ТИНЬКОФФ

Сколько памяти нужно для сборки?

Андроид разработчикам о JVM

Юрий Анисимов

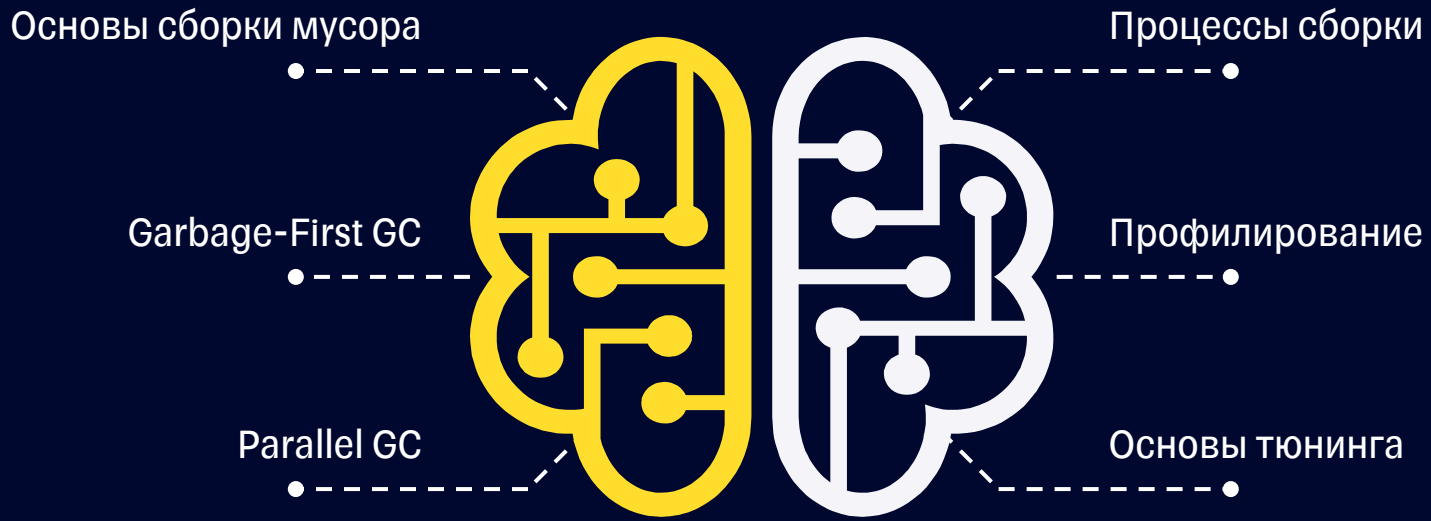
Сборка замедляется постепенно

Замедлить сборку может, как слишком малое, так и излишне большое выделение памяти.

Использование оперативной памяти

- Как оценить количество выделенной памяти?
- Поможет ли увеличение памяти ускорить сборку?
- Как определить приемлемое количество памяти?
- Сколько памяти потребляет Java процесс?

Что будет в докладе



В докладе не будет советов в стиле «делайте так...»

«No Silver Bullet»

Frederick P. Brooks, Jr.

Юрий Анисимов

Разработчик Инфра.

y.anisimov@tinkoff.ru

Мобильный банк

Случайное приложение

- ~95k строк кода
- 100+ модулей
- полностью на Котлин
- Hilt, Ktor, Compose

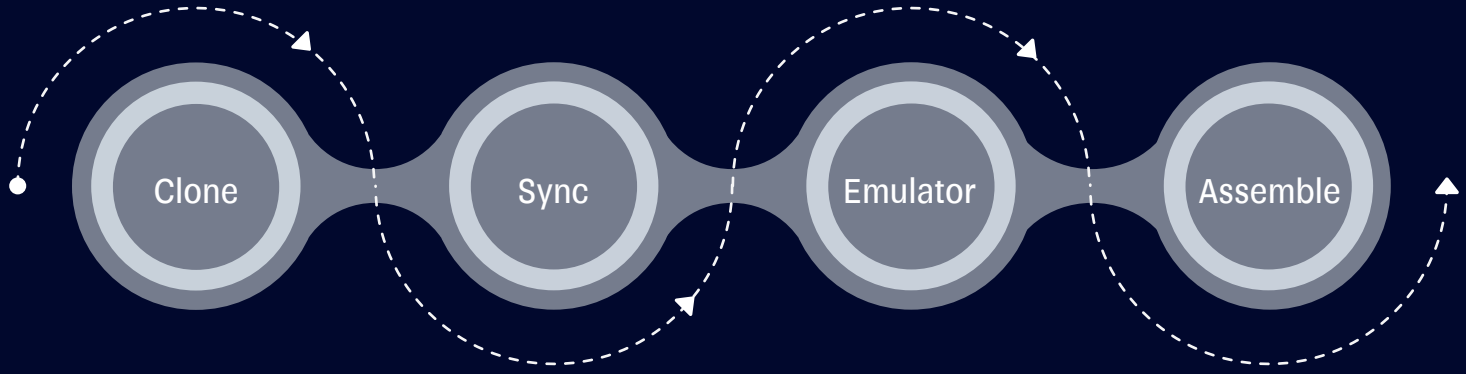
Настройки Java Virtual Machine (JVM)

gradle.properties

```
org.gradle.jvmargs=-Xmx6g -Dfile.encoding=UTF-8
…
```

Heap space — область для динамического
выделения памяти под Java объекты
и классы JRE.

Первый запуск приложения новым разработчиком



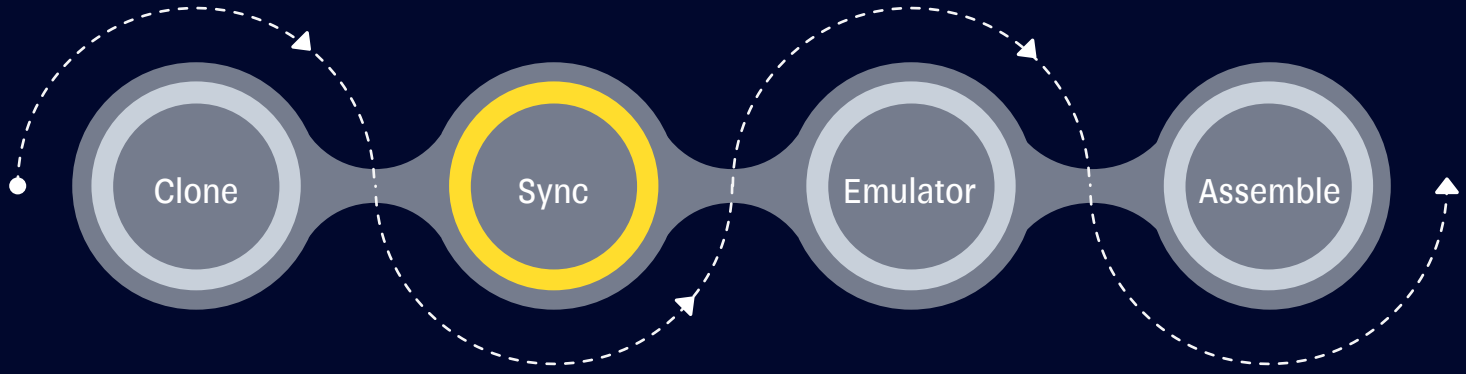
Клон проекта

Синк с IDE

Эмулятор

Сборка и запуск

Первый запуск приложения новым разработчиком

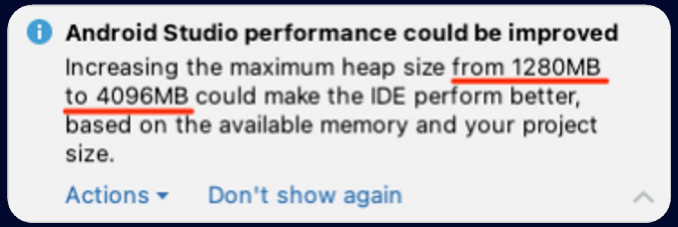


Клон проекта

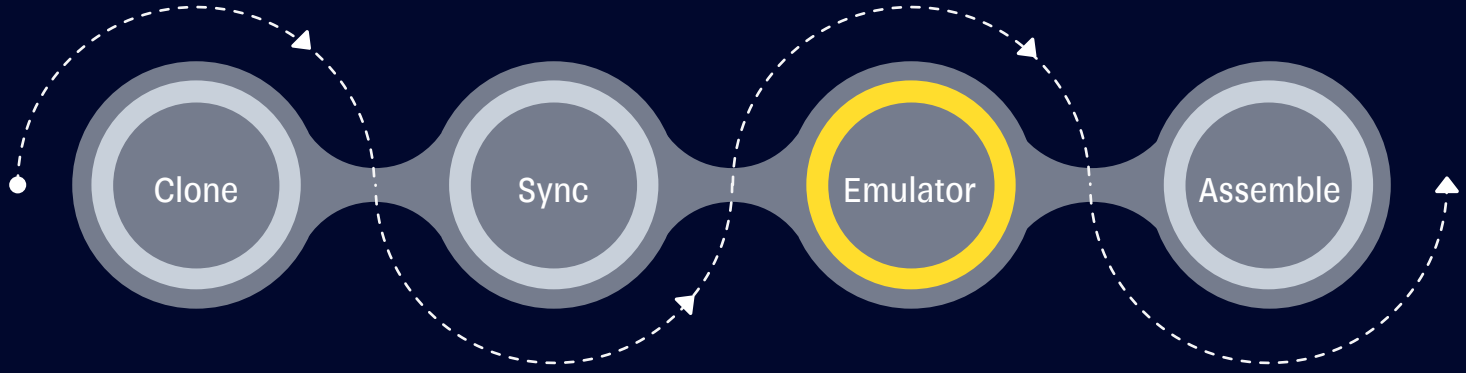
Синк с IDE

Эмулятор

Сборка и запуск



Первый запуск приложения новым разработчиком



Клон проекта

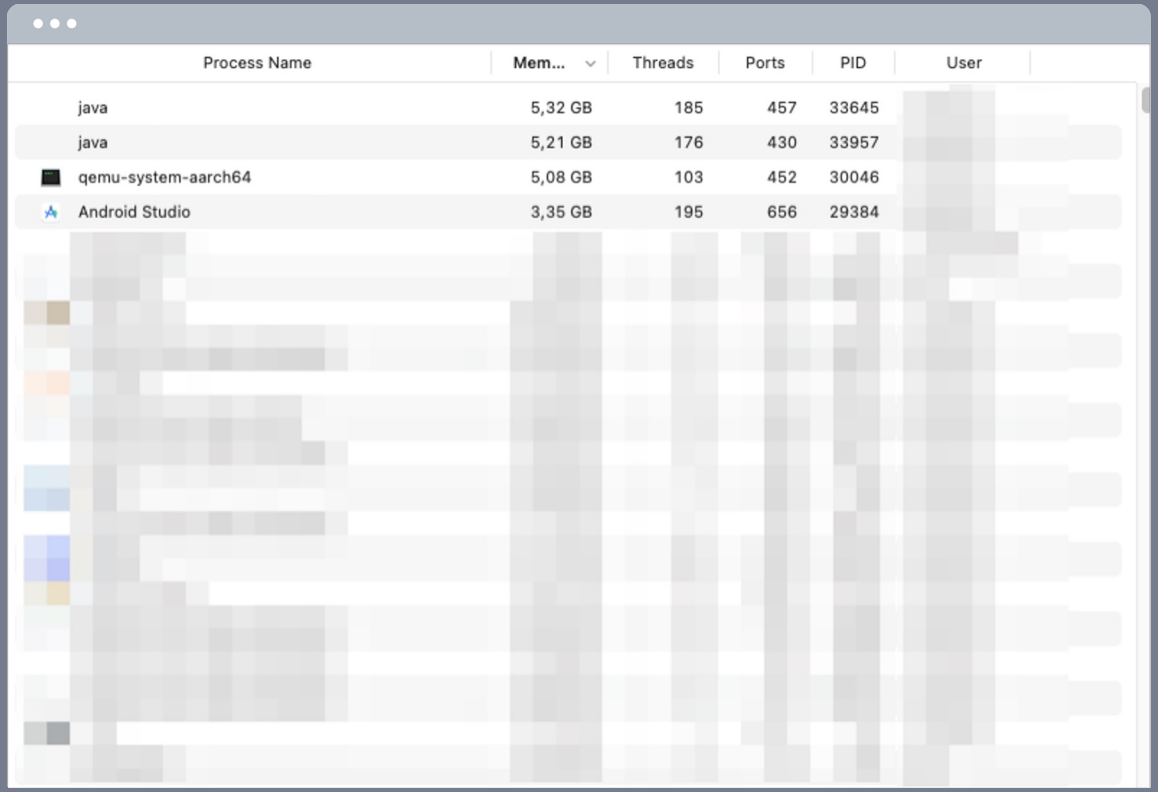
Синк с IDE

Эмулятор

- Pixel 4
- Android 13
- Настройки по умолчанию

Сборка и запуск

ТОП процессов (по памяти)



Process Name	Mem...	Threads	Ports	PID	User
java	5,32 GB	185	457	33645	
java	5,21 GB	176	430	33957	
qemu-system-aarch64	5,08 GB	103	452	30046	
Android Studio	3,35 GB	195	656	29384	

ТОП процессов (по памяти)

Результаты

- Откуда взялся второй Java процесс?
- Нужно ли такое количество RAM?
- Как быть, если доступно всего 16 Гб?

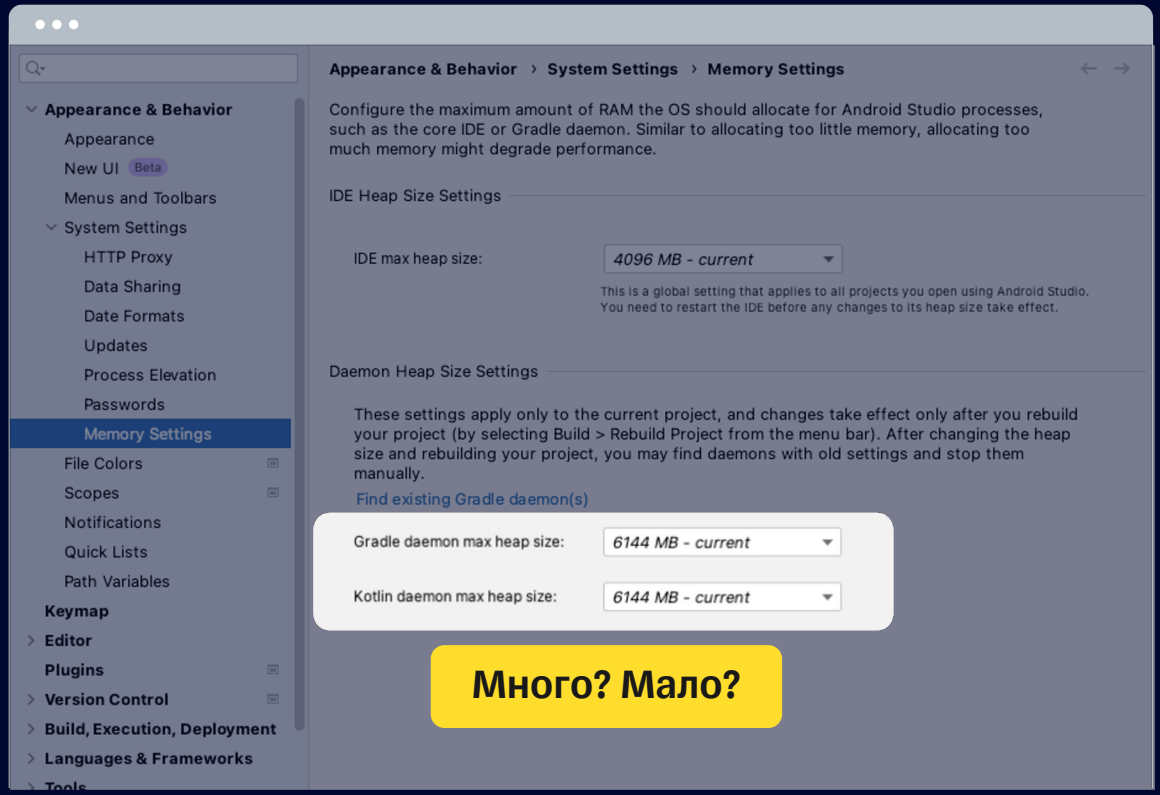
Список Java процессов

```
% jps
```

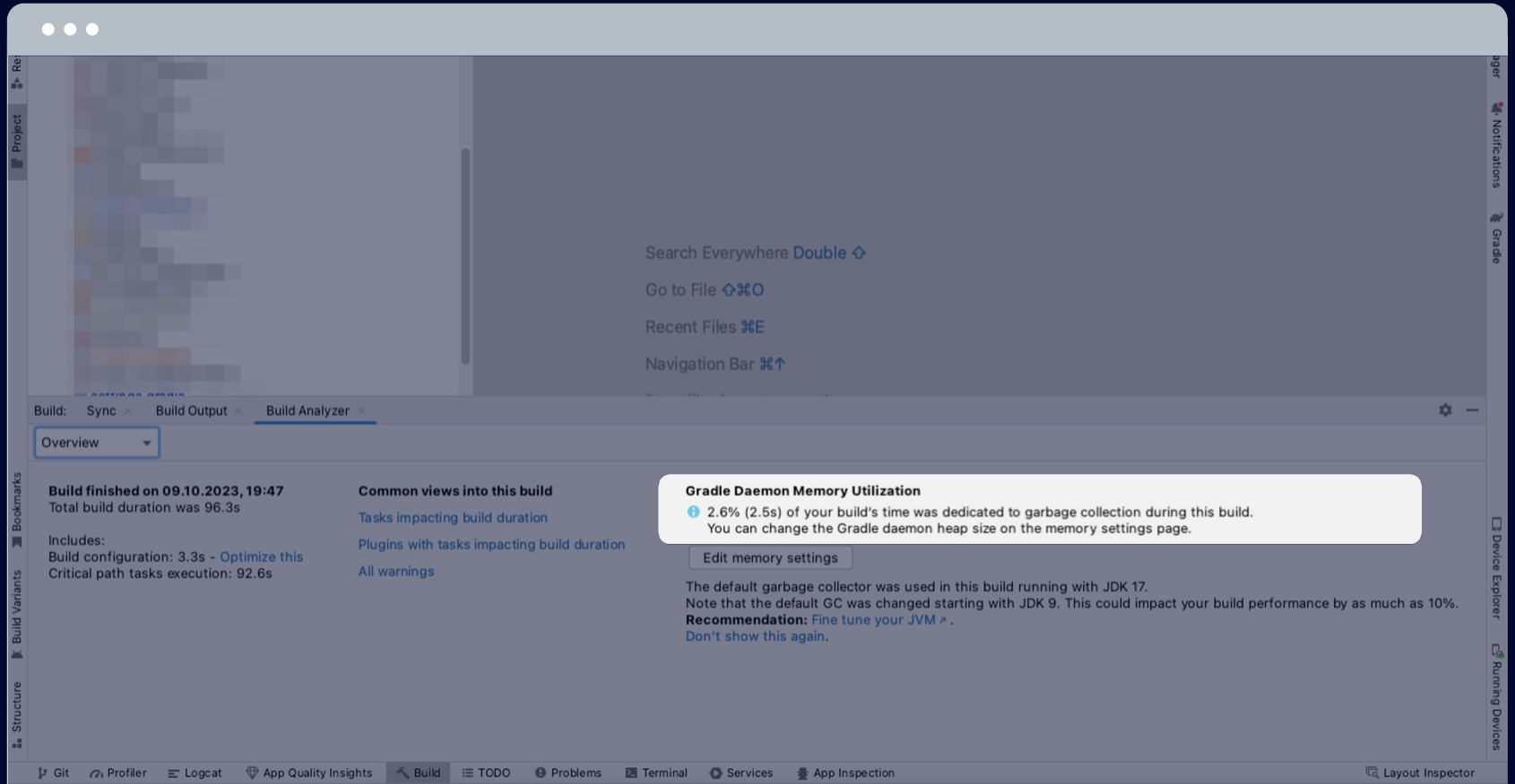
Список Java процессов

```
% jps
29384
39779 GradleDaemon
40011 KotlinCompileDaemon
40056 Jps
```

