmark.allocate:·gc.churn.G1_Old_Gen.norm	-1	thrpt	3	989.972 ±	1047.090	B/op
MyBenchmark.allocate:·gc.count	-1	thrpt	3	45.000		counts
MyBenchmark.allocate:·gc.time	-1	thrpt	3	194.000		ms
MyBenchmark.allocate	1	thrpt	3	63953472.178 ±	7972940.253	ops/s
MyBenchmark.allocate:·gc.alloc.rate	1	thrpt	3	1310.969 ±	88.302	MB/sec
MyBenchmark.allocate:·gc.alloc.rate.norm	1	thrpt	3	32.063 ±	2.003	B/op
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'	1	thrpt	3	0.660 ±	0.054	MB/sec
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'.norm	1	thrpt	3	0.016 ±	0.001	B/op
MyBenchmark.allocate:·gc.churn.G1_Old_Gen	1	thrpt	3	1171.104 ±	4422.376	MB/sec
MyBenchmark.allocate:·gc.churn.G1_Old_Gen.norm	1	thrpt	3	28.643 ±	108.154	B/op
MyBenchmark.allocate:·gc.count	1	thrpt	3	56.000		counts
MyBenchmark.allocate:·gc.time	1	thrpt	3	512.000		ms

C2

Создали мусора за операцию

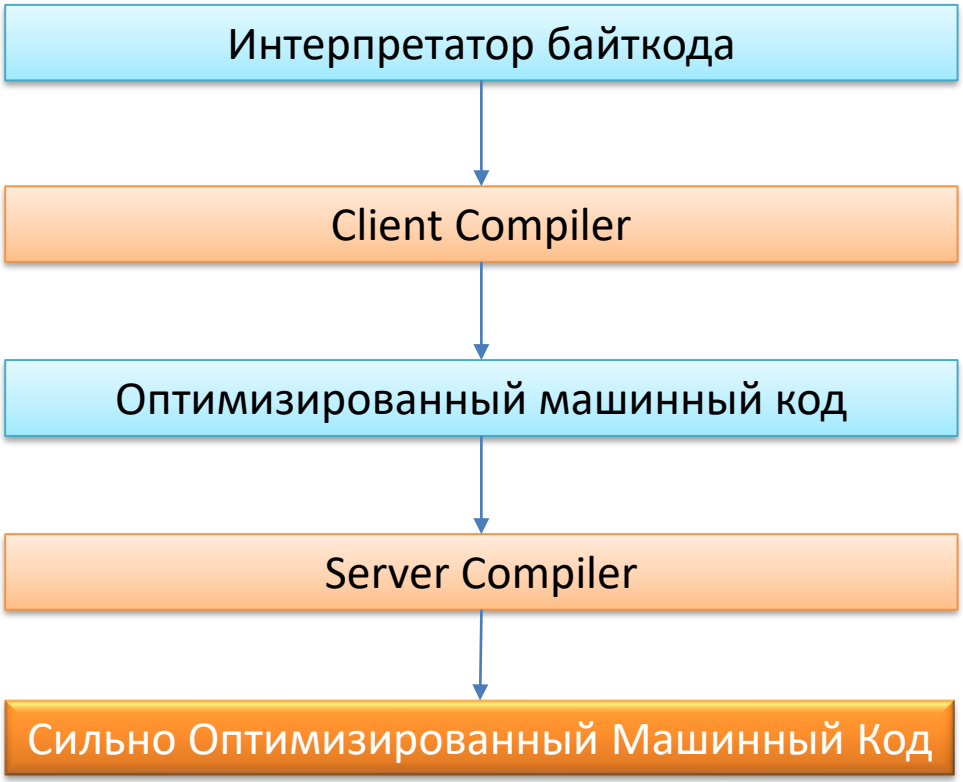
1079 или 32 байта

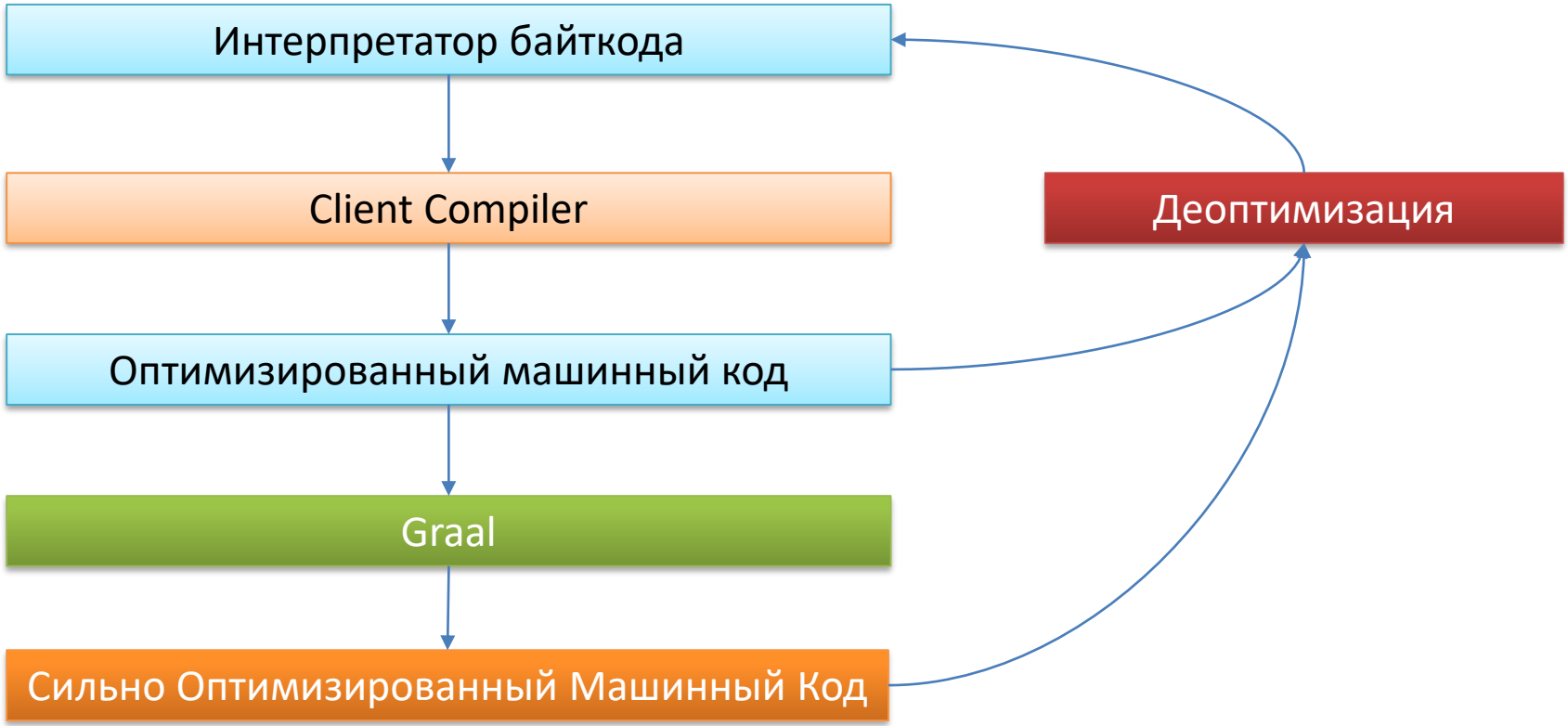
Benchmark	(value)	Mode	Cnt	Score	Error	Units
MyBenchmark.allocate	-1	thrpt	3	237794.268 ±	3696713.408	ops/s
MyBenchmark.allocate:·gc.alloc.rate	-1	thrpt	3	197.686 ±	2623.288	MB/sec
MyBenchmark.allocate:·gc.alloc.rate.norm	-1	thrpt	3	1490.626 ±	5954.845	B/op
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'	-1	thrpt	3	0.677 ±	6.713	MB/sec
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'.norm	-1	thrpt	3	9.684 ±	211.412	B/op
MyBenchmark.allocate:·gc.churn.G1_Old_Gen	-1	thrpt	3	194.483 ±	2830.895	MB/sec
MyBenchmark.allocate:·gc.churn.G1_Old_Gen.norm	-1	thrpt	3	1334.354 ±	1679.830	B/op
MyBenchmark.allocate:·gc.churn.G1_Survivor_Space	-1	thrpt	3	0.221 ±	6.986	MB/sec
MyBenchmark.allocate:·gc.churn.G1_Survivor_Space.norm	-1	thrpt	3	0.777 ±	24.547	B/op
MyBenchmark.allocate:·gc.count	-1	thrpt	3	16.000		counts
MyBenchmark.allocate:·gc.time	-1	thrpt	3	507.000		ms
MyBenchmark.allocate	1	thrpt	3	1565252119.556 ±	10305151632.296	ops/s
MyBenchmark.allocate:·gc.alloc.rate	1	thrpt	3	19.381 ±	612.409	MB/sec
MyBenchmark.allocate:·gc.alloc.rate.norm	1	thrpt	3	0.033 ±	1.056	B/op
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'	1	thrpt	3	0.145 ±	4.593	MB/sec
MyBenchmark.allocate:·gc.churn.CodeHeap_'non-profiled_nmethods'.norm	1	thrpt	3	≈ 10^{-4}		B/op
MyBenchmark.allocate:·gc.churn.G1_Old_Gen	1	thrpt	3	13.045 ±	412.216	MB/sec
MyBenchmark.allocate:·gc.churn.G1_Old_Gen.norm	1	thrpt	3	0.022 ±	0.711	B/op
MyBenchmark.allocate:·gc.count	1	thrpt	3	2.000		counts
MyBenchmark.allocate:·gc.time	1	thrpt	3	36.000		ms

