Здоровье вашей Gradle сборки

и моей

Авито, Speed Android, Сергей Боиштян

Обо мне

- Все хотят, чтобы я был тимлидом, но я все еще рад быть Sowtfare Engineer
- Был Backend, Android, Teamlead
- Последние 4 года в Авито в команде Speed
- https://github.com/sboishtyan

О нашей команде

- Делаем инструменты для: тестирования, CI, CD, Android разработки
- Infra в open source
- Есть канал, где можно обсуждать CI и сборку для Android
  - https://t.me/avito_android_opensource

Весь доклад на одном слайде

- У нас большой проект, в котором много автоматизированных проверок
- Мы поняли, что время сборки очень хрупкое. Есть много изменений, которые могут ускорить, замедлить или изменить стабильность проверок
- Чтобы сделать время сборки предсказуемым, чтобы определять плохие и хорошие сборки, нужны инструменты диагностики и мониторинга
- Я расскажу
  - за чем и почему мы следим
  - Какие инструменты для диагностики используем

Disclaimer 1
Все интерпретации и определения субъективны. Могут отличаться от ваших. Не теряйте за ними сути, пожалуйста

Disclaimer 2
Доклад больше про мониторинг, а не про ускорения.

Disclaimer 3
Невозможно рассказать обо всем за 45 минут. Поэтому ловите меня на конфе. С радостью обсужу ваши сценарии и расскажу, как у нас

-1

Пролог

Все началось с CI

CI простыми словами

Умные люди подумали и решили:

- Чем быстрее написан код
- Чем быстрее он проверен
- Чем быстрее он вмержен и проверен после интеграции

Тем быстрее можно сделать новую версию

Android CI в Avito

Чтобы быстро писать код

Мы используем monorepo. Сейчас у нас около 1000 модулей

- Легко переиспользовать код
- Легко делать глобальные рефакторинги
- Нет проблем с синхронизацией нескольких репозиториев

Чтобы проверять все быстро

У нас все проверки запускаются уже на PR*

- ~ 15000 Robolectric Unit тестов
- ~ 2000 Instrumentation тестов с замоканой сетью
- ~ 600 E2E тестов
- Detekty linty и много написанных правил

* Кроме E2E тестов

Чтобы сделать новую версию быстрее

Каждую ночь мы запускаем специальную сборку

- Запускаем все-все тесты на develop ветке
- Делаем отчеты для каждой команды с результатами их тестов
- Команды регулярно автоматизируют свой регресс план, чтобы проверки запускались на PR

Итого

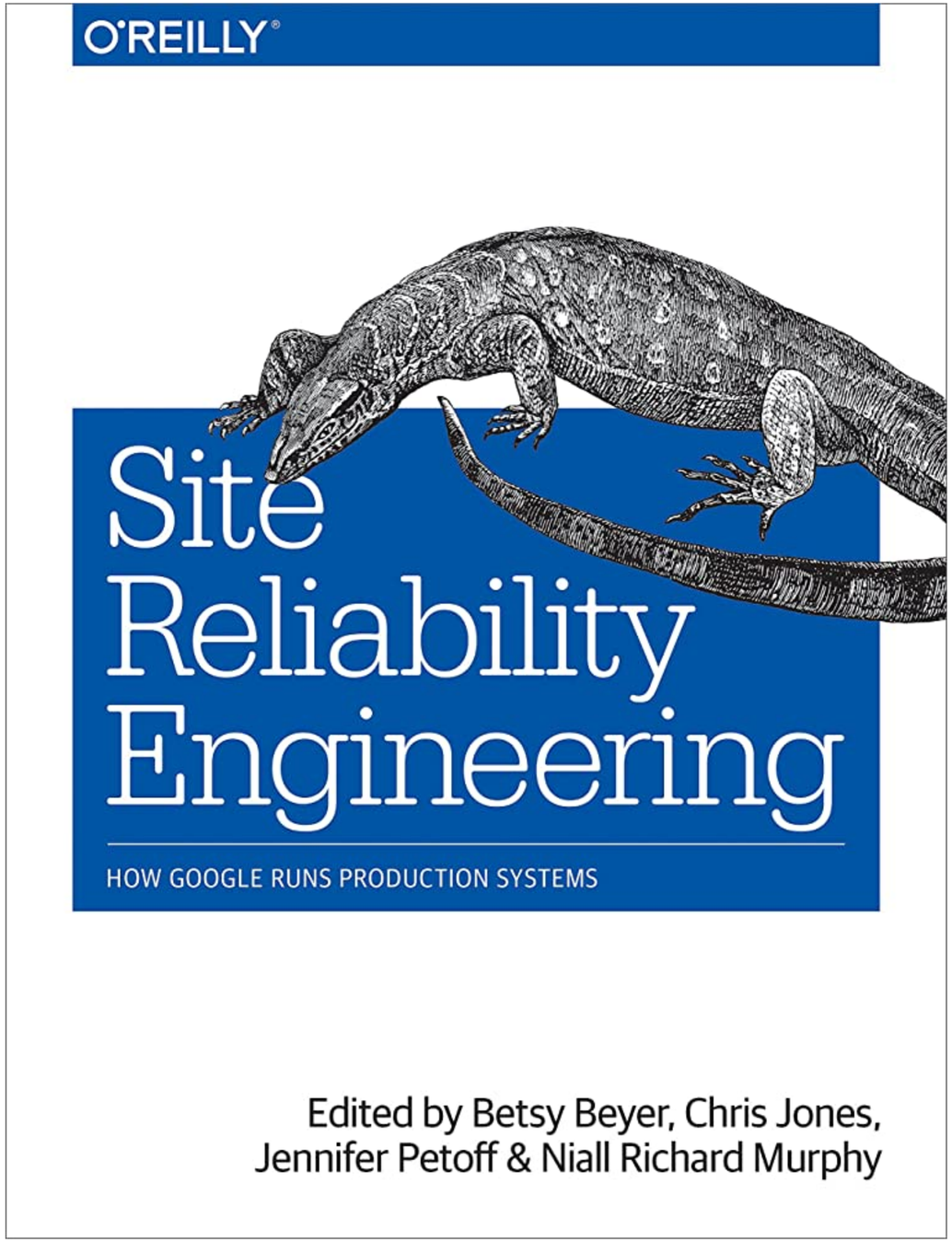
- 1000 модулей
- ~ 15000 Robolectric Unit тестов
- ~ 2000 Instrumentation тестов с замоканой сетью
- ~ 600 E2E тестов
- Detekty линты и много написанных правил
- Регулярные ночные сборки
- Релизы раз в неделю
- ~ 80 Android разработчиков
- ~ 100 только PR сборок в день

В один момент мы поняли, что наша CI/CD система стала сложной для понимания

На практике узнали, что даже пытаясь все улучшить, можно все ухудшить

Мы искали ответы

А изменила все одна книга



В этом квартале

Я сделал ROADMAP для улучшений за 1 час

Итого

- Хочу добавить импакт анализ, чтобы запускать только нужные UI и Unit тесты
- Разобраться почему в конце сборок есть 6 минутная задержка. Скорее всего дело в построении build-trace и отправке build-metric
- Убрать в Авито сборку лишних build variant: ruStore, Xiamoi, staging можем спокойно не собирать
- Увеличить кол-во девайсов для avito UI тестов, чтобы ускорить эту задачу
- По Gradle cache нужно разбираться, что чаще всего в miss и сколько оно занимает

*Моя жена сказала, что тут не хватает нескольких запятых. К сожалению, это скриншот

0

«Здоровье» сборки

Чтобы определить «здоровье», нужно понимать сборку

Понимать — уметь интерпретировать телеметрию сборки

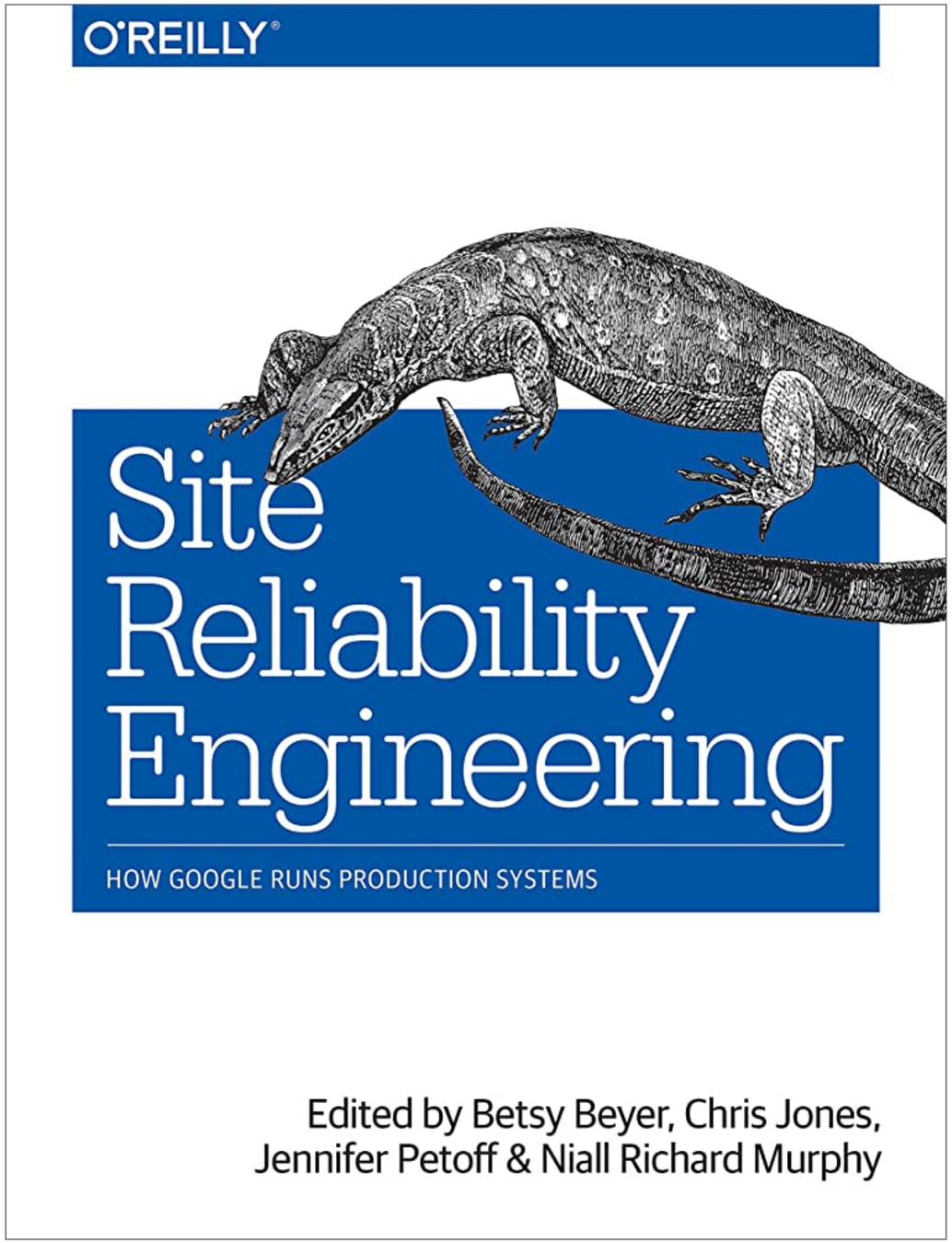
Телеметрия

- Метрики
- Логи
- Трейсы
- И т.д.

А потом вам нужно зафиксировать желаемый уровень «здоровья»

Начало 2020

Внедряем SLA



SLA — Service Level Agreement

Договориться и установить уровень гарантий обслуживания

SLA	SLOs	SLIs
Promise _____	Goal _____	How did we do? _____
Promise _____	Goal _____	How did we do? _____
Promise _____	Goal _____	How did we do? _____

Для простоты я буду называть все SLA

Наш SLA с Android разработчиками

- Время сборки на PR
- Стабильность. Отсутствие Flaky сборок на PR

SLA пример

Мы гарантируем, что 90% сборок на Pull request будут проходить максимум за 40 минут

Из 100 сборок, 90 шт. должны идти 40 минут и меньше

Если SLA не нарушен — ничего не ускоряем

Если SLA нарушен — ускоряем

«Здоровье» сборки

- Набор SLA которые вы для себя заключили
- Набор SLA субъективен
- Набор SLA исходит от ваших пользователей
- Набор SLA исходит от ваших ресурсов и возможностей

«Здоровая» сборка

SLA соблюдены

«Больная» сборка

SLA не соблюдены

Кому нужны SLA?

- У кого есть пользователи
- У кого ограничены ресурсы: железо, кол-во разработчиков

SLA помогает

- Ставить приоритеты
- Фокусироваться на важном
- Измерять работу команды
- Делать работу прозрачней для пользователей

1

Что ломает SLA
«Болезни» сборок

Наши SLA — это опыт сломанных сборок и недели попыток их починить

Сейчас я вам покажу, что сборки очень хрупкие

Примеры «болезней», с которыми сталкивались последнее время

Обновлянка

Обновлянка

Обновления ключевых зависимостей:

- Gradle
- AGP
- Kotlin

Может повлиять на время, стабильность и Gradle cache hit rate

Обновлянка Gradle

Обновляли Gradle на 7.6

Vladislav Yundin added a comment - 10/Feb/23 14:03

Обновили версию, следили с 24.01.23 по 31.01.23.

После обновления появились флаки-билды, падающие по OOM. После отката пропали.

Также значительно выросло время в очереди. После отката вернулось не сразу, возможна связь с кучей перезапусков релиза в тот период.

Ожидаем 7.6.1, где поправят проблемы с памятью https://github.com/gradle/gradle/issues/23215

Gradle 7.6: high memory usage (android project) #23215

Closed **rmarma** opened this issue on Dec 17, 2022 · 26 comments

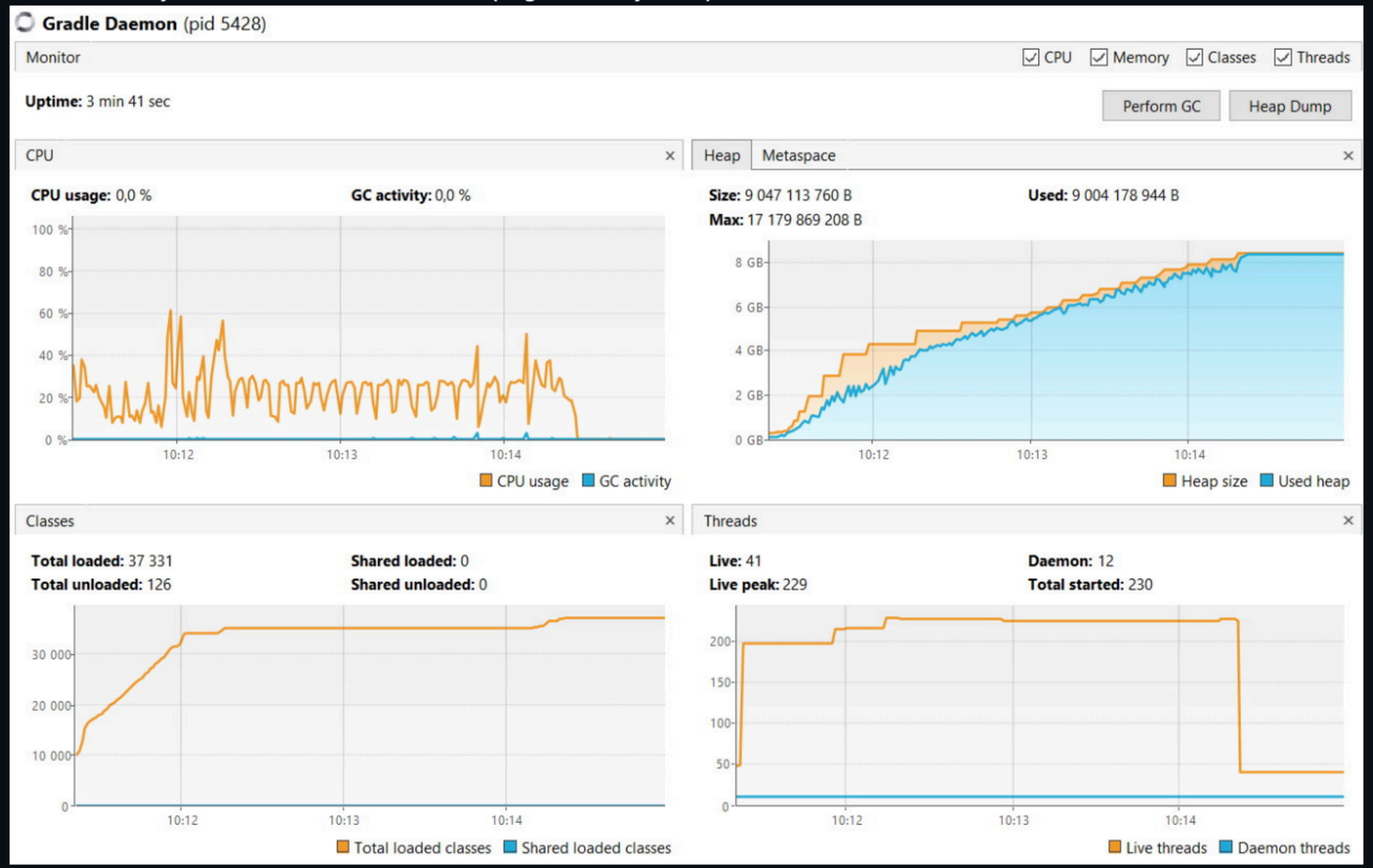
8e

rmarma commented on Dec 17, 2022 • edited

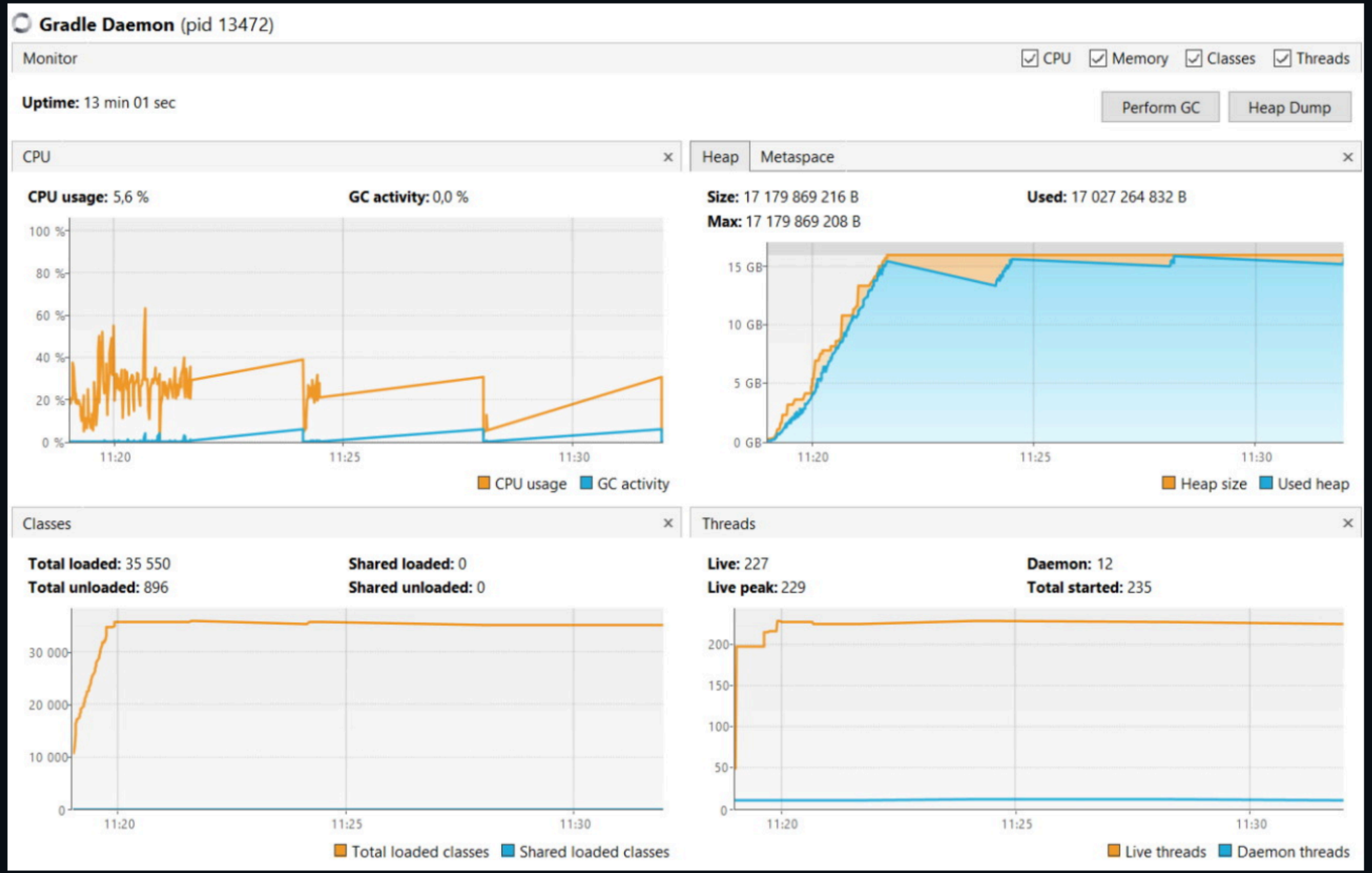
After upgrading to version 7.6, memory consumption has increased significantly.

Empty project with 1000 android modules.

Gradle 7.5.1: sync is successful in 3 minutes (10gb memory used)



Gradle 7.6: Syncing failed after 15 minutes (memory needs more than 16gb)



Обновлянка Kotlin

~~KT-57867~~ Created by Ben Lee a month ago Updated by Alexander Udalov 3 weeks ago Visible to issue readers

Performance issues in Kotlin 1.8 causing 10+ minute regression in clean builds

16

Relates to 1

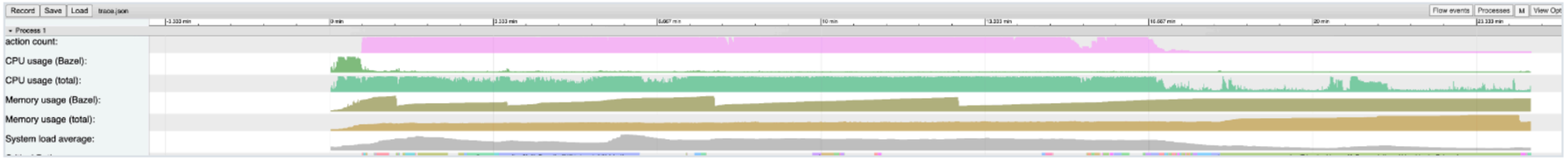
RELATES TO 1 ISSUE (0 UNRESOLVED)

M ~~KT-56789~~ Metaspace memory leak in CoreJrtFileSystem 2

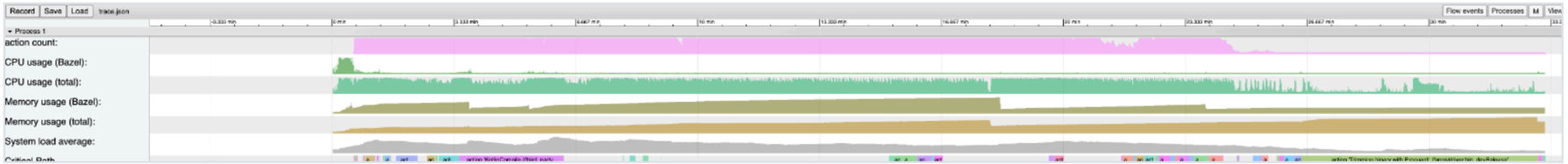
We are working on updating from Kotlin 1.7.10 to 1.8.20 and are seeing 10+ minute regression in clean build times, and a more than 2x increase in memory consumption. We are able to reproduce this on all versions of Kotlin 1.8 (1.80, 1.8.10, 1.8.20).

Profiles:

Kotlin 1.7 profiles where you can see that the "Memory usage total" peaks at around 30gb.



Kotlin 1.8 profiles where you can see that the "Memory usage total" steadily climbs to about 63gb.



