Яндекс

Перформанс характеристики:

Илья Богин

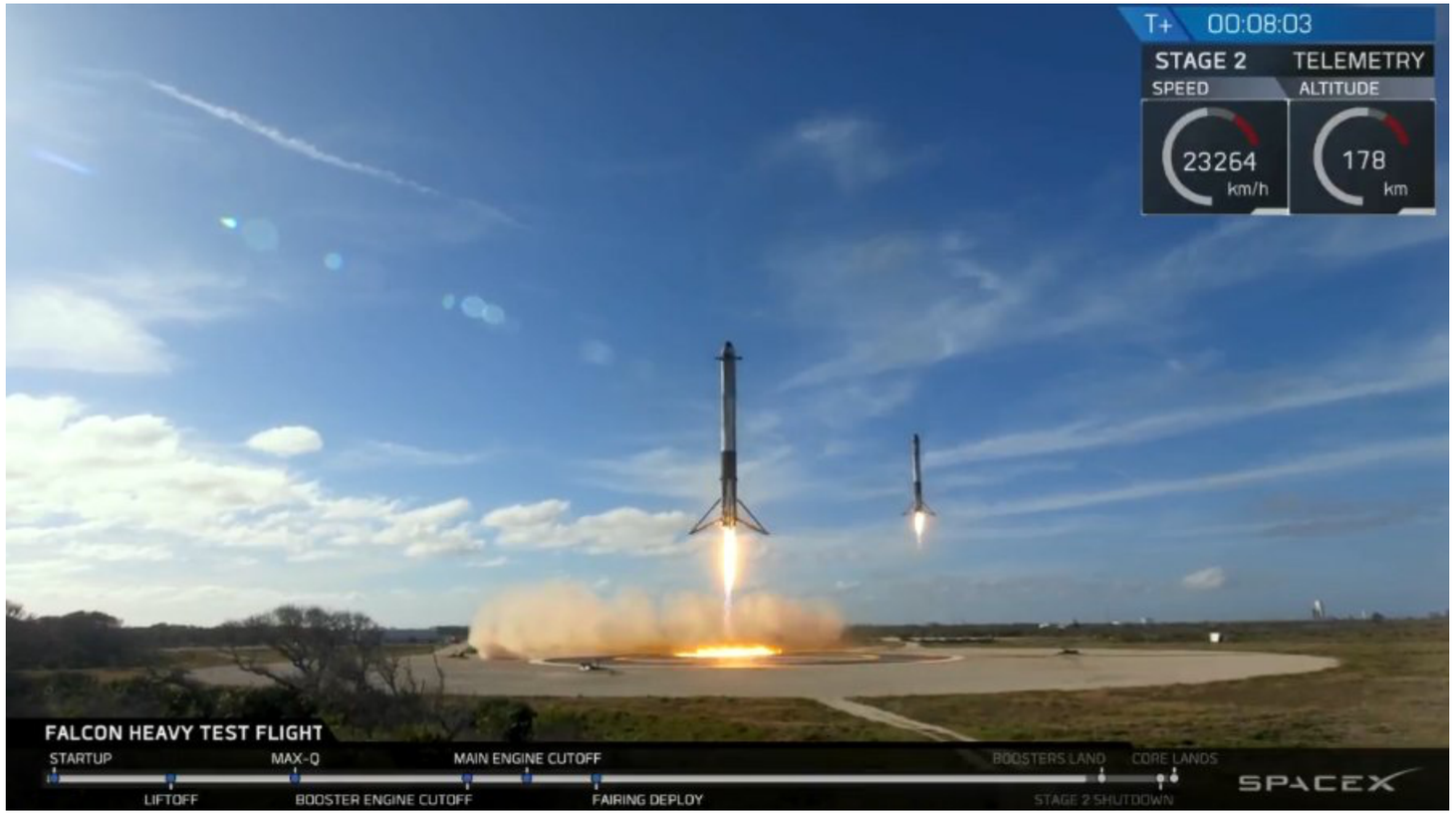
Борьба за производительность. Сегодня

В.П. Чкалов

- Программисты-асы досконально разбирают приложение и находят критические точки
- Результатом работы являются значительные разовые достижения

https://topwar.ru/39175-v-nebe-chkalov.html

CI для перформанс-характеристик. Завтра

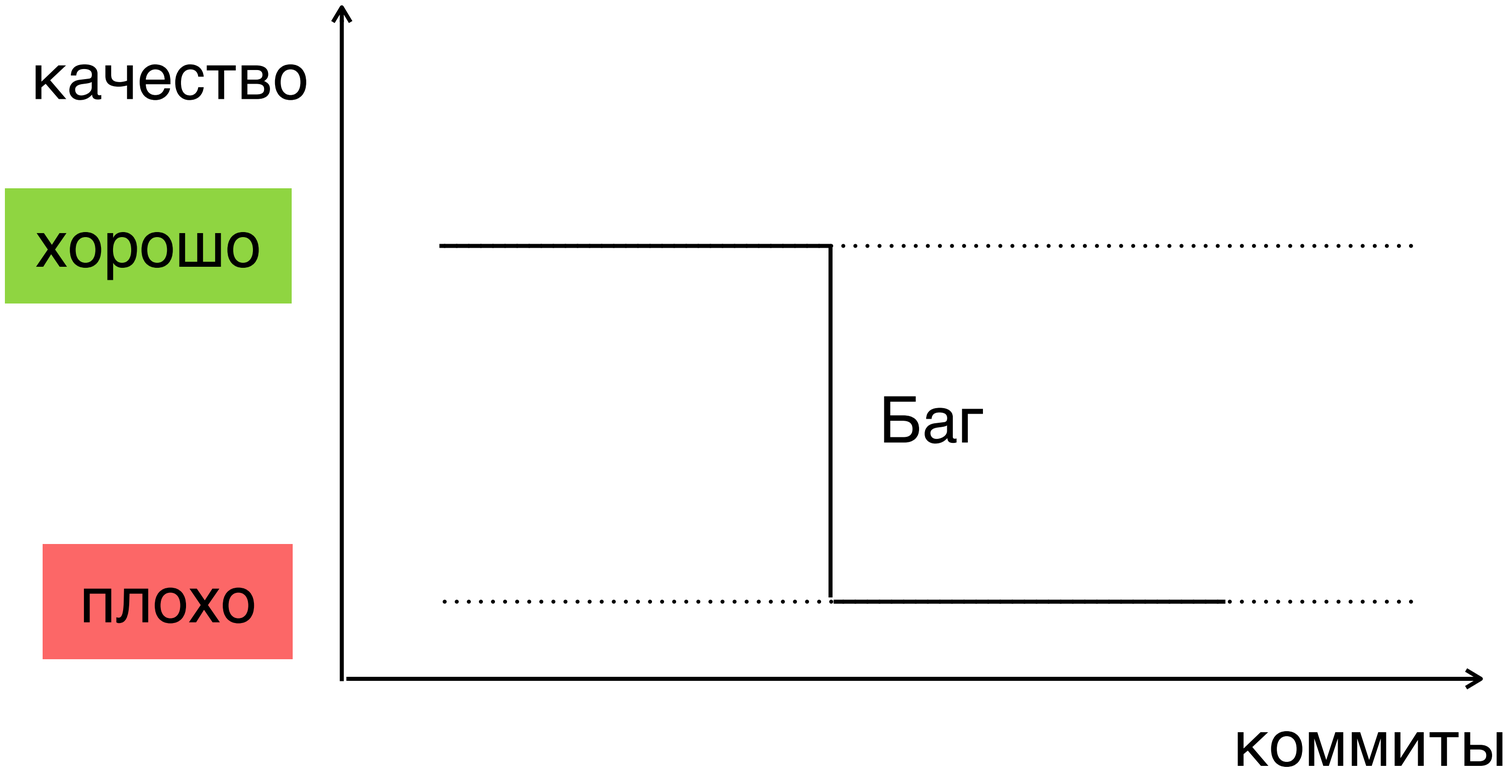


- Мы хотим добиться автоматической многоразовой и постоянной работы по улучшению перформанс-характеристик

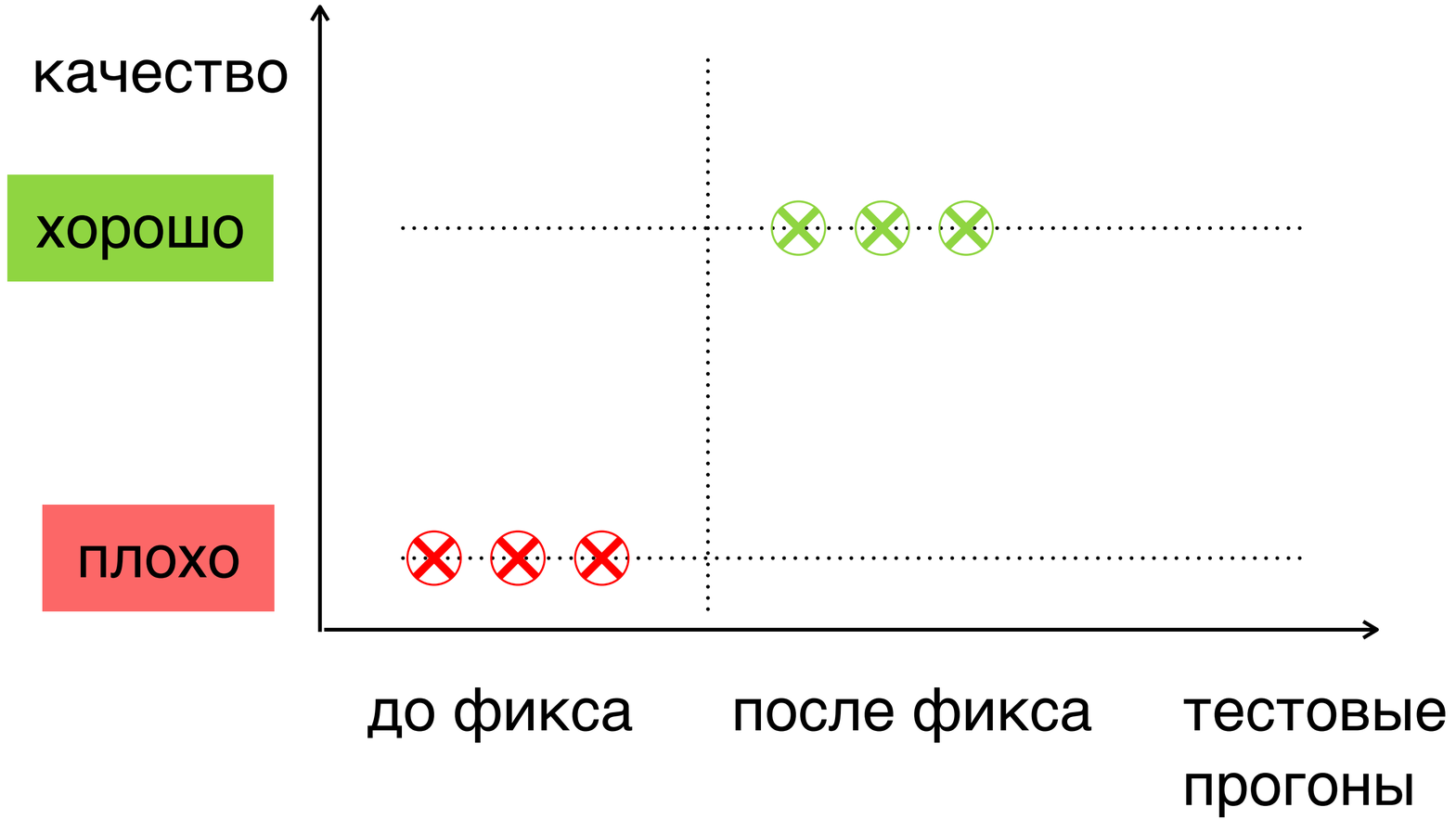
https://twitter.com/SpaceX/status/960980119312453632/photo/1

