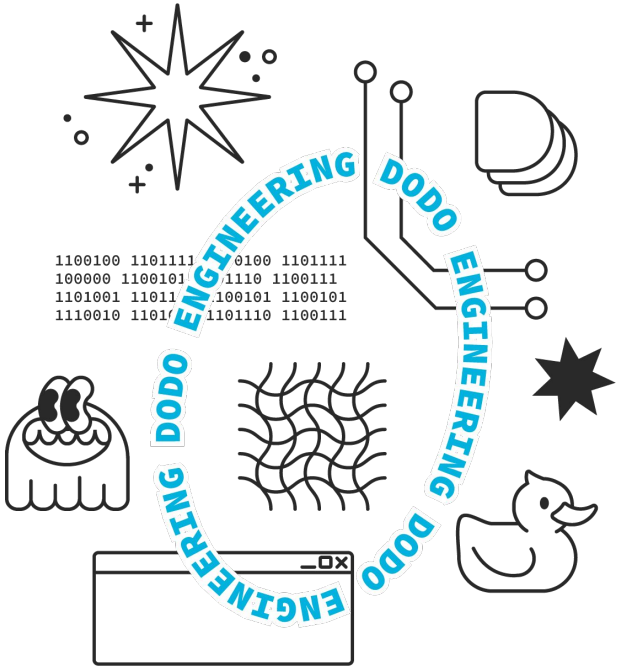
DODO ENGINEERING

Измеряй это! Оживление нагрузочных тестов на Java



Дмитрий Тучс

Heisenbug 2023 Spring online

Head of QA @ Dodo Engineering

dtuchs@gmail.com

dtuchs

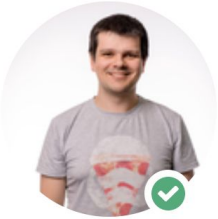
Преподаю в QA.GURU

Опыт в разработке, аналитике, управлении проектами более 14 лет

Для кого?

- Для всех, кто пишет на Java в QA
- Для тех, кто пишет на Java нагрузку - вдвойне
- Для любого уровня (ошибки совершают все)

Проблема #0



Alexander Andronov
CEO at Dodo Engineering

Dodo IS

5800 запросов в секунду

350 заказов в минуту

679 заказов в минуту
– максимальная нагрузка

900+ пиццерий

16 стран

Нагрузочные тесты Dodo IS

5800 запросов в секунду

350 заказов в минуту

679 заказов в минуту
– максимальная нагрузка

x3

X3 (“икс три”)

GET: /menu

POST: /order

~500 мс

+

~500 мс

= 2 rps / thread

Модель тестового фреймворка

```
interface Sampler { // Шаблон
    void run();
}

class GetMenu implements Sampler {
    @Override
    void run() {

        GET: /menu        // 500 ms

    }
}
```

Модель тестового фреймворка

```
class ThreadGroup {

  void start() {
    Threads count + Run
  }
}
```

```
// RPS: 6
GET: /menu
GET: /menu
GET: /menu
```

Модель тестового фреймворка

```
class ThreadGroup {
  private final int rps;
  private final Sampler sampler;
  // constructor

  void start() {
    int threadCount = rps / 2; // Помним, что Sampler ~ 500 мс
    for (int i = 0; i < threadCount; i++) {
      new Thread(() -> {while(true) {sampler.run();}}).start();
    }
  }
}
```

// Бесконечно выполняем "запросы"

Модель тестового фреймворка

```
public static void main(String[] args) {
  new ThreadGroup(100)
                GET: /menu
  new ThreadGroup(100)
                POST: /order
}
```

= 200 rps

Модель тестового фреймворка

```
public static void main(String[] args) {
  new ThreadGroup(100, new GetMenu())
      .start();
  new ThreadGroup(100, new PostOrder())
      .start();
}
```

Вы поняли принцип работы JMeter

APACHE JMeter™

Вы поняли принцип работы JMeter

Sampler из модели:

Package org.apache.jmeter.protocol.java.sampler

Interface JavaSamplerClient

```
SampleResult          runTest(JavaSamplerContext context)
```

GET: /menu

Вы поняли принцип работы JMeter

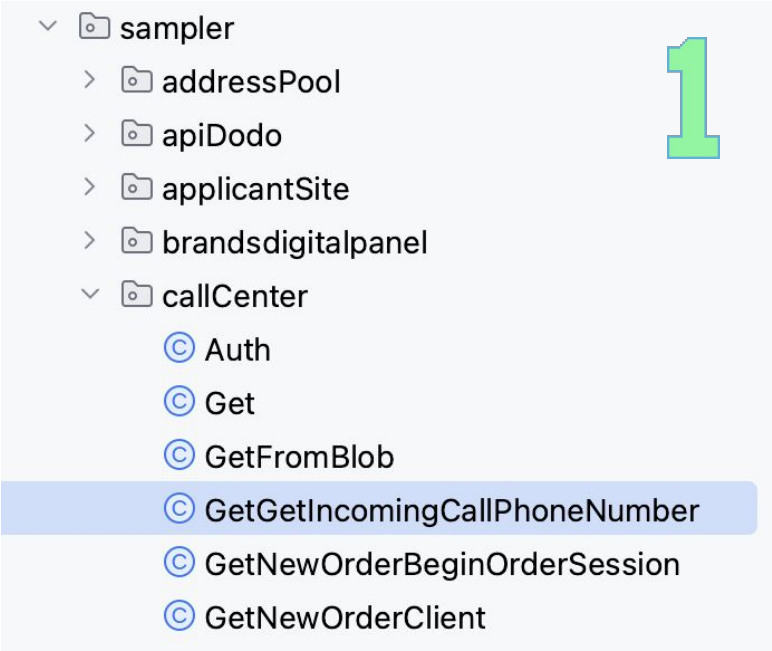
- ThreadGroup из модели:

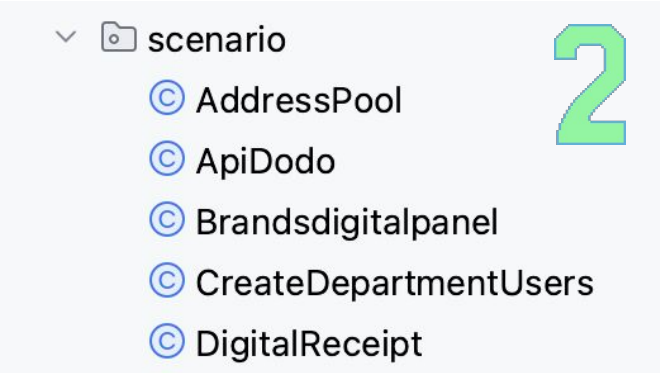
Package org.apache.jmeter.threads

Class ThreadGroup

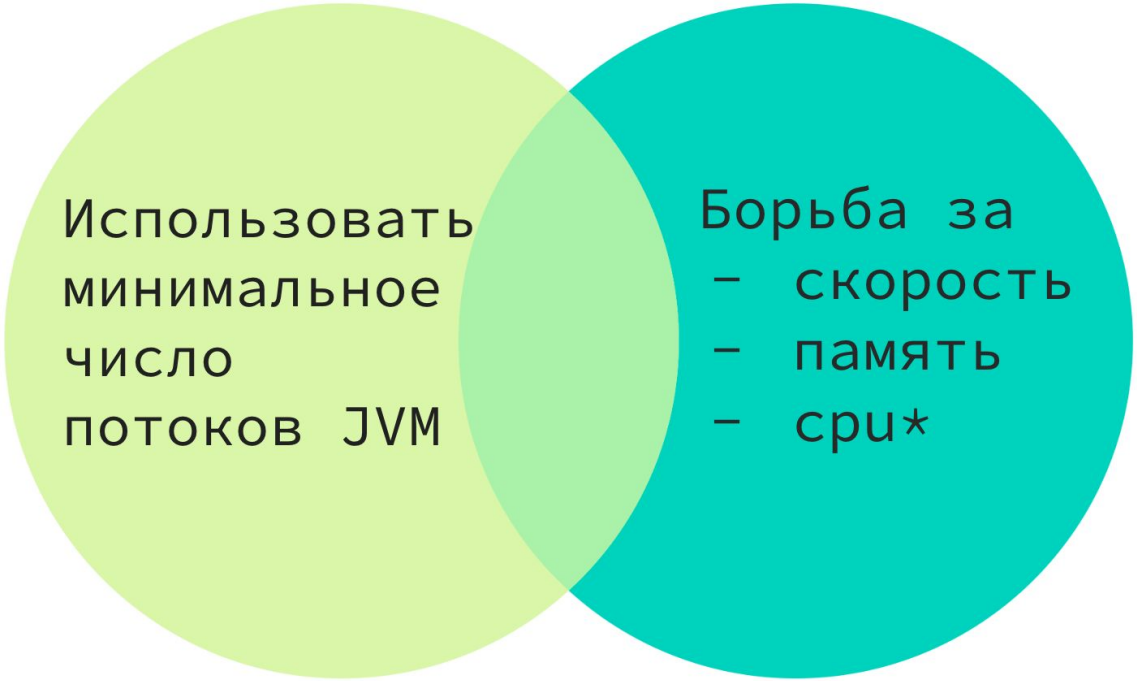
```
void                    setNumThreads(int numThreads)
```