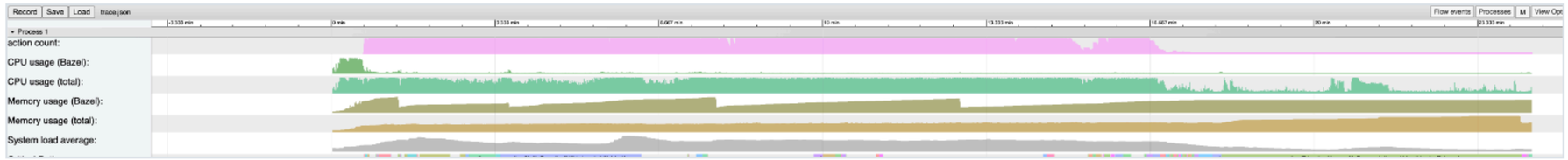
Apart from a few dependency updates for 1.8 compatibility (KSP, compose, ...) there are no major differences to our 1.7 and 1.8 build graphs.

Обновлянка Kotlin

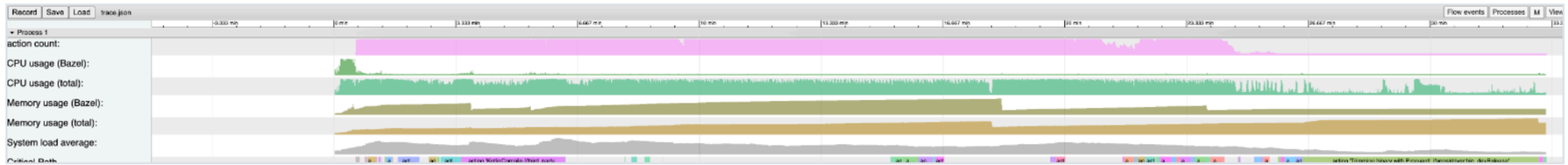
We are working on updating from Kotlin 1.7.10 to 1.8.20 and are seeing 10+ minute regression in clean build times, and a more than 2x increase in memory consumption. We are able to reproduce this on all versions of Kotlin 1.8 (1.80, 1.8.10, 1.8.20).

Profiles:

Kotlin 1.7 profiles where you can see that the "Memory usage total" peaks at around 30gb.



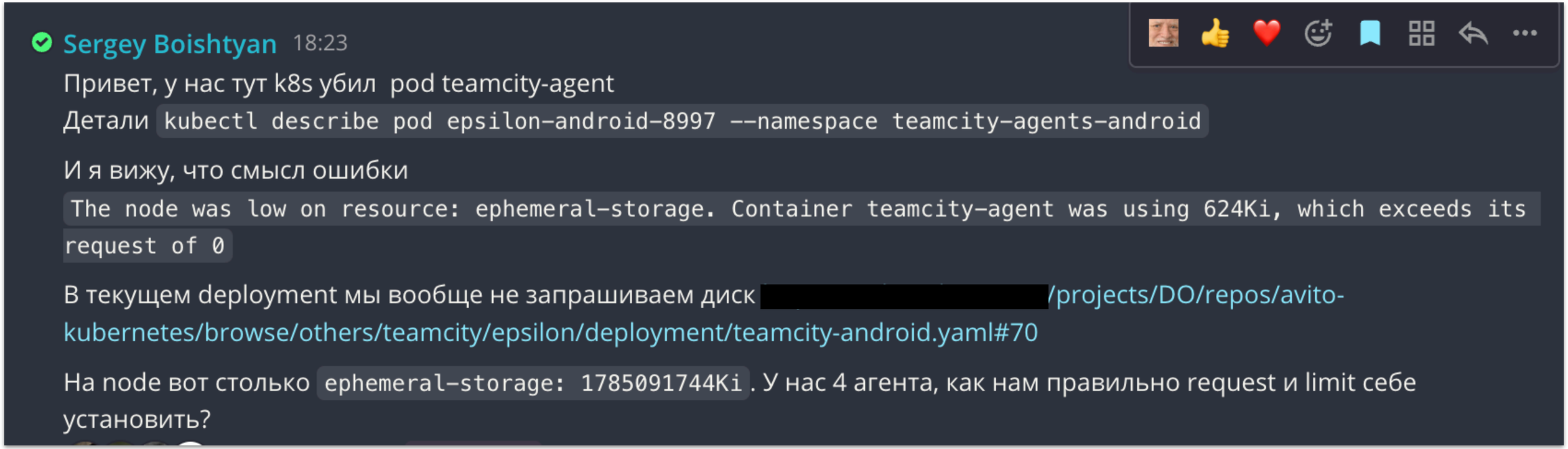
Kotlin 1.8 profiles where you can see that the "Memory usage total" steadily climbs to about 63gb.



Apart from a few dependency updates for 1.8 compatibility (KSP, compose, ...) there are no major differences to our 1.7 and 1.8 build graphs.

Дефицит «железа»

Нехватка диска



OOM

FAILURE: Build completed with 3 failed tasks

1: What went wrong:

```
Execution failed for task ':avito-api:abuse:testReleaseUnitTest'.
       > Process 'Gradle Test Executor 11' finished with non-zero exit value 137
This problem might be caused by incorrect test process configuration.
Please refer to the test execution section in the User Manual at https://docs.gradle.org/7.6.1/userguide/java_testing.html#sec:test_execution
```

Error logs:

No error logs

2: What went wrong:

```
Execution failed for task ':avito-api:auto-catalog:testReleaseUnitTest'.
       > Process 'Gradle Test Executor 2' finished with non-zero exit value 137
This problem might be caused by incorrect test process configuration.
Please refer to the test execution section in the User Manual at https://docs.gradle.org/7.6.1/userguide/java_testing.html#sec:test_execution
```

Error logs:

No error logs

3: What went wrong:

```
Execution failed for task ':avito-api:delivery:testReleaseUnitTest'.
       > Process 'Gradle Test Executor 6' finished with non-zero exit value 137
This problem might be caused by incorrect test process configuration.
Please refer to the test execution section in the User Manual at https://docs.gradle.org/7.6.1/userguide/java_testing.html#sec:test_execution
```

Error logs:

No error logs

Тестовый диабет

Медленные тесты

Speed Android / MBSA-948

Analytics clickstream тесты стали идти 40 минут на CI

Edit | Add comment | Assign | More | Closed

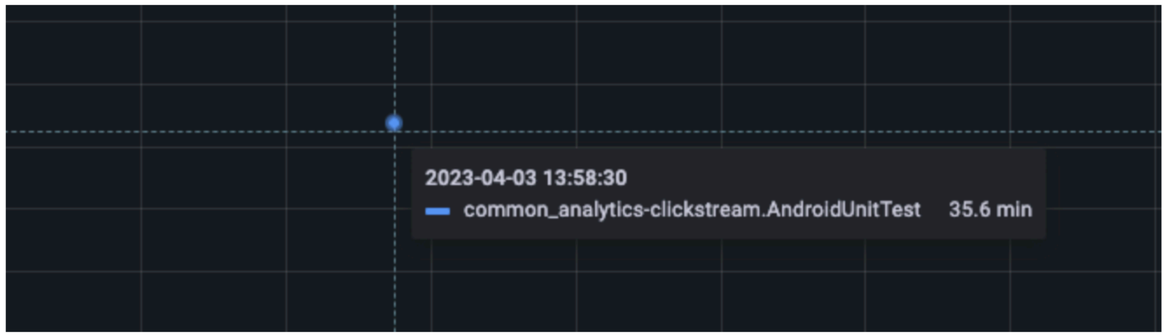
Details

Type:	Bug	Resolution:	Fixed
Priority:	Normal	Fix Version/s:	android_150.5, android_151.0
Component/s:	CI		
Labels:	okr		
Priority for Bug:	P1		

Description

Описание

На дашборде с медленными задачами обнаружили, что тесты в простом модуле идут 40 минут



Тесты зависли

Speed Android / MBSA-952

Зависающая сборка на ПРе

Edit | Add comment | Assign | More | Closed

Details

Type:	Bug	Resolution:	Fixed
Priority:	Normal	Fix Version/s:	None
Component/s:	CI		
Labels:	SLA		
Priority for Bug:	P1		

Danil Garmanov 07 Apr 2023

Чего-то бесконечно таймаутится - ребята из Speed сказали, что как раз пытаются с такой же проблемой разобраться. Жду.

Reply

```
160   -     latch.await()
    169 +   latch.await(10, TimeUnit.SECONDS)
161 170     }
```

Sergey Boishtyan 07 Apr 2023

Из-за того что тут запустил работу на IO потоке она успешно упала с ошибкой и тест не дождался

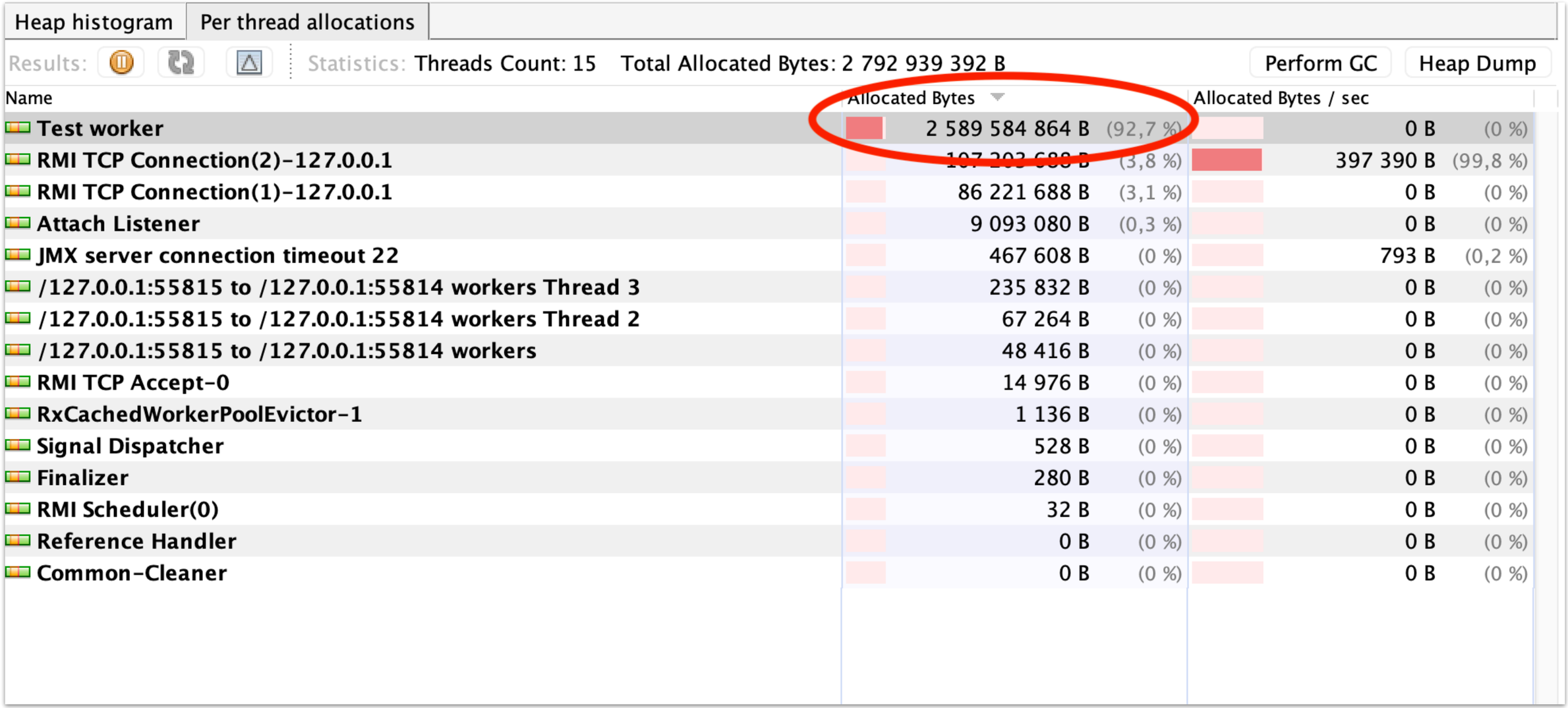
Reply · Edit

Sergey Boishtyan 07 Apr 2023

TLDR если бы ты локально запустил, то увидел что у тебя тесты висят.
Нам прилошлось на горячую агенты дебажить чтобы увидеть, что тестовый процесс висит)))

Reply · Edit

Тяжелые Robolectric тесты



Долгие Robolectric тесты

«Нет времени объяснять»

Итого выиграли 700 секунд суммарно за счет настройки

maxParallel и forkEvery

На текущей конфигурации:
maxParallel(1)
forkEvery(Long.Max_value)
Медиана сборки - 3344 сек
Суммарное время по всем таскам - 39750 сек
Суммарное время по тест таскам - 16463 сек

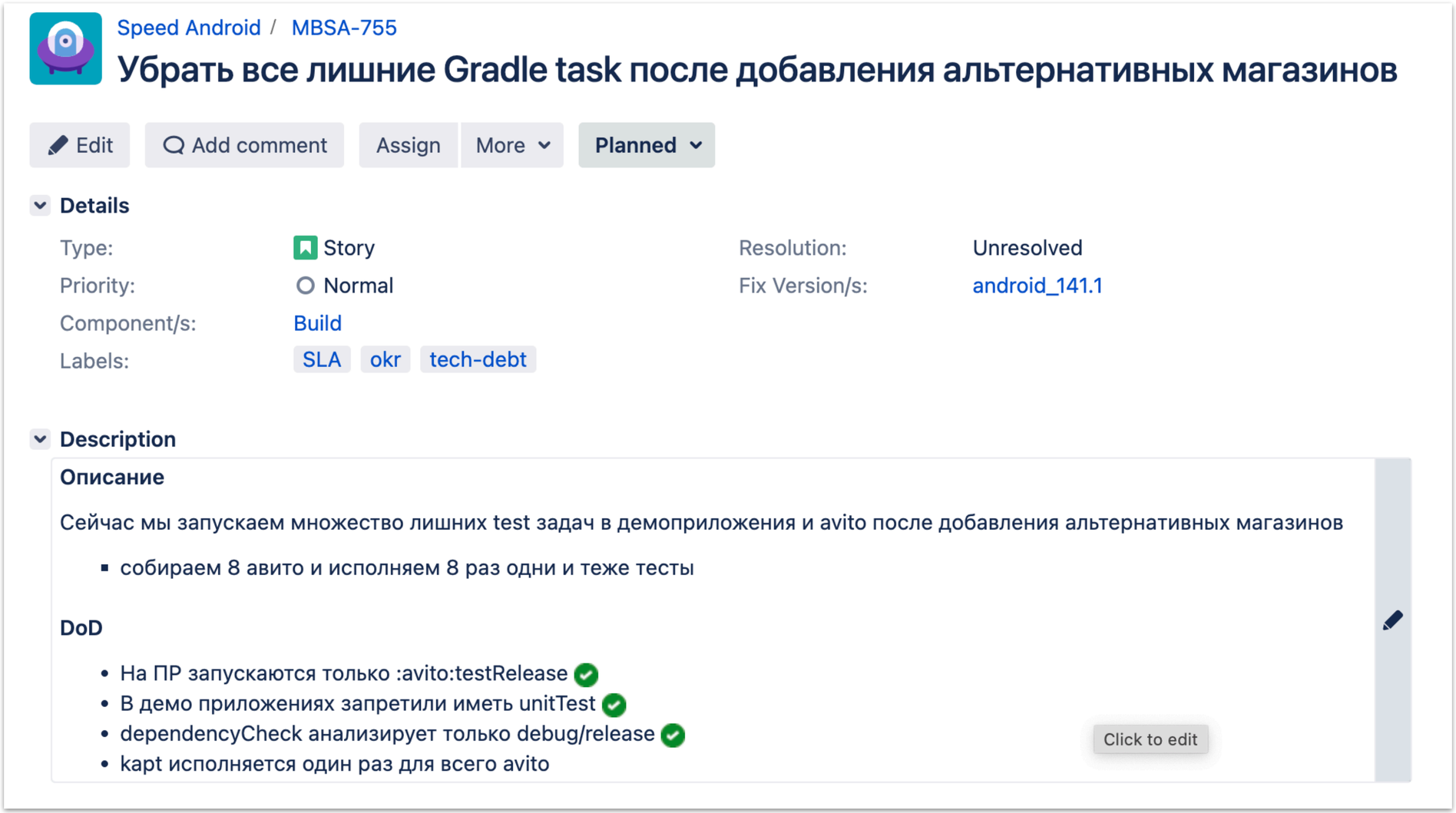
На конфигурации:
maxParallel(1)
forkEvery(25)
Медиана сборки - 2 945 сек
Суммарное время по всем таскам - 36104 сек
Суммарное время по тест таскам - 15787 сек

Итог:
На самих тестах выграли всего ~ 700 сек суммарно. Если разделить на 15 потоков то на 46 сек в каждом. Линейно 45 секунд
Сама сборка ускорилась на ~ 400 сек. Примерно 6.6 мин
По всем таскам суммарно выйграли ~ 3600 сек. То есть примерно 60 мин

Почему так. Моя изначальная гипотеза была что утечки в тестах сильно нагружают и не дают память другим задачам, идет борьба за ререурсы. Теперь остальные задачи работают эффективнее, а тесты практически так же.

Кодовая диарея

Новые buildVariant



Gradle задачи разных build variant вынуждены делать одну и ту же работу

Новые build variant

Представим, у вас есть один release variant в модуле app

Новые build variant

Вы добавили новые 5 variant:

Huawei

RuStore

Xiaomi

Samsung

Debug

Поздравляю
Теперь у вас в конце сборки в 6 раз больше задач

2

«Градусники»
Метрики и дашборды наших SLA

За чем следить?

- Заключаете SLA
- Пытаетесь их соблюдать
- Метрики дают понимание, что делать

Наши SLA

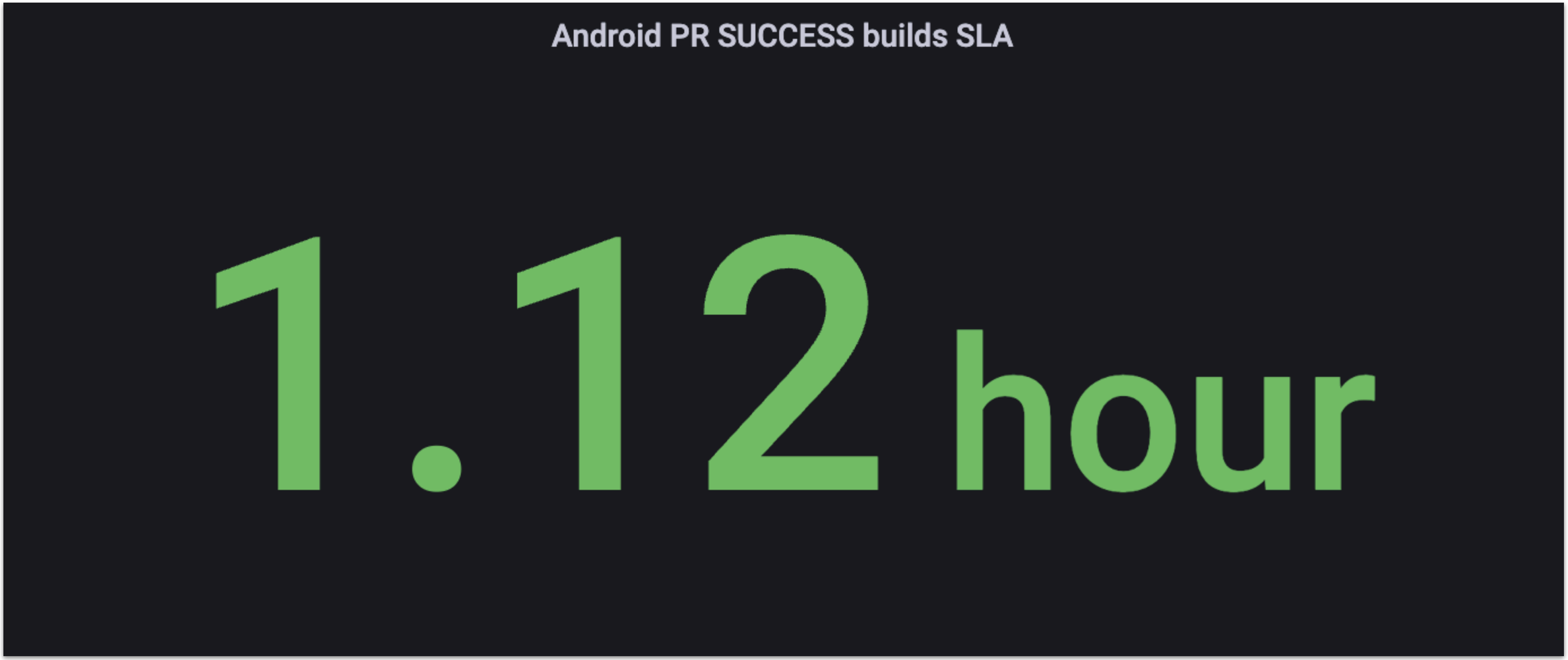
Мы отвечаем, чтобы сборки в CI были:

- Быстрее
- Стабильные

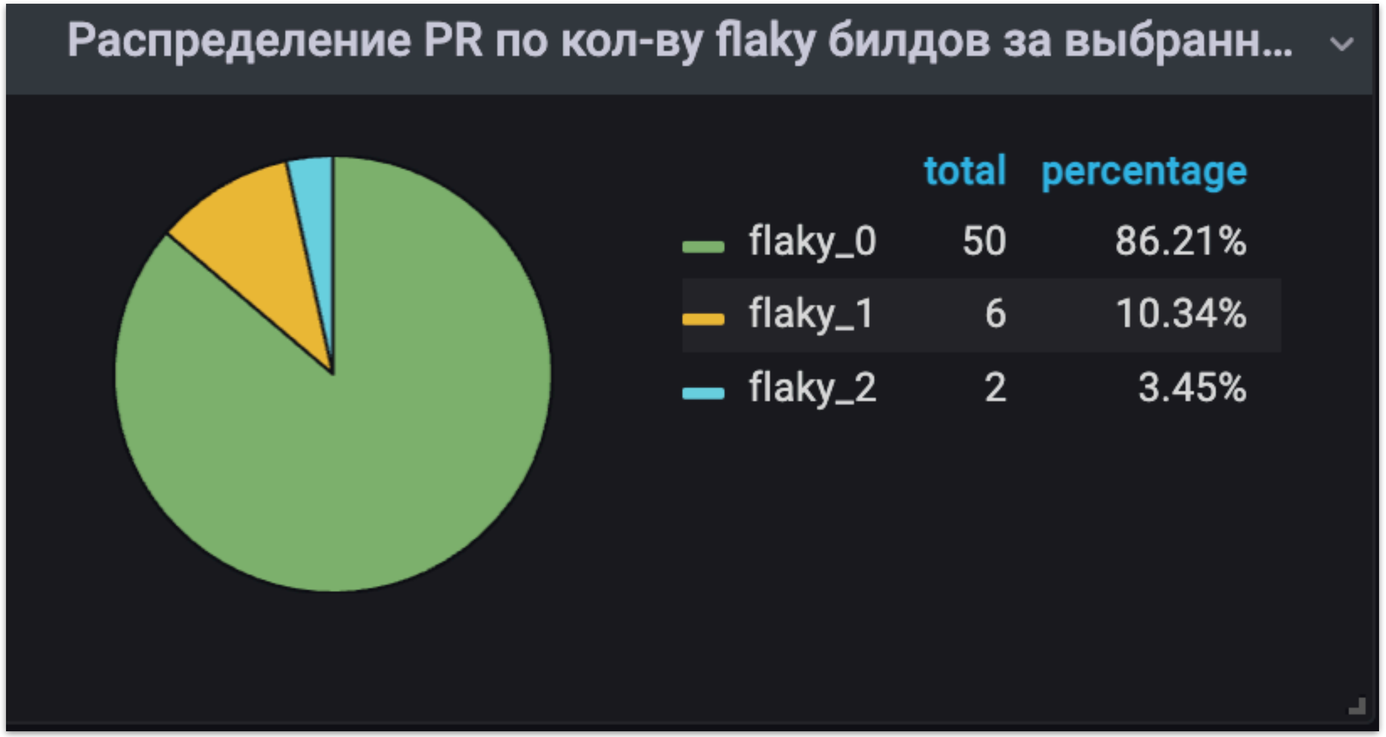
Наши SLA

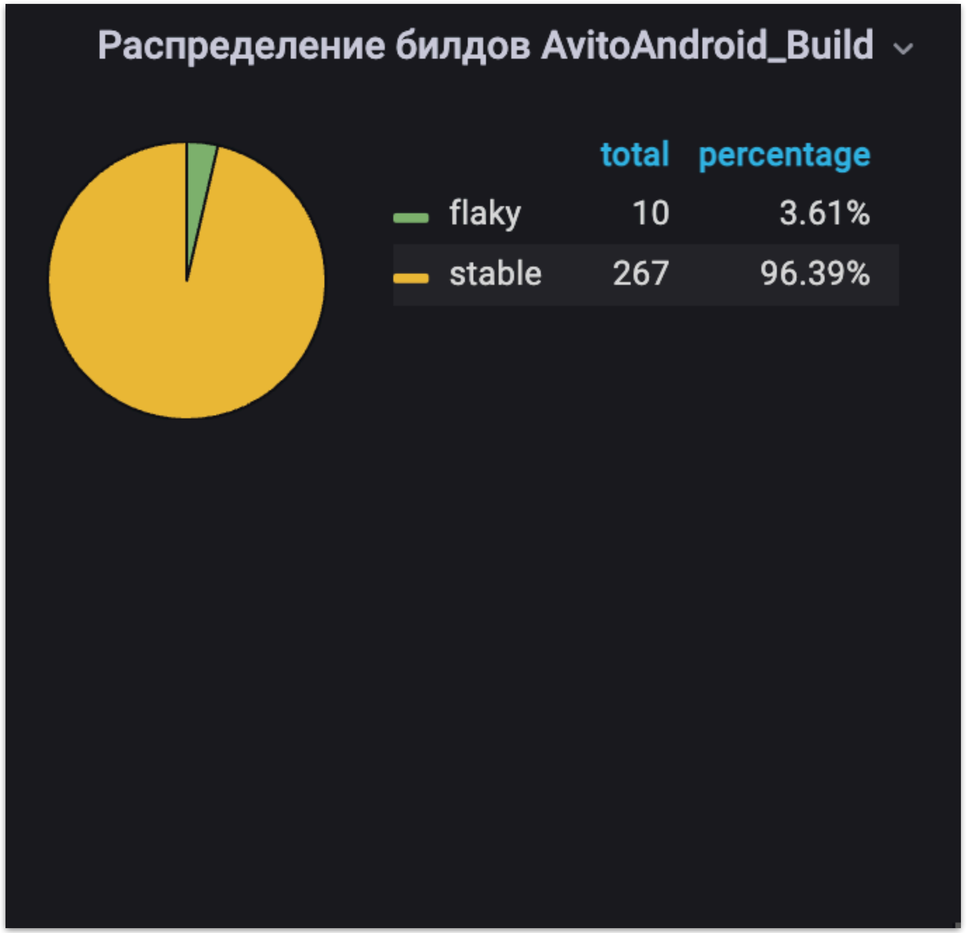
- Время 90%% успешных сборок на PR — **Скорость**
- Кол-во PR с Flaky сборками — **Стабильность**