Сравнение функциональных багов и регрессий перформанс-характеристик

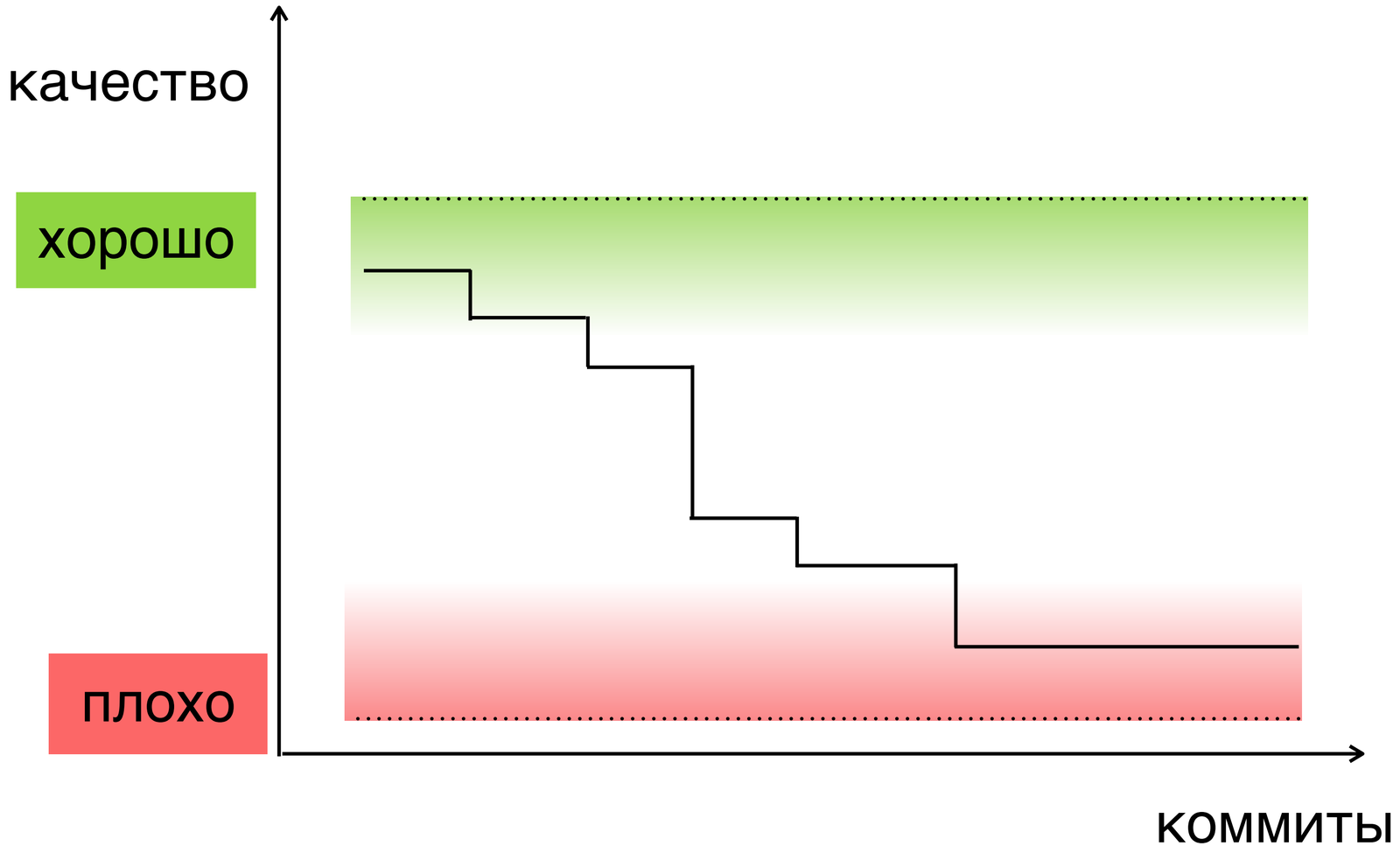
1.1 Функциональные баги



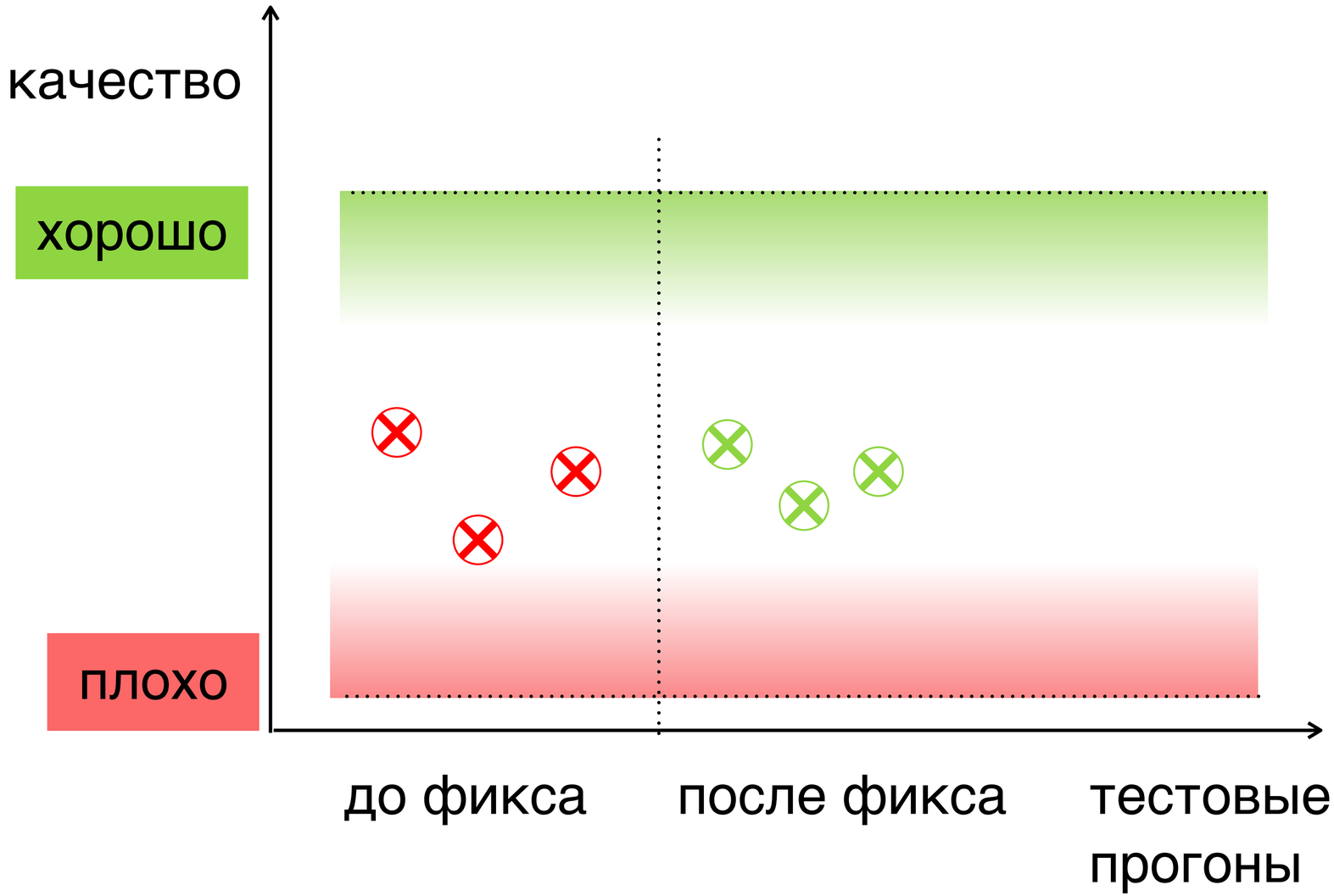
1.2 Тестирование функциональных багов



1.3 Регрессии перформанс-характеристик



1.4 Тестирование регрессий



Если вы не можете это измерить вы не сможете это улучшить

2.1 Измерение перформанс характеристик

Что мы хотим измерять?

- Скорость старта приложения
- Скорость основных сценариев работы
- Плавность скролла
- Плавность анимаций
- Потребление RAM
- Потребление траффика
- Потребление батарейки
- Количество пробуждений приложения

2.2 Измерение перформанс характеристик

Проблемы измерения:

- Гетерогенная многоядерность (big.LITTLE архитектура)
- Троттлинг частоты ядер
- Изменение характеристик работы при повышении температуры
- Различные условия сборки приложений

2.3 Внимание, вопрос

Однажды, мы столкнулись с периодическими аномалиями времени старта Android браузера.

Анализ привёл нас к тому что все аномальные результаты показывают сборки с одного и того же агента.

В чем может быть проблема?