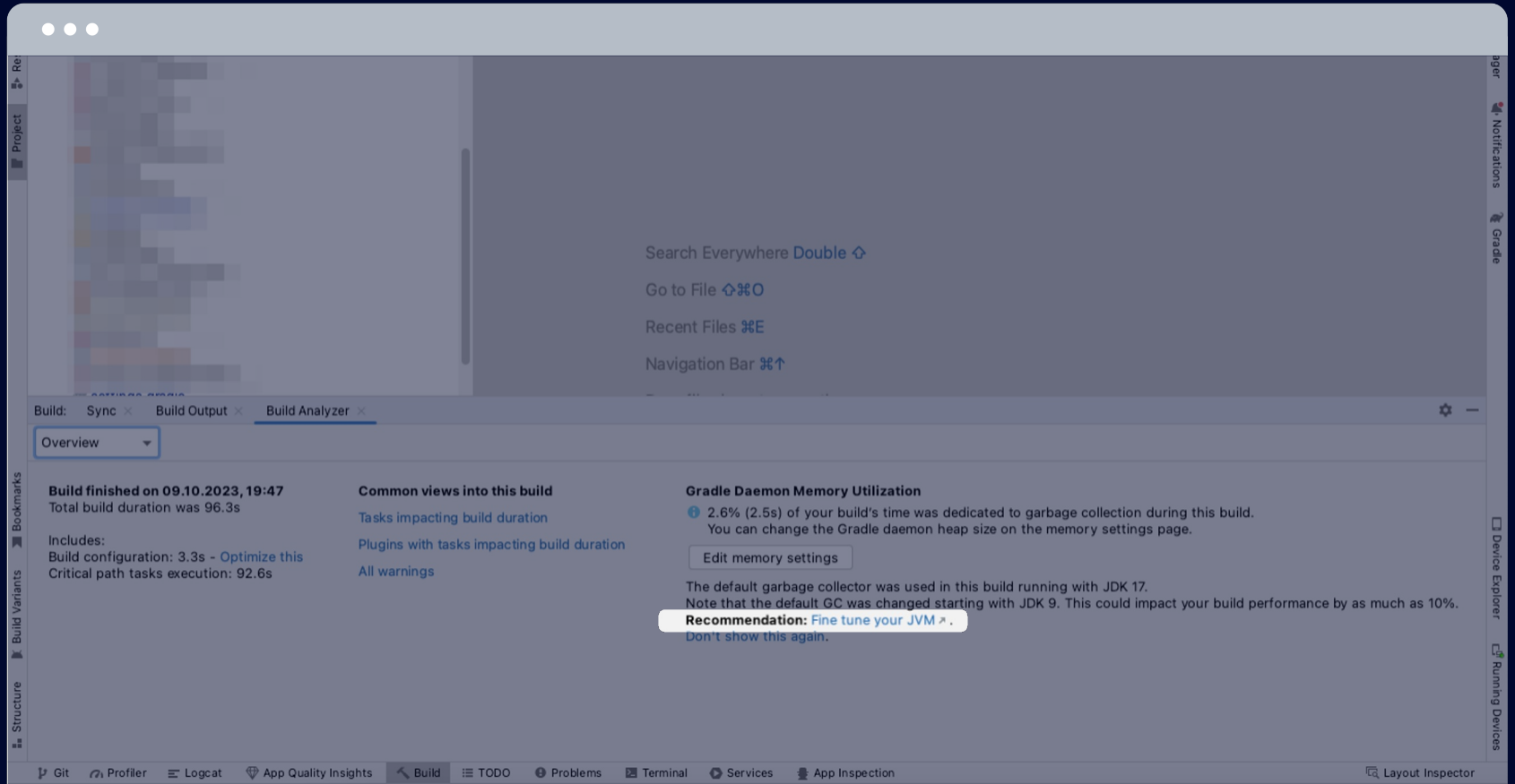
Настройки памяти



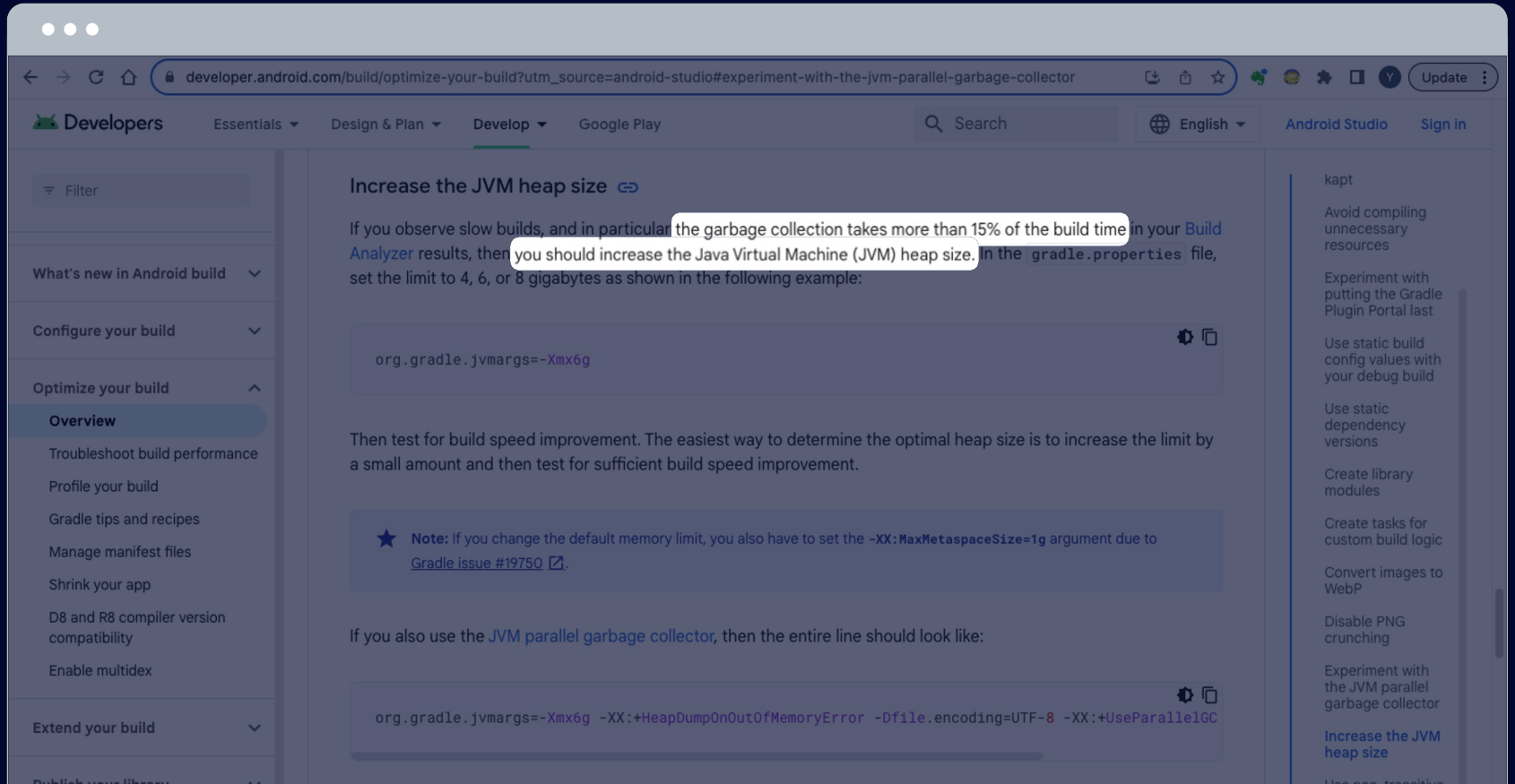
Build Performance Analyzer



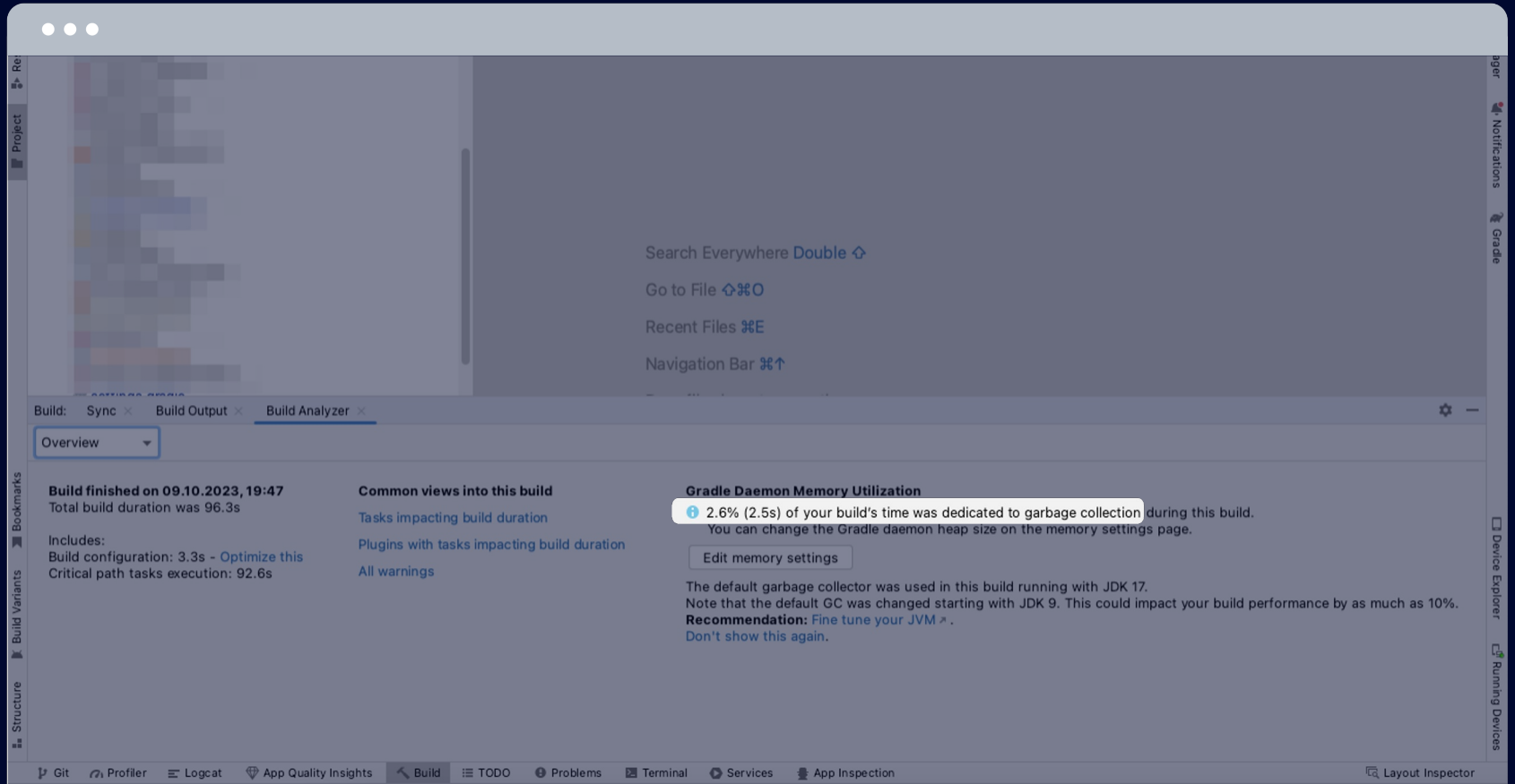
Build Performance Analyzer



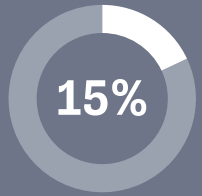
Тонкая настройка JVM



Build Performance Analyzer

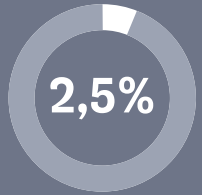


До какого размера уменьшать?



Сколько будет нормально?

Судя по документации, 15% это — крайне плохо.



Откуда взялось 2,5%?

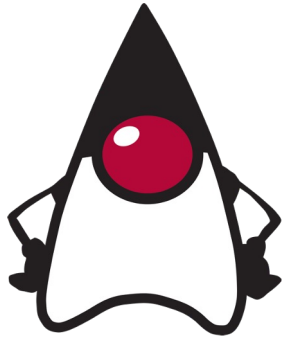
При том, что потребление памяти не достигло лимита.

Java Platform, Standard Edition

HotSpot Virtual Machine Garbage Collection Tuning Guide

Release 17
F41050-02
April 2022

Release 17



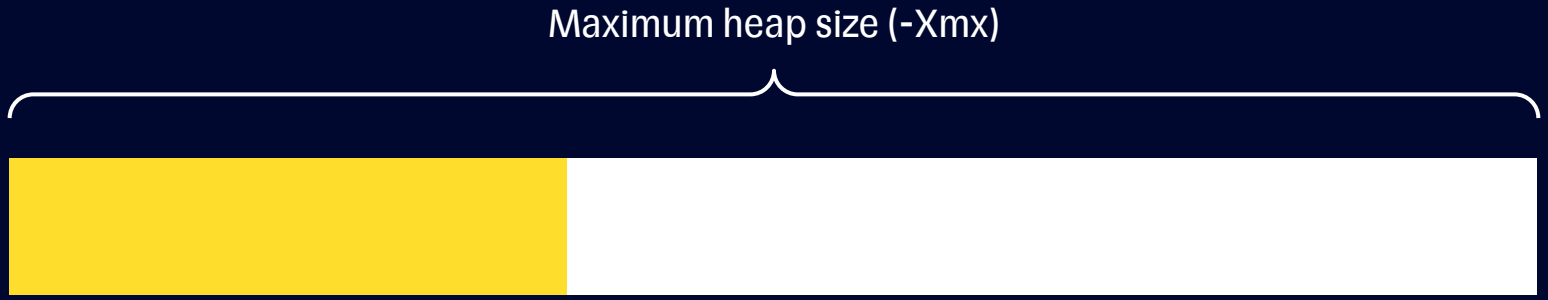
Управление памятью

- Поиск «мертвых» объектов
- Освобождение неиспользуемых блоков

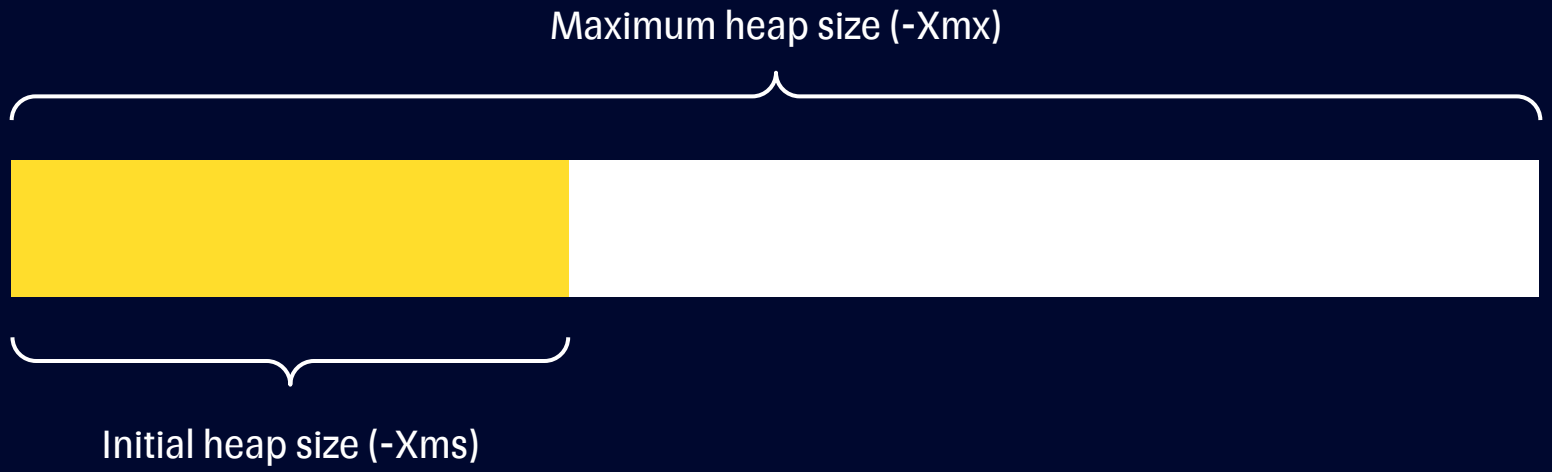
Управление памятью

- Поиск «мертвых» объектов
- Освобождение неиспользуемых блоков
- **Получение памяти из ОС**
- **Выделение памяти приложению**

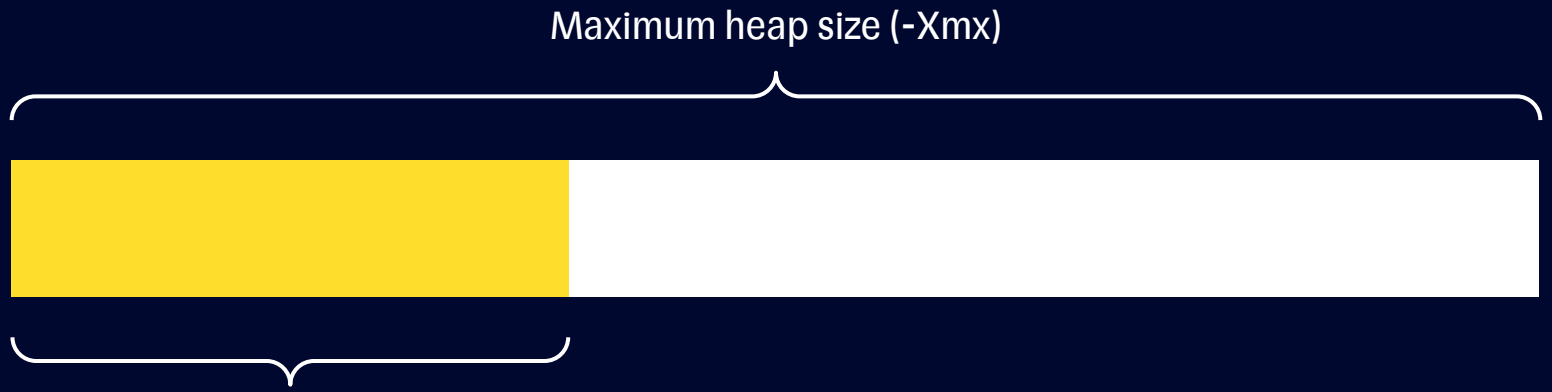
Размер Java Heap



Размер Java Heap

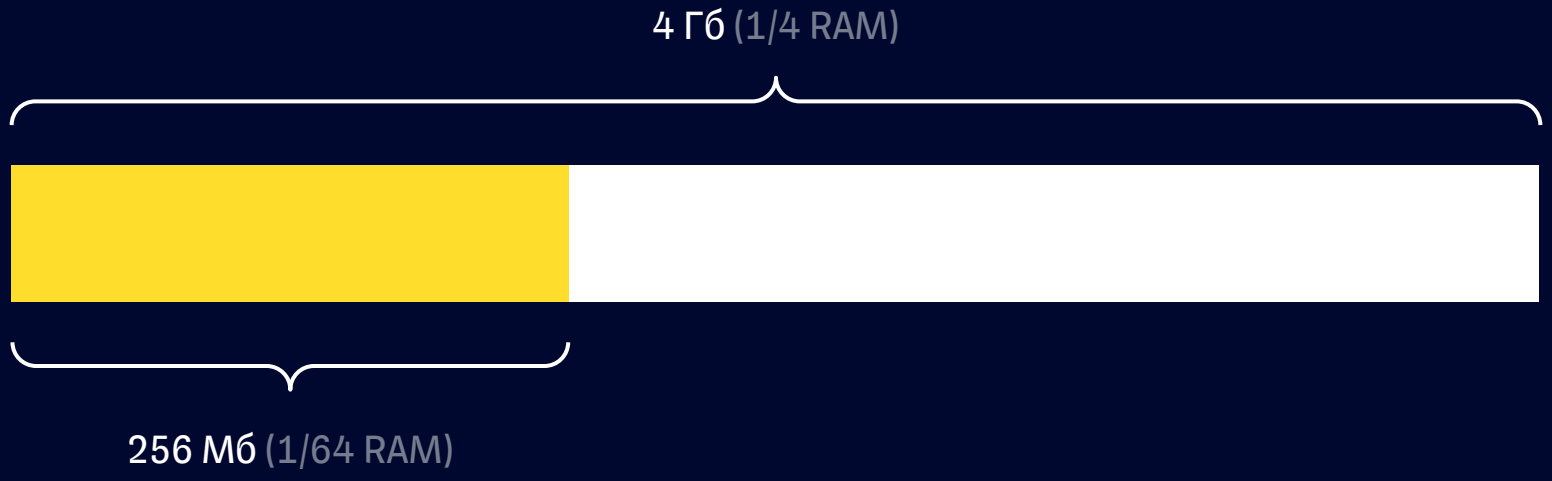


Начальный размер для 16 Гб RAM?