Время 90%% успешных сборок



PR с Flaky сборками





Чем проще дашборд, тем легче понять, соблюдаются ли SLA

На этих простых дашбордах сразу видно, если все хорошо

Например

- Всего один показатель
- Есть явная грань 37, когда все плохо
- Легко использовать
- Быстрая обратная связь

Но если все плохо, то ничего не понятно

Например

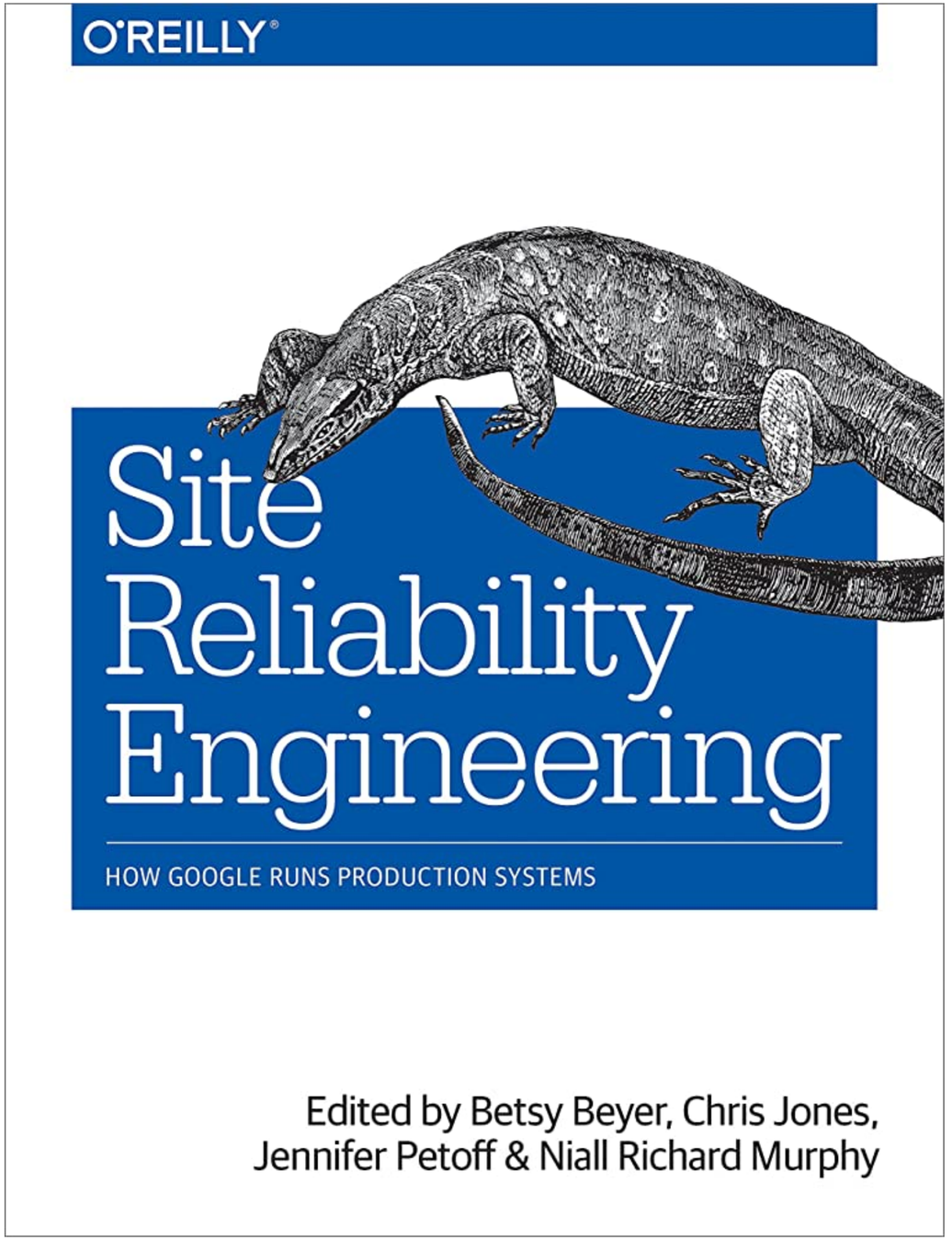
- Если на градуснике 38 — мы заболели
- Нужно лечиться
  - Можно наугад
  - Можно делать дополнительные обследования

Нам нужны дополнительные «анализы», чтобы определить проблему

Какие дополнительные дашборды нам нужны?

«Золотые сигналы»

- Latency
- Traffic
- Errors
- Saturation



THE FOUR Golden Signals OF MONITORING

"If you can only measure four metrics... focus on these four."

LATENCY

Amount of time to service a request

Slow feedback is a symptom of degraded performance ~somewhere~

TRAFFIC

Number of
- HTTP requests?
- sessions?
- transactions?

per second

ERRORS

Rate of failed requests

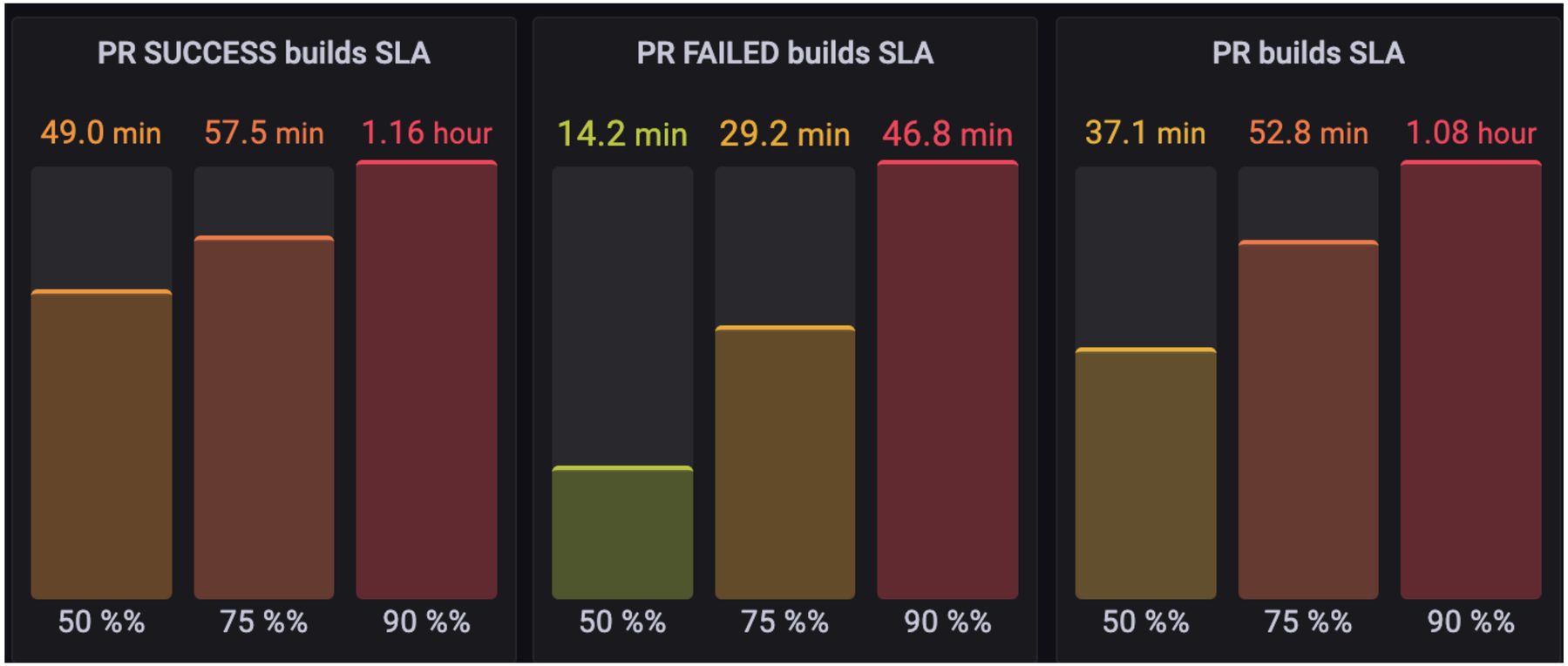
SATURATION

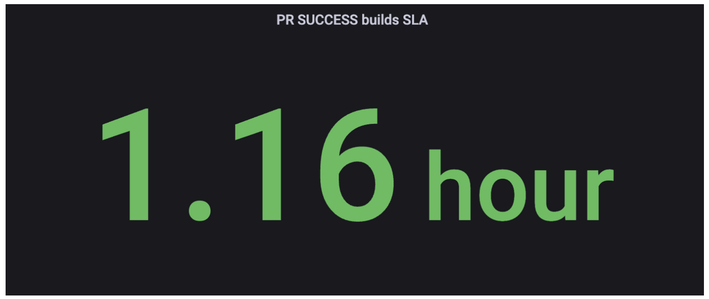
How close are you to 100% utilization? Many services degrade in performance as they approach saturation!

From Chapter 6, "Monitoring Distributed Systems" in Site Reliability Engineering

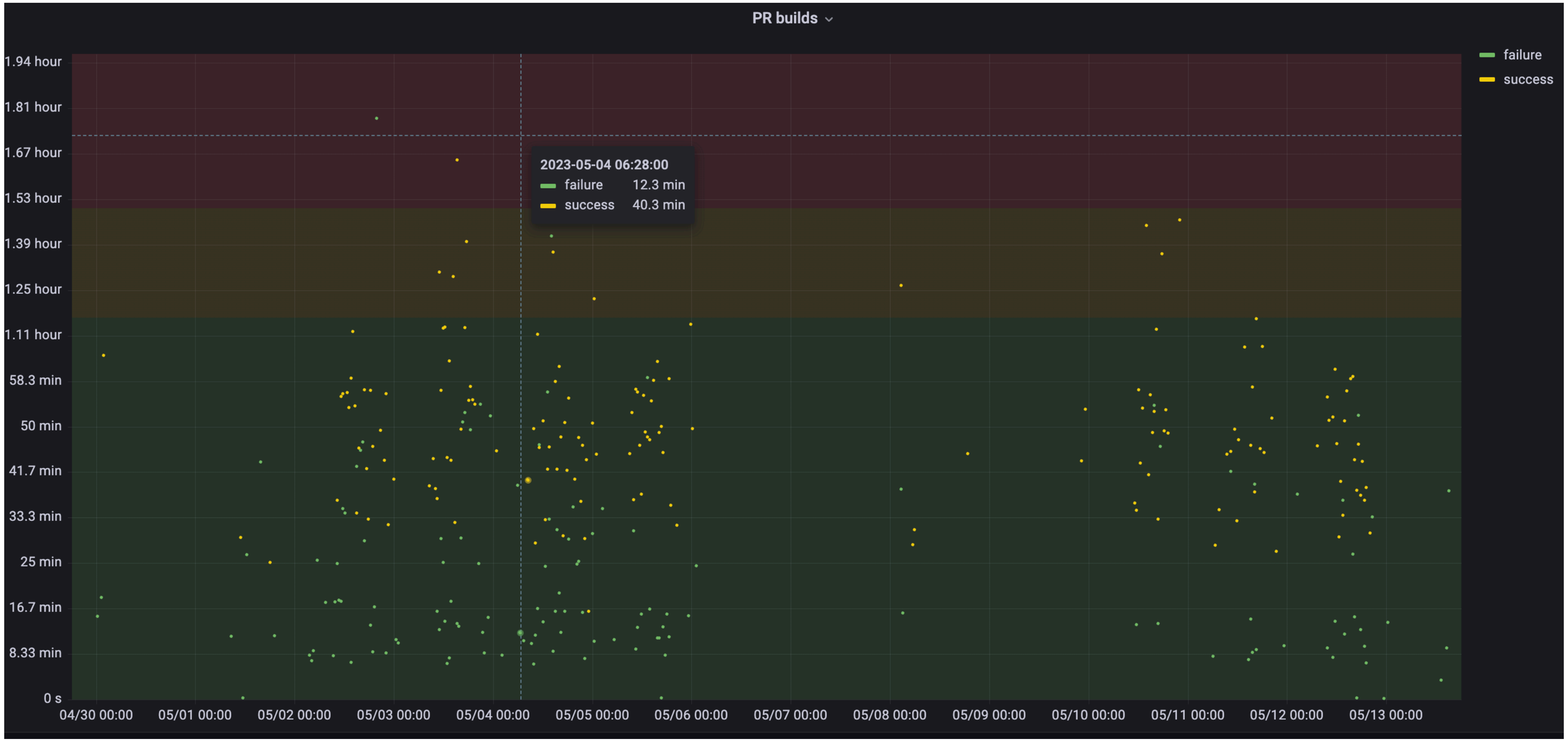
Золотые сигналы нашей сборки

Latency

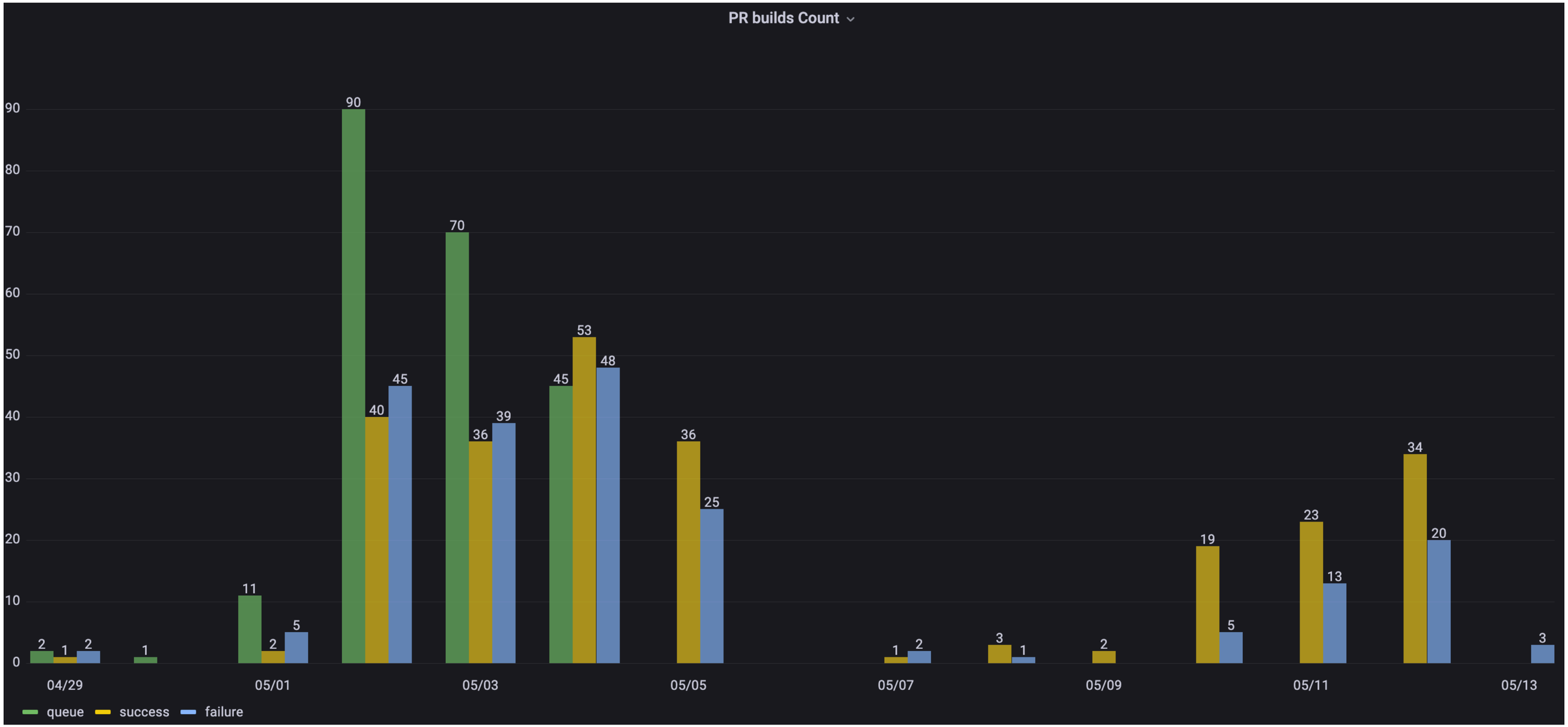

PR SUCCESS builds SLA
49.0 min
57.5 min
1.16 hour
50 %%
75 %%
90 %%
PR FAILED builds SLA
14.2 min
29.2 min
46.8 min
50 %%
75 %%
90 %%
PR builds SLA
37.1 min
52.8 min
1.08 hour
50 %%
75 %%
90 %%


PR SUCCESS builds SLA
1.16 hour

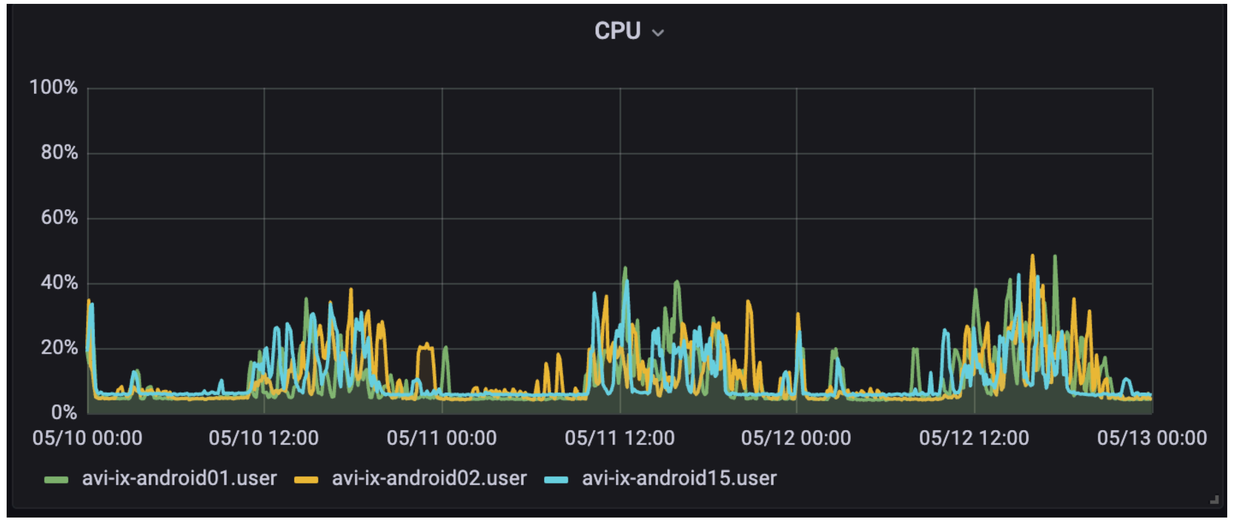
Traffic

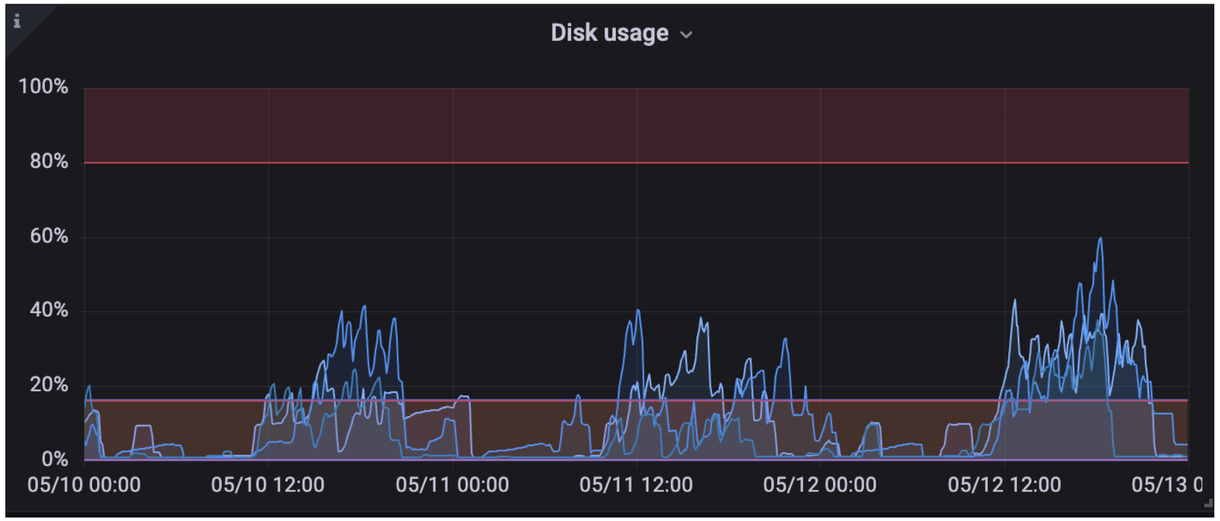

PR builds
failure
success
1.94 hour
1.81 hour
1.67 hour
1.53 hour
1.39 hour
1.25 hour
1.11 hour
58.3 min
50 min
41.7 min
33.3 min
25 min
16.7 min
8.33 min
0 s
2023-05-04 06:28:00
failure 12.3 min
success 40.3 min
04/30 00:00
05/01 00:00
05/02 00:00
05/03 00:00
05/04 00:00
05/05 00:00
05/06 00:00
05/07 00:00
05/08 00:00
05/09 00:00
05/10 00:00
05/11 00:00
05/12 00:00
05/13 00:00