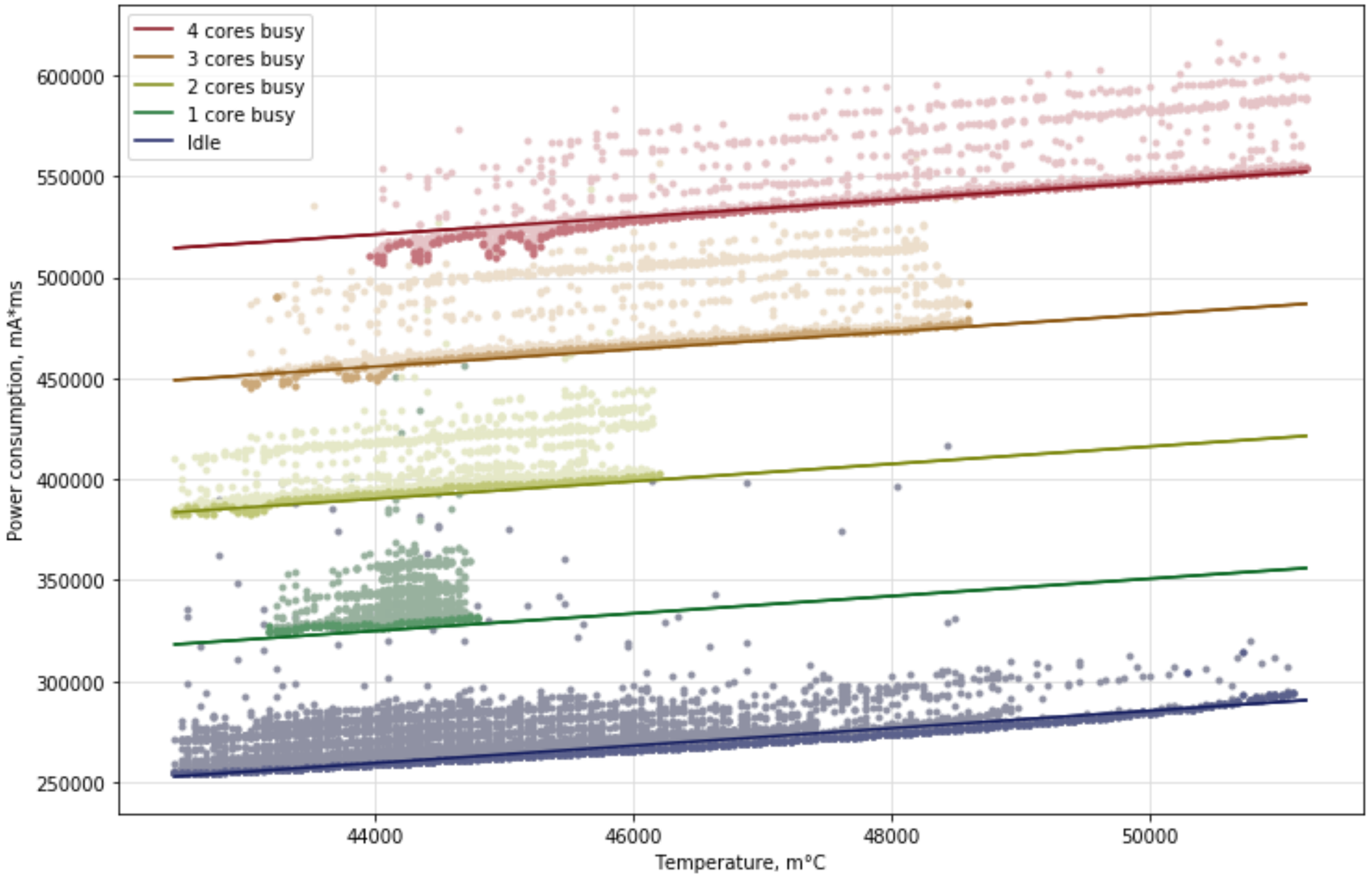
1. На этом агенте использовался gcc для сборки нативного кода, на остальных clang
2. Проблема осталась неразгаданной и исчезла при переналивке агента
3. На этом агенте использовался JDK 1.7, на остальных уже обновили до 1.8

2.4 Энергопотребление

- Чем замерять?
- Как замерить с достаточной точностью и частотой съёма метрик?
- Как выделить влияние конкретного приложения?
- Как компенсировать зависимость мощности от температуры?

https://store.artlebedev.ru/postcards/grandpa-pihto/

2.5 Разброс замеров энергопотребления



2.6 Косвенное измерение энергопотребления

Чёрная кошка
в чёрной комнате

2.7 Косвенное измерение энергопотребления

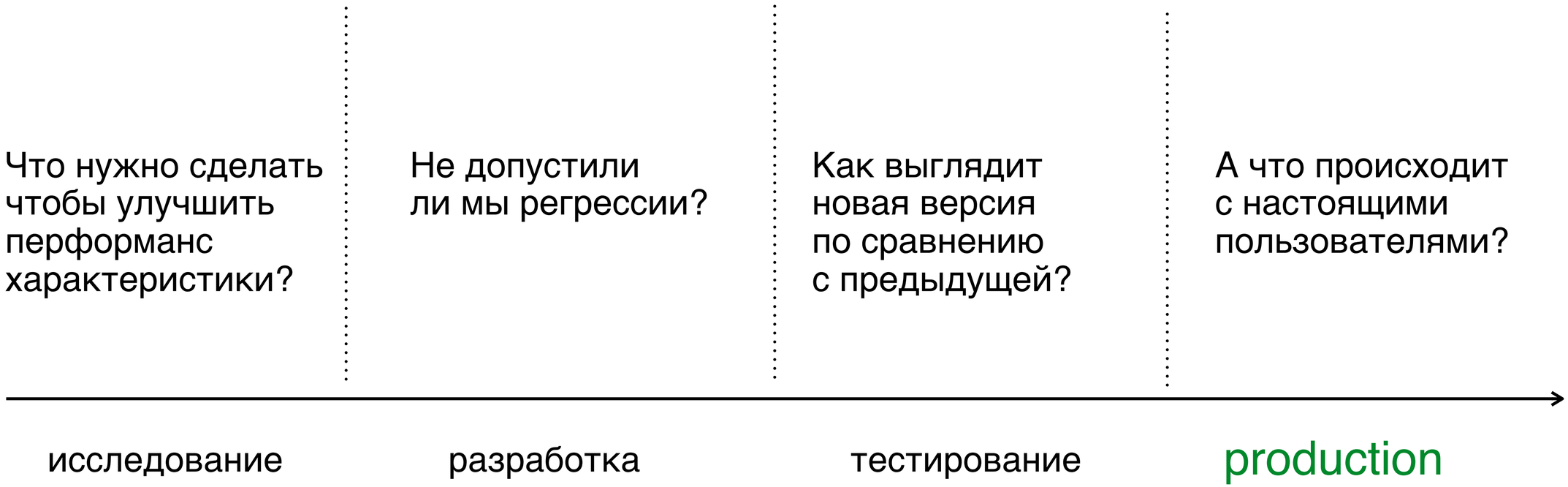
- /proc/[pid]/stat содержит полезную информацию по загрузке процессора
- Эксперименты показывают устойчивую корреляцию CPU-тиков и потребляемой батарейкой
- Имеют ограничение по частоте обновления информации
- Можем пропустить регрессии вызванные работой GPS, радиомодуля и т.д.

2.8 Способы измерения энергопотребления

Способ	Точность	Масштабируемость запуска тестов
CPU тики	**Средняя**	**Высокая**
Fuel Gauge	**Высокая**	**Низкая**
Внешние мониторы (Яндекс.Вольта)	**Очень высокая**	**Средняя**

https://github.com/yandex-load/volta

2.9 Жизненный цикл приложения



Перф-тестирование как часть Continuous Integration

3.1 Инфраструктура перф-тестирования

Ферма устройств

- мобильные устройства
- устройства измерения
- шкафы

High Level CI

- запуск сравнении
- регулярный прогон
- поиск аномалий
- перф-бисект

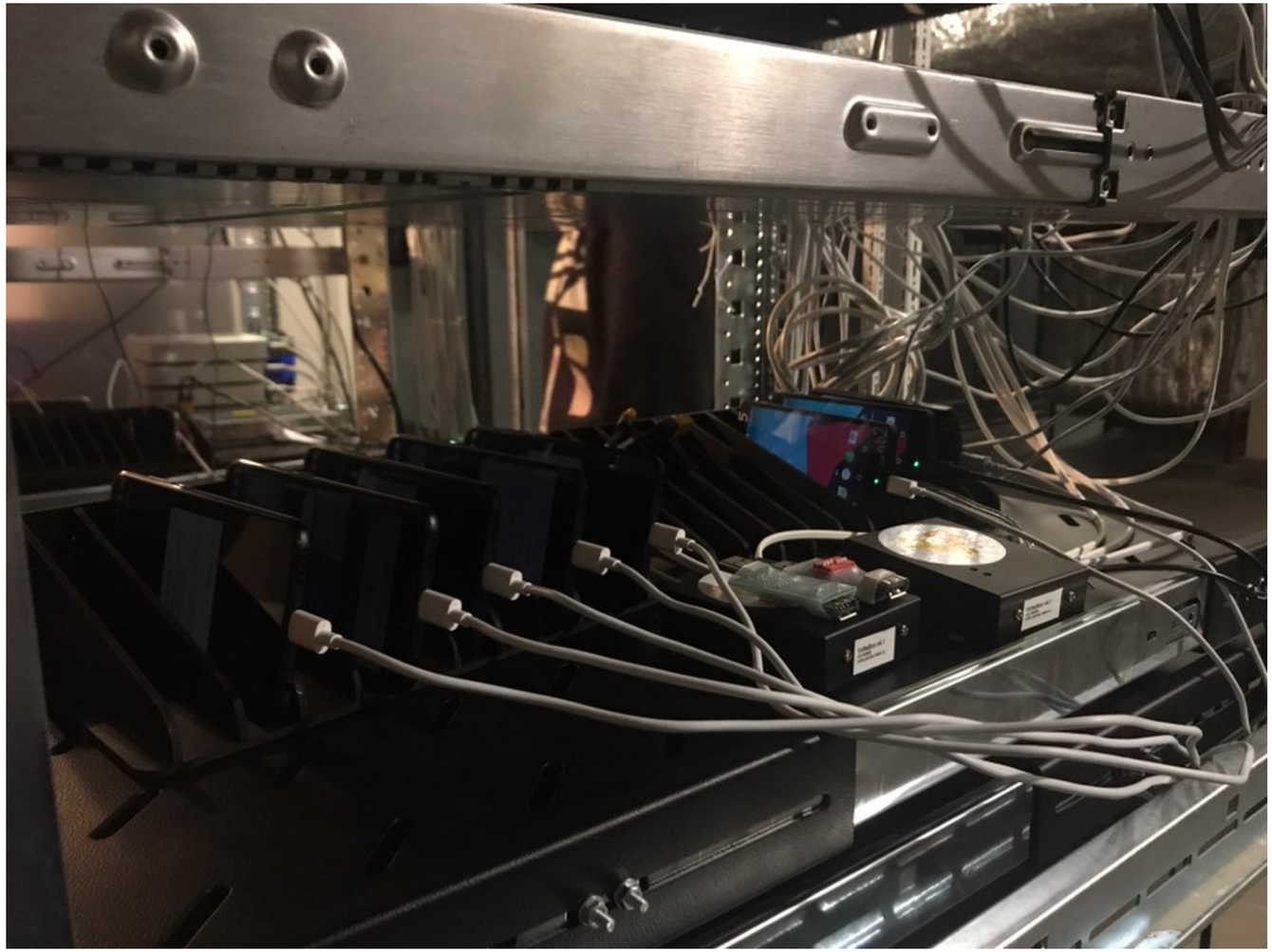
Low Level CI

- калибровка устройств
- мониторинг
- запуск тестов
- очереди сборок

3.2 Ферма мобильных устройств

- Серверные стойки
- До пяти 1U blade server в стойке
- PCI-E USB контроллеры
- До 8 устройств с прямым подключением к серверу