- 256 Мб
- 512 Мб

Начальный размер для 16 Гб RAM?



Эргономика

Процесс улучшения перфоманса приложения за счет эвристик JVM и GC. Например основанных на поведении приложения

Эргономика

Процесс улучшения перфоманса приложения за счет эвристик JVM и GC. Например основанных на поведении приложения

Throughput Goal

Время GC относительно всего времени работы

-XX:GCTimeRatio=<nnn>

Эргономика

Процесс улучшения перфоманса приложения за счет эвристик JVM и GC. Например основанных на поведении приложения

Throughput Goal

Время GC относительно всего времени работы

-XX:GCTimeRatio=<nnn>

Maximum Pause-Time Goal

Максимальное время остановки приложения

-XX:MaxGCPauseMillis=<nnn>

Эргономика

Процесс улучшения перфоманса приложения за счет эвристик JVM и GC. Например основанных на поведении приложения

Throughput Goal

Время GC относительно всего времени работы

-XX:GCTimeRatio=<nnn>

Maximum Pause-Time Goal

Максимальное время остановки приложения

-XX:MaxGCPauseMillis=<nnn>

Footprint (heap size)

Занимаемая область памяти

-Xms, -Xmx

Эргономика

Процесс улучшения перфоманса приложения за счет эвристик JVM и GC. Например основанных на поведении приложения

Throughput Goal

-XX:GCTimeRatio=<nnn>

Maximum Pause-Time Goal

-XX:MaxGCPauseMillis=<nnn>

Footprint (heap size)

-Xms, -Xmx

Варианты реализации GC

Serial Collector

Parallel Collector

Garbage-First

The Z GC

Варианты реализации GC




Serial Collector

-XX:+UseSerialGC

Для однопроцессорных машин с малым потреблением памяти (100 Мб)

Parallel Collector

Garbage-First

The Z GC

Варианты реализации GC

Serial Collector

-XX:+UseSerialGC

Для однопроцессорных машин с малым потреблением памяти (100 Мб)

Parallel Collector

-XX:+UseParallelGC

Высокая производительность. Длинные паузы для очистки мусора.

Garbage-First

The Z GC

Варианты реализации GC

Serial Collector

-XX:+UseSerialGC

Для однопроцессорных машин с малым потреблением памяти (100 Мб)

Parallel Collector

-XX:+UseParallelGC

Высокая производительность. Длинные паузы для очистки мусора.

Garbage-First

-XX:+UseG1GC

Баланс между короткими паузами и высокой производительностью.

The Z GC

Варианты реализации GC

Serial Collector

-XX:+UseSerialGC

Для однопроцессорных машин с малым потреблением памяти (100 Мб)

Parallel Collector

-XX:+UseParallelGC

Высокая производительность. Длинные паузы для очистки мусора.

Garbage-First

-XX:+UseG1GC

Баланс между короткими паузами и высокой производительностью.

The Z GC

-XX:+UseZGC

Минимальные паузы для любого размера хипа за счет потери в производительности.

Варианты реализации GC

Serial Collector

-XX:+UseSerialGC

Для однопроцессорных машин с малым потреблением памяти (100 Мб)

Parallel Collector

-XX:+UseParallelGC

Высокая производительность. Длинные паузы для очистки мусора.

Garbage-First

-XX:+UseG1GC

Баланс между короткими паузами и высокой производительностью.

The Z GC

-XX:+UseZGC

Минимальные паузы для любого размера хипа за счет потери в производительности.

Выбор по умолчанию в Java 17?

Serial Collector

Parallel Collector

Garbage-First

The Z GC

```
org.gradle.jvmargs=-Xmx6g -Dfile.encoding=UTF-8
…
```

Выбор по умолчанию в Java 17?




Serial Collector

Parallel Collector

Основной до Java 8

Garbage-First

Начиная с Java 9

The Z GC

Релиз в Java 15

```
org.gradle.jvmargs=-Xmx6g -Dfile.encoding=UTF-8
…
```

Варианты реализации GC

Serial Collector

Parallel Collector

Garbage-First

The Z GC

Generational Garbage Collection

Варианты реализации GC

Serial Collector

Parallel Collector

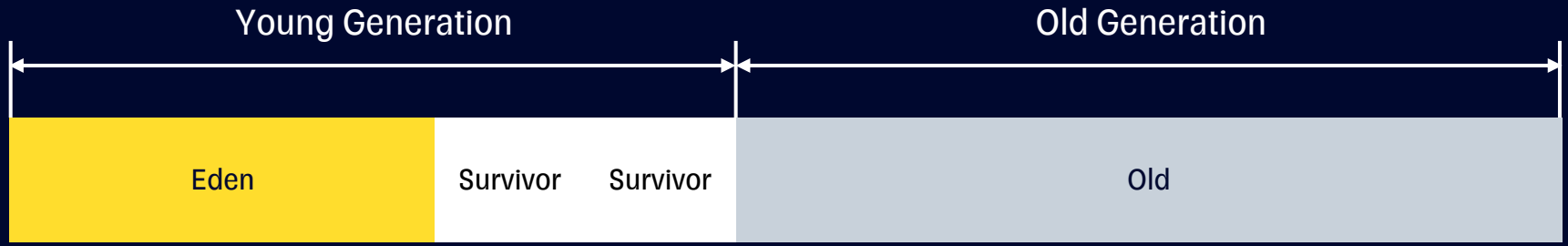
Garbage-First

The Z GC

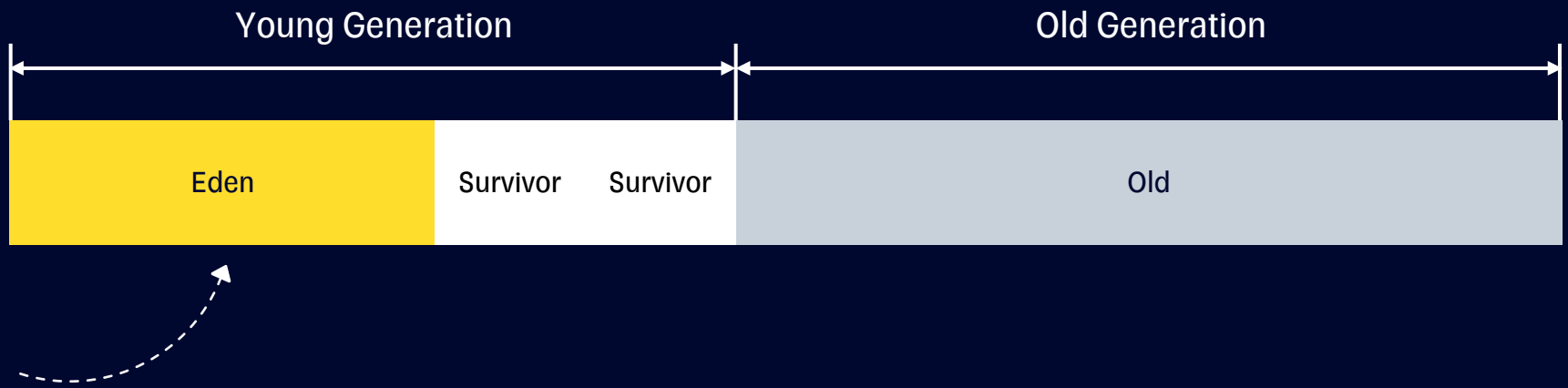
Generational Garbage Collection

Большинство объектов «умирают» вскоре после создания.

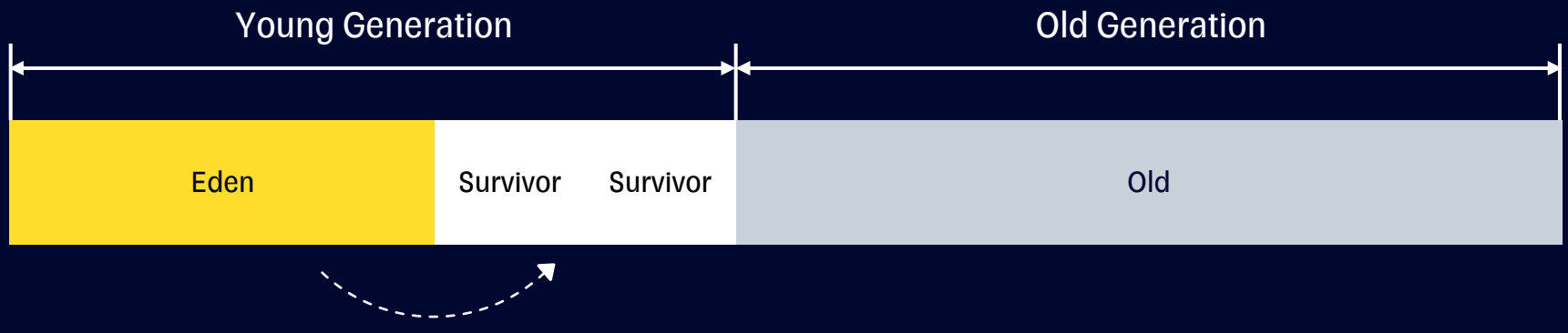
Generational collection



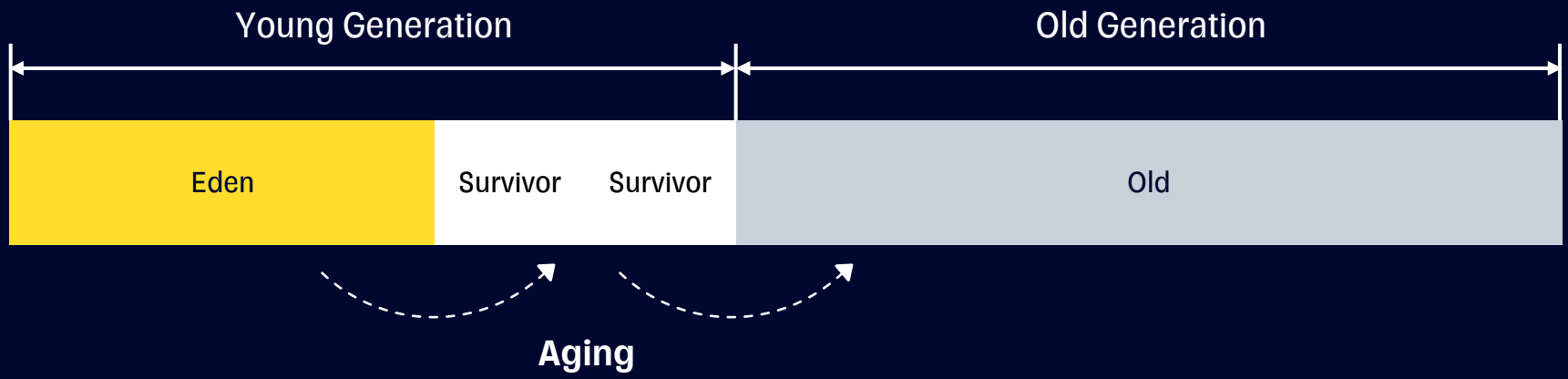
Generational collection



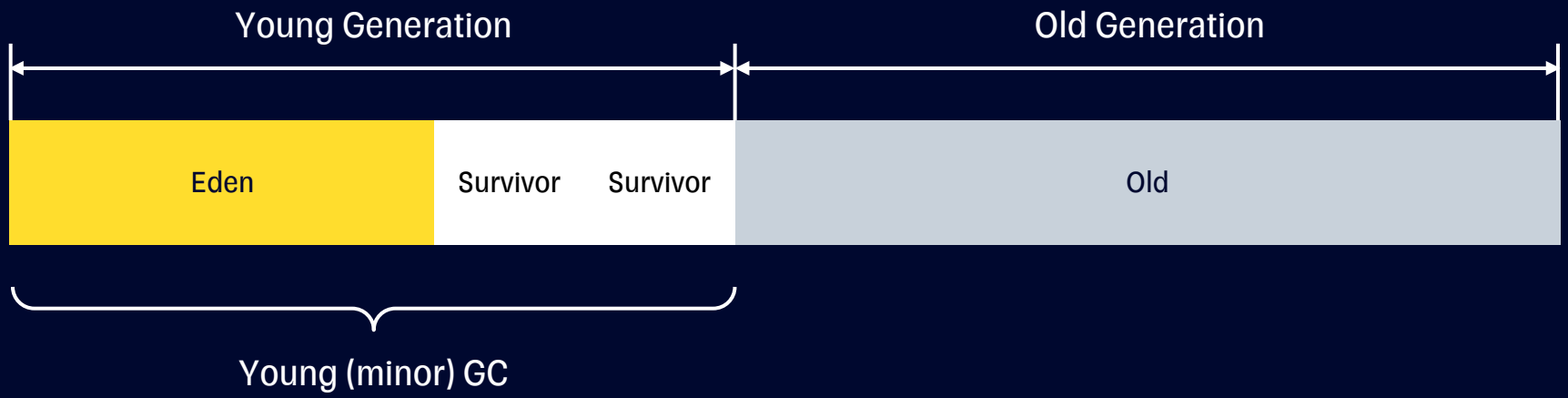
Generational collection



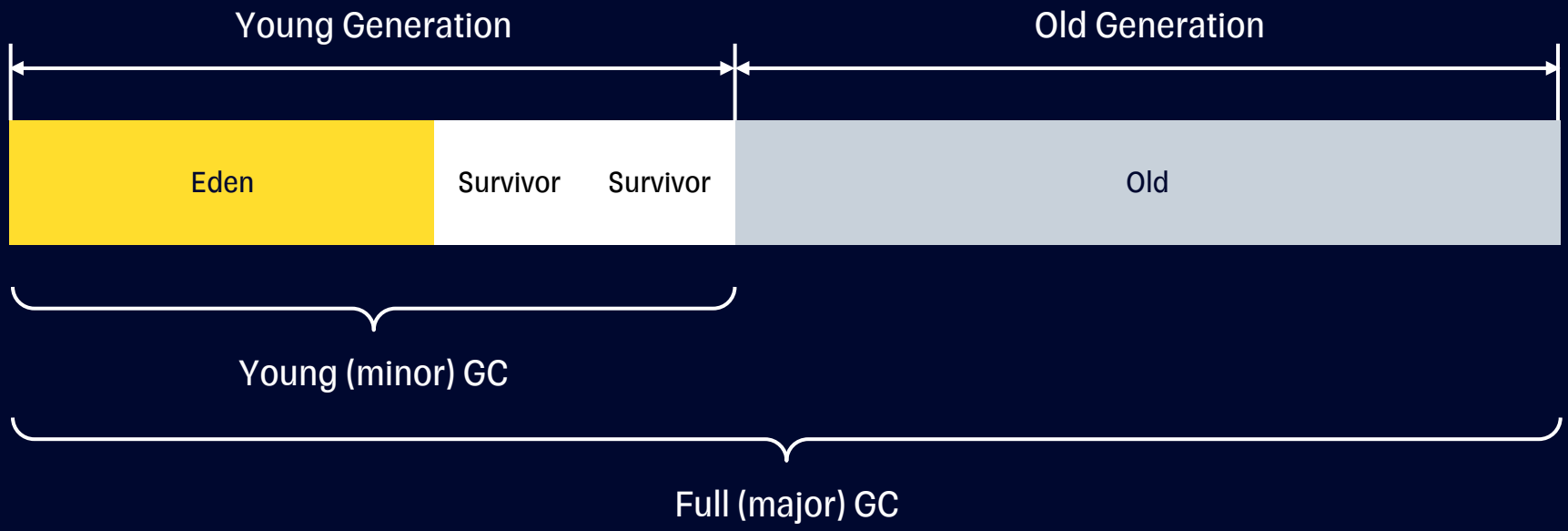
Generational collection



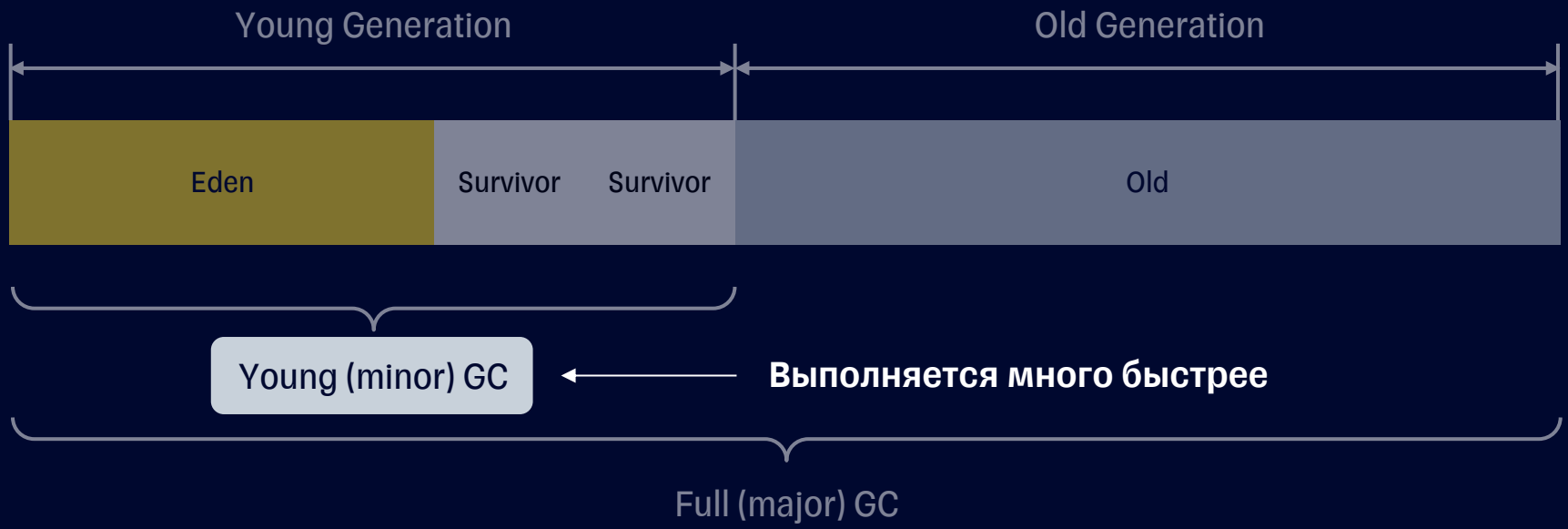
Generational collection



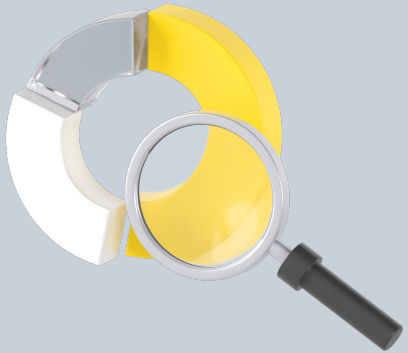
Generational collection



Generational collection



До какого размера уменьшать?



