Тестирование производительности React компонентов в CI

SBER DEVICES

???

perftool

github.com/salute-developers/perftool

Я Артём

Senior Dev

2.5+ лет в СберДевайсах

3.5+ лет в Яндекс.Маркете

тех и продуктовые команды

сейчас занимаюсь скоростью фронтовых сервисов

О чем сегодня поговорим

История

Как работает

Статистика

Лечение нестабильности

Интеграции

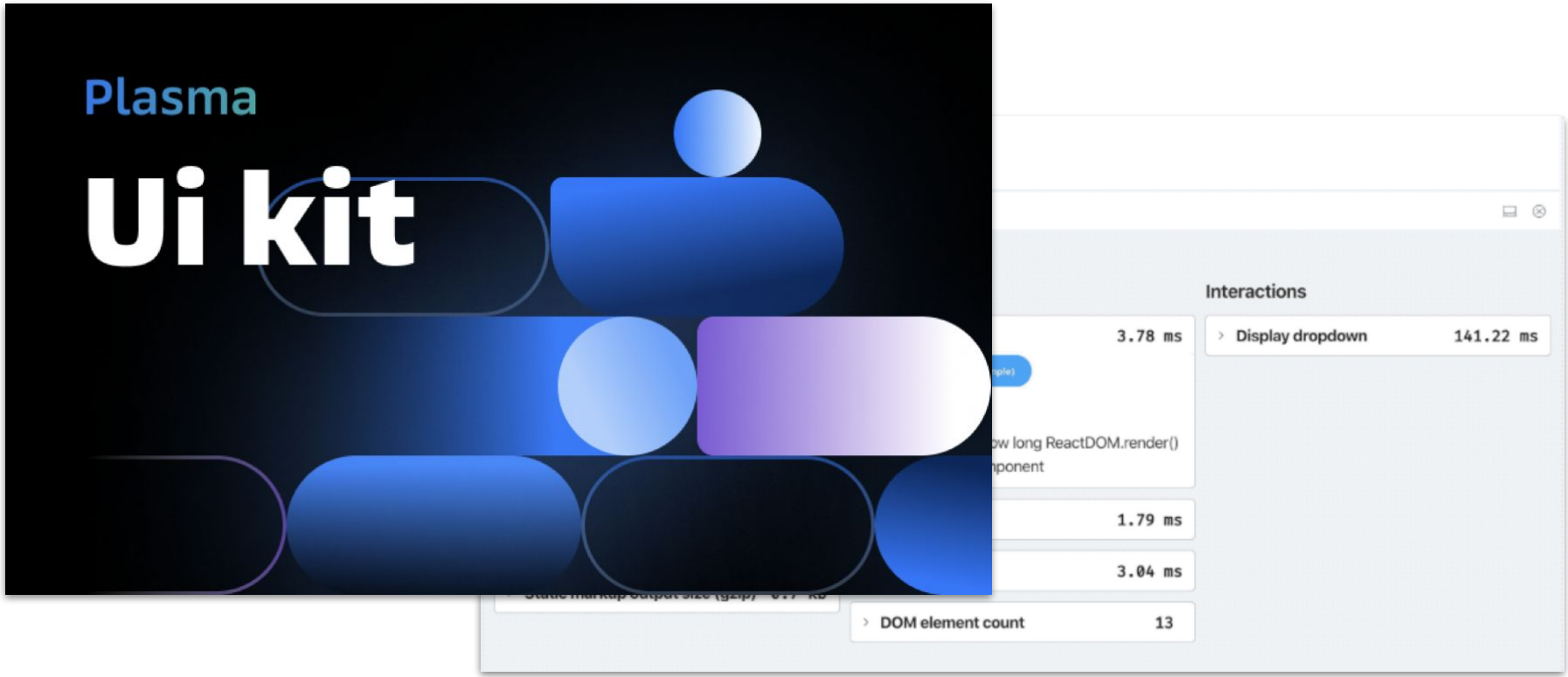
Пример работы

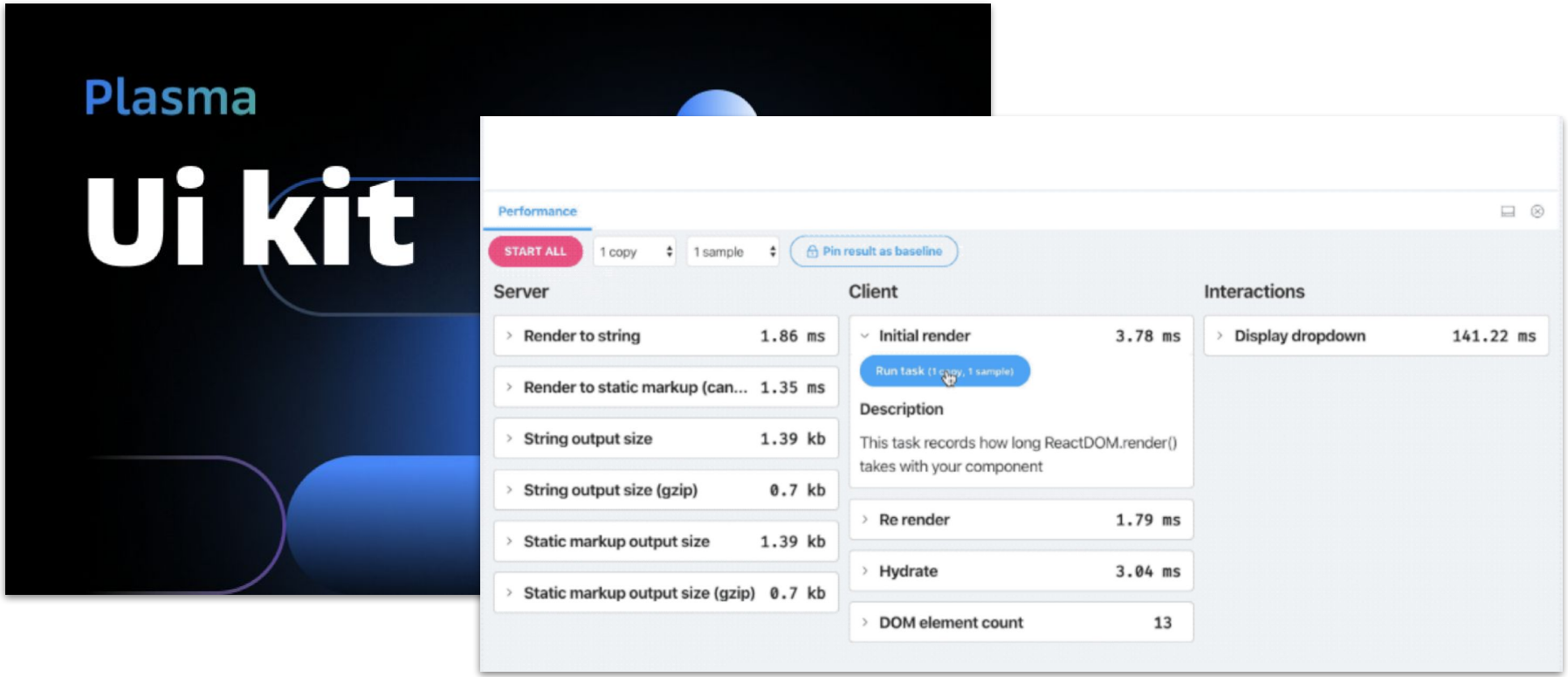
Предпосылки

Метрики

- Лабораторные
- Полевые

Немного истории


Plasma
Ui kit
Interactions
3.78 ms
Display dropdown
141.22 ms
ow long ReactDOM.render()
nponent
1.79 ms
3.04 ms
DOM element count
13


Plasma
Ui kit
Performance
START ALL
1 copy
1 sample
Pin result as baseline
Server
Render to string 1.86 ms
Render to static markup (can... 1.35 ms
String output size 1.39 kb
String output size (gzip) 0.7 kb
Static markup output size 1.39 kb
Static markup output size (gzip) 0.7 kb
Client
Initial render 3.78 ms
Run task (1 copy, 1 sample)
Description
This task records how long ReactDOM.render() takes with your component
Re render 1.79 ms
Hydrate 3.04 ms
DOM element count 13
Interactions
Display dropdown 141.22 ms

Где разместить

- precommit hook?

Где разместить

- precommit hook?
- release pipeline?