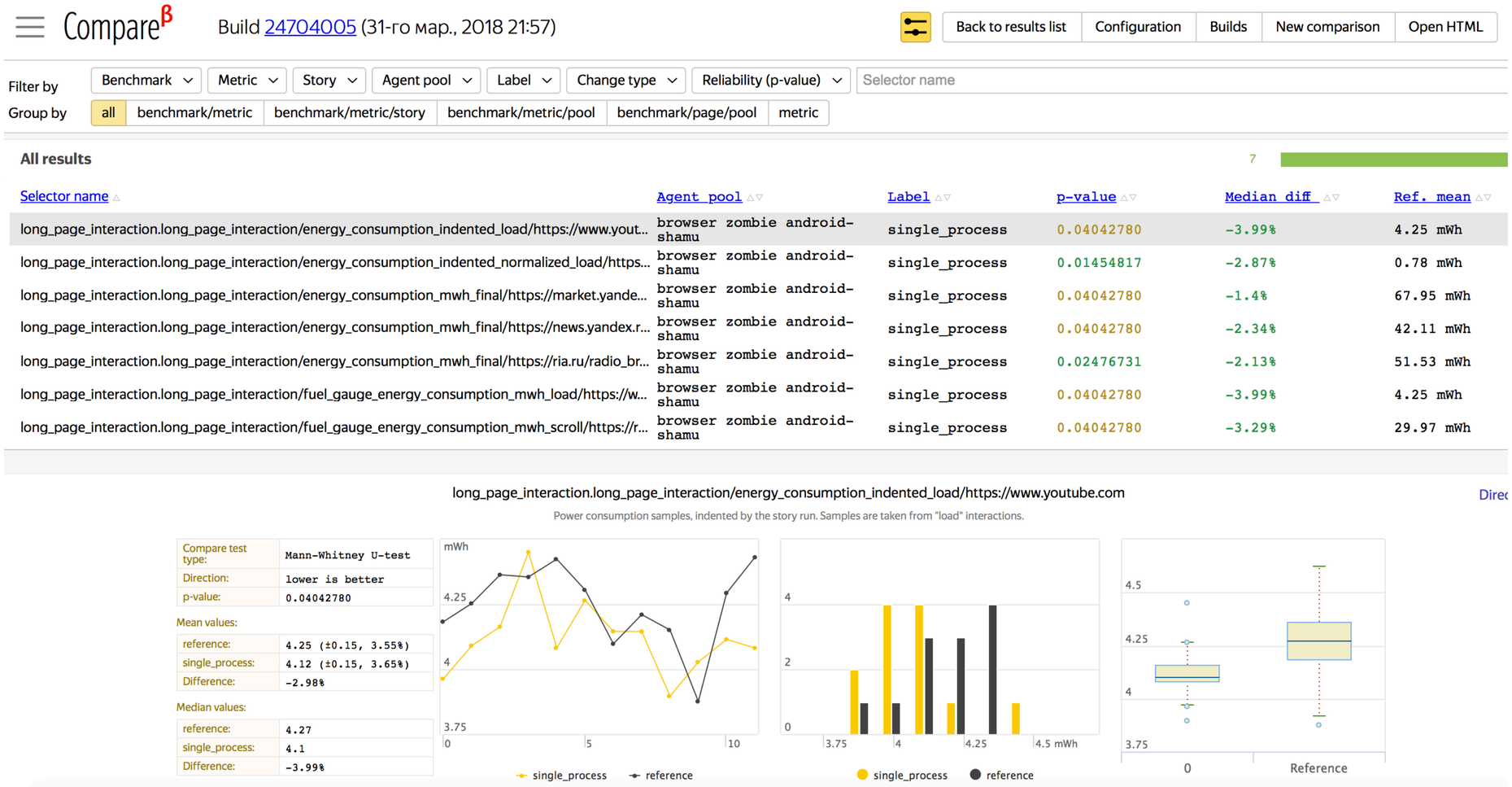
3.3 Ферма устройств: за кулисами



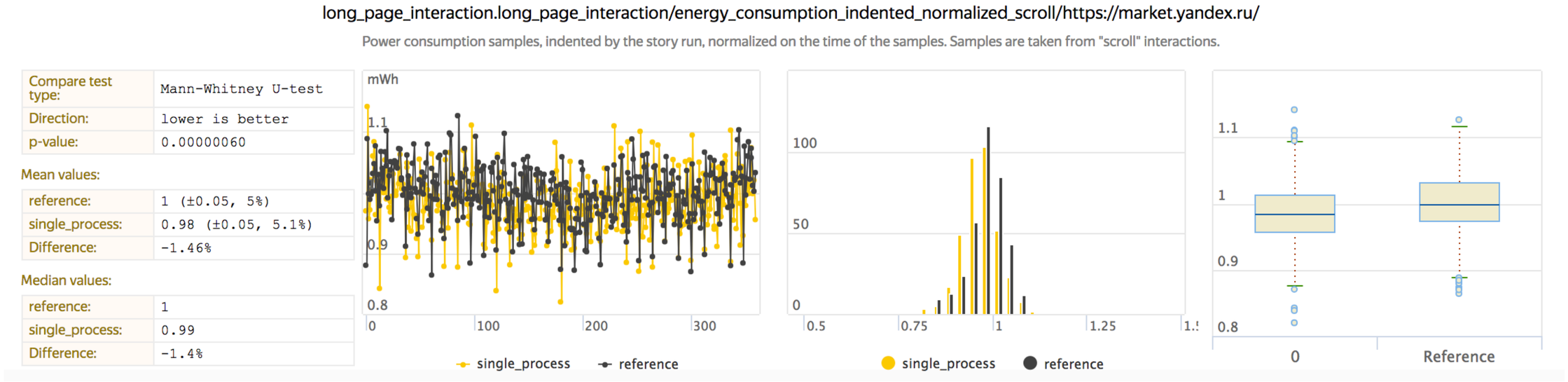
3.4 Устройства или x86 эмуляторы

- По нашим замерам дисперсия значений замеров скорости на x86 выше чем на устройствах от 1.5 до 3 раз
- Скорость прогона теста на x86 эмуляторе может быть как выше, так и ниже чем на реальном устройстве
- Количество прогонов будет расти от 3 до 4 раз
- Стоимость: телефон - 10-15 к₽, dev-board - от 6к₽, хост - ?

3.5 Режим сравнения сборок



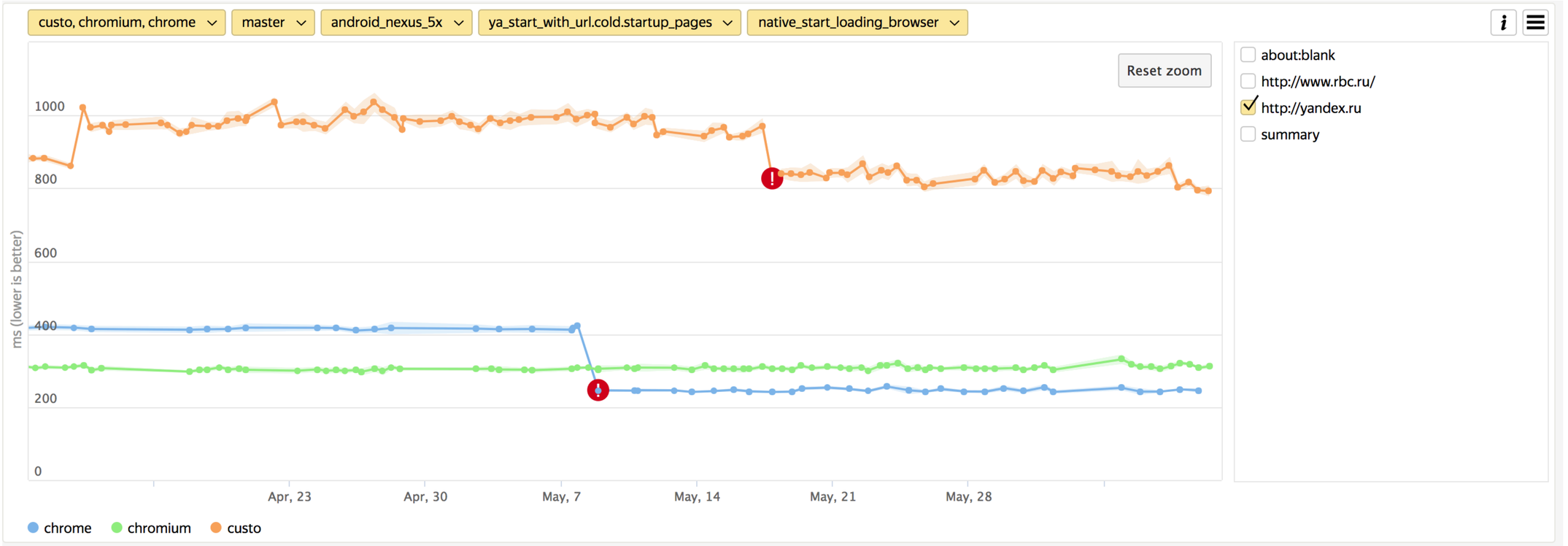
3.6 Статистическая значимость измерений



Получения статистически значимых результатов даже в условиях когда оцениваемые изменения меньше дисперсии измеряемой метрики

