
Яндекс

Яндекс

Системный подход к скорости: онлайн-измерения

Андрей Прокопюк

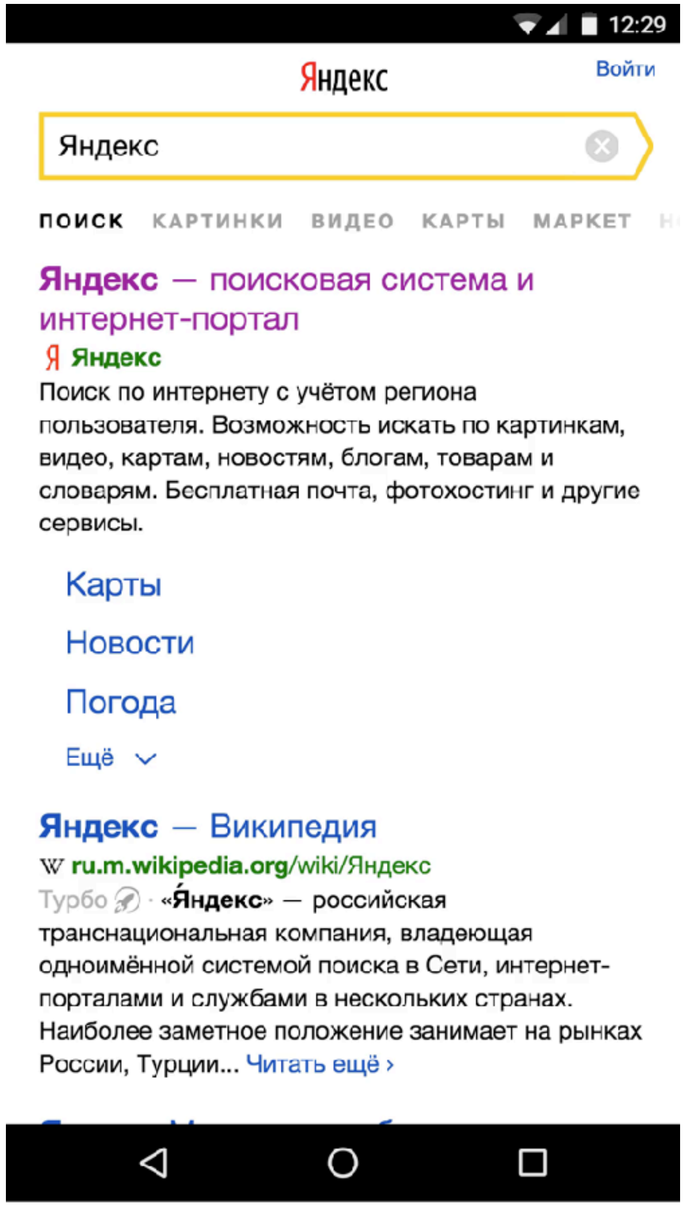
Андрей Прокопюк

- Родился в роддоме
- Учился в школе
- Работаю в Яндексе, в команде скорости

Команда скорости

- Делаем оптимизации фронтенда Поиска
- Делаем инфраструктуру для сбора метрик и оффлайнового тестирования скорости
- Анализируем данные
- Помогаем другим сервисам Яндекса советом и инструментами

Поисковая выдача



Важность скорости

Важность скорости

- Голос разума: быстрый интерфейс нравится пользователям больше, чем медленный
- Голос жадности: а как это влияет на бизнес? Надо ли тратить время?

История шрифта

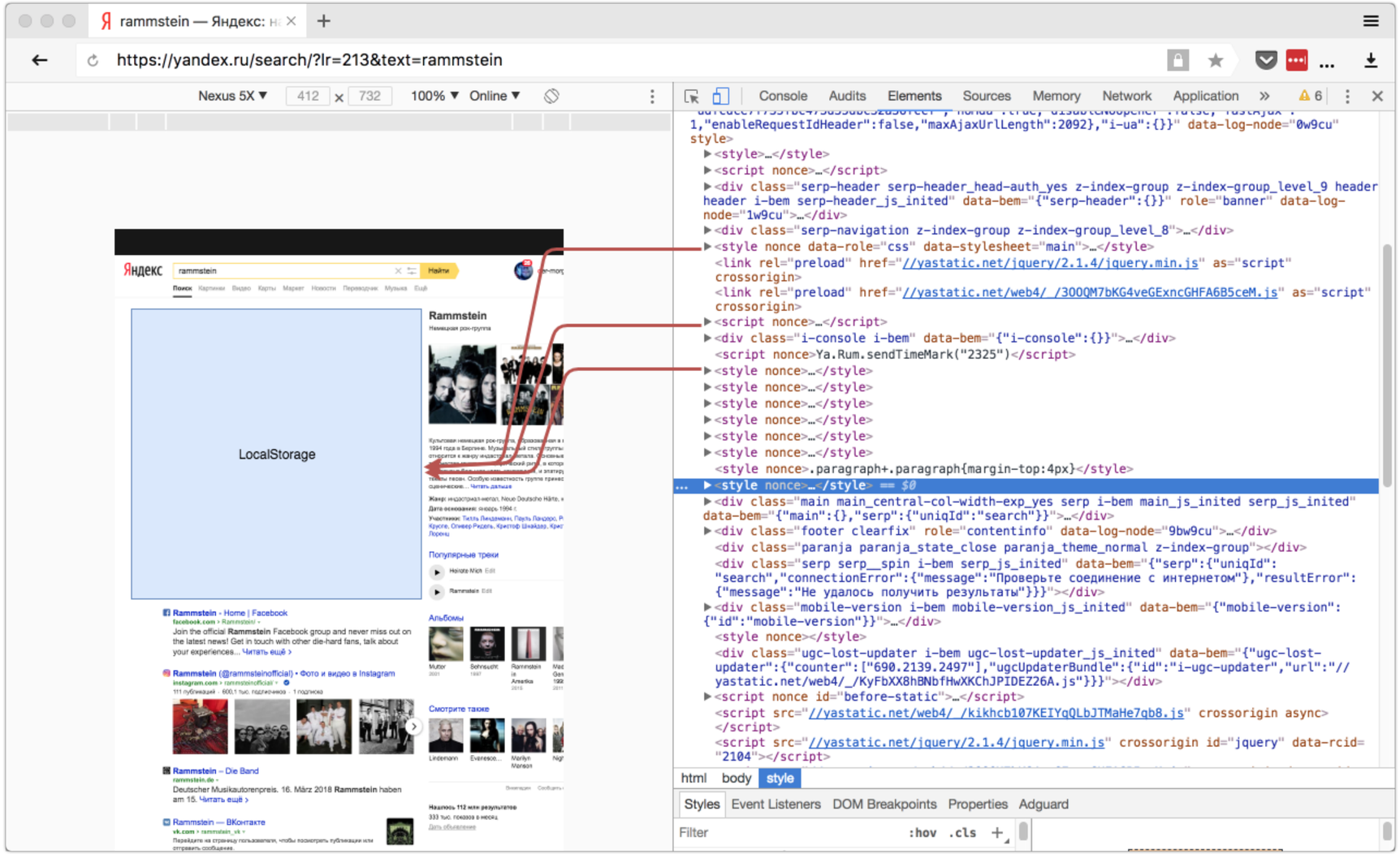
История внедрения специфичного веб-шрифта