Где разместить

- precommit hook?
- release pipeline?
- PR!

Критерии к инструменту

Работает в CI, прогоняется в PR/MR

Критерии к инструменту

Работает в CI, прогоняется в PR/MR

Измеряет производительность в браузере (native env)

Критерии к инструменту

Работает в CI, прогоняется в PR/MR

Измеряет производительность в браузере (native env)

Стабильная, не стреляет слишком много

Критерии к инструменту

Работает в CI, прогоняется в PR/MR

Измеряет производительность в браузере (native env)

Стабильная, не стреляет слишком много

Измеряет не только время рендеринга, но и асинхронные изменения в дочерних элементах, изменения в DOM

Критерии к инструменту

Работает в CI, прогоняется в PR/MR

Измеряет производительность в браузере (native env)

Стабильная, не стреляет слишком много

Измеряет не только время рендеринга, но и асинхронные изменения в дочерних элементах, изменения в DOM

Работает с React 18

	storybook-addon-performance	reassure	react-performance-testing	FluentUI performance testing	**perftool**
Работает в CI без приседаний	❌	✅	✅	✅	✅
Рендерит в браузере	✅	❌	❌	✅	✅
Не стреляет на каждый чих, стабильная	❌	⚠️	⚠️	✅	✅
Измеряет не только рендер, но и DOM изменения	❌	❌	❌	❌	✅
React 18	❌	✅	❌	✅	✅
Активно поддерживается	❌	✅	❌	⚠️	✅
Есть документация	✅	✅	✅	❌	✅

