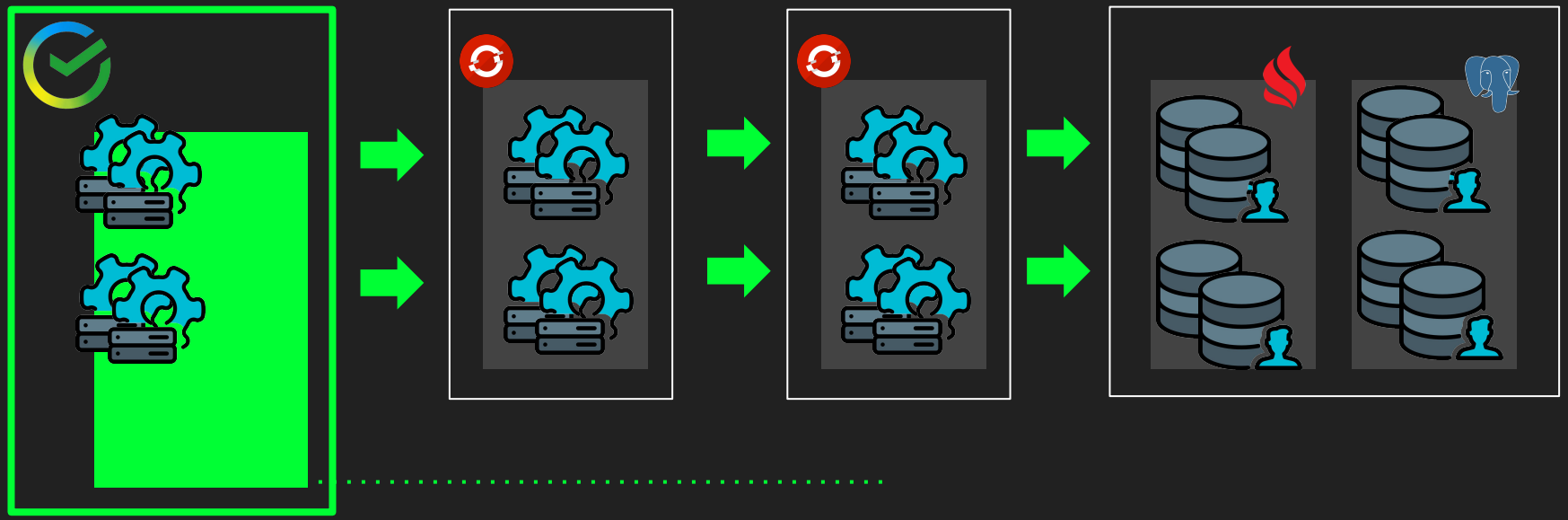
Java-разработчик

Раньше программировал на XML,
теперь больше на YAML.

В Сбере разрабатывает хранилище для клиентских данных.

GraalVM Native Image: польза в реальной жизни

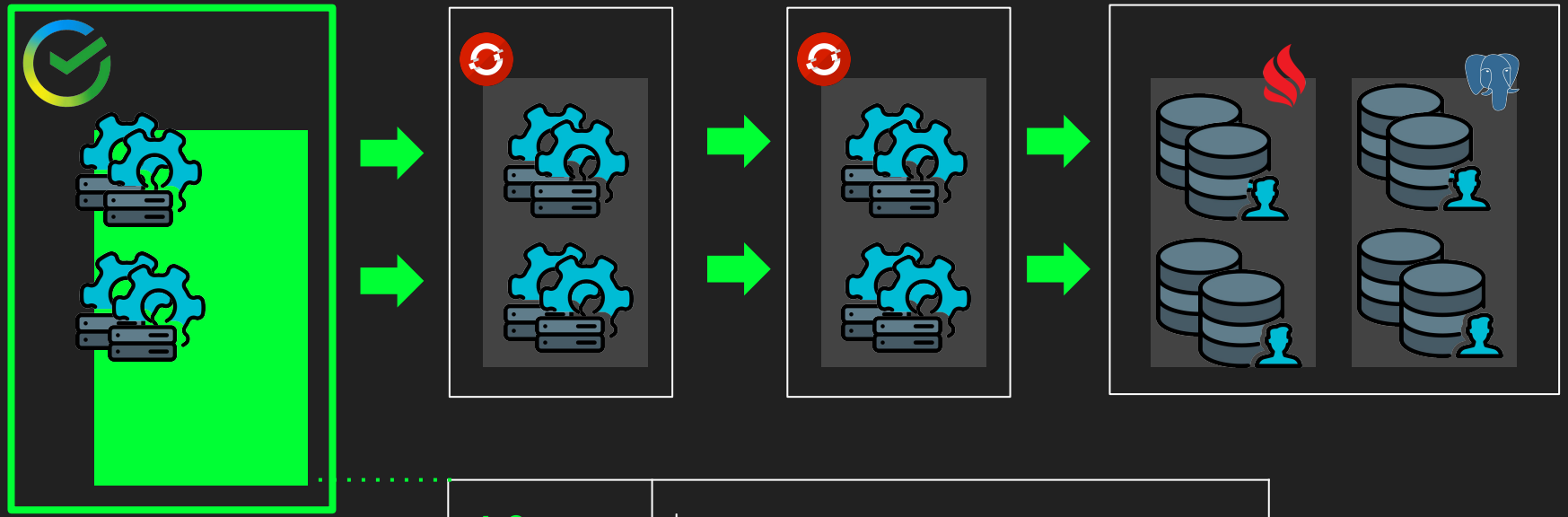
Наши потребители



Сбербанк онлайн

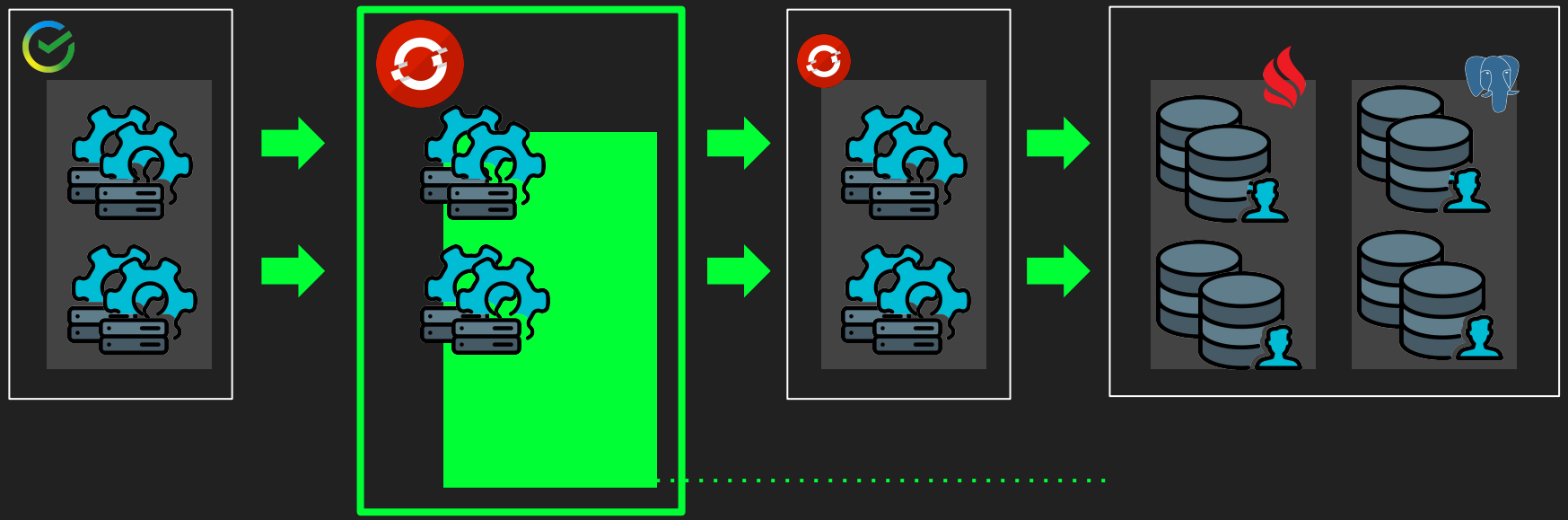
90% сервисов Сбера и компаний-партнёров

Наши потребители



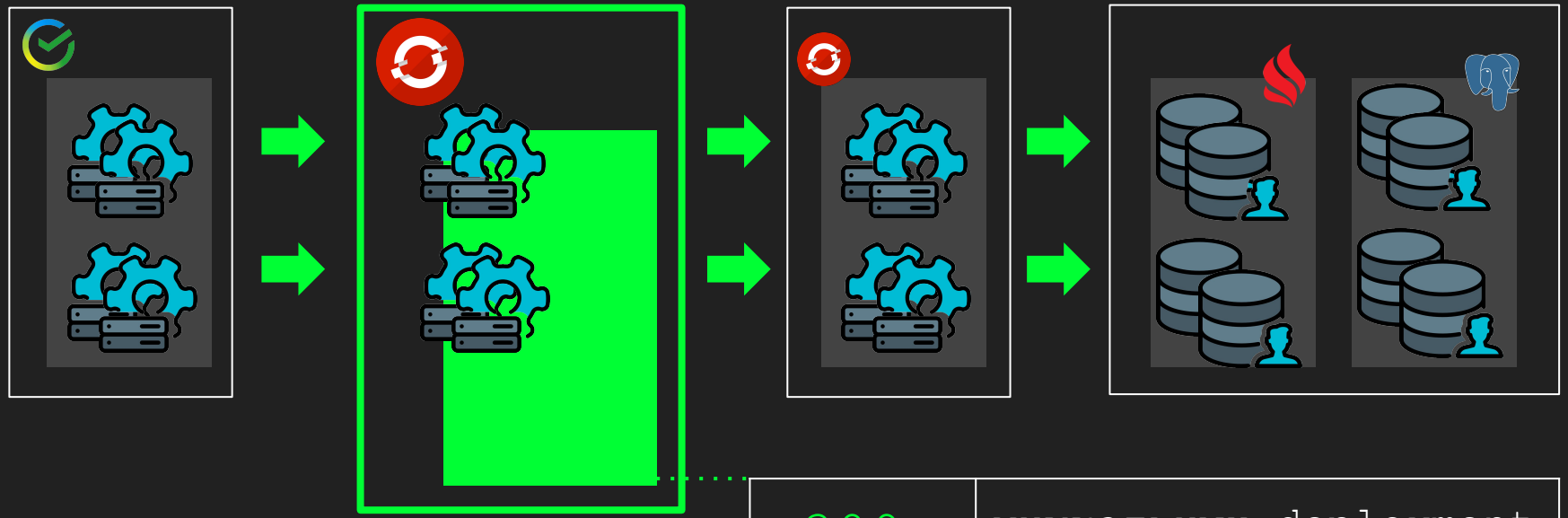
40K	tps нагрузка на чтение
10K	tps нагрузка на запись
>300	потребителей

Бизнес слой



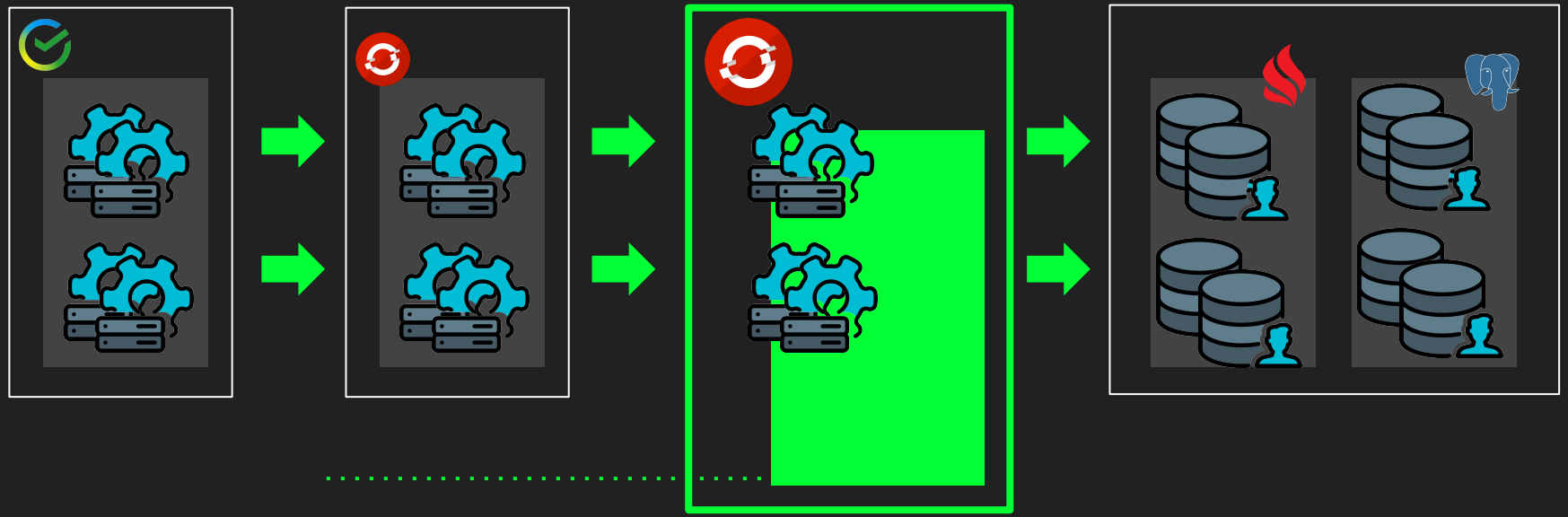
Бизнес слой

Бизнес слой



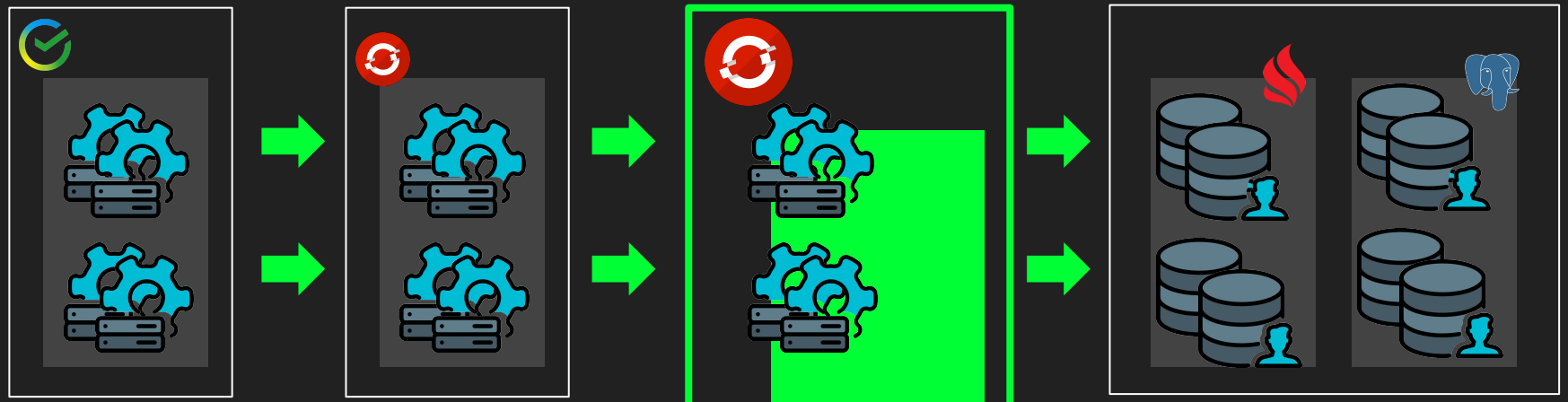
>300	уникальных deployment
>1500	pod-ов
99.99	доступность

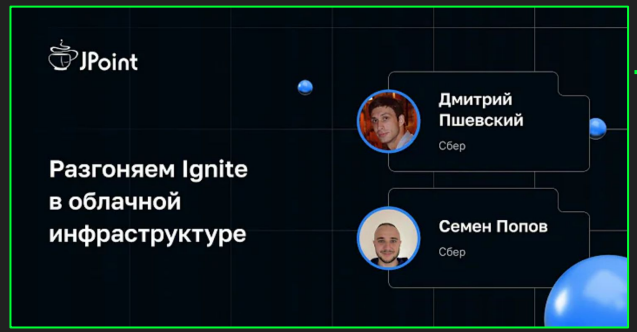
Core слой



Core слой

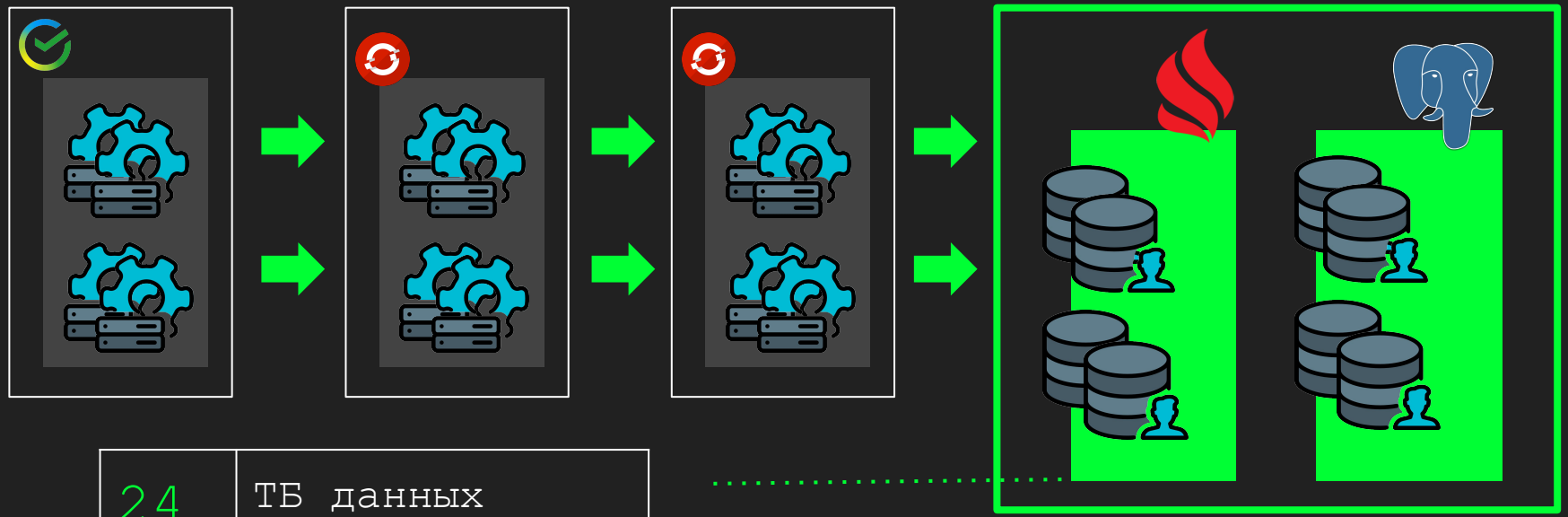
Core слой





https://www.youtube.com/watch?v=Aw177IZplj0&t=56s

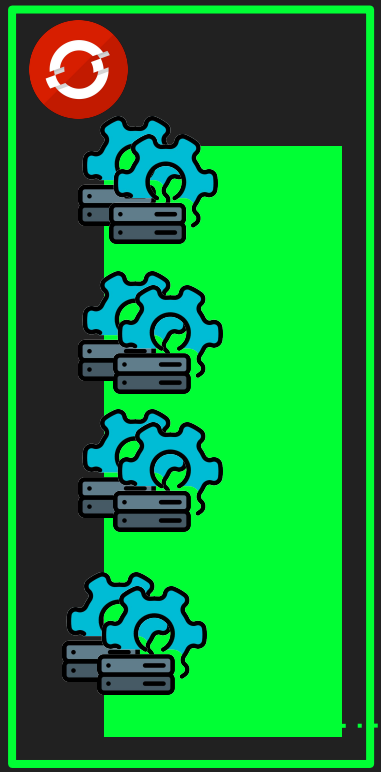
Слой хранения



24	ТБ данных
6	postgre
60	узлов ignite

Слой хранения

Речь пойдет про бизнес слой

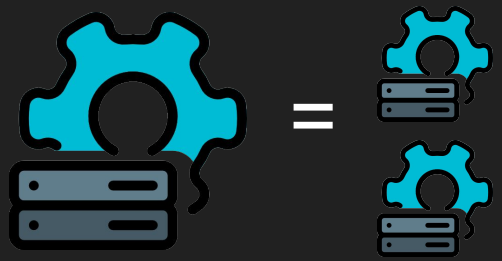


95% deployment на бизнес слое:

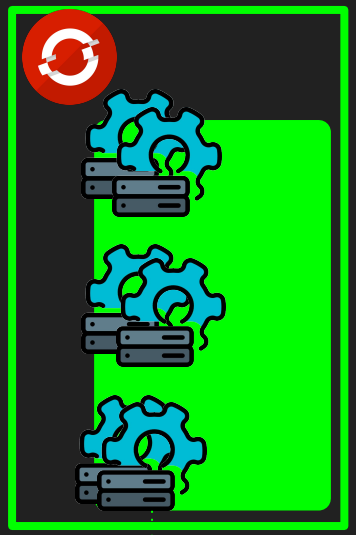
- Частые релизы
- маленький 1-2 endpoint
- Нагрузка 10/40 rps на pod
- 30ms/50ms latency в 99%
- 0.8 m 700 Mi на pod

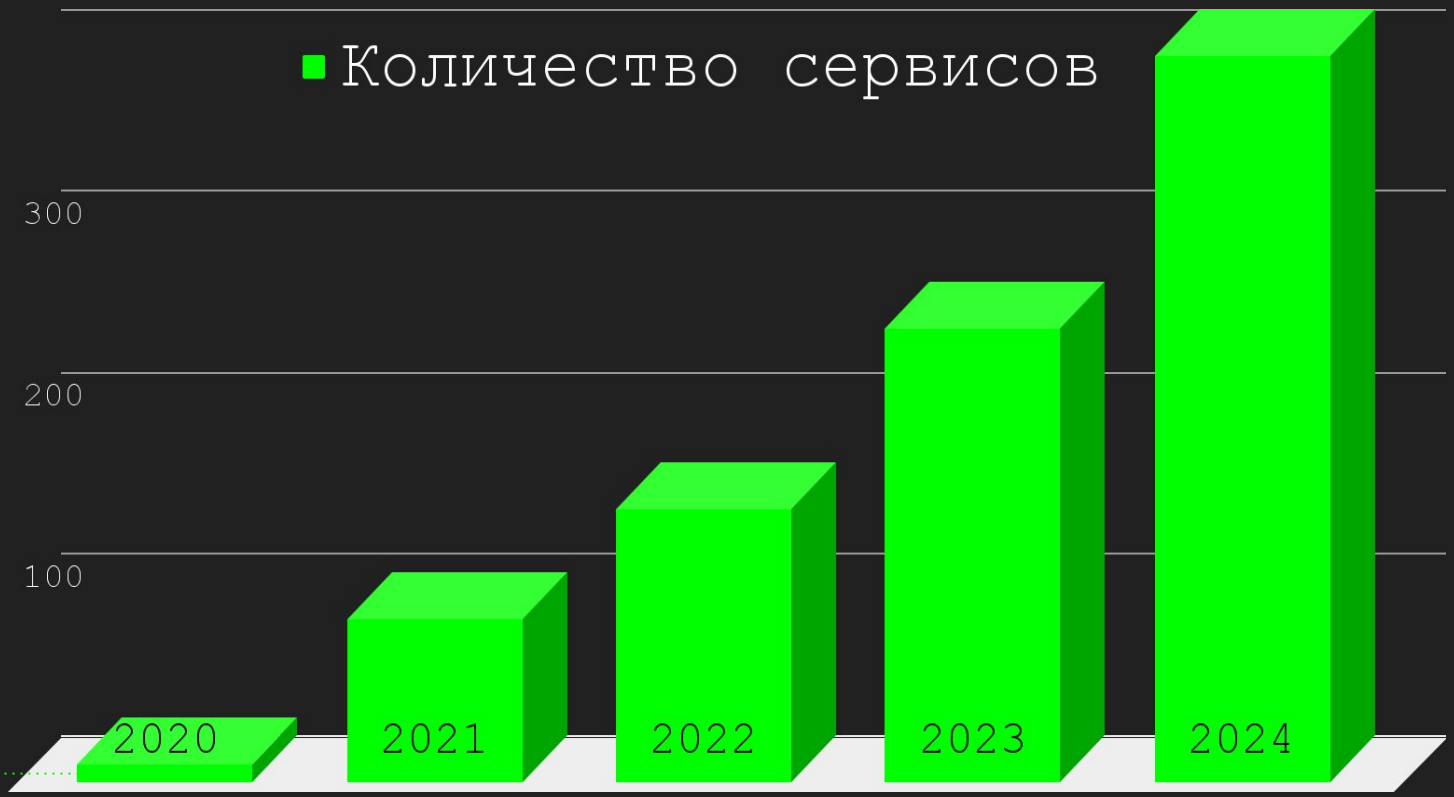
под каждого потребителя

deployment

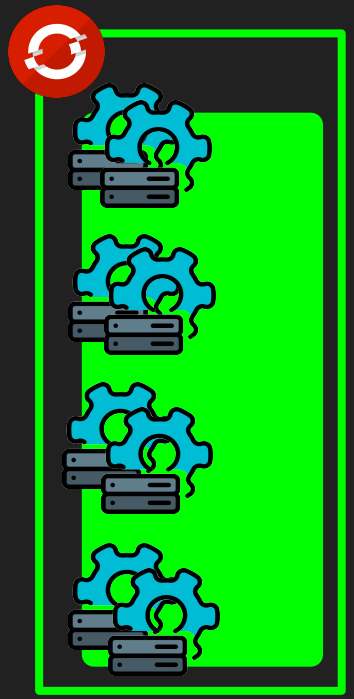


Количество потребителей росло…

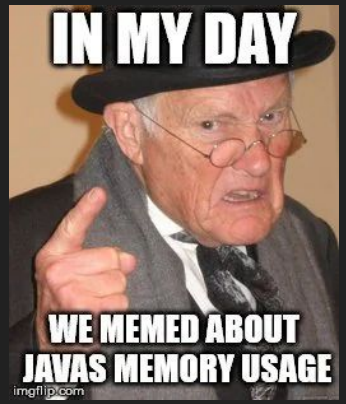




Остро вставал вопрос: откуда брать ресурсы



>1500*0.8 m	cpu
>1500*700 Mi	memory



Зачем вообще столько pod-у с 20 rps?