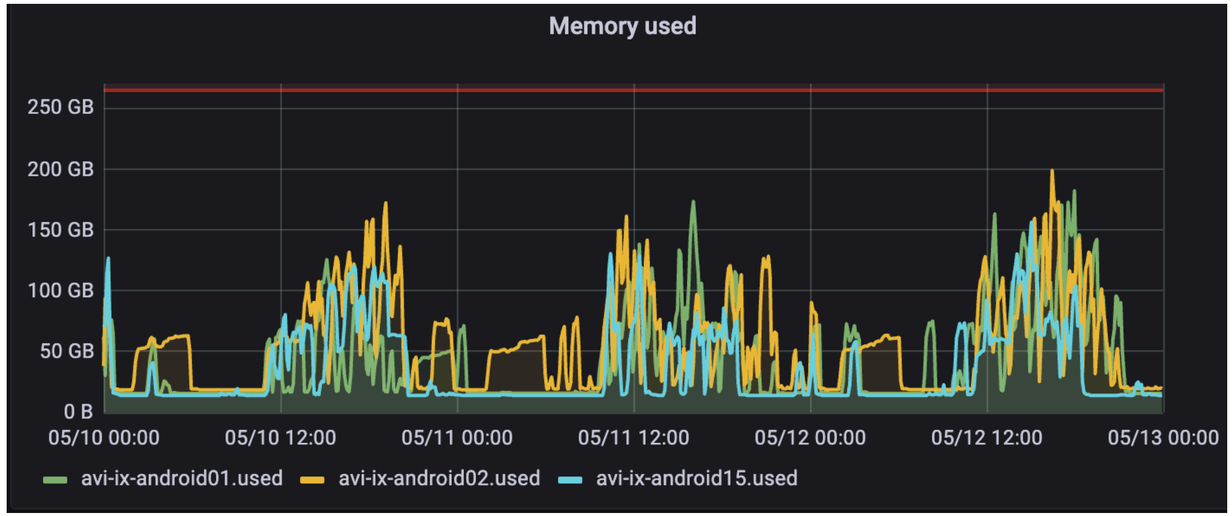
Traffic

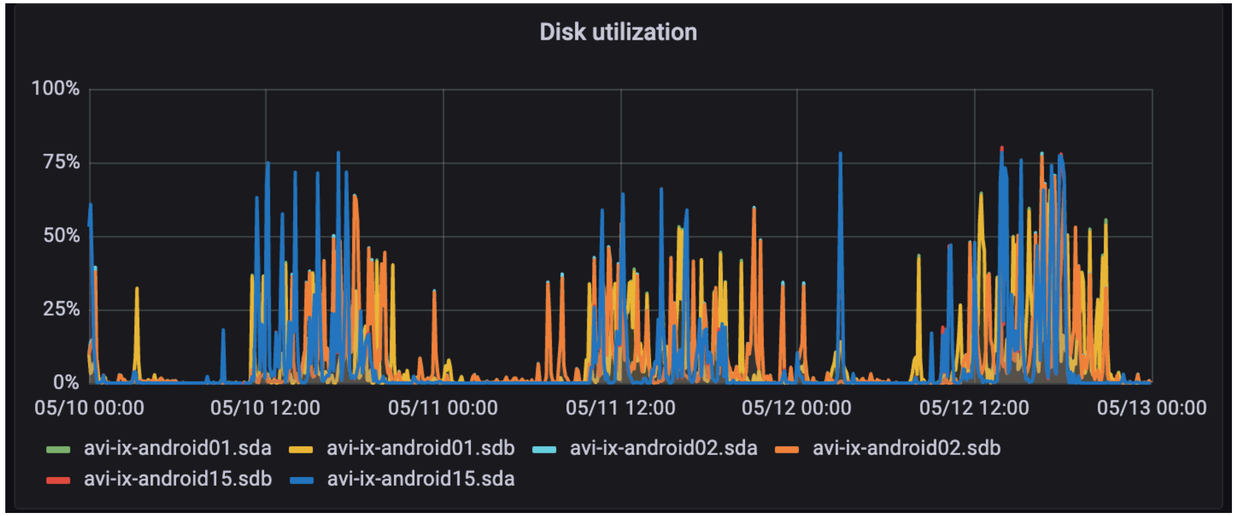

PR builds Count
90
80
70
60
50
40
30
20
10
0
2
1
2
1
11
2
5
90
40
45
70
36
39
45
53
48
36
25
1
2
3
1
2
19
5
23
13
34
20
3
04/29
05/01
05/03
05/05
05/07
05/09
05/11
05/13
queue
success
failure

Saturation

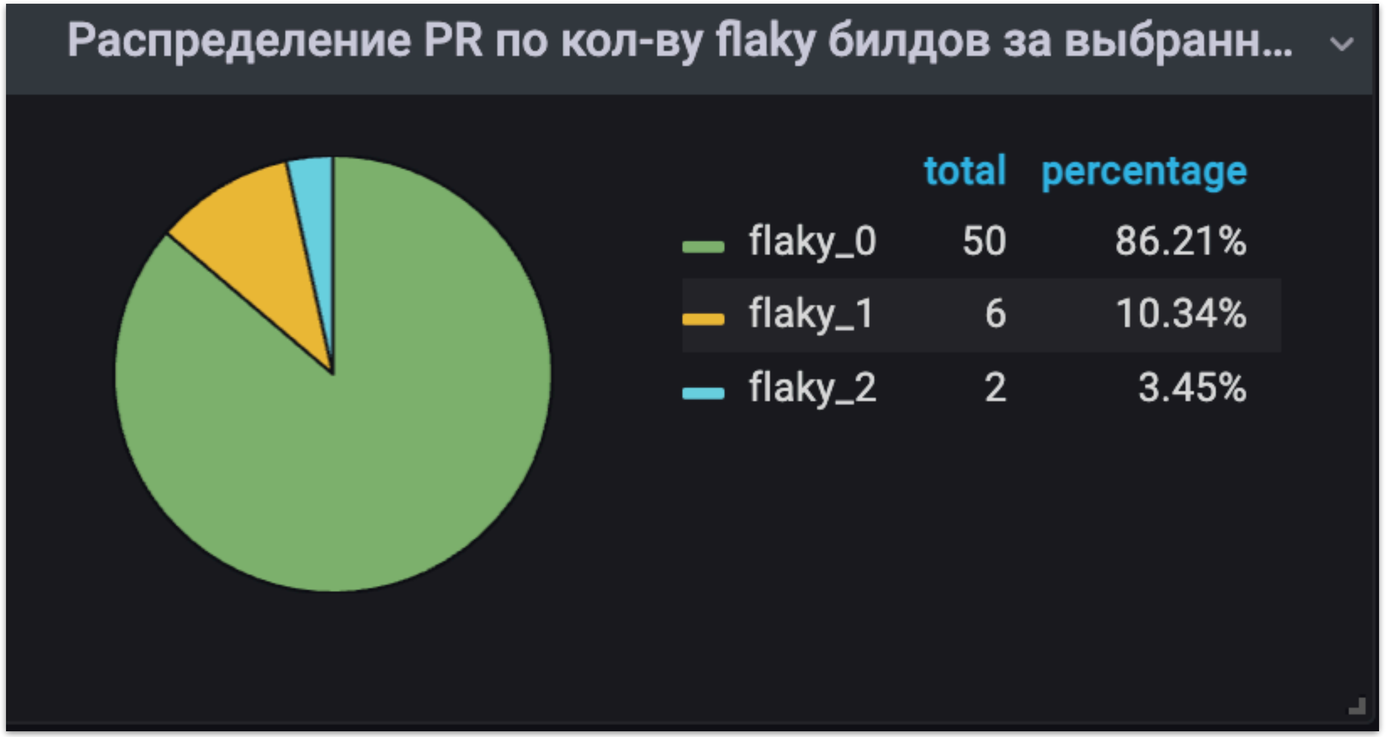

CPU
100%
80%
60%
40%
20%
0%
05/10 00:00
05/10 12:00
05/11 00:00
05/11 12:00
05/12 00:00
05/12 12:00
05/13 00:00
avi-ix-android01.user
avi-ix-android02.user
avi-ix-android15.user

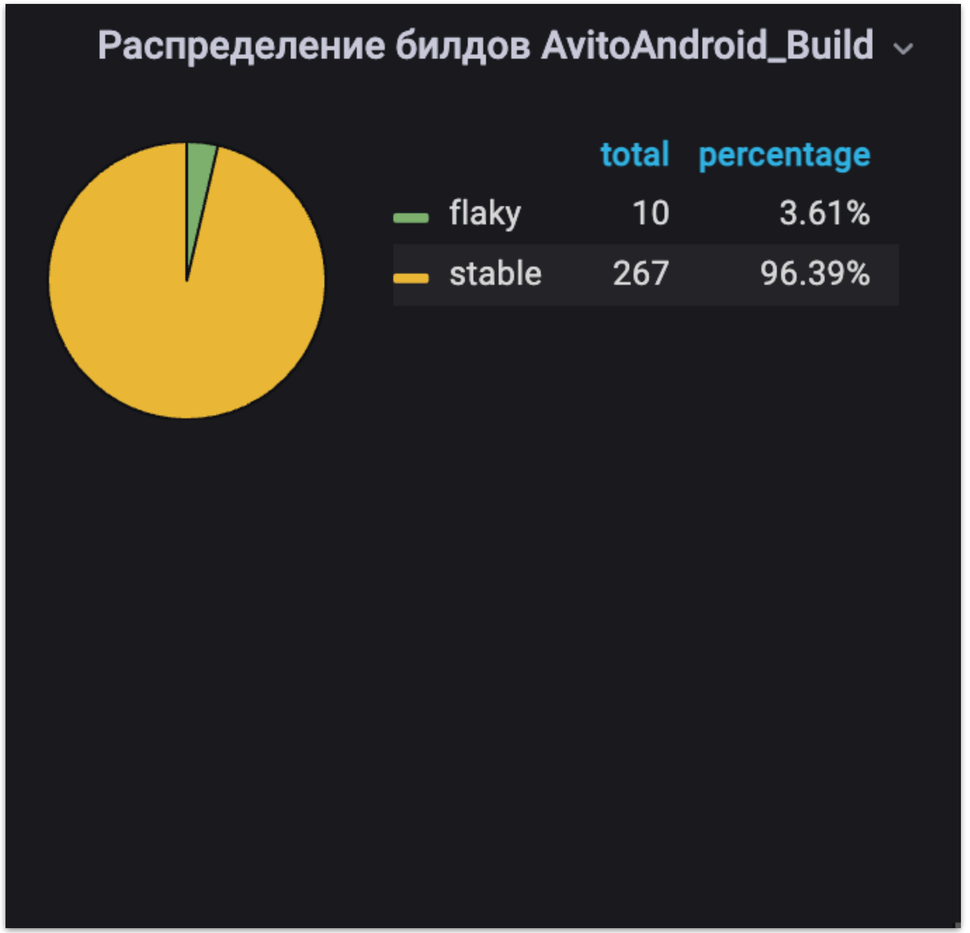

Disk usage
100%
80%
60%
40%
20%
0%
05/10 00:00
05/10 12:00
05/11 00:00
05/11 12:00
05/12 00:00
05/12 12:00


Memory used
250 GB
200 GB
150 GB
100 GB
50 GB
0 B
05/10 00:00
05/10 12:00
05/11 00:00
05/11 12:00
05/12 00:00
05/12 12:00
05/13 00:00
avi-ix-android01.used
avi-ix-android02.used
avi-ix-android15.used


Disk utilization
100%
75%
50%
25%
0%
05/10 00:00
05/10 12:00
05/11 00:00
05/11 12:00
05/12 00:00
05/12 12:00
05/13 00:00
avi-ix-android01.sda
avi-ix-android01.sdb
avi-ix-android02.sda
avi-ix-android02.sdb
avi-ix-android15.sdb
avi-ix-android15.sda

Errors


Распределение PR по кол-ву flaky билдов за выбранн...
total
percentage
flaky_0 50 86.21%
flaky_1 6 10.34%
flaky_2 2 3.45%


Распределение билдов AvitoAndroid_Build
total
percentage
flaky 10 3.61%
stable 267 96.39%

Это БАЗА

Они помогут только при серьезных проблемах

Откуда наше вдохновение для метрик и дашбордов?

Наши метрики, как право в США, прецедентные

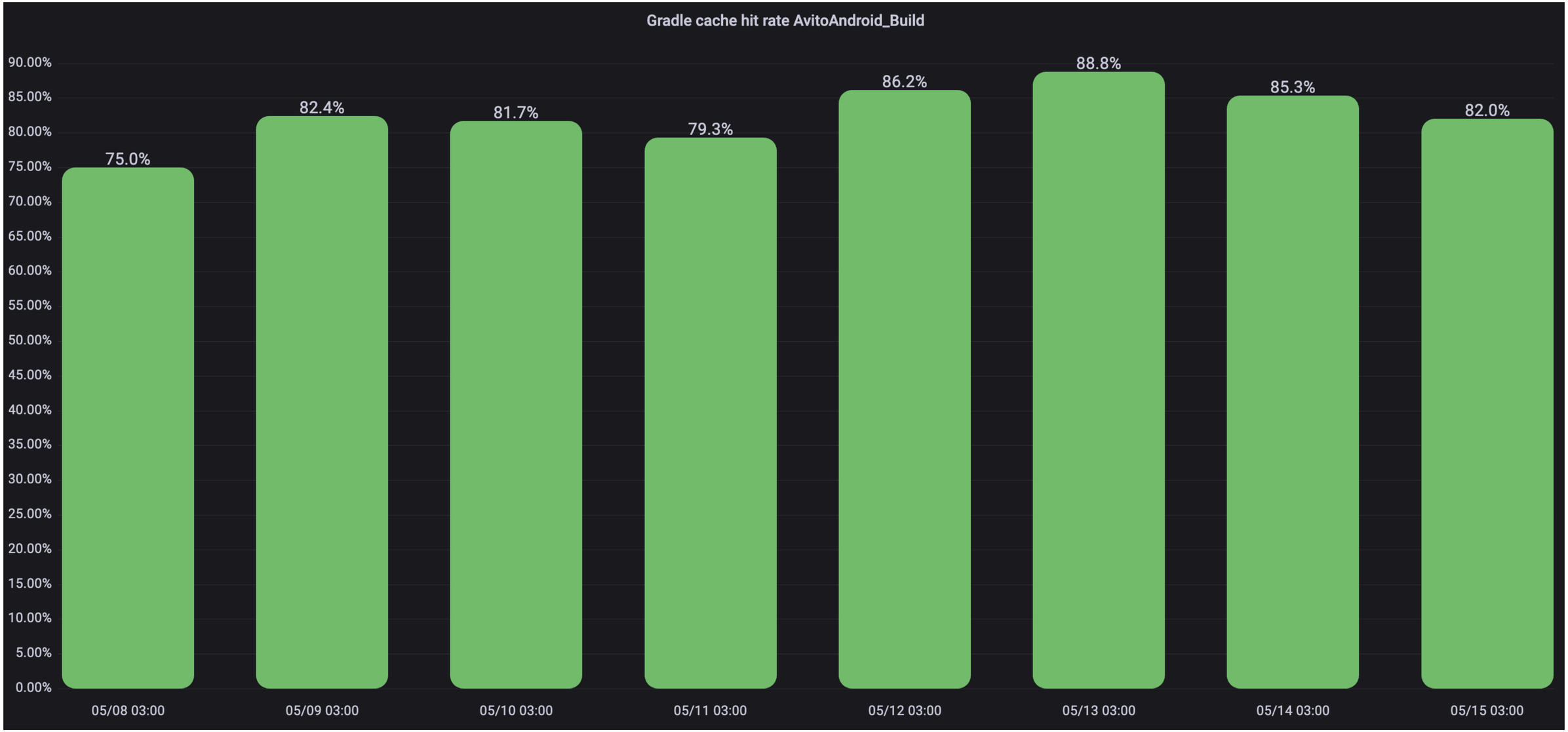
Вопросы для создания метрик

- Что больше всего увеличивает время сборки?
- Что потребляет больше всего ресурсов?
- Из чего состоит время сборки? Сколько занимает каждая из частей?