- Откуда взялись 2,5%?
- Сколько будет нормально?

Статистика Young generation

```
% jstat -gcnew <PID>
```

Опции jstat:

- -gcnew
- -gcold
- -gcutil
- …

Статистика Young generation

```
% jstat -gcnew <PID>
S0C   S1C      S0U   S1U       TT  MTT  DSS        EC          EU         YGC   YGCT
0.0   61440.0  0.0   58942.6   15  15   141312.0   2441216.0   262144.0   103   2.510
```

Опции jstat:

- -gcnew
- -gcold
- -gcutil
- …

Статистика Young generation

```
% jstat -gcnew <PID>
S0C   S1C      S0U   S1U       TT   MTT   DSS         EC          EU          YGC   YGCT
0.0   61440.0  0.0   58942.6   15   15    141312.0    2441216.0   262144.0    103   2.510
```

- Размер Eden space: 2+ Гб

Статистика Young generation

```
% jstat -gcnew <PID>
S0C    S1C      S0U    S1U      TT  MTT  DSS       EC          EU          YGC   YGCT
0.0    61440.0  0.0    58942.6  15  15   141312.0  2441216.0   262144.0    103   2.510
```

- Размер Eden space: 2+ Гб
- **Количество срабатываний Young GC: 100+**
- **Время работы Young GC: 2.50 сек.**

До какого размера уменьшать?

- Откуда взялись 2,5%?
- Сколько будет нормально?

Эргономика

Garbage-First GC

Maximum Pause-Time Goal

-XX:MaxGCPauseMillis=200

Throughput Goal

-XX:GCTimeRatio=12

Эргономика

Garbage-First GC

Maximum Pause-Time Goal

-XX:MaxGCPauseMillis=200

Throughput Goal

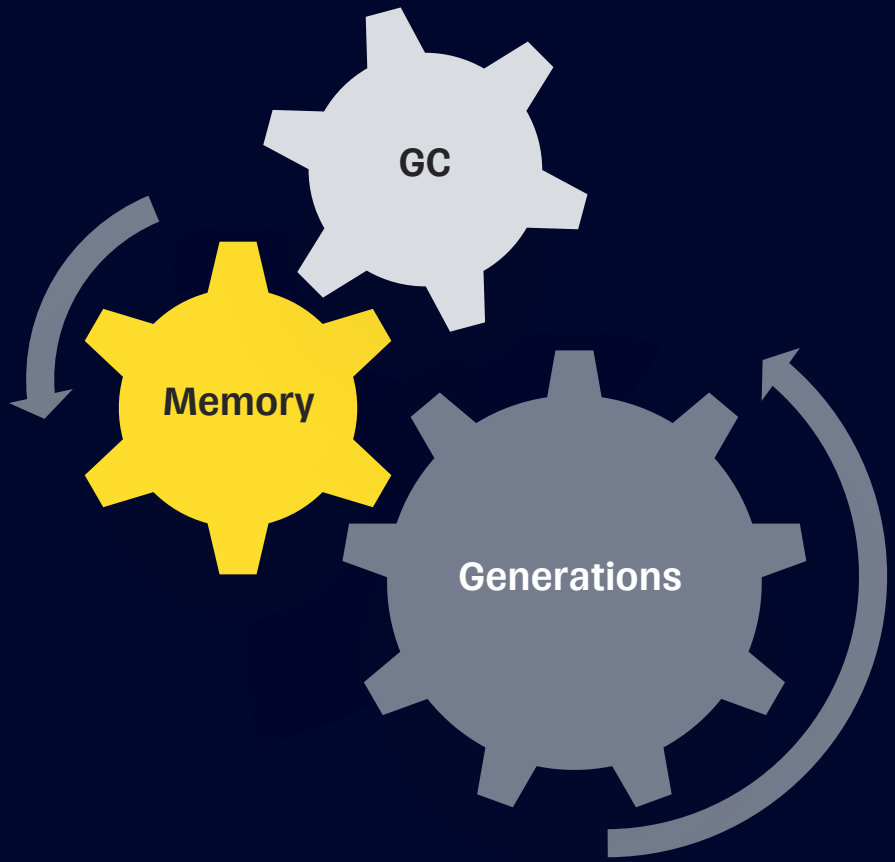
-XX:GCTimeRatio=12

$$\frac{1}{\text{GCTimeRatio} + 1} = \frac{1}{12 + 1} \sim 8\%$$

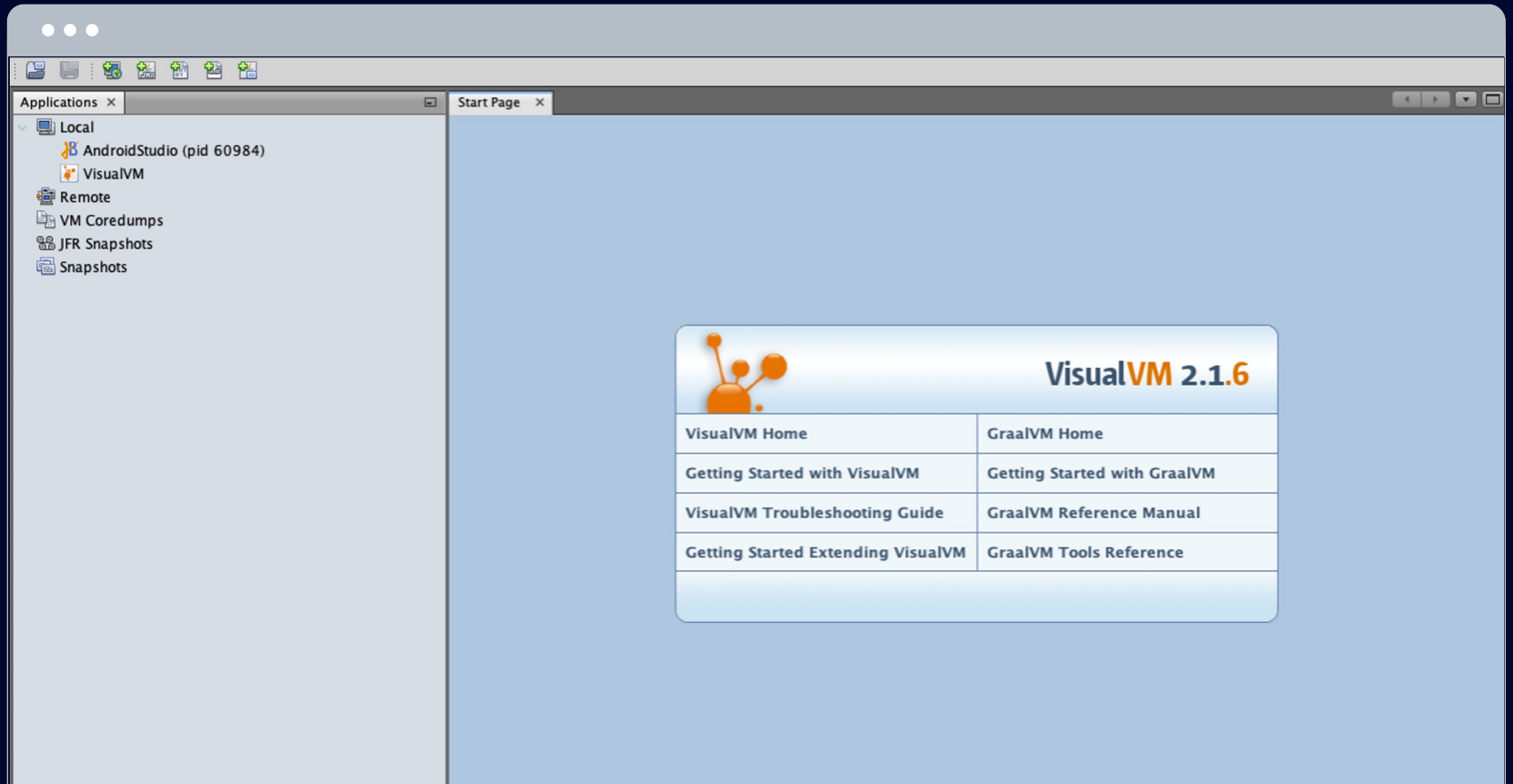
«Для Garbage-First приемлемое время сборки мусора будет в пределах 8% (из практики — 5%)»

Профилирование

- Активность GC
- Выделение
  и освобождение памяти
- Использование областей
  Java heap



VisualVM с плагином Visual GC

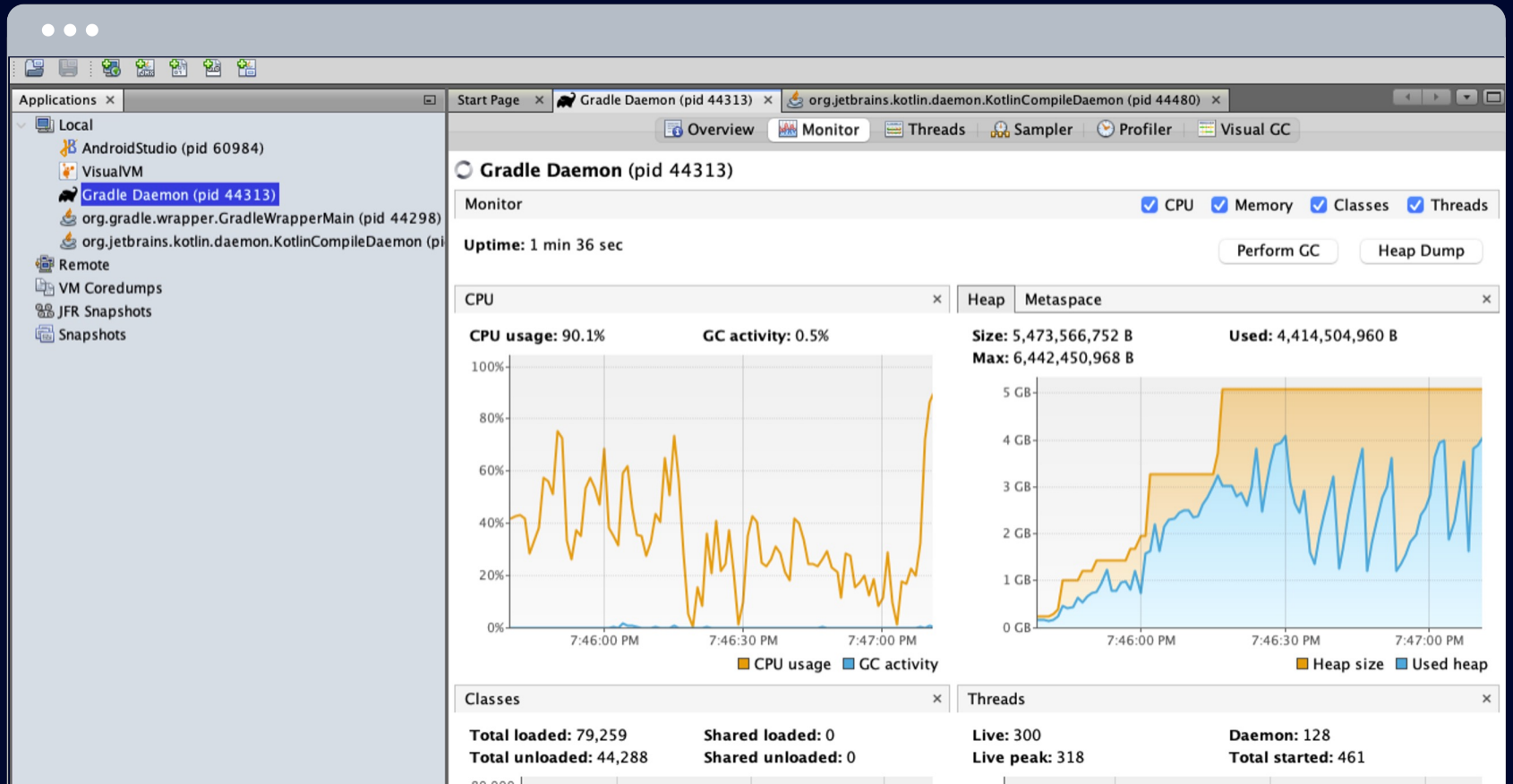


Сценарий профилирования

```
% ./gradlew --stop
% ./gradlew :app:assembleDebug --rerun-tasks --no-daemon
```

- Сборка отладочной версии
- Крайний случай с запуском всех тасок
- Показатели GC в рамках одного сеанса