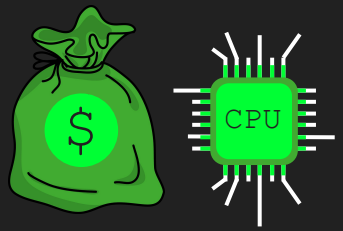
0.8 m	cpu
700 Mi	memory

?

Попробуем поэкономить?

0.8 m	cpu
700 Mi	memory

0.1 m	cpu
400 Mi	memory



Экономим?

Нюанс


~~Петька~~
Экономия
Нюанс
~~Василий Иванович~~
Скорость старта
и Прогрев

cpu/Скорость старта

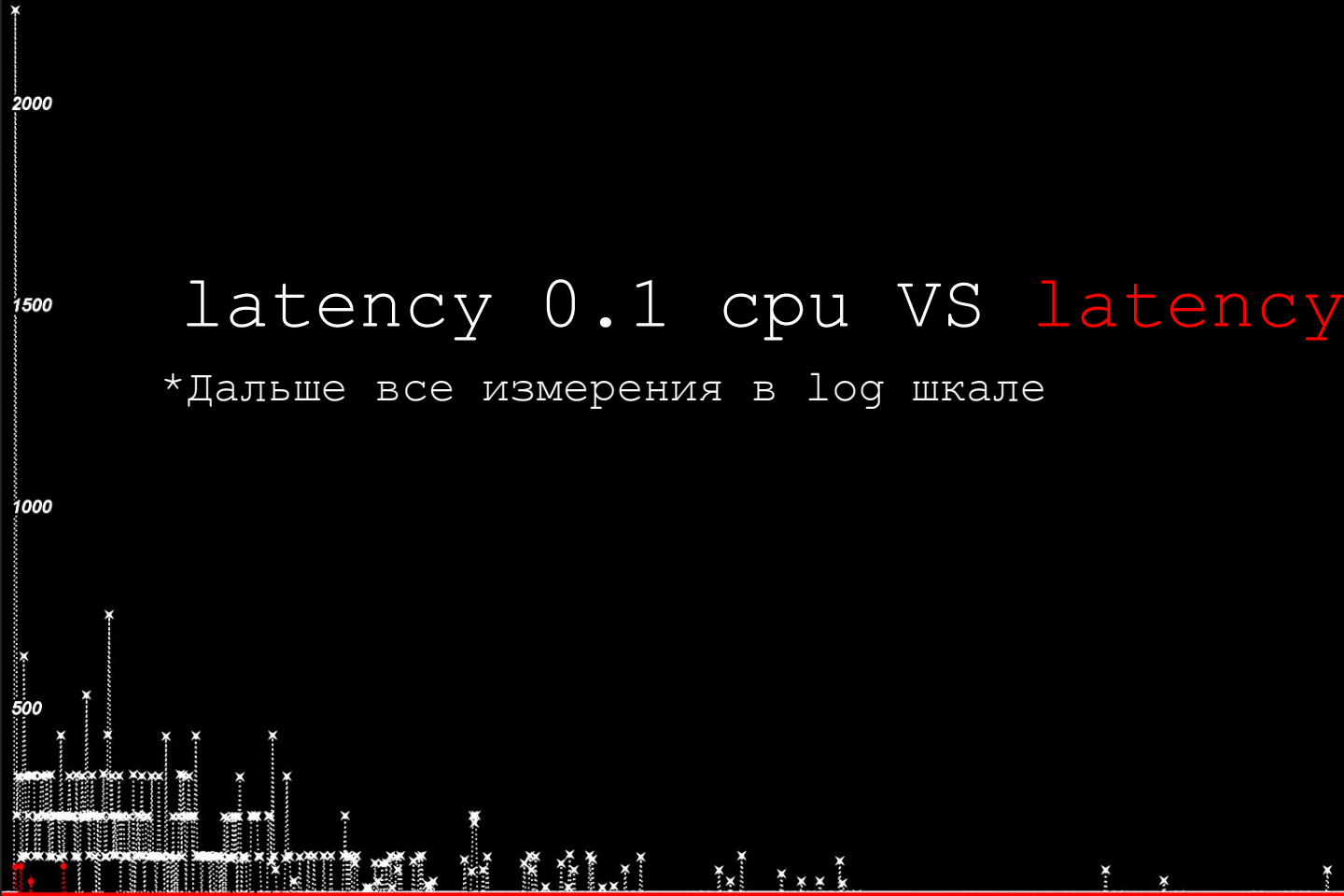
cpu	скорость старта
0.1	started in 41.506s
0.2	started in 18.401s
0.3	started in 9.776s
0.5	started in 4.524s
0.8	started in 3.543s

cpu/Время прогрева 20 rps

cpu	Время прогрева
0.1	~5 min
0.2	~3 min
0.3	~3 min
0.5	~1 min
0.8	~<1 min

latency ms 10 min 20rps 99%%

×0.1cpu ●0.8cpu



latency 0.1 cpu VS latency 0.8 cpu

*Дальше все измерения в log шкале

latency ms 10 min 20rps 99%%

× 0.1cpu ● 0.8cpu

500

100

50

10

5

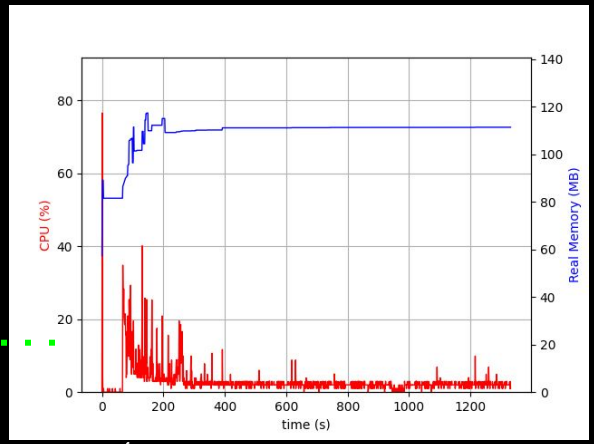
1

0,5

0,1

~5 min JVM 0.1 cpu

30 ms "норма"

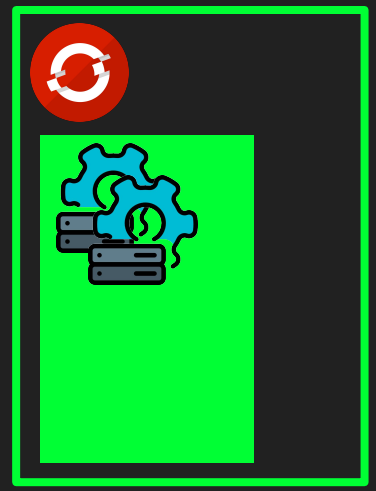


Пример с автомасштабированием

Дано:

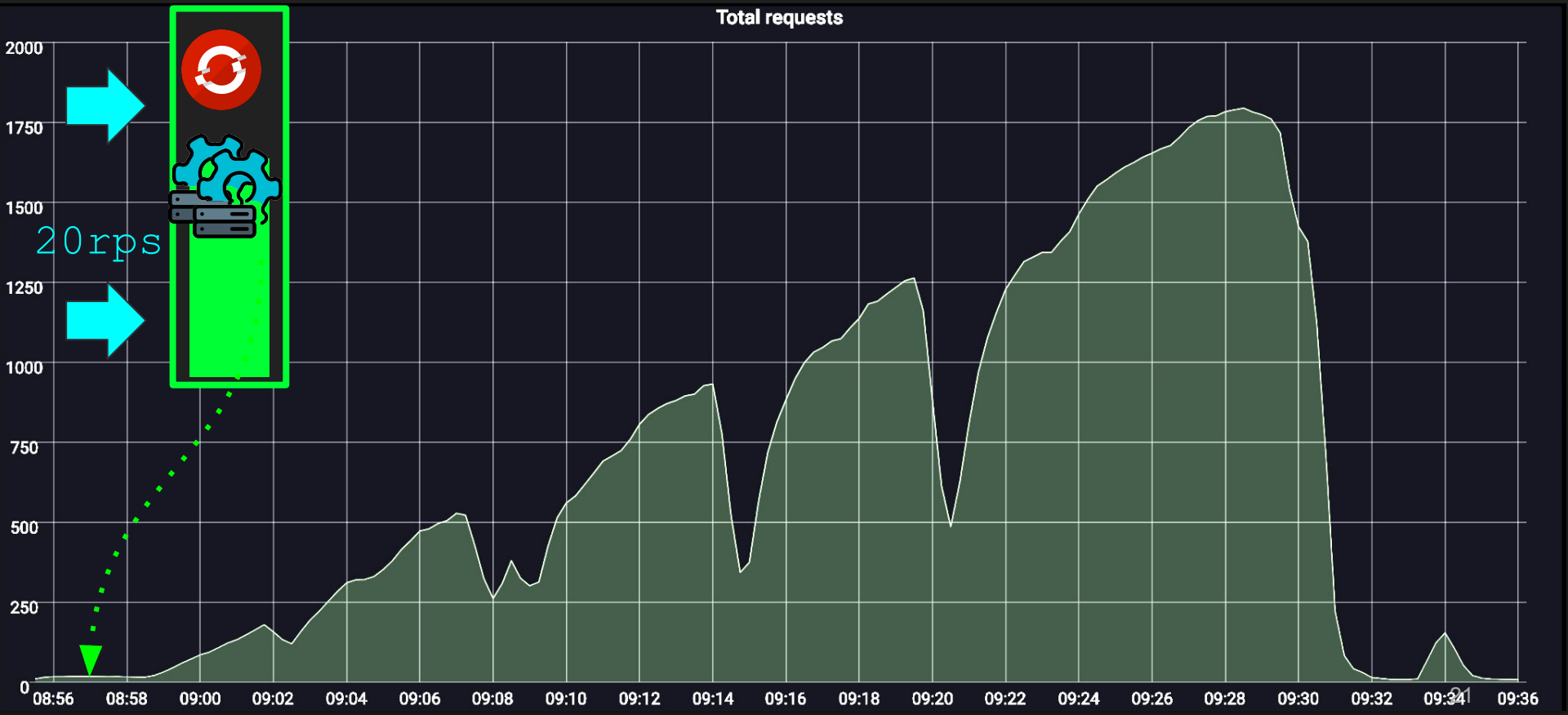
- deployment с Grpc
- 0.1 m
- 300 Mi

20/2000rps

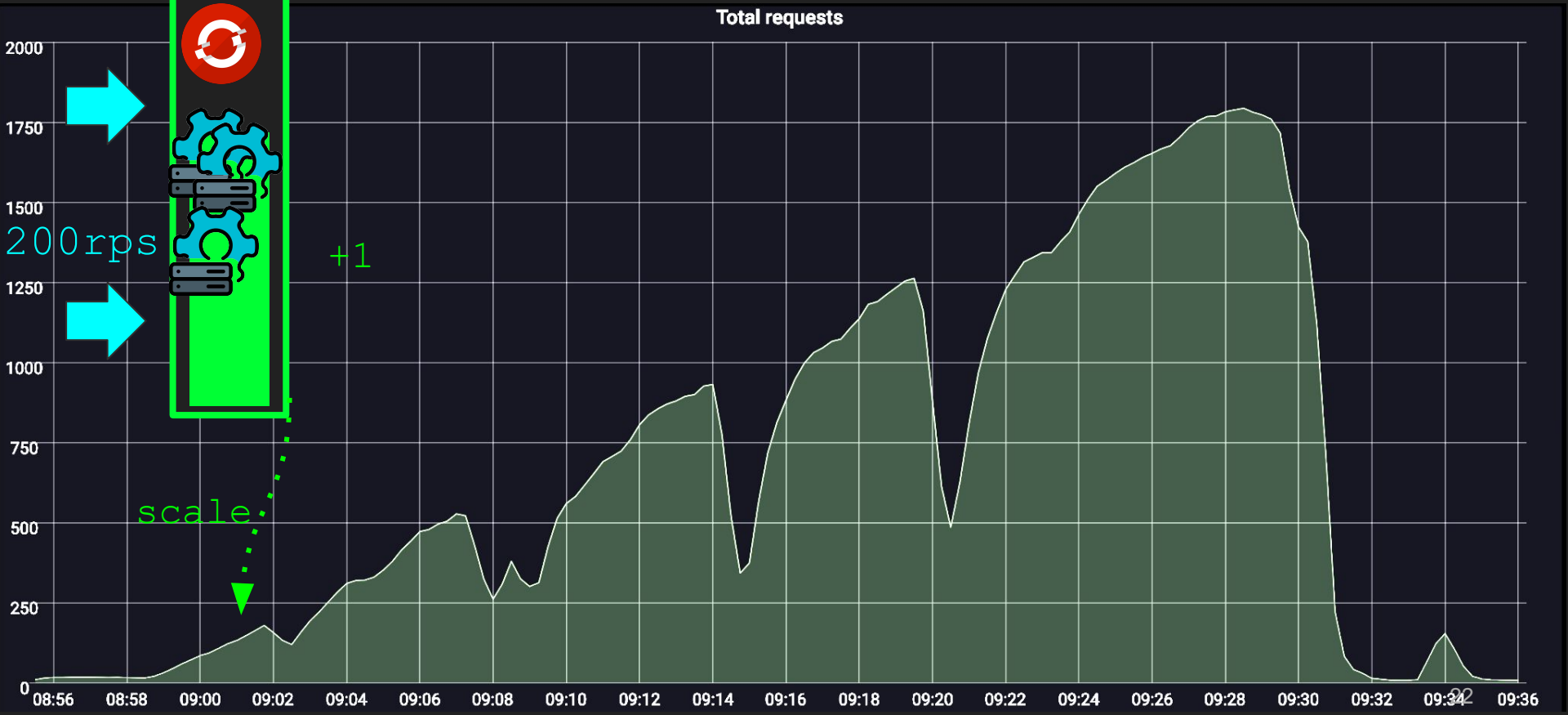


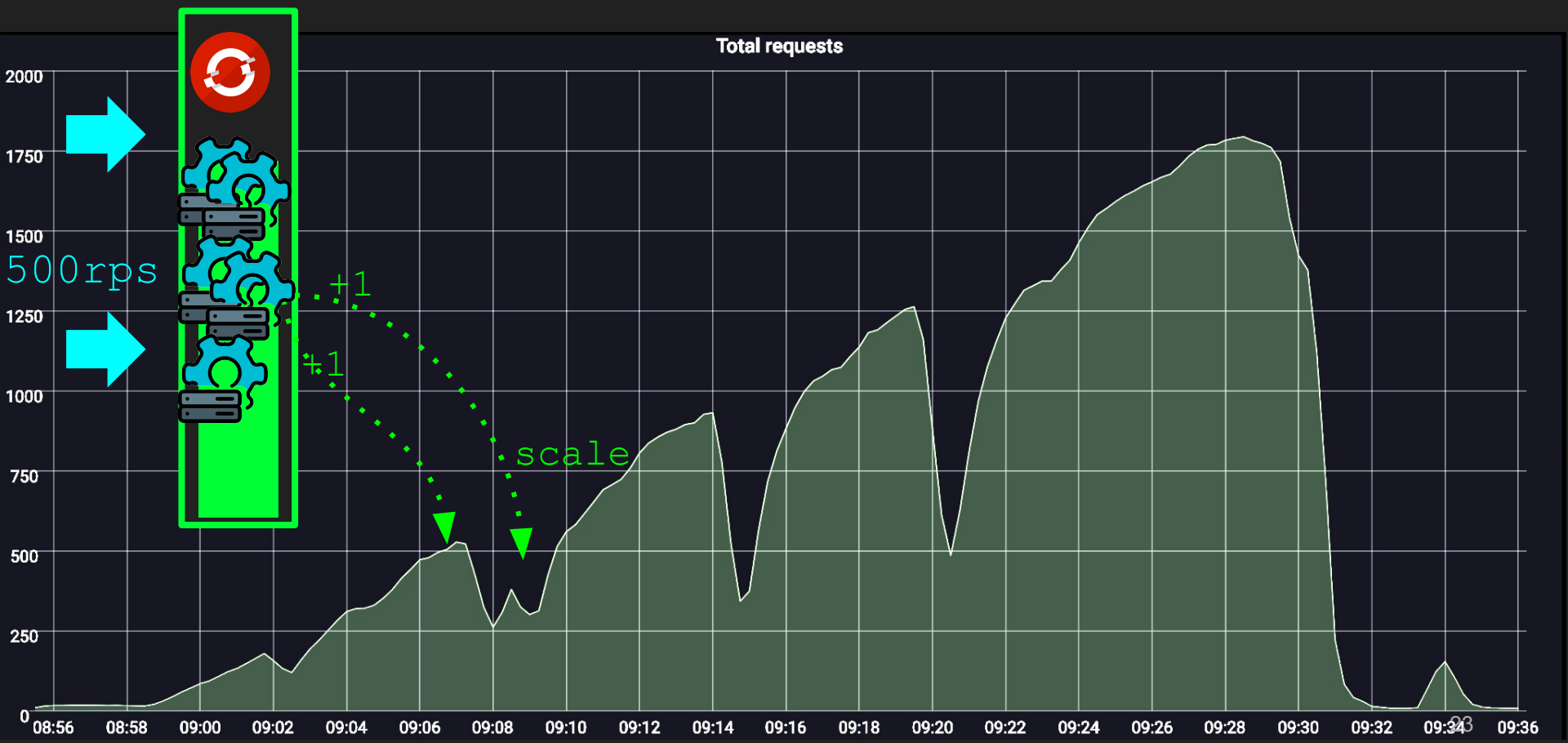
- Монотонно возрастающий профиль нагрузки
- 30 мин
- от 0 до 2000 rps
- не попадаем в 30 ms - retry
- rate-limiter на 200 rps

Старт, на графике успешные запросы

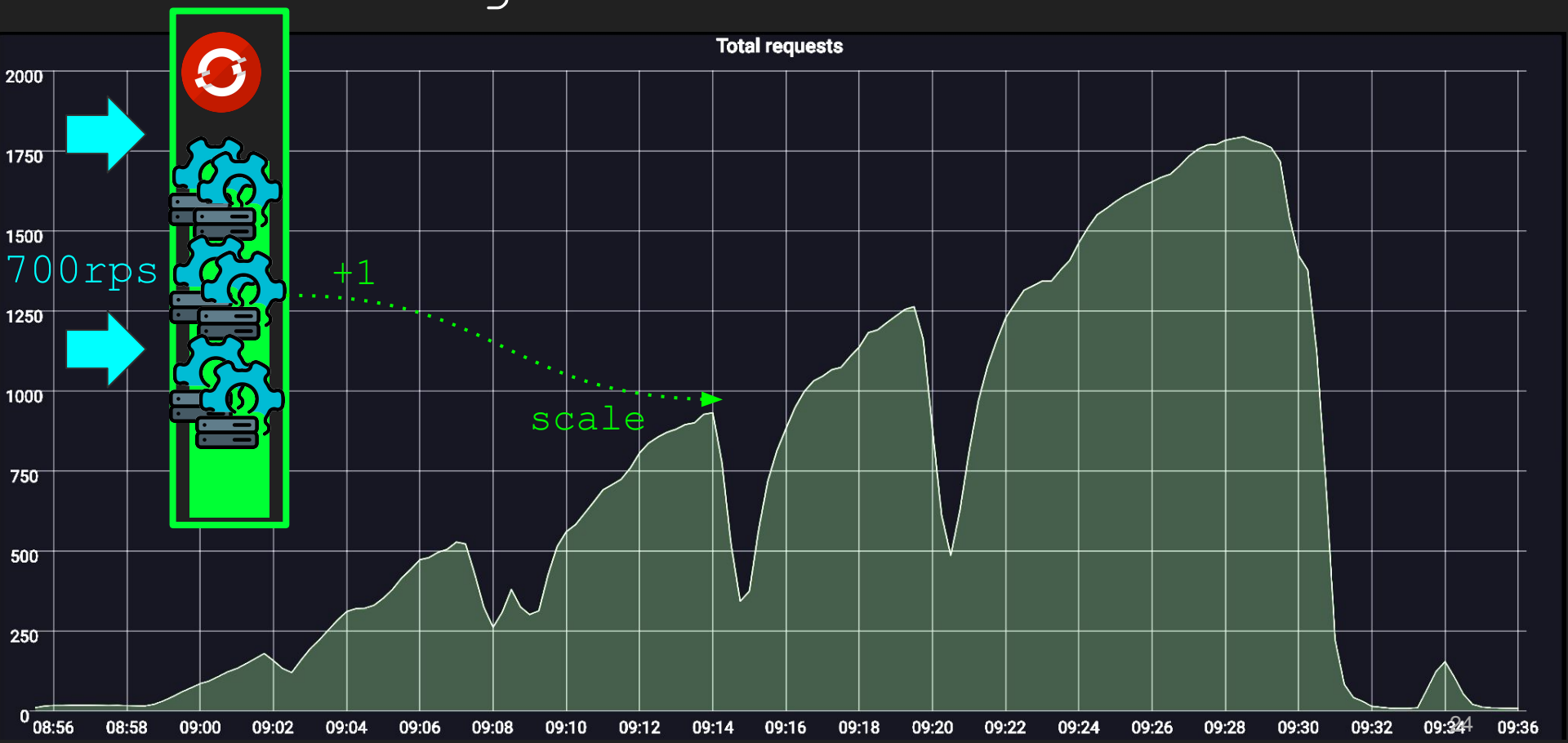


+1

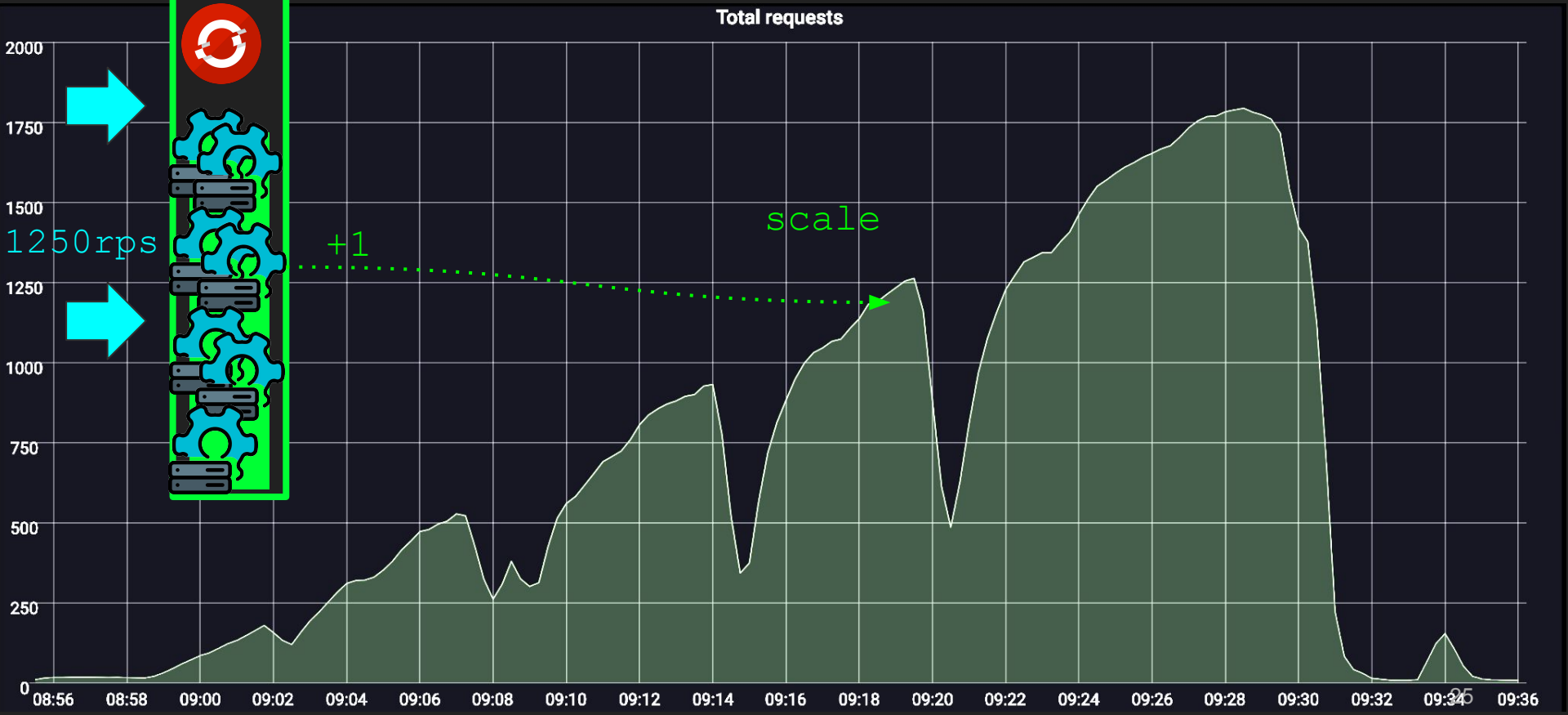

Total requests
2000
1750
1500
1250
1000
750
500
250
0
200rps
+1
scale
08:56
08:58
09:00
09:02
09:04
09:06
09:08
09:10
09:12
09:14
09:16
09:18
09:20
09:22
09:24
09:26
09:28
09:30
09:32
09:34
09:36


Total requests
2000
1750
1500
1250
1000
750
500
250
0
500rps
+1
+1
scale
08:56
08:58
09:00
09:02
09:04
09:06
09:08
09:10
09:12
09:14
09:16
09:18
09:20
09:22
09:24
09:26
09:28
09:30
09:32
09:34
09:36