Наши тесты поверх JMeter

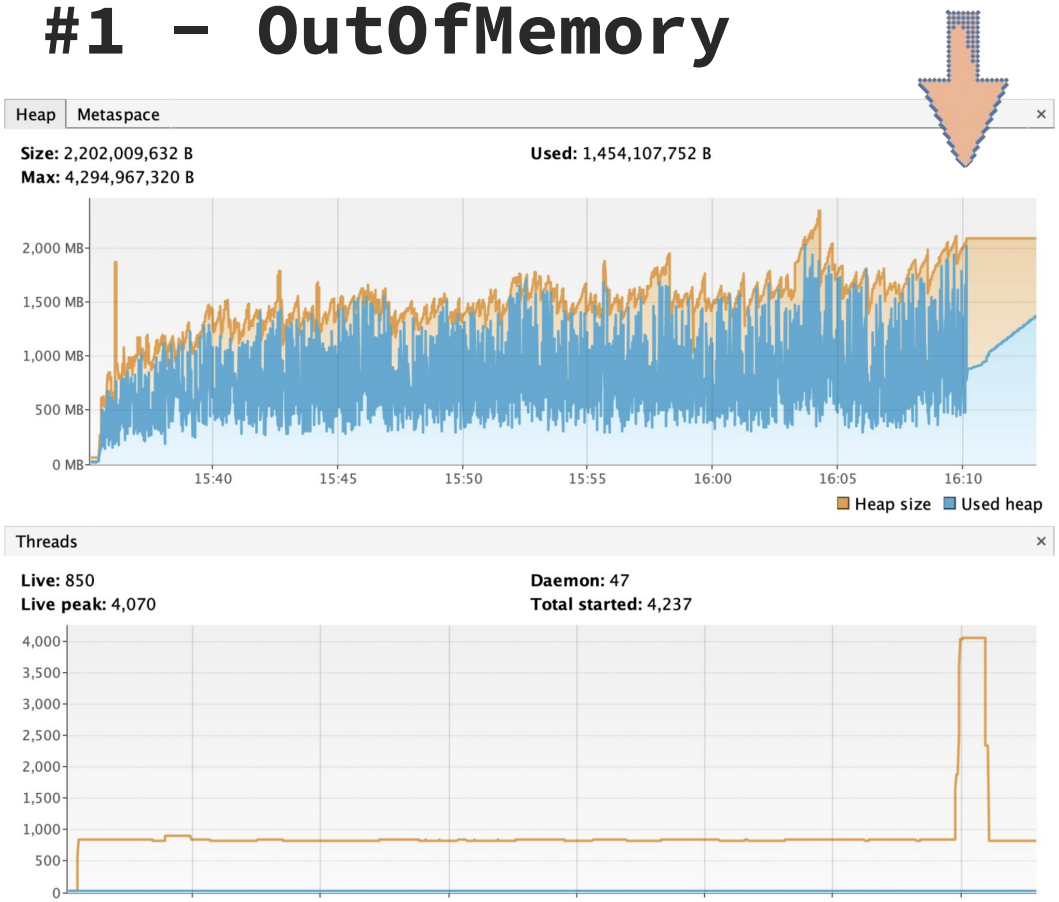




Вызовы перед performance QA



Проблема #1 – OutOfMemory



Что еще, кроме OutOfMemory?

Неправильно конфигурируем http

Неправильно считаем число потоков

Шлем в кафку то, что не надо

Неправильно парсим json и html

Не используем асинхронность

Что еще, кроме OutOfMemory?

Неправильно конфигурируем http

Неправильно считаем число потоков

Шлем в кафку то, что не надо

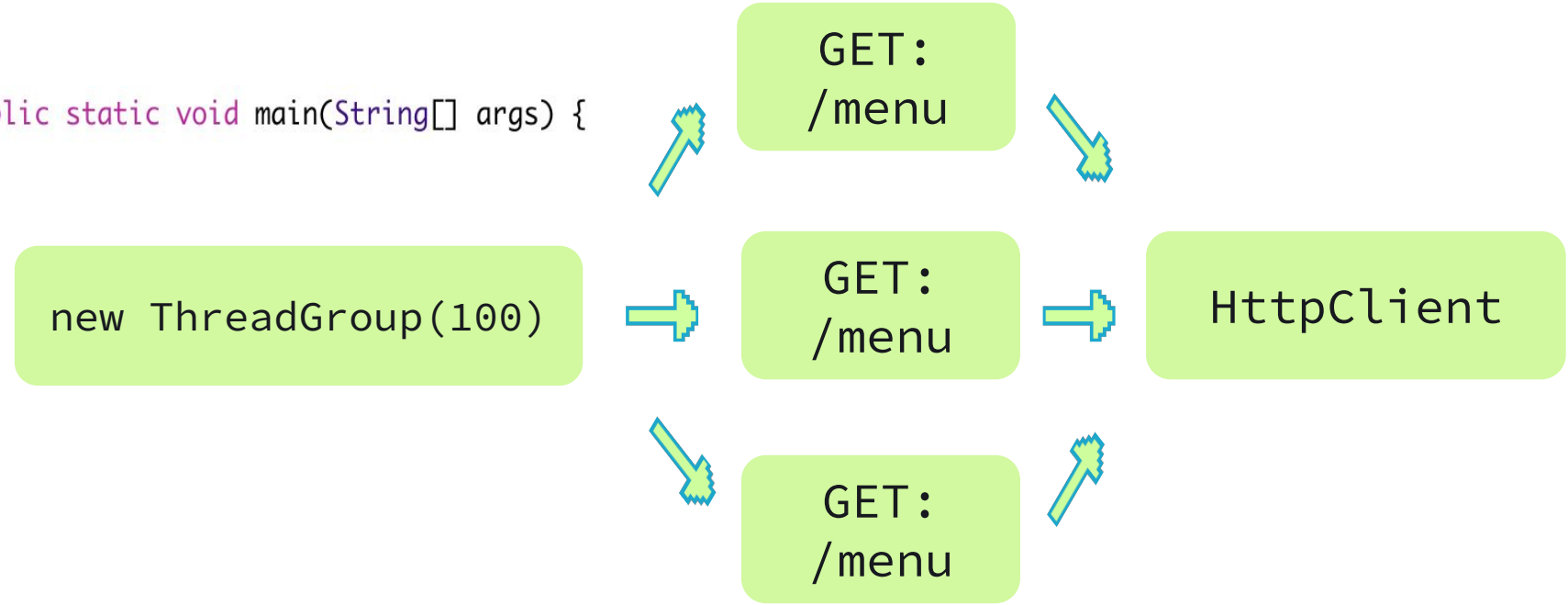
Неправильно парсим json и html

Не используем асинхронность

Не измеряем свой код

Http

```
public static void main(String[] args) {

}
```

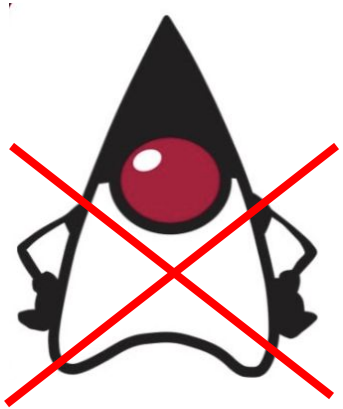


Http

GET:
/menu

GET:
/menu

GET:
/menu

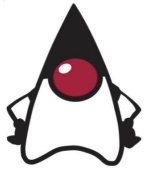


java.net.http.httpclient

Apache HttpClient

Как ускорить (распараллелить)?