3.7 Регулярные прогоны перф-тестов

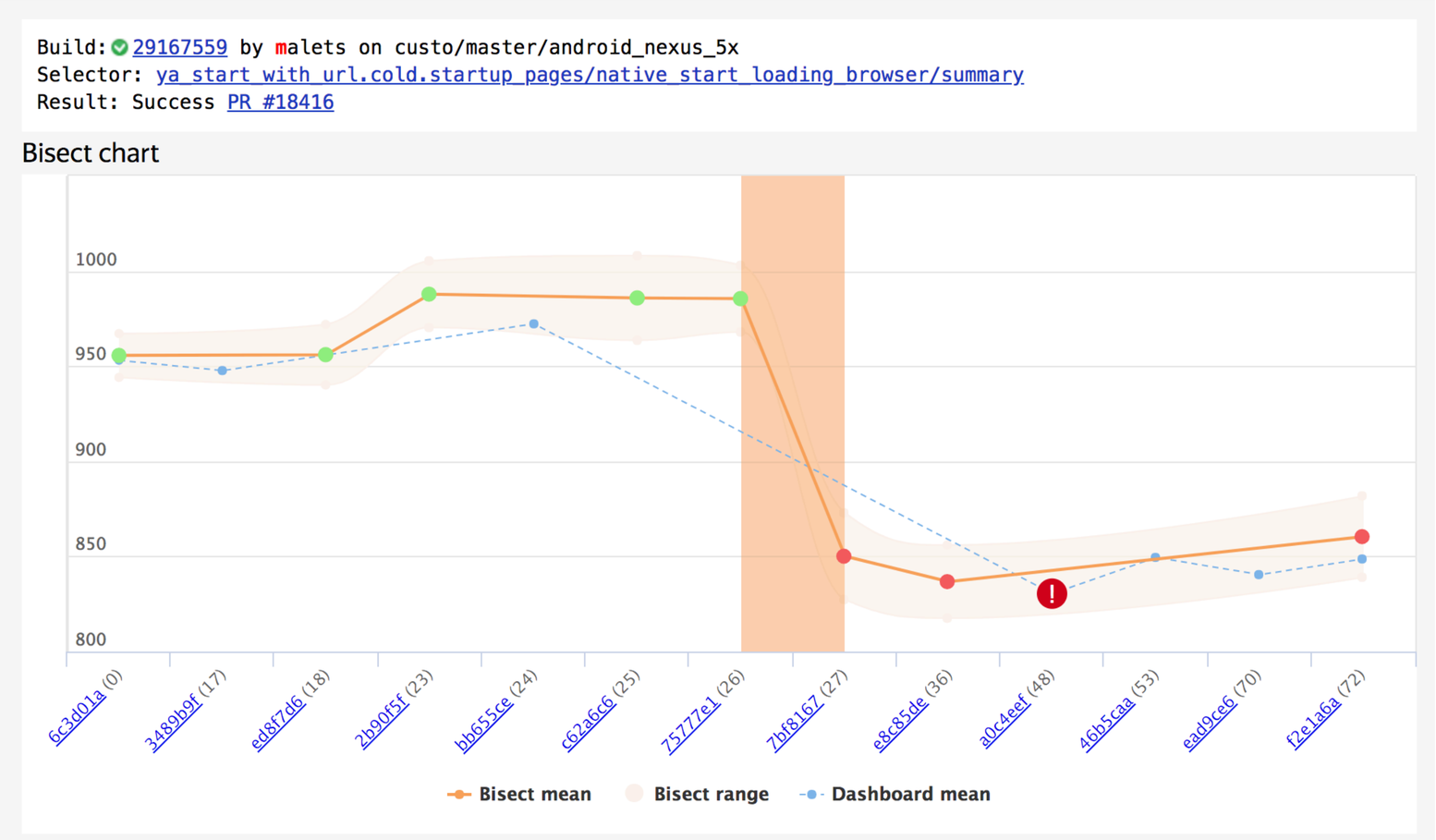
Регулярный прогон перф-тестов с визуализацией на дашбордах



3.8 Автоматический детект аномалий

Обнаружение аномалий с помощью бисекта для выявления коммитов, вызвавших регрессию

3.9 Перф-бисект как основа машстабирования



Рекурсивный прогона перф-бисекта с выявлением набора аномалий

Мониторинг данных после выпуска приложения

4.1 Инфраструктура сбора данных

Клиентские библиотеки

- клиенты для iOS и Android

Первичная обработка (YT)

- MapReduce для пред-обработки данных
- почасовое обновление логов

Данные для анализа (Clickhouse)

- подготов-ленные данные для фронтенда
- выполнение запросов к данным

Фронтенд

- диаграмма релизов
- приемочные карты
- срезы по спекам
- срезы по экспериментам

4.2 Инфраструктура. Проблемы

Проблемы

- Объём дополнительных данных на клиентском устройстве
- Объём трафика между клиентом и сервером
- Объём хранилища на сервере
- Объём операций для обработки данных
- Скорость анализа данных по кастомным запросам

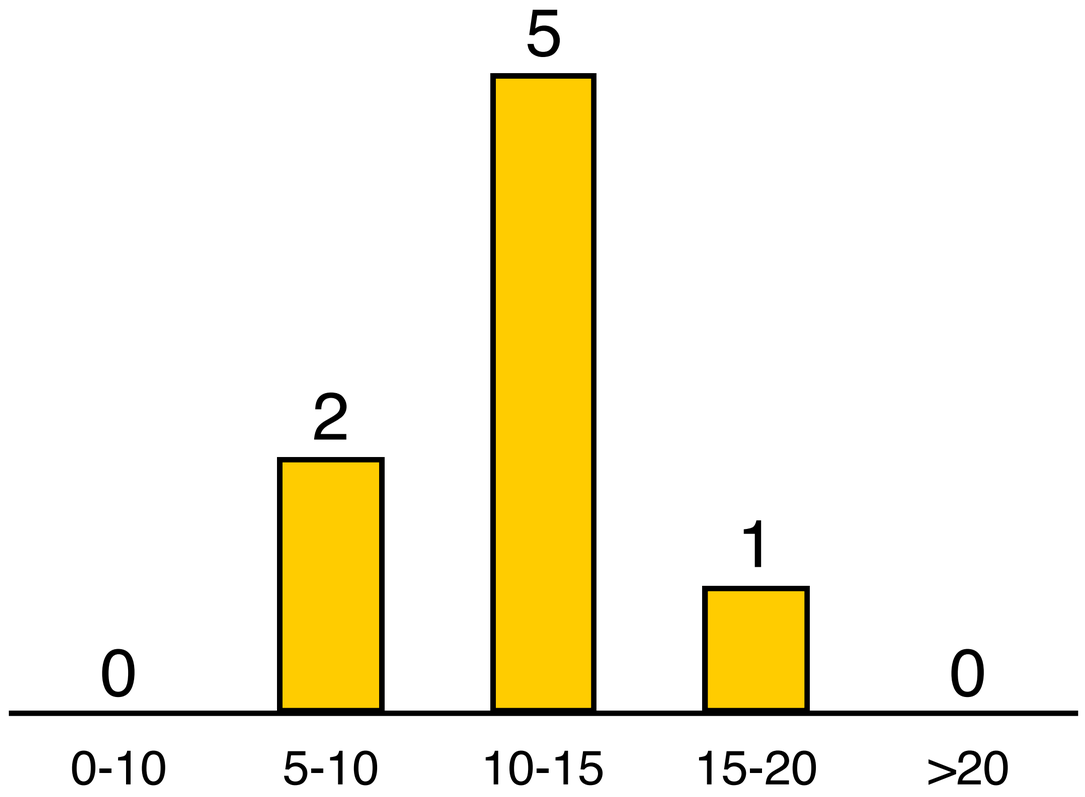
Решения

- Дискретизация
- Агрегация на клиенте
- Семплирование
- Предрасчеты

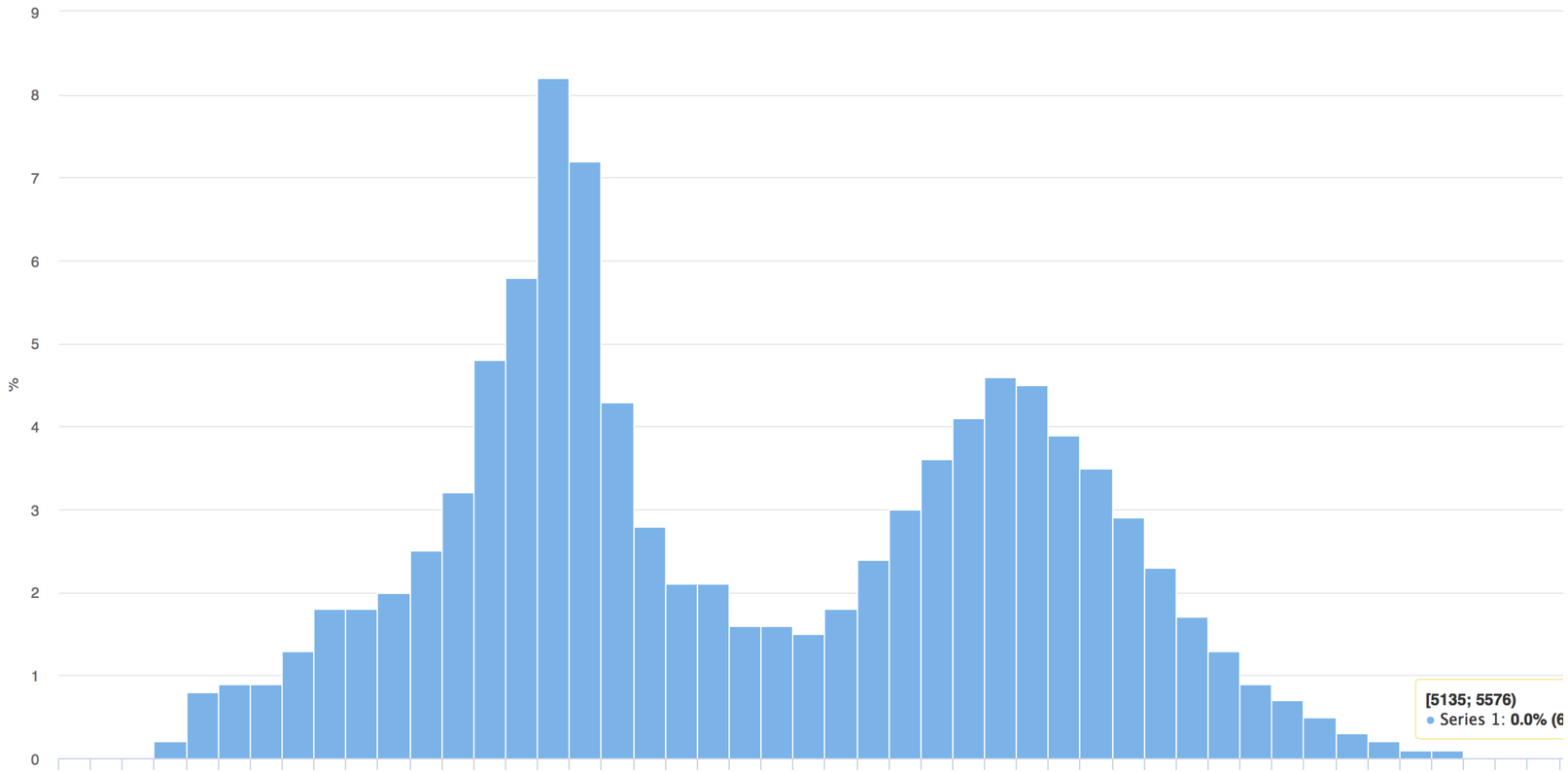
4.3 Агрегация данных

```
{
   name: «HowFastIsMyApp»,
   buckets: [
     {min:  5, max: 10, count: 2},
     {min: 10, max: 15, count: 5},
     {min: 15, max: 20, count: 1}],
   sum: 55
}
```