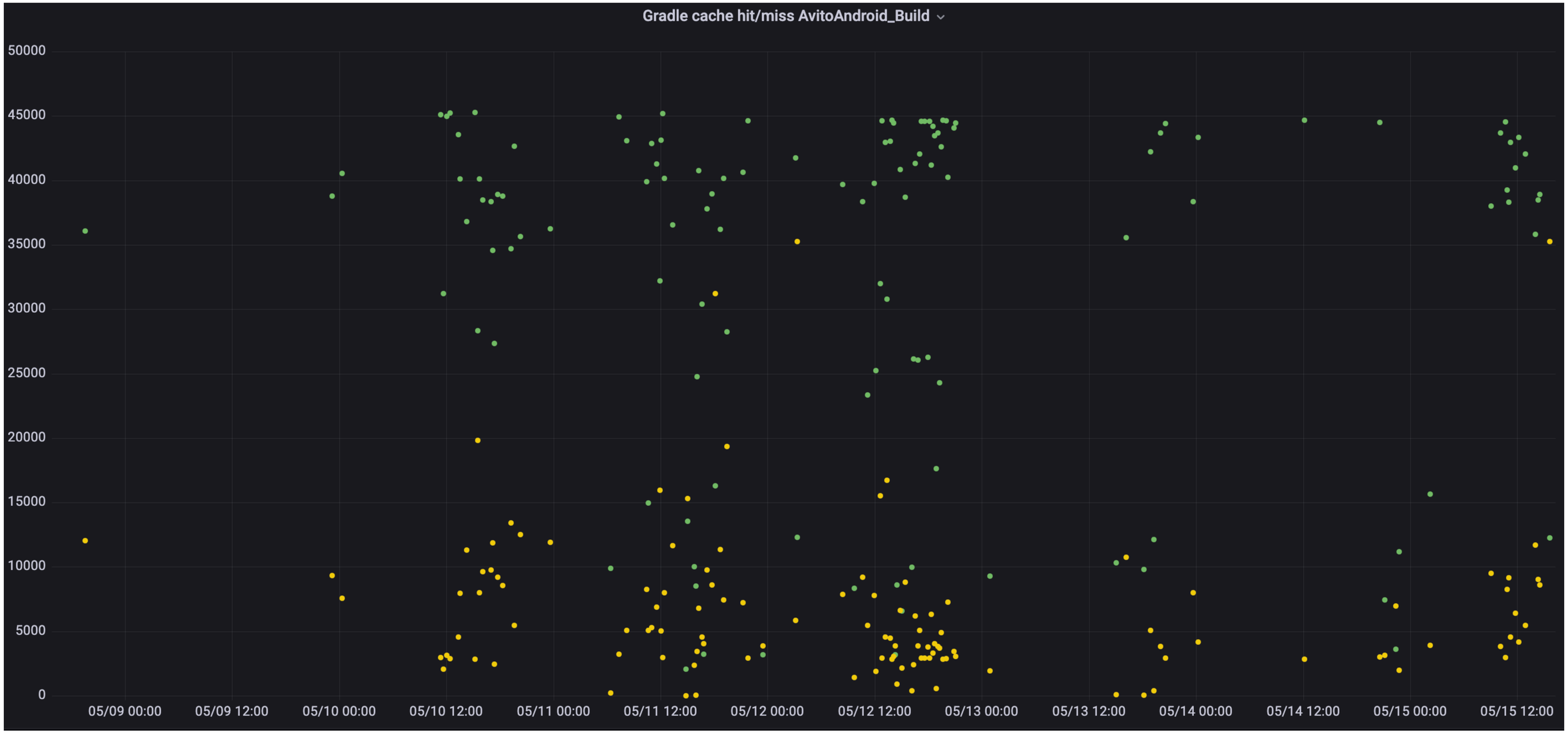
Примеры наших дашбордов

Gradle cache метрики

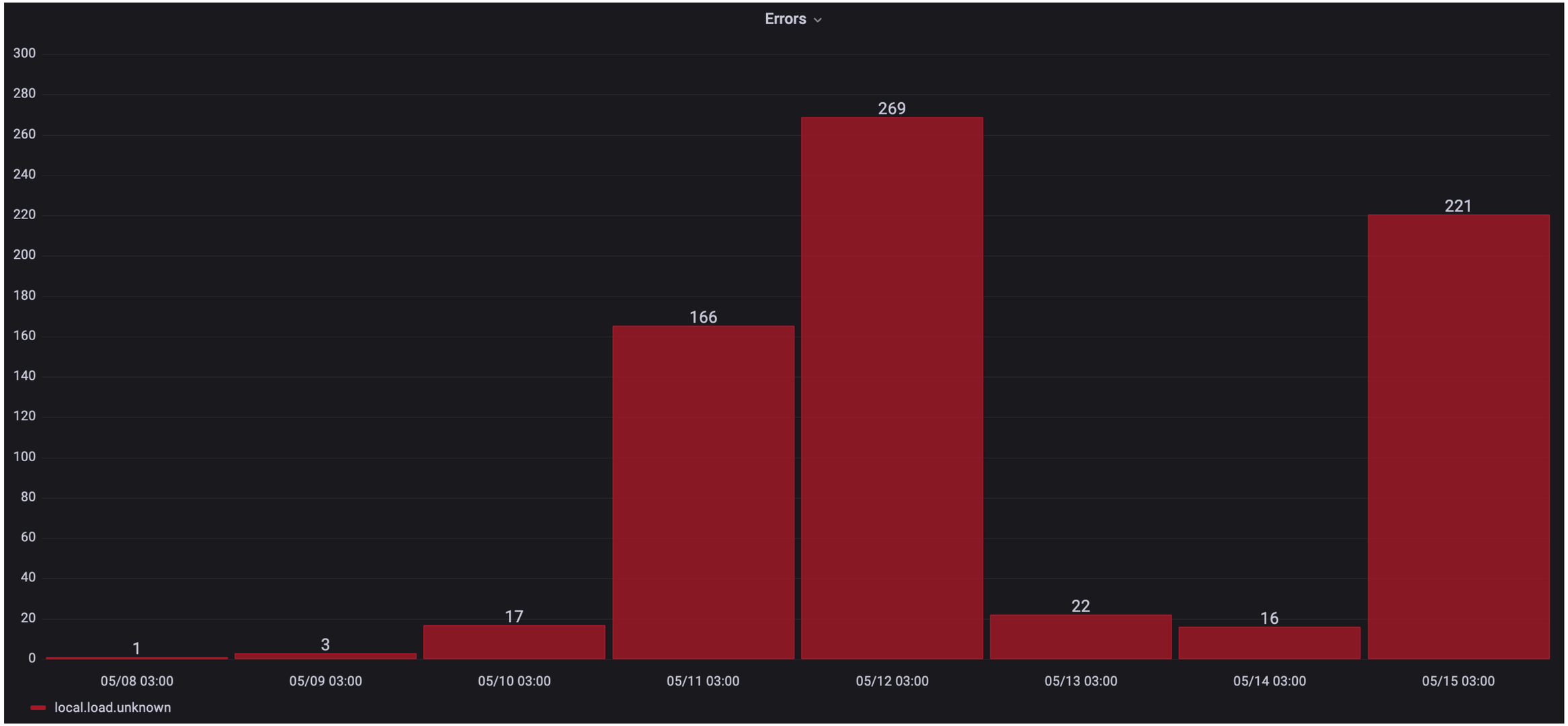
Gradle cache hit rate



Gradle cache hit/miss

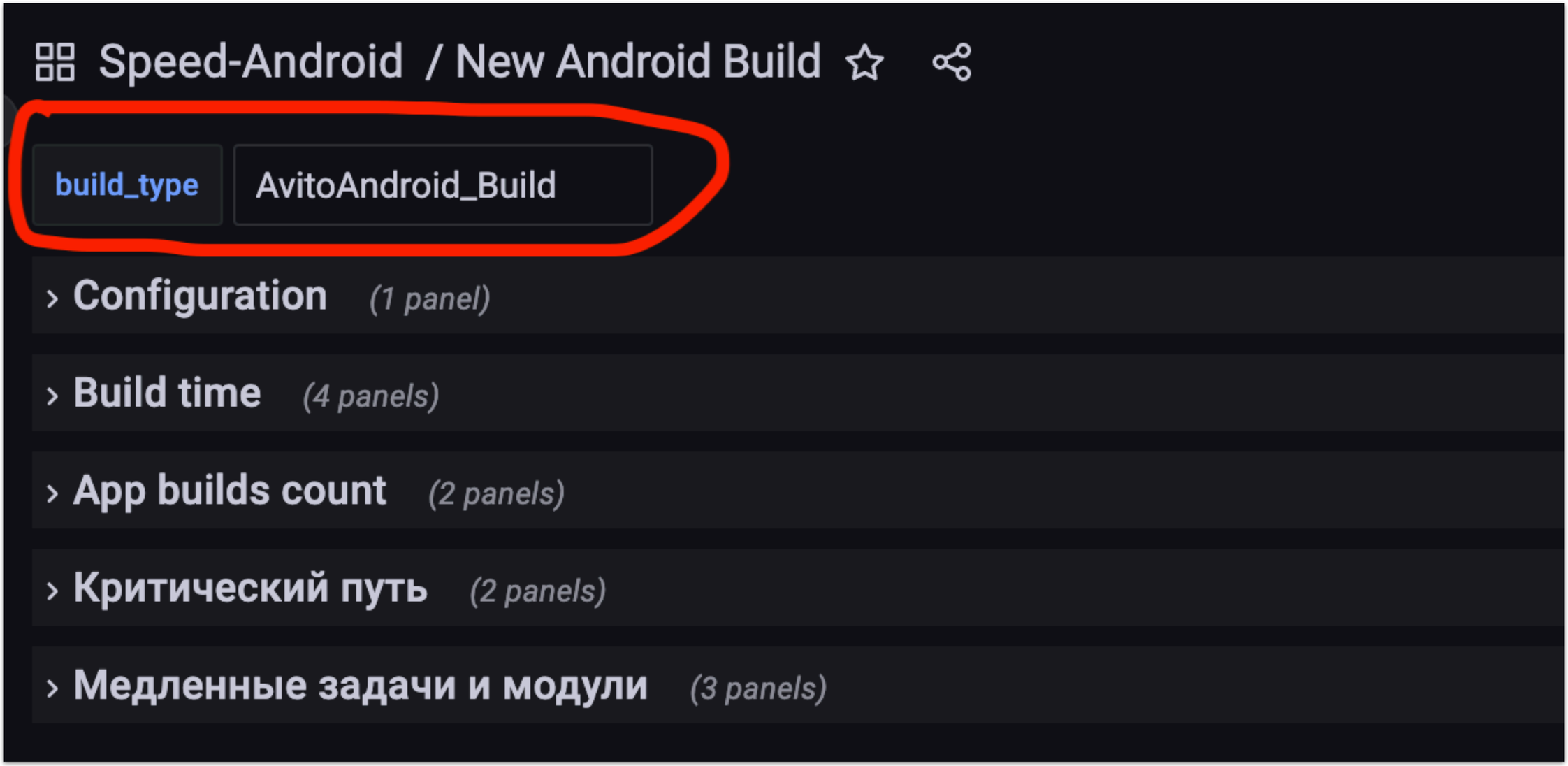


Gradle cache errors

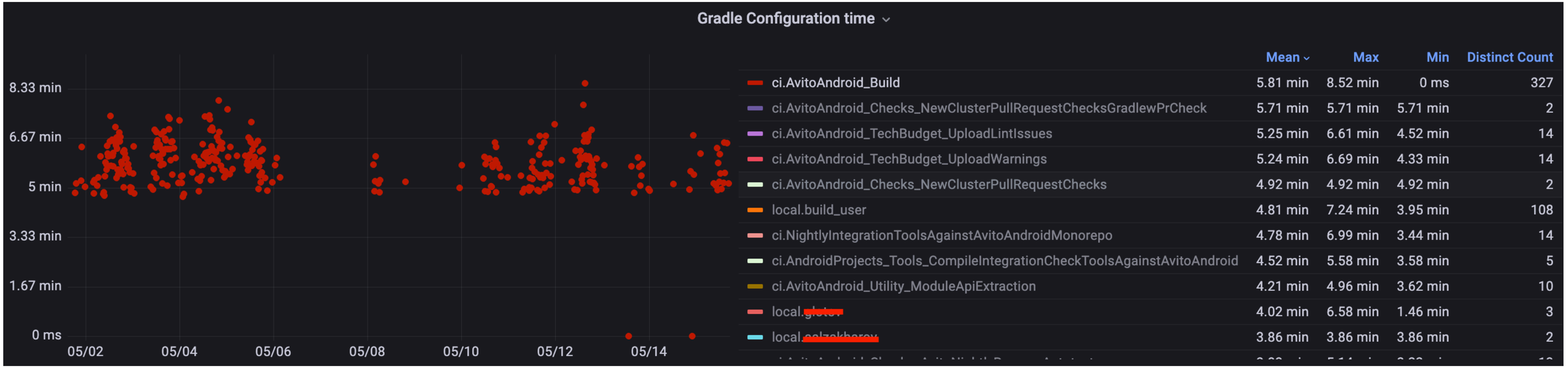


Gradle Build Internals метрики

Gradle build internals

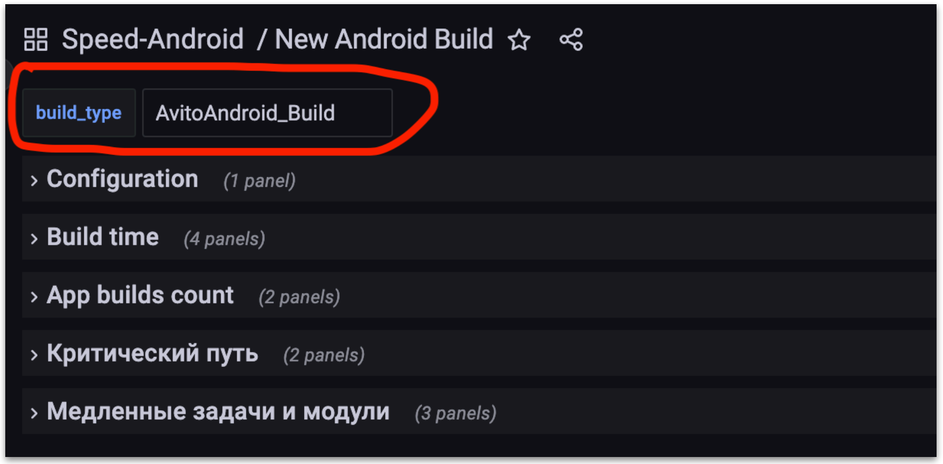


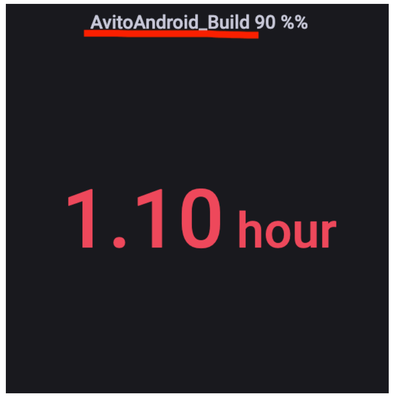
Configuration

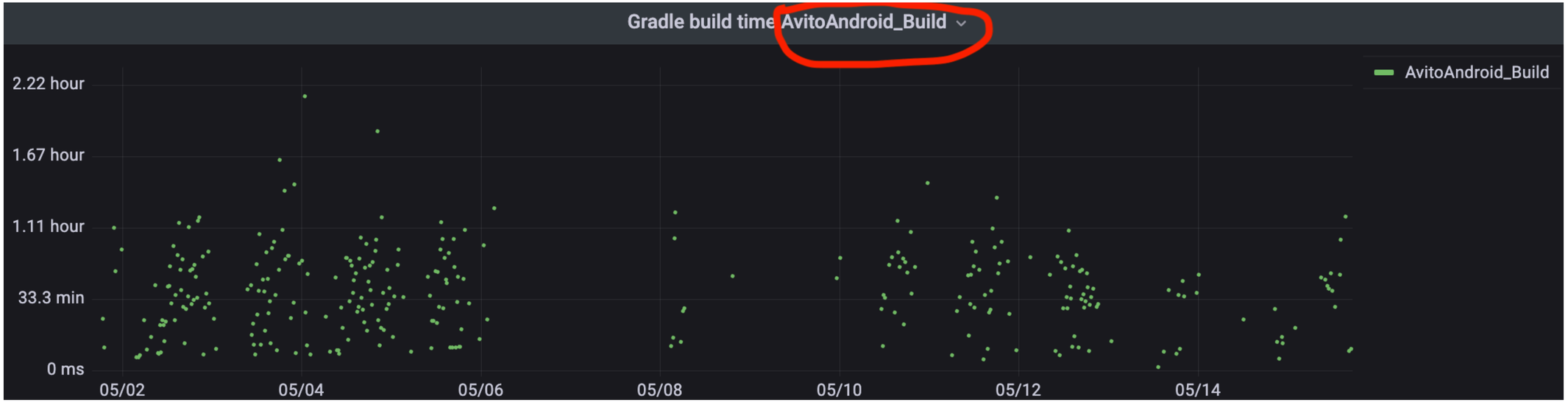


	Mean	Max	Min	Distinct Count
ci.AvitoAndroid_Build	5.81 min	8.52 min	0 ms	327
ci.AvitoAndroid_Checks_NewClusterPullRequestChecksGradlewPrCheck	5.71 min	5.71 min	5.71 min	2
ci.AvitoAndroid_TechBudget_UploadLintIssues	5.25 min	6.61 min	4.52 min	14
ci.AvitoAndroid_TechBudget_UploadWarnings	5.24 min	6.69 min	4.33 min	14
ci.AvitoAndroid_Checks_NewClusterPullRequestChecks	4.92 min	4.92 min	4.92 min	2
local.build_user	4.81 min	7.24 min	3.95 min	108
ci.NightlyIntegrationToolsAgainstAvitoAndroidMonorepo	4.78 min	6.99 min	3.44 min	14
ci.AndroidProjects_Tools_CompileIntegrationCheckToolsAgainstAvitoAndroid	4.52 min	5.58 min	3.58 min	5
ci.AvitoAndroid_Utility_ModuleApiExtraction	4.21 min	4.96 min	3.62 min	10
local.[redacted]	4.02 min	6.58 min	1.46 min	3
local.[redacted]	3.86 min	3.86 min	3.86 min	2

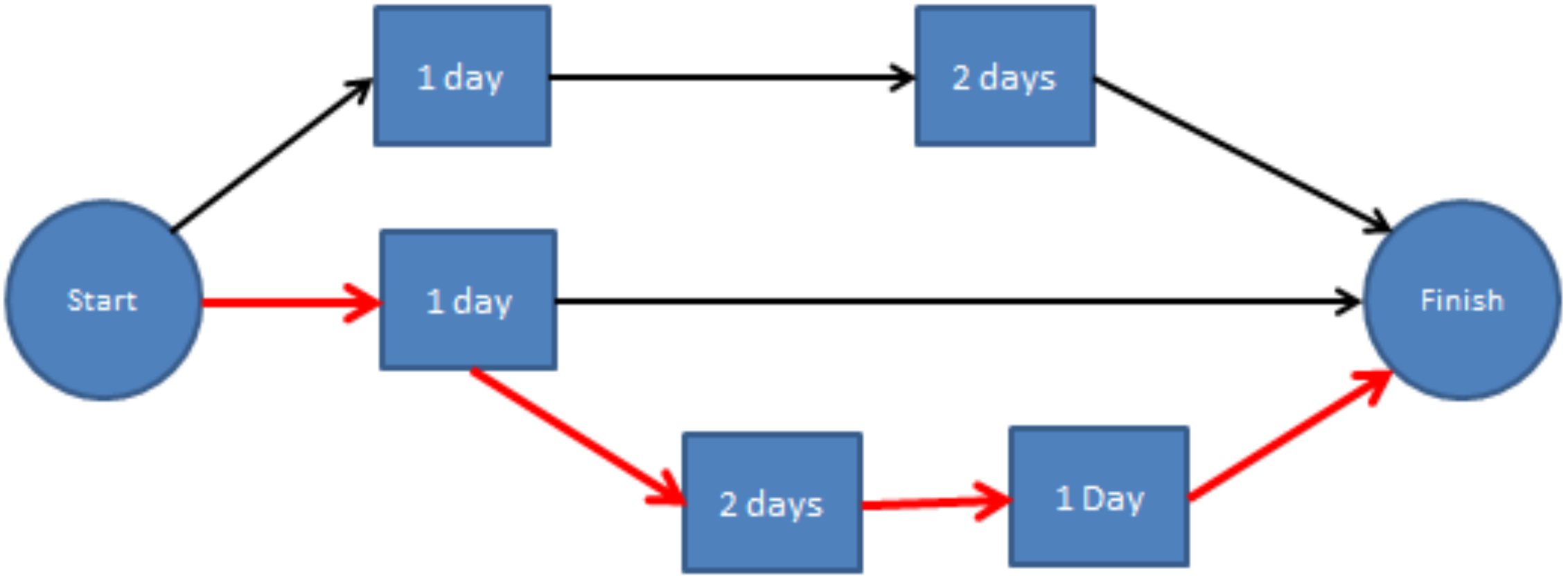
Build time


Speed-Android / New Android Build
build_type
AvitoAndroid_Build
Configuration (1 panel)
Build time (4 panels)
App builds count (2 panels)
Критический путь (2 panels)
Медленные задачи и модули (3 panels)


AvitoAndroid_Build 90 %%
1.10 hour

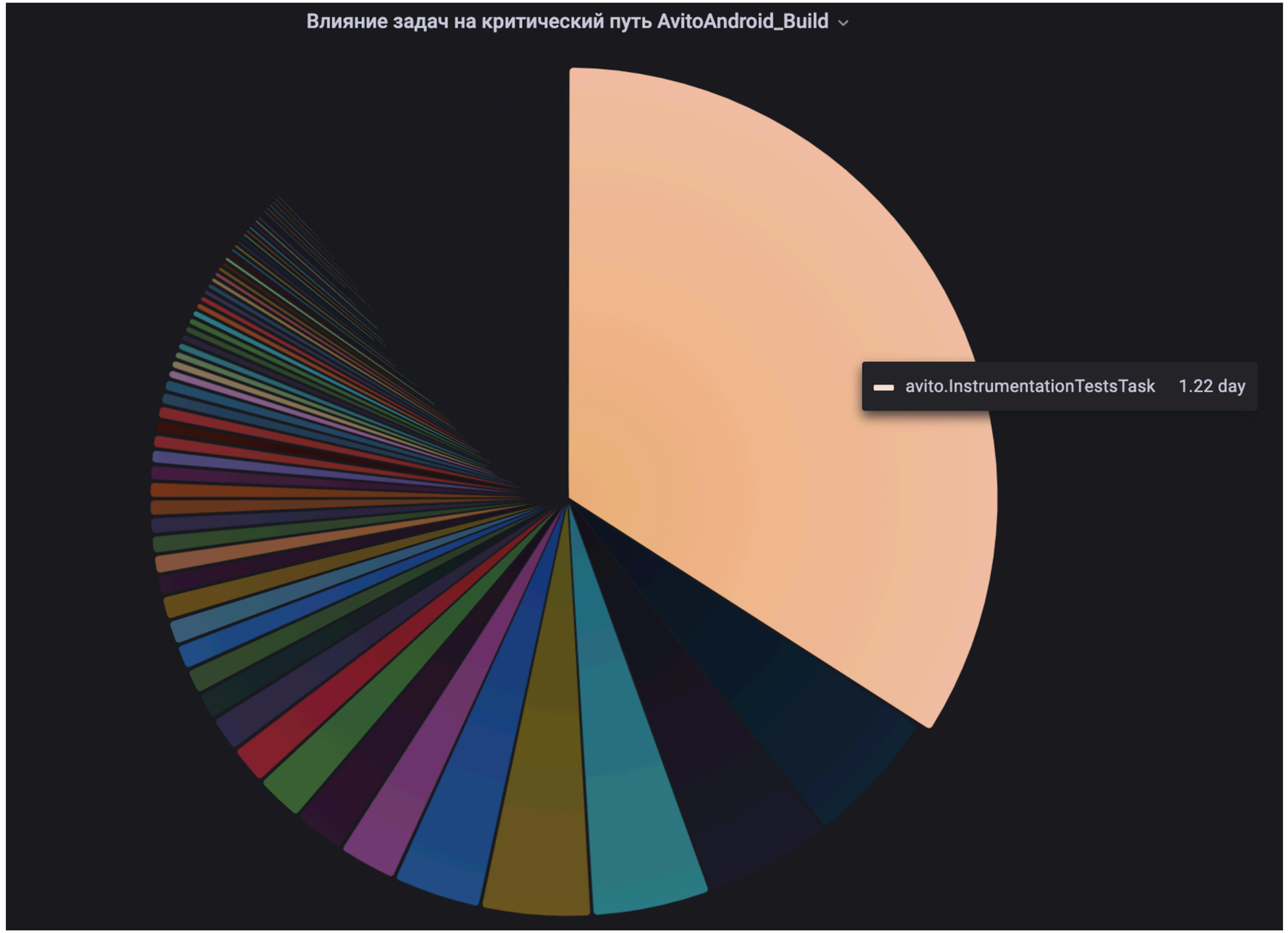

Gradle build time AvitoAndroid_Build
AvitoAndroid_Build
2.22 hour
1.67 hour
1.11 hour
33.3 min
0 ms
05/02
05/04
05/06
05/08
05/10
05/12
05/14