- среднее время до отрисовки контента: +3% (62 мс)
- время до первого клика: +1,5%
- кликнутых страниц: -0,4%
- уменьшились времена присутствия
- увеличились времена отсутствия

История шрифта

Беда-беда, огорчение – не катим

История кэширования в LS



История кэширования в LS

Провели эксперимент с отключением

- среднее время доставки HTML: +12%
- время до отрисовки шапки: -3%
- время начала парсинга контента: -1%
- время до инициализации JS: -1%
- время до первого клика: -0,6%

История кэширования в LS

Отключили – и ликованию не было конца

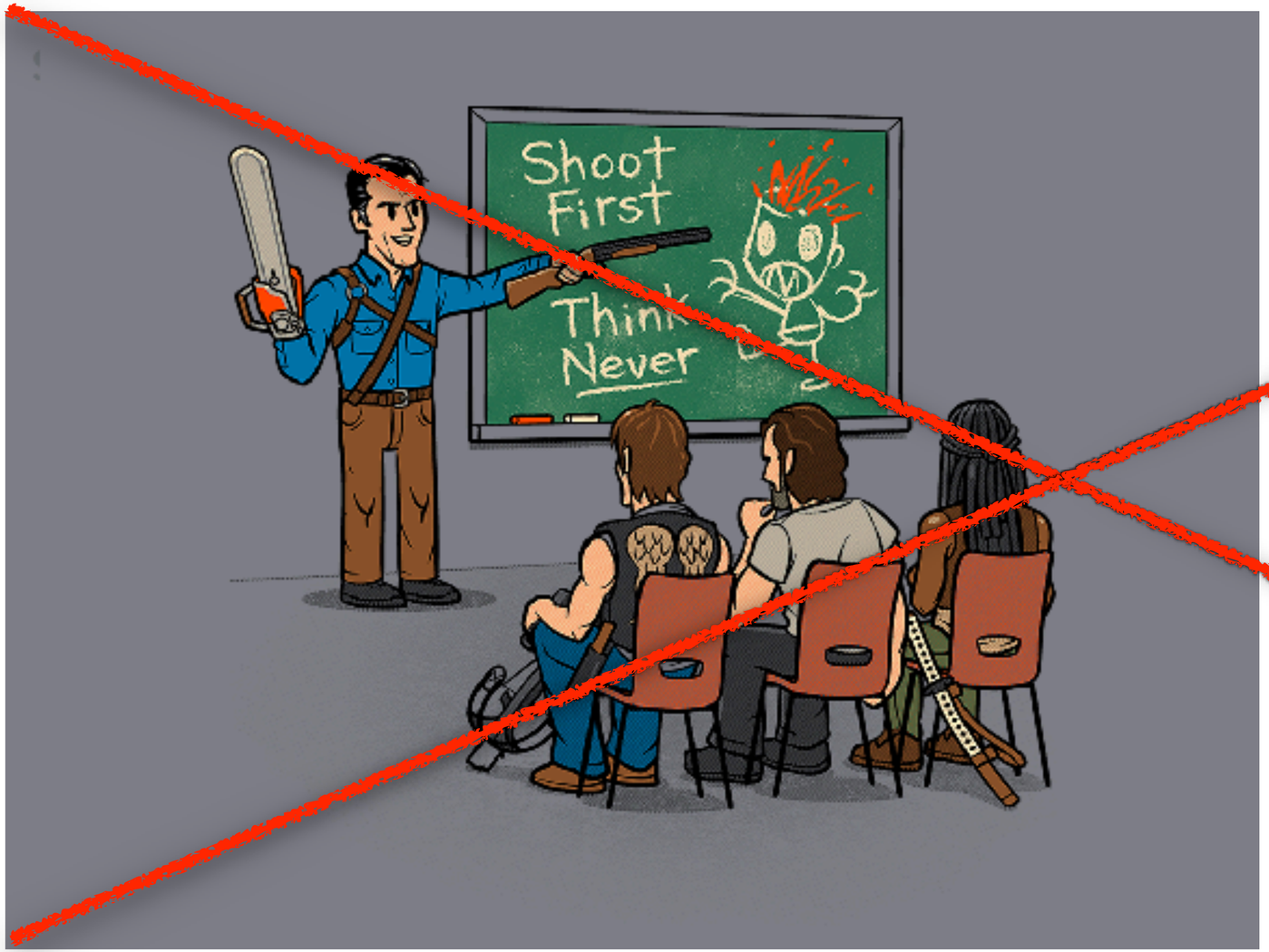
Важность скорости

Скорость влияет на метрики бизнеса

Важность скорости

> Оптимизации должны предваряться измерениями

Измерять перед оптимизацией



Сначала стрелять, потом думать –

или вообще не думать

Важность скорости

Измерения должны отражать пользовательский опыт

Что и как измерить?

Что измерить?

- Время до первой отрисовки контента: TTFCP
- Время до отрисовки значимого контента: TTFMP
- Время до инициализации фреймворка: JS inited
- Время до первой интерактивности: TTI

Как измерить TTFCP?

```
// Время до первой отрисовки контента
// Paint Timing API

for (const ev of performance.getEntriesByType('paint')) {
  if (ev.name == 'first-contentful-paint') {
    send('ttfcp', ev.startTime)
    break
  }
}
```

Как измерить TTFCP?

```
// Время до первой отрисовки контента
// Paint Timing API

for (const ev of performance.getEntriesByType('paint')) {
  if (ev.name == 'first-contentful-paint') {
    send('ttfcp', ev.startTime)
    break
  }
}
```

Как измерить TTFCP?

```
// Время до первой отрисовки контента
// Paint Timing API

for (const ev of performance.getEntriesByType('paint')) {
  if (ev.name == 'first-contentful-paint') {
    send('ttfcp', ev.startTime)
    break
  }
}
```

Как измерить TTFCP?

```
// Время до первой отрисовки контента
// Paint Timing API

for (const ev of performance.getEntriesByType('paint')) {
  if (ev.name == 'first-contentful-paint') {
    send('ttfcp', ev.startTime)
    break
  }
}
```

Как измерить TTFMP?

Как измерить момент отрисовки значимого контента?

- Нет готового API
- Можно использовать requestAnimationFrame
- Можно использовать IntersectionObserver

Как измерить TTFMP?

```
<!-- Время отрисовки значимого контента, rAF -->

<script>
requestAnimationFrame(() => {
  send('ttfmp_low', performance.now())
  requestAnimationFrame(() => send('ttfmp_high', performance.now()))
})
</script>

<div class="main-content">Foo</div>
```

Как измерить TTFMP?