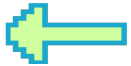
```
var response = httpClient.send(
        HttpRequest.newBuilder().GET().uri(new URI("url")).build(),
        HttpResponse.BodyHandlers.ofString()
);
// String res =  response.body()
```



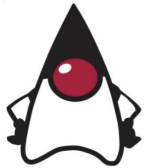
```
try (var response = httpClient.execute(new HttpGet("url"))) {
  // String res = EntityUtils.toString(response.getEntity(), UTF_8)
}
```

Как ускорить (распараллелить)?

```
httpClient.sendAsync(
        HttpRequest.newBuilder().GET().uri(new URI("url")).build(),
        HttpResponse.BodyHandlers.ofString()
).thenApply(...)
 .thenAccept(...)
```



Перепиши все на callback-и!



Просто настрой его

```
try (var response = httpClient.execute(new HttpGet("url"))) {
  // String res = EntityUtils.toString(response.getEntity(), UTF_8)
}
```

Причины отказа от Java 11 http

- Unlimited* connection pool
- Но нужен асинхронный API
- У нас (у всех) тесты на синхронном API
- Дешевле спрятать параллелизм внутрь

И кроме того:

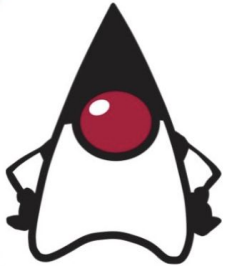
GZIP

Multipart

Documentation
(StackOverFlow)

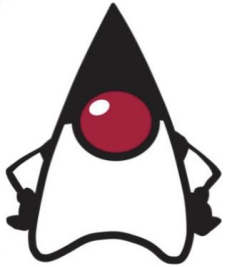
Java 11 http: overengineering

```java
private static String ungzip(byte[] compressed) {
    try {
        final ByteArrayInputStream bis = new ByteArrayInputStream(compressed);
        final InputStream gis = new GZIPInputStream(bis);
        final byte[] bytes = IOUtils.toByteArray(gis);
        return new String(bytes, StandardCharsets.UTF_8);
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}
```



Java 11 http: overengineering

```java
public static HttpRequest.BodyPublisher MultipartData(Map<Object, Object> data, String boundary) {
    var byteArrays = new ArrayList<byte[]>();
    byte[] separator = ("--" + boundary
            + "\r\nContent-Disposition: form-data; name=")
            .getBytes(StandardCharsets.UTF_8);

    for (Map.Entry<Object, Object> entry : data.entrySet()) {
        byteArrays.add(separator);

        if (entry.getValue() instanceof Path) {
            var path = (Path) entry.getValue();
            String mimeType = Files.probeContentType(path);
            byteArrays.add(("\""
                    + entry.getKey()
                    + "\"; filename=\"" + path.getFileName()
                    + "\"\r\nContent-Type: " + mimeType + "\r\n\r\n")
                    .getBytes(StandardCharsets.UTF_8));
            byteArrays.add(Files.readAllBytes(path));
            byteArrays.add("\r\n".getBytes(StandardCharsets.UTF_8));
        } else {
            byteArrays.add(("\""
                    + entry.getKey() + "\"\r\n\r\n"
                    + entry.getValue() + "\r\n")
                    .getBytes(StandardCharsets.UTF_8));
        }
    }
    byteArrays.add(("--" + boundary + "--").getBytes(StandardCharsets.UTF_8));
    return HttpRequest.BodyPublishers.ofByteArrays(byteArrays);
}
```



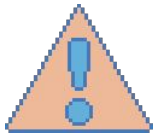
Причины переезда на Apache HTTP

```
MultipartEntityBuilder entityBuilder = MultipartEntityBuilder.create()
    .setBoundary(boundary)
    .setMode(HttpMultipartMode.BROWSER_COMPATIBLE);
```

Причины переезда на Apache HTTP

- Настраиваемый connection pool
- Параллелизм “из коробки”
- Удобный Redirect / cookie management
- Поддерживаемость

Follow-redirects / Cookies



```
response.headers().get("set-cookie");

response.headers().get("Location");
```

```
public interface CookieStore

  HttpClients.custom().setRedirectStrategy(...)
    .build()
```

Конфигурация Apache http client

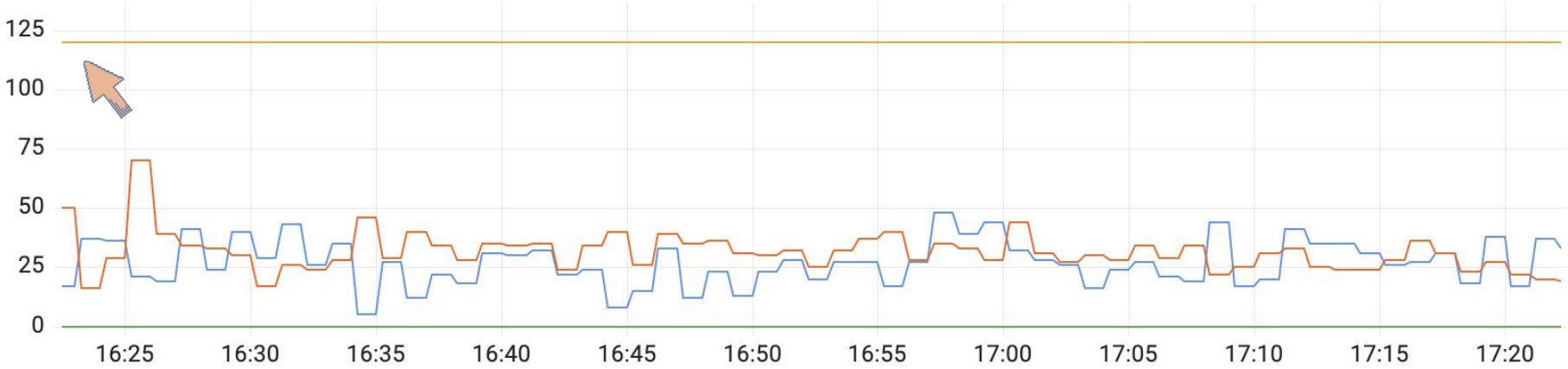
```
var connManager = new PoolingHttpClientConnectionManager();
connManager.setValidateAfterInactivity(5000);
connManager.setMaxTotal(connectionPoolSize);
connManager.setDefaultMaxPerRoute(connectionPoolSize);
```

Конфигурация Apache http client

```
var connManager = new PoolingHttpClientConnectionManager();
connManager.setValidateAfterInactivity(5000);
connManager.setMaxTotal(connectionPoolSize);
connManager.setDefaultMaxPerRoute(connectionPoolSize);
```

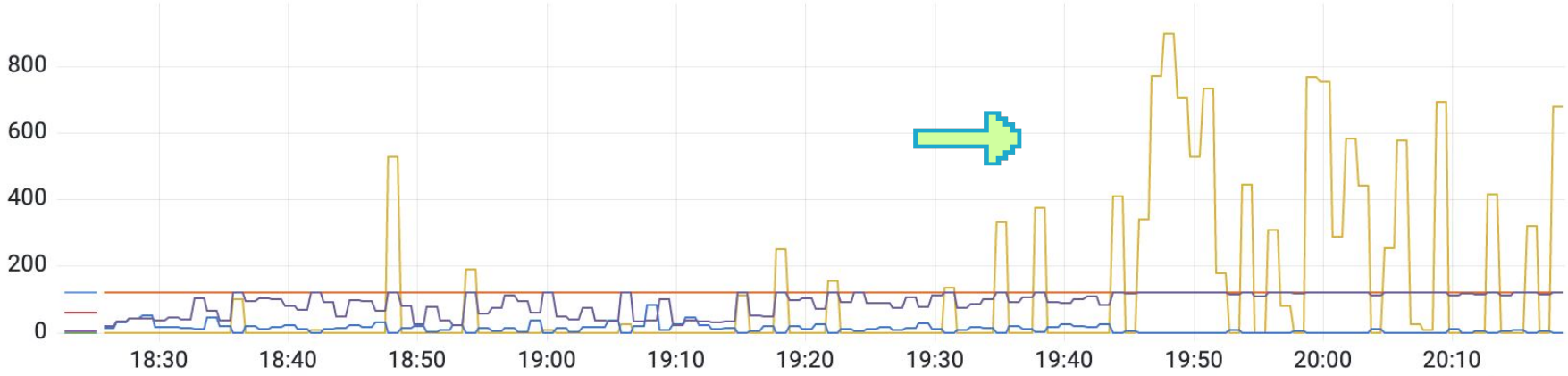
connectionPoolSize ?