Критический путь

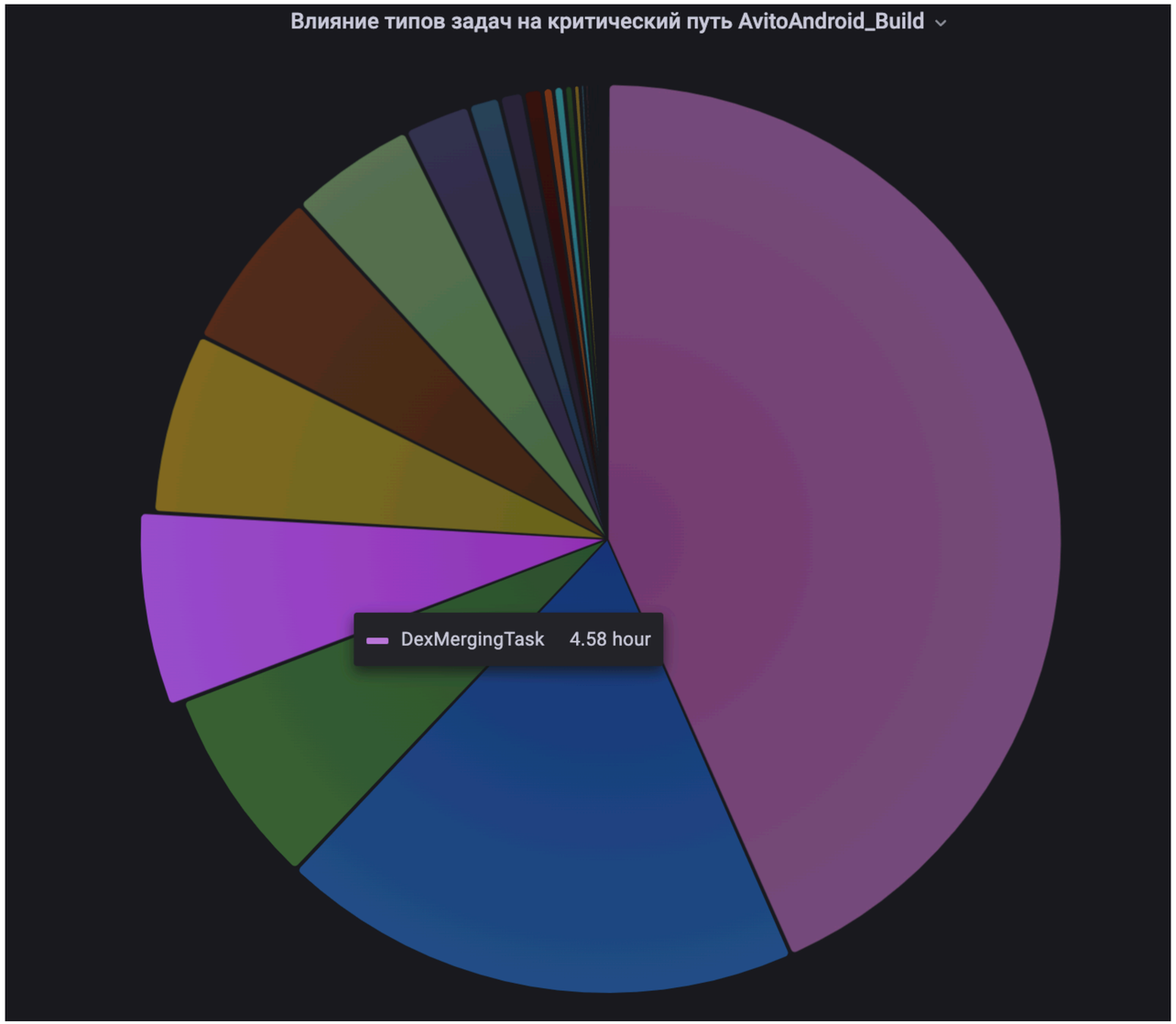


Critical Path = 4 Days

Влияние задач на критический путь

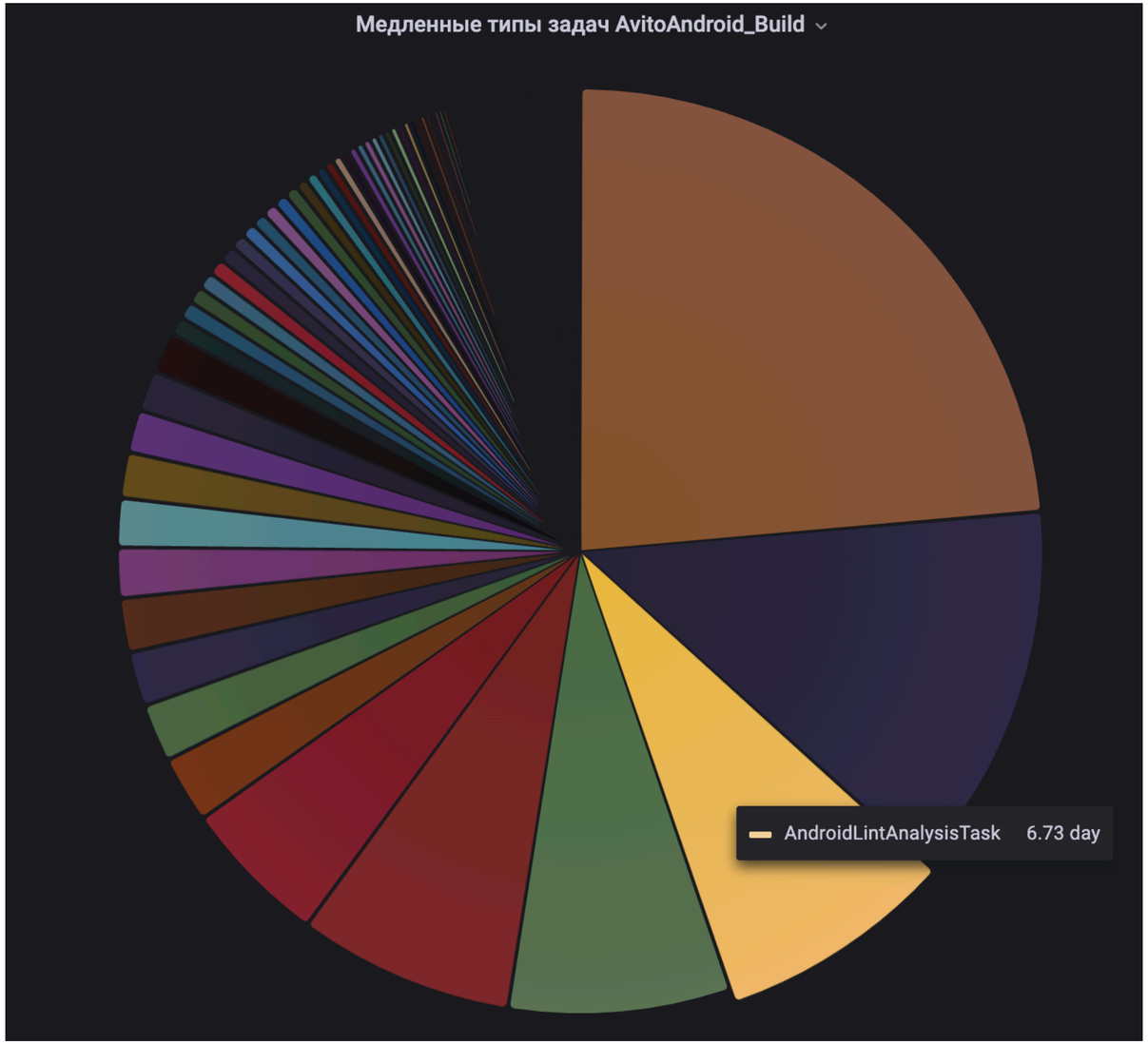


Влияние типов задач на критический путь

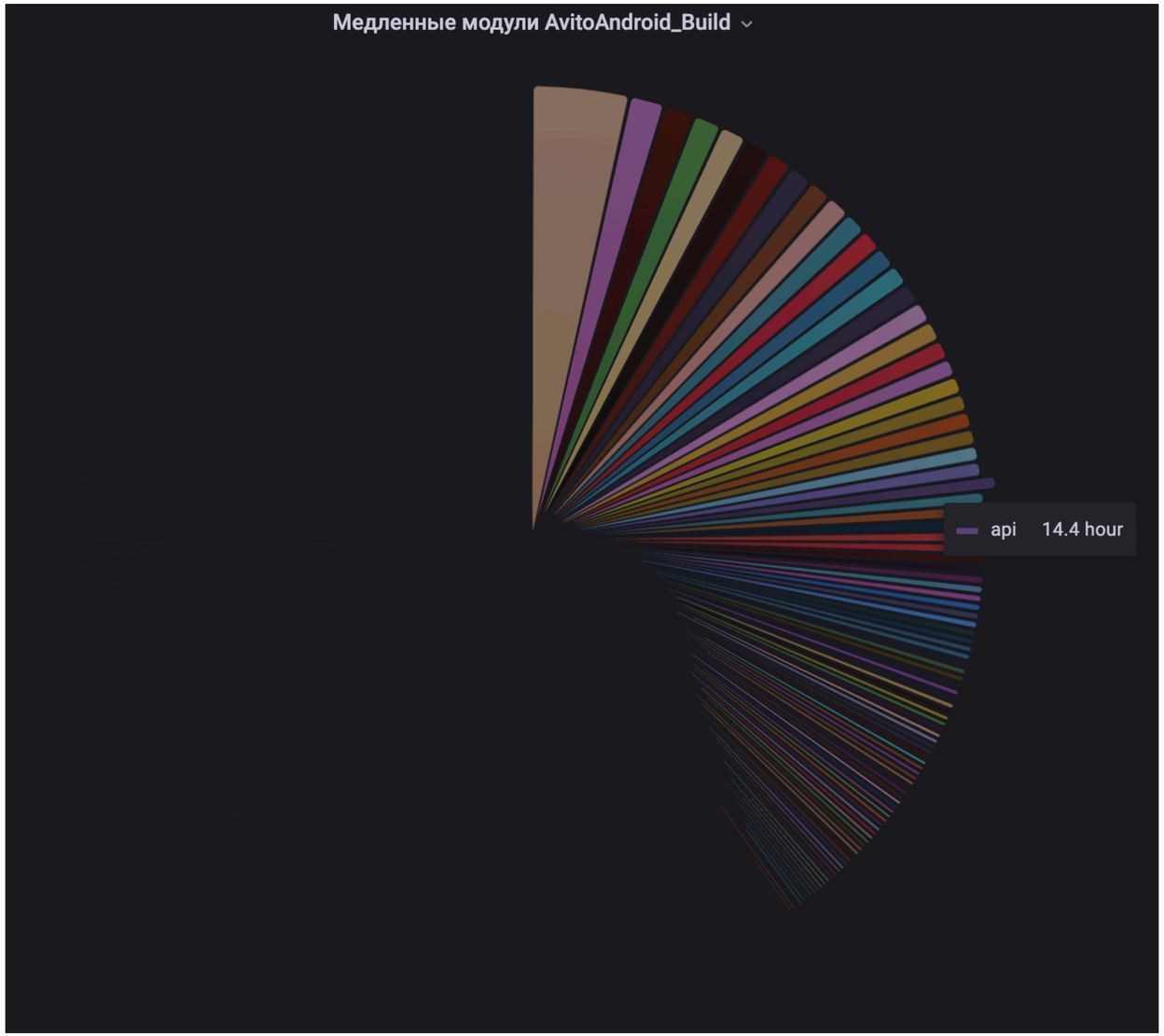


Медленные задачи и модули

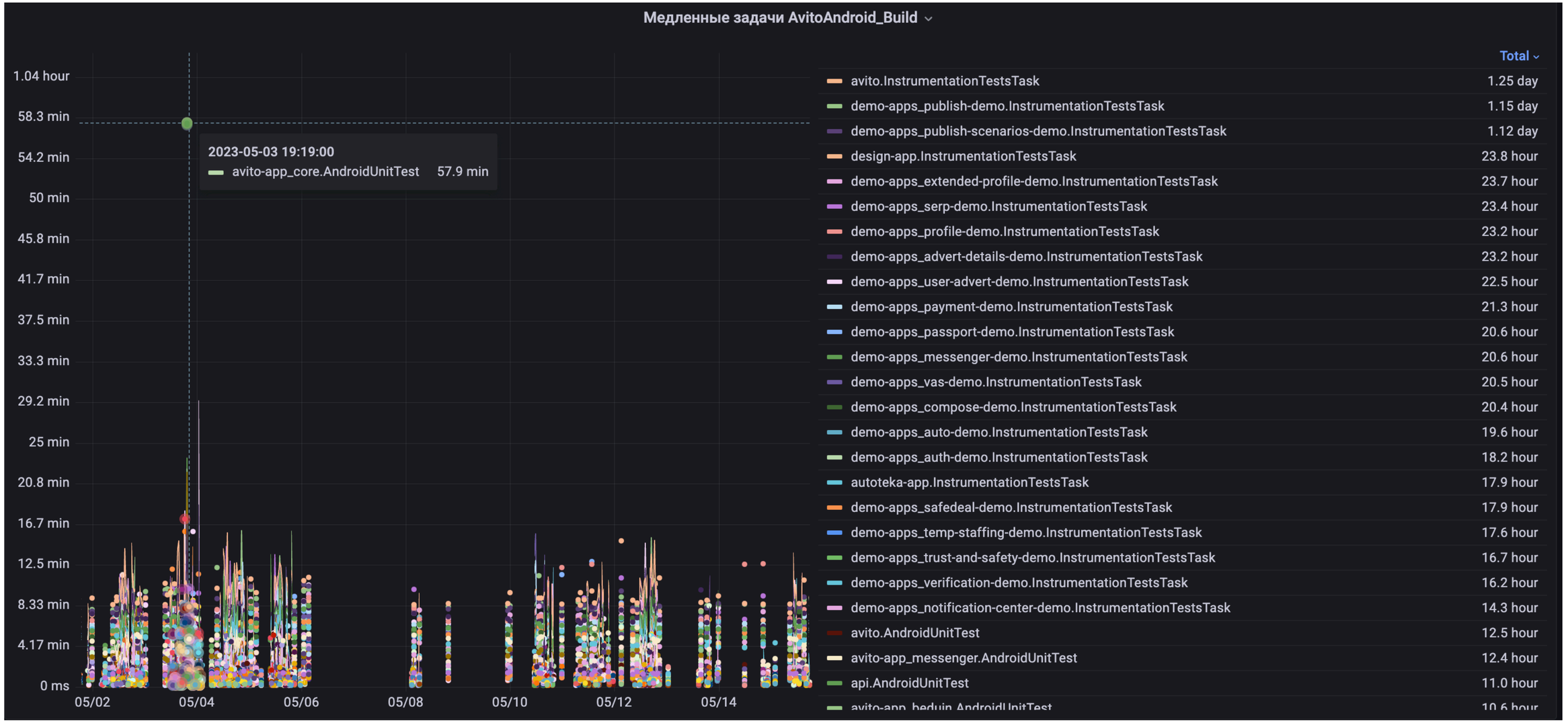
Медленные типы задач



Медленные модули

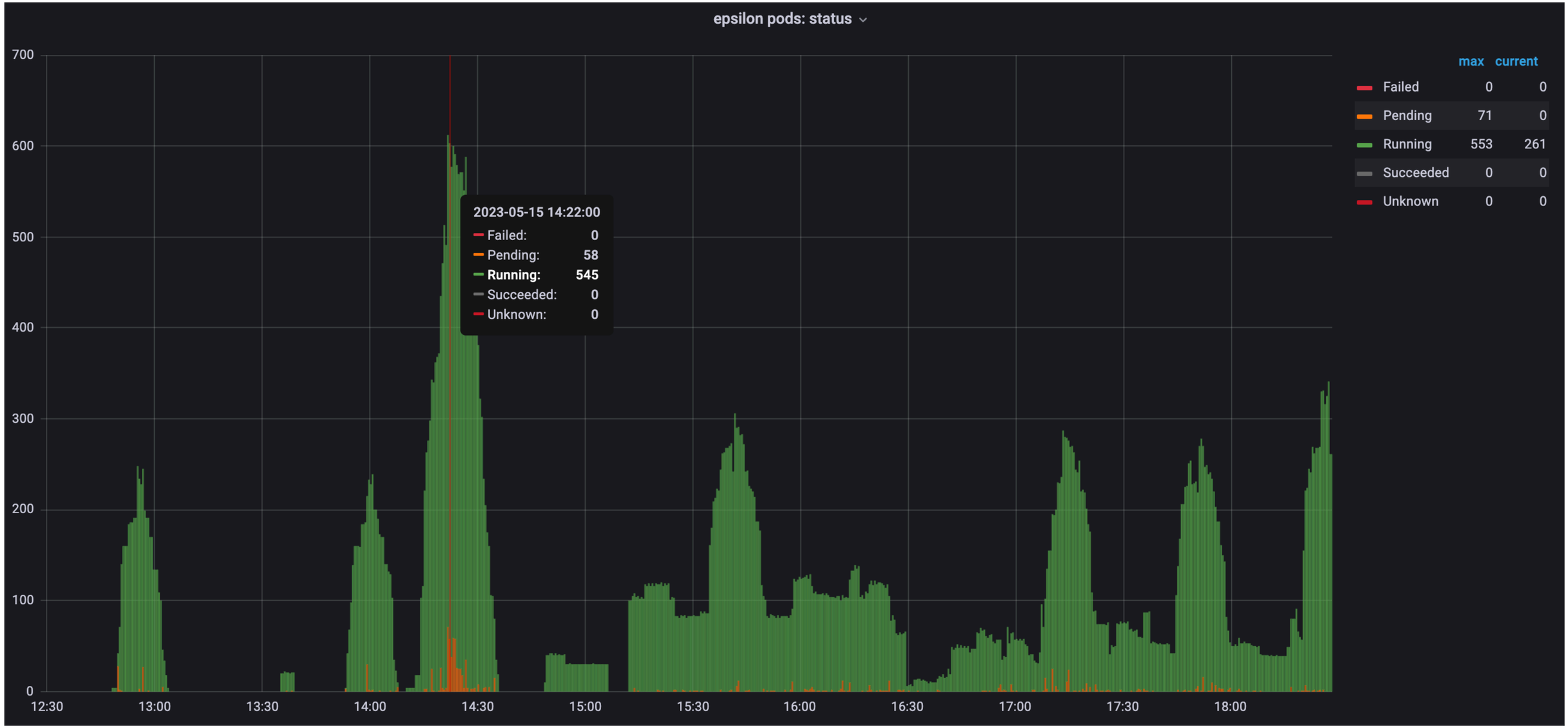


Медленные задачи

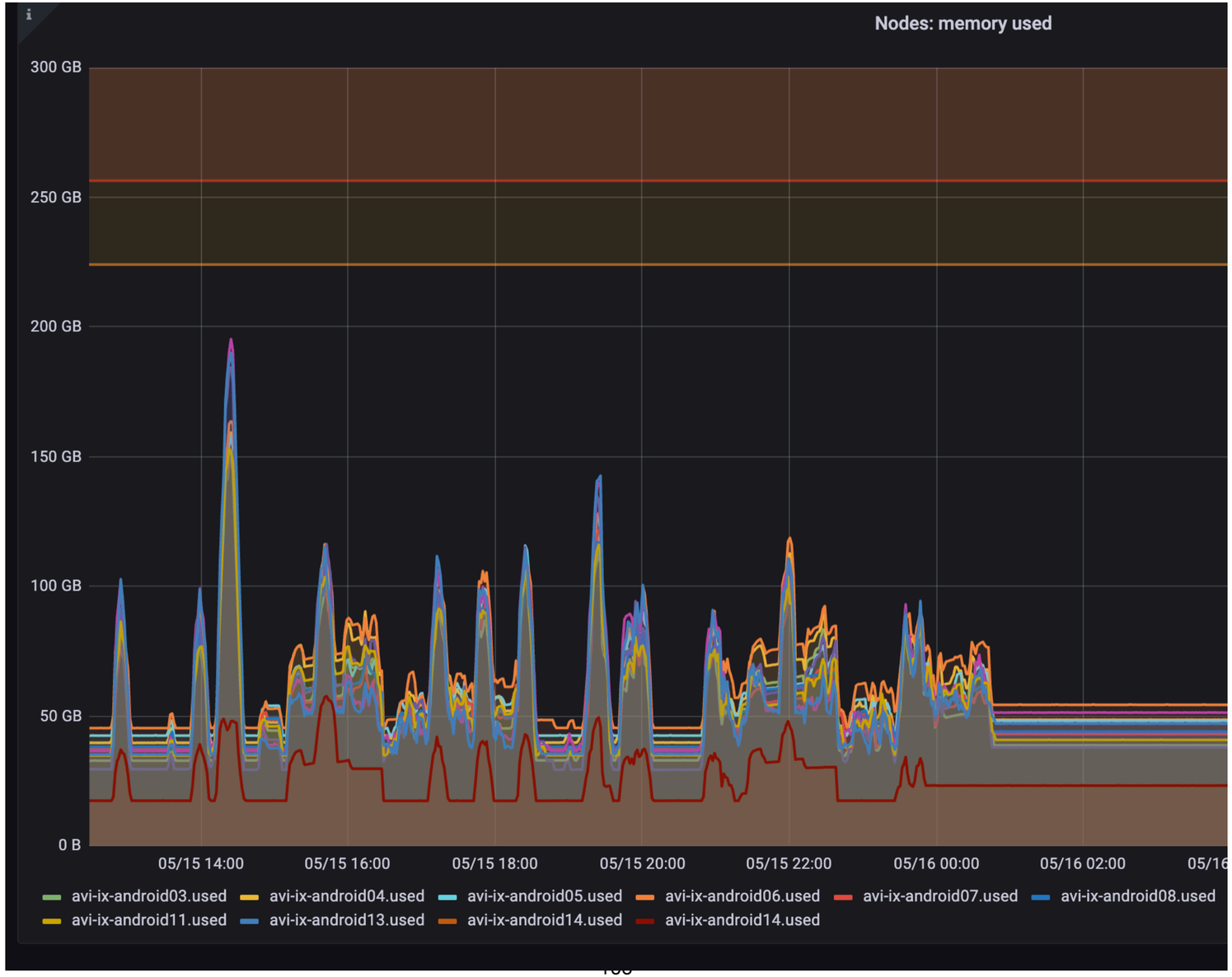


Android Emulators Nodes метрики

Emalators статус

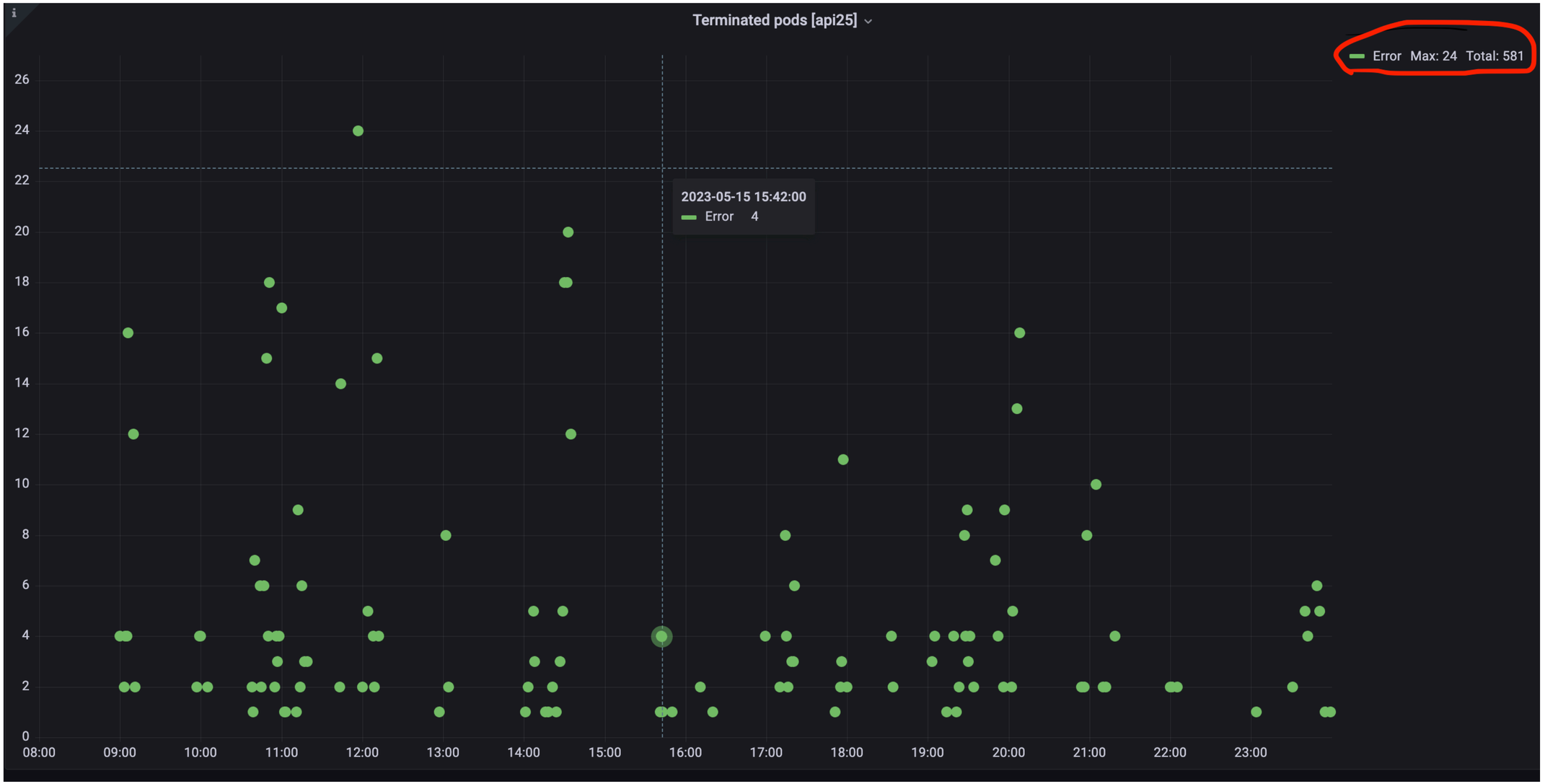


Потребление памяти



Потребление памяти эмулятором

«Умершие» эмуляторы

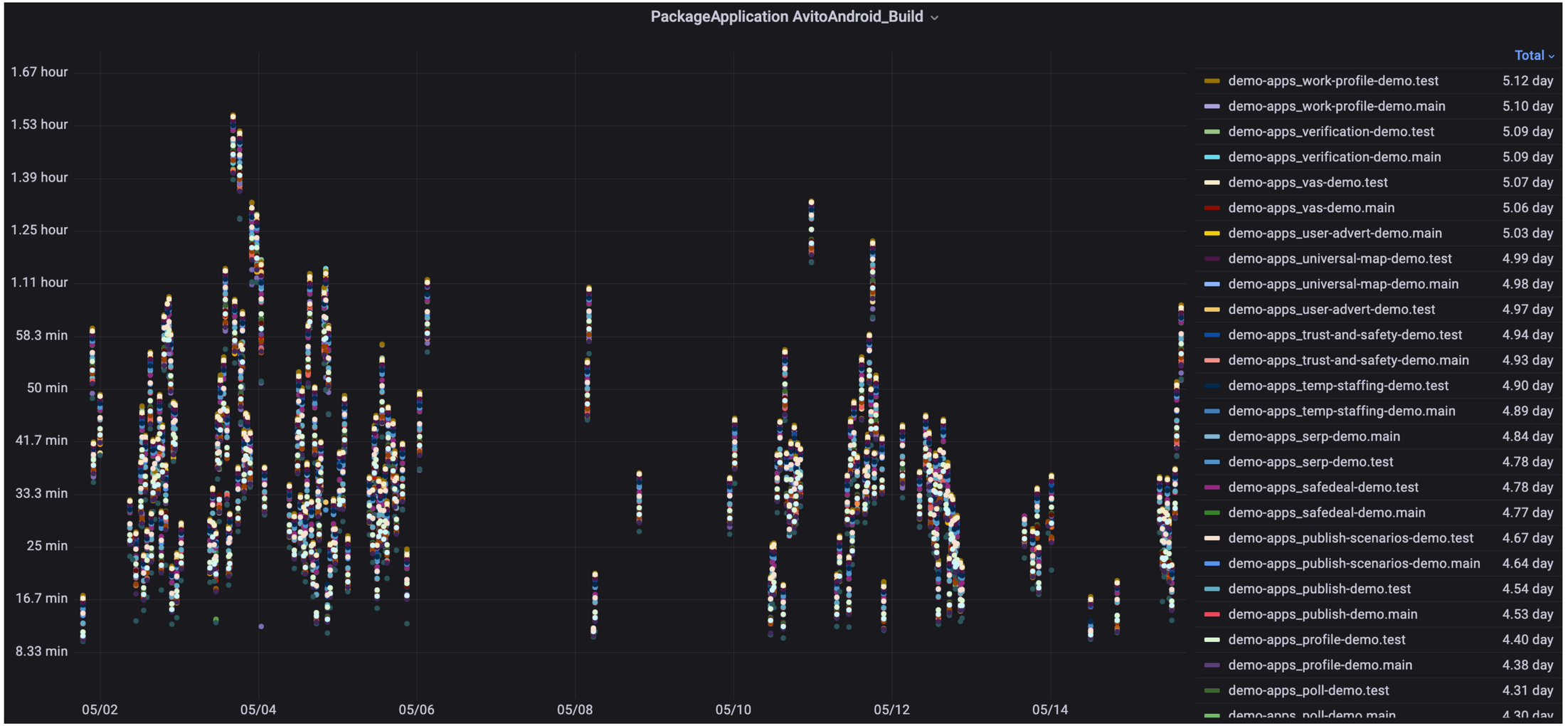


Demo приложения

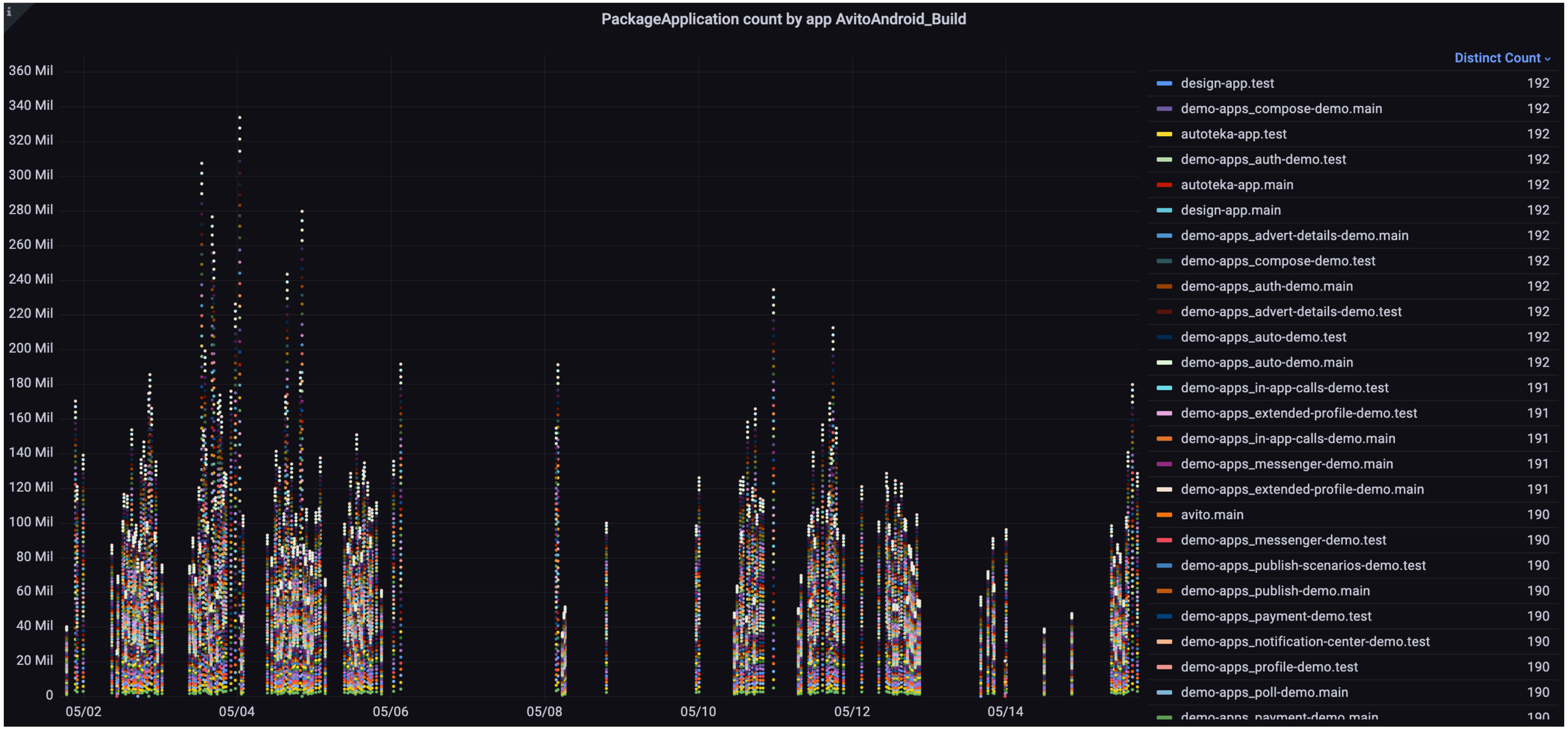
Demo приложение

- Отдельный app
- Подключена только часть модулей
- Ускоряет тестирование и разработку
- Используется для модуляризации Instrumentation тестов