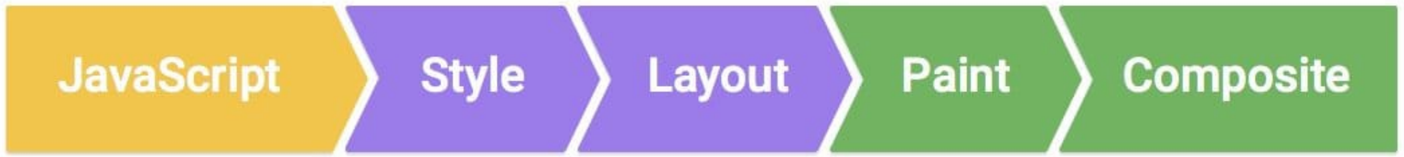
```
<!-- Время отрисовки значимого контента, rAF -->

<script>
requestAnimationFrame(() => {
  send('ttfmp_low', performance.now())
  requestAnimationFrame(() => send('ttfmp_high', performance.now()))
})
</script>

<div class="main-content">Foo</div>
```

Как измерить TTFMP?

```
<!-- Время отрисовки значимого контента, rAF -->

<script>
requestAnimationFrame(() => {
  send('ttfmp_low', performance.now())
  requestAnimationFrame(() => send('ttfmp_high', performance.now()))
})
</script>

<div class="main-content">Foo</div>
```

Почему это работает?

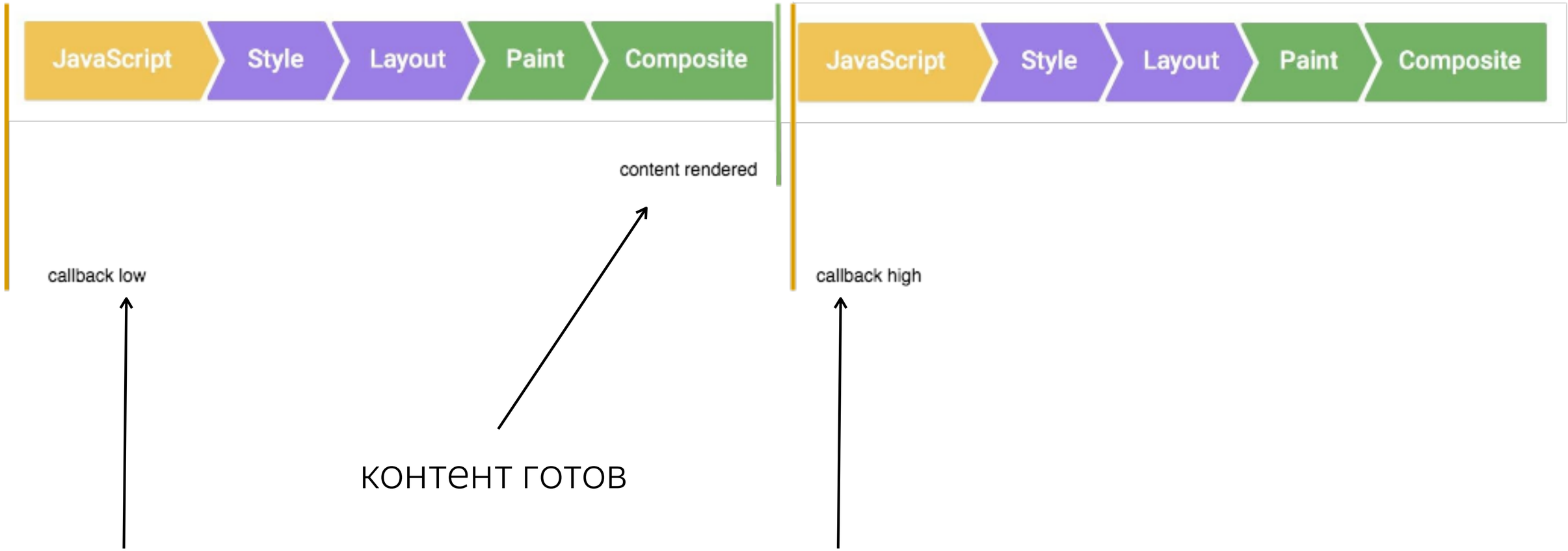
Почему это работает?



первый requestAnimationFrame

контент отрисован

Почему это работает?



контент готов

нижняя граница отрисовки

верхняя граница отрисовки

Как измерить TTFMP?

```
<!-- Время отрисовки главного контента, IntersectionObserver -->

<div id="main-content">Foo</div>
<script>
const domNode = document.querySelector('#main-content')
const io = new IntersectionObserver((entries, observer) => {
  send('ttfmp', performance.now())
  observer.unobserve(domNode)
})
io.observe(domNode)
</script>
```

Как измерить TTFMP?

```
<!-- Время отрисовки главного контента, IntersectionObserver -->

<div id="main-content">Foo</div>
<script>
const domNode = document.querySelector('#main-content')
const io = new IntersectionObserver((entries, observer) => {
  send('ttfmp', performance.now())
  observer.unobserve(domNode)
})
io.observe(domNode)
</script>
```

Как измерить TTFMP?

```
<!-- Время отрисовки главного контента, IntersectionObserver -->

<div id="main-content">Foo</div>
<script>
const domNode = document.querySelector('#main-content')
const io = new IntersectionObserver((entries, observer) => {
  send('ttfmp', performance.now())
  observer.unobserve(domNode)
})
io.observe(domNode)
</script>
```

Почему это работает?

viewport

main-content

Как измерить JS Inited?

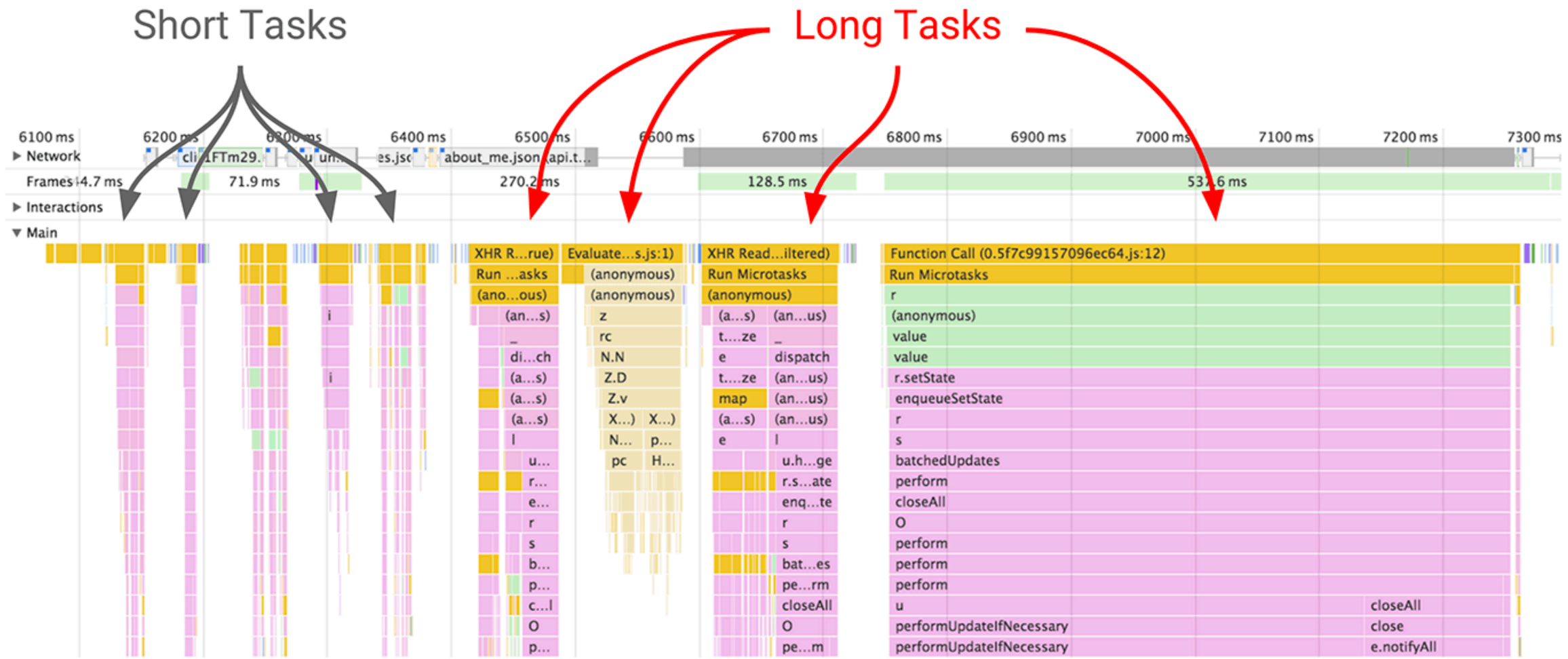
```
// Время до инициализации клиентского фреймворка

MyFramework.addEventListener('init', () => {
  send('js-framework-inited', performance.now())
})

// Индивидуально для каждого фреймворка
```

Как измерить TTI?