Fluent UI

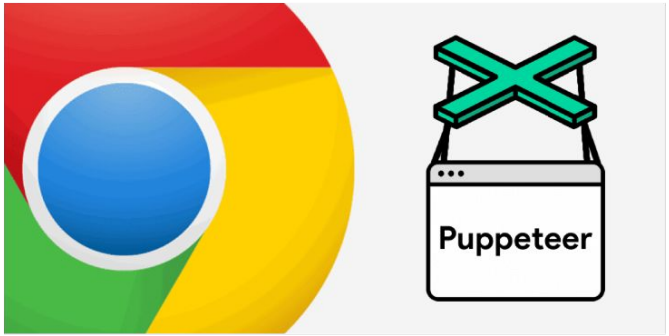
«ticks» vs ms

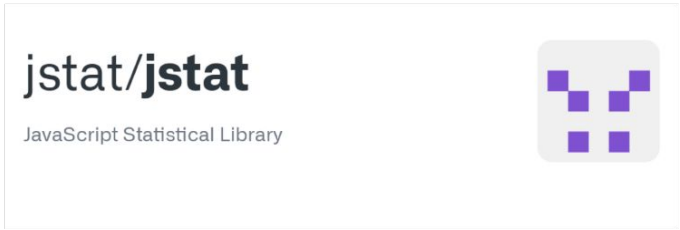
Архитектура и что вообще под капотом


TS
ESM


TS node


webpack


Puppeteer


jstat/jstat
JavaScript Statistical Library


Express
JS

Про статистику

Андрей Акиньшин

Программист, автор постов, статей про математику и программирование, книг по бенчмаркингу

Занимается производительностью приложений в JetBrains

aakinshin.net

Метрики

- Mean
- Median

Метрики

- Mean
- Median
- Geometric Mean
- Harmonic Mean
- Interquartile range
- Interdecile range
- Midhinge
- Trimean
- 5-25% Truncated Mean
- Hodges-Lehmann estimator

Метрики

- Mean
- Median
- Geometric Mean
- Harmonic Mean
- Interquartile range
- Interdecile range
- Midhinge
- Trimean
- 5-25% Truncated Mean
- Hodges-Lehmann estimator

Определение значимости изменения

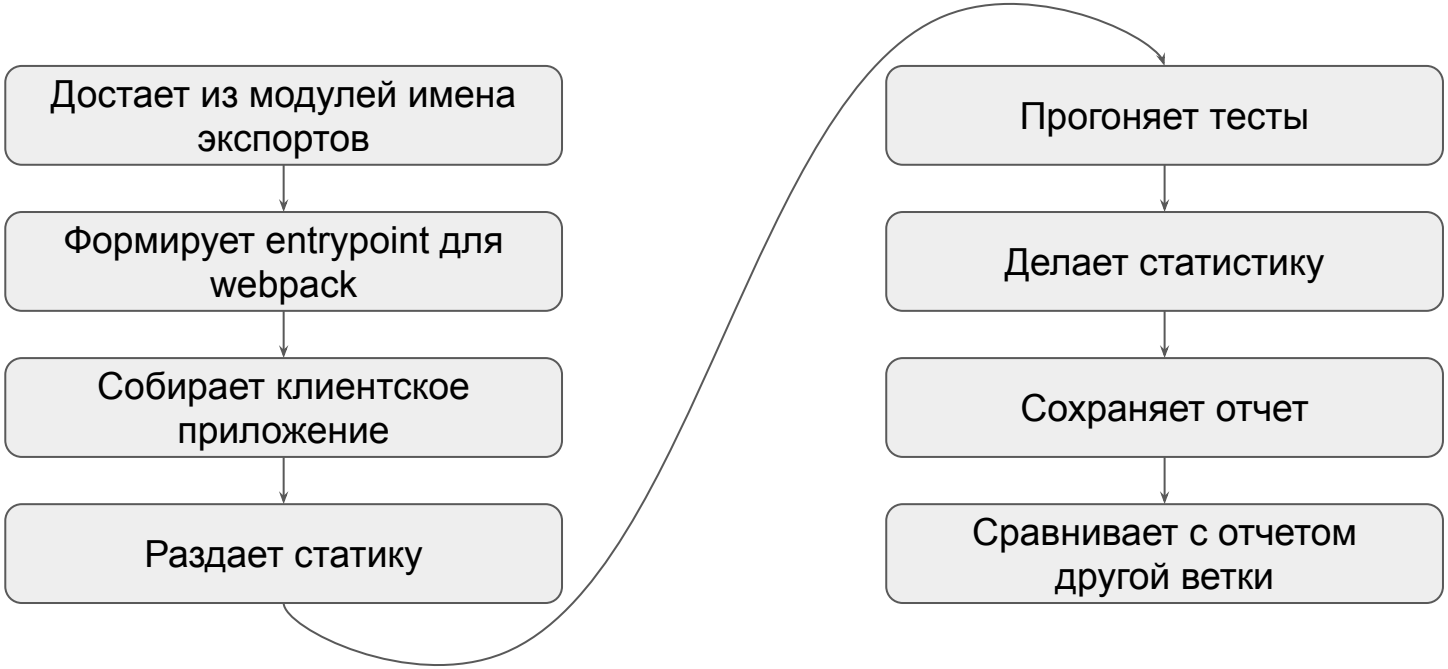
- .99-доверительный интервал по статистике
- На основании пересечения интервалов