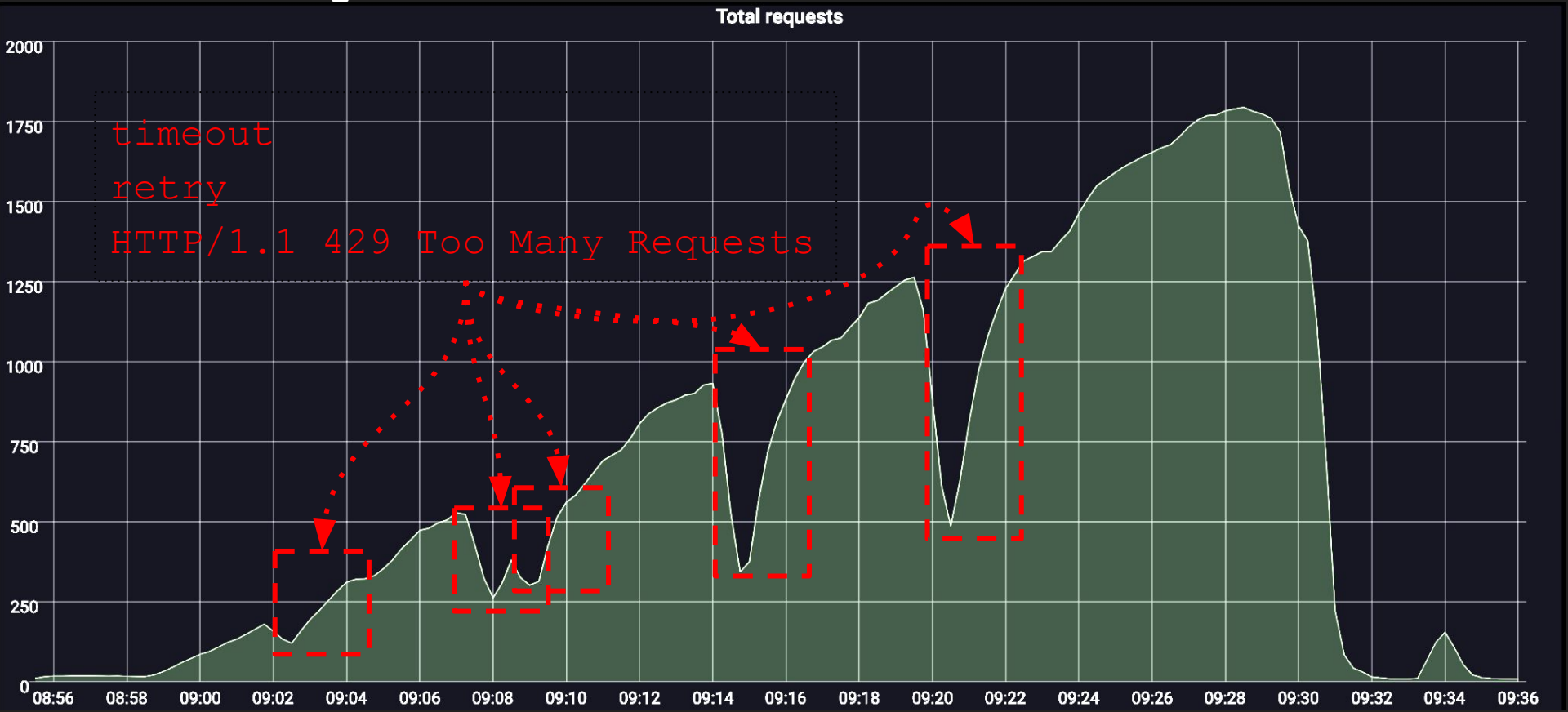
autoscaling



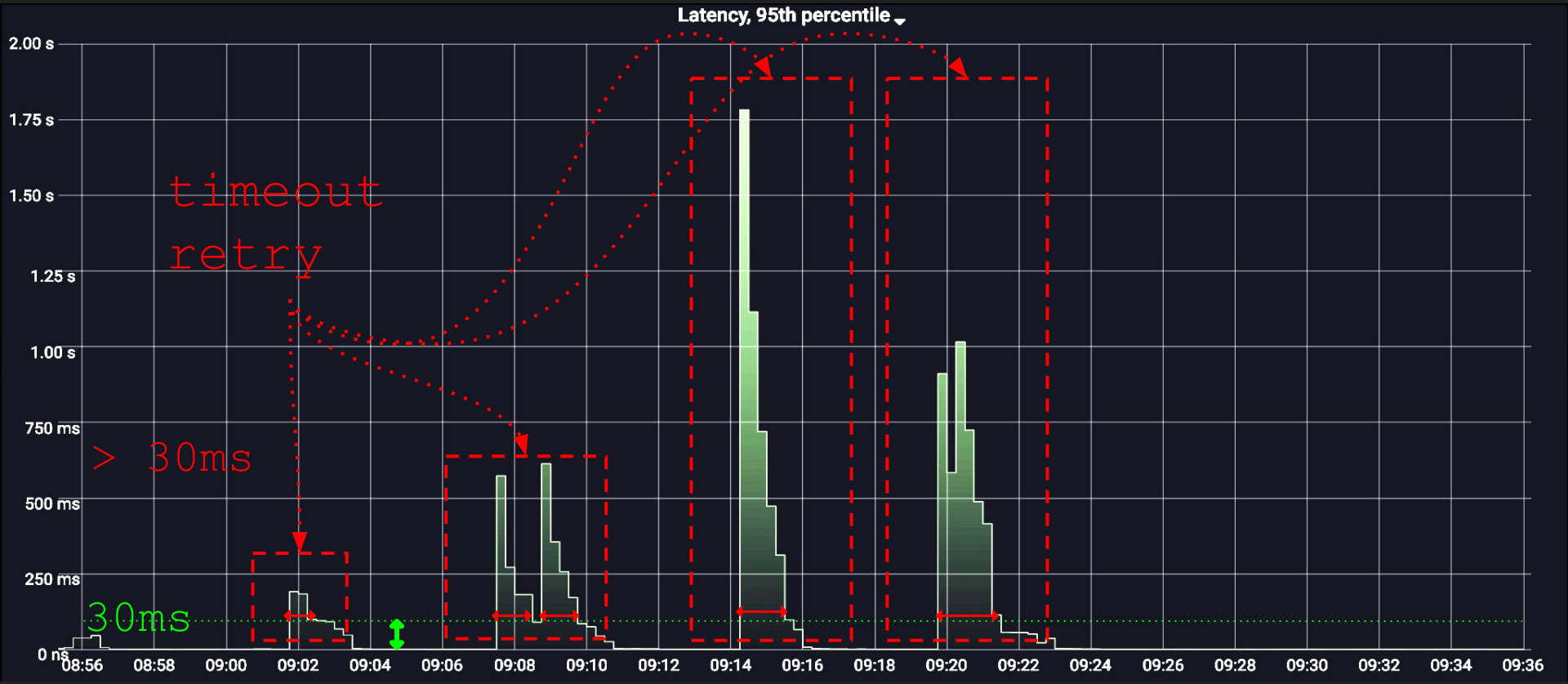
autoscaling



Общая пропускная способность падает

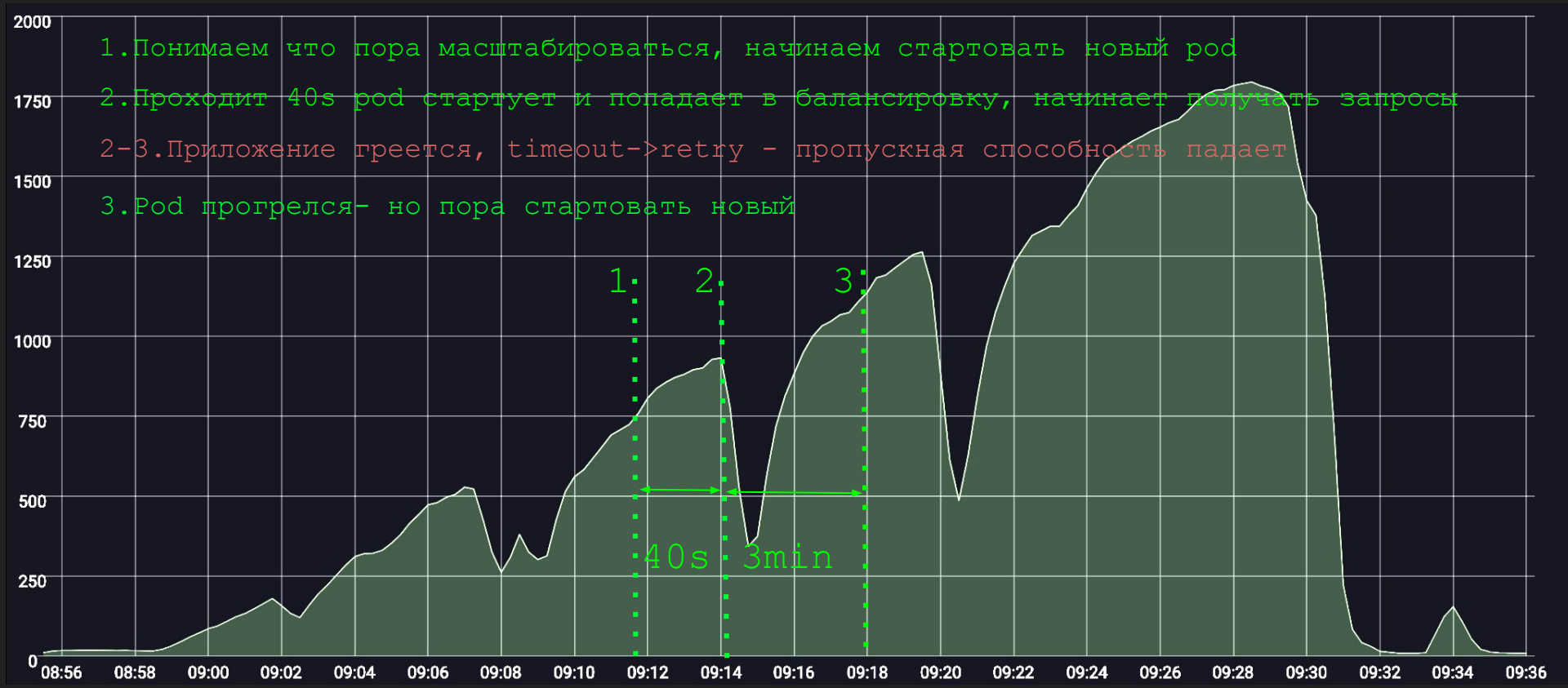


Latency



HTTP 200

Почему так происходит?



Как можно попробовать решить проблему?

Сэкономили

Уменьшили cpu

- Скорость старта упала
- Latency в первые минуты работы ухудшился

Безальтернативность выбора?

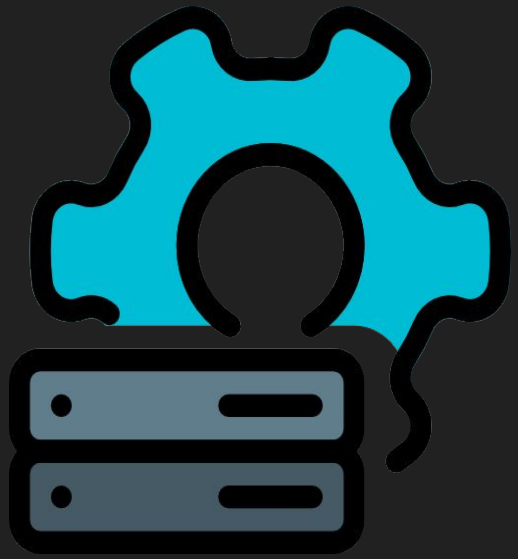
Тратим лишние ресурсы – но приложения быстро стартуют и греются

VS

Экономим – стартуем и “греемся” долго

Выделим важные характеристики

- ? cpu,memory
- ? start speed
- ? Warm-up period



- ? devops
- ? development
- ? throughput

К чему хотим прийти?

- [ ] cpu,memory

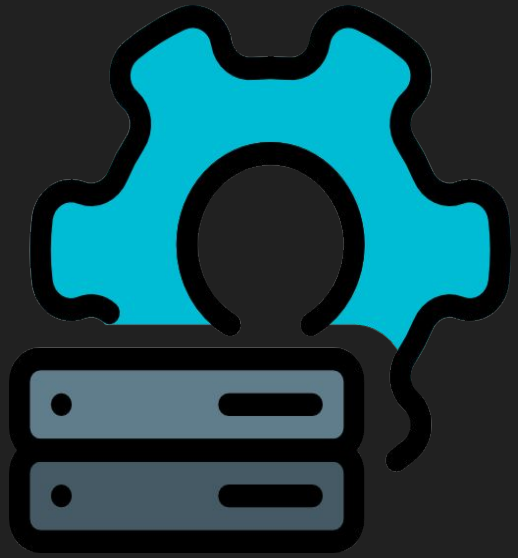
  0.1 cpu 300Mi

- [ ] start speed

  < 1 сек

- [ ] Warm-up period

  < 1 сек



- [ ] devops
- [ ] development
- [ ] throughput

оставить как есть

Нет ничего лучше и точнее рандомных исследований из интернета:

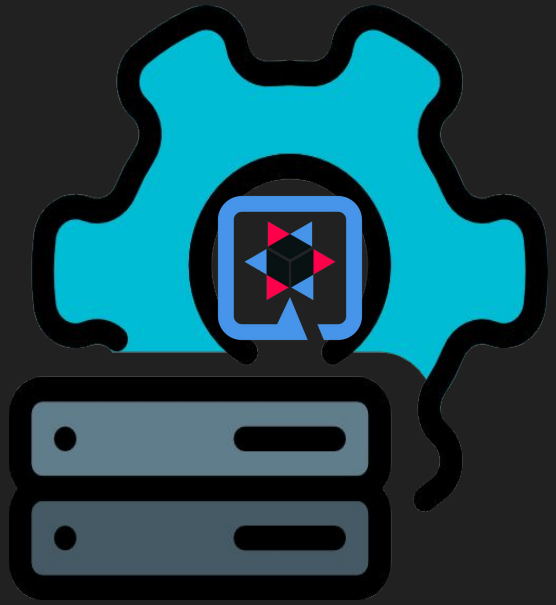
Static proxies in Quarkus **VS** Dynamic proxy is a proxy created at runtime.

https://dev.to/nutrymaco/static-proxies-in-quarkus-4ebf

https://www.baeldung.com/spring-boot-vs-quarkus

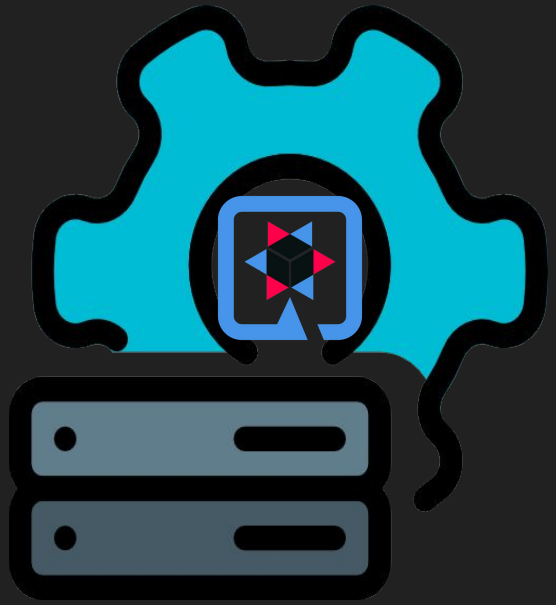
to Успехи!

- cpu,memory
- start speed
- Warm-up period



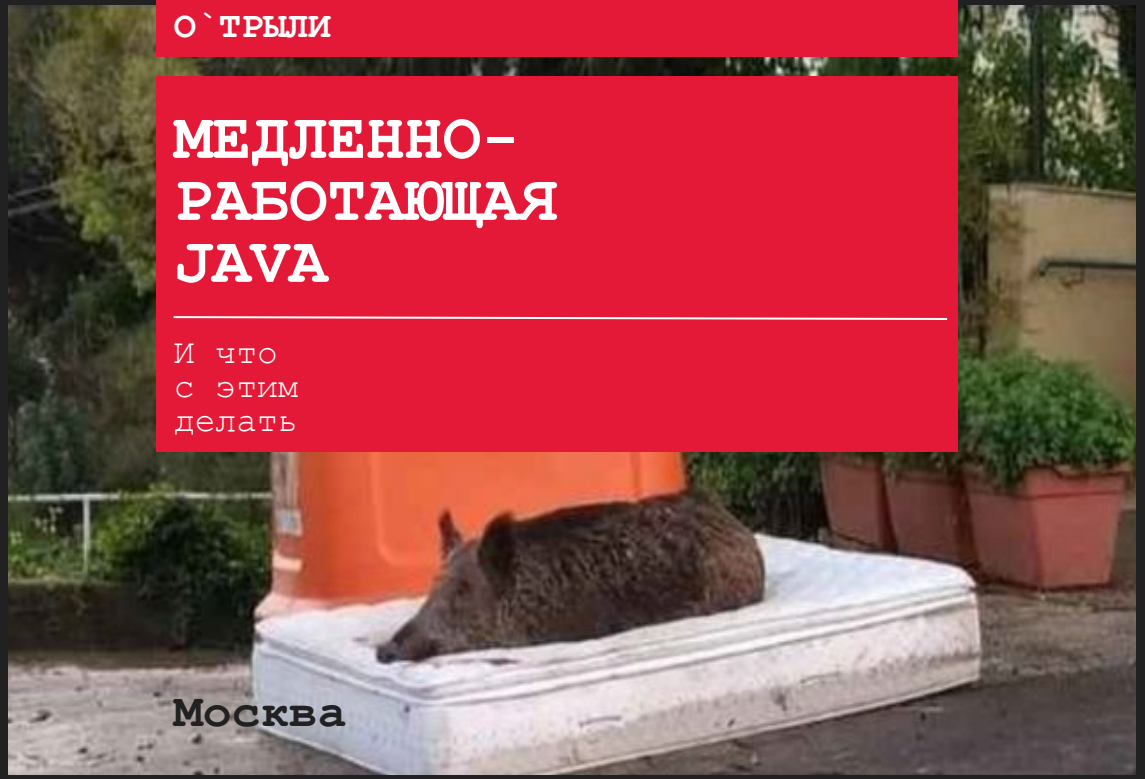
- devops
- development
- throughput

Quarkus помог, но чуть-чуть)

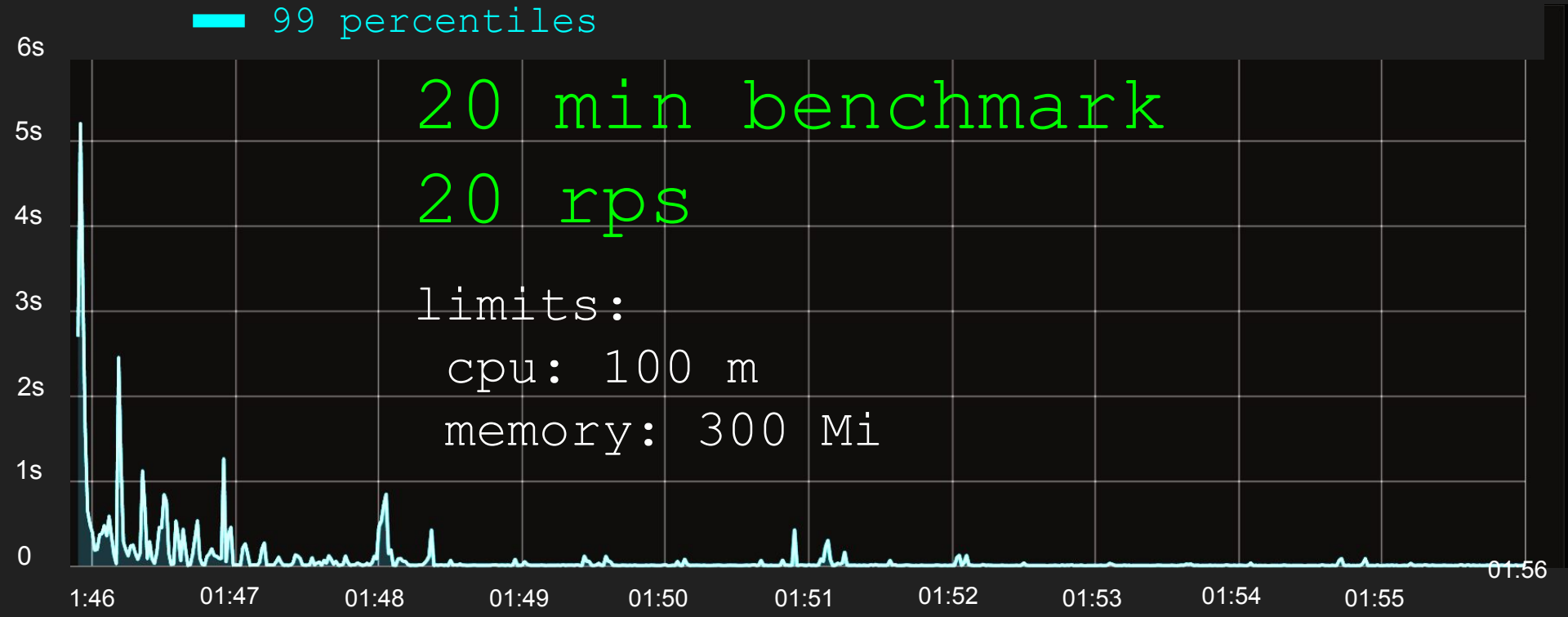


- cpu,memory (чуть-чуть)
- start speed (чуть-чуть)
- Warm-up period (чуть-чуть)
- devops
- development
- throughput (чуть-чуть)