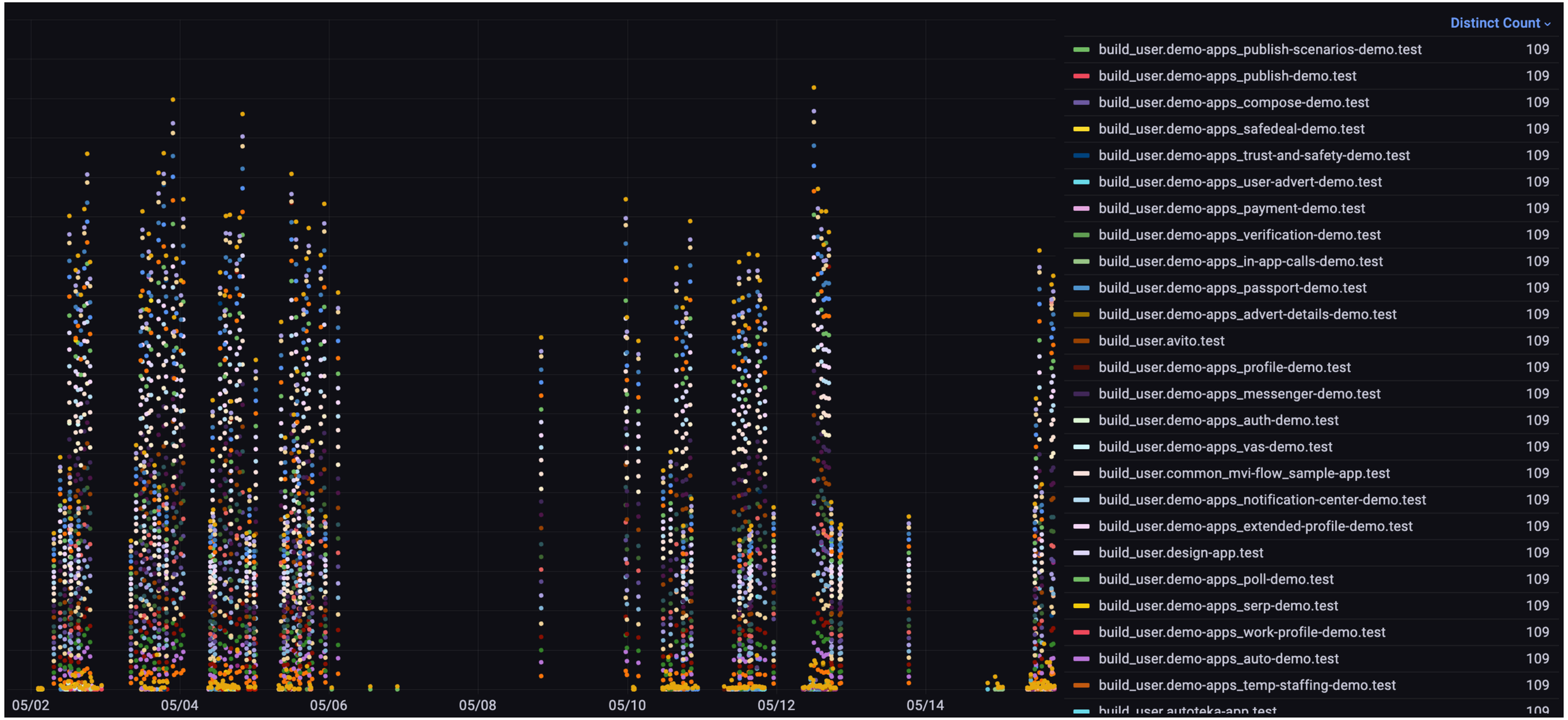
Package Application task



PackageApplication task count



Local PackageApplication task count



3

«Рентген» для сборки

Инструменты для диагностики

Инструменты для

- Сбора метрик
- Анализа метрик
- Реакции на деградации метрик
- Анализа внутренностей сборки

Клиент для сбора метрик

- Gradle Enterprise
- Talaiot
- Avito Build metrics plugin

Gradle Enterpise

- Дорого + невозможно купить в РФ
- Нет возможности контрибьютить
- Нет возможности считать агрегаты на стороне клиента

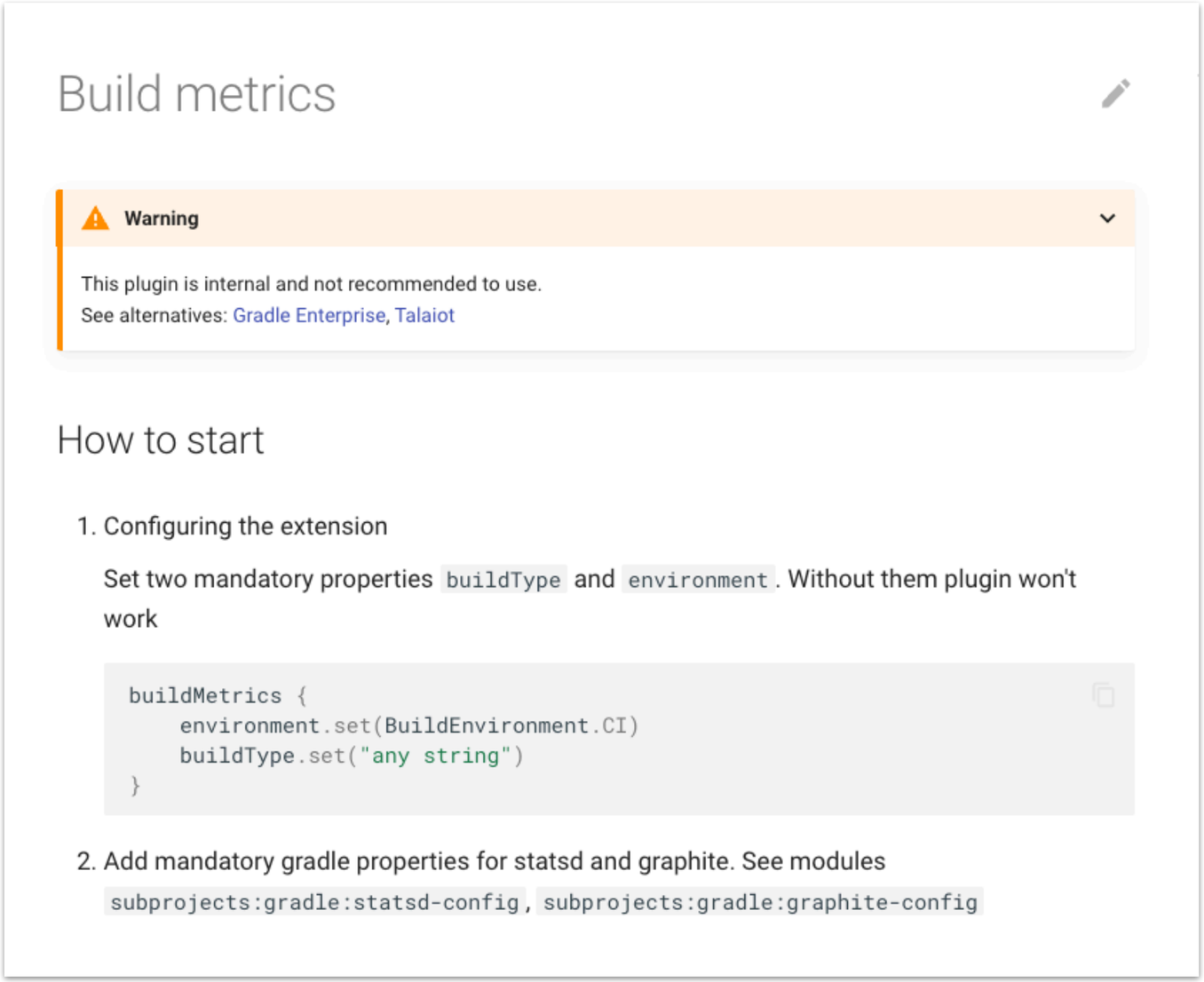
Talaiot

- Когда мы начинали — был сырым
- Последний коммит ~ год назад
- Нет поддержки configuration cache
- Нет агрегатов на стороне клиента

Главное — никто не рассказывает, что делать с метриками

Avito Build metrics plugin

- Наш open source plugin
- Активно разрабатываем и используем
- Пока нет поддержки configuration cache
- Можно подключить себе
  - Но лучше предварительно обсудить



Build metrics

Warning

This plugin is internal and not recommended to use.
See alternatives: Gradle Enterprise, Talaiot

How to start

1. Configuring the extension

   Set two mandatory properties `buildType` and `environment`. Without them plugin won't work

```
buildMetrics {
    environment.set(BuildEnvironment.CI)
    buildType.set("any string")
}
```

2. Add mandatory gradle properties for statsd and graphite. See modules `subprojects:gradle:statsd-config`, `subprojects:gradle:graphite-config`

Backend для сбора метрик

- У нас graphite-api + clickhouse
- У вас любое хранилище, какое уже есть в компании

Анализ метрик

- У нас все дашборды в Grafana
- У вас любое совместимое с вашим backend для сбора метрик

Реакции на деградации метрик

- У нас Moira
- У вас любой инструмент, в котором можно написать правило для alertов + есть поддержка нужных каналов связи

Анализ внутренностей сборки

Gradle scan

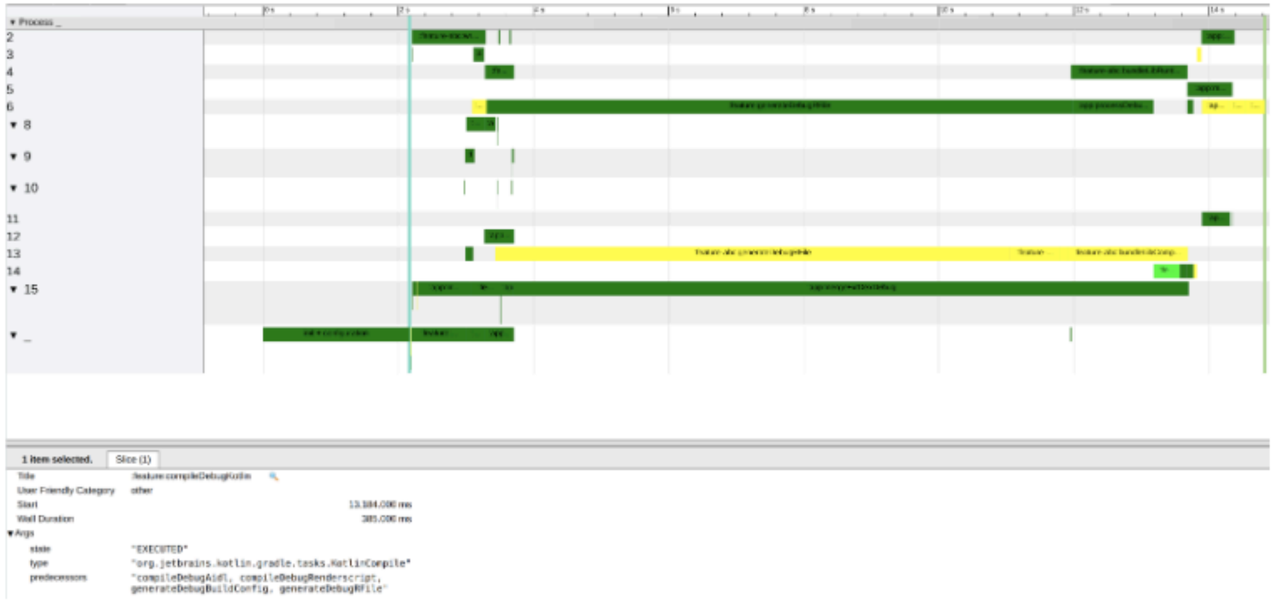
- В целом супер топ
- Не всем подойдет публикация своей инфы или покупка Gradle Enterpise
- Иногда наши сканы не отправляются из-за размера
- Нет критического пути сборки и любых доработок, если захотим

Avito build trace plugin

Build trace Gradle plugin

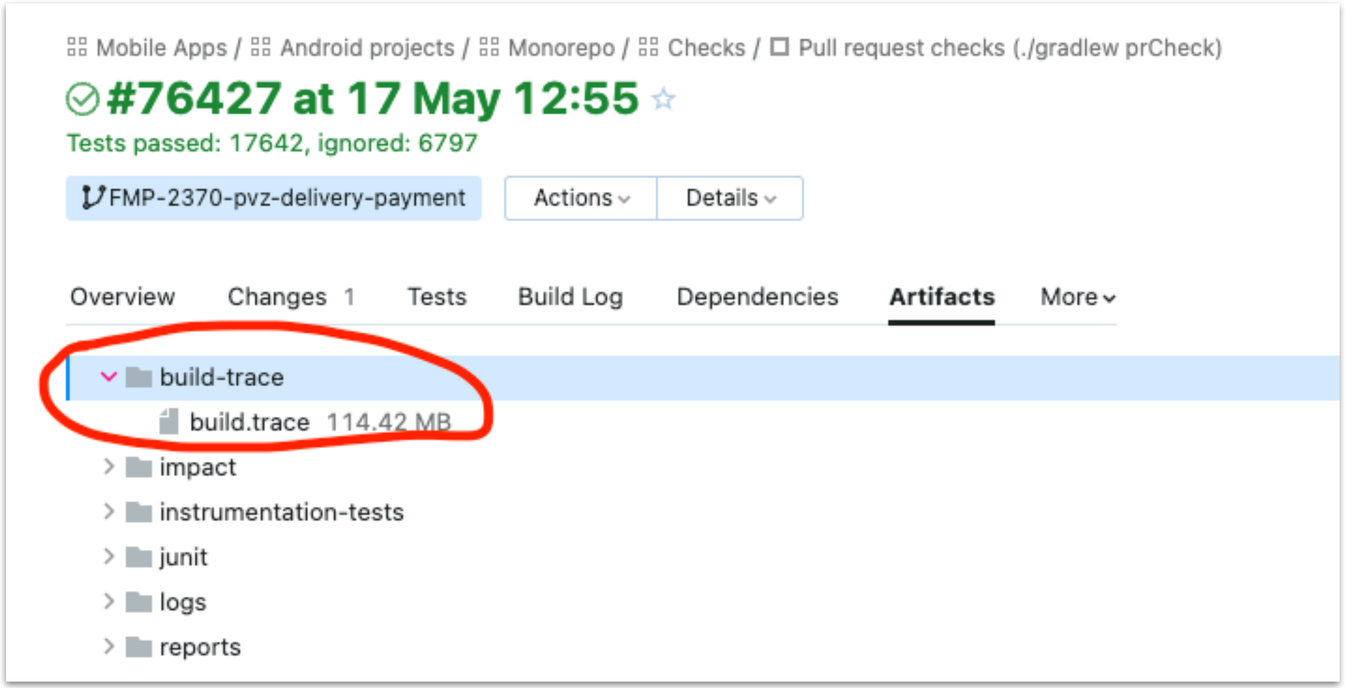
This plugin is a primitive analog of Gradle build scan. Use it if you can't use a build scan for any reason.

This plugin collects tasks execution time in a trace event format.



Avito Build trace plugin

- Файл trace хранится в CI сборки. Его можно скачать и прикрепить к задаче
- Данные вашей сборки не нужно отправлять вовне
- Trace файл есть всегда
- Содержит информацию про критический путь

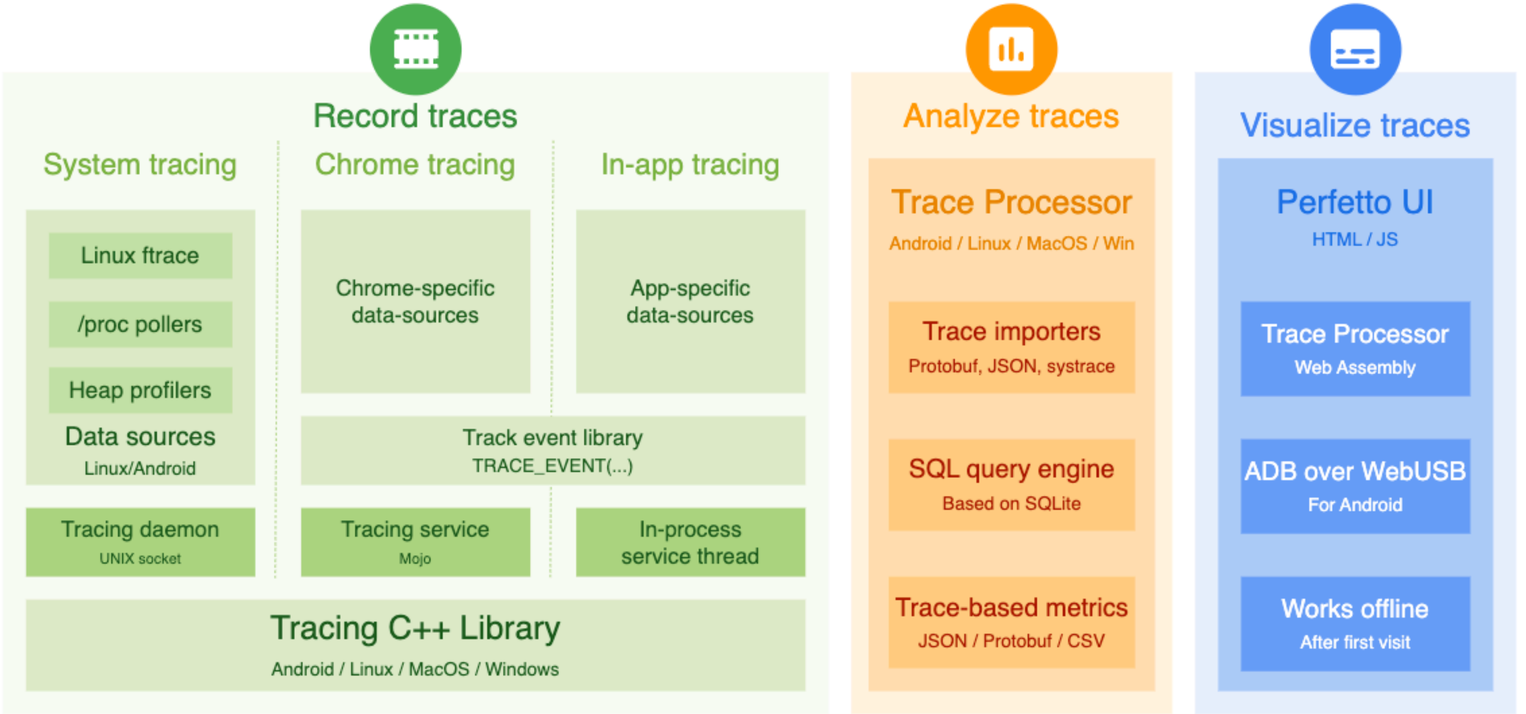


Avito Build trace plugin

- Trace формат общепринят
- Совместим с
  - Chrome tracing
  - Perfetto.dev

Perfetto - System profiling, app tracing and trace analysis

Perfetto is a production-grade open-source stack for performance instrumentation and trace analysis. It offers services and libraries for recording system-level and app-level traces, native + java heap profiling, a library for analyzing traces using SQL and a web-based UI to visualize and explore multi-GB traces.