TTI наступает, когда главный поток браузера освобождается

Поможет концепция долгих задач и Long Task API

Длинные задачи



Как измерить TTI?

Как измерить TTI?

```
<!-- Время до первой интерактивности, шаг 1 -->

<body>
<script>
  let longTaskEvents = []
  const obs = new PerformanceObserver(tasks => {
    longTaskEvents = longTaskEvents.concat(tasks.getEntries())
  })
  obs.observe({ events: ['longtask'] })
</script>
```

Как измерить TTI?

```
<!-- Время до первой интерактивности, шаг 1 -->

<body>
<script>
  let longTaskEvents = []
  const obs = new PerformanceObserver(tasks => {
    longTaskEvents = longTaskEvents.concat(tasks.getEntries())
  })
  obs.observe({ events: ['longtask'] })
</script>
```

Как измерить TTI?

```
<!-- Время до первой интерактивности, шаг 1 -->

<body>
<script>
  let longTaskEvents = []
  const obs = new PerformanceObserver(tasks => {
    longTaskEvents = longTaskEvents.concat(tasks.getEntries())
  })
  obs.observe({ events: ['longtask'] })
</script>
```

Как измерить TTI?

```
<!-- Время до первой интерактивности, шаг 2 -->
<script>
  (function checkTTI() {
    const last = longTaskEvents[longTaskEvents.length - 1]
    const lastLtEnd = last.startTime + last.duration

    performance.now() - lastLtEnd >= 3000 ?
      send('tti', lastLtEnd) :
      setTimeout(checkTTI, 3000)
  })()
</script></body>
```

Как измерить TTI?

```
<!-- Время до первой интерактивности, шаг 2 -->
<script>
  (function checkTTI() {
    const last = longTaskEvents[longTaskEvents.length - 1]
    const lastLtEnd = last.startTime + last.duration

    performance.now() - lastLtEnd >= 3000 ?
      send('tti', lastLtEnd) :
      setTimeout(checkTTI, 3000)
  })()
</script></body>
```

Как измерить TTI?

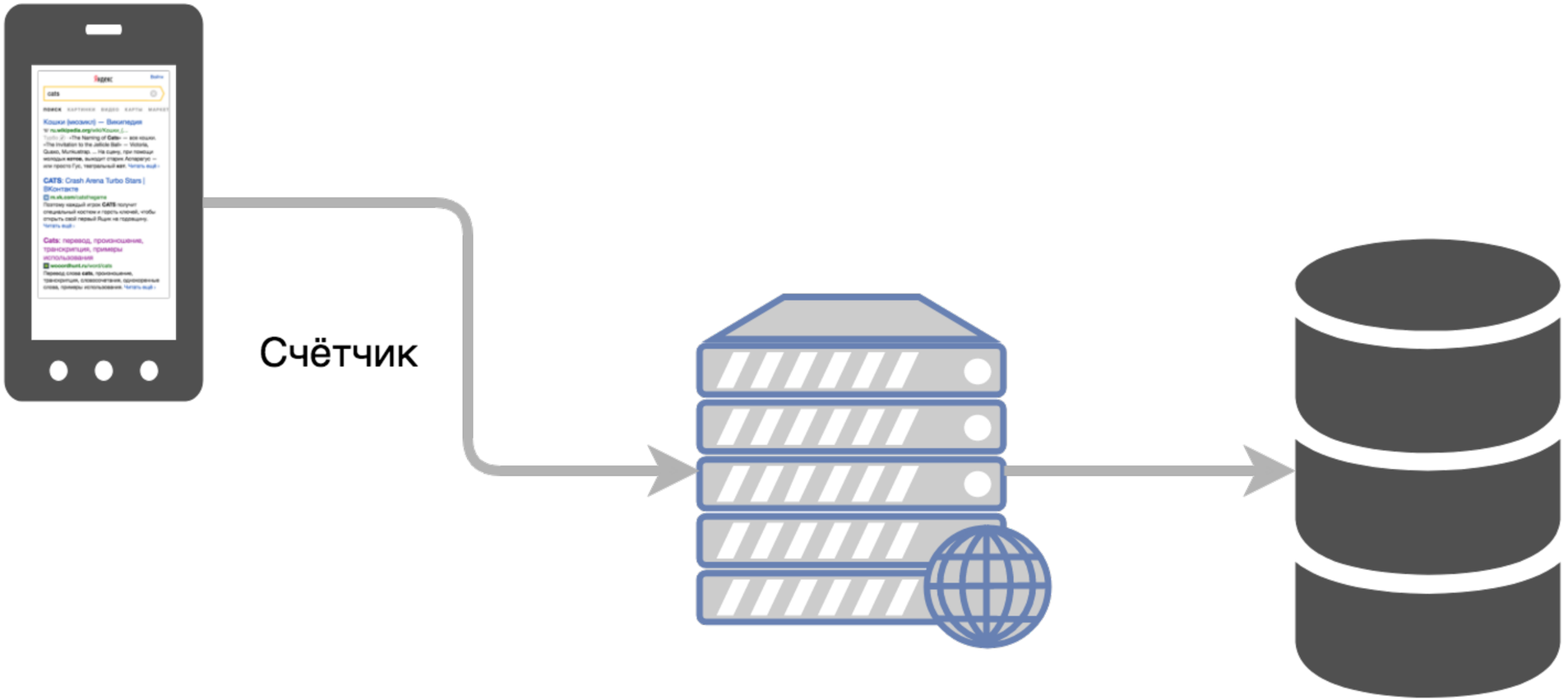
```
<!-- Время до первой интерактивности, шаг 2 -->
<script>
  (function checkTTI() {
    const last = longTaskEvents[longTaskEvents.length - 1]
    const lastLtEnd = last.startTime + last.duration

    performance.now() - lastLtEnd >= 3000 ?
      send('tti', lastLtEnd) :
      setTimeout(checkTTI, 3000)
  })()
</script></body>
```

Как измерить TTI?

```
<!-- Время до первой интерактивности, шаг 2 -->
<script>
  (function checkTTI() {
    const last = longTaskEvents[longTaskEvents.length - 1]
    const lastLtEnd = last.startTime + last.duration

    performance.now() - lastLtEnd >= 3000 ?
      send('tti', lastLtEnd) :
      setTimeout(checkTTI, 3000)
  })()
</script></body>
```

Как обработать данные?