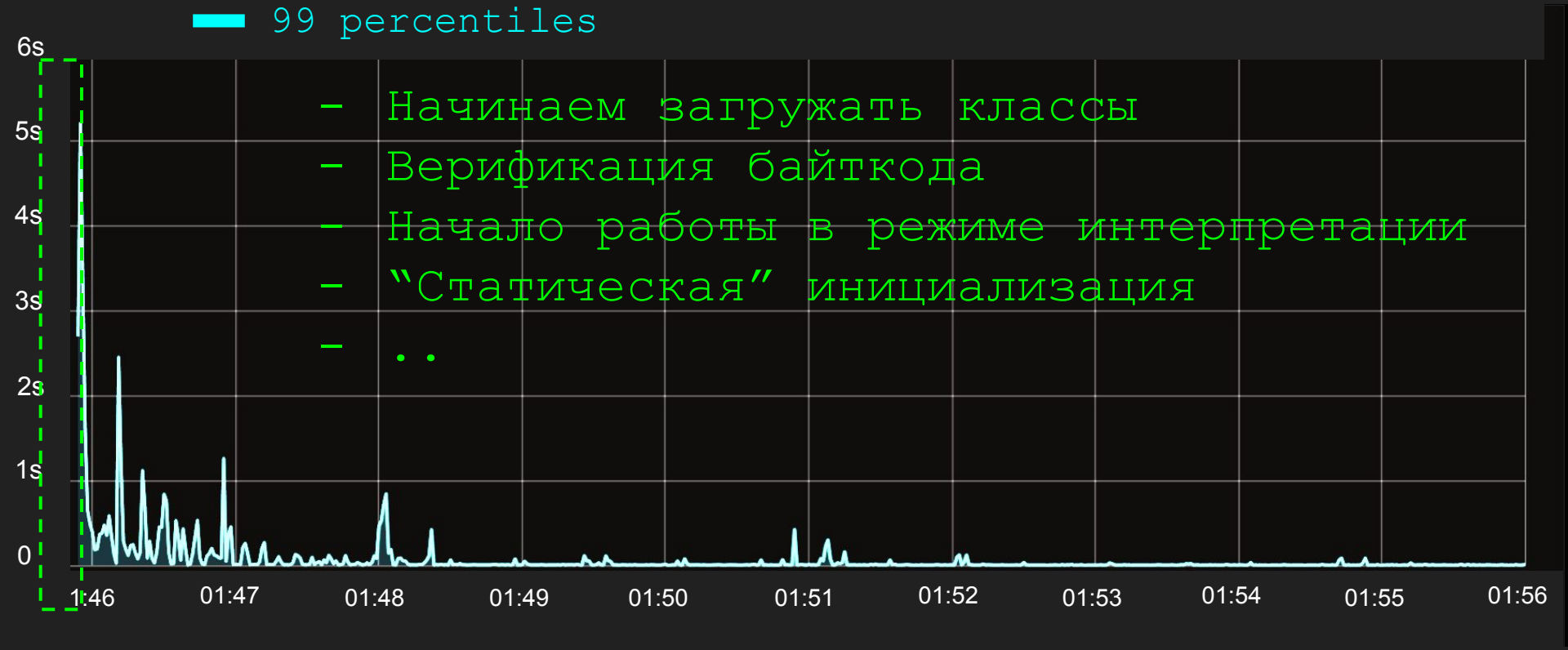
Во всем виновата - JAVA



latency/time java app

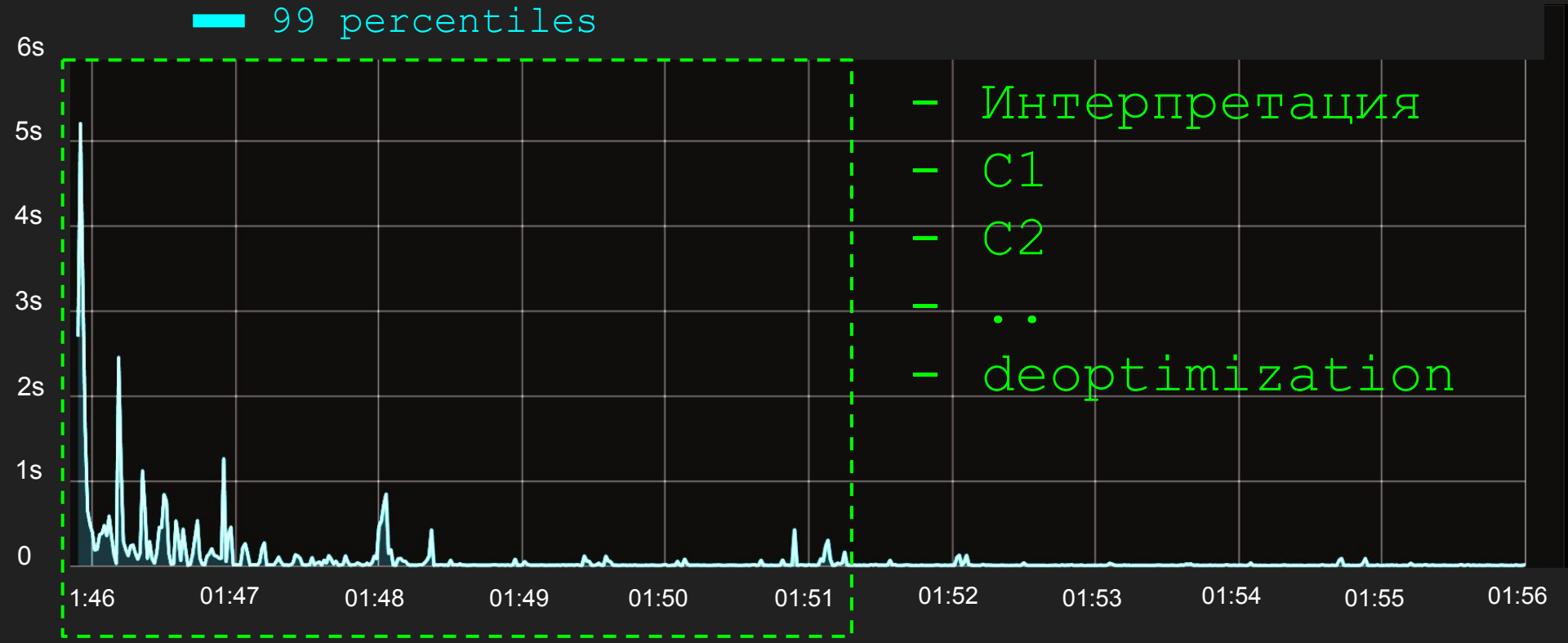


Инициализация и старт

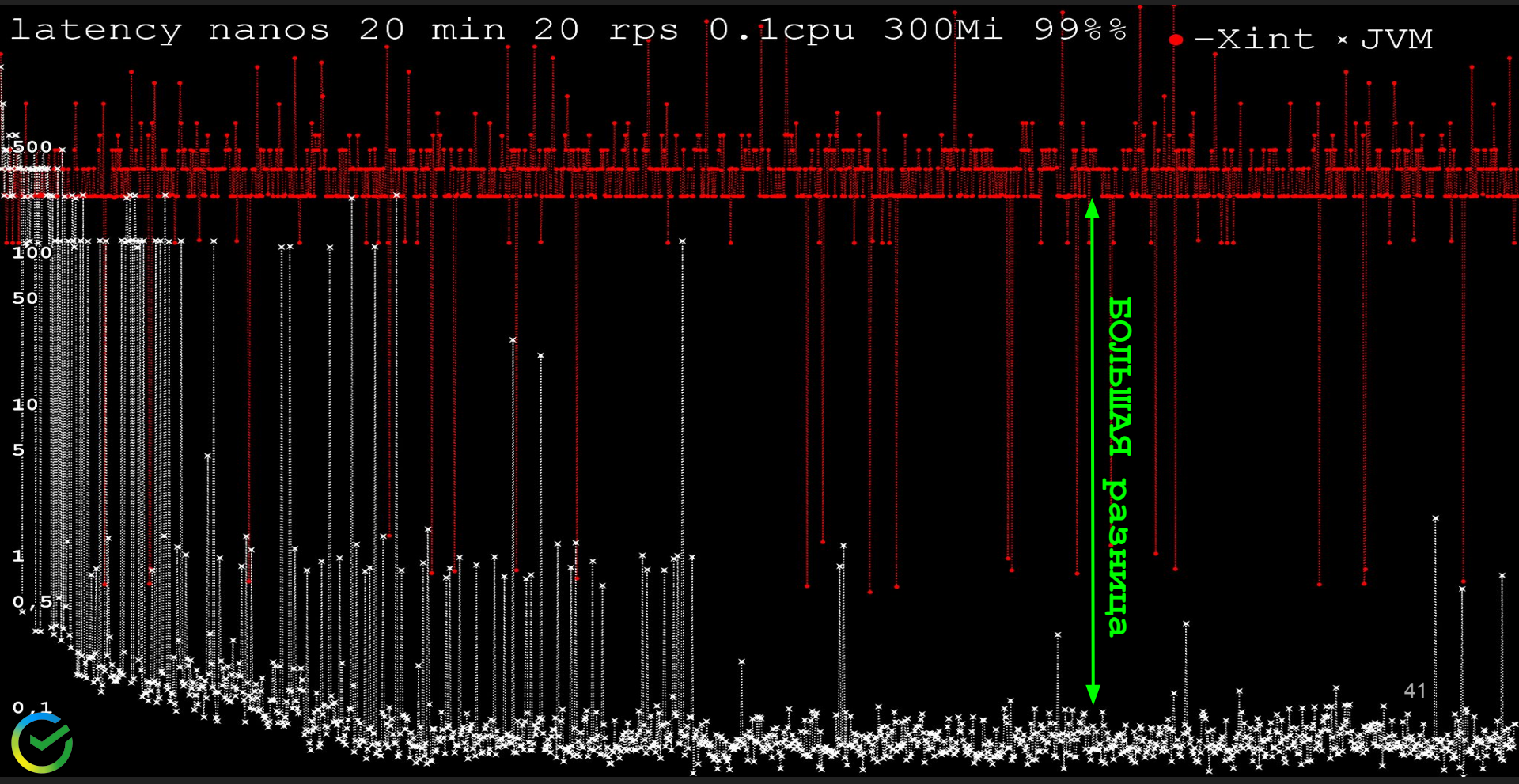


- Начинаем загружать классы
- Верификация байткода
- Начало работы в режиме интерпретации
- “Статическая” инициализация
- ..

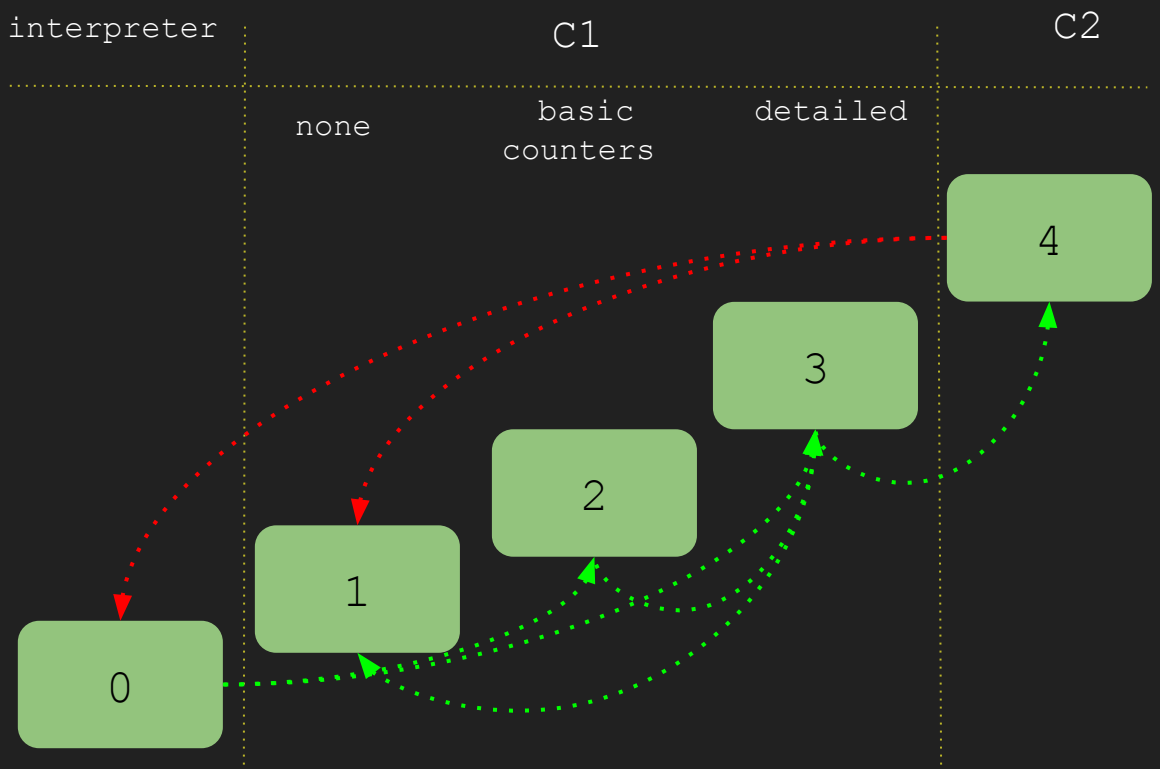
Прогрев - разберемся почему?

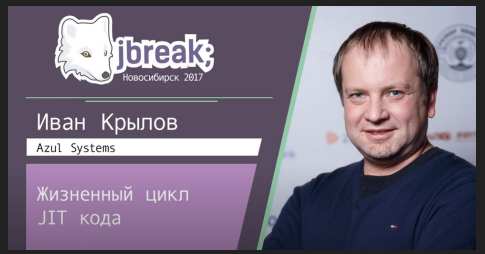


compilation vs interpretation -Xint



compiler diagram lvl



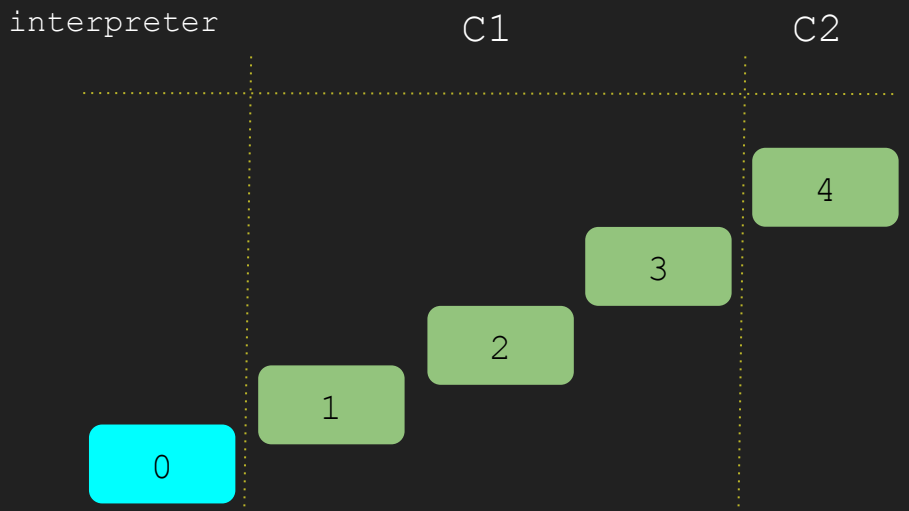


https://youtu.be/9valLOxgDbI

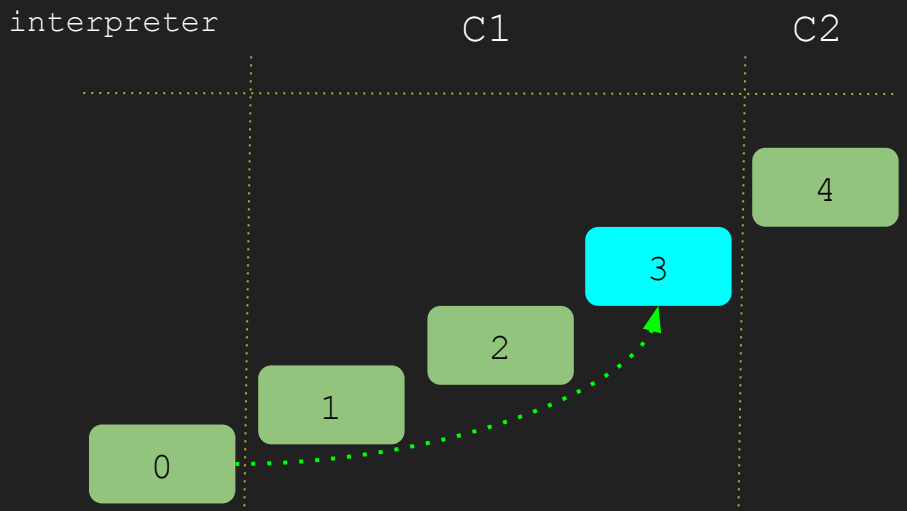
Адский код: сколько раз это скомпилируется?

```
public final class CheckNullUtils {
    private CheckNullUtils() {}

    public static String checkNull(
            String s
    ) {
        if (s == null) {
            throw new RuntimeException("эксепшн");
        } else {
            return s;
        }
    }
}
```


```
for (int i = 0; i < 10; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```

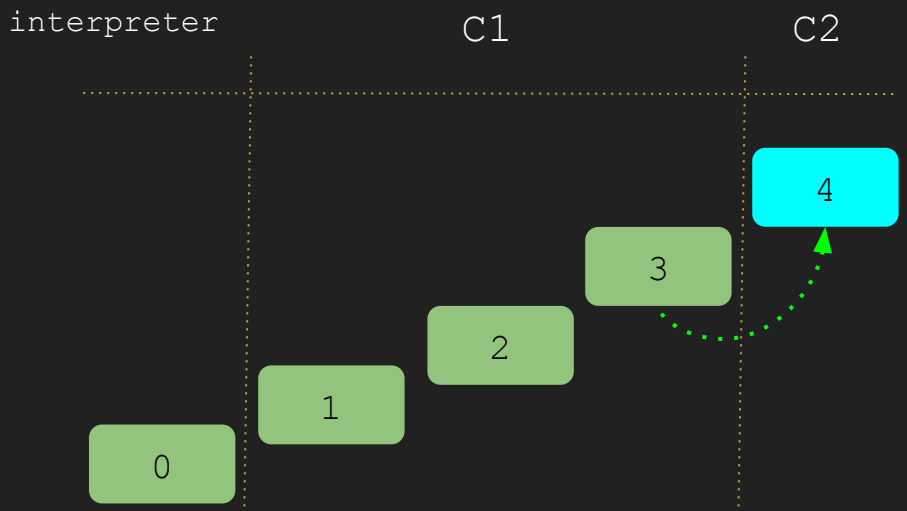


```
Thread.sleep(3000);//не просто так
for (int i = 0; i < 200; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```



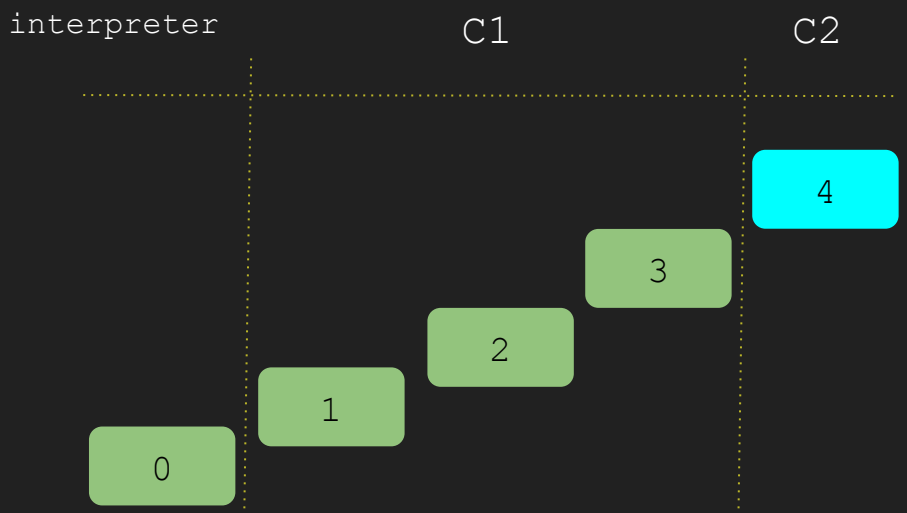
-XX:Tier3InvocationThreshold
default(200)

```
Thread.sleep(3000);//не просто так
for (int i = 0; i < 5000; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```



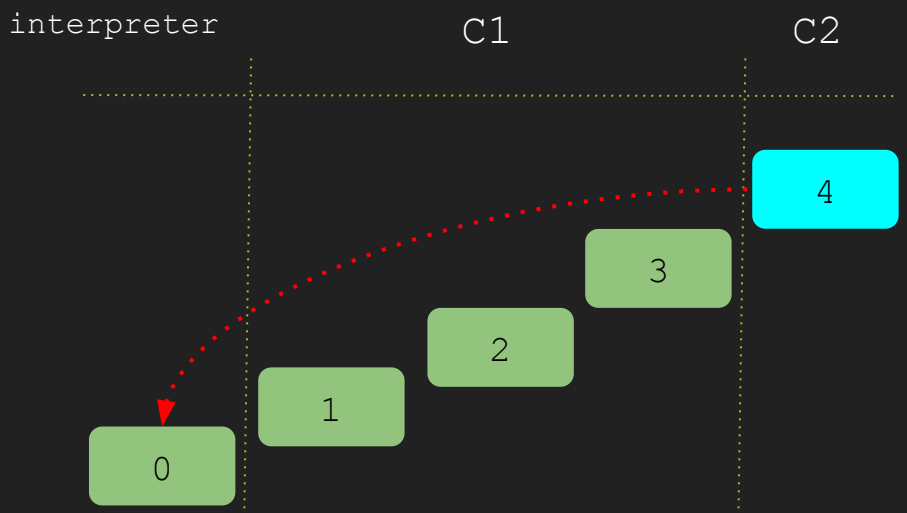
-XX:Tier4InvocationThreshold
default(5000)

```
Thread.sleep(3000);//не просто так
for (int i = 0; i < 10; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```



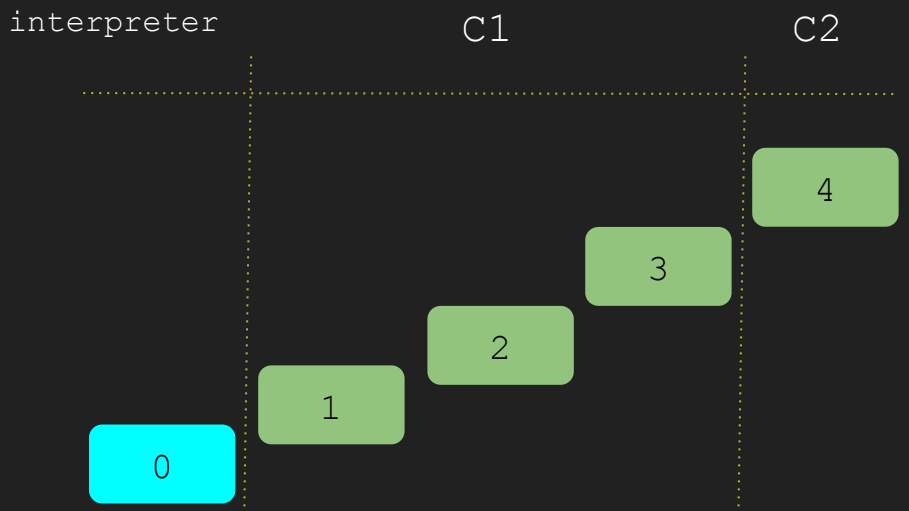

```
if (s == null) {
    uncommon_trap;
} else {
    return s;
}
```

```
Thread.sleep(3000);//не просто так
try {
    CheckNullUtils.checkNull(null);
} catch (Exception e) {//TODO}
```

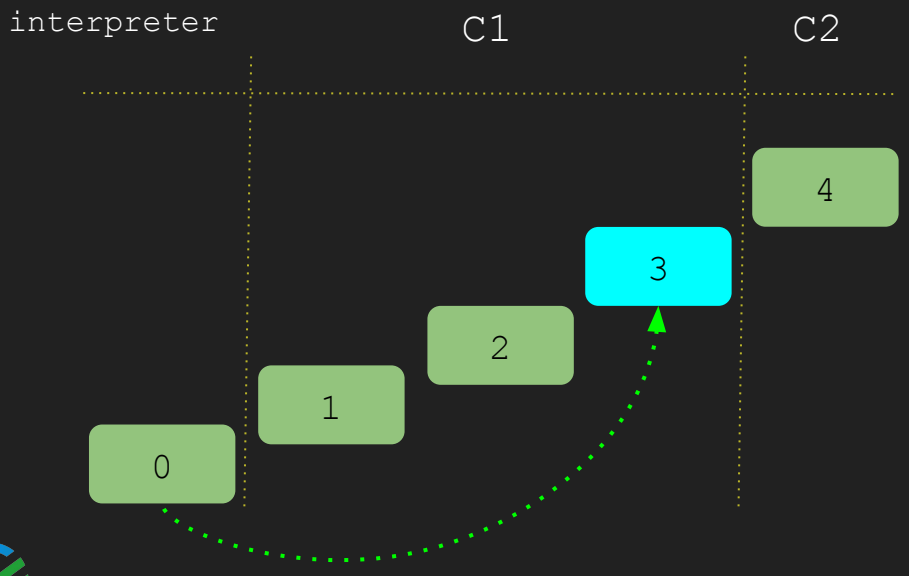



```
if (s == null) {
    uncommon_trap;
} else {
    return s;
}
```

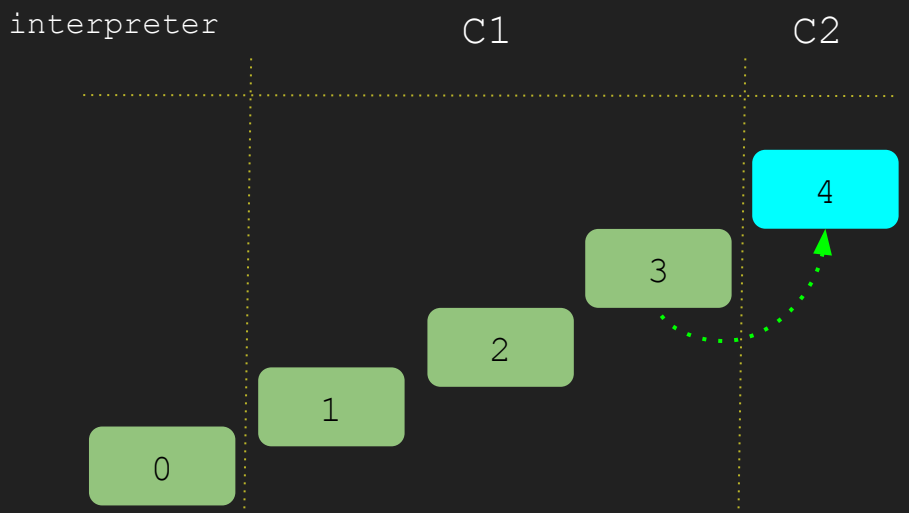
```
Thread.sleep(3000);
for (int i = 0; i < 10; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```



```
Thread.sleep(3000);
for (int i = 0; i < 200; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```



```
Thread.sleep(3000);
for (int i = 0; i < 5000; i++) {
    CheckNullUtils.checkNull(String.valueOf(i));
}
```