Http: connectionPoolSize



	Last *	Mean	Max
perf-test-jvm-jmeter-ru-mapi-slxqn HttpclientPoolTotalMax_DEFAULT	120	120	120
perf-test-jvm-jmeter-ru-mapi-slxqn HttpclientPoolTotalConnections_DEFAULT_state_leased	19	31.4	70
perf-test-jvm-jmeter-ru-mapi-slxqn HttpclientPoolTotalConnections_DEFAULT_state_available	33	26.8	48
perf-test-jvm-jmeter-ru-mapi-slxqn HttpclientPoolTotalPending_DEFAULT	0	0	0

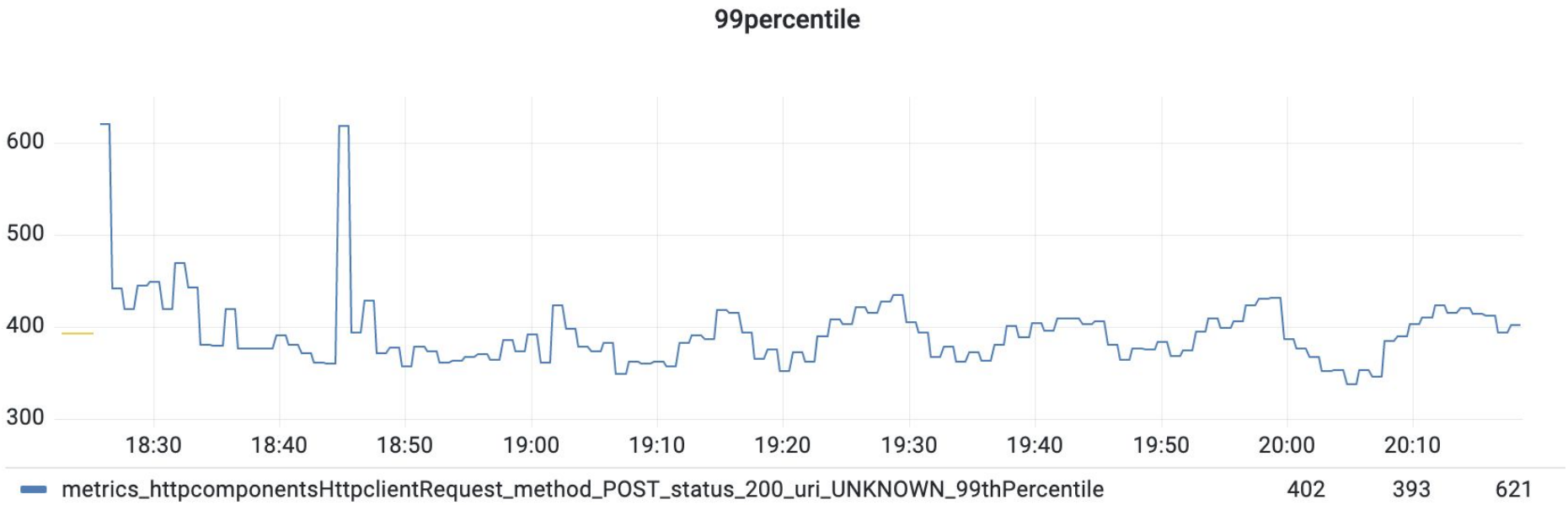
Http: connectionPoolSize



	Last *	Mean	Max
perf-test-jvm-jmeter-ru-mapi-48nmz HttpclientPoolTotalMax_DEFAULT	120	120	120
perf-test-jvm-jmeter-ru-mapi-lfggq HttpclientPoolTotalMax_DEFAULT	120	120	120
perf-test-jvm-jmeter-ru-mapi-lfggq HttpclientPoolTotalPending_DEFAULT	679	119	897
perf-test-jvm-jmeter-ru-mapi-lfggq HttpclientPoolTotalConnections_DEFAULT_state_leased	120	89.5	120

Http: Micrometer

Http: response time



Http: Micrometer – connection pool

```
new PoolingHttpClientConnectionManagerMetricsBinder(
    connManager,
    "Apache-Http " + redirectMode.name()
).bindTo(JMX_REGISTRY);
```

*где connManager – PoolingHttpClientConnectionManager

Http: Micrometer – response time

```
HttpClientBuilder clientBuilder = HttpClients.custom()
    .setConnectionManager(connManager)
    .setRequestExecutor(MicrometerHttpRequestExecutor
        .builder(JMX_REGISTRY)
        .build());
```

Http: System metrics

node_sockstat_TCP_

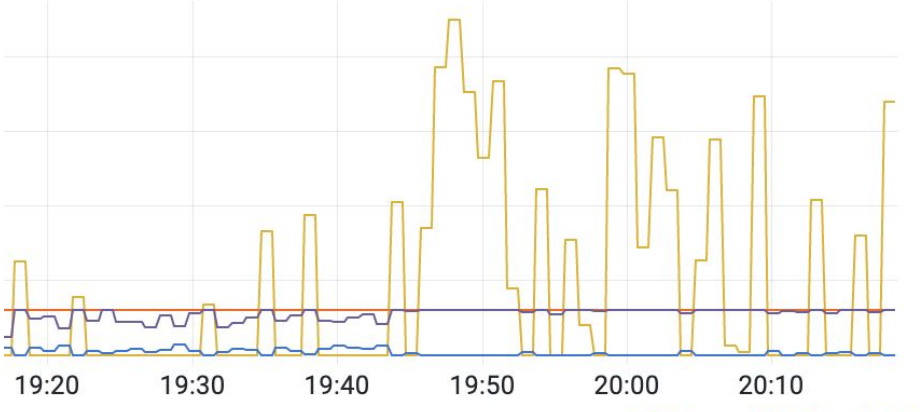
node_sockstat_TCP_alloc	gauge
node_sockstat_TCP_inuse	gauge
node_sockstat_TCP_mem	gauge
node_sockstat_TCP_mem_bytes	gauge
node_sockstat_TCP_orphan	gauge
node_sockstat_TCP_tw	gauge
node_sockstat_TCP6_inuse	gauge

Number of TCP sockets in state alloc.

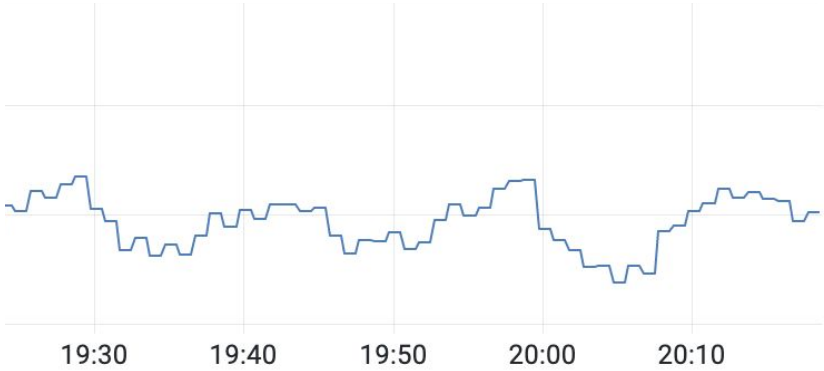
- Сокеты, диапазоны портов
- Keep-alive,tcp_tw_reuse

https://www.youtube.com/watch?v=rQCspOA30Bc

Http: еще раз о response time



Connection pending



Avg response time

Общее время выполнения сценария

$$X = (R + L + T)$$

Response time

“Бизнес логика”

Сохраняем результаты

Если дело не в response time, то в чем?