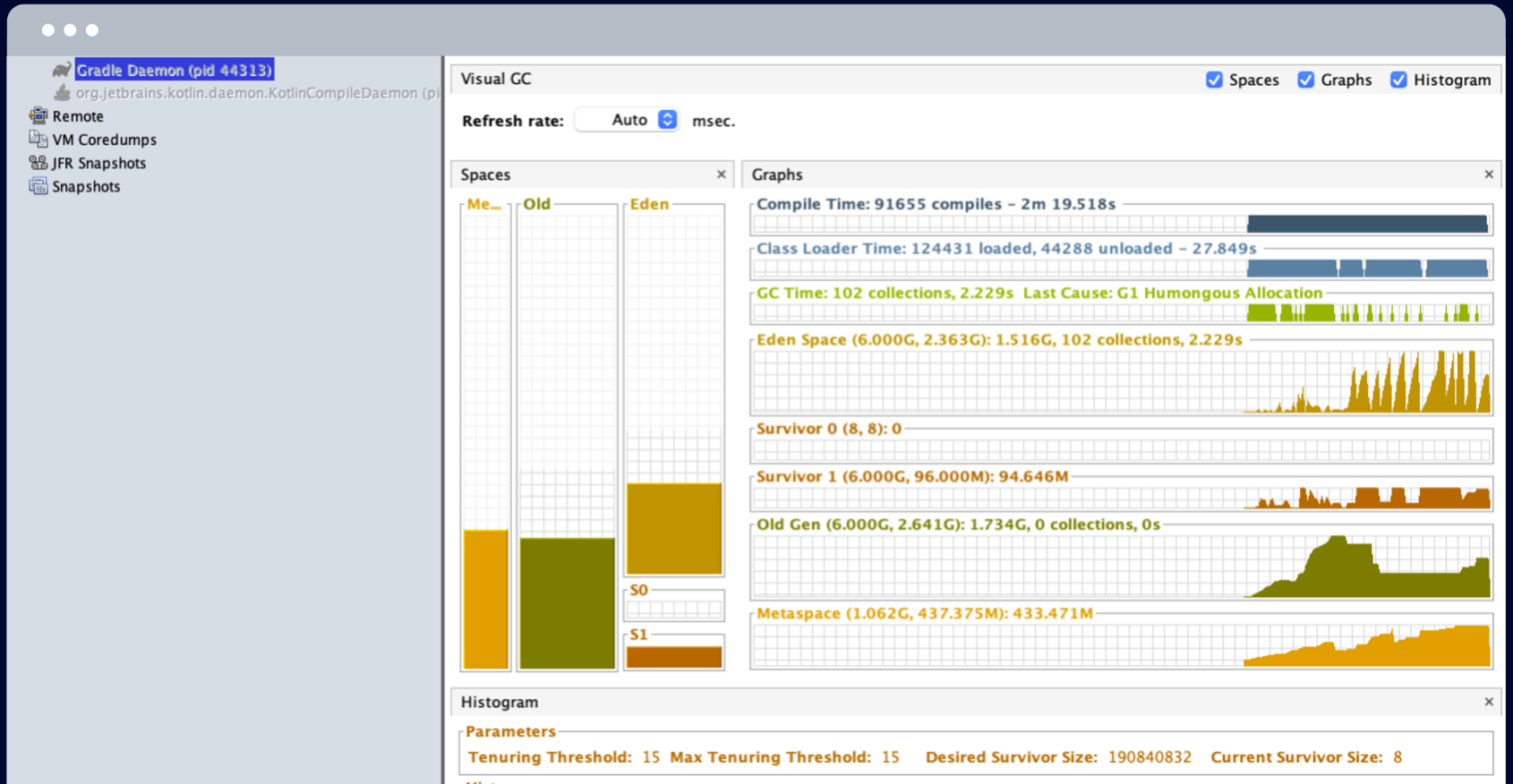
Gradle демон: Monitor



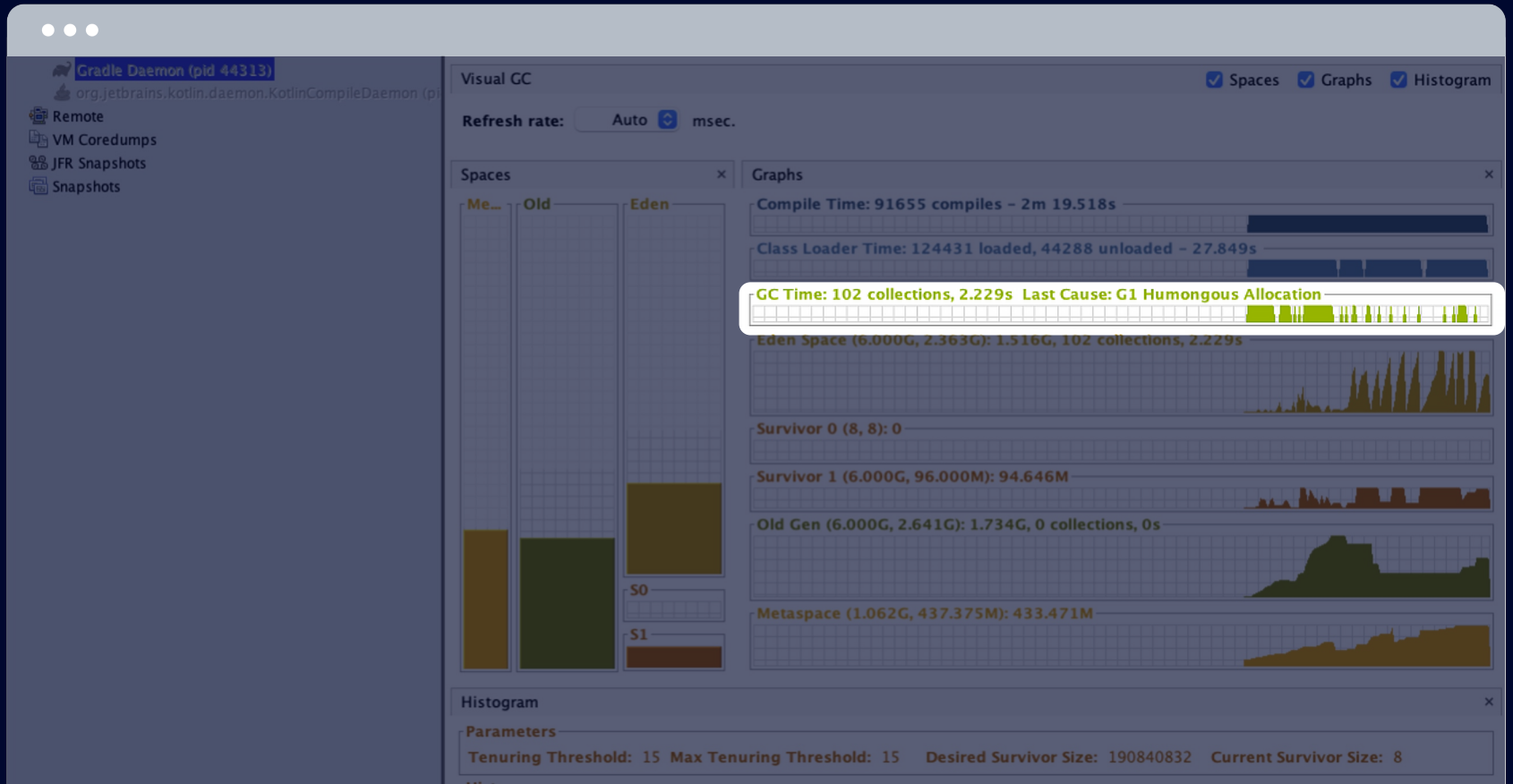
Gradle демон: Monitor

Gradle демон: Monitor

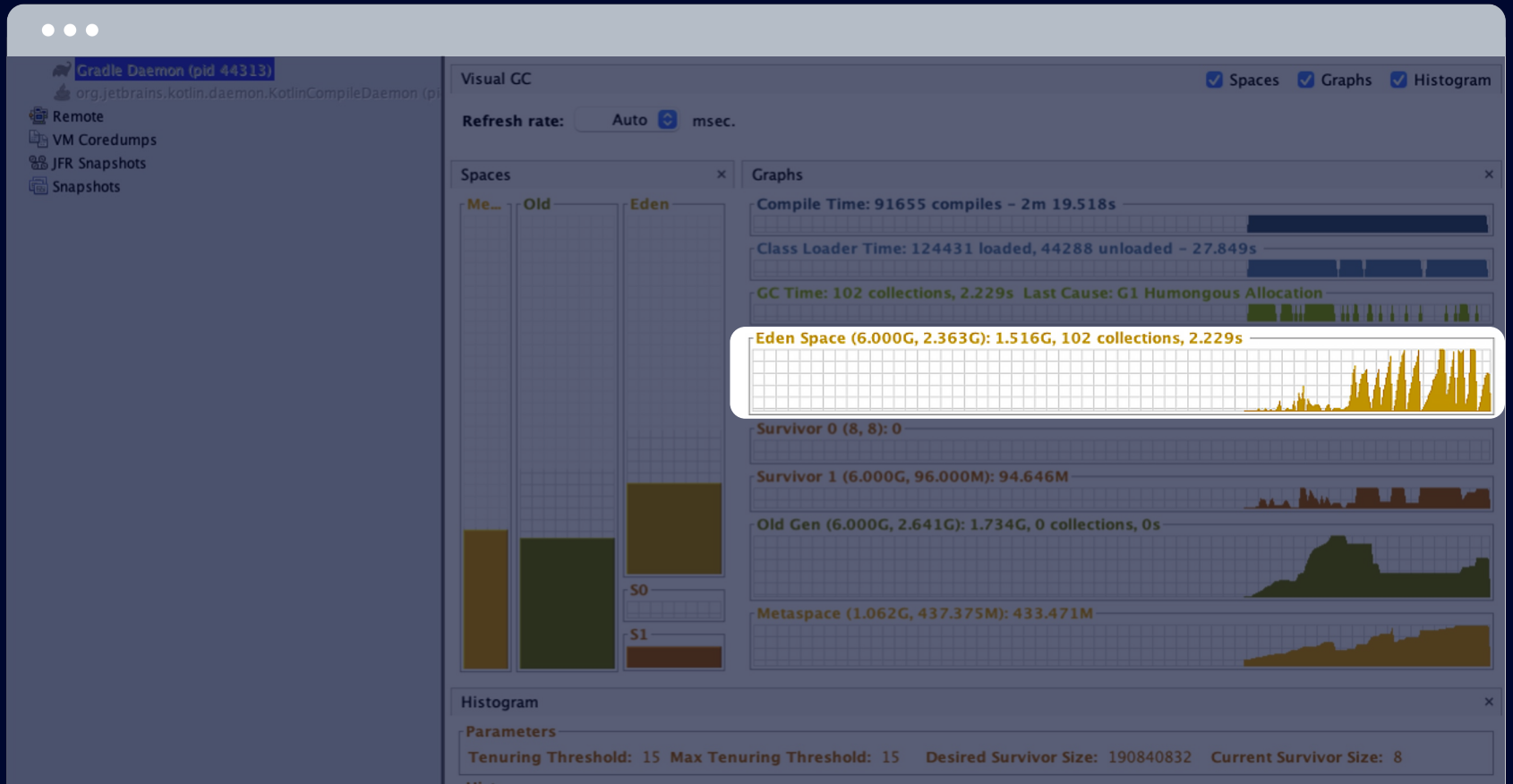
Gradle демон: Visual GC



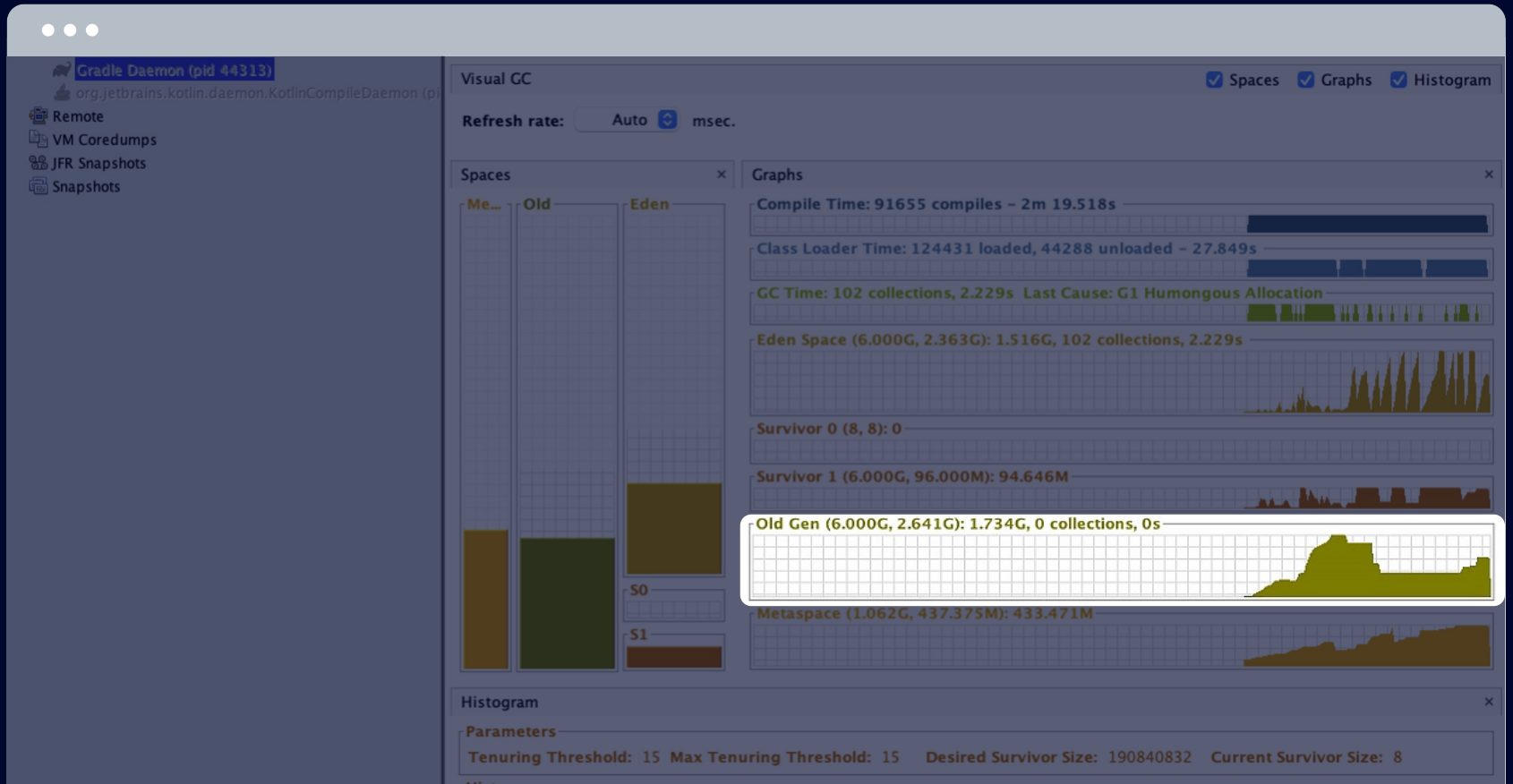
Gradle демон: Visual GC



Gradle демон: Visual GC

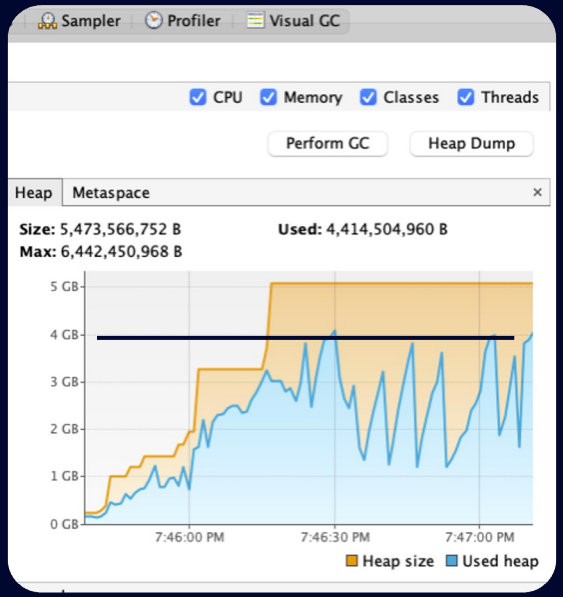


Gradle демон: Visual GC

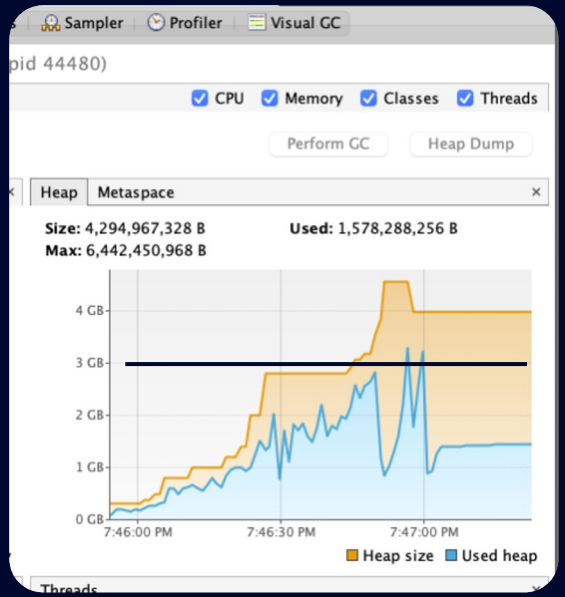


Kotlin демон: Monitor

Ограничение Java heap



Gradle daemon
-Xmx4g



Kotlin daemon
-Xmx3g

Опции JVM для Kotlin демона

gradle.properties

```
org.gradle.jvmargs=-Xmx4g -Dfile.encoding=UTF-8
kotlin.daemon.jvmargs=-Xmx3g
```

Способы задания опций:

- Gradle daemon arguments
- kotlin.daemon.jvm.options
- kotlin.daemon.jvmargs
- kotlin extension
- Specific task definition

Влияние на перфоманс

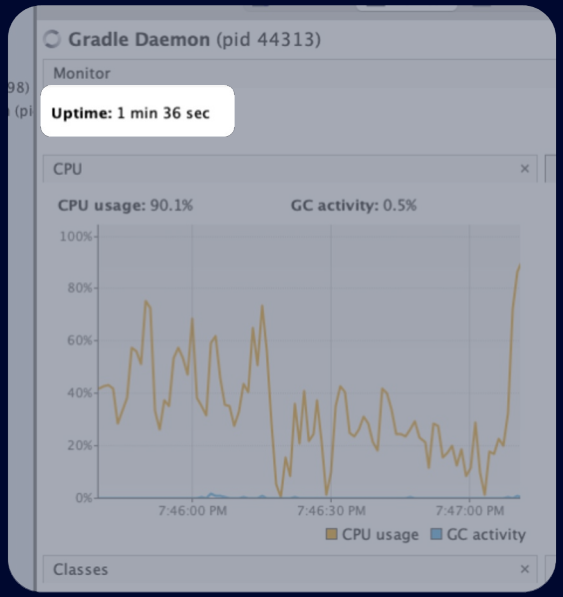
Размер Java Heap

Производительность обратно пропорциональна количеству доступной памяти.

Размер Young Generation

Больше размер, реже срабатывает Young GC. При этом чаще срабатывает Full GC.