JitWatch поможет еще подробней посмотреть

1 https://github.com/AdoptOpenJDK/jitwatch

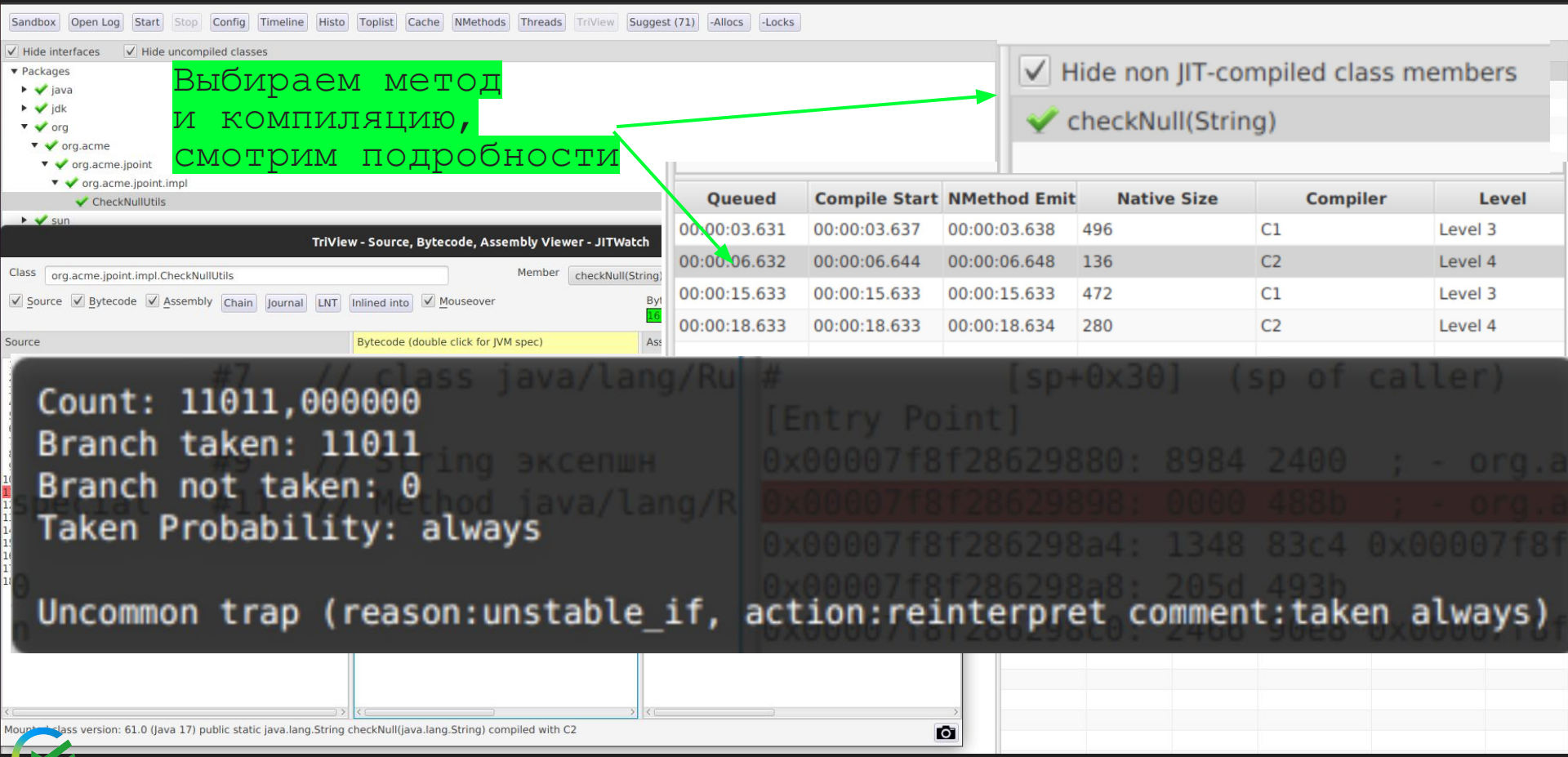
Загружаем JitWatch

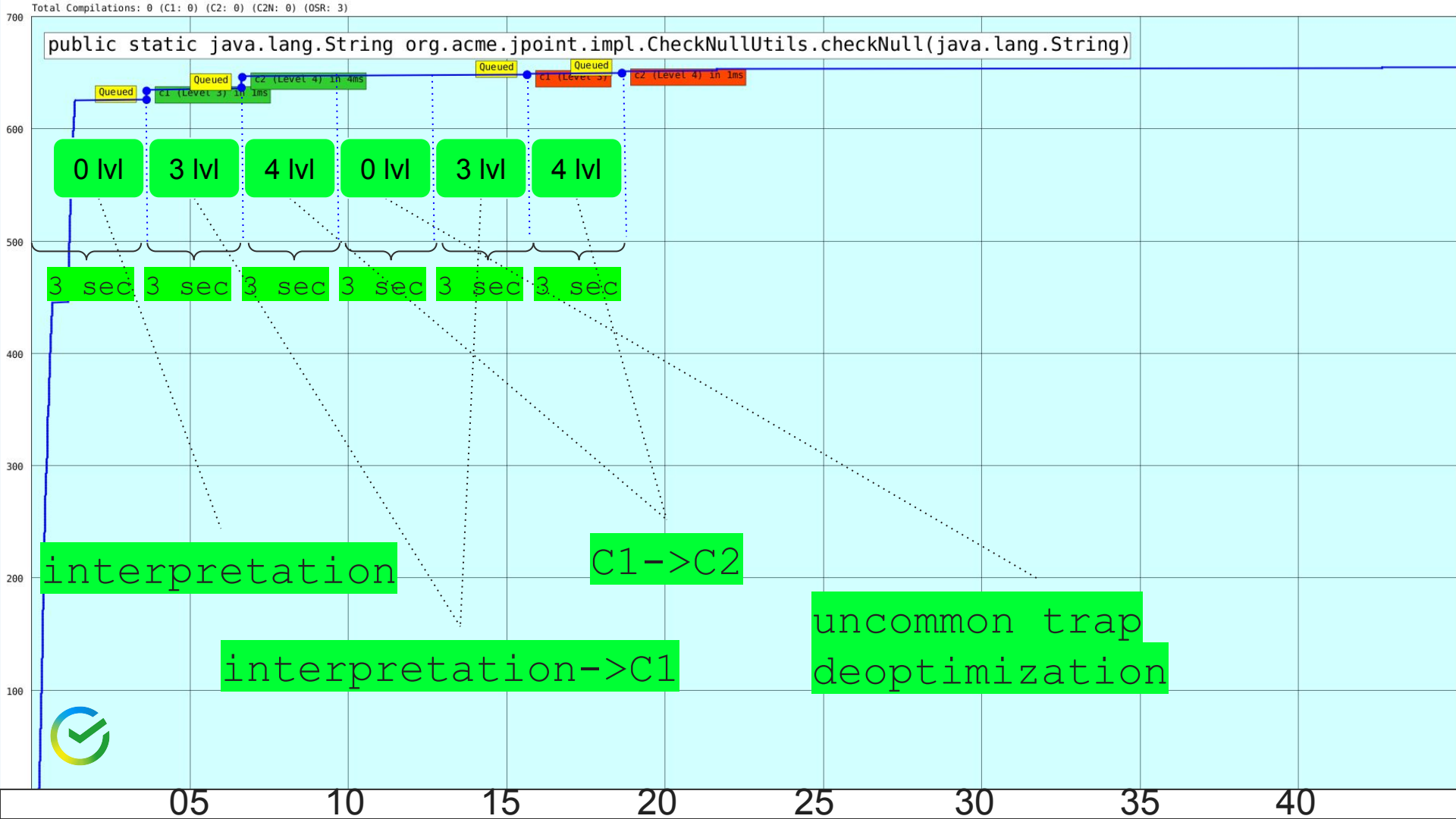
2
```
-XX:+UnlockDiagnosticVMOptions
-XX:+LogCompilation
-XX:+PrintAssembly
```

Собираем логи компиляции

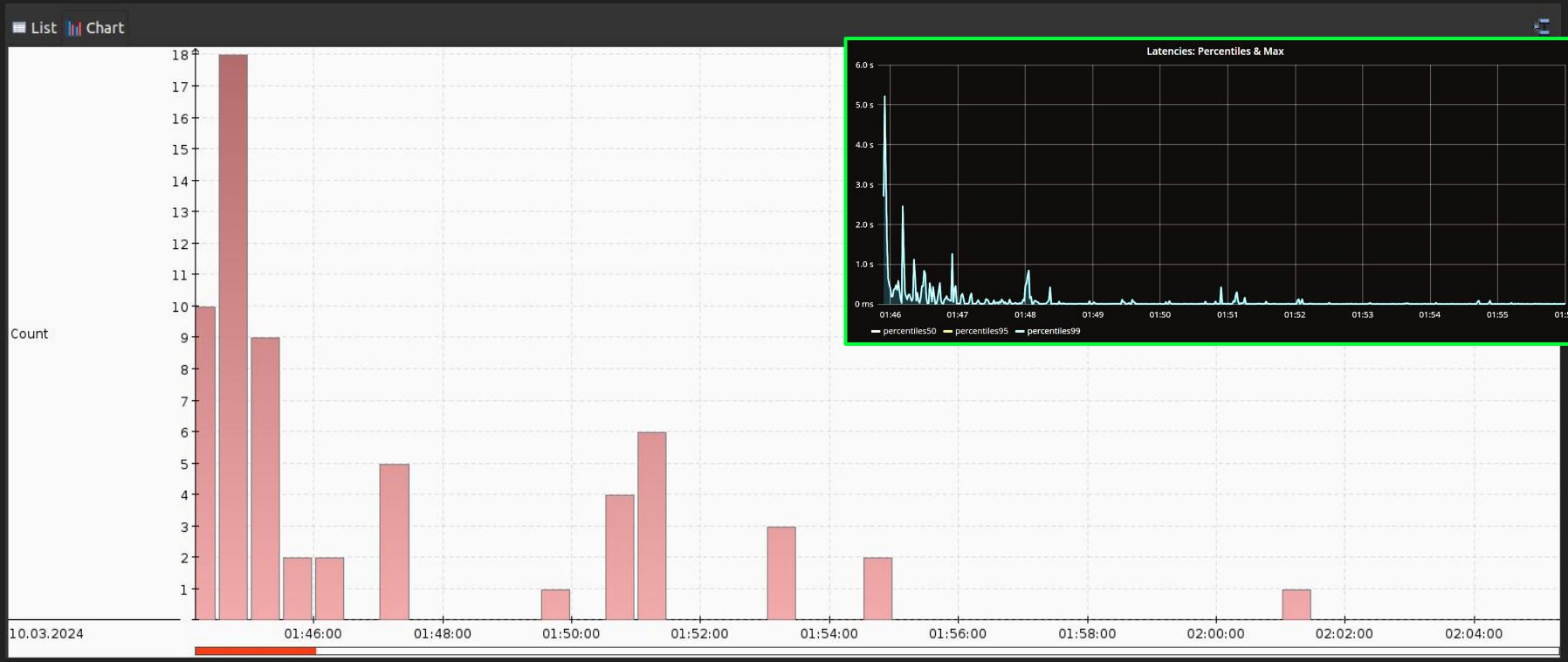
```
<task_queued compile_id='361' method='o.a.j.i.CheckNullUtils checkNull
(Ljava/lang/String;)Ljava/lang/String;'  level='3' stamp='3,631' comment='tiered'/>

<nmethod compile_id='361' compiler='c1' level='3'm
method='o.a.j.i.CheckNullUtils checkNull (Ljava/lang/String;)Ljava/lang/String;' bytes='16'
        stamp='3,638'/>

<task_queued compile_id='366' method='o.a.j.i.CheckNullUtils checkNull
(Ljava/lang/String;)Ljava/lang/String;' comment='tiered'/>

<nmethod compile_id='366' compiler='c2'  level='4' method='o.a.j.i.CheckNullUtils checkNull
(Ljava/lang/String;)Ljava/lang/String;/>
```


JitWatch


Sandbox
Open Log
Start
Stop
Config
Timeline
Histo
Toplist
Cache
NMethods
Threads
TriView
Suggest (71)
-Allocs
-Locks
Hide interfaces
Hide uncompiled classes
Packages
java
jdk
org
org.acme
org.acme.jpoint
org.acme.jpoint.impl
CheckNullUtils
sun
Выбираем метод
и компиляцию,
смотрим подробности
Hide non JIT-compiled class members
checkNull(String)
Queued
Compile Start
NMethod Emit
Native Size
Compiler
Level
00:00:03.631
00:00:03.637
00:00:03.638
496
C1
Level 3
00:00:06.632
00:00:06.644
00:00:06.648
136
C2
Level 4
00:00:15.633
00:00:15.633
00:00:15.633
472
C1
Level 3
00:00:18.633
00:00:18.633
00:00:18.634
280
C2
Level 4
TriView - Source, Bytecode, Assembly Viewer - JITWatch
Class
org.acme.jpoint.impl.CheckNullUtils
Member
checkNull(String)
Source
Bytecode
Assembly
Chain
Journal
LNT
Inlined into
Mouseover
Source
Bytecode (double click for JVM spec)
Count: 11011,000000
Branch taken: 11011
Branch not taken: 0
Taken Probability: always
Uncommon trap (reason:unstable_if, action:reinterpret comment:taken always)
[Entry Point]
0x00007f8f28629880: 8984 2400
0x00007f8f28629898: 0000 488b
0x00007f8f286298a4: 1348 83c4
0x00007f8f286298a8: 205d 493b
Major class version: 61.0 (Java 17) public static java.lang.String checkNull(java.lang.String) compiled with C2


Total Compilations: 0 (C1: 0) (C2: 0) (C2N: 0) (OSR: 3)
public static java.lang.String org.acme.jpoint.impl.CheckNullUtils.checkNull(java.lang.String)
Queued
c1 (Level 3) in 1ms
Queued
c2 (Level 4) in 4ms
Queued
Queued
c1 (Level 3)
c2 (Level 4) in 1ms
0 lvl
3 lvl
4 lvl
0 lvl
3 lvl
4 lvl
3 sec
3 sec
3 sec
3 sec
3 sec
3 sec
interpretation
interpretation->C1
C1->C2
uncommon trap
deoptimization
700
600
500
400
300
200
100
05
10
15
20
25
30
35
40

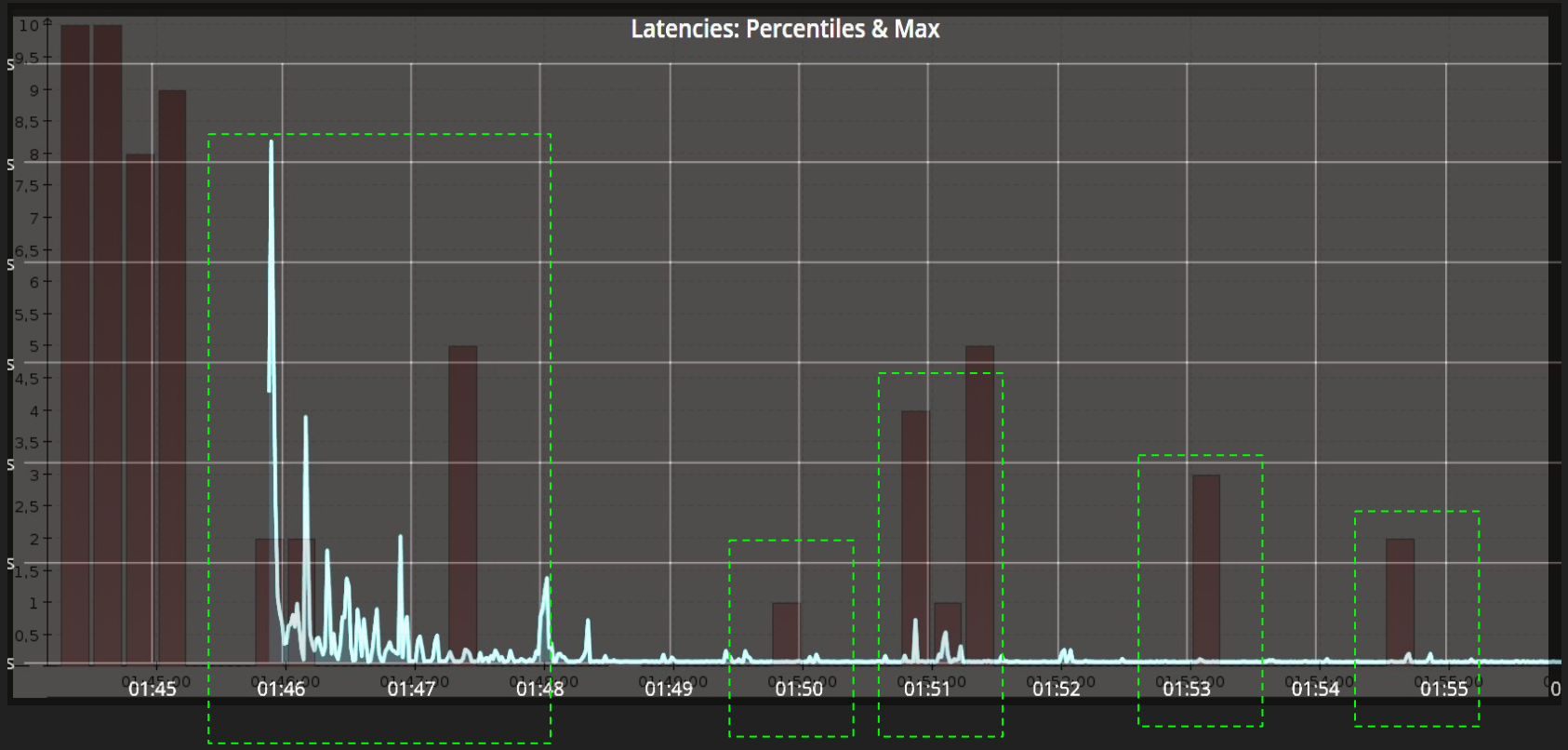
Deoptimization analysis jfr



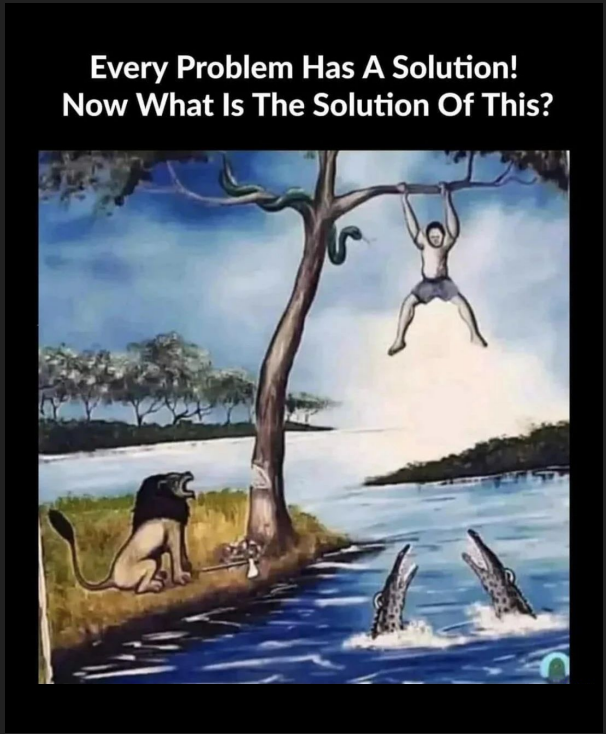
Deoptimization group by method

Method	Count
HashMap$Node java.util.HashMap.getNode(Object)	7
Object java.util.HashMap.putVal(int, Object, Object, boolean, boolean)	6
Object java.util.concurrent.ConcurrentHashMap.putVal(Object, Object, boolean)	5
void java.util.Arrays.fill(Object[], int, int, Object)	5
Object jdk.internal.misc.Unsafe.allocateUninitializedArray(Class, int)	4
void java.util.concurrent.ConcurrentHashMap.addCount(long, int)	3
Promise io.netty.util.concurrent.DefaultPromise.addListener(GenericFutureListener)	3
Object io.netty.util.internal.shaded.org.jctools.queues.MpscArrayQueue.poll()	2
boolean io.netty.util.internal.shaded.org.jctools.queues.BaseMpscLinkedArrayQueue.offer(C	2
boolean io.smallrye.config.PropertyName.equals(String, String)	2
int java.util.HashMap.hash(Object)	2
void io.netty.util.concurrent.AbstractEventExecutor.runTask(Runnable)	2
boolean io.netty.buffer.PoolThreadCache.allocate(PoolThreadCache$MemoryRegionCache, I	2
void java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await()	1
int sun.nio.ch.EPollSelectorImpl.doSelect(Consumer, long)	1
boolean java.lang.String.isLatin1()	1
void java.util.concurrent.locks.ReentrantLock$Sync.lock()	1
void io.vertx.ext.web.impl.HandlersList.invokeInReverseOrder(Object)	1
int io.smallrye.config.PropertyName.hashCode()	1
Object io.netty.util.internal.shaded.org.jctools.queues.BaseMpscLinkedArrayQueue.poll()	1

latency/deoptimization



Выявили причины тормозов на старте?

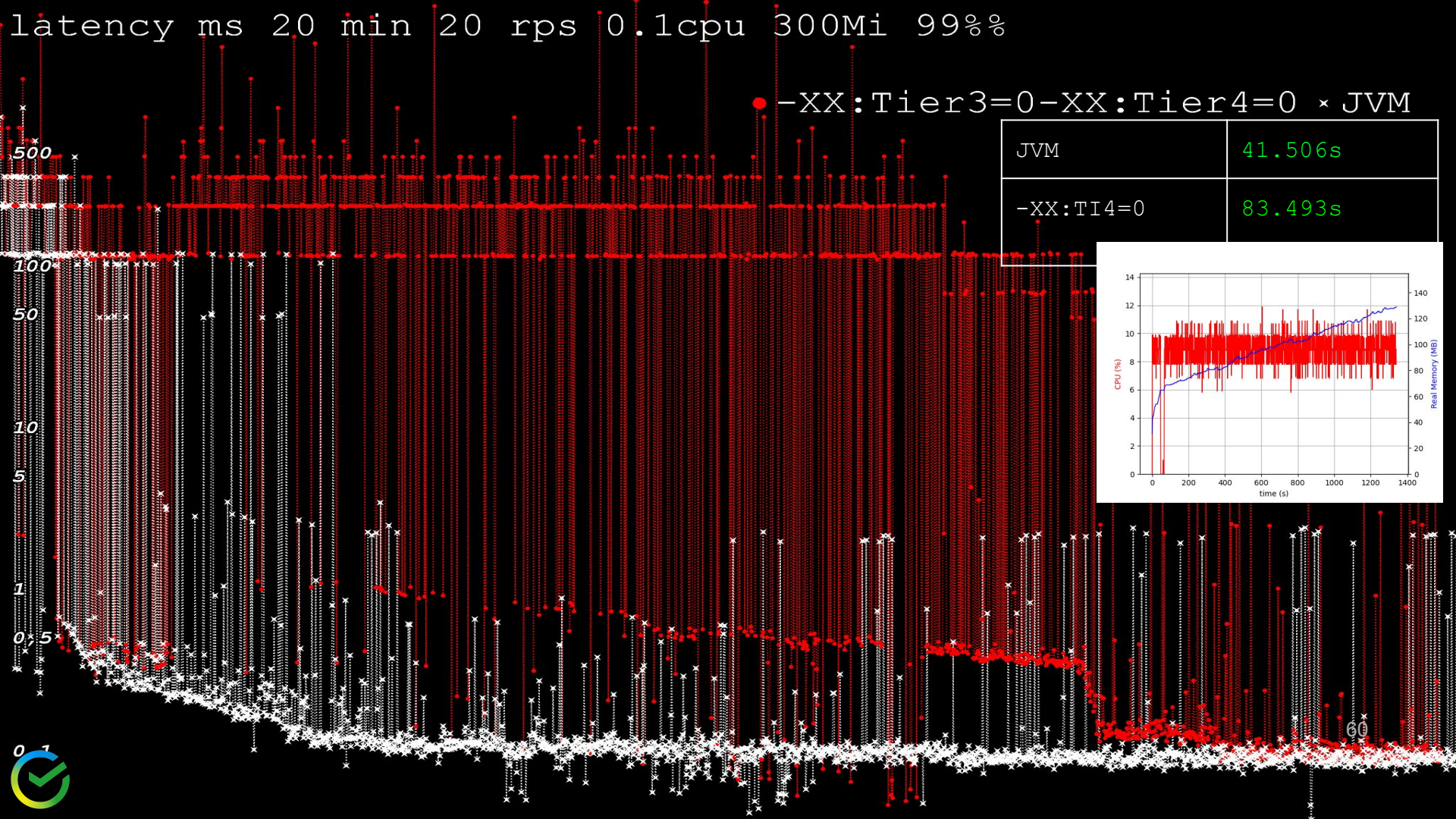


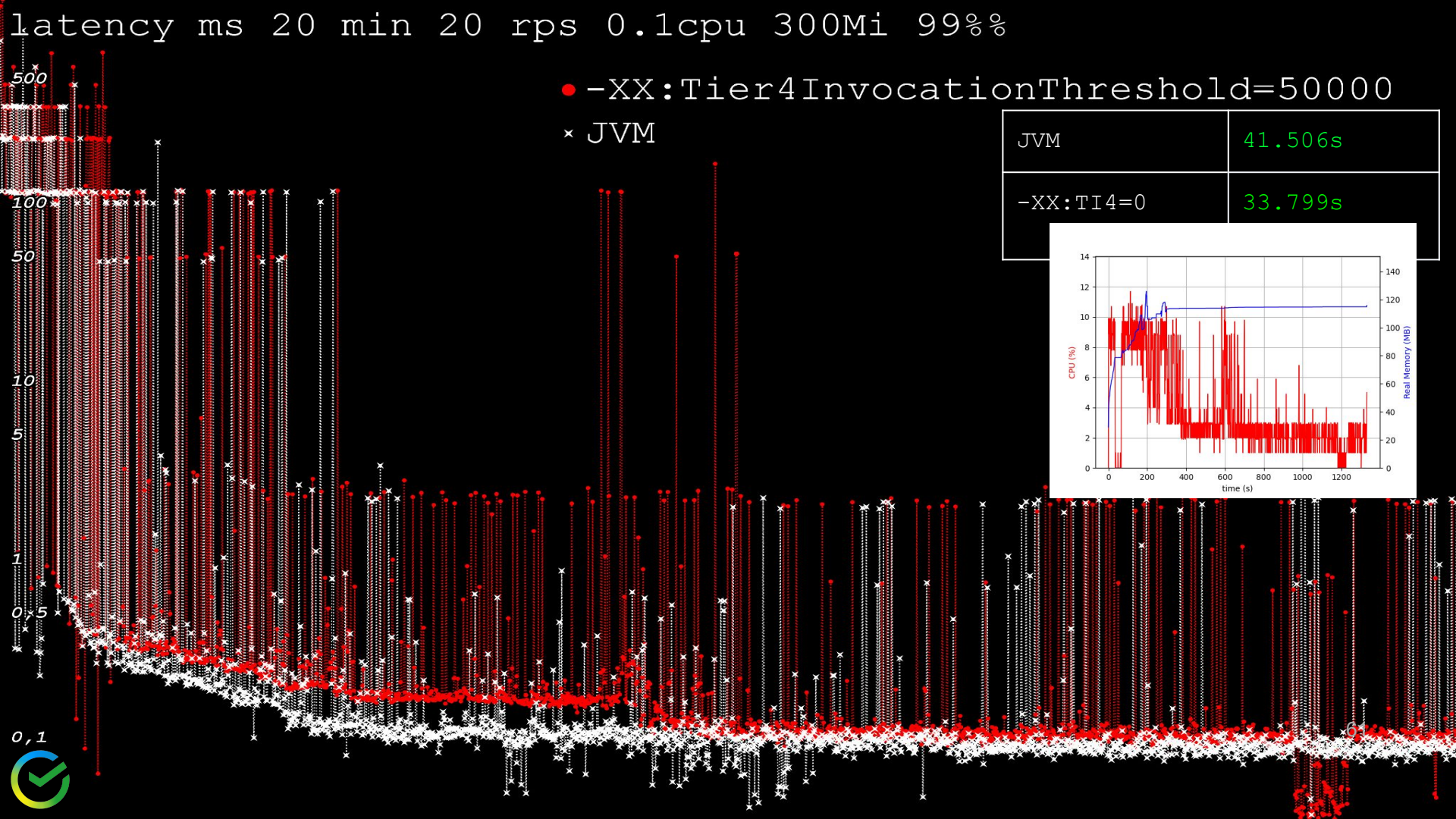
- Работа в режиме интерпретации на старте
- deoptimization
- Динамизм java,загрузка классов и т.д.
- хоть и мало, тратим ресурсы на компиляцию

Сразу понятно что делать)- нужно играть с ключиками

-XX:Tier3InvocationThreshold=?