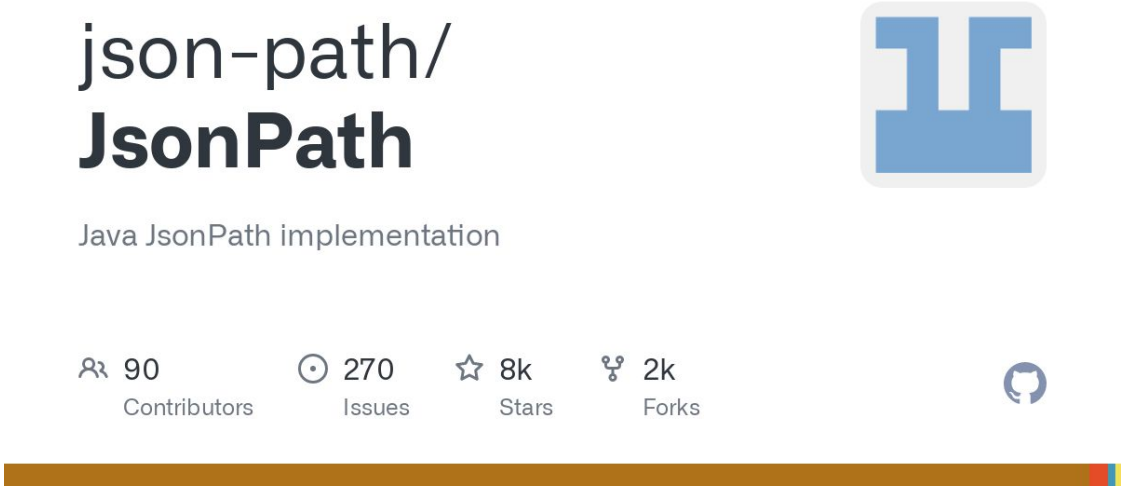
Если вычесть response time (R) из общего времени (X), то увидим проблемные места

$$X = (R + \boxed{L + T})$$

“Бизнес-логика” тестов

```
class GetMenu implements Sampler {
    @Override
    void run() {

      GET: /menu
                    response
                    parsing
                                assert*
                                          result
                                          publishing

    }
}
```

“Бизнес-логика” – читаем респонсы



Читаем респонсы – было

1

```
pizza = common.Parser.json(
        String.valueOf(response.body()),
        "$.items.[?(@.category==1)].shoppingItems[?(@.price>650)].id",
        null,
        String.class);
```

2

```
drink = common.Parser.json(
        String.valueOf(response.body()),
        "$.items.[?(@.category==2)].shoppingItems..id",
        null,
        String.class);
```

Читаем респонсы – было

3

```
sides = common.Parser.json(
            String.valueOf(response.body()),
            "$.items.[?(@.category==3)].shoppingItems..id",
            null,
            String.class);
```

4

```
sauce = common.Parser.json(
            String.valueOf(response.body()),
            "$.items.[?(@.category==4)].shoppingItems..id",
            null,
            String.class);
```

Читаем респонсы – было

5

```
dessert = common.Parser.json(
        String.valueOf(response.body()),
        "$.items.[?(@.category==6)].shoppingItems..id",
        null,
        String.class);
```

6

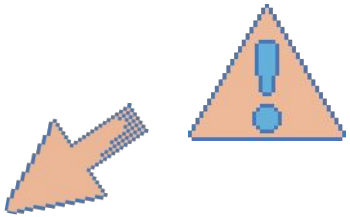
```
goods = common.Parser.json(
        String.valueOf(response.body()),
        "$.items.[?(@.category==5)].shoppingItems..id",
        null,
        String.class);
```

Читаем респонсы – было

6 Аллокаций стрингов (1,5 Мб * 6)

6 LinkedHashMap внутри JsonProcessor

Статический метод провоцирует на **элементарные** ошибки



```
goods = common.Parser.json(
        String.valueOf(response.body()),
        "$.items.[?(@.category==5)].shoppingItems..id",
        null,
        String.class);
```

Статический метод провоцирует на **элементарные** ошибки

```
public static String json(String jsonString,
                          String jsonPath,
                          Integer index,
                          Class<?> returnType) {

 Object parsedJson = JsonPath.read(jsonString, jsonPath);
```

Решение

- Долой String* (до востребования)
- Долой статические методы
- Сохранять респонсы в immutable объект

Читаем респонсы – стало

```
sides = httpContext.getResponseParser().json(
    "$.items.[?(@.category==3)].shoppingItems..id");

sauce = httpContext.getResponseParser().json(
    "$.items.[?(@.category==4)].shoppingItems..id");
```

Читаем респонсы – стало

```
sides = httpContext.getResponseParser().json(
    "$.items.[?(@.category==3)].shoppingItems..id");

sauce = httpContext.getResponseParser().json(
    "$.items.[?(@.category==4)].shoppingItems..id");
```

Читаем респонсы – стало

```
public class CommonHttpContext implements HttpContext {

  private final HttpRequestBase request;
  private final CloseableHttpResponse response;
  private final JsonParser responseJsonParser;
```

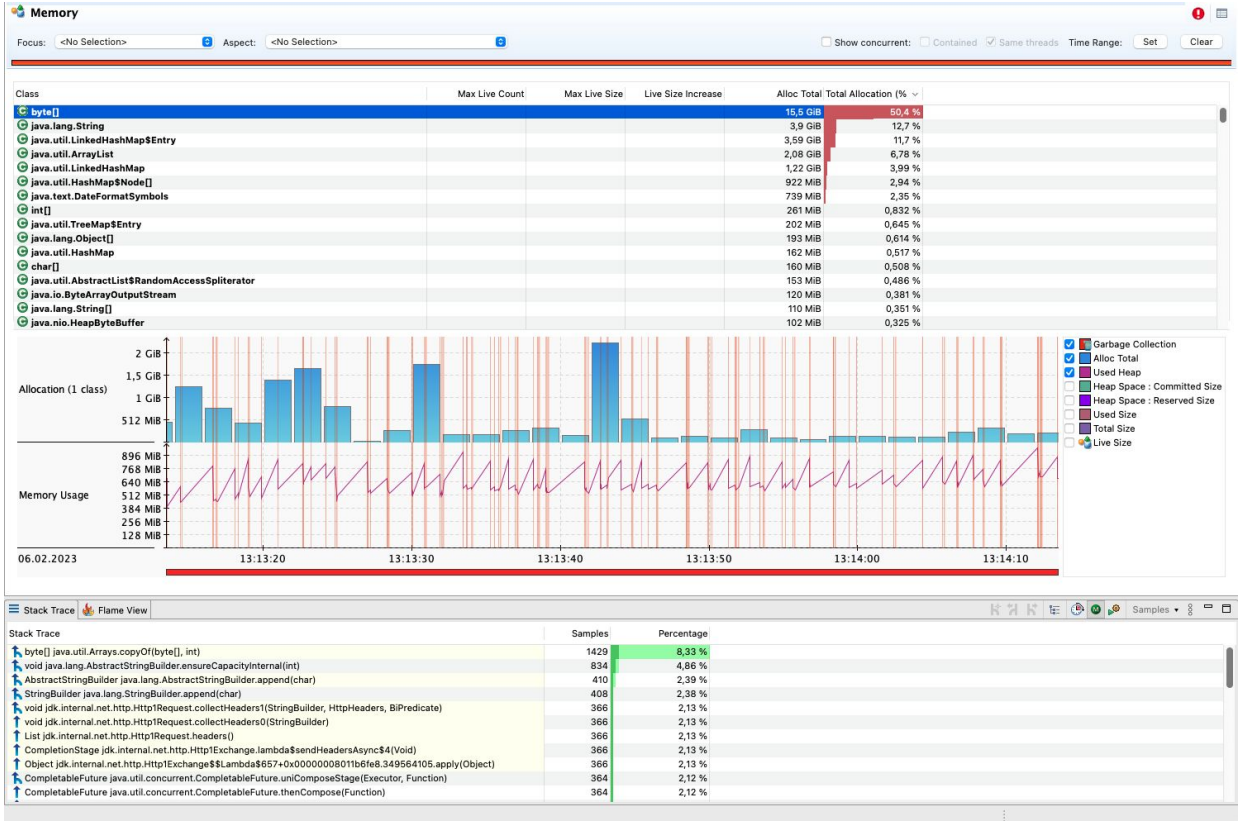
```
    public class InMemJsonParser implements JsonParser {

      private final DocumentContext documentContext;
```

Читаем респонсы – стало

0 * Аллокаций стрингов (используем InputStream)

1 LinkedHashMap внутри JsonProcessor

Как заметить проблему?