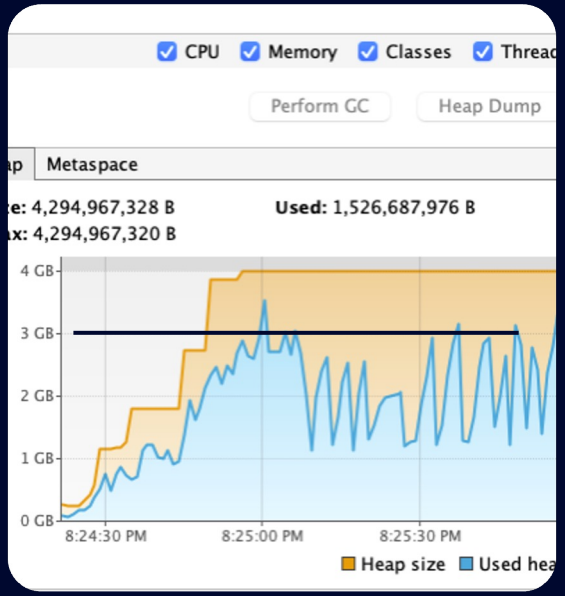
Ключевые параметры



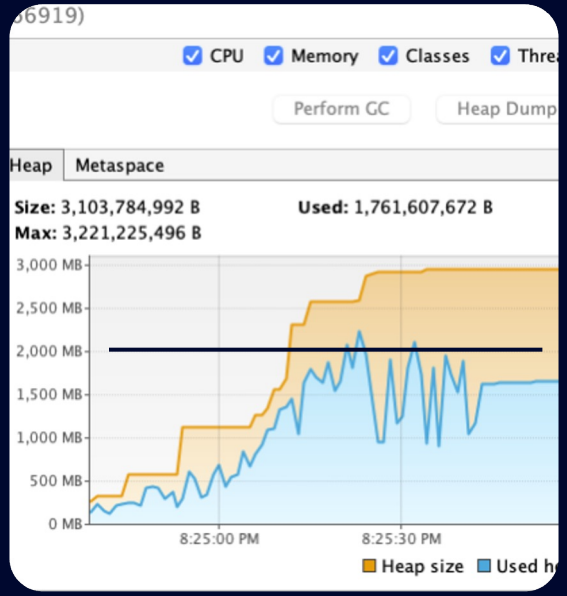
Время работы процесса

Время сборки мусора

Ограничение Java heap



Gradle daemon
-Xmx3g

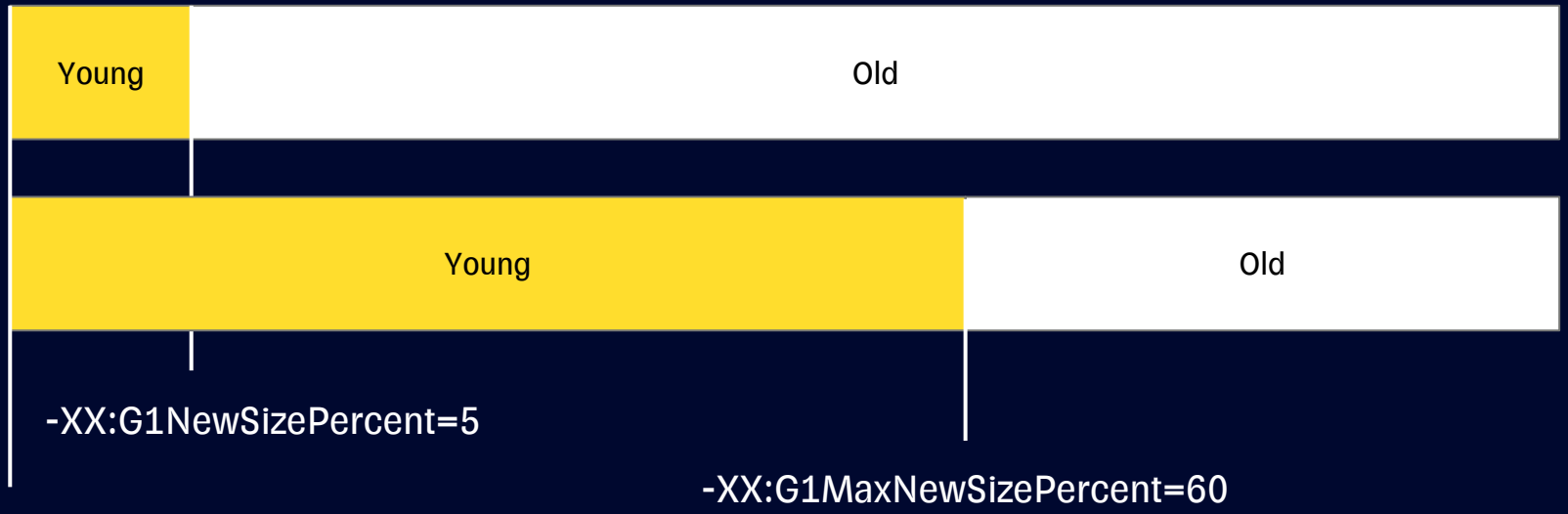


Kotlin daemon
-Xmx2g

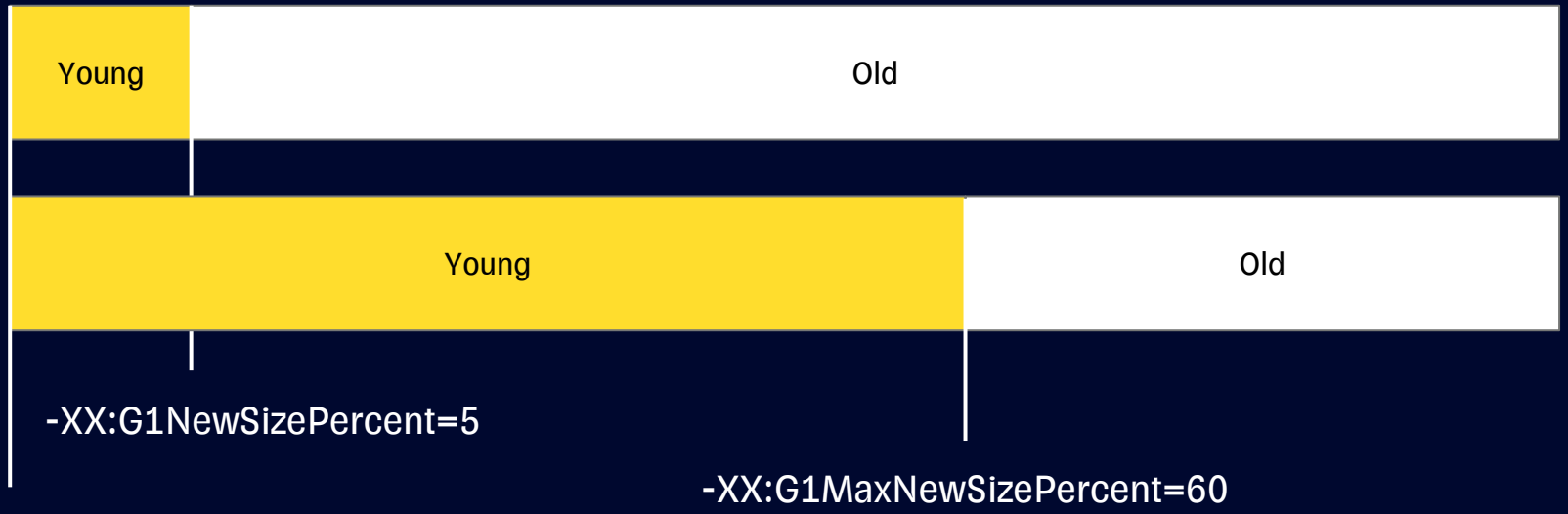
Предел уменьшения Java heap

Все «живые» объекты + 10-20%

Garbage-First: размер Young Generation

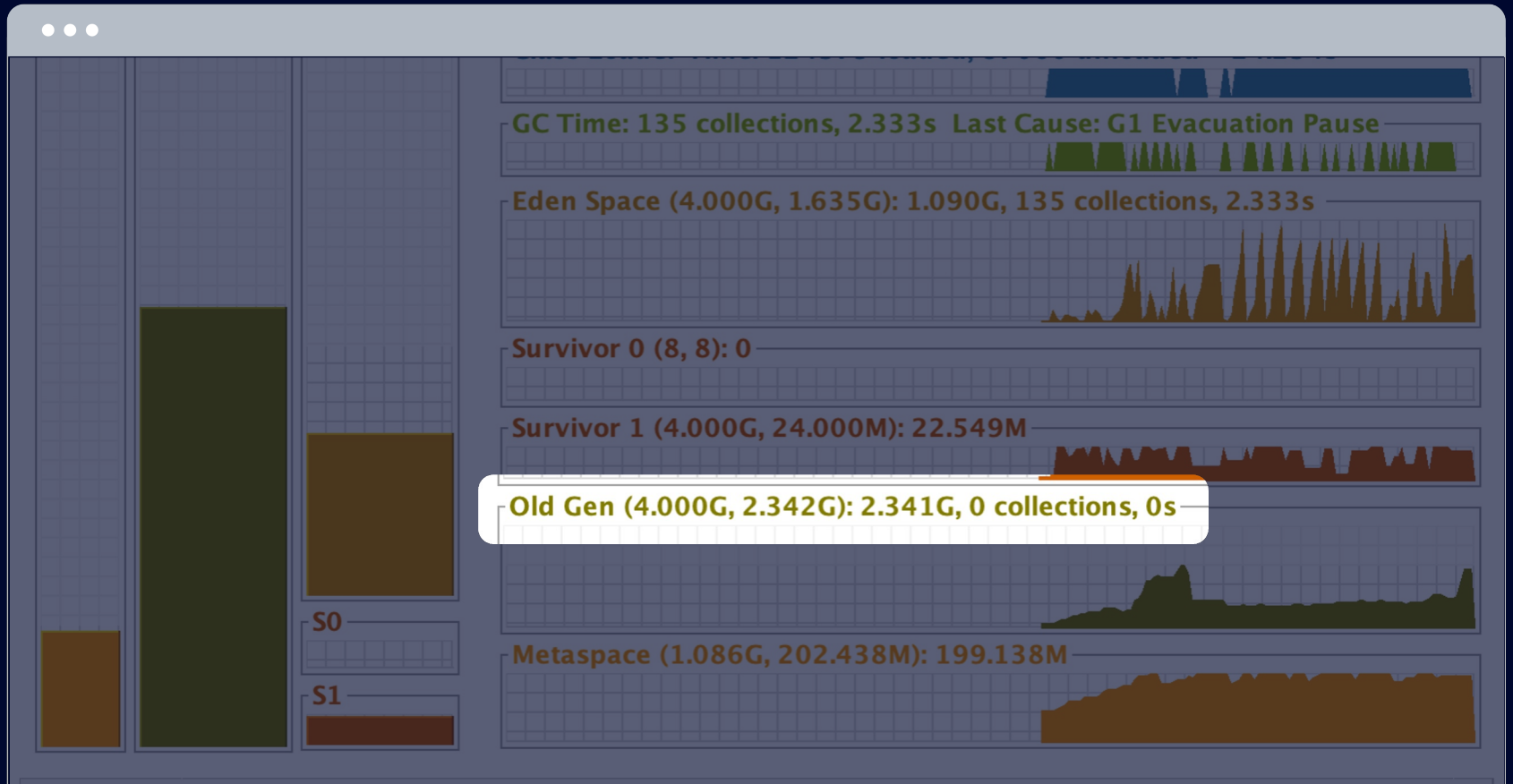


Garbage-First: размер Young Generation

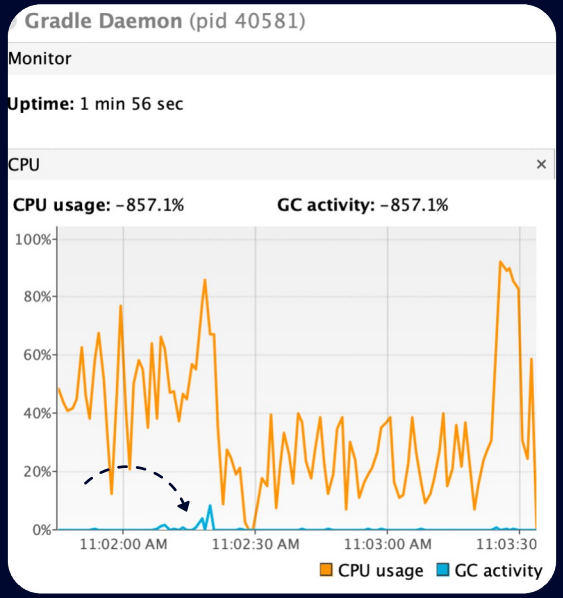


Heap size = <Old Gen used> + 5%

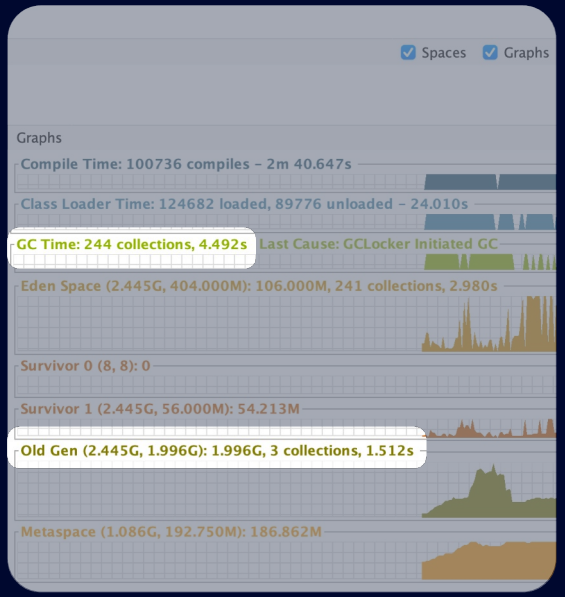
Visual GC: Gradle daemon



Gradle
-Xmx2500m

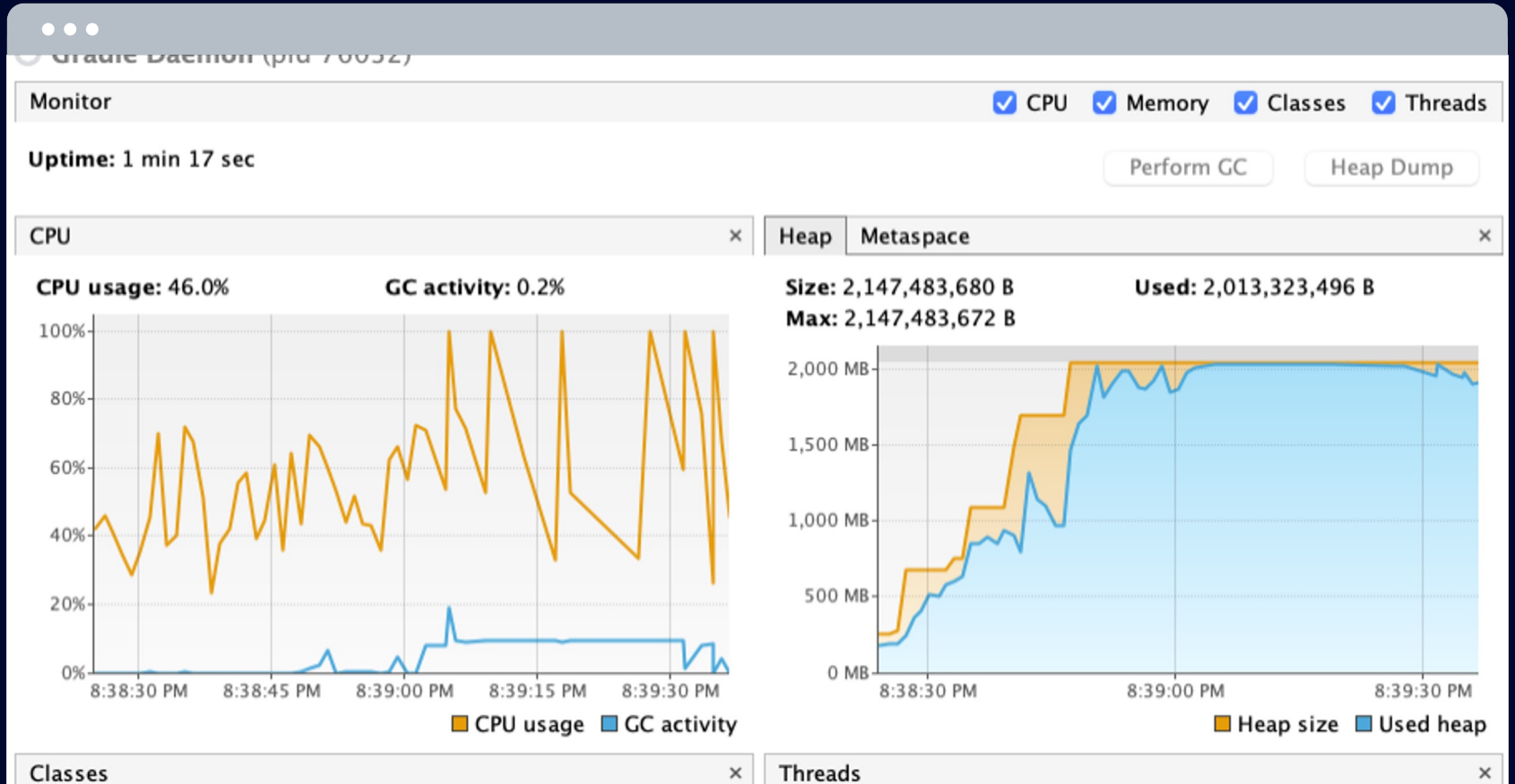


Monitor

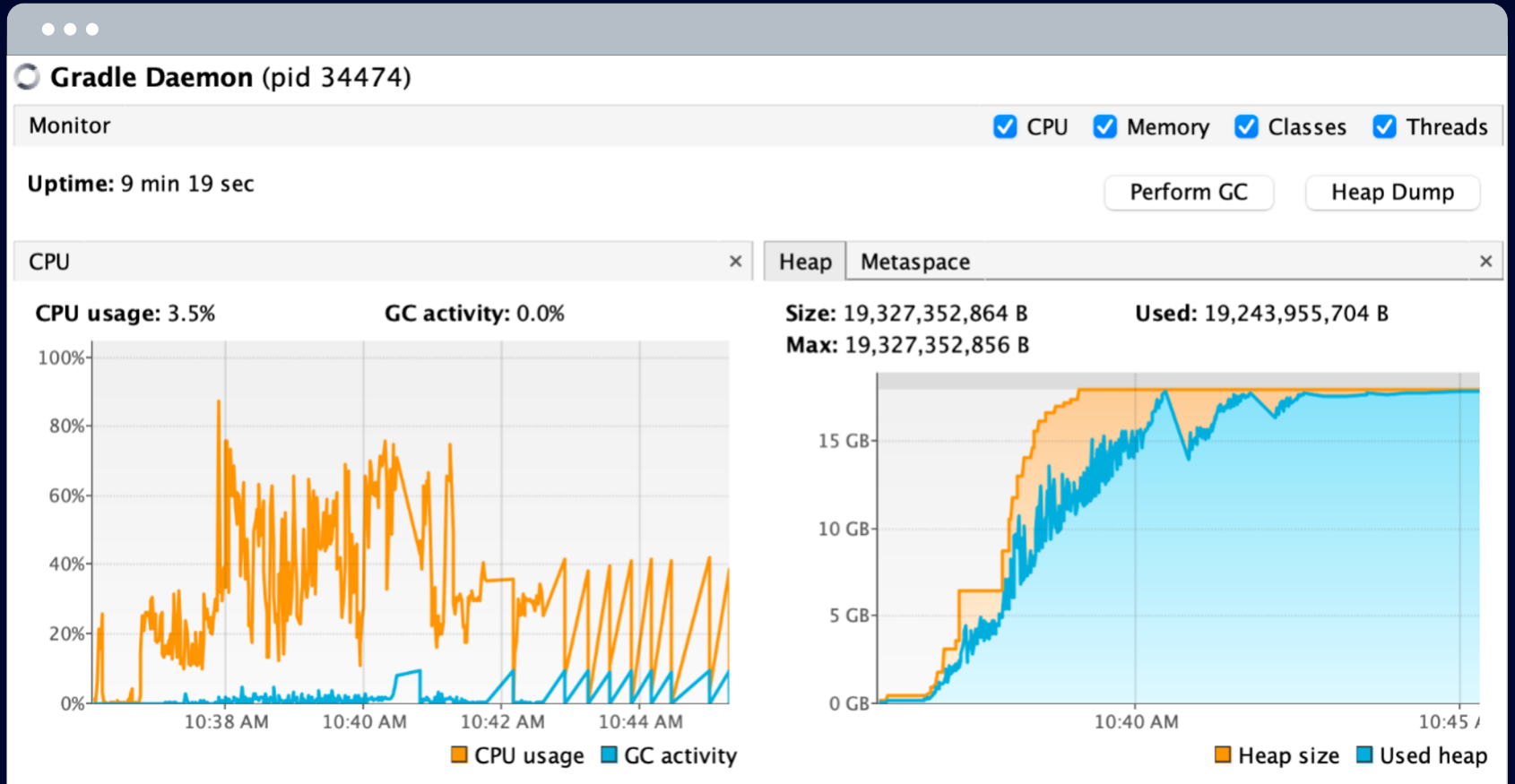


Visual GC

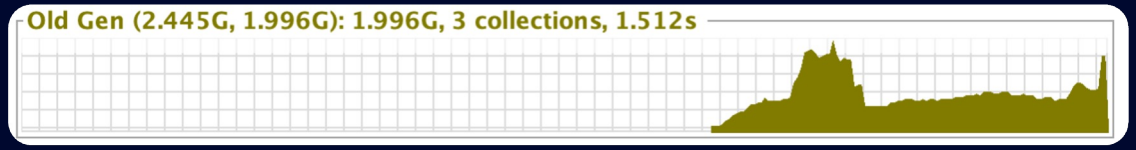
Ошибка: OutOfMemoryError



Время стремится в бесконечность



Всплески на графиках



:app:processDebugResources

:app:dexBuilderDebug

:app:mergeExtDexDebug

:app:packageDebug

Узкое место — задачи, которые затрагивают всю кодовую базу

Зависят от размера кодовой базы

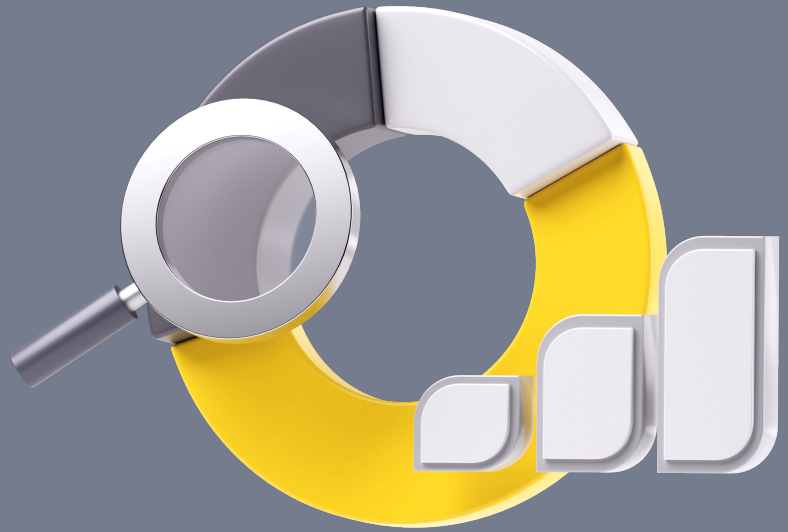
- Синхронизация с IDE
- Сборка релизной версии
- Статический анализ (Lint)

Тестирование результатов

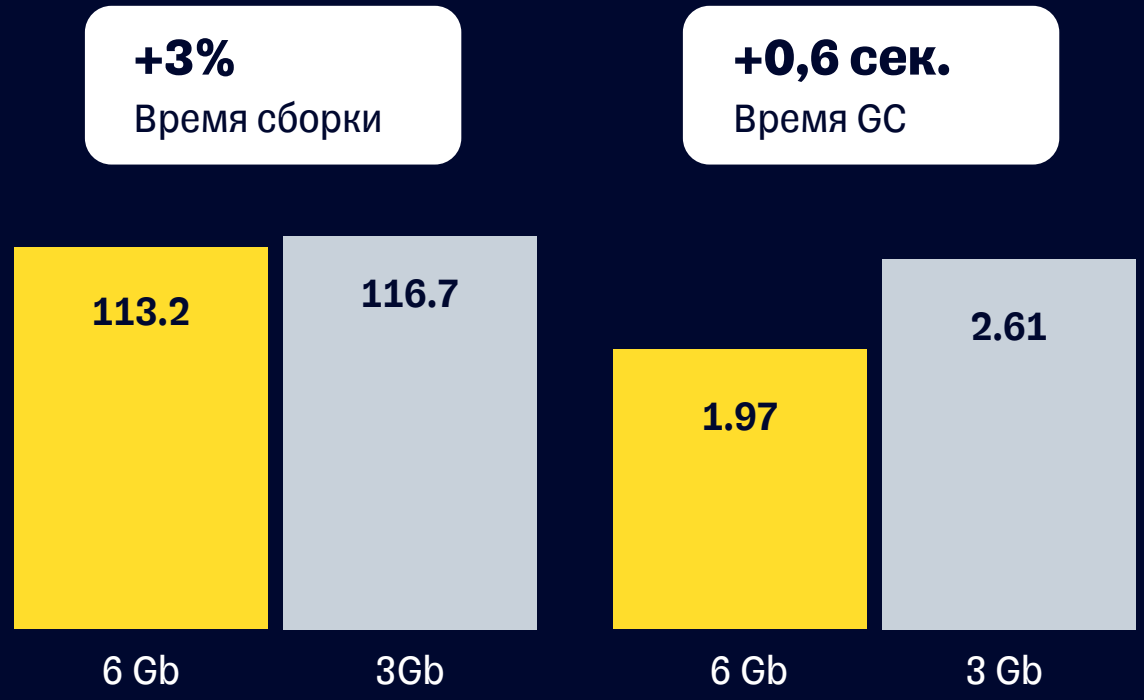
```
org.gradle.jvmargs=-Xmx3g -Dfile.encoding=UTF-8
kotlin.daemon.jvmargs=-Xmx1g
…
```

Gradle Profiler

- 2 прогревочных запуска
- 10 замеров Apple
- M1 Max
- 10 воркеров



G1 GC: изменение Heap size



Влияние размера памяти на время сборки

Gradle	Gradle + Kotlin	Время сборки
+100%	+200%	-3%
Увеличение размера Java heap для Gradle демона	Увеличение размера Java heap с учетом тюнинга Kotlin демона	Сокращение времени сборки

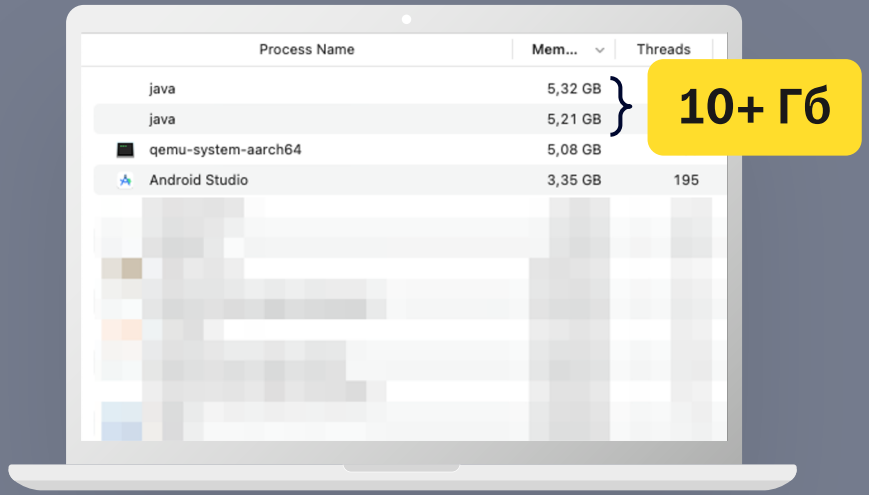
«Профит от увеличения памяти имеет предел. С определенного момента перестает влиять на время сборки.»