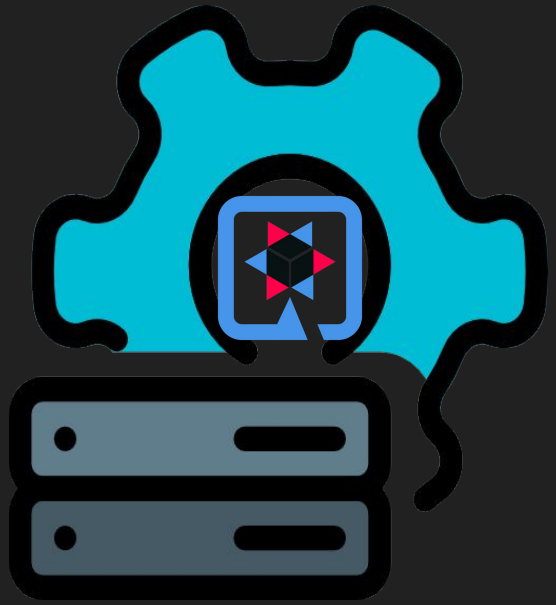
-XX:Tier4InvocationThreshold=?


latency ms 20 min 20 rps 0.1cpu 300Mi 99%%
-XX:Tier3=0-XX:Tier4=0 × JVM
JVM 41.506s
-XX:TI4=0 83.493s
500
100
50
10
5
1
0.5
0.1
60
CPU (%)
Real Memory (MB)
time (s)
0
200
400
600
800
1000
1200
1400
2
4
6
8
10
12
14
20
40
60
80
100
120
140


latency ms 20 min 20 rps 0.1cpu 300Mi 99%%
-XX:Tier4InvocationThreshold=50000
JVM
500
100
50
10
5
1
0,5
0,1
JVM 41.506s
-XX:TI4=0 33.799s
CPU (%)
Real Memory (MB)
time (s)

-XX:Tier3InvocationThreshold -XX:Tier4InvocationThreshold

- ? cpu,memory
- ? start speed
- ? Warm-up period



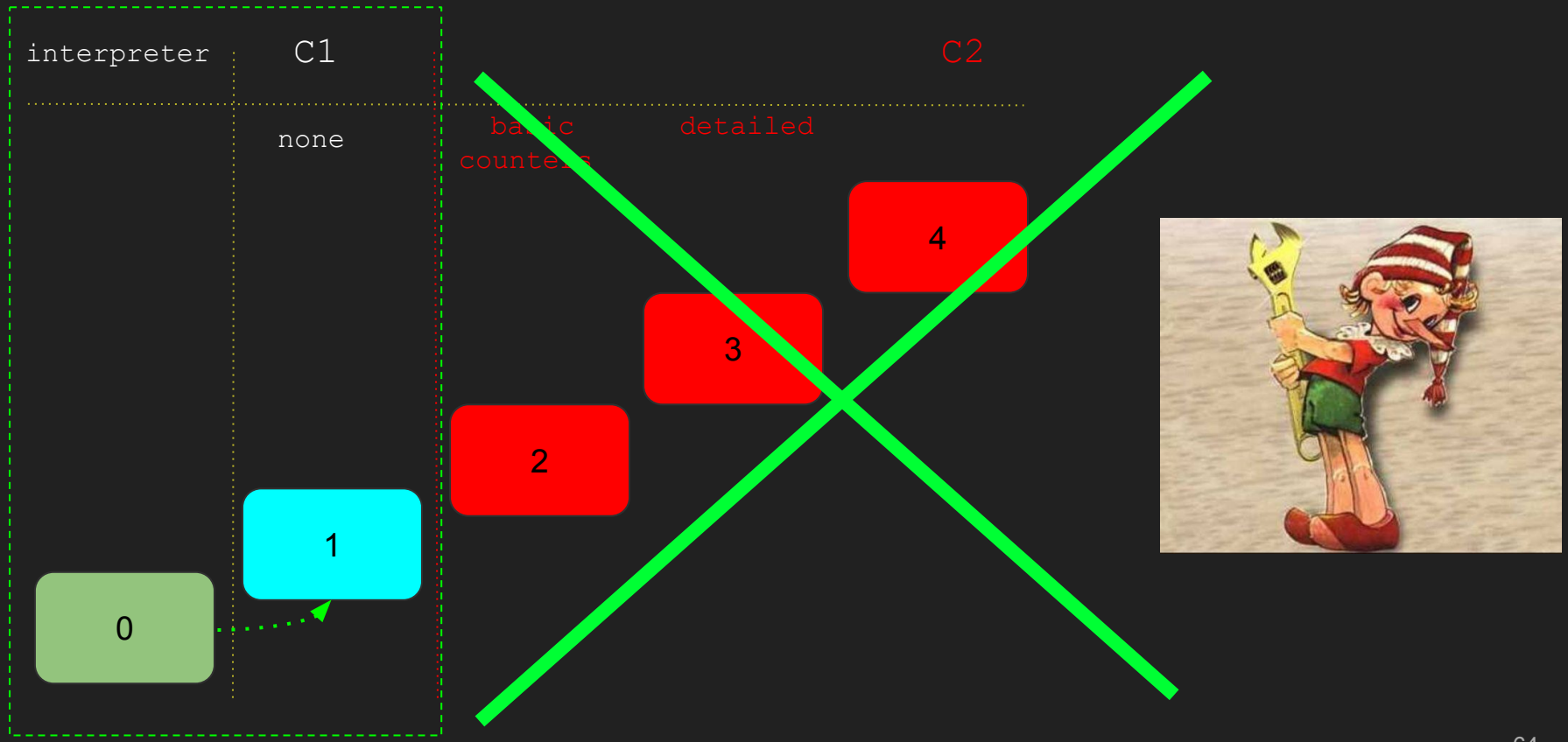
- devops
- development
- ? throughput

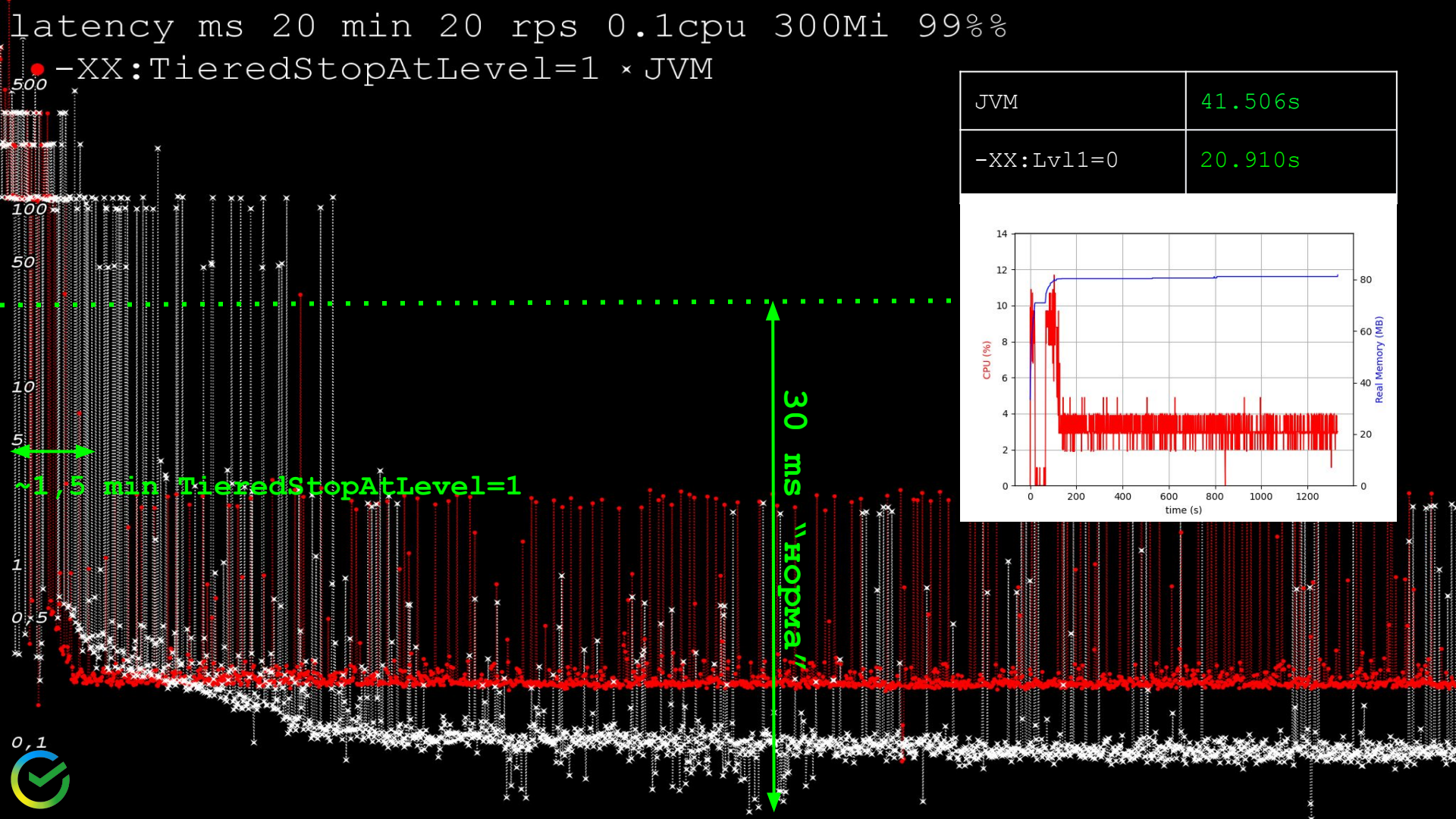
Играл с ключиками и проиграл?

-XX:Tier3InvocationThreshold=?

-XX:Tier4InvocationThreshold=?

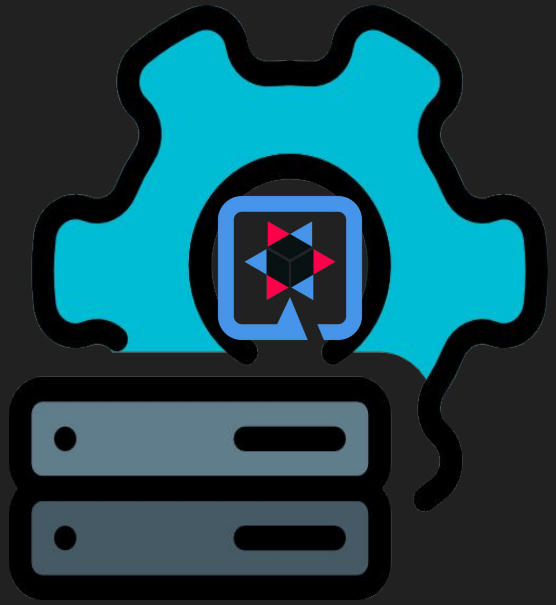
-XX:TieredStopAtLevel=1




latency ms 20 min 20 rps 0.1cpu 300Mi 99%%
-XX:TieredStopAtLevel=1 × JVM
500
100
50
10
5
1
0,5
0,1
~1,5 min TieredStopAtLevel=1
30 ms "норма"
JVM
41.506s
-XX:Lvl1=0
20.910s
CPU (%)
Real Memory (MB)
time (s)
0
200
400
600
800
1000
1200
2
4
6
8
10
12
14
20
40
60
80

-XX:TieredStopAtLevel=1

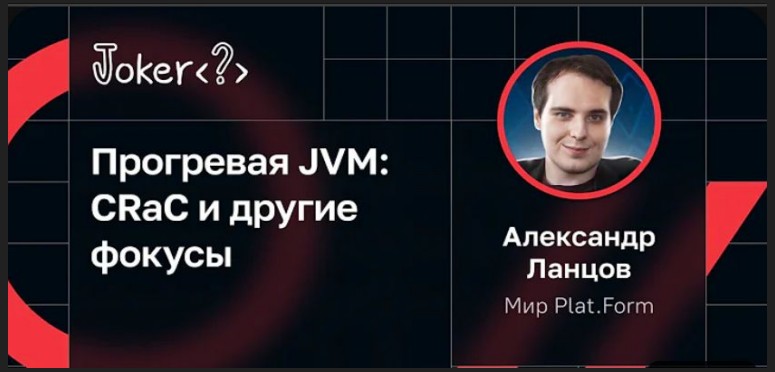
- cpu,memory
- start speed
- Warm-up period



- devops
- development
- throughput

CRaC

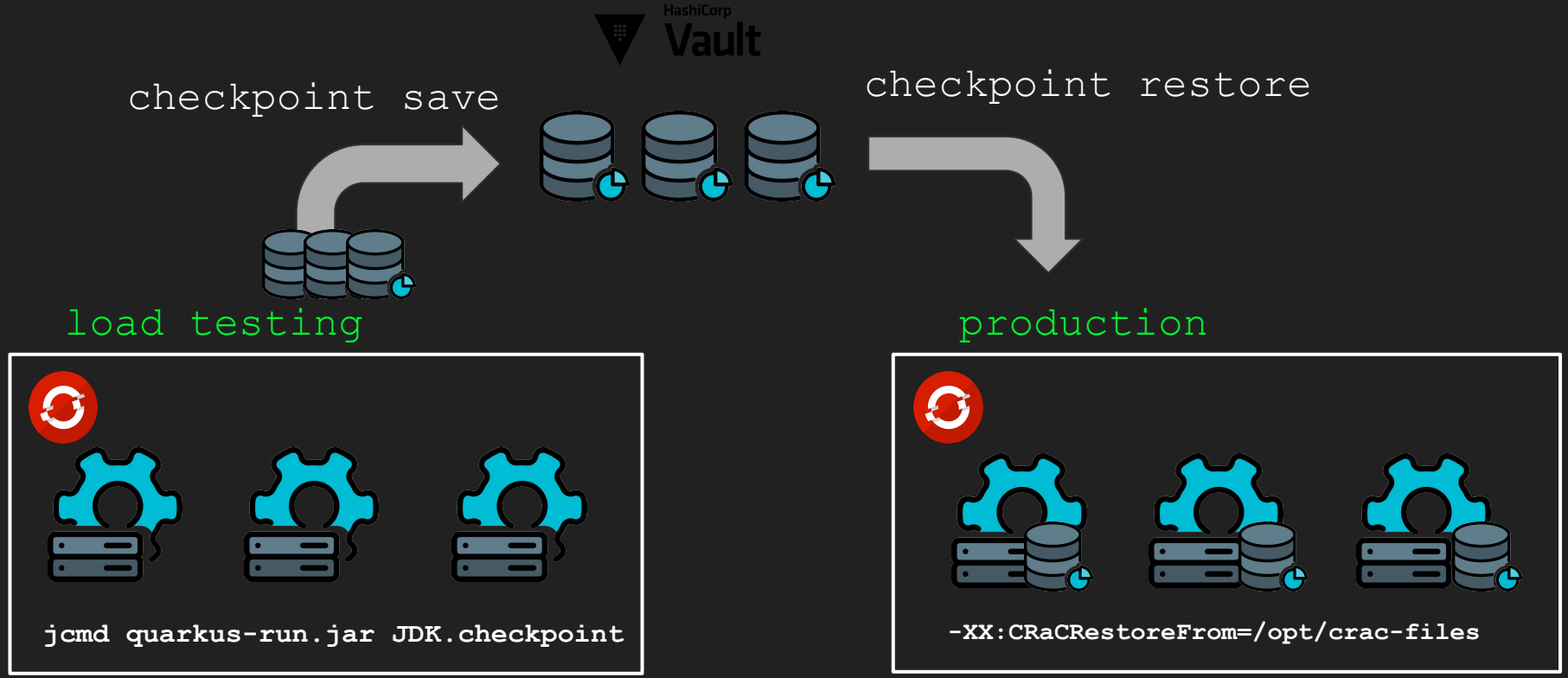
https://www.youtube.com/watch?v=RLFQj2mPqUM&t=182s



https://www.youtube.com/watch?v=2sPY6x5Lg8c&t=2418s

Хорошие ребята

CRaC


HashiCorp
Vault
checkpoint save
checkpoint restore
load testing
production
jcmd quarkus-run.jar JDK.checkpoint
-XX:CRaCRestoreFrom=/opt/crac-files

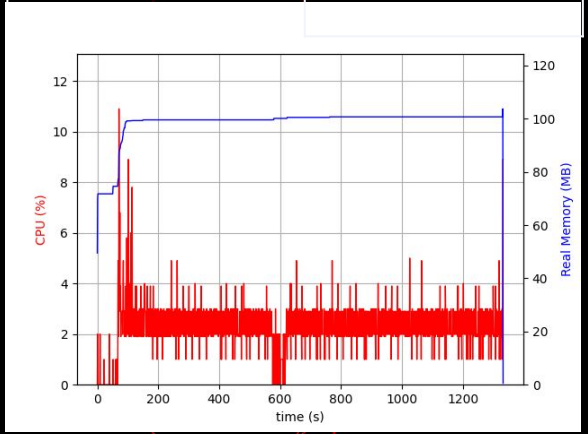
latency ms 20 min 20 rps 0.1cpu 300Mi 99%%

CRaC × JVM

500
100
50
10
5
1
0,5
0,1

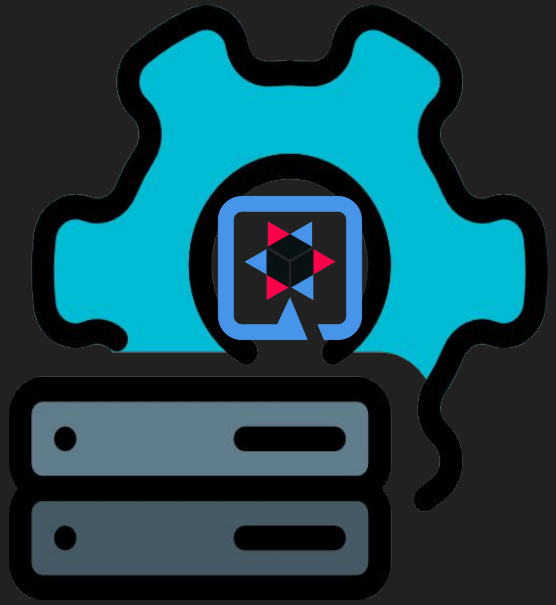
30 ms "норма"

JVM	41.506s
CrAC	->0



CRaC было бы идеально, НО

- cpu,memory
- start speed
- Warm-up period



- devops
- development
- throughput

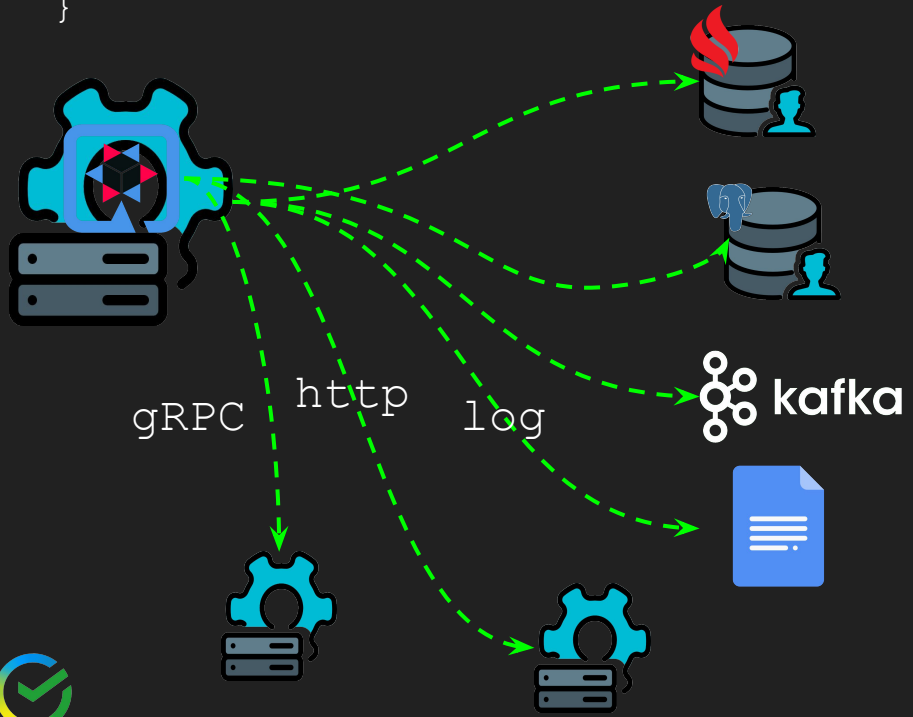
HO

An exception during a checkpoint operation:
jdk.internal.crac.mirror.CheckpointException

```
Suppressed: jdk.internal.crac.mirror.impl.CheckpointOpenSocketException:
sun.nio.ch.ServerSocketChannelImpl[/0.0.0.0:9000]
                at
java.base/jdk.internal.crac.JDKSocketResourceBase.lambda$beforeCheckpoint$0(JDKSo
cketResourceBase.java:68)
                at
java.base/jdk.internal.crac.mirror.Core.checkpointRestore1(Core.java:169)
                at
java.base/jdk.internal.crac.mirror.Core.checkpointRestore(Core.java:286)
                at
java.base/jdk.internal.crac.mirror.Core.checkpointRestoreInternal(Core.java:299)
```


START/STOP

```
public interface Resource {
    void beforeCheckpoint(Context<? extends Resource> var1) throws Exception;

    void afterRestore(Context<? extends Resource> var1) throws Exception;
}
```



pull request
add grpc CrAC
support

https://github.com/quarkusio/quarkus/pull/40169

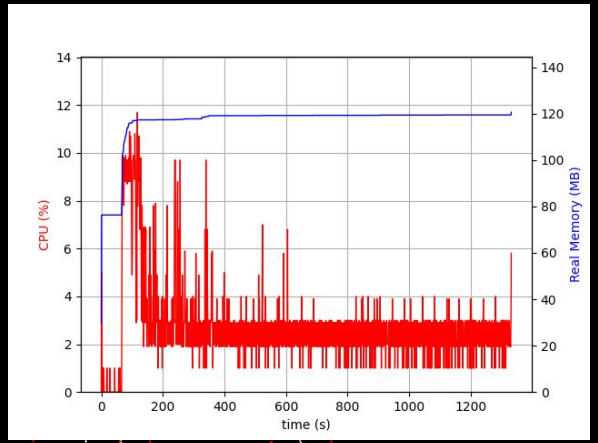
latency ms 20 min 20 rps 0.1mi 300Mi 99%%

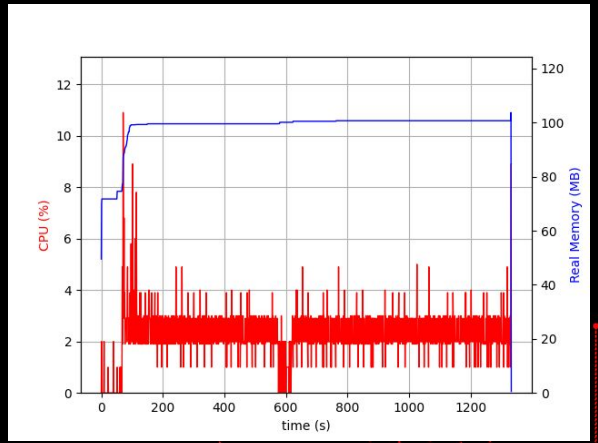
● CRaC × JVM

500
100
50
10
5
1
0,5
0,1

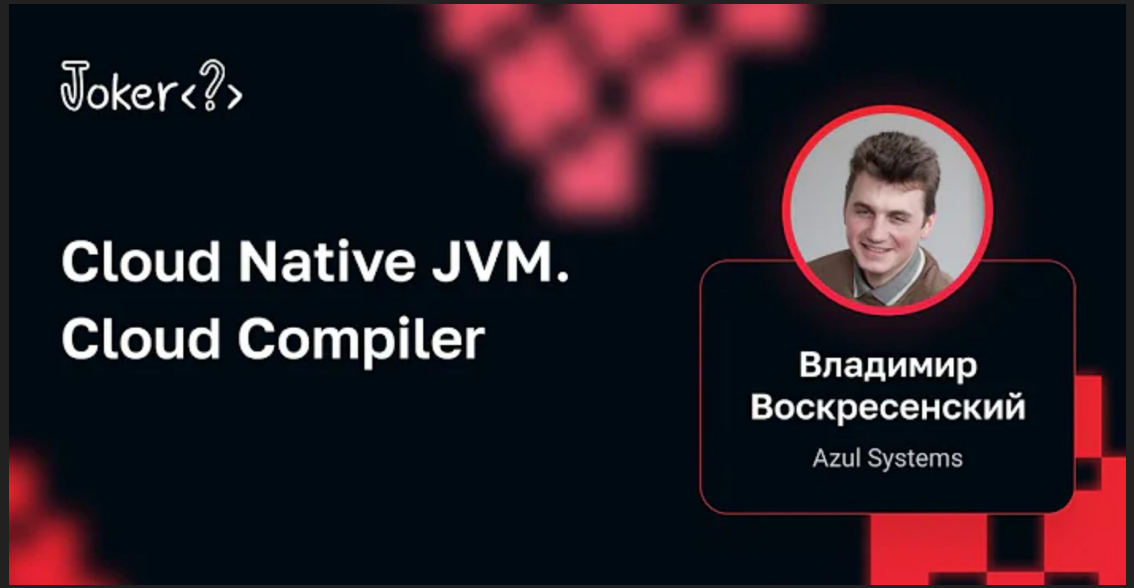
~3 min CRaC

30 ms \ hour



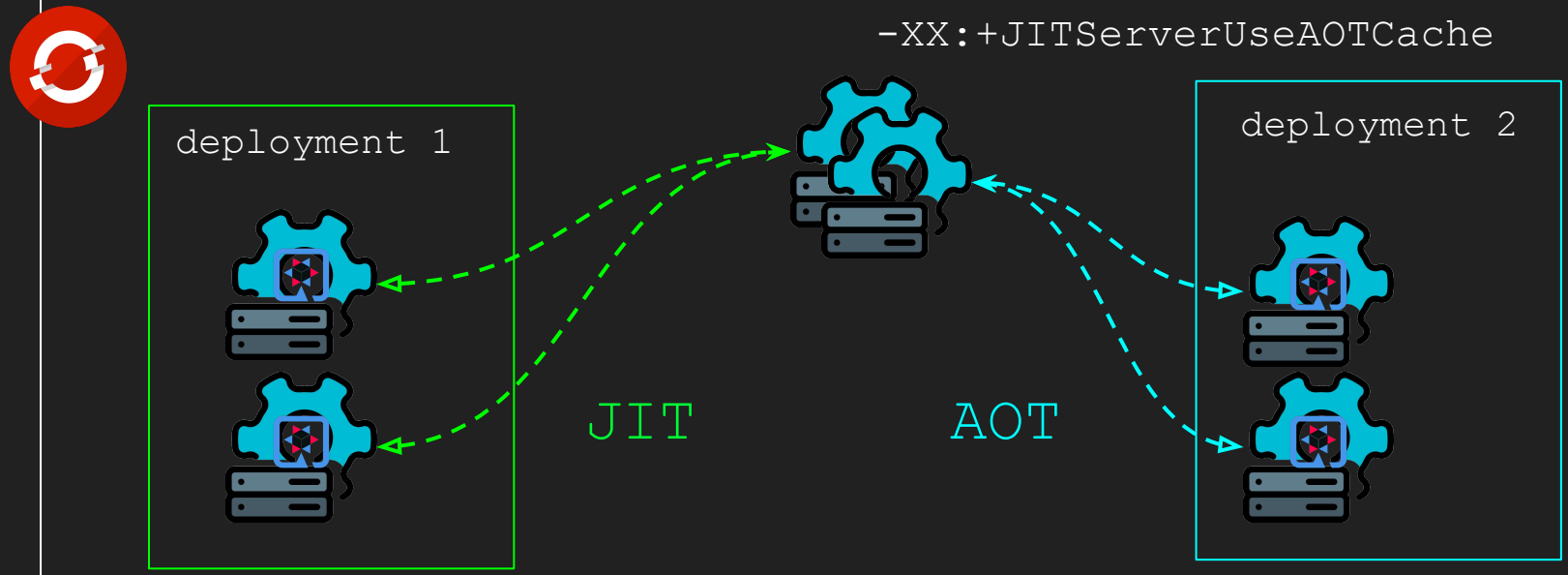


А что если попробовать “удаленно” компилировать или даже хранить AOT профиль?