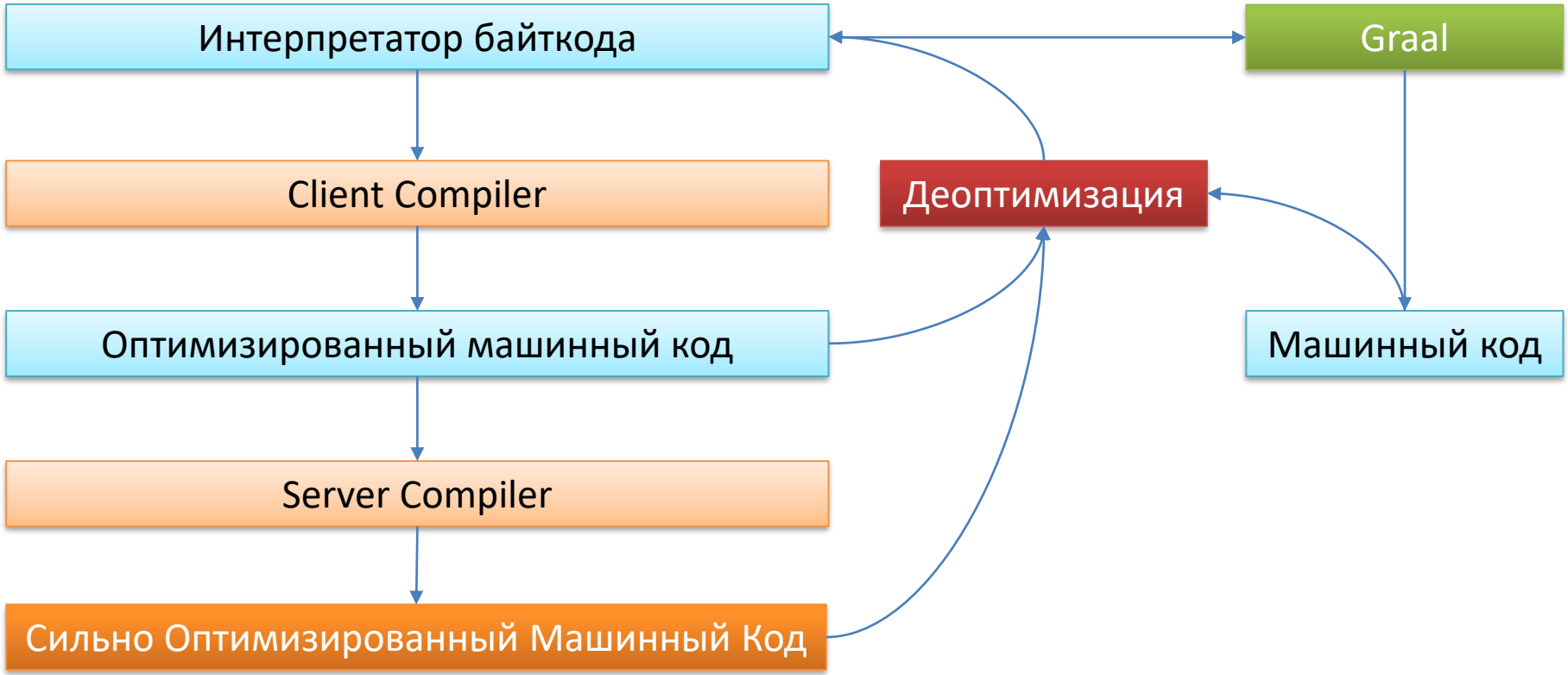
Graal

Создали мусора за операцию

1490 или 0.033 байта


Интерпретатор байткода
Client Compiler
Оптимизированный машинный код
Server Compiler
Сильно Оптимизированный Машинный Код


Интерпретатор байткода
Client Compiler
Оптимизированный машинный код
Graal
Сильно Оптимизированный Машинный Код
Деоптимизация


Интерпретатор байткода
Graal
Client Compiler
Деоптимизация
Оптимизированный машинный код
Машинный код
Server Compiler
Сильно Оптимизированный Машинный Код

JEP 243: Java-Level JVM Compiler Interface

http://openjdk.java.net/jeps/243

JEP 317: Experimental Java-Based JIT Compiler

http://openjdk.java.net/jeps/317

```
public interface JVMCICompiler {
    int INVOCATION_ENTRY_BCI = -1;

    /**
     * Services a compilation request. This object should compile the method to machine code and
     * install it in the code cache if the compilation is successful.
     */
    CompilationRequestResult compileMethod(CompilationRequest request);
}
```

```
public class CompilationRequest {

   private final ResolvedJavaMethod method;

   /**
    * Gets the method to be compiled.
    */
   public ResolvedJavaMethod getMethod() {
      return method;
   }

}
```

```
public interface ResolvedJavaMethod extends JavaMethod, InvokeTarget, ModifiersProvider, AnnotatedElement {

byte[] getCode();

int getCodeSize();
ResolvedJavaType getDeclaringClass();
int getMaxLocals();
int getMaxStackSize();
default boolean isFinal();
boolean isSynthetic();
boolean isVarArgs();
boolean isBridge();
boolean isDefault();
boolean isClassInitializer();
boolean isConstructor();
boolean canBeStaticallyBound();
ExceptionHandler[] getExceptionHandlers();
StackTraceElement asStackTraceElement(int      );
default ProfilingInfo getProfilingInfo()
```

```
final class CompilerToVM {

    /**
     * Installs the result of a compilation into the code cache.
     *
     * @param target the target where this code should be installed
     * @param compiledCode the result of a compilation
     * @param code the details of the installed CodeBlob are written to this object
     * @return the outcome of the installation which will be one of
    native int installCode(TargetDescription target,
                           HotSpotCompiledCode compiledCode,
                           InstalledCode code,
                           HotSpotSpeculationLog speculationLog);
```

src/jdk.internal.vm.ci/share/classes/jdk.vm.ci.hotspot/src/jdk/vm/ci/hotspot/CompilerToVM.java

```
-XX:+UnlockExperimentalVMOptions
-XX:+UseJVMCICompiler
-XX:+EnableJVMCI
-Djvmci.Compiler=graal
```

JDK 11 - не всё адаптировано
JDK 8 - устарел

mx igv
Error occurred during initialization of boot layer
java.lang.module.FindException: Module java.xml.bind not found

Joker<?> 2017

Chris Seaton

Oracle Labs

Understanding how Graal works — a Java JIT compiler written in Java

http://2017.jokerconf.com/2017/talks/ghdvtsu3y60qai68waayi/

https://github.com/oracle/graal/blob/master/docs/Publications.md

https://gitter.im/graalvm/graal-core

YouTube по запросу GraalVM

Отдел разработки лабораторного кластера супермассивов

Интервью с Валерием Выборновым на Хабре:

- Scala, Akka, Hadoop, Spark
- Data Science
- Big Data

Задача: интеграция JS фронта

- Легаси бэкенд на Java
  «кровавый энтерпрайз»

- Новый фронт на JavaScript + ReactJS
- Как организовать интероп?

Зачем вставать два раза?