Анализ мод

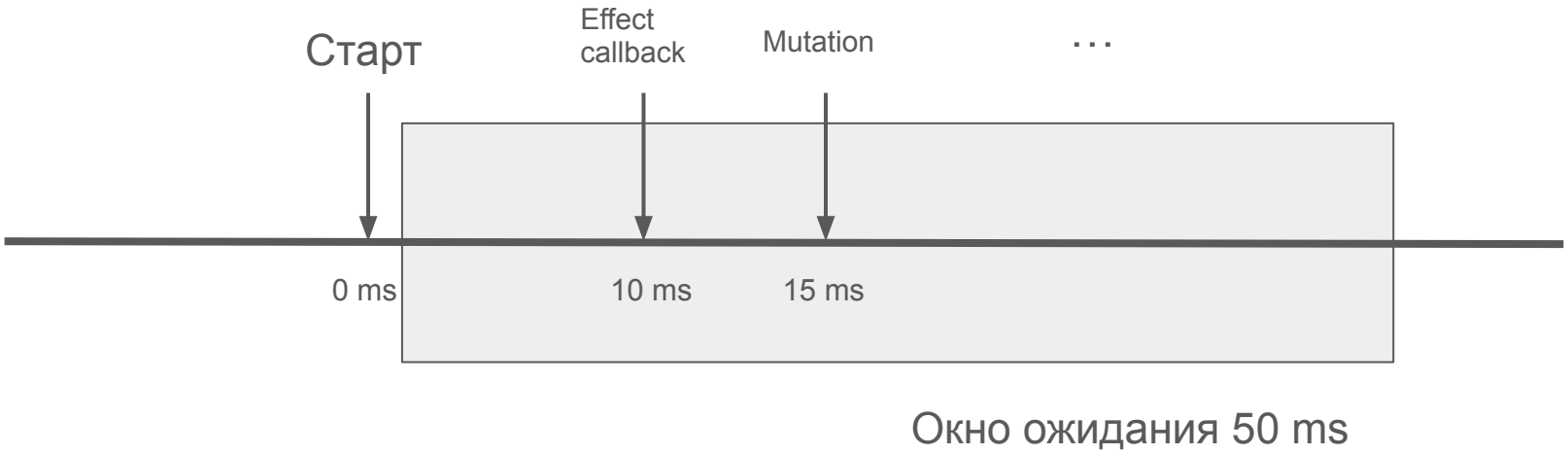
- Квантильная оценка Харрела-Дэвиса
- Параметрическое дрожание для одинаковых значений
- Авторская методика Андрея для определения модальностей

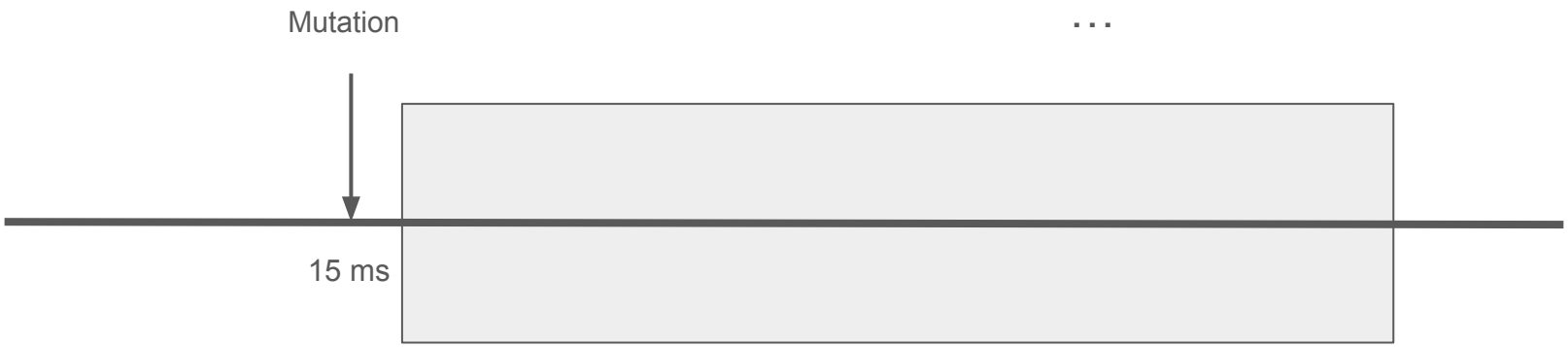
  (Lowland multimodality detection)