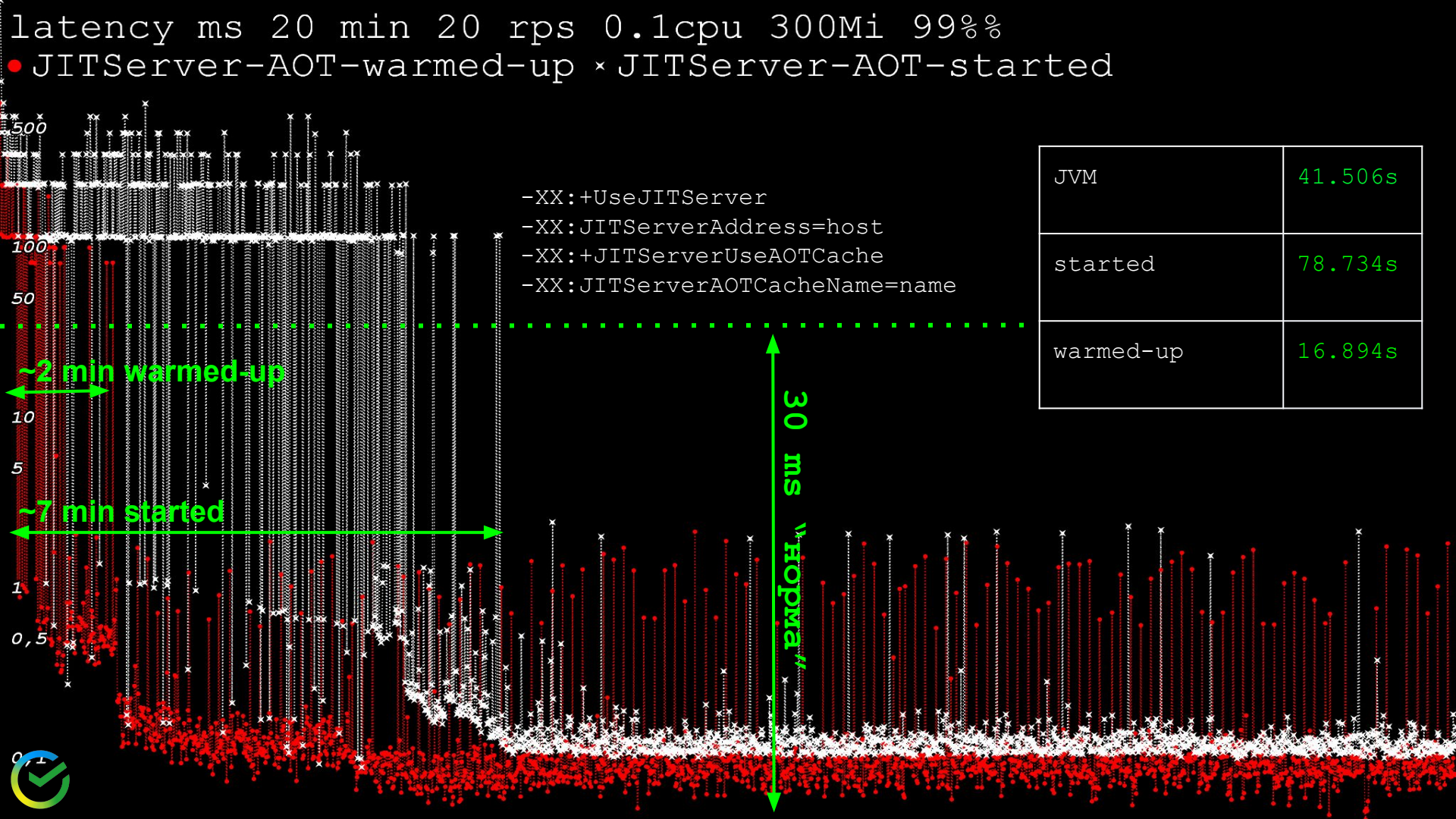
https://www.youtube.com/watch?v=KFgnB8I6p_U

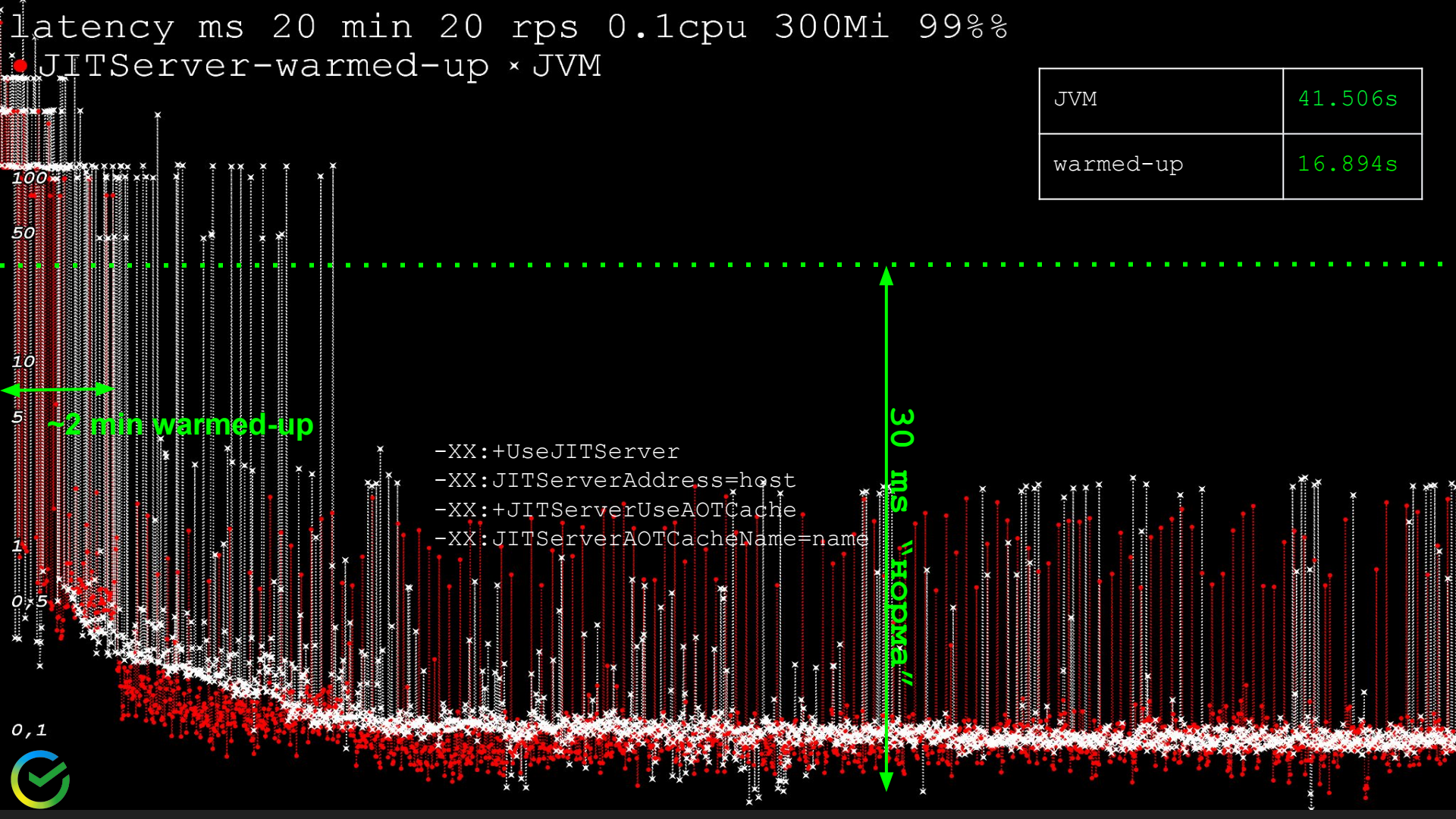
JitServer Eclipse OpenJ9



-XX:+UseJITServer
-XX:JITServerAddress=host

-XX:+UseJITServer
-XX:JITServerAddress=host
-XX:+JITServerUseAOTCache
-XX:JITServerAOTCacheName=name


latency ms 20 min 20 rps 0.1cpu 300Mi 99%%
JITServer-AOT-warmed-up
JITServer-AOT-started
500
100
50
10
5
1
0,5
0,1
-XX:+UseJITServer
-XX:JITServerAddress=host
-XX:+JITServerUseAOTCache
-XX:JITServerAOTCacheName=name
~2 min warmed-up
~7 min started
30 ms "норма"
JVM 41.506s
started 78.734s
warmed-up 16.894s


latency ms 20 min 20 rps 0.1cpu 300Mi 99%%
JITServer-warmed-up × JVM
JVM 41.506s
warmed-up 16.894s
100
50
10
5
1
0,5
0,1
~2 min warmed-up
-XX:+UseJITServer
-XX:JITServerAddress=host
-XX:+JITServerUseAOTCache
-XX:JITServerAOTCacheName=name
30 ms "HOPM"

latency ms 20 min 20 rps 0.1cpu 300Mi 99%%

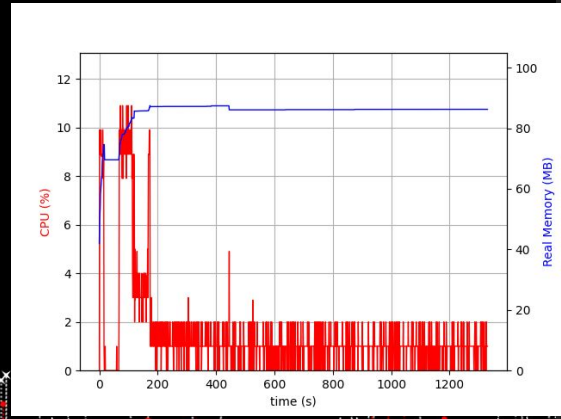
JVM ● JitServer-JIT

JVM	41.506s
started	13.672s

-XX:+UseJITServer
-XX:JITServerAddress=host

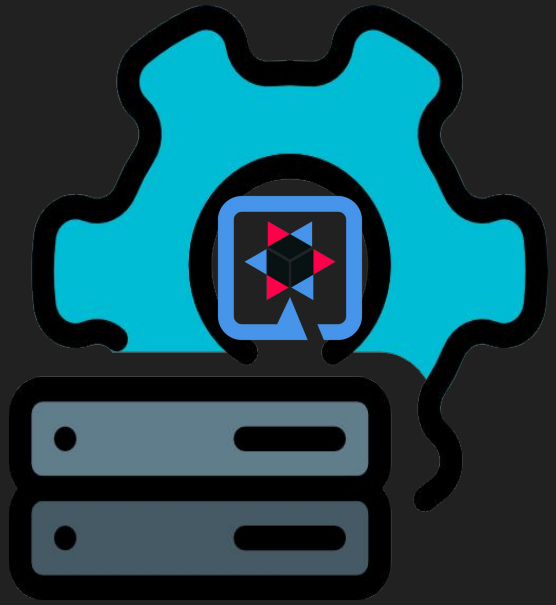
100
50
10
5
1
0,5
0,1

~1 min JitServer



JITServer

- cpu,memory
- start speed
- Warm-up period



- devops
- development
- throughput

GraalVM Native image

JIT

- ОС загружает исполняемый файл JVM
- VM начинает загружать классы
- верификация байт-кода
- интерпретация байт-кода
- Статические инициализаторы
- C1
- профилирование
- C2
- И только после этого всего работа с оптимизированным машинным кодом

Native image

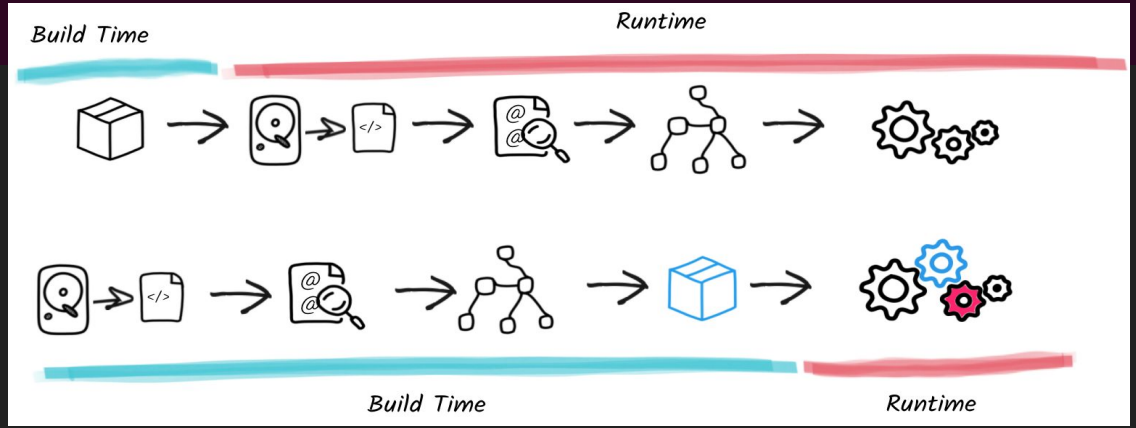
- ОС загружает исполняемый файл с подготовленной кучей
- Приложение запускается сразу с оптимизированным машинным кодом.

Native image start

-H:+PrintClassInitialization

BUILD_TIME VS RUN_TIME

```
andrey@andrey-desktop:~/IdeaProjects/latency-rest/target/latency-rest-1.0.0-SNAPSHOT-native-image-source-jar/reports$ cat class_initialization_report_20240326_193946.csv | grep BUILD_TIME -c
12944
andrey@andrey-desktop:~/IdeaProjects/latency-rest/target/latency-rest-1.0.0-SNAPSHOT-native-image-source-jar/reports$ cat class_initialization_report_20240326_193946.csv | grep RUN_TIME -c
166
```

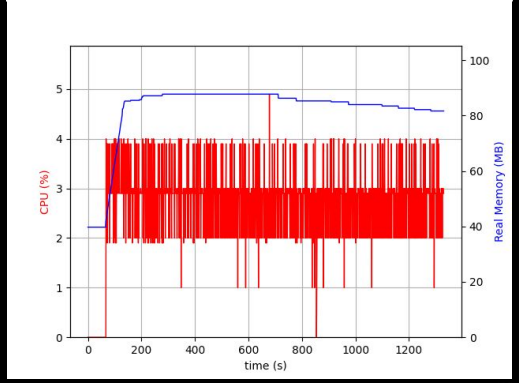


latency ms 20 min 20 rps 0.1cpu 300Mi 99%%

native × JVM

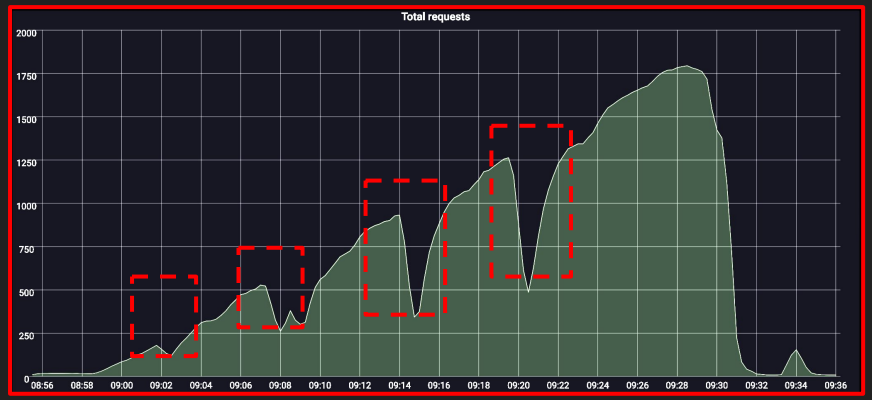
500
100
50
10
5
1
0,5
0,1

JVM	41.506s
started	0.11s



82

Автомасштабирование - возвращение



JVM -

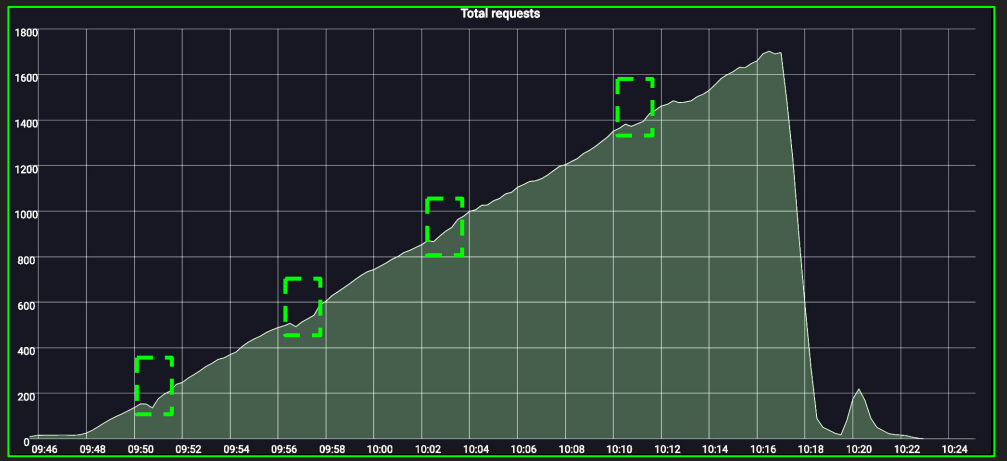
timeout

retry

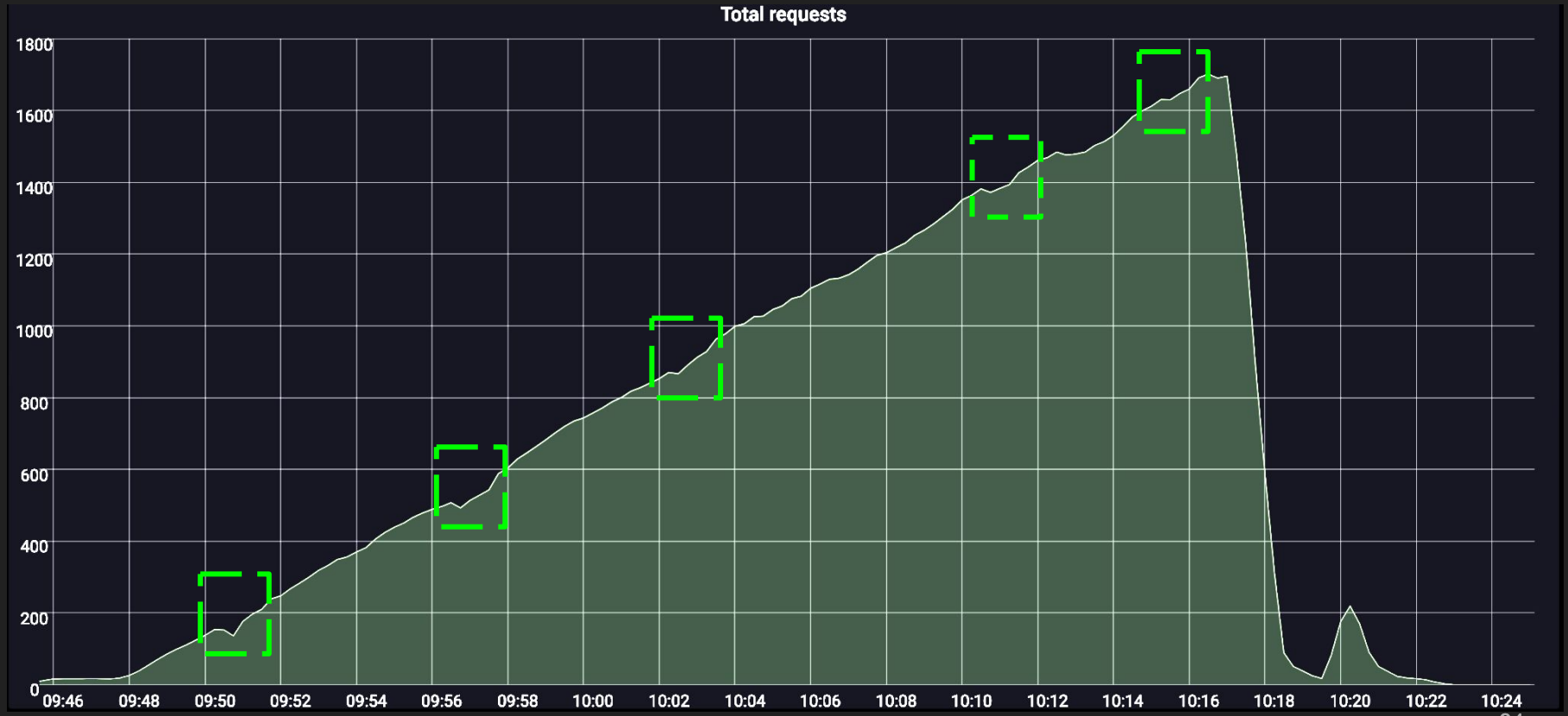
HTTP/1.1 429 Too Many Requests

Native image -

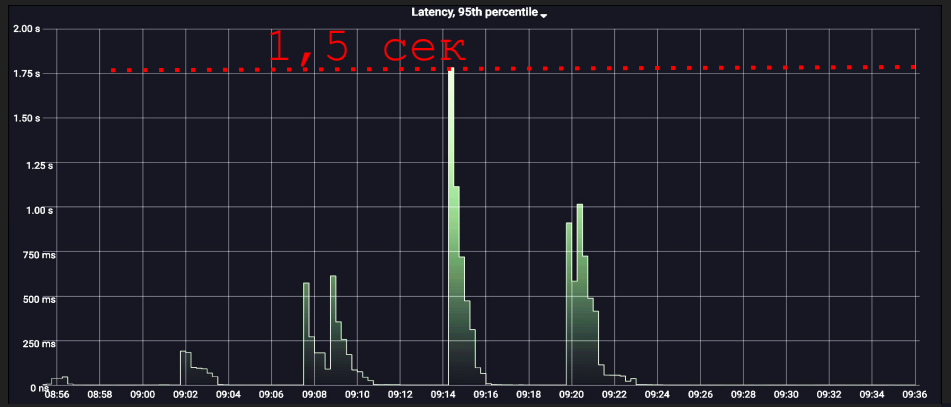
видно небольшие скачки при вводе в балансировку



Почти линейный график, красота!