Один валидатор на сервере и клиенте

Валидатор (shared.js)

```
var validate = function(target) {
  if (target > 0) {
    console.log("success");
  } else {
    console.log("fail");
  }
};
```

Цель

```
validate(1);
```

- Браузер
- Node.JS
- Java

Node.JS

```
var fs = require('fs');
var vm = require('vm');
var includeInThisContext = function(path) {
    var code = fs.readFileSync(path);
    vm.runInThisContext(code, path);
}.bind(this);
includeInThisContext(__dirname+"/shared.js");

validate(1);
```

Java

```
import org.graalvm.polyglot.*; //Nashorn тоже будет работать

String code = "validate(1);";

String shared = readFile(SHARED_JS) + "\n\n";
String poly = readFile(NASHORN_POLYFILL_JS) + "\n\n";

String codeAsFunction = String.format("function() { %s return true; }", code);
String actualCode = poly + shared + codeAsFunction;
System.out.println(actualCode);

Context context = Context.create();
Value function = context.eval("js", actualCode);
function.execute().asBoolean();
```

polyfill.js (Nashorn, Graal)

```
var global = this;
var window = this;
var process = {env:{}};

var console = {};
console.debug = print;
console.log = print;
console.warn = print;
console.error = print;
```

Зачем вставать два раза?

ЕСЛИ МОЖНО НЕ ВСТАВАТЬ ВООБЩЕ

Для генерации фронта всё еще нужна Java

Зачем нужна Java для генерации фронта?

- Генерация стабов и прокси-классов для общения по HTTP
- Состав поставки приложения (набор фич)
- Статическая генерация по Java-источникам (Hibernate)
- Перенос данных об окружении (номер версии из Maven)
- Причины возникают по ходу дела

Способы решения

- Сложные пайплайны с Maven и Gradle
- Общие properties-файлы

Способы решения

- Сложные пайплайны с Maven и Gradle
- Общие properties-файлы

- Вызвать Java из JavaScript!

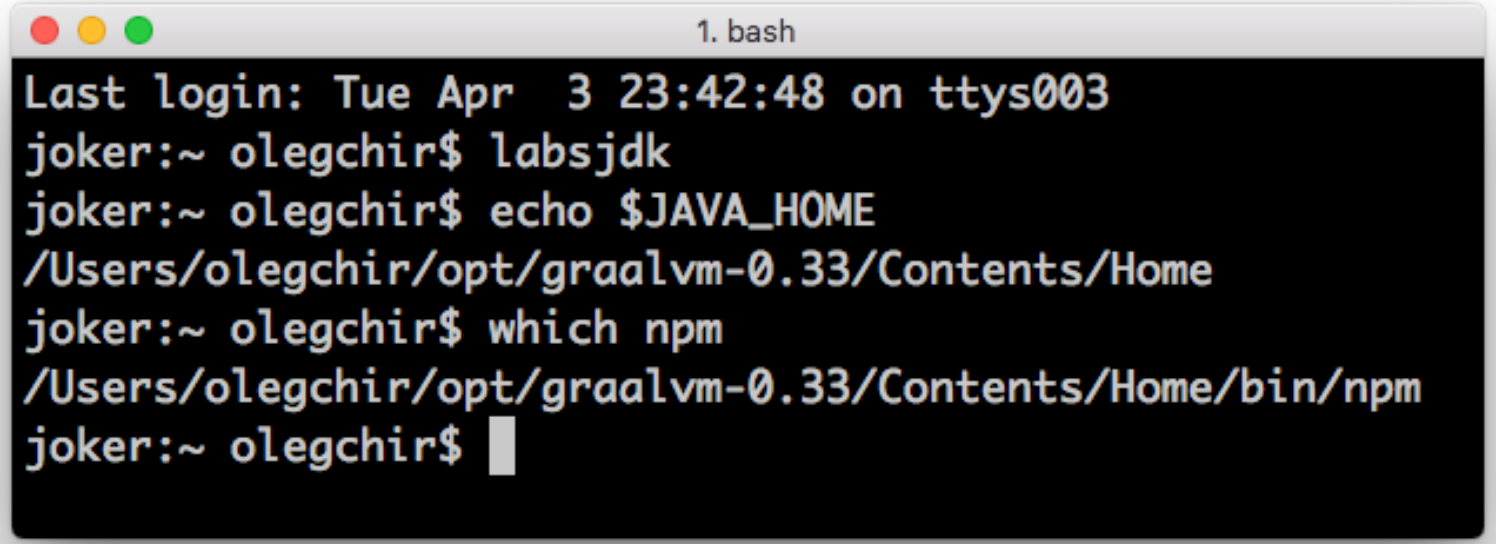
Эксперимент: React, Npm, Babel, Webpack

Загружаем GraalVM:
http://www.oracle.com/technetwork/oracle-labs/program-languages/downloads/index.html

Устанавливаем переменные окружения:

```
export LABSJDK=/Users/olegchir/opt/graalvm-0.33/Contents/Home
export LABSJRE=/Users/olegchir/opt/graalvm-0.33/Contents/Home/jre
export JAVA_HOME=$LABSJDK
export JRE_HOME=$LABSJRE
export JDK_HOME=$JAVA_HOME
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
```

Эксперимент: React, Npm, Babel, Webpack



Привет, JPoint!

Количество дичи уже сказано : 4

Сказать ещё дичи!

src/client/app/index.jsx

```
import React from 'react';
import {render} from 'react-dom';
import JPointComponent from './JPointComponent.jsx';

class App extends React.Component {
  render () {
    return (
      <div>
        <p>Привет, JPoint!</p>
        <JPointComponent />
      </div>
    );
  }
}
render(<App/>, document.getElementById('app'));
```

src/client/app/JPointComponent.jsx

```
render() {
  return (
    <div>
      Количество дичи уже сказано : <span>{this.state.dich}</span>
      <div><button onClick={this.onDich}>{this.state.button}</button></div>
    </div>
  );
 }
}
export default JPointComponent;
```

src/client/app/JPointComponent.jsx

```
import React from 'react';

class JPointComponent extends React.Component {

  constructor(props) {
    super(props);

    this.state = {dich : 0, button: buttonCaption};
    this.onDich = this.onDich.bind(this);
  }

  onDich () {
    let newDich = this.state.dich + 1;

    this.setState({dich: newDich, button: buttonCaption});
  }
```

loaders/preload.js

```
const loaderUtils = require("loader-utils"),
      schemaUtils = require("schema-utils");

module.exports = function main(source) {
  this.cacheable();

  var MyString = Java.type("java.lang.String");
  var data = new MyString("Сказать ещё дичи!");

  return `window.buttonCaption=\"${data}\";` + `${source}`;
};
```

И обычная JS рутина…

Каркас проекта

```
npm init
npm i webpack webpack-cli webpack-dev-server –S
npm i babel-core babel-loader
        babel-preset-es2015 babel-preset-react –S
npm i react react-dom -S

mkdir -p src/client/app
mkdir -p src/client/public
mkdir -p loaders
```

webpack.config.js

```
var p = require('path');
var webpack = require('webpack');
var BUILD_DIR = p.resolve(__dirname, 'src/client/public');
var APP_DIR = p.resolve(__dirname, 'src/client/app');

let defaults = {
   output: { path: BUILD_DIR, filename: 'bundle.js' },
   entry: APP_DIR + '/index.jsx',
   module : { rules : [ { test : /\.jsx?/, include : APP_DIR, loader : 'babel-loader' } ] },
   resolveLoader: { modules: ['node_modules', p.resolve(__dirname, 'loaders')] }
};

module.exports = function (content) {
   let dd = defaults;
   defaults.module.rules.push({ test : /index\.jsx/,  loader: "preload", options: {} });
   return dd;
};
```