Отправка данных



Расчёт статистики

```
SELECT
    AVG(ttfcp) AS ttfcp_avg,
    PERCENTILE(ttfcp, 50) AS ttfcp_p50,
    PERCENTILE(ttfcp, 75) AS ttfcp_p75,
    PERCENTILE(ttfcp, 95) AS ttfcp_p95,
    PERCENTILE(ttfcp, 99) AS ttfcp_p99
;
```

Расчёт статистики

```
SELECT
    AVG(ttfcp) AS ttfcp_avg, -- Среднее арифметическое
    PERCENTILE(ttfcp, 50) AS ttfcp_p50,
    PERCENTILE(ttfcp, 75) AS ttfcp_p75,
    PERCENTILE(ttfcp, 95) AS ttfcp_p95,
    PERCENTILE(ttfcp, 99) AS ttfcp_p99
;
```

Расчёт статистики

```
SELECT
    AVG(ttfcp) AS ttfcp_avg,
    PERCENTILE(ttfcp, 50) AS ttfcp_p50, -- Процентиль
    PERCENTILE(ttfcp, 75) AS ttfcp_p75,
    PERCENTILE(ttfcp, 95) AS ttfcp_p95,
    PERCENTILE(ttfcp, 99) AS ttfcp_p99
;
```

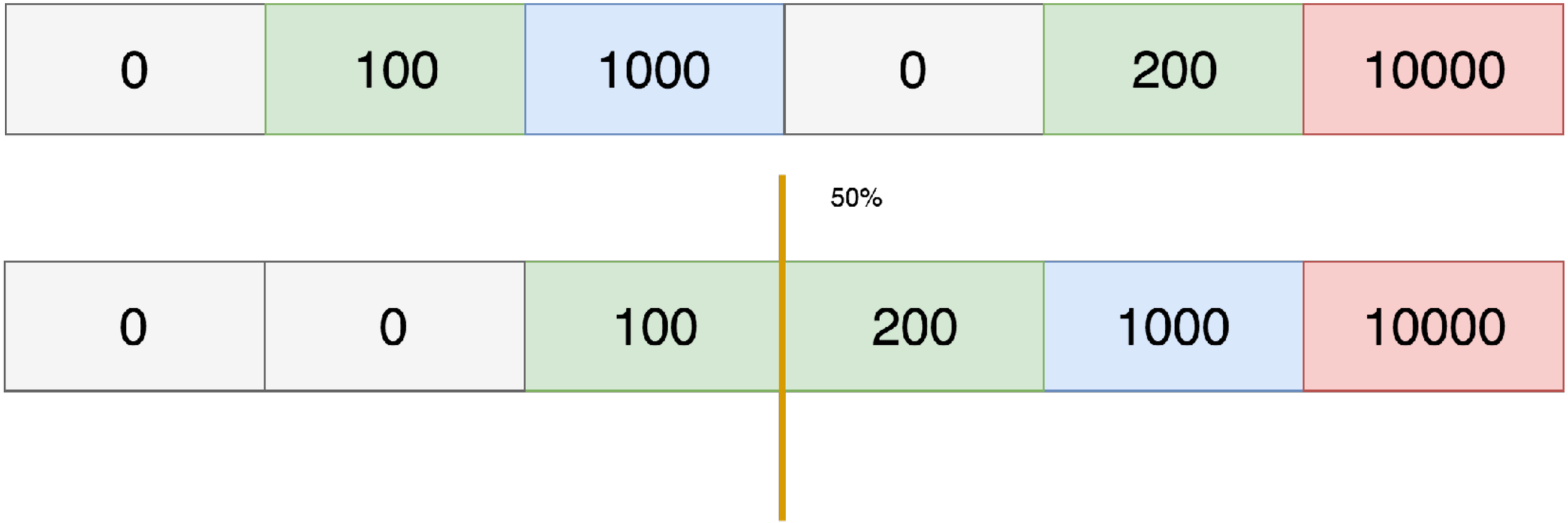
Среднее арифметическое

0	100	1000	0	200	10000

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

AVG = 1883,33

Процентиль



P50 = 150

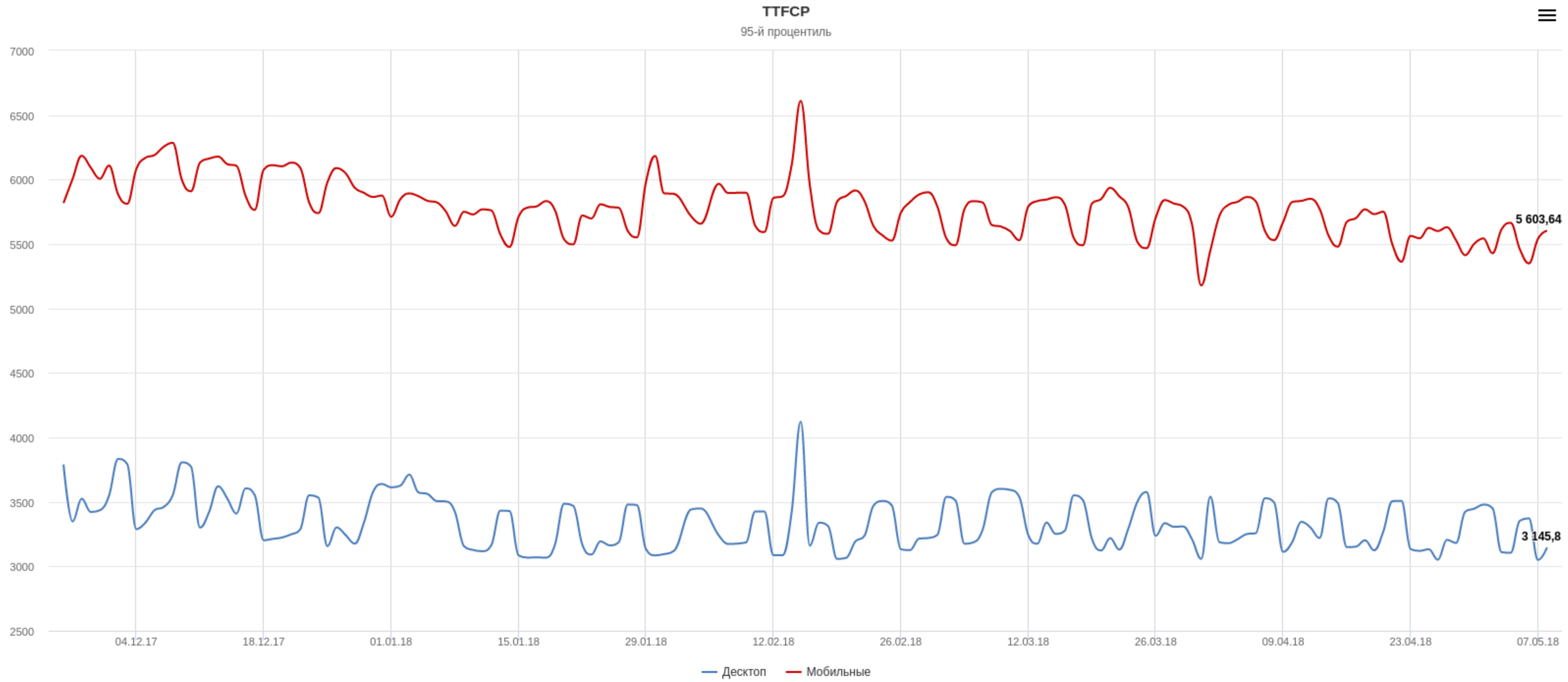
AVG = 1883,33

Агрегации

- Среднее арифметическое – чувствительна к выбросам
- 50-й процентиль (медиана) – 50% запросов укладываются в это время
- 75-й процентиль
- 95-й процентиль
- 99-й процентиль