Общая схема работы



useEffect + mutationObserver


Старт
Effect
callback
Mutation
...
0 ms
10 ms
15 ms
Окно ожидания 50 ms



Окно ожидания 50 ms

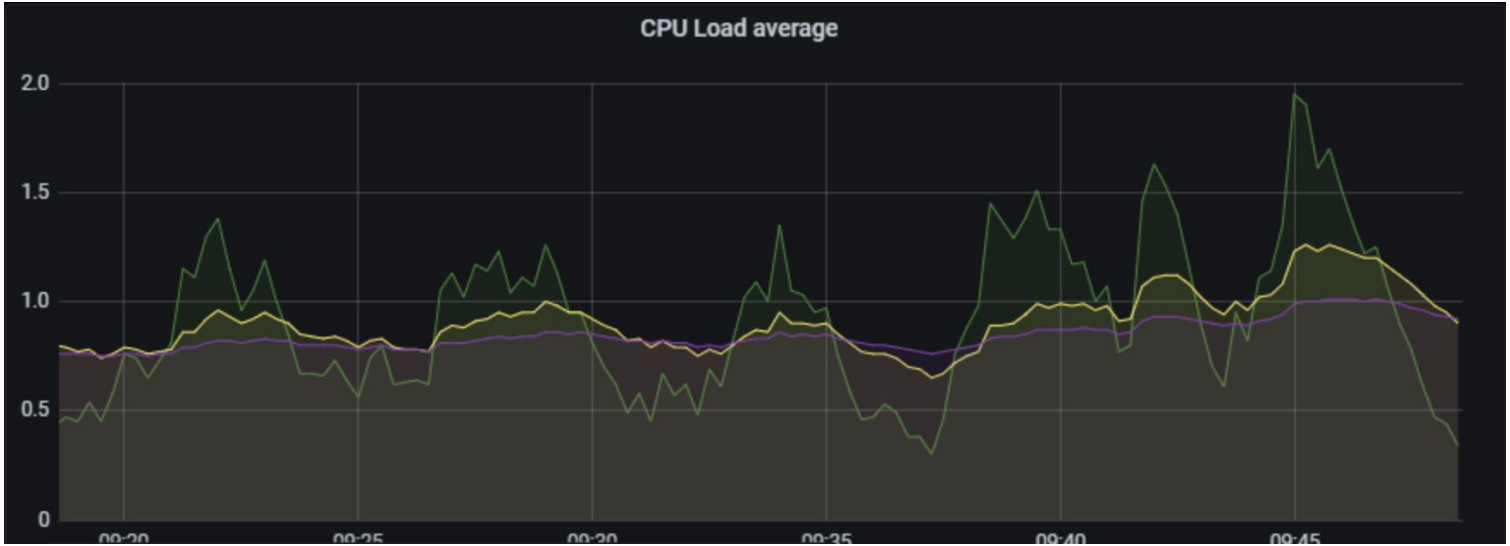
render = 15ms

Удобные фичи для написания тестов

- beforeTest
- перехват запросов
- установка размера и типа вьюпорта

Стабилизация

Внешняя нагрузка изменчива



Нельзя кэшировать результаты мастер ветки

Нельзя кэшировать результаты мастер ветки

- Нагрузка меняется в течение короткого времени, на больших промежутках времени изменения могут быть очень большими
- Плохие мастер отчеты стабильно выдают ложные результаты

Маленькие компоненты

Маленькие компоненты

- Небольшие изменения внешней нагрузки могут сильно отразиться на результатах
- Слишком маленькие значения и ошибка, частые ложные срабатывания
- Наиболее часто происходило на ререндере

Маленькие компоненты

- Небольшие изменения внешней нагрузки могут сильно отразиться на результатах
- Слишком маленькие значения и ошибка, частые ложные срабатывания
- Наиболее часто происходило на ререндере
- Решение — **Абсолютная ошибка**