src/client/index.html

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Hello JPoint!</title>
 </head>
 <body>
  <div id="app" />
  <script src="public/bundle.js" type="text/javascript"></script>
 </body>
</html>
```

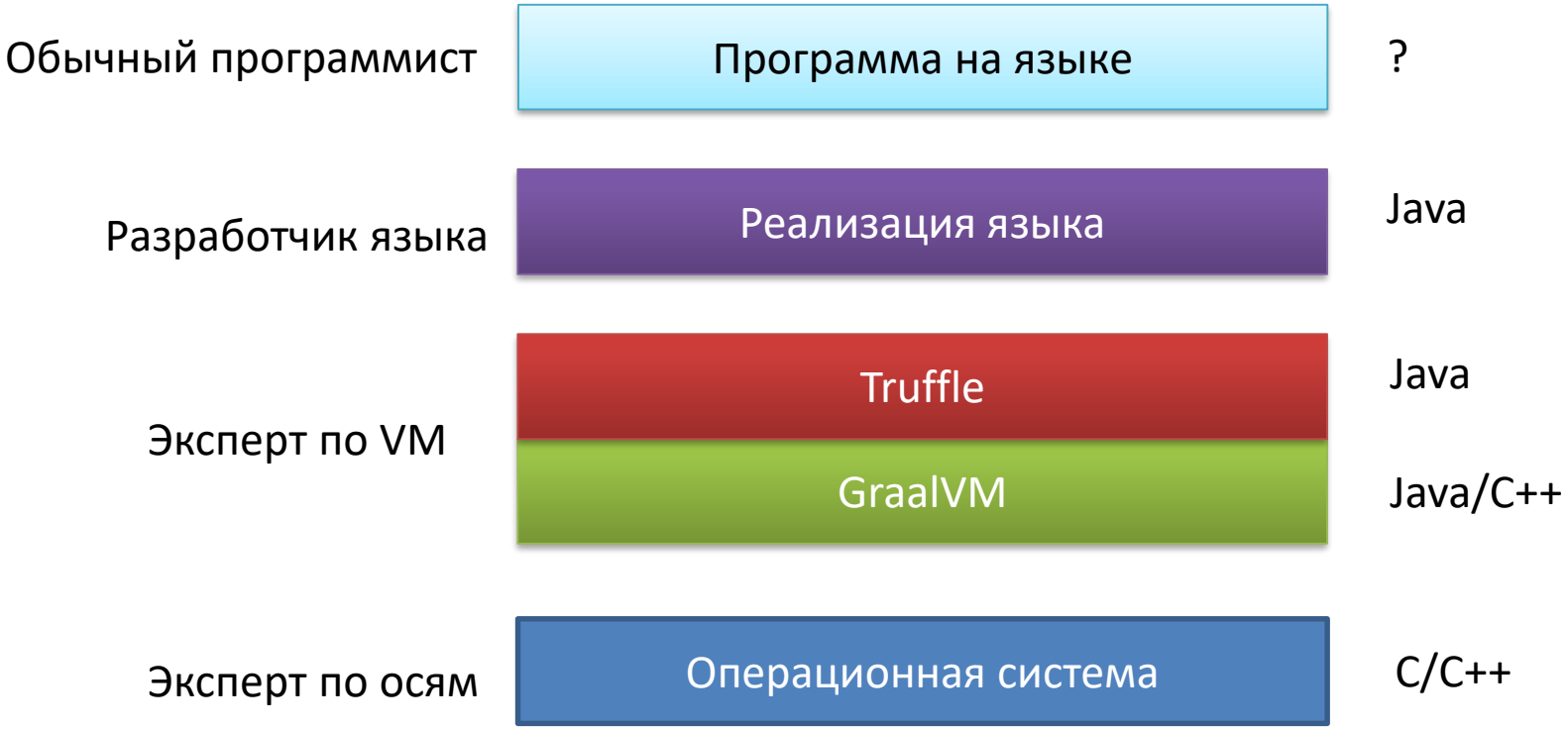
./.babelrc

```
{
  "presets" : ["es2015", "react"]
}
```

node --jvm node_modules/.bin/webpack -d

Кому это нужно?

(Сбертех и эпический квест по переходу на React)



Graal.JS - high-performance JavaScript on the JVM by Christian Wirth

Основные проблемы

- Использование наиболее подходящего языка
  (можно смешивать произвольные языки)
- Миграция между языками, поддержка легаси
  (взять две существующие кодовые базы и развивать их параллельно)
  (без написания костыльных переходников и врапперов)
- Избежать потерь при пересечении границы языка

Доступные языки в GraalVM

- Java
- JavaScript
- Ruby
- R

Доступные языки вообще

- C, C++, Fortran (Sulong – LLVM bitcode)
- Python (ZipPy)
- Можно написать свой