Как заметить проблему?

Memory

Focus: <No Selection> Aspect: <No Selection> Show concurrent: Contained Same threads Time Range: Set Clear

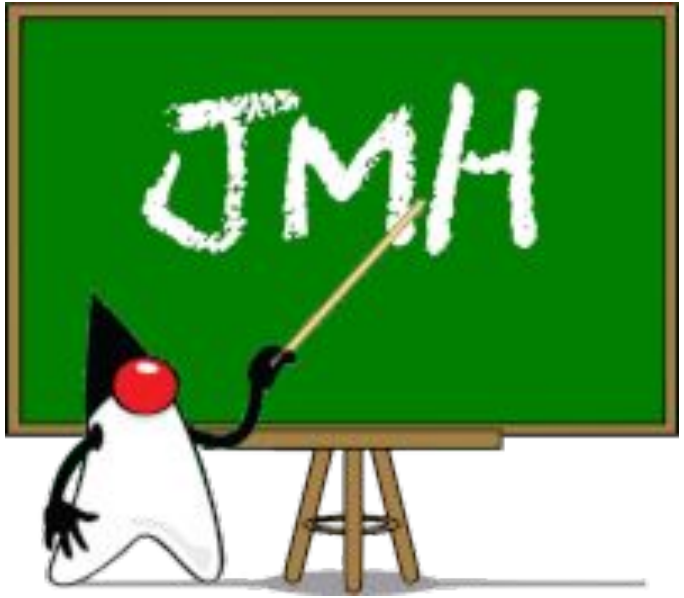
Class	Max Live Count	Max Live Size	Live Size Increase	Alloc Total	Total Allocation (%
byte[]				15,5 GiB	50,4 %
java.lang.String				3,9 GiB	12,7 %
java.util.LinkedHashMap$Entry				3,59 GiB	11,7 %
java.util.ArrayList				2,08 GiB	6,78 %
java.util.LinkedHashMap				1,22 GiB	3,99 %
java.util.HashMap$Node[]				922 MiB	2,94 %
java.text.DateFormatSymbols				739 MiB	2,35 %
int[]				261 MiB	0,832 %
java.util.TreeMap$Entry				202 MiB	0,645 %
java.lang.Object[]				193 MiB	0,614 %
java.util.HashMap				162 MiB	0,517 %

Class	Alloc Total	Total Allocation (%
byte[]	15,5 GiB	50,4 %
java.lang.String	3,9 GiB	12,7 %
java.util.LinkedHashMap$Entry	3,59 GiB	11,7 %

Stack Trace | Flame View

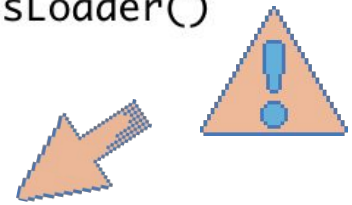
Stack Trace	Samples	Percentage
byte[] java.util.Arrays.copyOf(byte[], int)	1429	8,33 %
void java.lang.AbstractStringBuilder.ensureCapacityInternal(int)	834	4,86 %
AbstractStringBuilder java.lang.AbstractStringBuilder.append(char)	410	2,39 %
StringBuilder java.lang.StringBuilder.append(char)	408	2,38 %
void jdk.internal.net.http.Http1Request.collectHeaders1(StringBuilder, HttpHeaders, BiPredicate)	366	2,13 %
void jdk.internal.net.http.Http1Request.collectHeaders0(StringBuilder)	366	2,13 %
List jdk.internal.net.http.Http1Request.headers()	366	2,13 %
CompletionStage jdk.internal.net.http.Http1Exchange.lambda$sendHeadersAsync$4(Void)	366	2,13 %
Object jdk.internal.net.http.Http1Exchange$$Lambda$657+0x00000008011b6fe8.349564105.apply(Object)	366	2,13 %
CompletableFuture java.util.concurrent.CompletableFuture.uniComposeStage(Executor, Function)	364	2,12 %
CompletableFuture java.util.concurrent.CompletableFuture.thenCompose(Function)	364	2,12 %

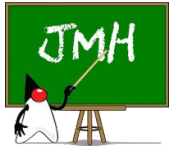
Доказываем корректность решения – JMH



Доказываем корректность решения – JMH

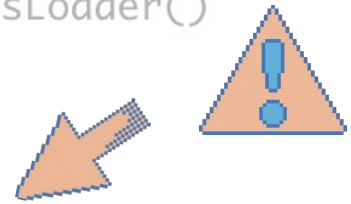
```
@Benchmark
public void documentContextPerQueryBenchmark() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("menu_delivery.json")) {
    byte[] bytes = resource.readAllBytes();

    Object result = PC.parse(new String(bytes, UTF_8))
        .read("$.items[?(@.category==1)].shoppingItems[?(@.price>650)].id");
    Object drink = PC.parse(new String(bytes, UTF_8))
        .read("$.items[?(@.category==2)].shoppingItems..id");

        ....
```

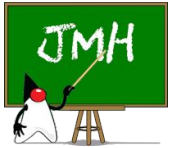




Доказываем корректность решения – JMH

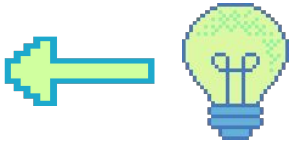
```
@Benchmark
public void documentContextPerQueryBenchmark() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("menu_delivery.json")) {
    byte[] bytes = resource.readAllBytes();

    Object result = PC.parse(new String(bytes, UTF_8))
        .read("$.items[?(@.category==1)].shoppingItems[?(@.price>650)].id");
    Object drink = PC.parse(new String(bytes, UTF_8))
        .read("$.items[?(@.category==2)].shoppingItems..id");

        ....
```

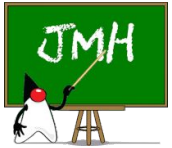




Доказываем корректность решения (JMH)

```
@Benchmark
public void singleDocumentContextBenchmark() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("menu_delivery.json")) {
    DocumentContext documentContext = PC.parse(resource);

    Object result = documentContext
        .read("$.items[?(@.category==1)].shoppingItems[?(@.price>650)].id");

        ...
```





Доказываем корректность решения (JMH)

Benchmark	Metric	Score	Units
documentContextPerQuery	performance	16.447	ops/sec
documentContextPerQuery	allocation rate	1887.68	MB/sec
singleDocumentContext	performance	69.356	ops/sec
singleDocumentContext	allocation rate	658.84	MB/sec

А можно еще лучше?

* Аллокаций стрингов?

* LinkedHashMap внутри JsonProcessor?

Streaming parsing

```
Collector collector = surfer.collector(inputStream);

ValueBox<Object> box1 = collector.collectOne(
    "$.items[?(@.category==1)].shoppingItems[?(@.price>650)].id");
ValueBox<Object> box2 = collector.collectOne(
    "$.items[?(@.category==2)].shoppingItems..id");
...
collector.exec();

box1.get();
box2.get();
```

Streaming parsing

wanglingsong/
JsonSurfer

A streaming JsonPath processor in Java

9 Contributors | 2 Used by | 263 Stars | 47 Forks

Доказываем корректность решения (JMH)

```
@Benchmark
public void jsurferBenchmark() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("menu_delivery.json")) {
    Collector collector = surfer.collector(resource);
    ValueBox<Object> pizza = collector
        .collectOne("$.items[?(@.category==1)].shoppingItems[?(@.price>650)].id");
    ValueBox<Object> drink = collector.
        collectOne("$.items[?(@.category==2)].shoppingItems..id");

        ...
```

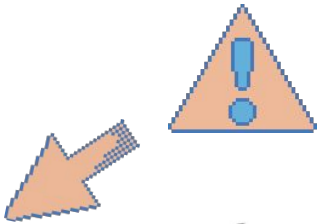
Доказываем корректность решения (JMH)

Benchmark	Metric	Score	Units
documentContextPerQuery	performance	16.447	ops/sec
documentContextPerQuery	allocation rate	1887.68	MB/sec
singleDocumentContext	performance	69.356	ops/sec
singleDocumentContext	allocation rate	658.84	MB/sec
jsurfer	performance	57.822	ops/sec
jsurfer	allocation rate	313.49	MB/sec

Промежуточные итоги

- InputStream – ваш лучший друг (а не String)
- Streaming processing может сильно помочь
- Статические методы провоцируют на ошибки
- Даже самый простой @Benchmark лучше никакого

Читаем HTML – было



```
String getManagmentDevicesDoughRole = common.Parser.css(
        response.body().toString(),
        "#SelectedRole option[value=tracking_23]");

String getManagmentDevicesHotAssemblyRole = common.Parser.css(
        response.body().toString(),
        "#SelectedRole option[value=tracking_24]");
```

Читаем HTML – стало

```
String getManagmentDevicesDoughRole = httpContext
        .getResponseHtmlParser()
        .css("#SelectedRole option[value=tracking_23]");
```

Читаем HTML – вечный спор

https://youtu.be/d4iJrWsdT4E

https://qagroup.com.ua/publications/xpath-vs-css-selector/

Читаем HTML – вечный спор

<htmlcleaner/>

transforms HTML to well-formed XML

Jsoup

code4craft/**xsoup**

When jsoup meets XPath.