Здесь живут драконы

Представление

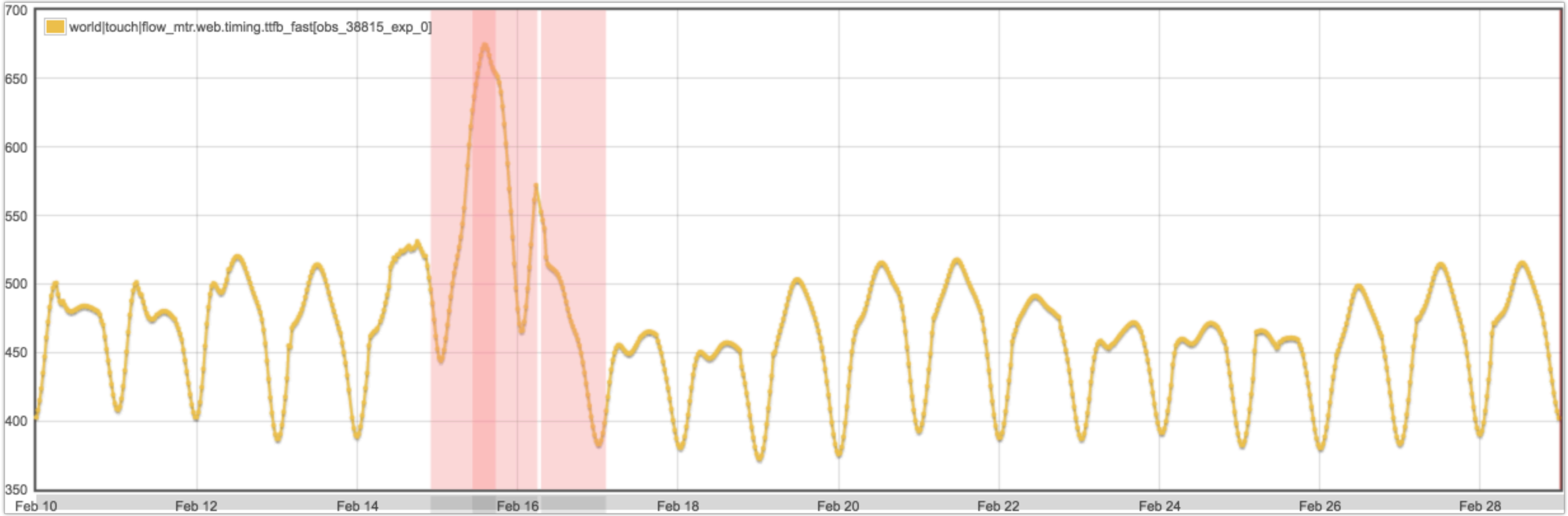


Мониторинг

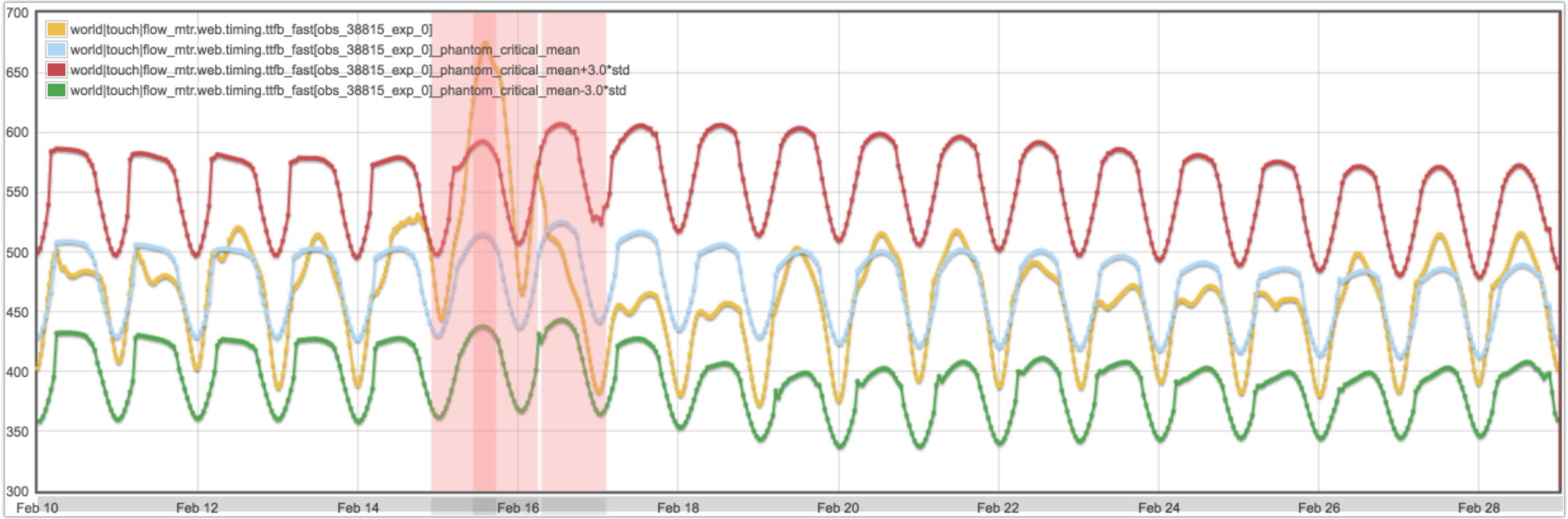
Мониторинг

Разладка – момент, когда случайный процесс меняет свои характеристики

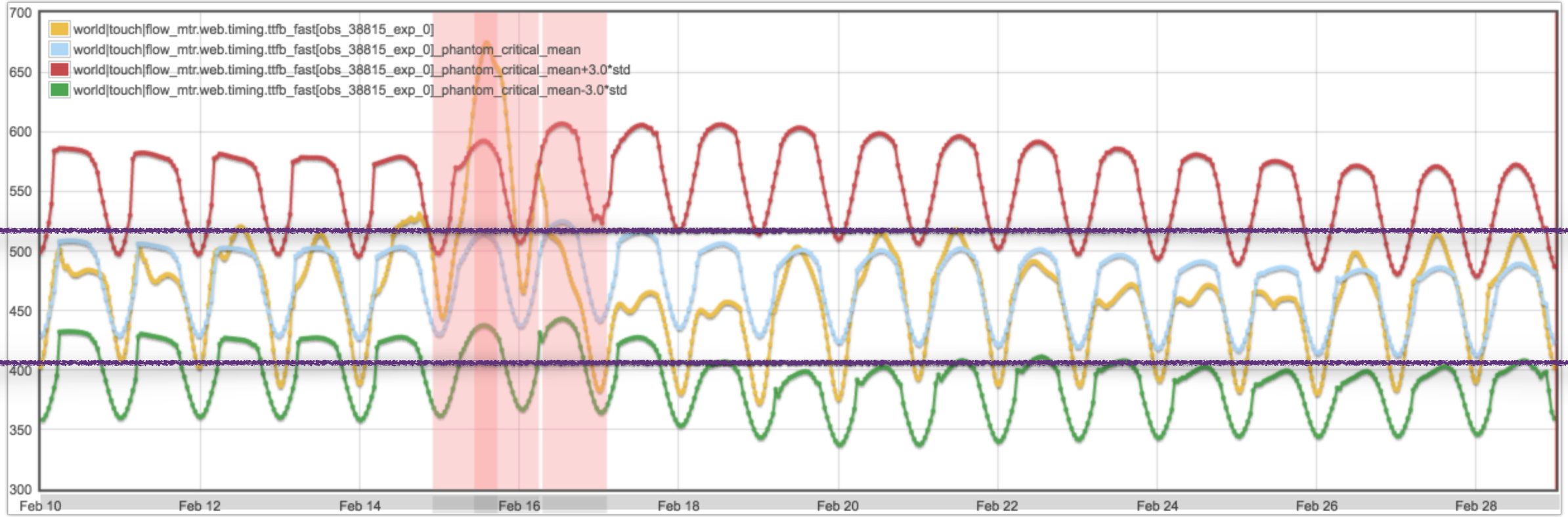
Разладка


world|touch|flow_mtr.web.timing.ttfb_fast[obs_38815_exp_0]
700
650
600
550
500
450
400
350
Feb 10
Feb 12
Feb 14
Feb 16
Feb 18
Feb 20
Feb 22
Feb 24
Feb 26
Feb 28

Разладка


world|touch|flow_mtr.web.timing.ttfb_fast[obs_38815_exp_0]
world|touch|flow_mtr.web.timing.ttfb_fast[obs_38815_exp_0]_phantom_critical_mean
world|touch|flow_mtr.web.timing.ttfb_fast[obs_38815_exp_0]_phantom_critical_mean+3.0*std
world|touch|flow_mtr.web.timing.ttfb_fast[obs_38815_exp_0]_phantom_critical_mean-3.0*std
700
650
600
550
500
450
400
350
300
Feb 10
Feb 12
Feb 14
Feb 16
Feb 18
Feb 20
Feb 22
Feb 24
Feb 26
Feb 28