[One VM to Rule Them All by Thomas Wuerthinger](https://gist.github.com/smarr/d1f8f2101b5cc8e14e12)
https://gist.github.com/smarr/d1f8f2101b5cc8e14e12

```
NodeInfo(shortName = "||")
public final class SLLogicalOrNode extends SLShortCircuitNode {

    public SLLogicalOrNode(SLExpressionNode left, SLExpressionNode right) {
        super(left, right);
    }

    @Override
    protected boolean isEvaluateRight(boolean left) {
        return !left;
    }

    @Override
    protected boolean execute(boolean left, boolean right) {
        return left || right;
    }
}
```

https://github.com/graalvm/simplelanguage

- Можем смешивать языки
- Можем написать свой собственный

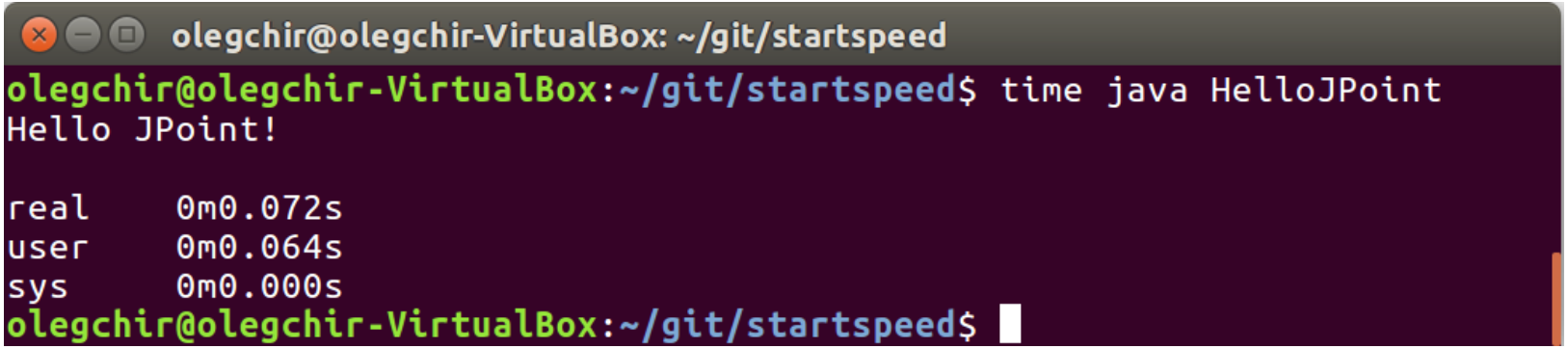

MAPFRE
ultimate
27
Bell & Ross
Computacenter
HECHTER

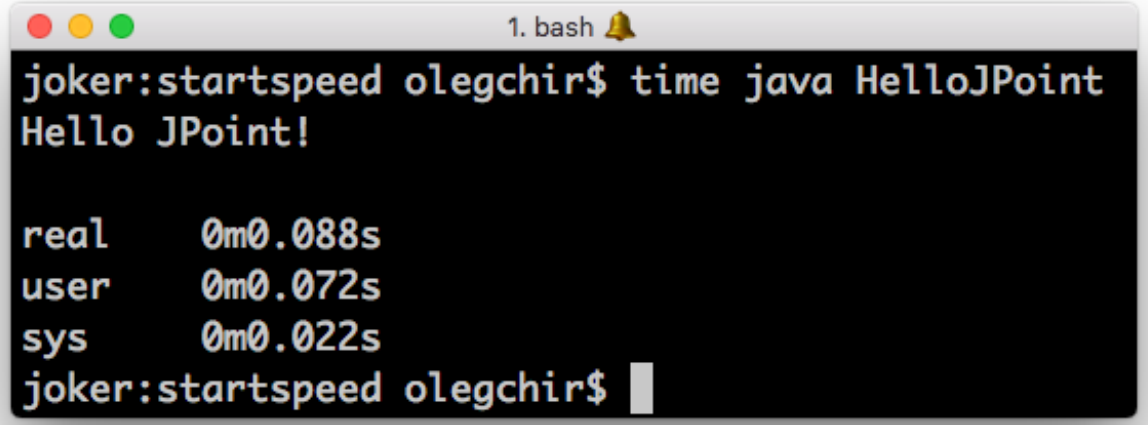
Скорость запуска

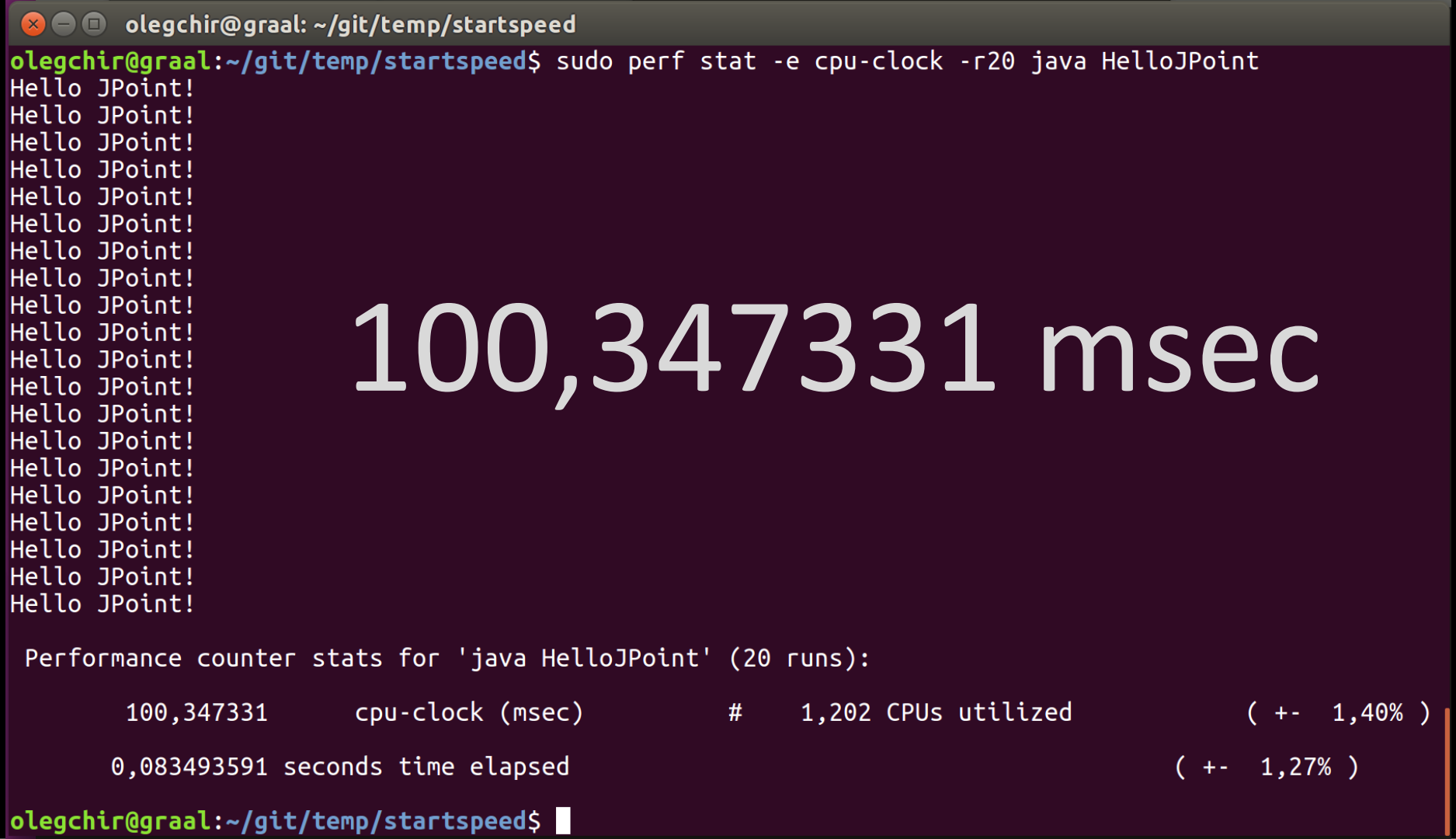
- Контейнеры (Docker, LXD, Vagrant-LXC, …)
- Консольные утилиты и GUI
- Пульт управления, использующий готовый код

AOT & CDS