Большие компоненты

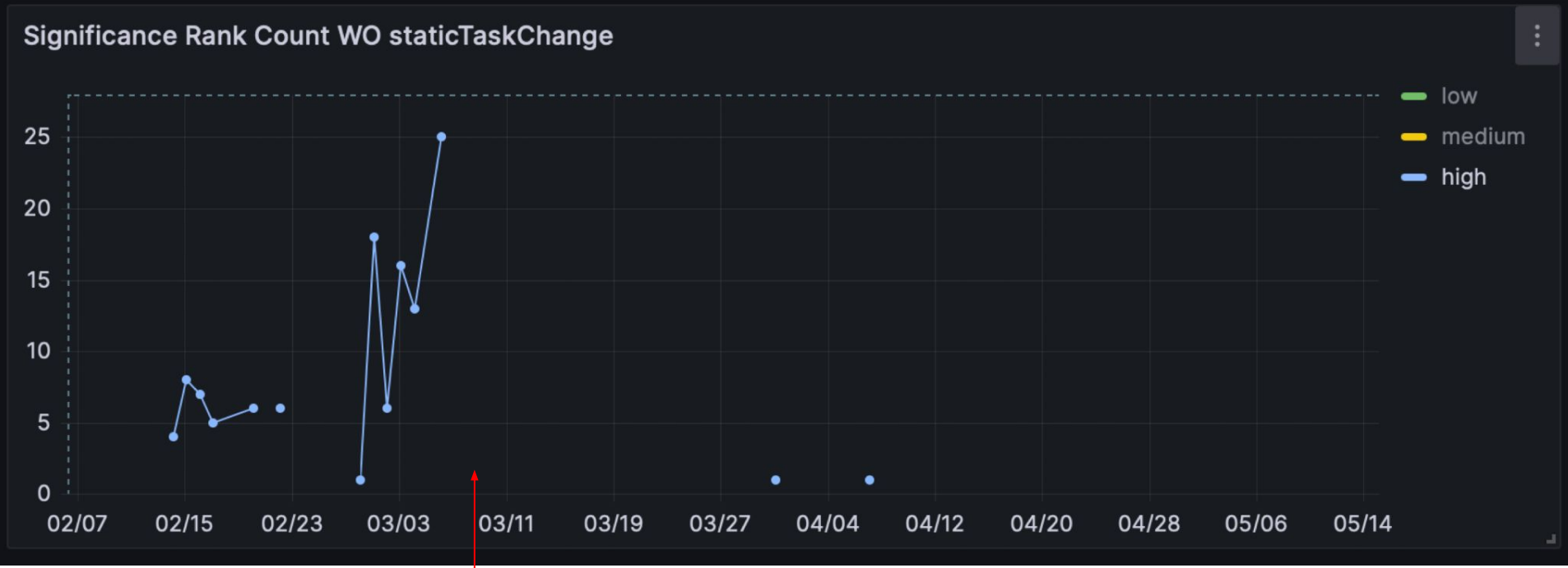
Большие компоненты

- Чем больше компонент, тем больше у него точек нестабильности
- Если тестируемый большой компонент нестабилен, то тест надо разбивать на несколько маленьких

Статичный компонент

Статичный компонент

- Прогоняем те же бенчмарки на компоненте, который не меняется
- Выражаем через метрики статичного компонента метрики тестируемого компонента
- Условные единицы вместо миллисекунд
- Изменение нагрузки частично купируется, ошибка пропорционально увеличивается



Тут выкатили улучшения стабильности

Выделенная машина

- Небольшие виртуалки-раннеры
- Одна джоба одновременно
- Уменьшилась ошибка и значения

Фильтрация выбросов

- На основе медианного абсолютного отклонения и квантильной оценки Харрела-Дэвиса
- Улучшает результаты среднего, поскольку оно не устойчиво к выбросам

Коллаборативный прогон

- Примерно один профиль нагрузки
- Чередование бенчмарков двух веток в пуле потоков
- Уменьшилась ошибка и флуктуации

Ошибки третьего рода

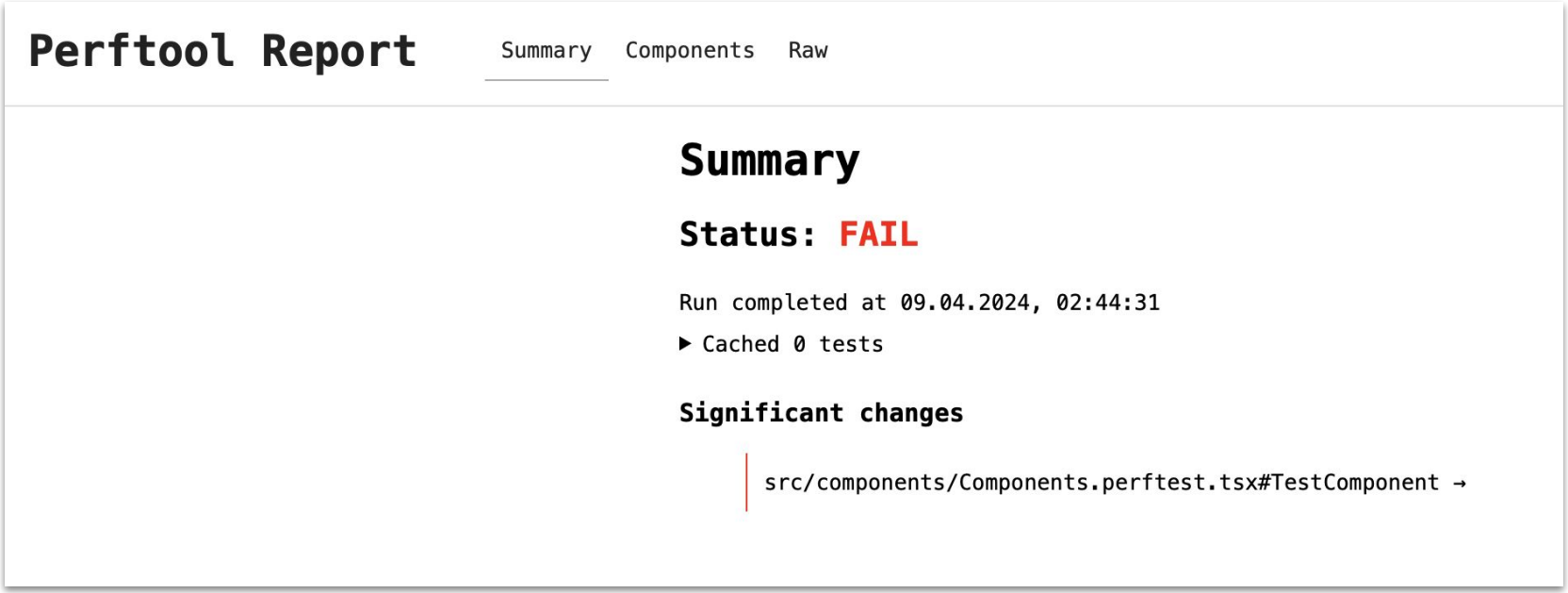
- С помощью анализа мод получаем количество модальностей
- При изменении количества мод нельзя принять или отвергнуть гипотезу