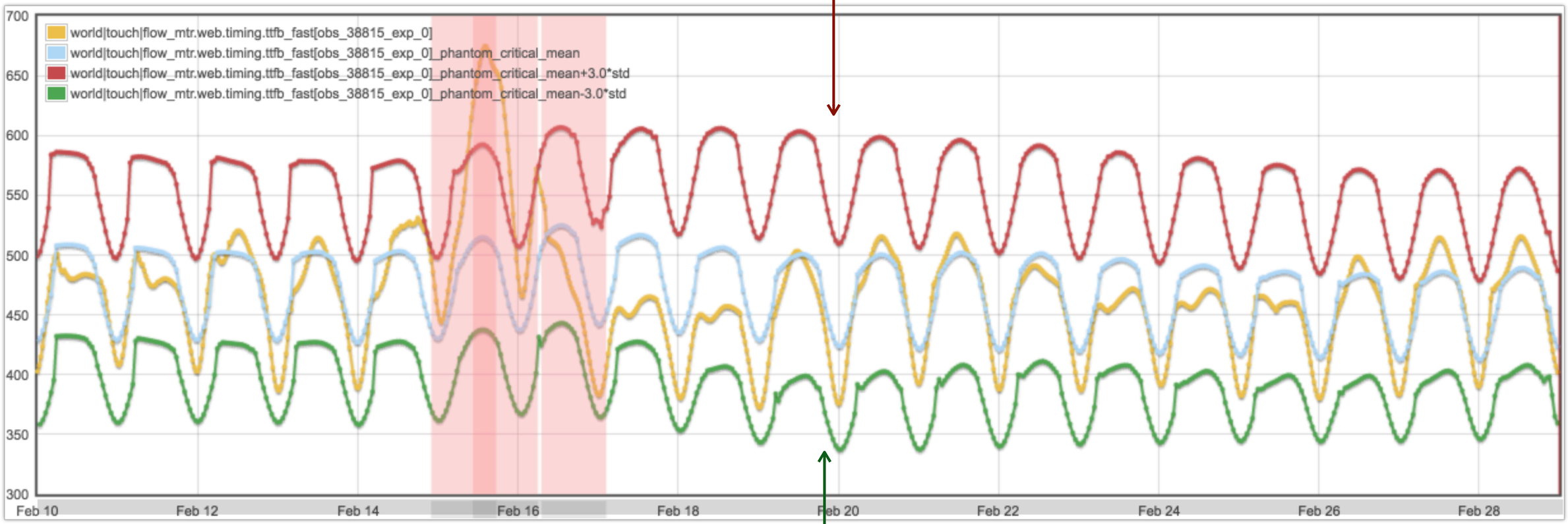
Разладка



Метрика и её скользящая средняя

Разладка

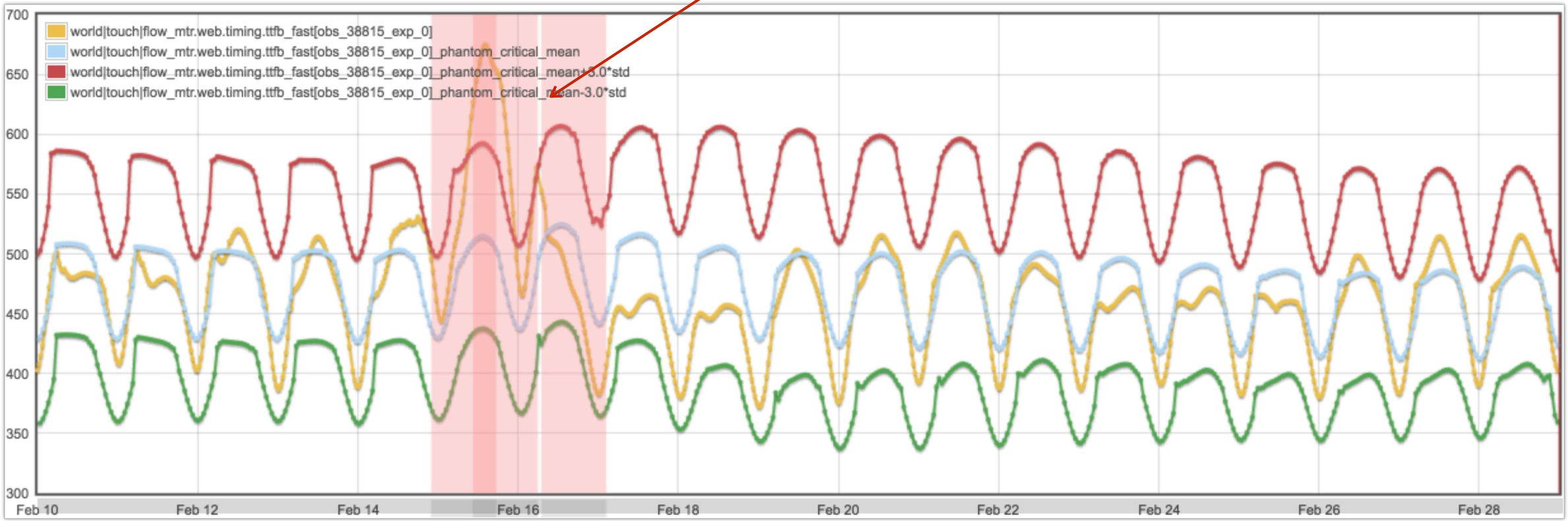
средняя + 3σ



средняя - 3σ

Разладка

выход из безопасного коридора

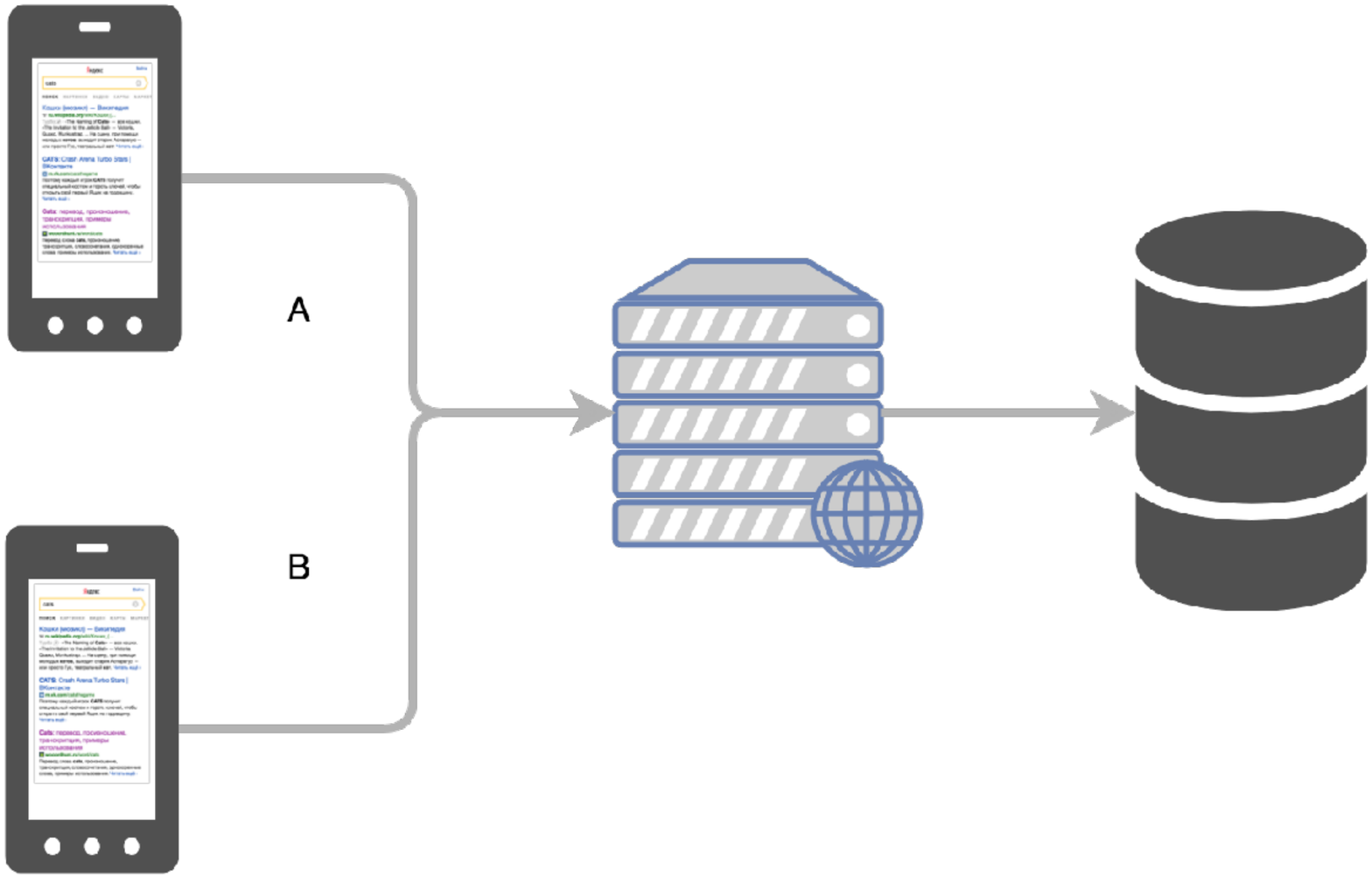


Проверка фичей на скорость

Проверка фичей на скорость

А/Б тестирование – сравнение метрик по контрольной и экспериментальной группе

Сбор данных



Результаты

Метрика	A	B	Δ	MW
TTFCP	1000	990	-10	99.9 %
TTFMP	2200	2150	-50	80 %
JS inited	5300	5350	50	100 %
TTI	6385	7000	615	99.9 %

Результаты

Метрика	A	B	Δ	MW
TTFCP	1000	990	-10	99.9 %
TTFMP	2200	2150	-50	80 %
JS inited	5300	5350	50	100 %
TTI	6385	7000	615	99.9 %

Результаты

Метрика	A	B	Δ	MW
TTFCP	1000	990	-10	99.9 %
TTFMP	2200	2150	-50	80 %
JS inited	5300	5350	50	100 %
TTI	6385	7000	615	99.9 %

Результаты

Метрика	A	B	Δ	MW
TTFCP	1000	990	-10	99.9 %
TTFMP	2200	2150	-50	80 %
JS inited	5300	5350	50	100 %
TTI	6385	7000	615	99.9 %

Результаты

Метрика	A	B	Δ	MW
TTFCP	1000	990	-10	99.9 %
TTFMP	2200	2150	-50	80 %
JS inited	5300	5350	50	100 %
TTI	6385	7000	615	99.9 %

Вывод после эксперимента

Беда-беда, огорчение – не катим

Как сделать у себя?

Вариант №1 – всё своё