```java
public class HelloJPoint {
    public static void main(String... args) {
        System.out.println("Hello JPoint!");
    }
}
```

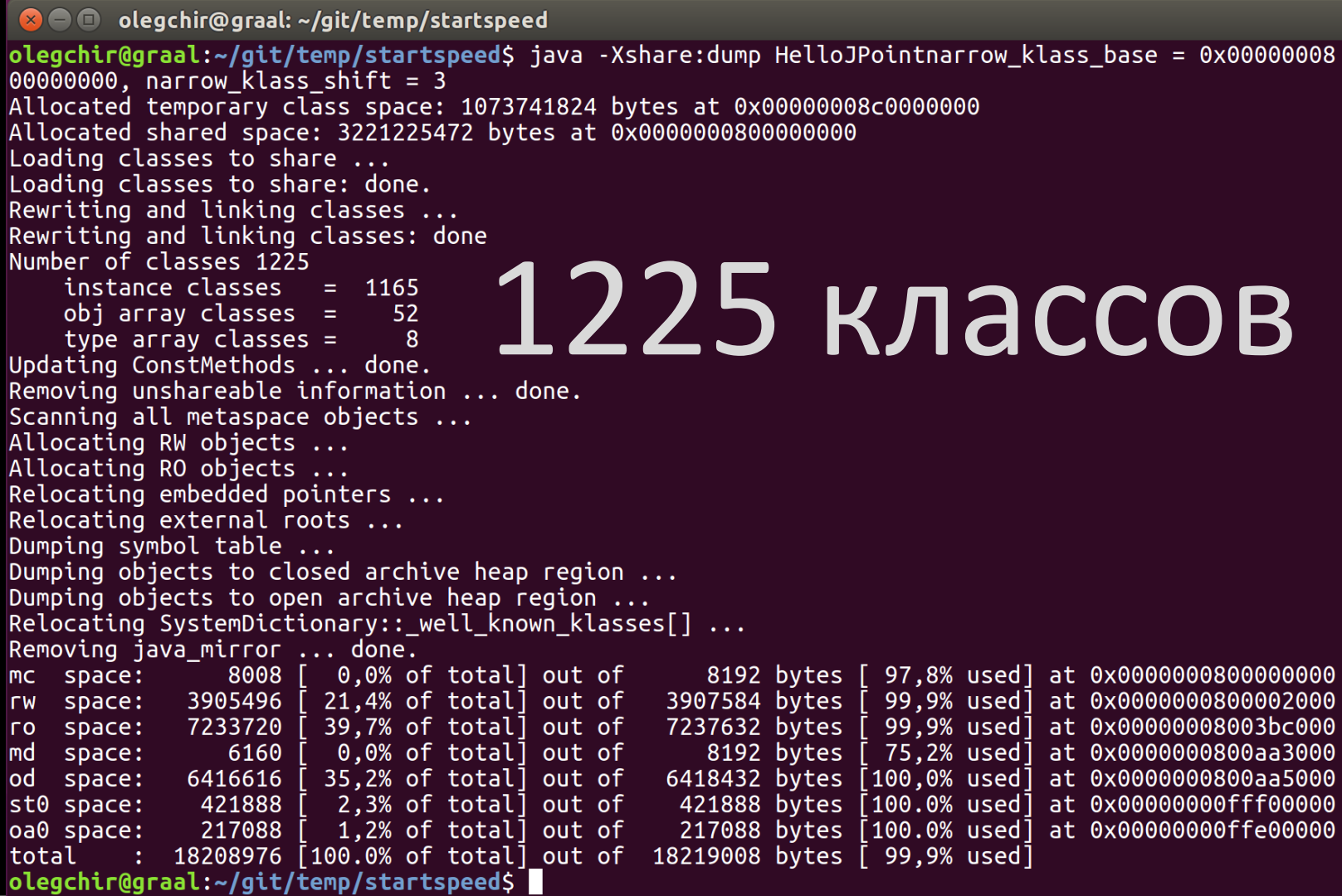

olegchir@olegchir-VirtualBox: ~/git/startspeed
olegchir@olegchir-VirtualBox:~/git/startspeed$ time java HelloJPoint
Hello JPoint!
real 0m0.072s
user 0m0.064s
sys 0m0.000s
olegchir@olegchir-VirtualBox:~/git/startspeed$

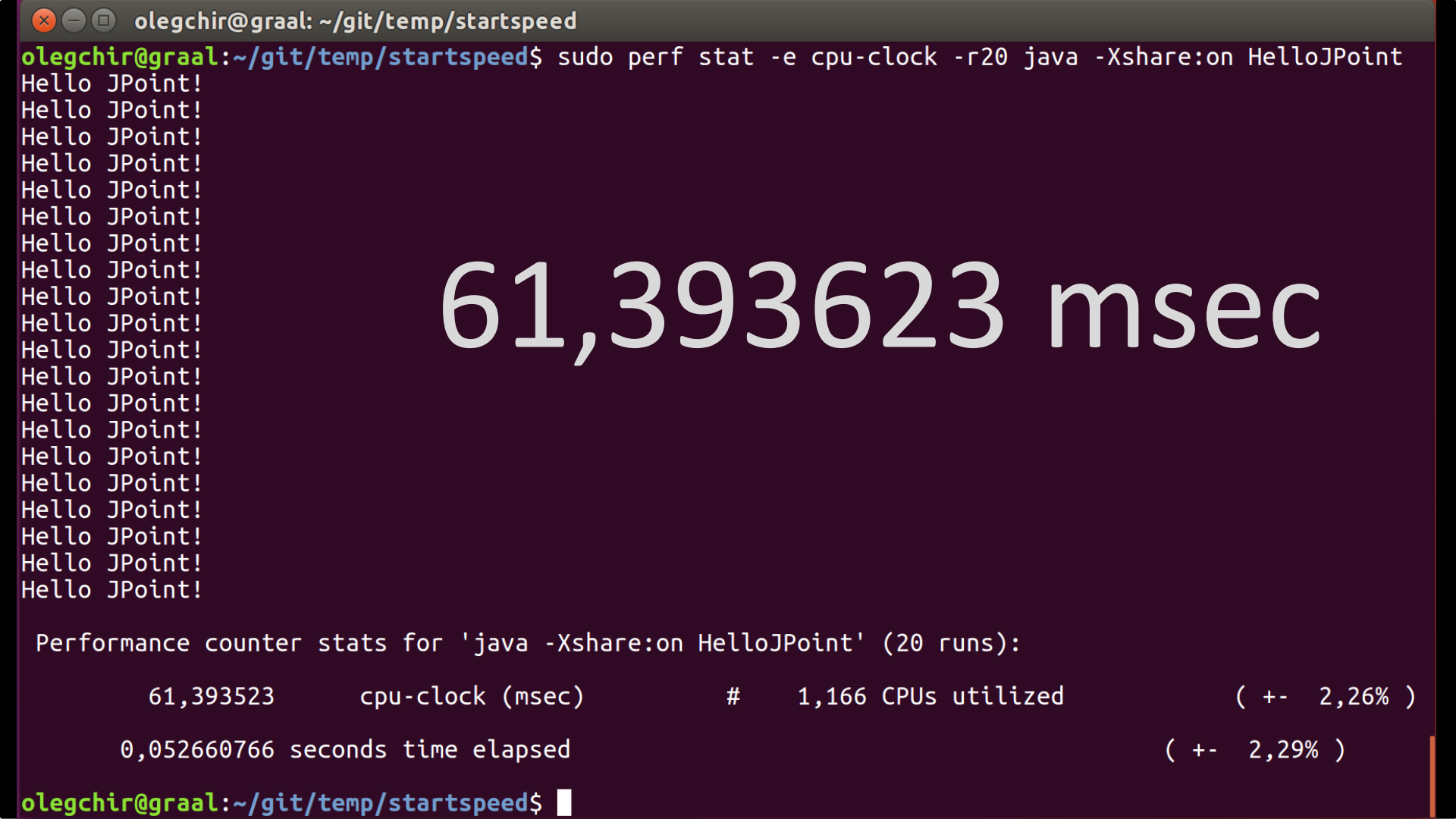

1. bash
joker:startspeed olegchir$ time java HelloJPoint
Hello JPoint!
real 0m0.088s
user 0m0.072s
sys 0m0.022s
joker:startspeed olegchir$


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ sudo perf stat -e cpu-clock -r20 java HelloJPoint
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
100,347331 msec
Performance counter stats for 'java HelloJPoint' (20 runs):
100,347331 cpu-clock (msec) # 1,202 CPUs utilized (+- 1,40%)
0,083493591 seconds time elapsed (+- 1,27%)
olegchir@graal:~/git/temp/startspeed$

Java 8: Class Data Sharing

1.-Xshare:dump
2.-Xshare:on


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ java -Xshare:dump HelloJPointnarrow_klass_base = 0x00000008
00000000, narrow_klass_shift = 3
Allocated temporary class space: 1073741824 bytes at 0x00000008c0000000
Allocated shared space: 3221225472 bytes at 0x0000000800000000
Loading classes to share ...
Loading classes to share: done.
Rewriting and linking classes ...
Rewriting and linking classes: done
Number of classes 1225
instance classes = 1165
obj array classes = 52
type array classes = 8
1225 классов
Updating ConstMethods ... done.
Removing unshareable information ... done.
Scanning all metaspace objects ...
Allocating RW objects ...
Allocating RO objects ...
Relocating embedded pointers ...
Relocating external roots ...
Dumping symbol table ...
Dumping objects to closed archive heap region ...
Dumping objects to open archive heap region ...
Relocating SystemDictionary::_well_known_klasses[] ...
Removing java_mirror ... done.
mc space: 8008 [0,0% of total] out of 8192 bytes [97,8% used] at 0x0000000800000000
rw space: 3905496 [21,4% of total] out of 3907584 bytes [99,9% used] at 0x0000000800002000
ro space: 7233720 [39,7% of total] out of 7237632 bytes [99,9% used] at 0x00000008003bc000
md space: 6160 [0,0% of total] out of 8192 bytes [75,2% used] at 0x0000000800aa3000
od space: 6416616 [35,2% of total] out of 6418432 bytes [100,0% used] at 0x0000000800aa5000
st0 space: 421888 [2,3% of total] out of 421888 bytes [100.0% used] at 0x00000000fff00000
oa0 space: 217088 [1,2% of total] out of 217088 bytes [100.0% used] at 0x00000000ffe00000
total : 18208976 [100.0% of total] out of 18219008 bytes [99,9% used]
olegchir@graal:~/git/temp/startspeed$


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ sudo perf stat -e cpu-clock -r20 java -Xshare:on HelloJPoint
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
61,393623 msec
Performance counter stats for 'java -Xshare:on HelloJPoint' (20 runs):
61,393523 cpu-clock (msec) # 1,166 CPUs utilized (+- 2,26%)
0,052660766 seconds time elapsed (+- 2,29%)
olegchir@graal:~/git/temp/startspeed$

61 msec (CDS) < 100 msec (база)

Всегда дает прирост, можно включать по умолчанию

Graal AOT!