Интеграции

⚡ **Тестирование производительности компонентов**

- Результат: 🟢 OK

Визуальный отчет



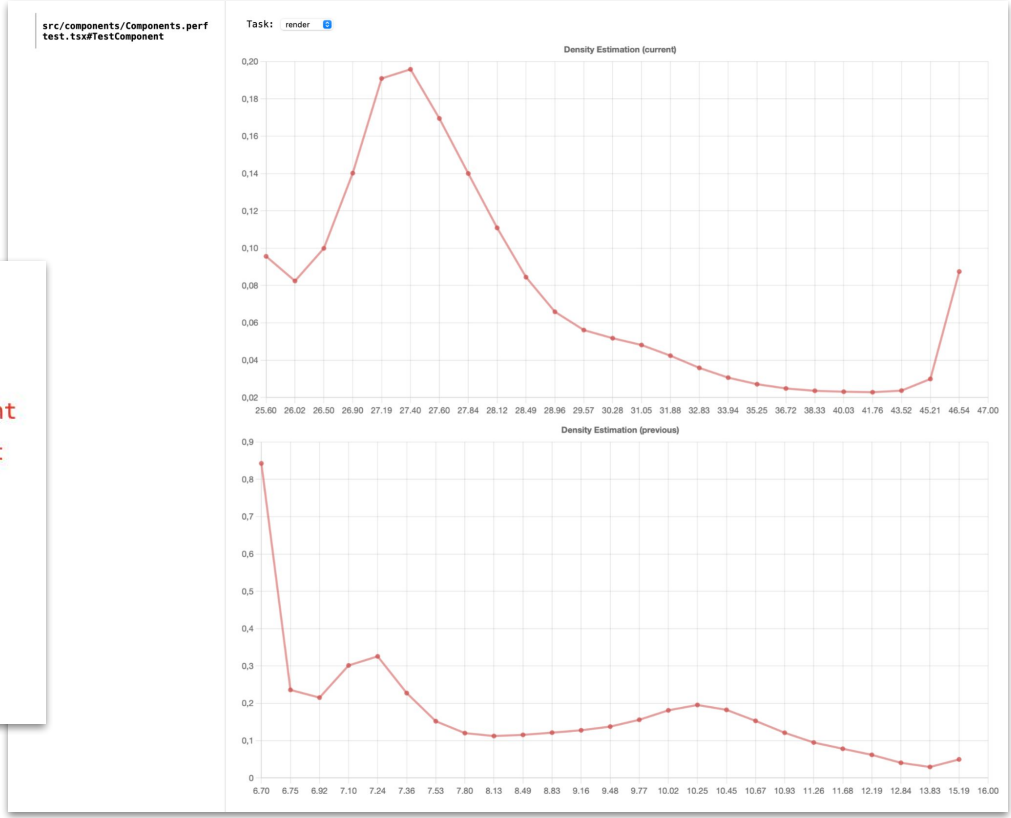
Визуальный отчет

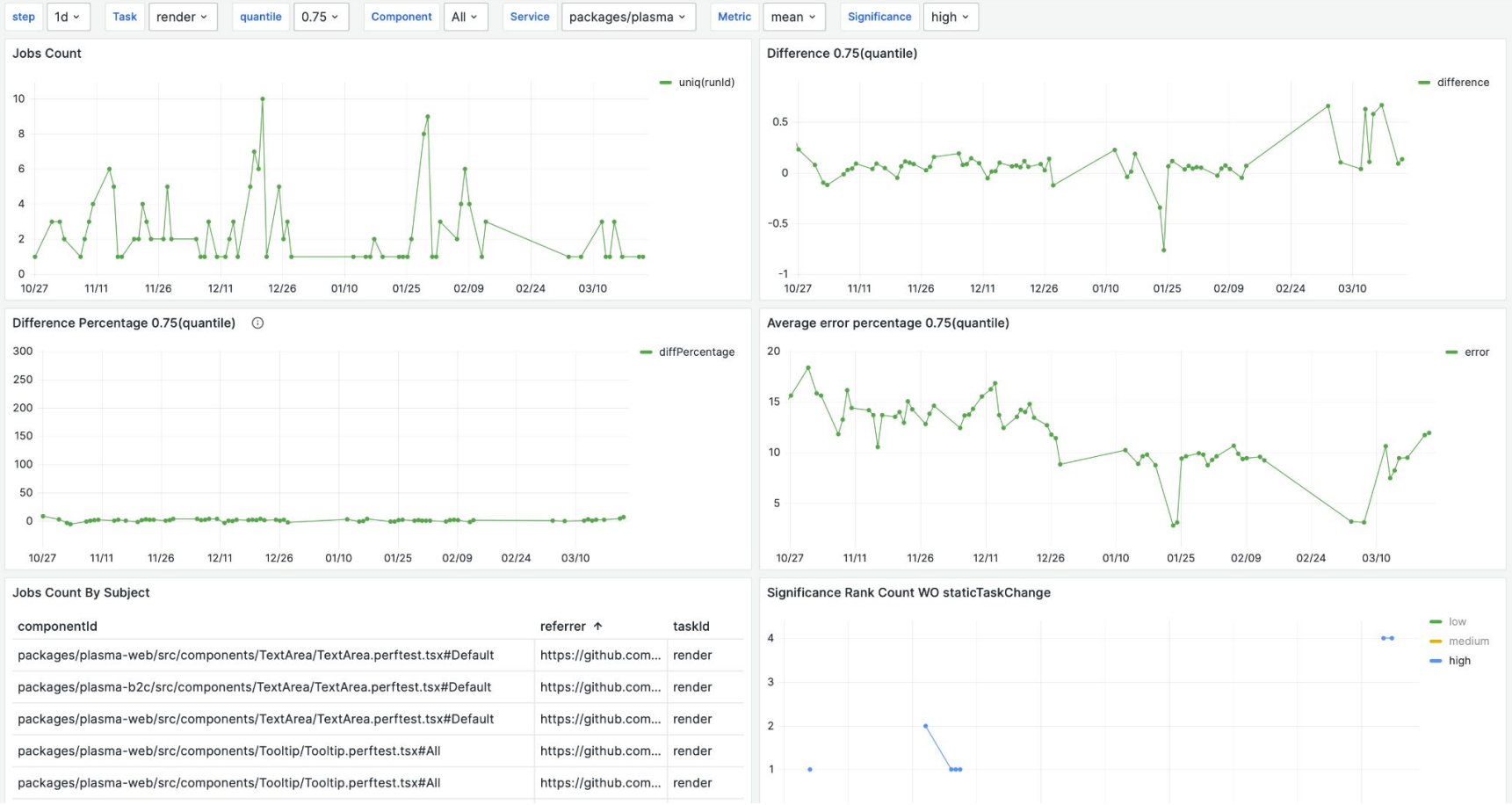
Metrics

midhinge 0.94 ± 0.24 → 3.71 ± 0.69 (+294.25%) significant
trimean 0.96 ± 0.24 → 3.55 ± 0.67 (+268.68%) significant

Mode #1

mean 9.79 ± 1.87 → 33.06 ± 3.40 (+237.79%) significant
hl 9.65 ± 1.87 → 32.50 ± 3.40 (+236.93%) significant
stdev 2.38 → 6.60 (+177.12%)

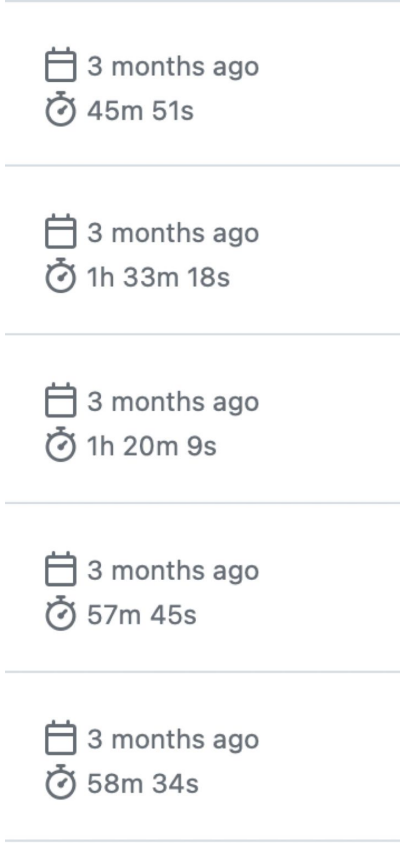



step 1d
Task render
quantile 0.75
Component All
Service packages/plasma
Metric mean
Significance high
Jobs Count
uniq(runId)
Difference 0.75(quantile)
difference
Difference Percentage 0.75(quantile)
diffPercentage
Average error percentage 0.75(quantile)
error
Jobs Count By Subject
componentId
referrer
taskId
packages/plasma-web/src/components/TextArea/TextArea.perftest.tsx#Default https://github.com... render
packages/plasma-b2c/src/components/TextArea/TextArea.perftest.tsx#Default https://github.com... render
packages/plasma-web/src/components/TextArea/TextArea.perftest.tsx#Default https://github.com... render
packages/plasma-web/src/components/Tooltip/Tooltip.perftest.tsx#All https://github.com... render
packages/plasma-web/src/components/Tooltip/Tooltip.perftest.tsx#All https://github.com... render
Significance Rank Count WO staticTaskChange
low
medium
high

Время прогона

Больше тестов
+ прогоны в одной ветке

= время прогона 45-90 минут



Решение

- Условный запуск
- Подбирать временное окно для каждого компонента
- Кэшировать значение временного окна между master-feature прогонами
- Частичный прогон: тестировать только изменившиеся компоненты

Прогон только изменившихся компонентов

- Использовать webpack
- Каждый тестовый компонент в отдельном чанке
- Сравнивать contentHash, чтобы узнать изменился ли компонент

Результат: до 4-9x ускорение прогона

Пример работы

10 ms ResizeObserver

- 21ms → 30ms
- создание и инициализация

  ResizeObserver в useEffect

Component performance testing

Result: 🔴 **Fail**

Check out report in job artifacts!

Top 5 significant render changes:
packages/plasma-ui/src/components/TextArea/TextArea.perftest.tsx#Default: 49.95%

Дальнейшее развитие

- Github Actions
- Server rendering и hydration
- Profiler API и ускорение за счет совмещения джоб
- Абстрагирование от реакта

perftool@github

Спасибо!

Мои контакты

SBER DEVICES