avg = sum / sum(bucket.count)



4.4 Агрегация данных: бакеты



- Равномерное распределение
- Экспоненциальное распределение
- Фиксированный набор значений

4.5 Агрегация данных. Проблемы

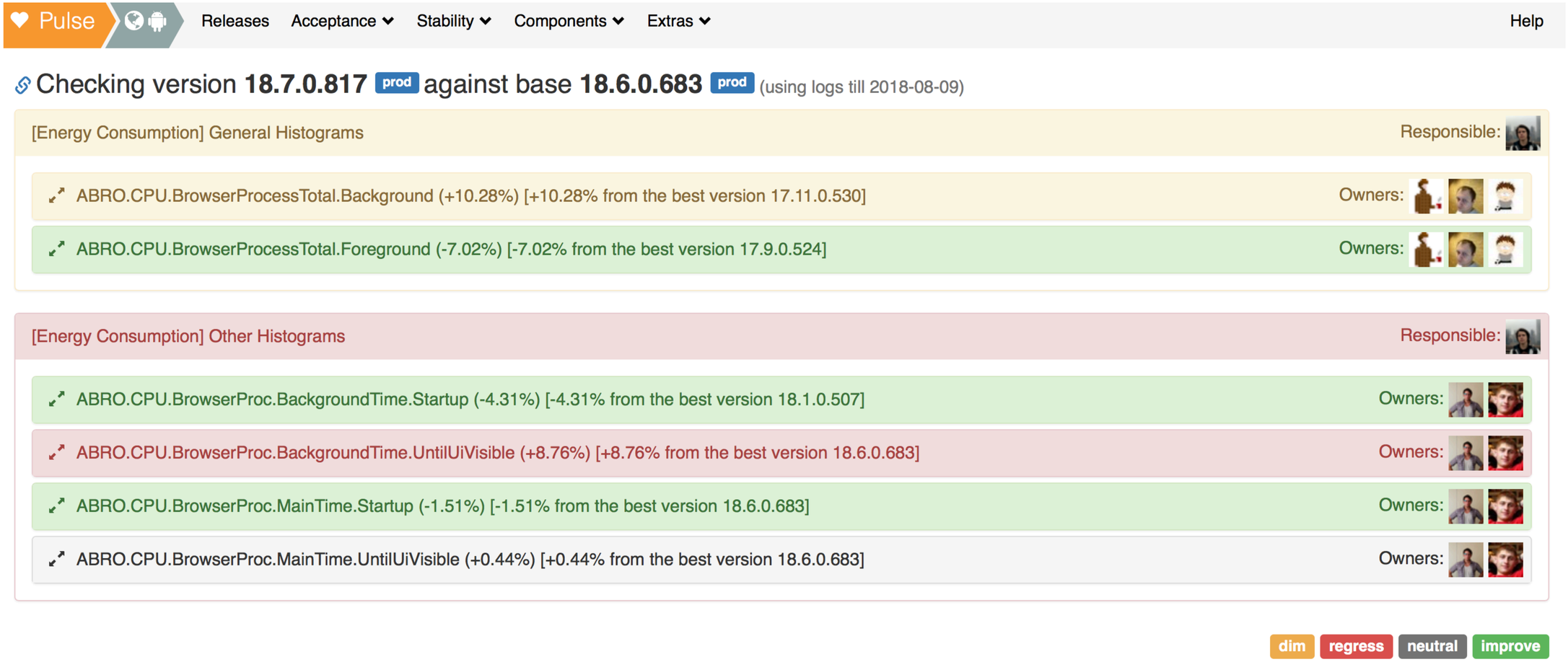
Проблемы

- Выбор границы последнего бакета
- Баланс между точностью съёма метрик и объемом данных/ вычислительной сложностью обработки
- Работа с гистограммами с фиксированным набором значений

Решения

- Закладывать высокую границу последнего бакета
- Экспоненциальные и кастомные границы бакетов

4.6 Приёмка версий. Что мы хотим



4.7 Приёмка версий. Проблемы

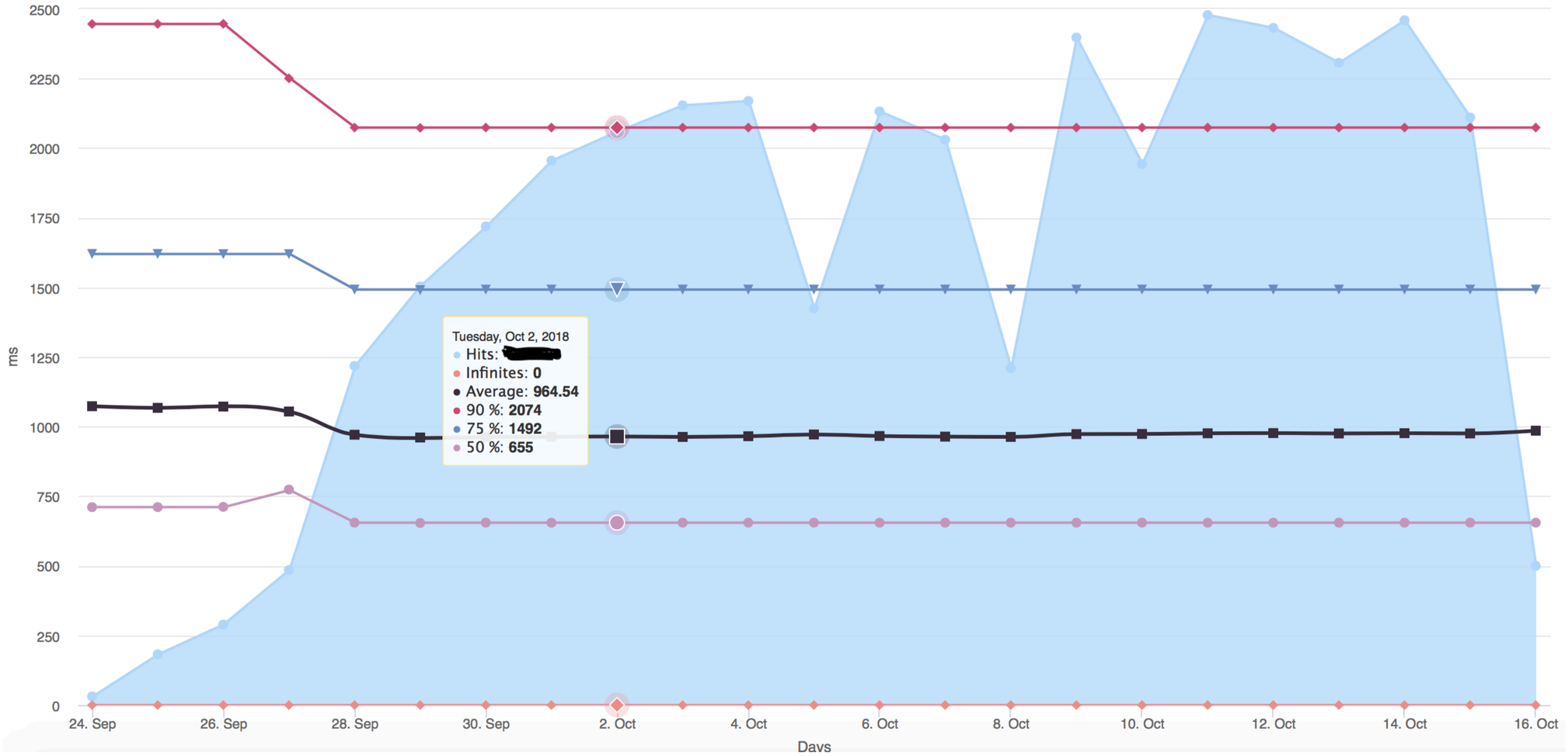
Проблемы

- Единая кумулятивная метрика может быть слишком чувствительной или недостаточно чувствительной
- Выбор периода сравнения влияет на результаты
- Стабилизация метрик требует времени