```
java -XX:+UnlockDiagnosticVMOptions \
-XX:+LogTouchedMethods \
-XX:+PrintTouchedMethodsAtExit \
HelloJPoint > methods
```

Java 9+

баг в трекере обсуждение

Hello JPoint!
Method::print_touched_methods version 1

java/lang/String.indexOf:(Ljava/lang/String;I)I
java/lang/String.getBytes:([BIB)V
java/io/FilePermission.<clinit>:()V

1708 строк

java/lang/String.indexOf:(Ljava/lang/String;I)I

compileOnly java.lang.String.indexOf(Ljava/lang/String;I)I

```
public class Convert {
  public static void main(String args[])
throws Throwable {
    int i; boolean inParams = false;
    while ((i = System.in.read()) >= 0) {
      switch (i) {
        case ':': continue; // skip
        case '/': if (!inParams) { i = '.'; } break;
        case '(': inParams = true; break;
        case '\n':
        case '\r': inParams = false; break;
      }
      System.out.write(i);
    }
  }
}
```

@mjg123

Matthew Gilliard

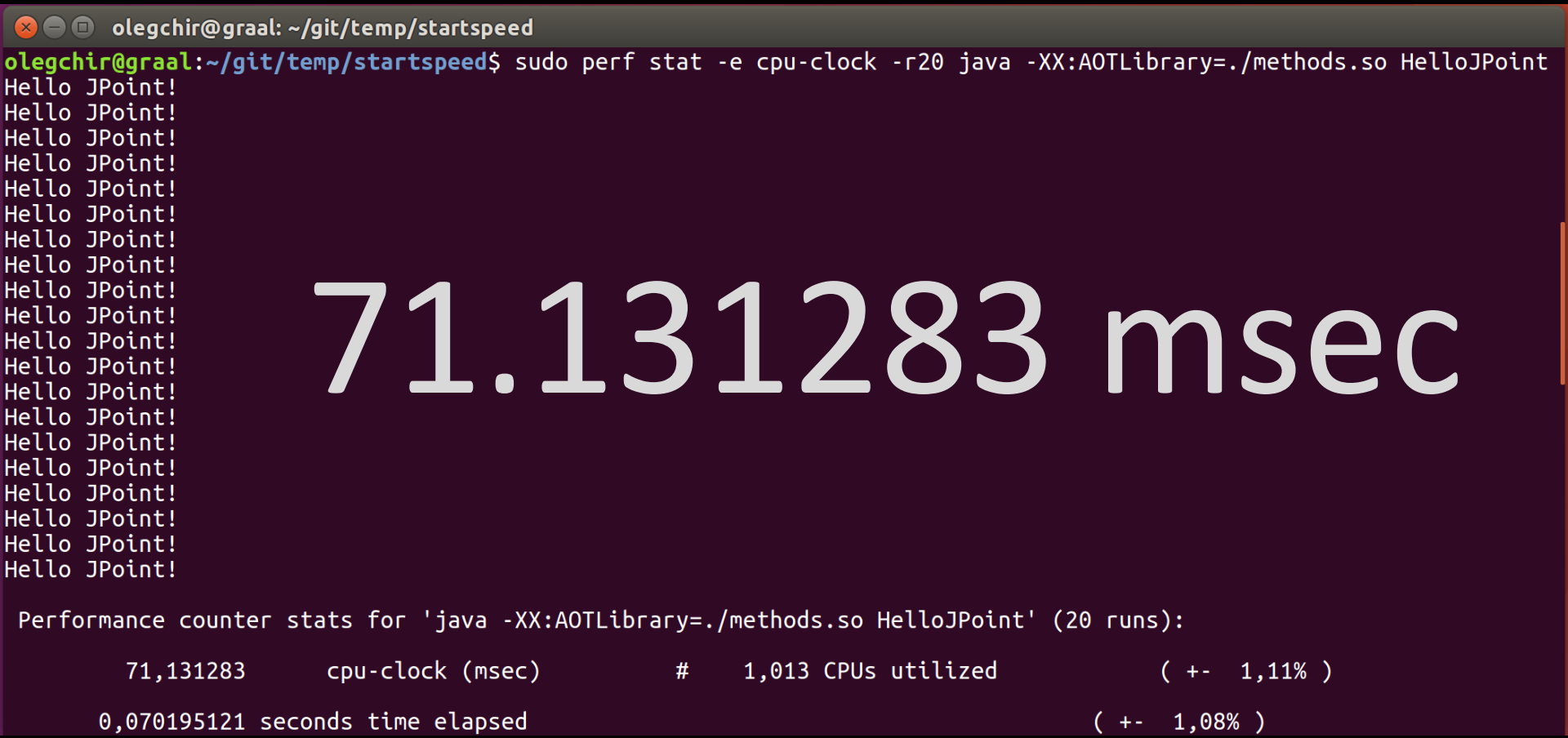
jdk/internal/module/SystemModules.hashes
jdk/internal/module/SystemModules.descriptors

```
grep -v 'Hello JPoint! ' methods | \
grep -v '^#' | \
grep -v jdk/internal/module/SystemModules.hashes | \
grep -v jdk/internal/module/SystemModules.descriptors | \
sed -e 's/^/compileOnly /' | \ java Convert > touched.aotcfg
```

```
jaotc --output methods.so \
        --compile-commands touched.aotcfg \
        --module java.base \
        --class-name HelloJPoint.class \
        --info
```


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ jaotc --output touched_methods.so \
>   --compile-commands touched.aotcfg \
>   --module java.base \
>   --class-name HelloJPoint.class \
>   --info
Compiling touched_methods.so...
5769 classes found (403 ms)
55051 methods total, 1605 methods to compile (559 ms)
Compiling with 2 threads
....
1605 methods compiled, 0 methods failed (19862 ms)
Parsing compiled code (78 ms)
Processing metadata (508 ms)
Preparing stubs binary (1 ms)
Preparing compiled binary (6 ms)
Creating binary: touched_methods (187 ms)
Creating shared library: touched_methods.so (157 ms)
Total time: 22794 ms
olegchir@graal:~/git/temp/startspeed$

```
sudo perf stat -e cpu-clock –r20 \
        java -XX:AOTLibrary=./methods.so \
        HelloJPoint
```


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ sudo perf stat -e cpu-clock -r20 java -XX:AOTLibrary=./methods.so HelloJPoint
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
71.131283 msec
Performance counter stats for 'java -XX:AOTLibrary=./methods.so HelloJPoint' (20 runs):
71,131283 cpu-clock (msec) # 1,013 CPUs utilized (+- 1,11%)
0,070195121 seconds time elapsed (+- 1,08%)

AOT + CDS?

olegchir@graal: ~/git/temp/startspeed