- Ручка для приёма данных с клиента
- MongoDB, PostgreSQL, MySQL…
- Агрегации из коробки
- Множество OS решений для представления

Вариант №2 – системы аналитики

- Яндекс.Метрика: metrika.yandex.ru
- Google Analytics: google.ru/analytics

Яндекс.Метрика

Адрес страницы	Просмотры ▼	Ответ сервера	Время до отрисовки	Время до загрузки DOM
Найти				
Итого и средние	23	00.120	00.772	01.412
https://andre.life/	23	00.120	00.772	01.412

Онлайн-измерения

Вспомним былое

Получение значений метрик на клиенте

- TTFCP – первая отрисовка
- TTFMP – важная отрисовка
- JS inited – все кнопки работают
- TTI – всё плавно и приятно

Вспомним былое

Агрегация данных

- Средняя – мониторинги, А/Б тестирование
- Процентили – ближе к пользователю

Вспомним былое

- Графики по дневным данным
- Мониторинг на быстрых данных
- А/Б тестирование

RUM – Real User Monitoring

Хорошее про онлайн-измерения

- Числа из реального мира
- Фидбек со всей аудитории

Недостаток онлайн-измерений

Только после релиза

Что делать?

Дополнить онлайн- оффлайн-измерениями

Спасибо! Вопросы?

Андрей Прокопюк

Техлид команды скорости

 https://andre.life

 @andre487

 @Andre_487

 andre487