9 Contributors | 27 Issues | 445 Stars | 151 Forks

Если все-таки нужен css и xpath

```
private final Document doc; // JSOUP
private final Node node; // HTML Cleaner

public InMemHtmlParser(@Nonnull InputStream is) throws IOException, ParserConfigurationException {
  ReusableInputStream ris = new ReusableInputStream(new BufferedInputStream(is));
  try {
    this.doc = Jsoup.parse(ris, StandardCharsets.UTF_8.toString(), "");
    this.node = new DomSerializer(new CleanerProperties()).createDOM(
        new HtmlCleaner().clean(ris, StandardCharsets.UTF_8.toString())
    );
  } finally {
    ris.getDecorated().close();
  }
}
```

Если все-таки нужен css и xpath

```
private final Document doc; // JSOUP
private final Node node; // HTML Cleaner

public InMemHtmlParser(@Nonnull InputStream is) throws IOException, ParserConfigurationException {
  ReusableInputStream ris = new ReusableInputStream(new BufferedInputStream(is));
  try {
    this.doc = Jsoup.parse(ris, StandardCharsets.UTF_8.toString(), "");
    this.node = new DomSerializer(new CleanerProperties()).createDOM(
        new HtmlCleaner().clean(ris, StandardCharsets.UTF_8.toString())
    );
  } finally {
    ris.getDecorated().close();
  }
}
```

Fail fast

Dmitry Tuchs 4:04 PM

у нас в некоторых классах Auth есть такой xpath

`"//div[contains(@class,'device_function') and contains(.,'Доставка')]/preceding-sibling::*[@class='device_id']/@value"`

Dmitry Tuchs 6:51 PM

Еще нерабочий регехр

`"viewOrderDetails.\\\\u0027([0-9]{1,})\\\\u0027."`

он никогда ничего не находит

Промежуточные итоги

- InputStream – ваш лучший друг (а не String)
- Откажитесь от xpath по возможности
- Если нет, то ReusableInputStream
- Все, сказанное про static методы, в силе

Регулярные выражения

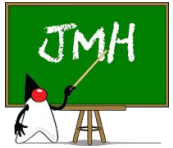
```
<div id="root"></div>
<script type="text/javascript">
  var userModel = {
  "unitRoles": [
    {
      "name": "DeliveryCashier"
    }
  ],
  "departmentRoles": [],
  "IsIntegrationMessageEnabled":
};
    var userInfo = {
  "FirstName": "",
```

Регулярные выражения

```
String getAuthenticateRolesRegex = common.Parser.regex(
        String.valueOf(response.body()),
        "var userModel =((.+?\\n)+(.+?));\\n\\tvar userInfo",
        1,
        -1);
```

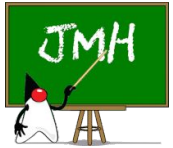
Регулярные выражения – измерим

```
@Benchmark
public void substringBetween() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("response.html")) {
      String result = StringUtils.substringBetween(new String(resource.readAllBytes(), UTF_8), "var userModel = ", ";");
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
}

@Benchmark
public void regex() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("response.html")) {
      Pattern pattern = Pattern.compile("var userModel = ((.+?\\n)+(.+?));\\n\\tvar userInfo", Pattern.UNIX_LINES);
      Matcher matcher = pattern.matcher(new String(resource.readAllBytes(), UTF_8));
      if (matcher.find()) {
        String result = matcher.group(1);
      }
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
}
```



Регулярные выражения – измерим

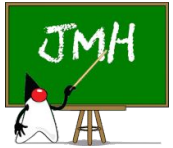
```
@Benchmark
public void substringBetween() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("response.html")) {
      String result = StringUtils.substringBetween(new String(resource.readAllBytes(), UTF_8), "var userModel = ", ";");
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
}

@Benchmark
public void regex() {
  try (InputStream resource = MyBenchmark.class.getClassLoader()
      .getResourceAsStream("response.html")) {
      Pattern pattern = Pattern.compile("var userModel = ((.+?\\n)+(.+?));\\n\\tvar userInfo", Pattern.UNIX_LINES);
      Matcher matcher = pattern.matcher(new String(resource.readAllBytes(), UTF_8));
      if (matcher.find()) {
        String result = matcher.group(1);
      }
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
}
```



Регулярные выражения – измерим

```
private static final Pattern pattern =
    Pattern.compile(
        "var userModel = ((.+?\\n)+(.+?));\\n\\tvar userInfo",
        Pattern.UNIX_LINES
        );
```

```
1    <!doctype html>
2    <html lang="ru-RU" dir="ltr">
3    <head>
4        <title>Выбор роли</title>
126  <script src="/app/dist/auth.bundle.js?v=1001.14028"
127  </body>
128  </html>
```

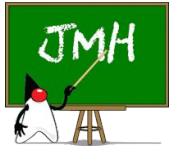


Доказываем корректность решения (JMH)

Benchmark	Metric	Score	Units
regexp	performance	22997	ops/sec
precompiledRegexp	performance	23865	ops/sec
substring	performance	43635	ops/sec

Немного усложним задачу…

```
@Benchmark
public void substringBetween() {
  ...
    for (int i = 0; i < 1000; i++) {
      String result = StringUtils.substringBetween(new String(resource.readAllBytes(), UTF_8), "var userModel = ", ";");
    }
  ...
}

@Benchmark
public void regex() {
  ...
    for (int i = 0; i < 1000; i++) {
      Pattern pattern = Pattern.compile("var userModel = ((.+?\\n)+(.+?));\\n\\tvar userInfo", Pattern.UNIX_LINES);
      Matcher matcher = pattern.matcher(new String(resource.readAllBytes(), UTF_8));
      if (matcher.find()) {
        String result = matcher.group(1);
      }
    }
  ...
}
```



Результат

Benchmark	Metric	Score	Units
precompiledRegexp	performance	1207	ops/sec
substring	performance	1197	ops/sec

Промежуточные итоги

- substring() неплох для извлечения данных из небольших файлов
- Регулярки надо заранее компилировать
- Даже самый простой @Benchmark лучше никакого

Публикация результатов

```
class GetMenu implements Sampler {
    @Override
    void run() {

        GET: /menu
                    response
                     parsing
                                assert*
                                          result
                                         publishing

    }
}
```

Публикация результатов

Тест уже завершился. Почему бы не освободить поток?