Использование RAM

```
org.gradle.jvmargs=-Xmx6g
```

→

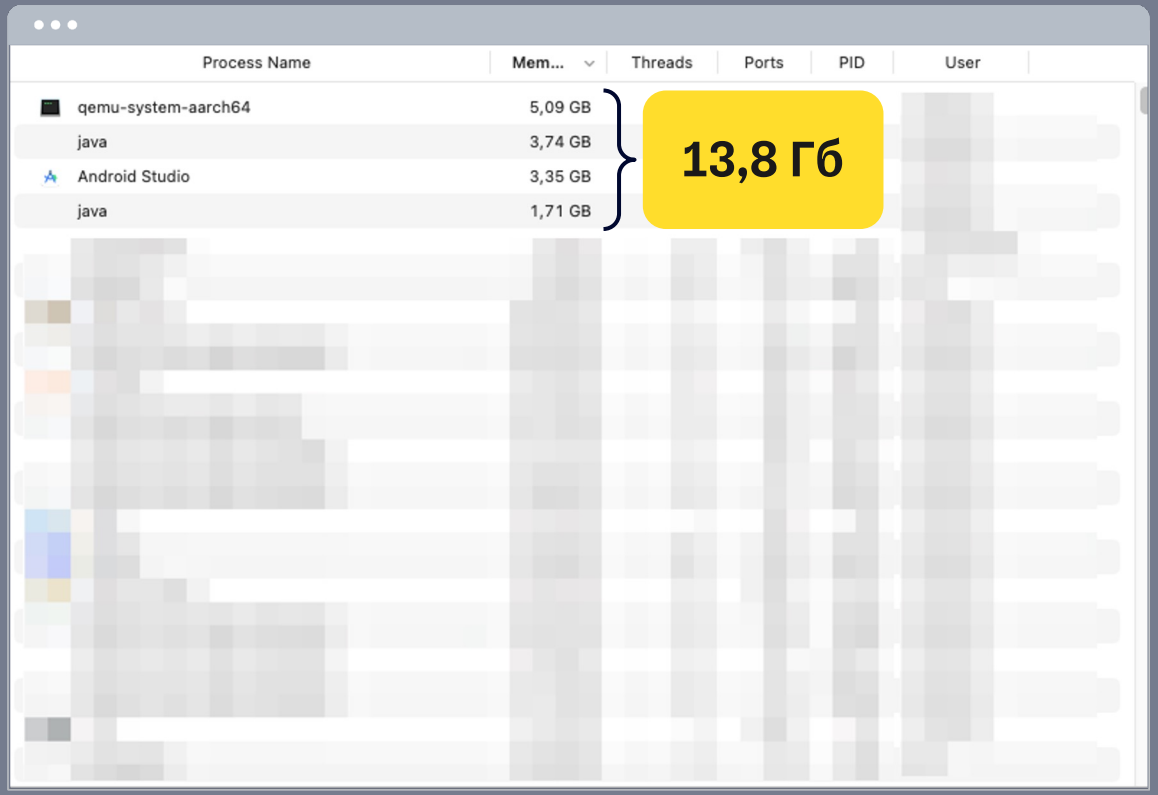
```
org.gradle.jvmargs=-Xmx3g
kotlin.daemon.jvmargs=-Xmx1g
```



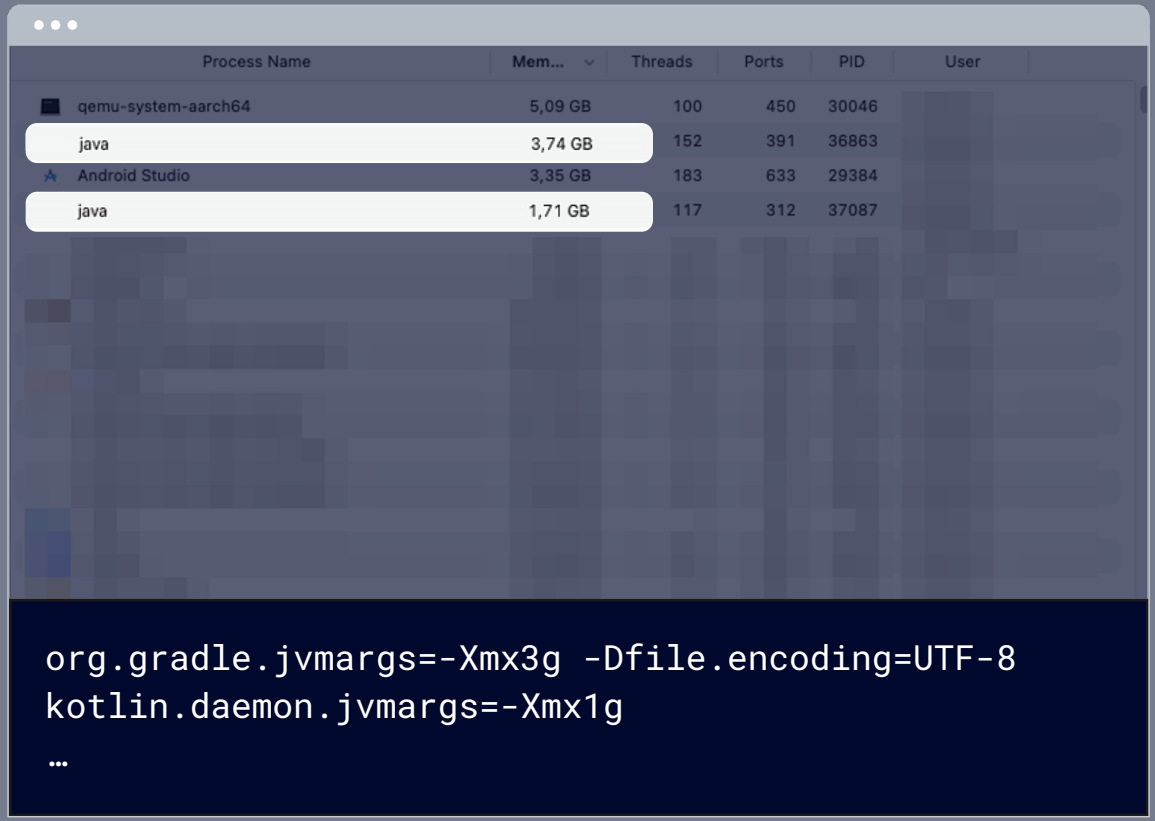
10+ Гб

3+1 ? 4

топ процессов (по памяти)



топ процессов (по памяти)



JVM Runtime

- Java Heap
- Class Loading
- JIT Compilation
- Threads

JVM Runtime

- Java Heap
- Class Loading
- JIT Compilation
- Threads

Java Heap: Heap memory

Class Loading, JIT Compilation, Threads: Non-Heap memory

Non-heap memory

- Metaspace
- Code Cache
- Thread stack
- Garbage collection

Native Memory Tracking

```
org.gradle.jvmargs=-Xmx3g … -XX:NativeMemoryTracking=summary
kotlin.daemon.jvmargs=-Xmx1g -XX:NativeMemoryTracking=summary
```

Enabling NMT causes a 5% - 10% performance overhead

Native Memory Tracking

```
org.gradle.jvmargs=-Xmx3g … -XX:NativeMemoryTracking=summary
kotlin.daemon.jvmargs=-Xmx1g -XX:NativeMemoryTracking=summary
```

Enabling NMT causes a 5% - 10% performance overhead

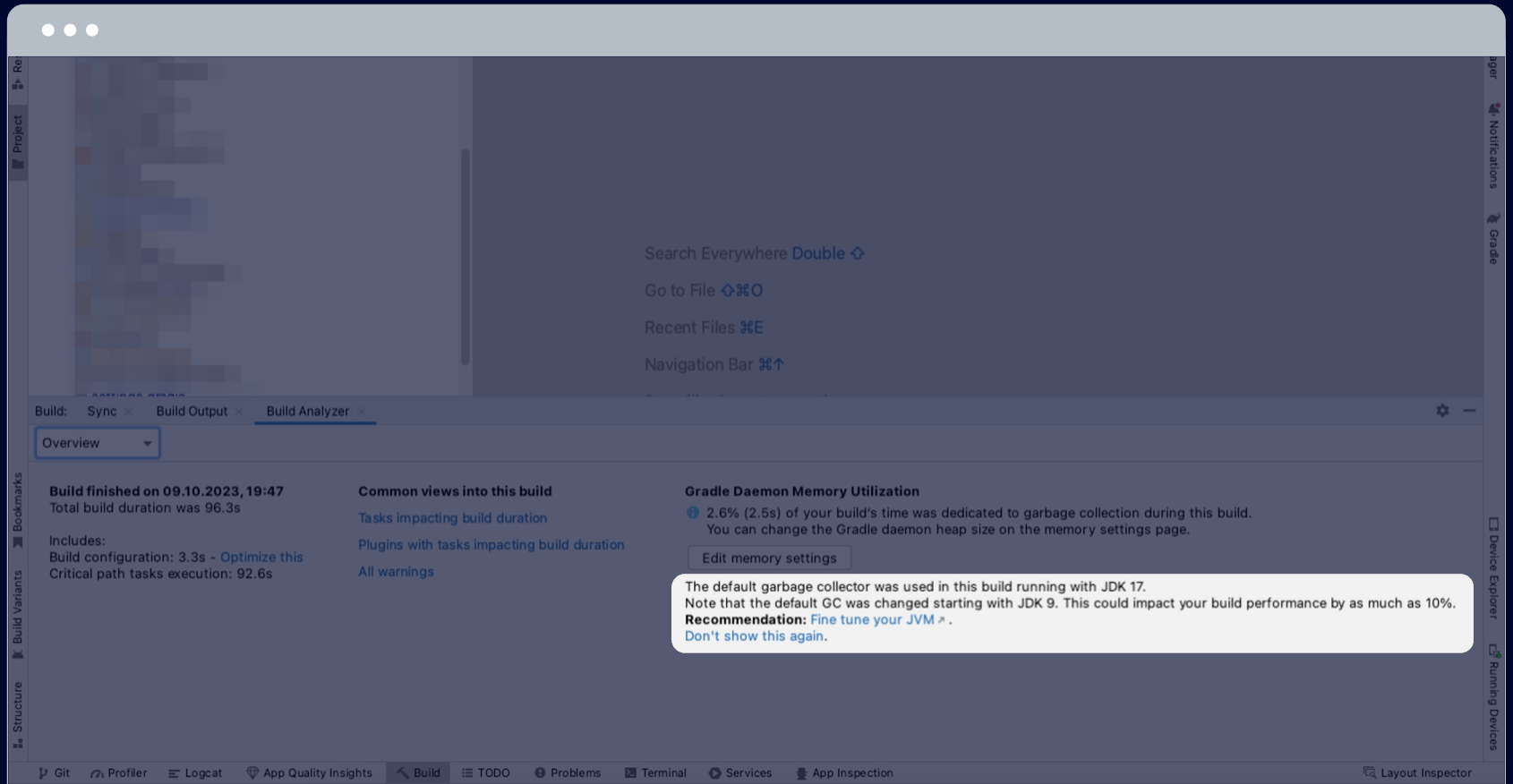
```
jcmd <PID> VM.native_memory summary
```

NMT Gradle daemon

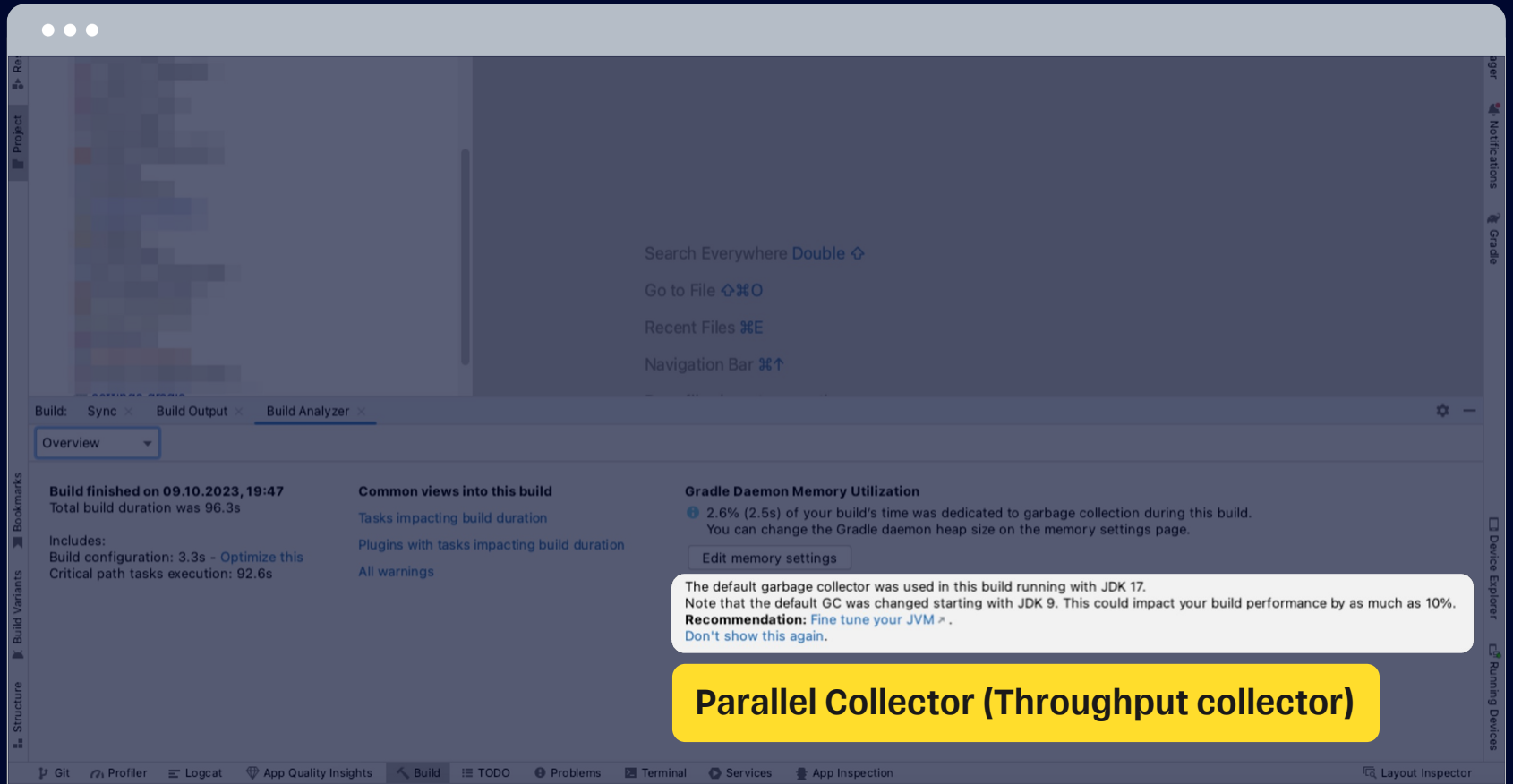
```
Total: reserved=5402797KB, committed=4156941KB

-                 Java Heap (reserved=3145728KB, committed=3145728KB)

-                     Class (reserved=5976KB, committed=5976KB)
                            (  Metadata:   )
                            (    reserved=303104KB, committed=174272KB)
                            (  Class space:)
                            (    reserved=1048576KB, committed=28864KB)

-                    Thread (reserved=324289KB, committed=324289KB)

-                      Code (reserved=259561KB, committed=162249KB)

-                        GC (reserved=205770KB, committed=205770KB)

-                    Symbol (reserved=34808KB, committed=34808KB)
```

«Для сборки нужно больше памяти, чем сумма Java heap для Gradle и Kotlin процессов.»

Build Performance Analyzer



Build Performance Analyzer



Эргономика

Parallel Collector

Maximum Pause-Time Goal

-XX:MaxGCPauseMillis=<MAX_ULONG>

Throughput Goal

-XX:GCTimeRatio=99

$$\frac{1}{\text{GCTimeRatio} + 1} = \frac{1}{99 + 1} = 1\%$$

Parallel GC

```
org.gradle.jvmargs=-Xmx3g -Dfile.encoding=UTF-8 -XX:+UseParallelGC
kotlin.daemon.jvmargs=-Xmx1g -XX:+UseParallelGC
…
```

Parallel GC

```
org.gradle.jvmargs=-Xmx3g -Dfile.encoding=UTF-8 -XX:+UseParallelGC
kotlin.daemon.jvmargs=-Xmx1g -XX:+UseParallelGC
…
```

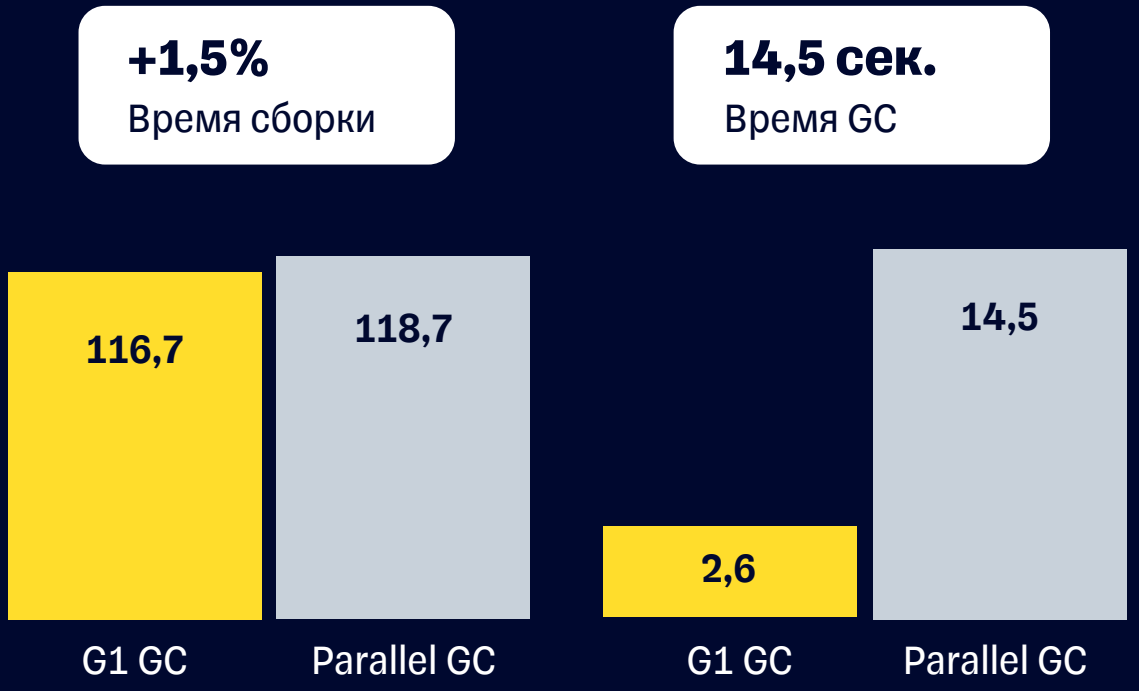
Как изменится время сборки?