Recording traces

Perfetto.dev

- Это новый chrome://tracing
- Используется для trace
  - Android
  - Linux
  - Chrome
- Есть SDK для создания trace файлов


Perfetto
Updated to v34.0-1b8de4452 and ready for offline use!
Dismiss
Navigation
Open trace file
Open with legacy UI
Record new trace
Current Trace
build (24).trace (120 MB)
Show timeline
Download
Query (SQL)
Metrics
Info and stats
Convert trace
Switch to legacy UI
Convert to .json
Example Traces
Open Android example
Open Chrome example
Support
Keyboard shortcuts
Documentation
Process 0
Thread 0
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
Thread 9
Thread 10
Thread 11
Thread 12
Thread 13
Thread 14
Current Selection
Slice Details
Name :avito:compileDebugAndroidTestKotlin
Category null
Start time 1m 16s 358ms 40ns
Duration 1m 38s 95ms
Thread duration 0s (0.00%)
Thread 4
Process 0
Slice ID 18469
Arguments
args
state EXECUTED
type org.jetbrains.kotlin.gradle.tasks.KotlinCompile
CRITICAL_PATH true
predecessors :api:bundleLibCompileToJarRelease, bundleDebugClassesToCompileJar, generateDebugAndroidTestBuildConfig, kaptDebugAndroidTestKotlin, processDebugAndroidTestResources, :avito-debug:bundleLibCompileToJarRelease, :avito-debug-lib:bundleLibCompileToJarRelease, :messenger:bundleLibCompileToJarRelease, :api-models:models:bundleLibCompileToJarRelease, :avito-api:abuse:bundleLibCompileToJarRelease, :avito-api:active-orders:bundleLibCompileToJarRelease, :avito-api:advert-details:bundleLibCompileToJarRelease, :avito-api:advert-stats:bundleLibCompileToJarRelease, :avito-

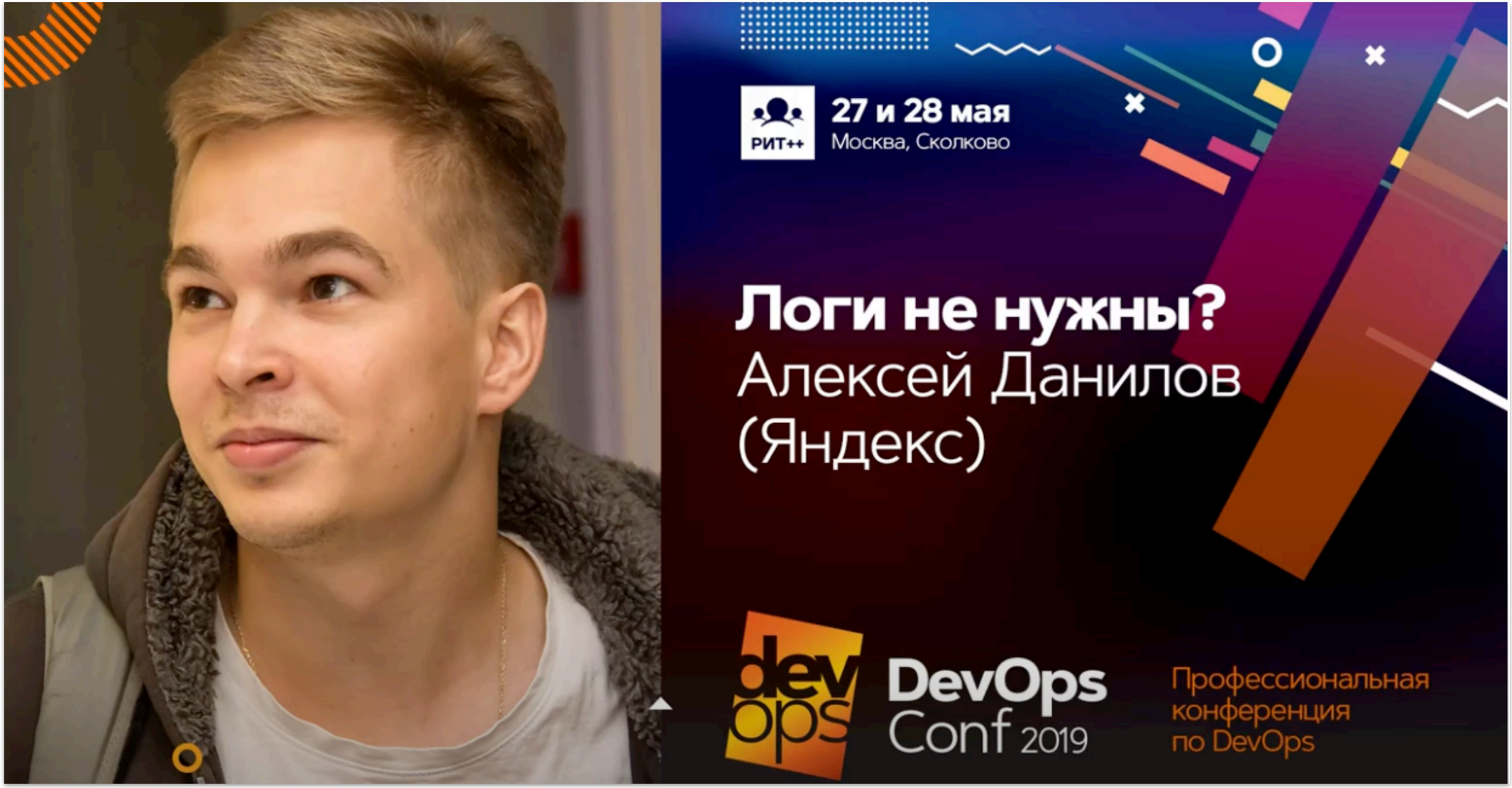
Updated to v34.0-1b8de4452 and ready for offline use! Dismiss

```
select name, (dur/ 1000000000) as duration_sec from slice order by dur desc
```

Query result - 54 ms `select name, (dur/ 1000000000) as duration_sec from slice order by dur desc` Copy query Copy result (.tsv) Close

name	duration_sec
:avito:instrumentationPRCheckKubernetes	674
:demo-apps:publish-demo:instrumentationPRCheckKubernetes	593
:demo-apps:publish-scenarios-demo:instrumentationPRCheckKubernetes	593
:demo-apps:profile-demo:instrumentationPRCheckKubernetes	527
:demo-apps:serp-demo:instrumentationPRCheckKubernetes	498
:demo-apps:user-advert-demo:instrumentationPRCheckKubernetes	483
:demo-apps:verification-demo:instrumentationPRCheckKubernetes	470
:design-app:instrumentationPRCheckKubernetes	455
:demo-apps:extended-profile-demo:instrumentationPRCheckKubernetes	449
:demo-apps:advert-details-demo:instrumentationPRCheckKubernetes	444
:demo-apps:payment-demo:instrumentationPRCheckKubernetes	438
:demo-apps:vas-demo:instrumentationPRCheckKubernetes	438
:demo-apps:messenger-demo:instrumentationPRCheckKubernetes	428
:demo-apps:passport-demo:instrumentationPRCheckKubernetes	416
:demo-apps:safedeal-demo:instrumentationPRCheckKubernetes	394
:demo-apps:trust-and-safety-demo:instrumentationPRCheckKubernetes	393
:demo-apps:auto-demo:instrumentationPRCheckKubernetes	371
:demo-apps:compose-demo:instrumentationPRCheckKubernetes	359
:demo-apps:auth-demo:instrumentationPRCheckKubernetes	342
:demo-apps:notification-center-demo:instrumentationPRCheckKubernetes	296
:autoteka-app:instrumentationPRCheckKubernetes	276
:demo-apps:poll-demo:instrumentationPRCheckKubernetes	250
:avito-libs:serp-core:testReleaseUnitTest	174
:avito:testDebugUnitTest	164
:demo-apps:in-app-calls-demo:instrumentationPRCheckKubernetes	118
:avito-app:serp:testReleaseUnitTest	114
:avito-app:advert-details:testReleaseUnitTest	106
:detektAll	104
:avito:lintAnalyzeRelease	104
:avito:mergeDebugAndroidTestJavaResource	98
:avito:compileDebugAndroidTestKotlin	98
:avito:stripDebugDebugSymbols	96

Логи

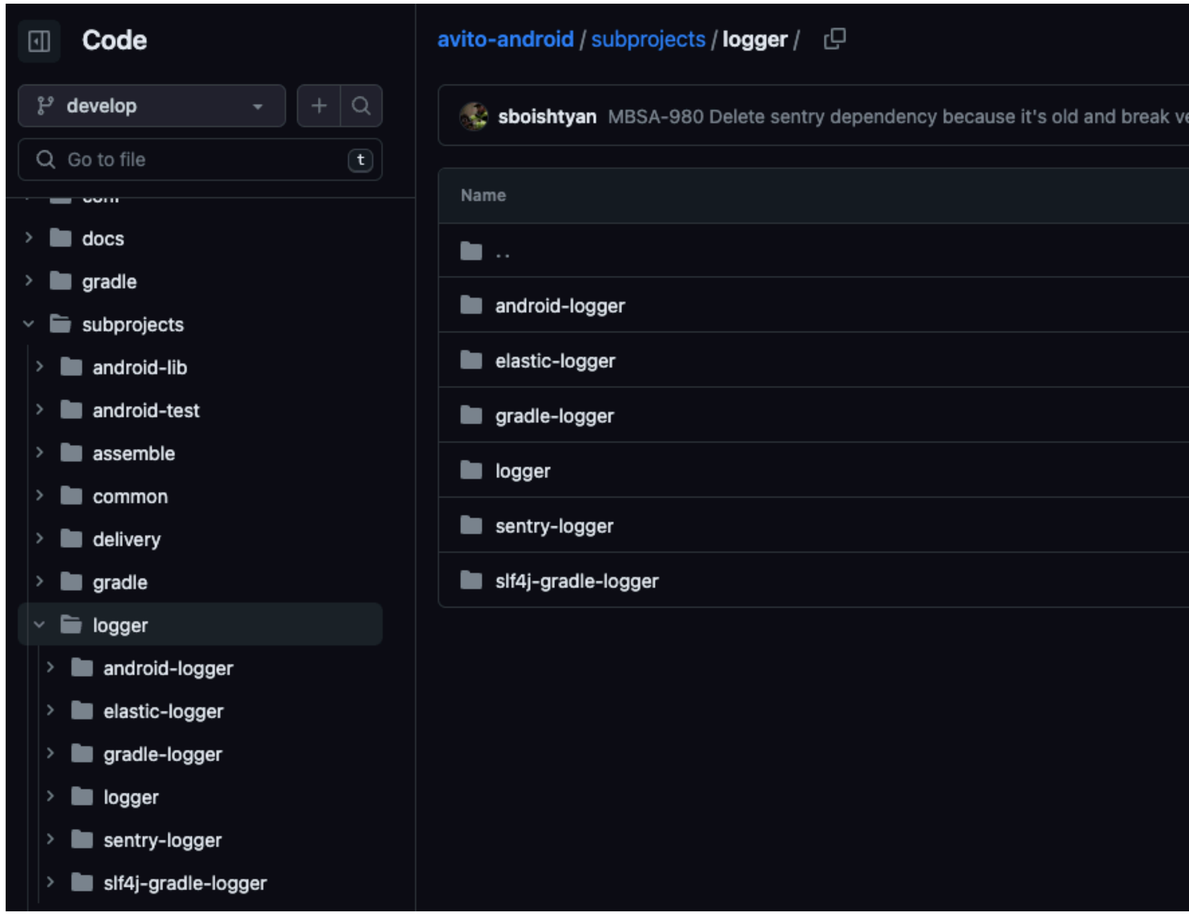

РИТ++
27 и 28 мая
Москва, Сколково
Логи не нужны?
Алексей Данилов
(Яндекс)
dev ops
DevOps Conf 2019
Профессиональная конференция по DevOps

Gradle log

- Нет возможности добавлять новые источники
- Нет возможности настроить разные уровни для разных частей
- Неудобная большая «простыня»
- В info уровне «миллион» строчек
- В debug вообще страшно включать на CI
- Поиск неудобный

Avito Gradle logger

- Расширяемый набор источников
  - Kibana
  - Sentry
  - FileSystem
- Возможность конфигурировать logger внутри кода



Avito build verdict

Overview | Changes 1 | Build Log | Dependencies | Artifacts | **Build verdict** | More

FAILURE: Build completed with 2 failed tasks

1: What went wrong:

```
Execution failed for task ':avito-test:beduin:compileReleaseKotlin'.
        > A failure occurred while executing org.jetbrains.kotlin.compilerRunner.GradleCompilerRunnerWithWorkers$GradleKotlinCompilerWorkAction
                > Compilation error. See log for more details
```

Error logs:

```
e: /Users/Shared/projects/avito-android/avito-test/beduin/src/main/kotlin/com/avito/android/api/models/BeduinPaymentMethodsGroupDsl.kt: (26, 5): No value passed for parameter 'isEnabled'
```

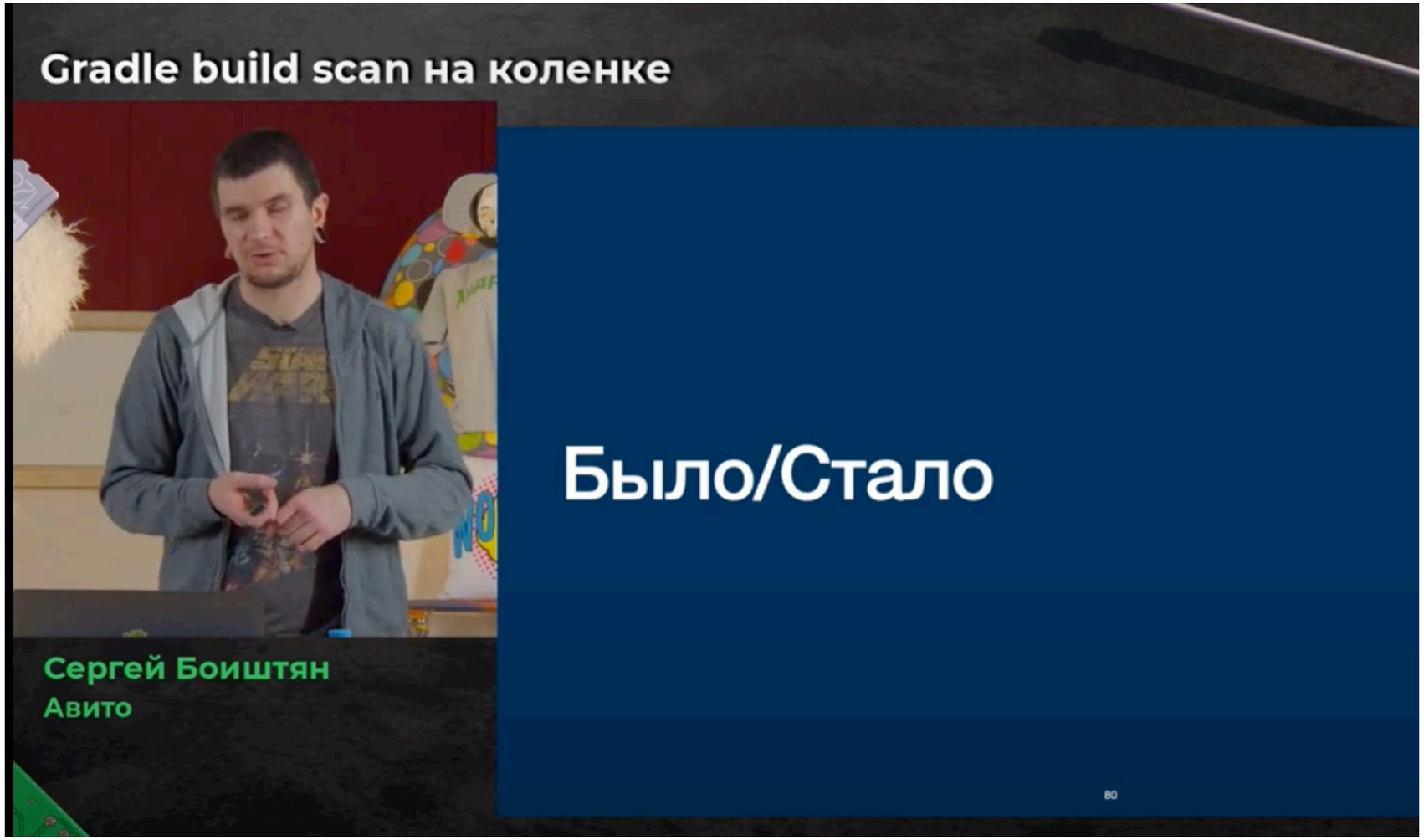
2: What went wrong:

```
Execution failed for task ':avito-app:beduin:compileReleaseUnitTestKotlin'.
        > A failure occurred while executing org.jetbrains.kotlin.compilerRunner.GradleCompilerRunnerWithWorkers$GradleKotlinCompilerWorkAction
                > Compilation error. See log for more details
```

Error logs:

```
e: /Users/Shared/projects/avito-android/avito-app/beduin/src/test/kotlin/com/avito/android/beduin/parsing/BeduinPaymentMethodSelectorParsingTest.kt: (61, 5): No value passed for parameter 'isEnabled'
```

Avito build verdict



4

Система «здравоохранения» для сборки

Зона ответственности

- Чем больше разработчиков, тем проще параллельно развивать разные направления
- Мы отвечаем за скорость и стабильность CI/CD
- Мы даем инструменты для тестирования и качества кода
  - Но не следим за ним
- Это помогает держать фокус и брать всю ответственность на себя

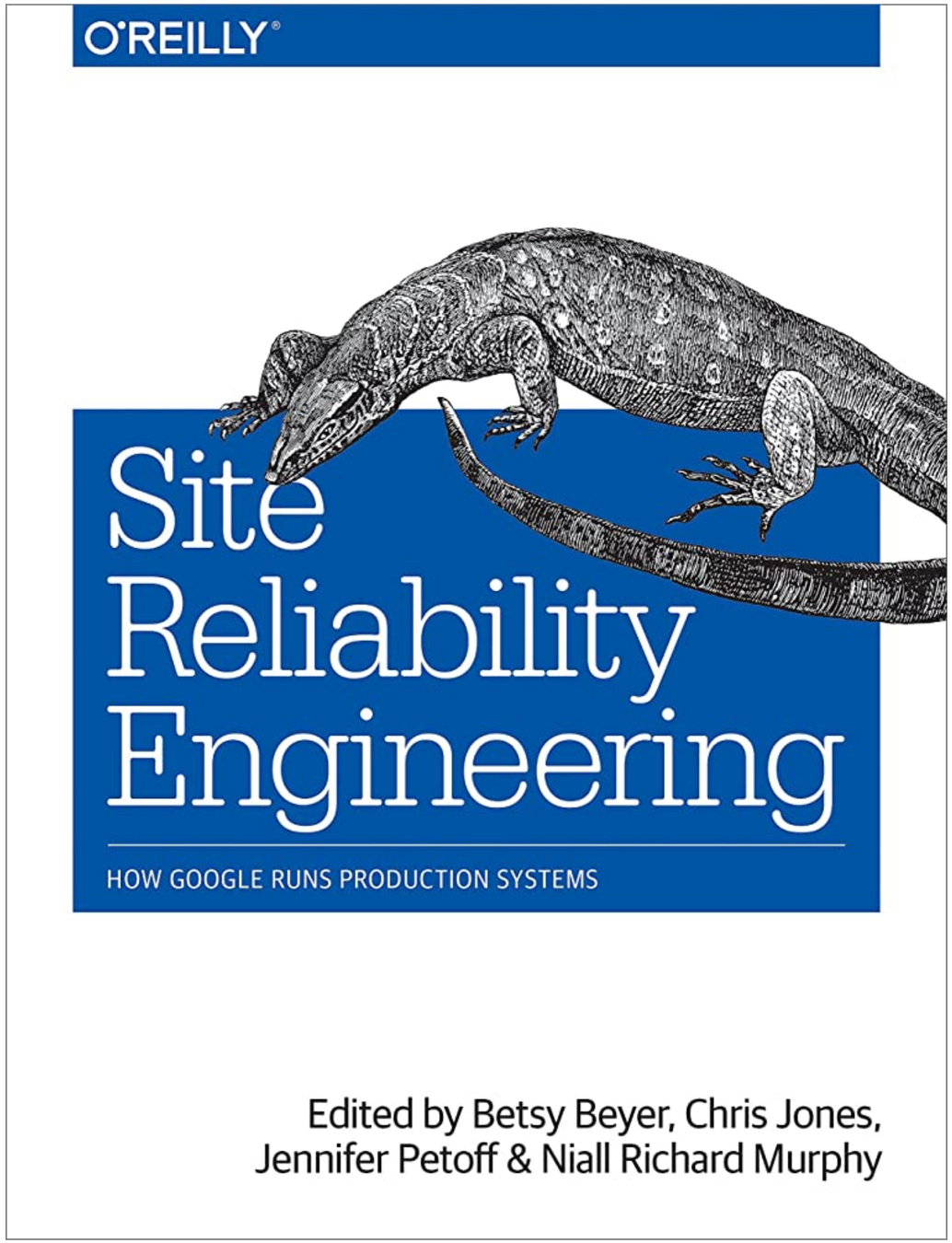
Скорая помощь

Главная задача скорой помощи

СКОРО оказать ПОМОЩЬ

Дежурство «скорая помощь»

- Дежурный загружен на 50%
- Его задача
  - Реагировать на Алерты
  - Смотреть дашборды
  - Отвечать пользователям
  - Готовить отчеты = Ретро



Зачем нужны дежурства?

- Помочь пользователям решить проблему в разумное время, не оставить их без ответа
- Ограждает команду от внешних отвлечений, выступает входной точкой
- Узнавать об авариях раньше. Иметь ответственных для их решения

Что было без дежурств:

- Трагедия общин, неравномерная нагрузка на команду. Либо никто не хотел отвечать, либо толпой наваливались.
- Хуже обменивались знаниями, было проще не сталкиваться с какой-то областью проблем

Ретро после дежурства

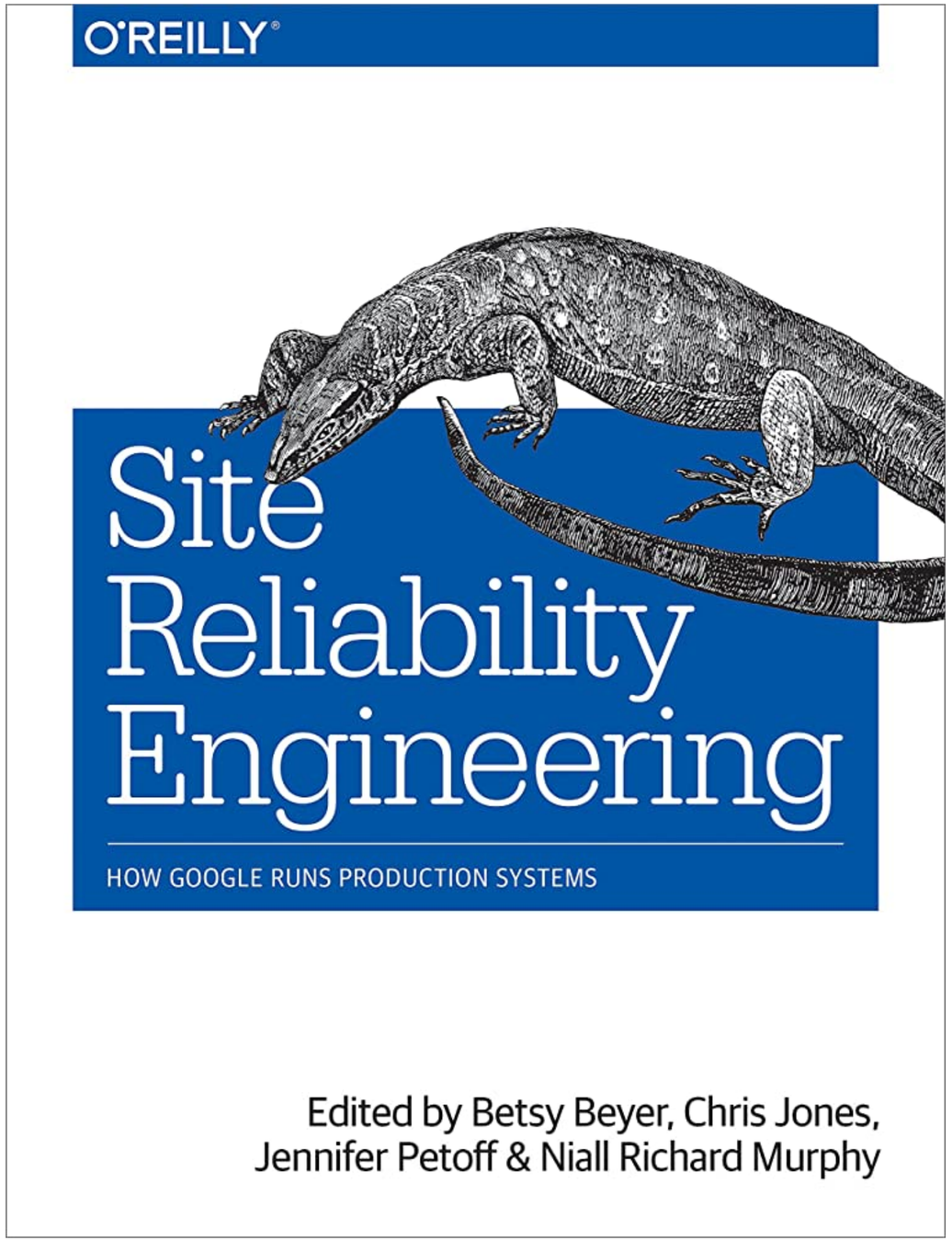
- Обсуждаем SLA за неделю
- Обсуждаем инциденты, которые решали на неделе
- Хватило ли для быстрого решения задач
  - Метрик
  - Алертов
  - Документации
- Что стоит запланировать

Разбор инцидентов

После «скорых» решений проблем нужен план, чтобы не допустить их в будущем

Обычно:

- Новые процессы
- Новые метрики
- Новые инструменты



Как понять, что система работает?

- Вы знаете типичное состояние системы
- Вы знаете, когда и как нетипично ваша система себя ведет
- В любой момент с помощью телеметрии вы можете понять состояние системы

Ссылки

- Плейлист с полезными докладами