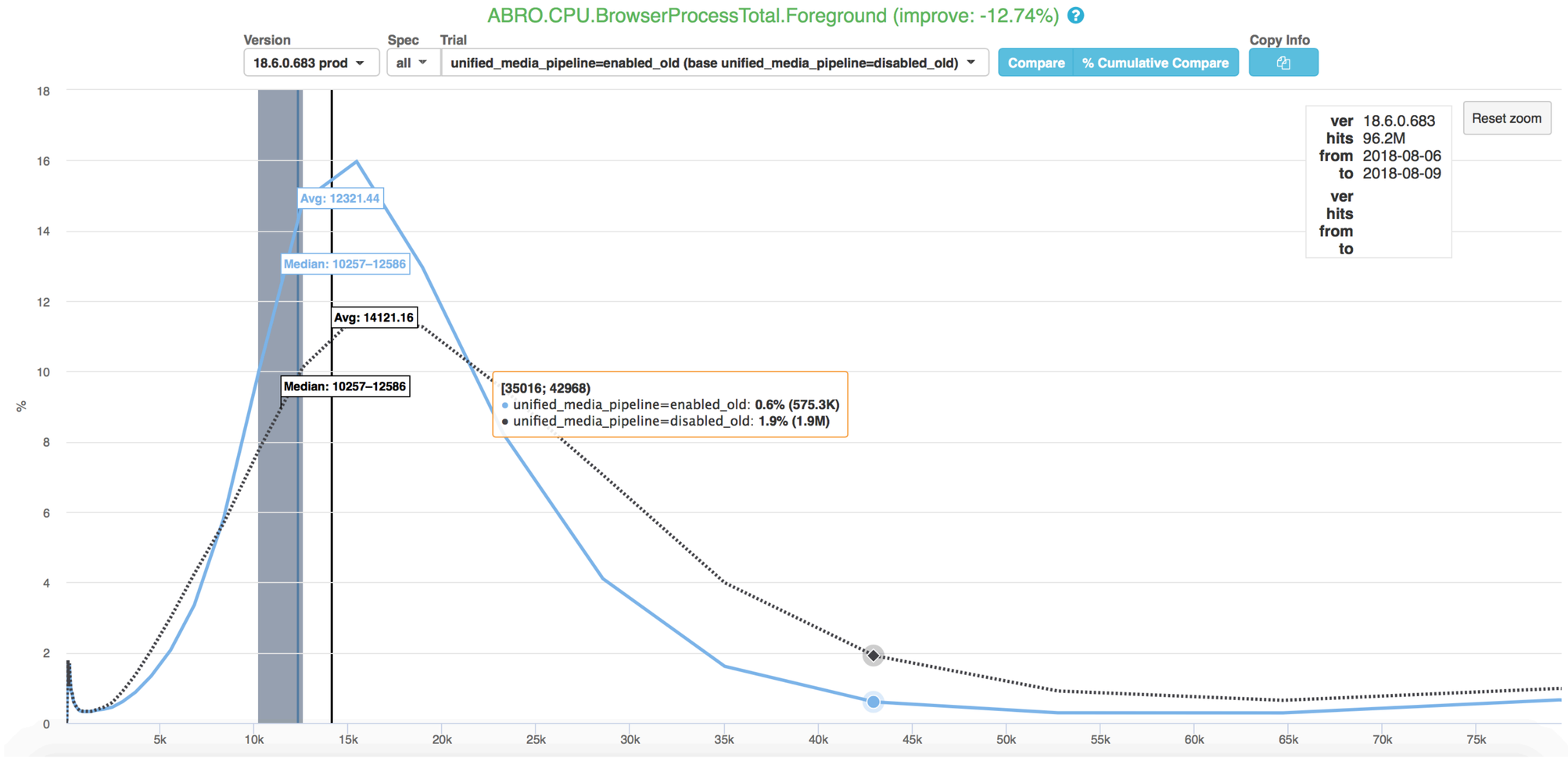
Решения

- Среднее + медиана/анализ персентилей
- Выравнивание по графикам раскатки
- Cross join аудитории

4.8 Анализ данных: персентили



4.9 Приёмка эксперимента в production



4.10 Просмотр данных по срезам

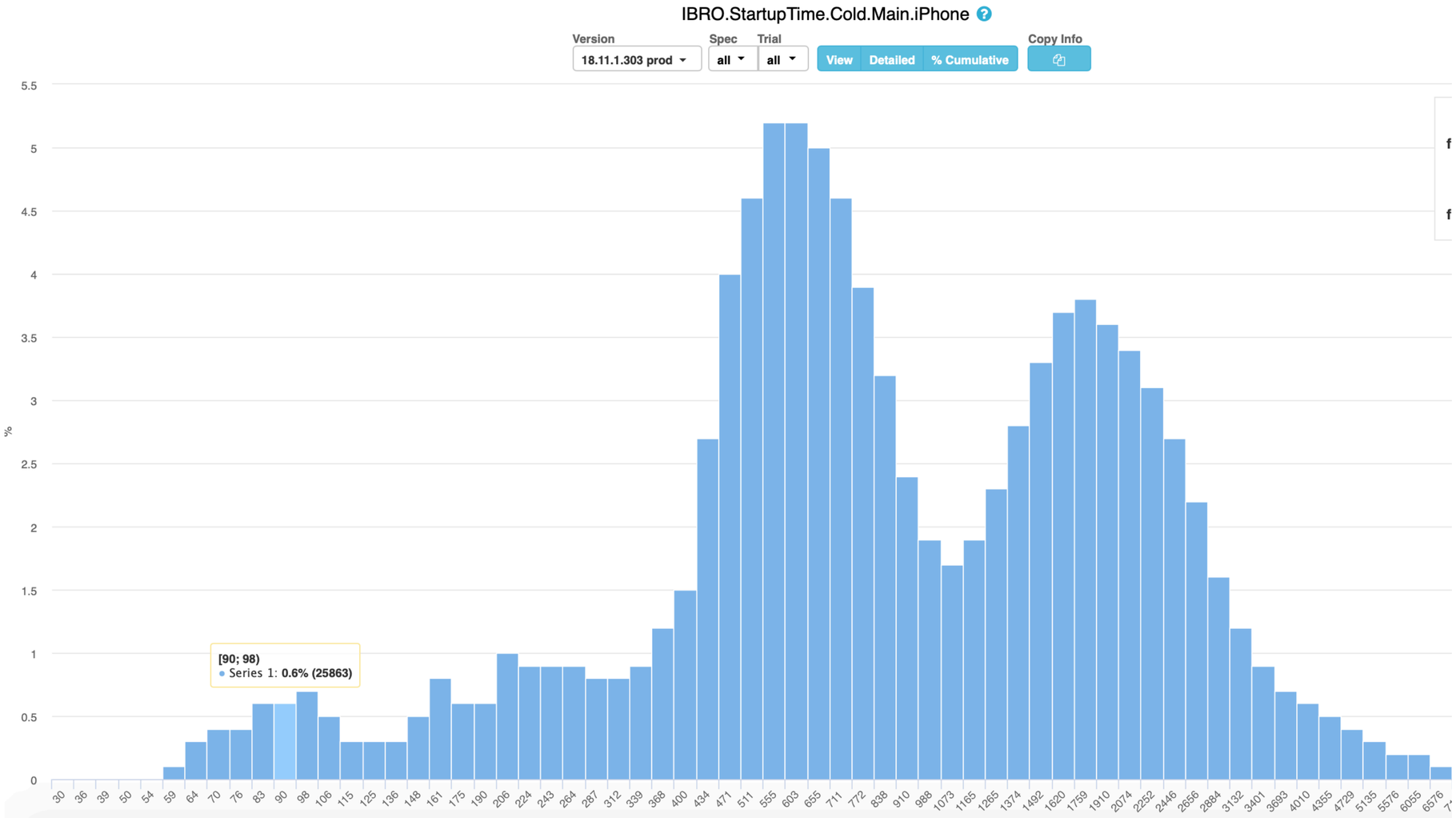
cpu_cores	cpu_cores=1 +6.03% 2498 / 939107 (0.27% of required)	cpu_cores=2 -1.30% 38462 / 939107 (4.10% of required)	cpu_cores=4 -7.64%	cpu_cores=8 -5.93%
cpu_type	cpu_type=aarch64 -14.87% 140514 / 939107 (14.96% of required)	cpu_type=armv7l -5.49%	cpu_type=armv8l -7.72%	cpu_type=x86 -10.42% 16490 / 939107 (1.76% of required)
ram	ram=0.5 -4.55% 17645 / 939107 (1.88% of required)	ram=1 -3.78% 374305 / 939107 (39.86% of required)	ram=2 -10.30% 716752 / 939107 (76.32% of required)	ram=3 -5.61% 554993 / 939107 (59.10% of required)
gpu	gpu=Adreno (TM) 405 -13.98% 38426 / 939107 (4.09% of required)	gpu=PowerVR SGX 544MP +3.18% 19299 / 939107 (2.06% of required)	gpu=Mali-450 MP -15.73% 35240 / 939107 (3.75% of required)	gpu=Mali-T720 -11.13% 395287 / 939107 (42.09% of required)
screen	screen=3.9 -0.68% 49724 / 939107 (5.29% of required)	screen=4.6 -7.60%	screen=7.4 +10.23% 47294 / 939107 (5.04% of required)	screen=9.4 -4.44% 117715 / 939107 (12.53% of required)
os_major	os_major=4 +0.52% 264102 / 939107 (28.12% of required)	os_major=5 -7.68% 334522 / 939107 (35.62% of required)	os_major=6 -14.11% 513188 / 939107 (54.65% of required)	os_major=7 -6.91% 732590 / 939107 (78.01% of required)

4.11 Просмотр данных по срезам. Проблемы

Проблемы

- Списки моделей девайсов/CPU/GPU очень объемны и постоянно изменяются
- Срезы с малым количеством хитов могут шуметь
- Аддитивная (для независимого) либо мультипликативная (при использовании с другими срезами) сложность расчетов
- Фиксированные Топ-X с периодическим обновлением
- Авторасчёт отклонений
- Замена предрасчёта на пострасчёт