Сократится

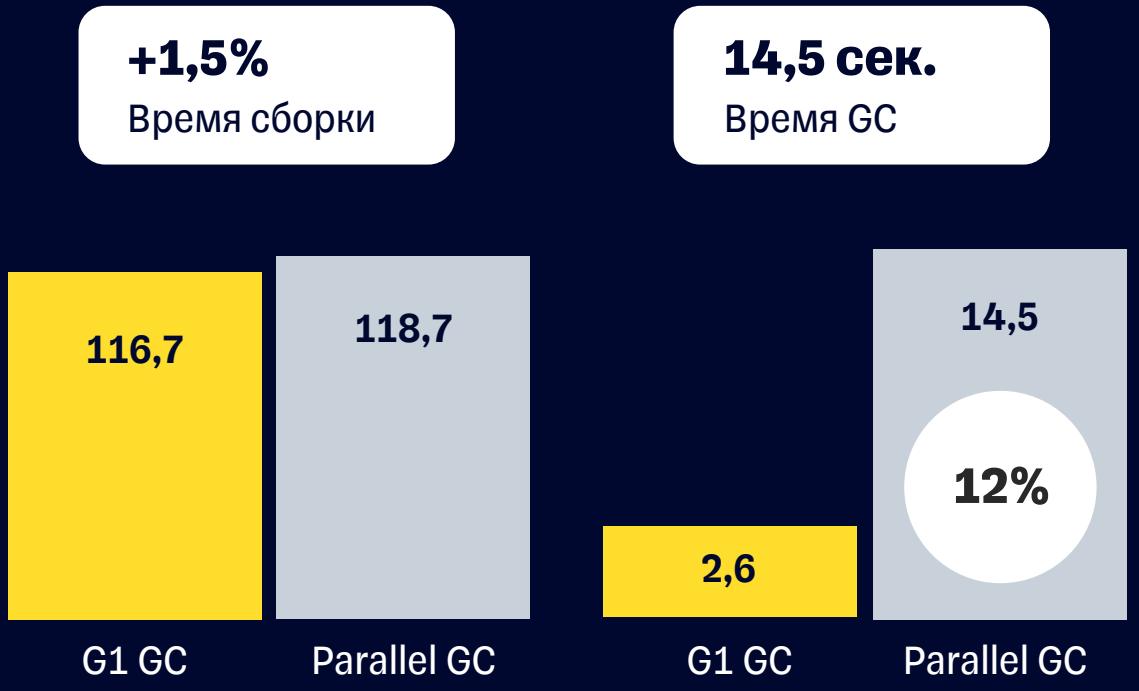
Увеличится

Сохранится

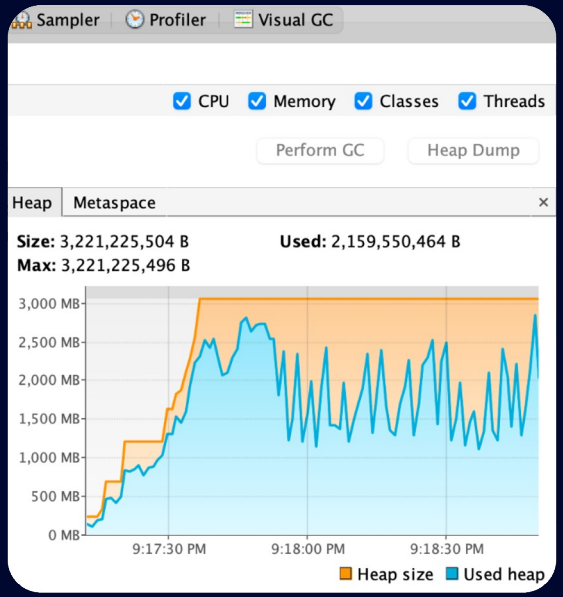
G1 GC против Parallel GC



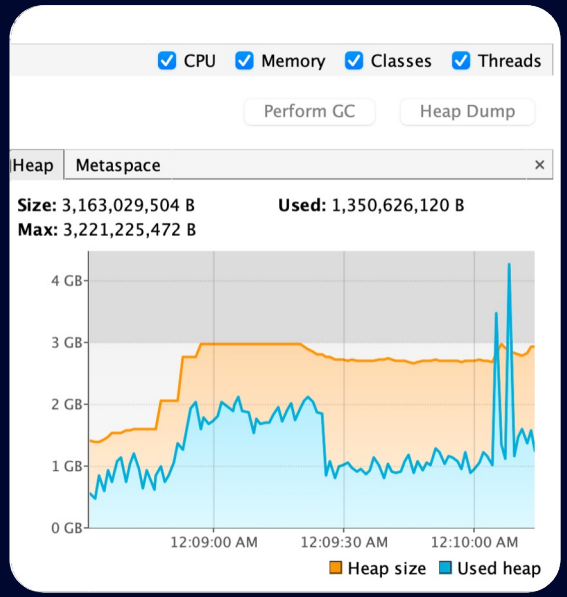
G1 GC против Parallel GC



Размер Java heap Gradle демон



-XX:+UseG1GC



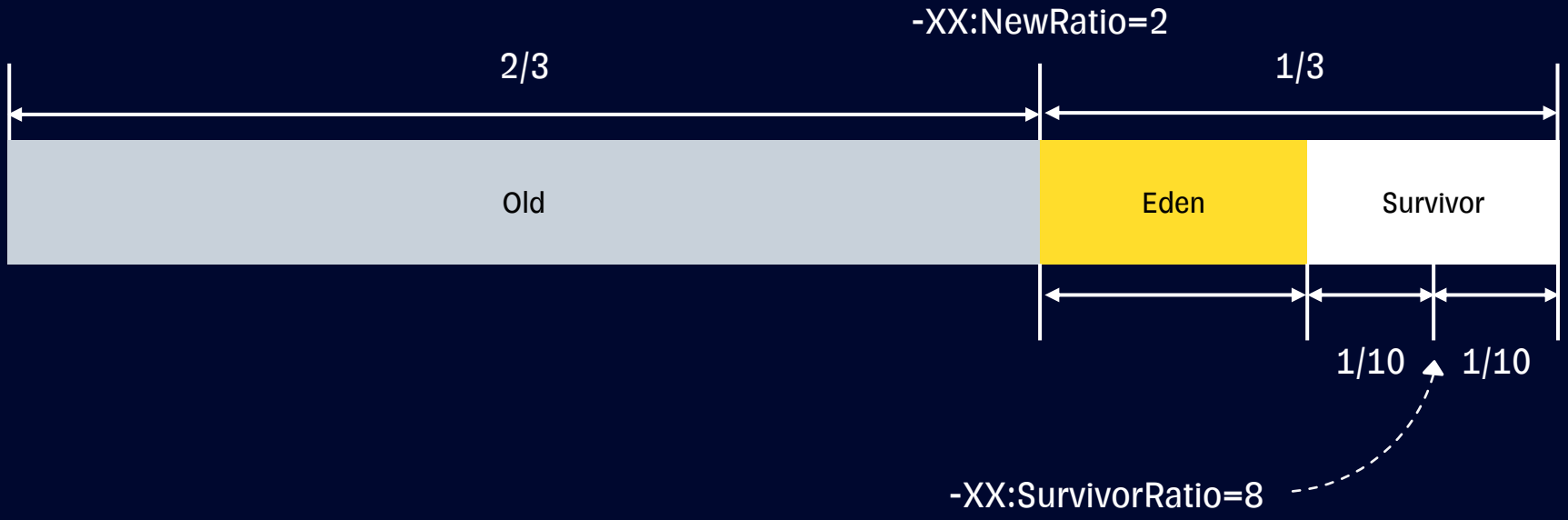
-XX:+UseParallelGC

Parallel GC

Расположение блоков памяти

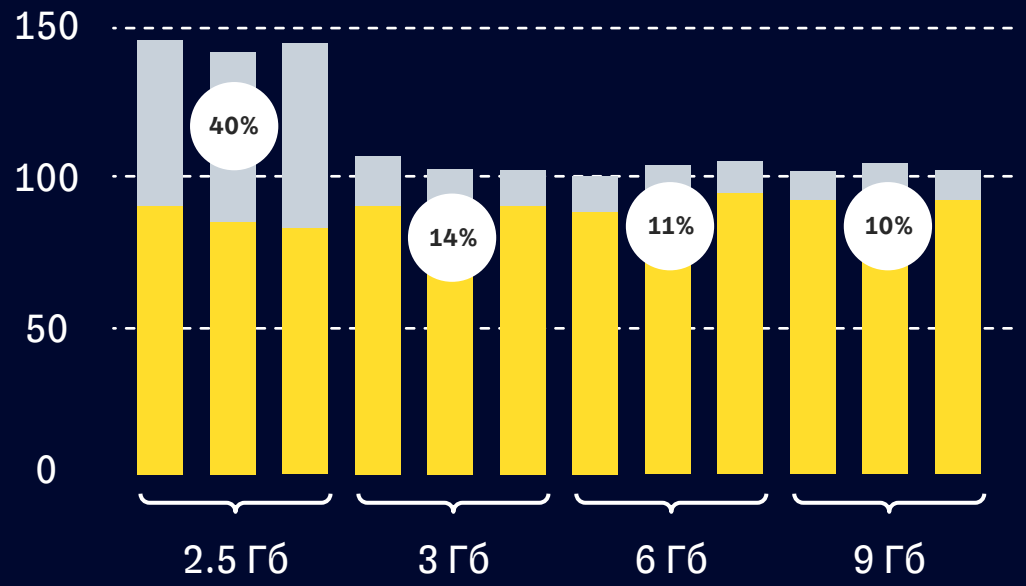
Parallel GC

Расположение блоков памяти



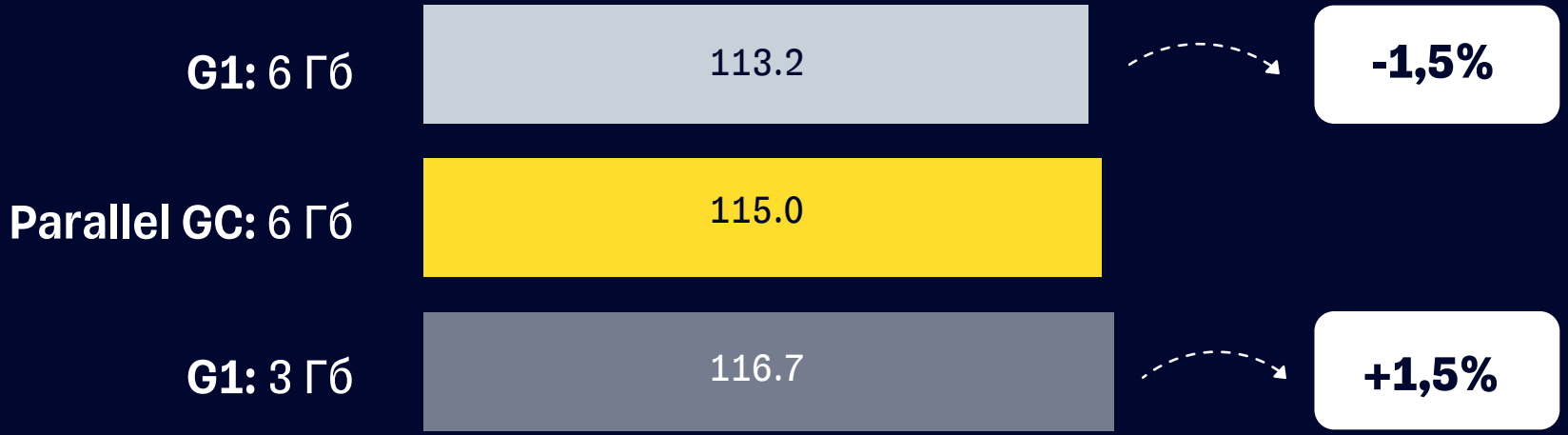
Parallel GC: изменение Heap size

Время сборки vs. размер Java heap



- Прямая зависимость между временем сборки и GC
- Минимальное время сборки мусора — 10%

Parallel GC против Garbage-First



«Любые советы по улучшению производительности стоит проверять на своем окружении.»

Заключение

Заключение

На время сборки влияет размер Java heap

Профит от увеличения памяти имеет предел

Выигрыш можно оценить по времени работы GC

Java процессу нужна память для своих задач

Проверяйте советы из статей по перфомансу

Заключение

На время сборки влияет размер Java heap

Профит от увеличения памяти имеет предел

Выигрыш можно оценить по времени работы GC

Java процессу нужна память для своих задач

Проверяйте советы из статей по перфомансу

Заключение

На время сборки влияет размер Java heap

Профит от увеличения памяти имеет предел

Выигрыш можно оценить по времени работы GC

Java процессу нужна память для своих задач

Проверяйте советы из статей по перфомансу

Заключение

На время сборки влияет размер Java heap

Профит от увеличения памяти имеет предел

Выигрыш можно оценить по времени работы GC

Java процессу нужна память для своих задач

Проверяйте советы из статей по перфомансу

Заключение

На время сборки влияет размер Java heap

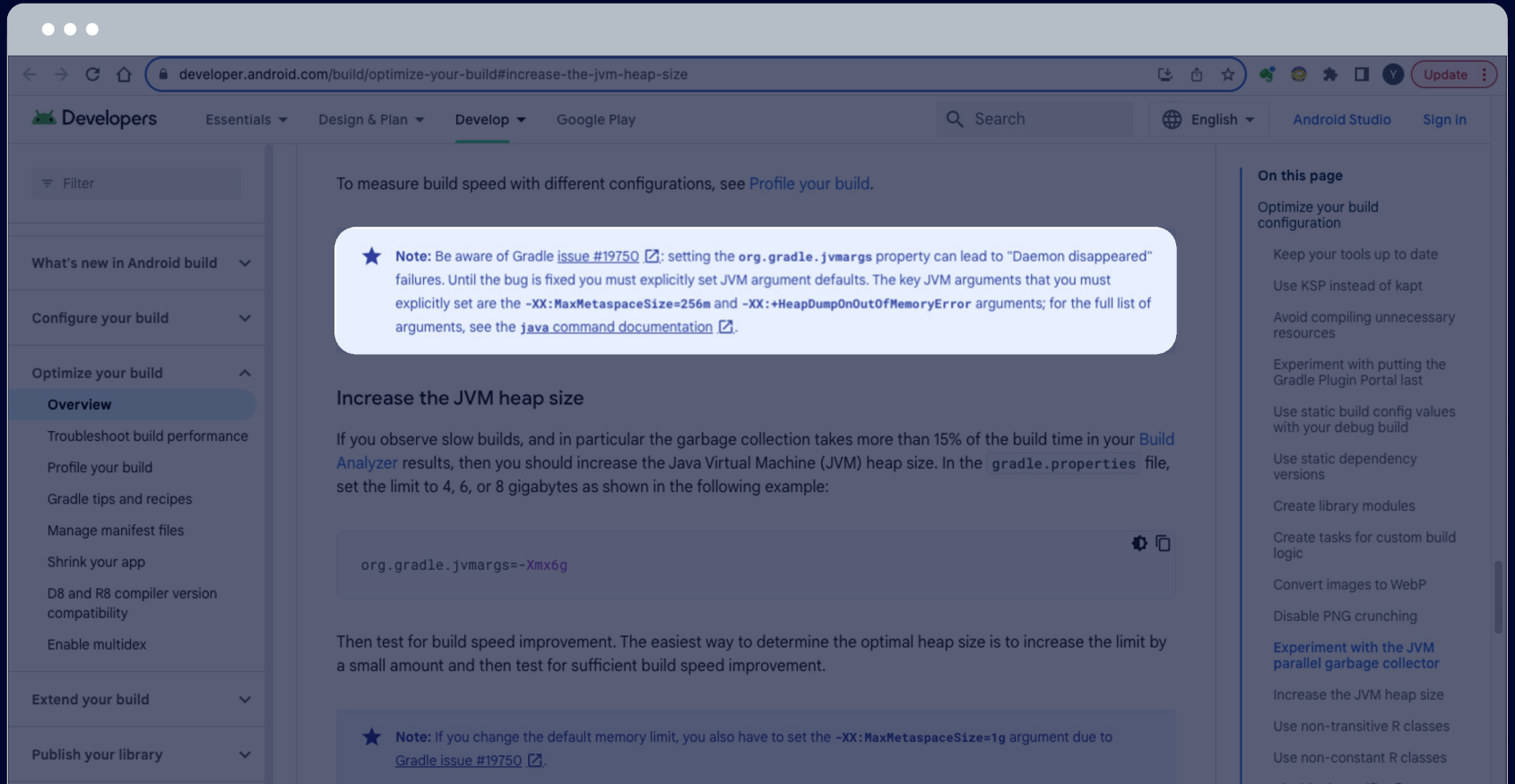
Профит от увеличения памяти имеет предел

Выигрыш можно оценить по времени работы GC

Java процессу нужна память для своих задач

Проверяйте советы из статей по перфомансу

Устаревшая информация (Java 8)

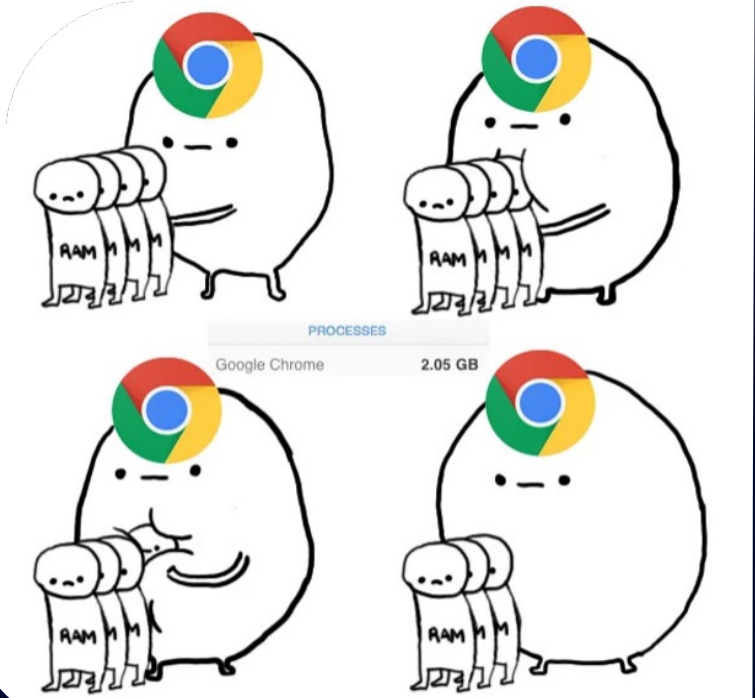


Полезные ссылки

- JDK 17 Documentation
- Garbage Collection Tuning Guide
- Java Virtual Machine Guide
- JDK Tool Specifications
- Troubleshooting Guide

Памяти много не бывает

- Google Chrome
- Android Studio
- Android Emulator
- Gradle



ТИНЬКОФФ

Слайды

Юрий Анисимов | Разработчик Инфра.

y.anisimov@tinkoff.ru

ТИНЬКОФФ