native latency

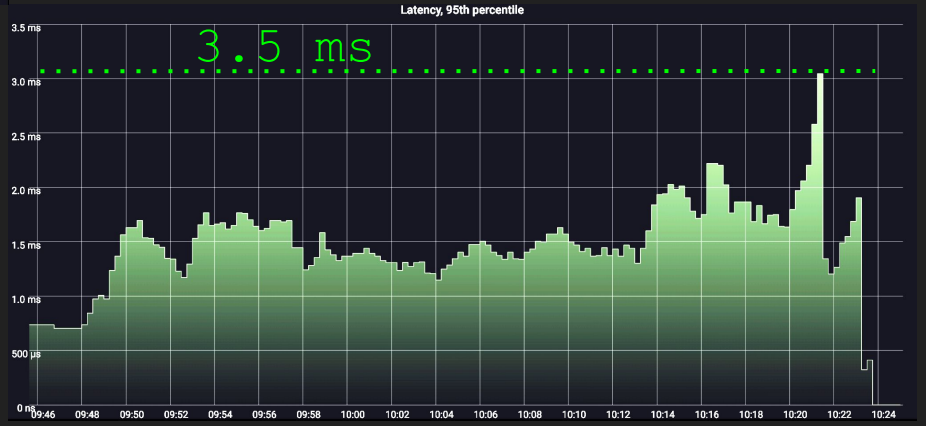


JVM –

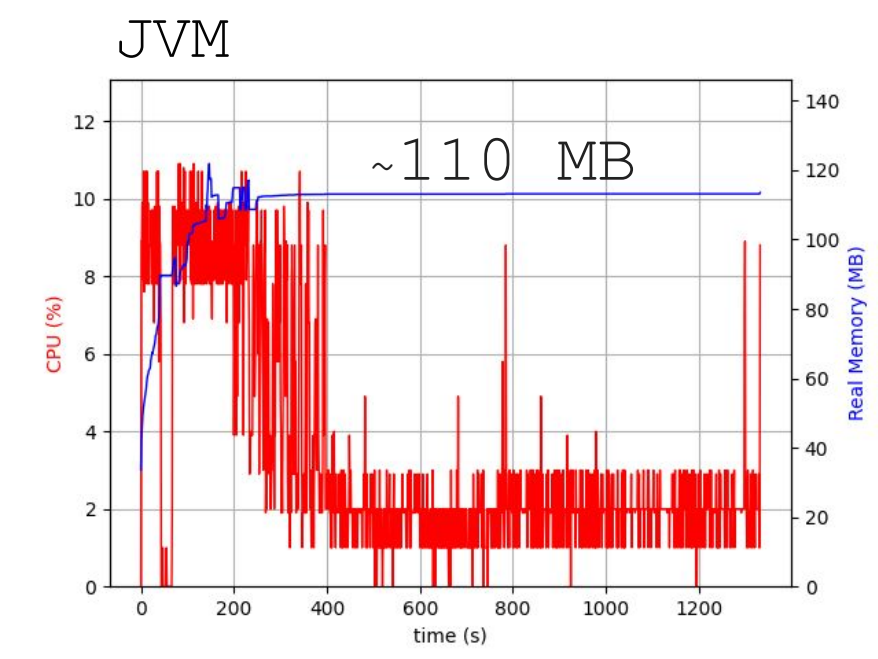
max latency >1.5 sec

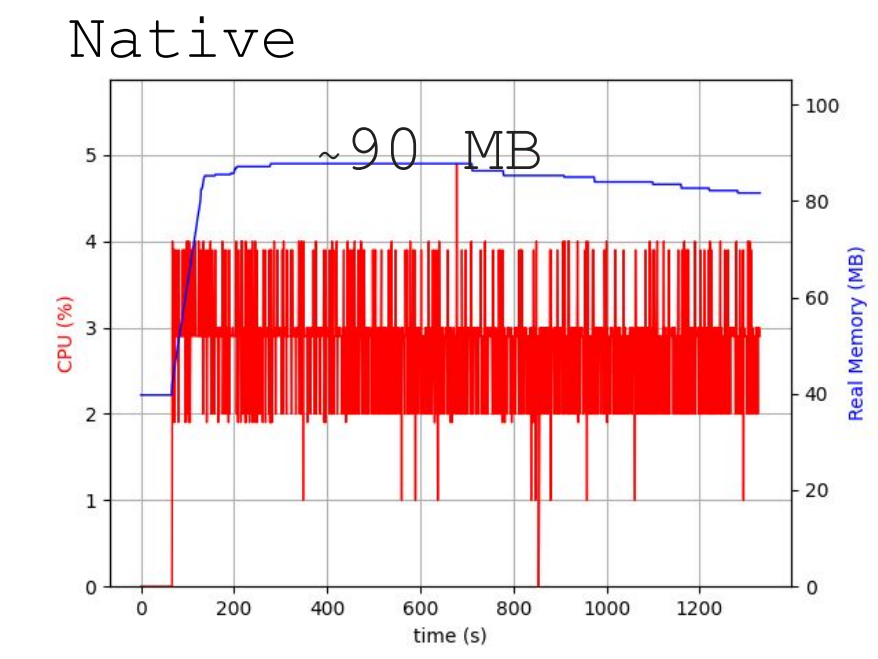
Native image–

max latency < 3.5 ms



Native память





Native память

JIT

Garbage Collector	Virtual Machine Runtime and Compiler
Dynamic Code Cache	Metaspace Class Files
Profiling Feedback	Compilation Data Structures
Application payload	

AOT

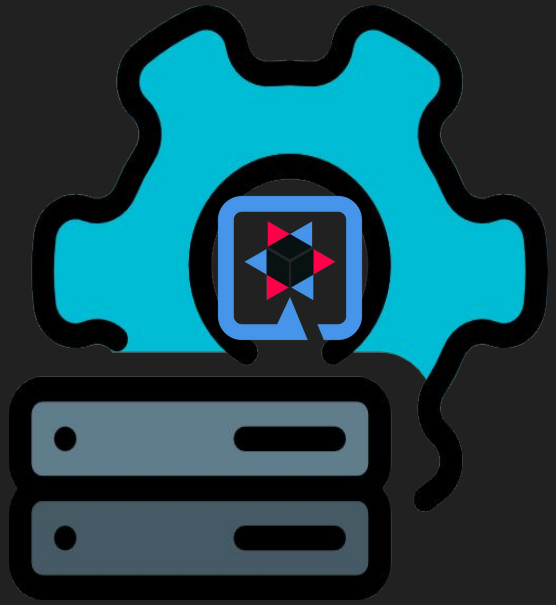
Garbage Collector	Application Machine Code
Application payload	

Native - успех!

cpu,memory
0.1 m
150 Mi

start speed
< 1 сек

Warm-up period
0



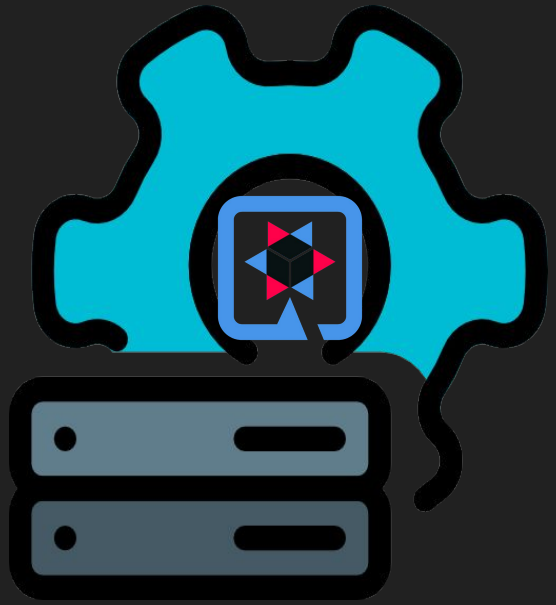
devops

development

throughput

Native - но все таки, не все бесплатно!

- cpu,memory
- start speed
- Warm-up period



- devops
- development
- throughput

Native build

JVM

```
[INFO] BUILD SUCCESS
[INFO]
------------------------
------------------------
------------------
[INFO] Total time:
14.236 s
```

VS

NATIVE

```
[INFO] BUILD SUCCESS
[INFO]
------------------------
------------------------
------------------
[INFO] Total time:
01:31 min
```

Graal compiler: optimization level: 2 - default

агрессивные оптимизации

-Dquarkus.native.additional-build-args=-O0

44.237 s

Native тесты для слабаков

- Unit тесты достаточно сложно написать в случае с native,есть агент https://graalvm.github.io/native-build-tools/latest/maven-plugin-quickstart.html
- Интеграционные — пожалуйста.
- Ну и опять же, прогнать на jvm, хотя это уже не то.

Придется программировать помимо XML и YAML на json

predefined-classes-config.json

proxy-config.json

resource-config.json

serialization-config.json

reflect-config.json

jni-config.json

Динамизм JAVA

Reflection

```
public class BlackBox implements Serializable {

    private String boxName;

    public BlackBox(String boxName) {
        this.boxName = boxName;
    }

    public String getBoxName() {
        return boxName;
    }

    public void setBoxName(String boxName) {
        this.boxName = boxName;
    }
}
```


Reflection

```
import com.fasterxml.jackson.databind.ObjectMapper;
…
public class ReflectionProblemService {

    private static final ObjectMapper objectMapper = new ObjectMapper();

    public String invokeReflection() {
        BlackBox box = new BlackBox("box");
        return objectMapper.writeValueAsString(box);
    }
}
```


Reflection

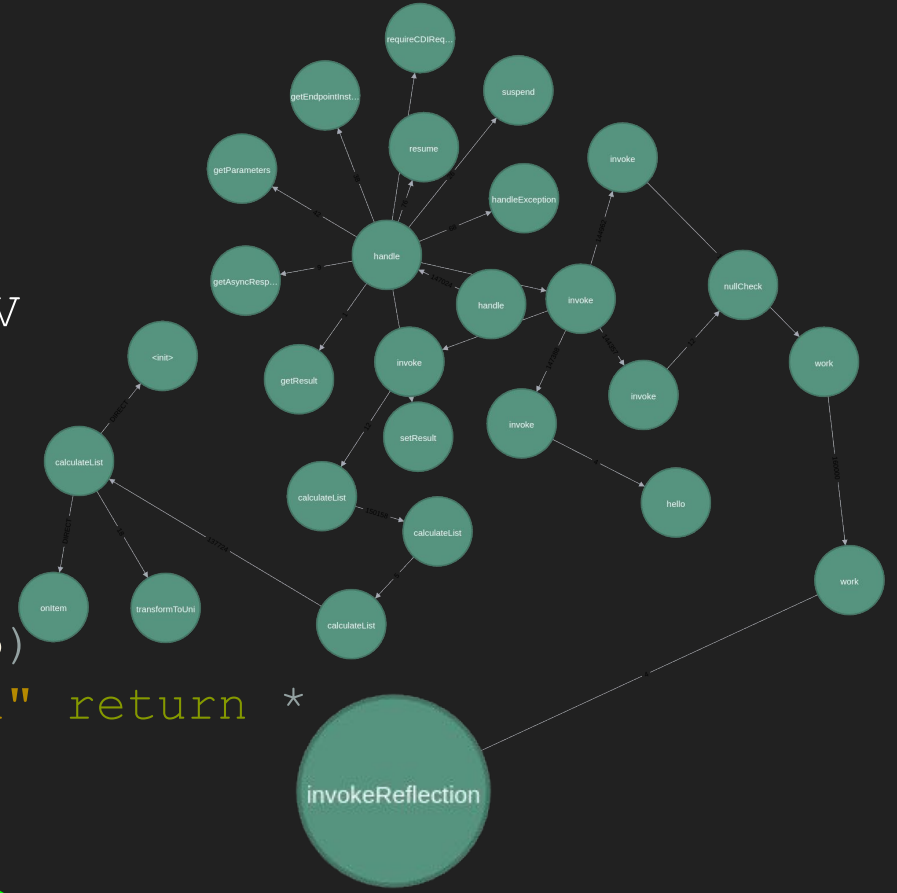
```
2024-03-24 19:35:57,510 ERROR
[io.qua.res.rea.jac.run.map.NativeInvalidDefinitionExceptionMapper]
(vert.x-eventloop-thread-3) Jackson was unable to serialize type
'org.acme.nativeproblem.BlackBox'. Consider annotating the class with
'@RegisterForReflection' or using
'org.jboss.resteasy.reactive.RestResponse' as a response type of
'org.acme.JpoinController#work:
com.fasterxml.jackson.databind.exc.InvalidDefinitionException: No
serializer found for class org.acme.nativeproblem.BlackBox and no
properties discovered to create BeanSerializer (to avoid exception,
disable SerializationFeature.FAIL_ON_EMPTY_BEANS). This appears to be
a native image, in which case you may need to configure reflection for
the class that is to be serialized
```


Reflection

Собираем с
-H:+PrintAnalysisCallTree,
-H:PrintAnalysisCallTreeType=CSV

кладем в neo4j

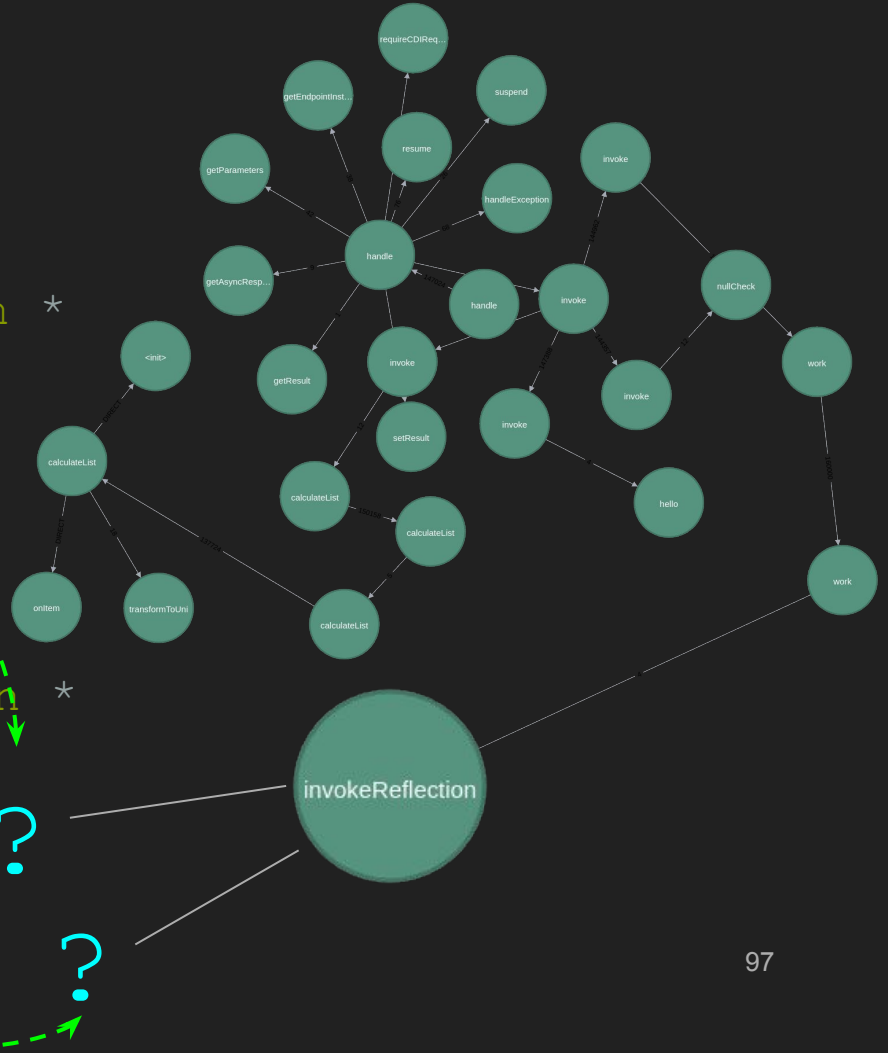
```
match (m:Method) <- [*1..5] - (o)
where m.name = "invokeReflection" return *
```



Reflection

```
match (m:Method) <- [*1..6] - (o)
where m.name = "setBoxName" return *
```

```
match (m:Method) <- [*1..6] - (o)
where m.name = "getBoxName" return *
```



Reflection. Нужно подсказать native image

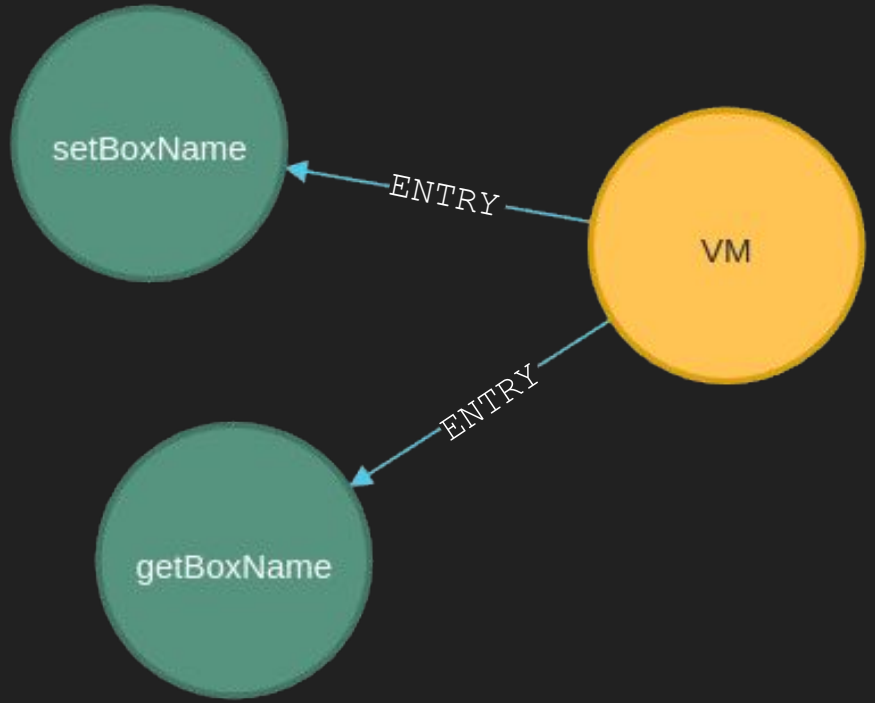
```
@RegisterForReflection
public class BlackBox
```

ИЛИ

```
[
 {
   "name" : "org.acme.nativeproblem.BlackBox",
   "allDeclaredConstructors" : true,
   "allPublicConstructors" : true,
   "allDeclaredMethods" : true,
   "allPublicMethods" : true,
   "allDeclaredFields" : true,
   "allPublicFields" : true
 }
]
```


Reflection

```
match (m:Method) <- [*1..2] - (o) where m.name = "setBoxName" or m.name = "getBoxName" return *
```



resource

Jansi — это небольшая Java-библиотека, которая позволяет использовать коды ANSI для форматирования вывода консоли, которая работает даже в Windows.

```
{
 "resources": [
   {"pattern": "org/fusesource/jansi/jansi.properties"},
   {"pattern": "org/fusesource/jansi/jansi.txt"},
   {"pattern": "org/fusesource/jansi/internal/native/.*"}
 ]
}
```


jni

```
public static class SMALL_RECT {
    public SMALL_RECT() {
    }
    private static native void init();
    public short width() {
        return (short)(this.right - this.left);
    }
    public short height() {
        return (short)(this.bottom - this.top);
    }
    static {
        JansiLoader.initialize();
        init();
    }
}
```


jni

```
[
  {
   "name" : "org.fusesource.jansi.internal.Kernel32$SMALL_RECT",
   "allDeclaredConstructors" : true,
   "allPublicConstructors" : true,
   "allDeclaredMethods" : true,
   "allPublicMethods" : true,
   "methods" : [
     { "name" : "height", "parameterTypes" : [] },
     { "name" : "width", "parameterTypes" : [] }
   ]
 }
]
```


Dynamic Proxy

```
public interface SimpleService {
    String simpleMethod();
}

public class SimpleServiceImpl implements SimpleService
{

    @Override
    public String simpleMethod() {
        return "simpleString";
    }
}
```


Dynamic Proxy

```
SimpleService simpleService = new SimpleServiceImpl();
SimpleService loggingSimpleServiceProxy =
        (SimpleService) java.lang.reflect.Proxy.newProxyInstance(
                ProxyProblemService.class.getClassLoader(),
                classes, (proxy, method, args) -> {
                    System.out.println("start");
                    Object object = method.invoke(simpleService, args);
                    System.out.println("finish");
                    return object;
                });
return loggingSimpleServiceProxy.simpleMethod();
```


Dynamic Proxy

```
2024-03-26 19:46:41,468 ERROR
[io.qua.ver.htt.run.QuarkusErrorHandler] (vert.x-eventloop-thread-4)
HTTP Request to /jpoint/work/andrey failed, error id:
a4d51c9e-cf81-416f-b68f-d15b3f8f2a5a-135:
org.graalvm.nativeimage.MissingReflectionRegistrationError: The
program tried to reflectively access the proxy class inheriting
[org.acme.nativeproblem.ProxyProblemService$SimpleService] without it
being registered for runtime reflection. Add
[org.acme.nativeproblem.ProxyProblemService$SimpleService]to the
dynamic-proxy metadata to solve this problem.Note: The order of
interfaces used to create proxies matters. See
https://www.graalvm.org/latest/reference-manual/native-image/metadata
/#dynamic-proxy for help.
```


Native все проблемы с json можно обойти с помощью java агента

```
exec /graalvm-ce-java11-22.2.0/bin/java $JAVA_OPTS
-agentlib:native-image-agent=config-output-dir=/app/native-image-
generated-configs,config-write-period-secs=10 -jar /app/*.jar $@
```


Native profiling

Perf:

```
Map<String, String> map = new TreeMap<>();
int count = 0;
do {
   map.put(name + count, name + count);
   count = count + 1;
} while (true);
```

```
-Dquarkus.native.additional-build-args=
-H:+PreserveFramePointer,-H:-DeleteLocalSymbols
```

```
perf script | sed -E "s/thread-[0-9]*/thread/" |
${FG_HOME}/stackcollapse-perf.pl |
${FG_HOME}/flamegraph.pl > flamegraph.svg
```


Perf в принципе можно найти проблемы

Native profiling

Jfr:

Поддержка JFR
в настоящее время неполная

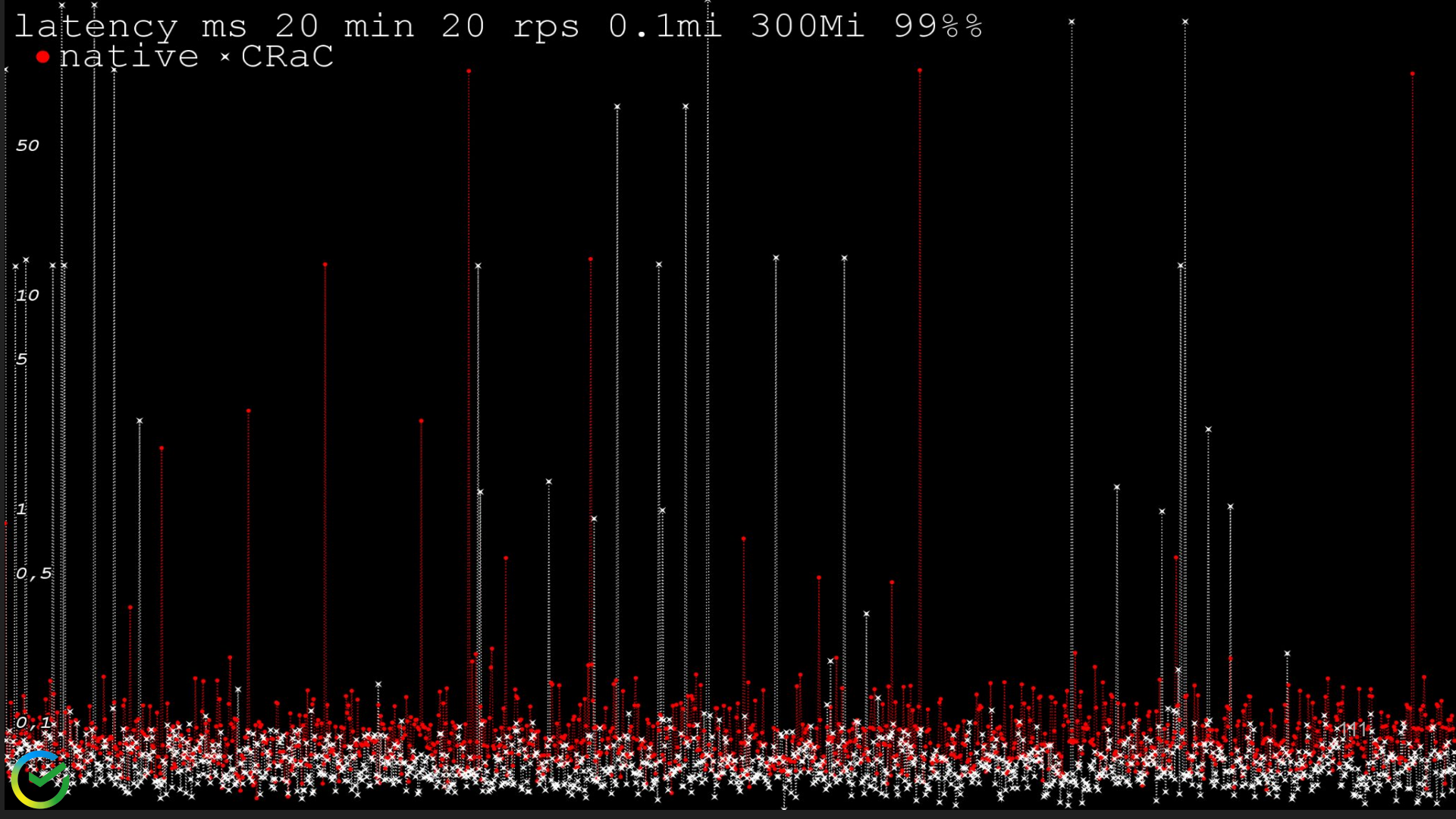
BUILD -H:+AllowVMInspection

https://github.com/oracle/graal/issues/5410

RUN -XX:+FlightRecorder
-XX:StartFlightRecording="filename=recording.jfr"

Как итог - мы пришли к успеху

- Удалось уменьшить потребление ресурсов
  ~1500*(0.8-0.1) = ~ 1000 ядер -
  что стоит как чугунный мост
- Увеличили отказоустойчивость
  и масштабируемость
- В итоге получили “идеальный” сервис
  на 0.1 cpu и стартующий за полсекунды
  на максимум производительности


latency ms 20 min 20 rps 0.1mi 300Mi 99%%
native
CRaC
50
10
5
1
0,5
0,1