4.12. Двугорбое распределение StartTime

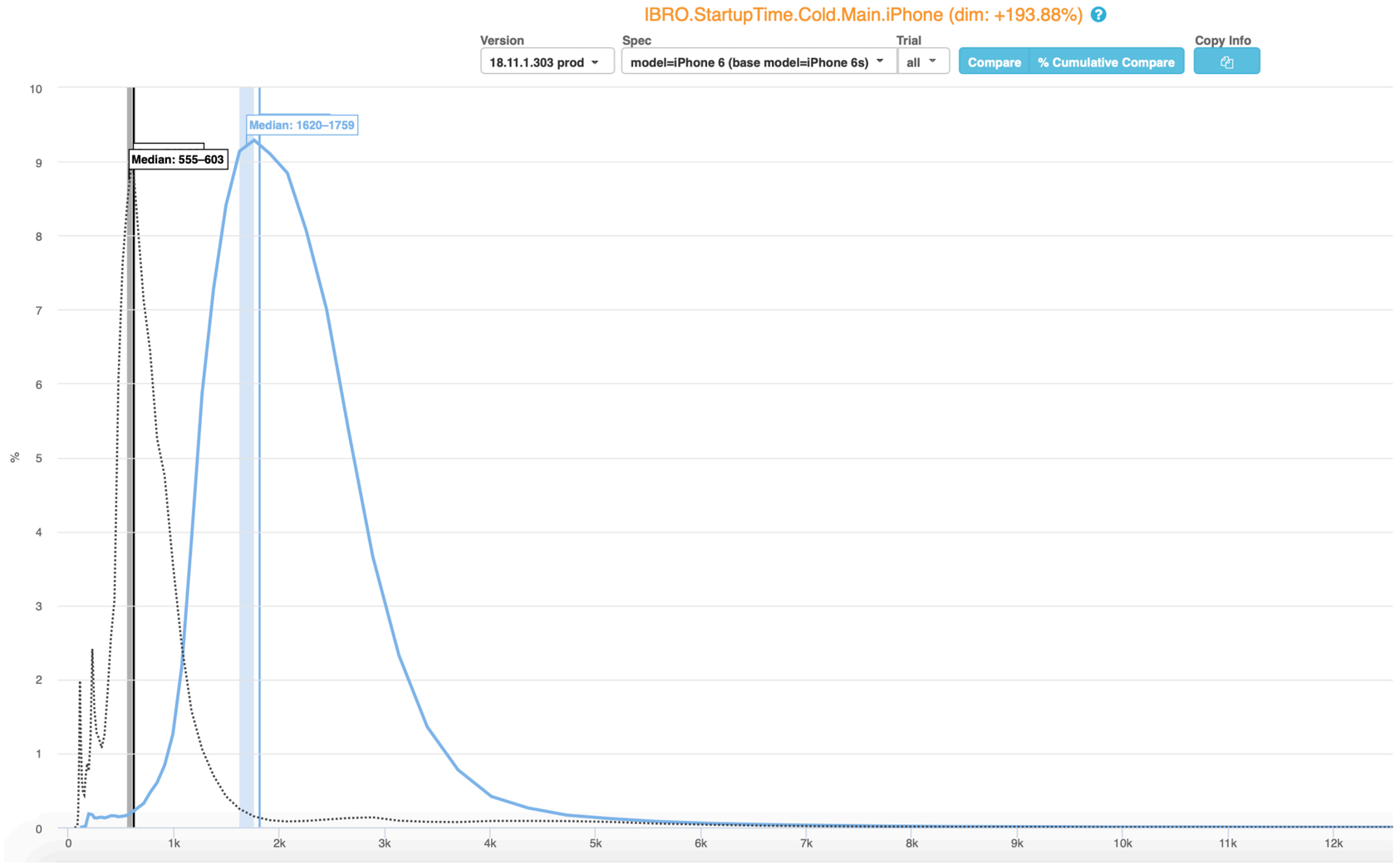


4.13 Внимание: вопрос!

> Чем обусловлены два пика на всех гистограммах времени старта на iOS?

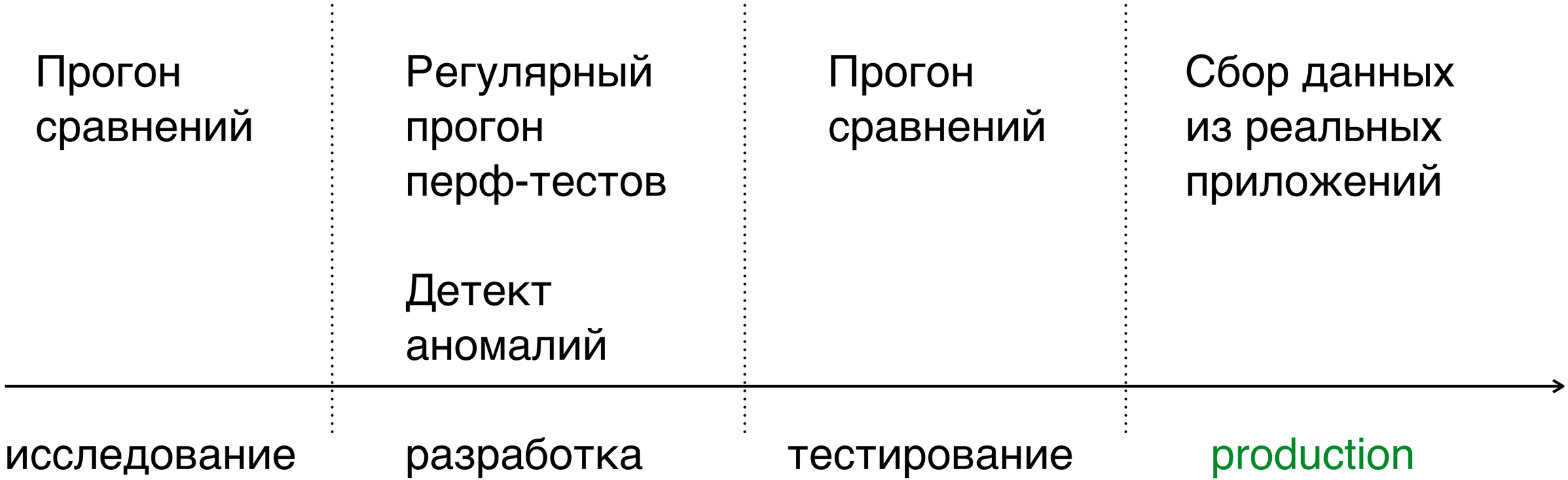
1. iOS 11 дала значительный буст производительности
2. Москва и Санкт-Петербург различаются не только произношением некоторых слов но и скоростью запуска приложения
3. Устройства до iPhone 6s значительно хуже новых устройств

4.14 Apple A8 -> A9 (20нм -> 14 нм)



Что нам стоит дом построить…

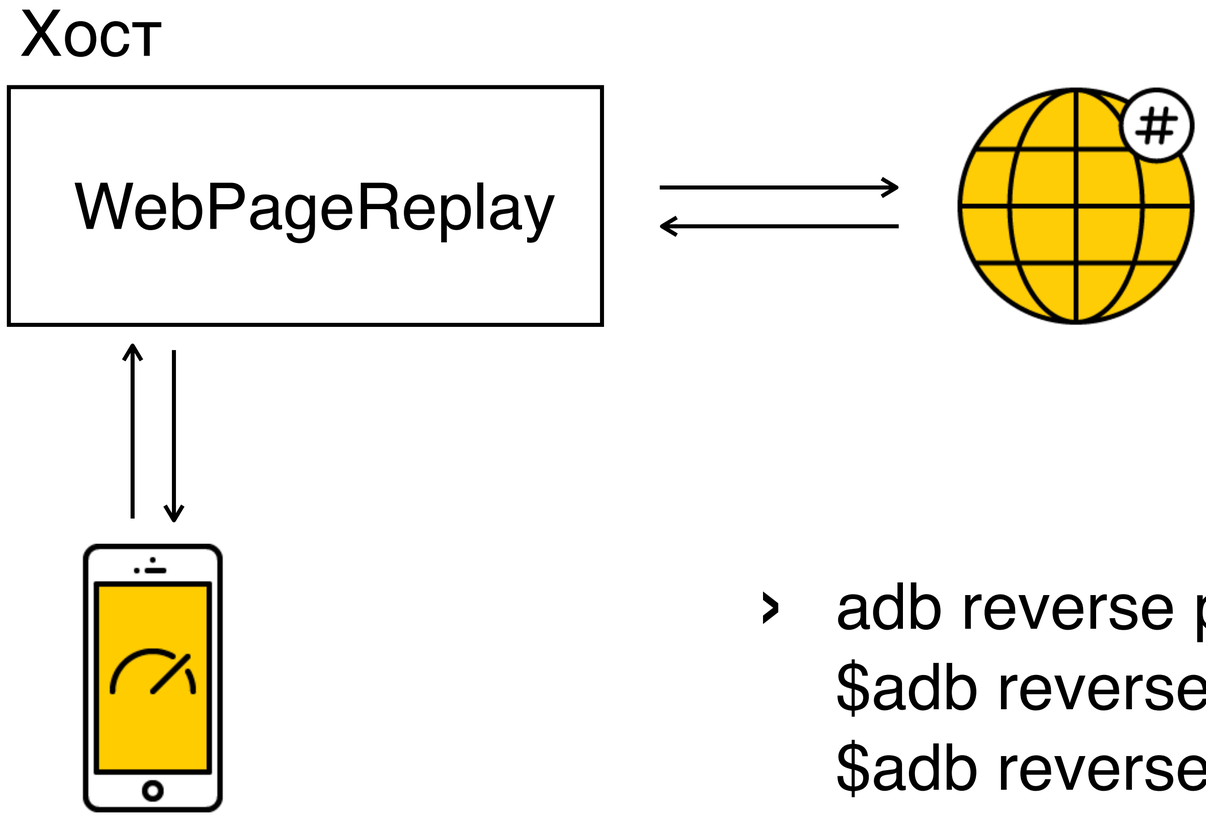
5.1 Жизненный цикл приложения



5.2 Создание инфраструктуры перф-тестов

- Один хост и одно устройство — хорошая база для старта
- Выбор девайсов и их калибровка — 80% успеха
- Каждый коммит должен содержать все необходимое для прогона теста
- Сначала режим сравнения, потом регулярные прогоны
- Сеть — большой источник шума и угроза воспроизводимости результатов
  Необходимо либо мокать, либо предзаписывать траффик

5.3 Проксирование сетевого взаимодействия



- adb reverse port forwarding
  $adb reverse tcp:8080 tcp:8080
  $adb reverse tcp:8081 tcp:8081

https://github.com/catapult-project/catapult/blob/master/web_page_replay_go/README.md

5.4 Сбор данных от реальных пользователей

Клиентская часть

- Агрегируйте данные на клиенте
- Семплируйте клиентов для снижения объемов
- Начните с ограниченного набора простых метрик (время старта)

5.5 Сбор данных от реальных пользователей

Серверная часть

- Запустите регулярные предрасчеты в MapReduce
- Сохраняйте результаты предрасчетов (MongoDB etc.)
- Поднимите простой фронтенд (Node.JS, Flask)
- Используйте JS-библиотеки для визуализации (Higcharts)

6.1 Внимание: вопрос!

После этого доклада:

1. Мы усовершенствуем свою текущую систему контроля перформанс-характеристик воспользовавшись советами
2. Мы создадим новую систему контроля перформанс-характеристик и улучшим производительность нашего приложения
3. Ой, доклад уже закончился? А я все проспал… Ладно, потом узнаю у соседа о чем же шла речь :-)

Илья Богин
Руководитель службы разработки ядра мобильного браузера

 lafkadio@yandex-team.ru

 @ilya_bogin