```
java: Aborted
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x00007fbf8d1fb3ae, pid=17352, tid=17353
#
# JRE version:  (11.0+7) (build )
# Java VM: OpenJDK 64-Bit Server VM (11-ea+7, mixed mode, aot, sharing, tiered, compressed oops, g1 gc, linux-amd64
)
# Problematic frame:
# V  [libjvm.so+0x6983ae]  AccessInternal::PostRuntimeDispatch<G1BarrierSet::AccessBarrier<1146998ul, G1BarrierSet>
, (AccessInternal::BarrierType)1, 1146998ul>::oop_access_barrier(oopDesc*, long, oopDesc*)+0x1e
#
# Core dump will be written. Default location: Core dumps may be processed with "/usr/share/apport/apport %p %s %c
%d %P" (or dumping to /home/olegchir/git/temp/startspeed/core.17352)
#
# An error report file with more information is saved as:
# /home/olegchir/git/temp/startspeed/hs_err_pid17352.log
#
# If you would like to submit a bug report, please visit:
#   http://bugreport.java.com/bugreport/crash.jsp
#
java: Aborted
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x00007fe9ac0553ae, pid=17360, tid=17361
#
# JRE version:  (11.0+7) (build )
```

61 (CDS) < 71 (AOT) < 100 (база)

AOT не всегда дает прирост, нужно включать с умом

Что почитать по теме?

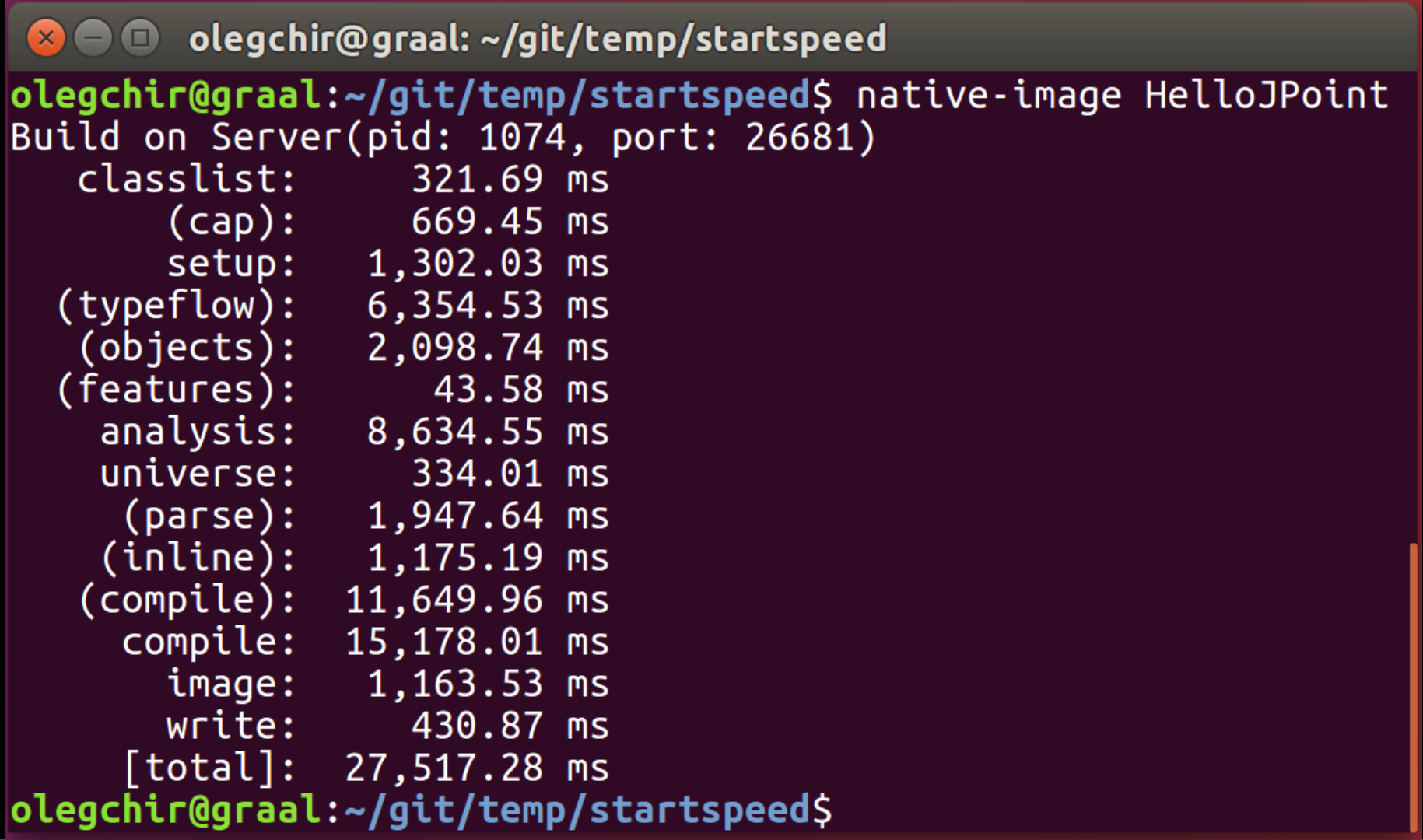
Matthew Gilliard

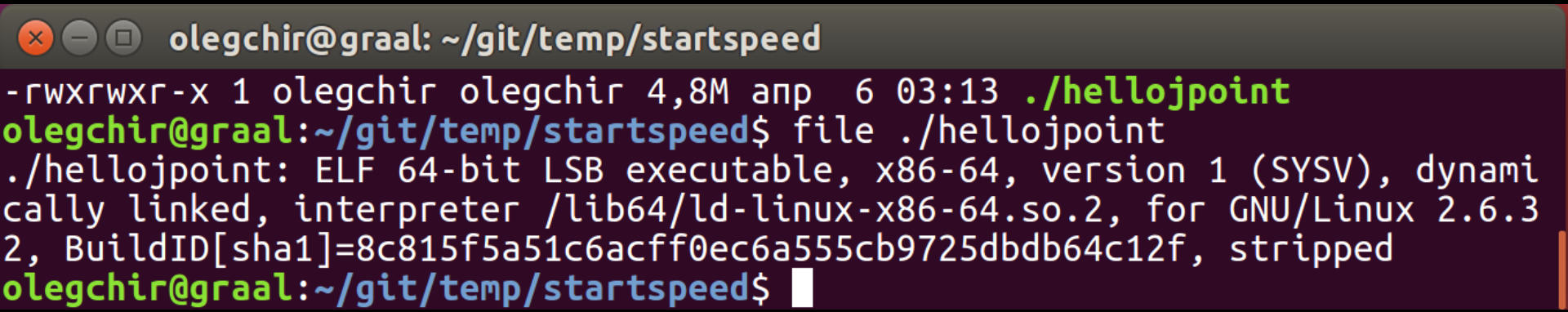
Fast JVM startup with JDK 9

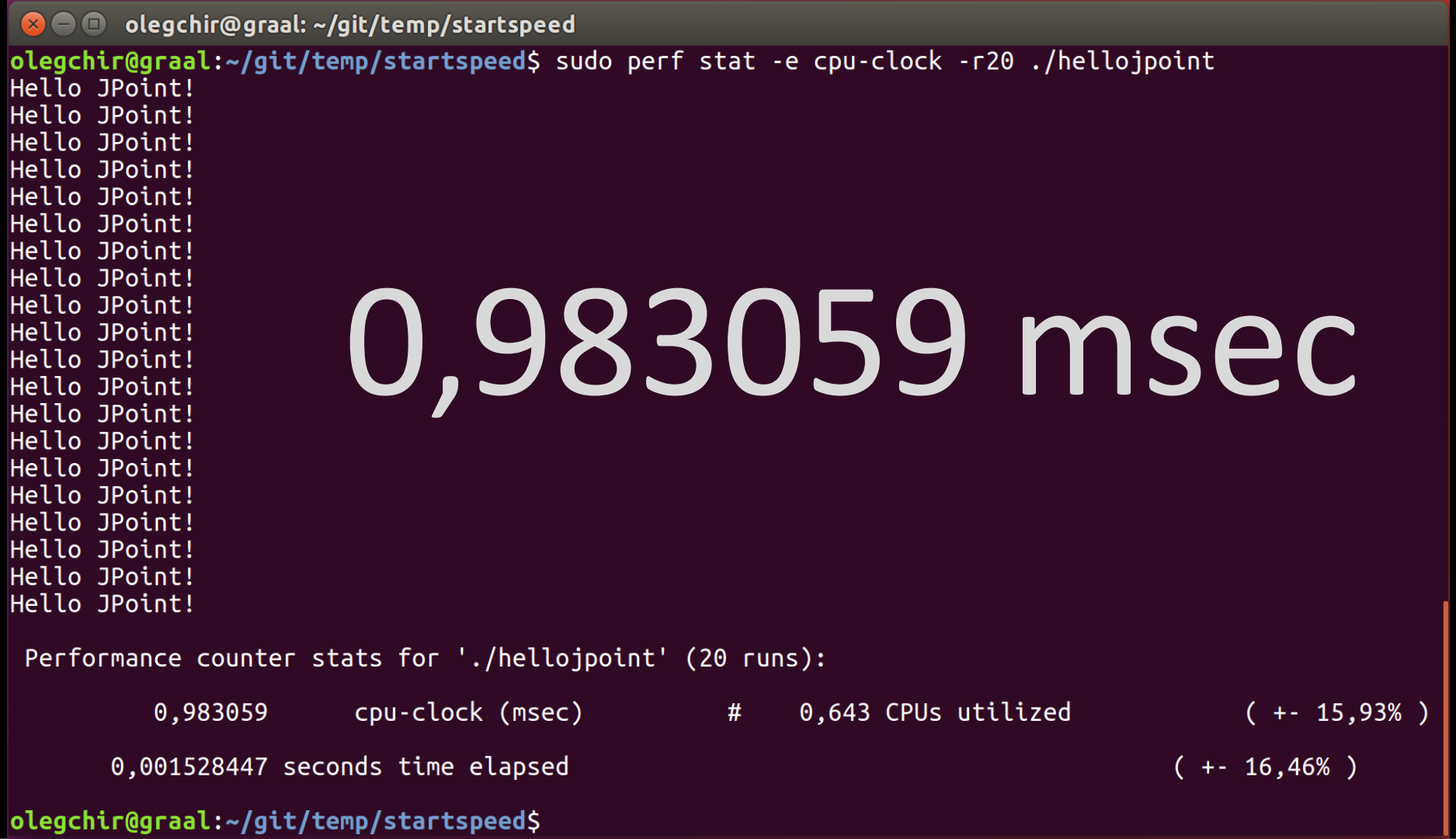
Better Containerized JVMs in JDK10

Java in a World of Containers

Substrate VM


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ native-image HelloJPoint
Build on Server(pid: 1074, port: 26681)
classlist: 321.69 ms
(cap): 669.45 ms
setup: 1,302.03 ms
(typeflow): 6,354.53 ms
(objects): 2,098.74 ms
(features): 43.58 ms
analysis: 8,634.55 ms
universe: 334.01 ms
(parse): 1,947.64 ms
(inline): 1,175.19 ms
(compile): 11,649.96 ms
compile: 15,178.01 ms
image: 1,163.53 ms
write: 430.87 ms
[total]: 27,517.28 ms
olegchir@graal:~/git/temp/startspeed$


olegchir@graal: ~/git/temp/startspeed
-rwxrwxr-x 1 olegchir olegchir 4,8M апр 6 03:13 ./hellojpoint
olegchir@graal:~/git/temp/startspeed$ file ./hellojpoint
./hellojpoint: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynami
cally linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.3
2, BuildID[sha1]=8c815f5a51c6acff0ec6a555cb9725dbdb64c12f, stripped
olegchir@graal:~/git/temp/startspeed$


olegchir@graal: ~/git/temp/startspeed
olegchir@graal:~/git/temp/startspeed$ sudo perf stat -e cpu-clock -r20 ./hellojpoint
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
Hello JPoint!
0,983059 msec
Performance counter stats for './hellojpoint' (20 runs):
0,983059 cpu-clock (msec) # 0,643 CPUs utilized (+- 15,93%)
0,001528447 seconds time elapsed (+- 16,46%)
olegchir@graal:~/git/temp/startspeed$

0,9 (SVM) <<< 61 (CDS) < 71 (AOT) < 100 (база)

Metropolis: city of tomorrow

http://openjdk.java.net/projects/metropolis/

http://cr.openjdk.java.net/~jrose/metropolis/Metropolis-Proposal.html