Публикация результатов

```
@Override
public void handleSampleResults(List<SampleResult> results, BackendListenerContext context) {
  final KafkaHandler kh = new KafkaHandler(
      results,
      context,
      publisher,
      buildNumber,
      fields,
      filters
  );

  CompletableFuture.runAsync(kh, EXECUTOR);
}
```

Публикация результатов

```
EXECUTOR = Executors.newFixedThreadPool(25);
```

vs

```
EXECUTOR = Executors.newCachedThreadPool();
```

Публикация результатов

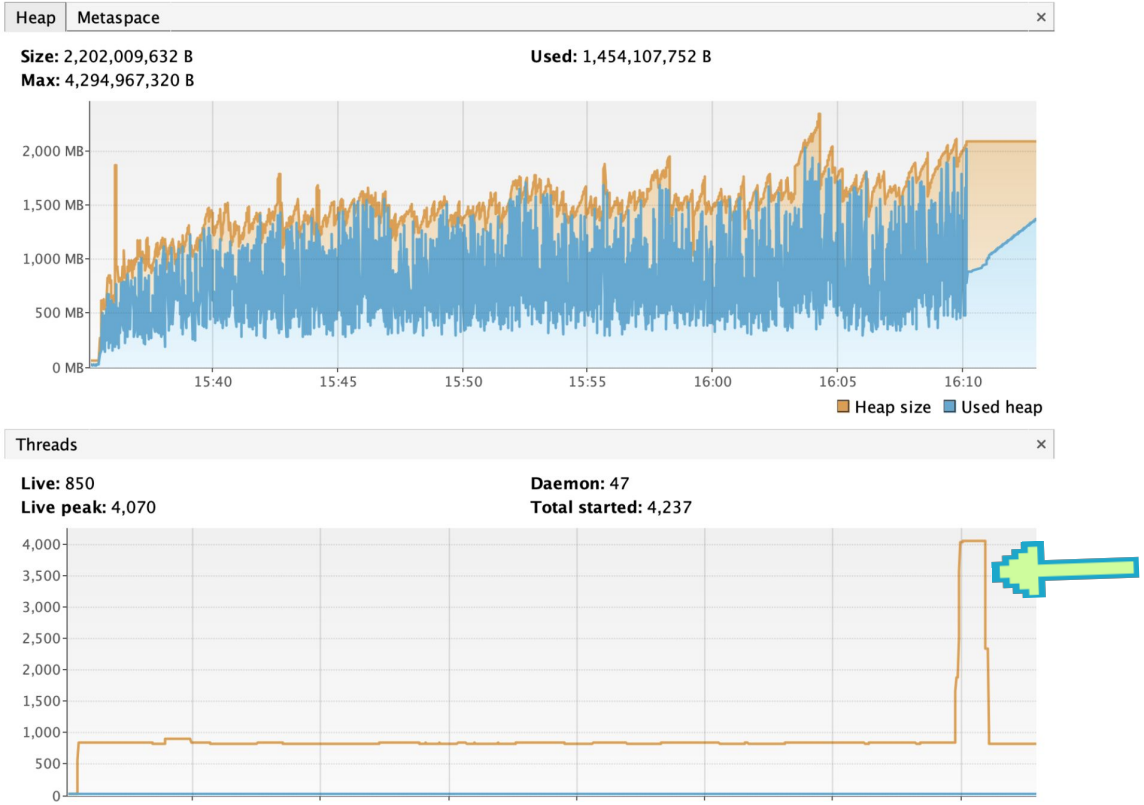
```
private static final ExecutorService EXECUTOR;

static {
  EXECUTOR = Executors.newFixedThreadPool( nThreads: 25);
  new ExecutorServiceMetrics(EXECUTOR,  executorServiceName: "Kafka-executor",  tags: null)
      .bindTo(Jmx.JMX_REGISTRY);
}
```

Публикация результатов: FixedThreadPool

executor.pool.size

executor.active

Публикация результатов: CachedThreadPool



One More Thing
java.lang.String

One More Thing

String-и все равно повсюду

One More Thing

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence,
               Constable, ConstantDesc {

    /** The value is used for character storage. ...*/
    @Stable
    private final byte[] value;
```

One More Thing

Что, если все одинаковые String-и будут смотреть на один и тот же byte[]?

One More Thing

```
"-Xmx6g",
"-XX:+UseG1GC",
"-XX:+UseStringDeduplication",
```

Выводы

Неправильно конфигурируем http

Параллелизм, валидация, измеряемость

Неправильно считаем чис потоков

Мин. число потоков под задачу

Шлем в кафку то, что не

Отправляем мин. необходимую информацию

Неп парс

OOP + InputStream + streaming processing

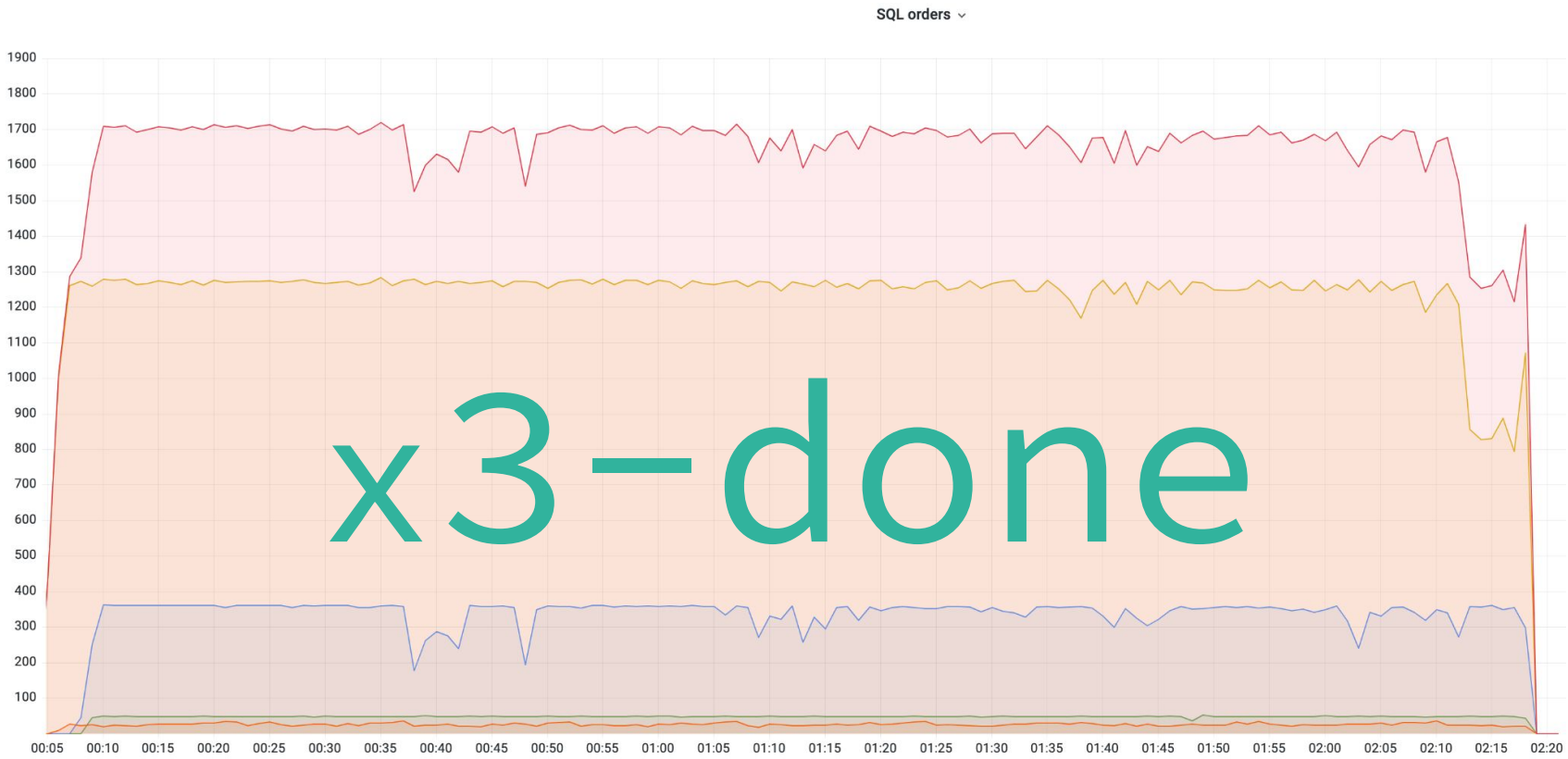
Не используем

Async handleSamleResult()

Не из сво

Micrometer, JMH, Grafana

Выводы


SQL orders
1900
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
00:05
00:10
00:15
00:20
00:25
00:30
00:35
00:40
00:45
00:50
00:55
01:00
01:05
01:10
01:15
01:20
01:25
01:30
01:35
01:40
01:45
01:50
01:55
02:00
02:05
02:10
02:15
02:20
x3-done

Выводы

- Когда идет речь о тысячах потоков, надо измерять
- Измерять надо и runtime, и в процессе разработки
- Пишите нормальный* код, это интересно

Выводы

- Знайте и уважайте свои инструменты

linkedin.com/in/dtuchs/

dtuchs

dtuchs@gmail.com