



HEISENBUG

Ускоряем Apache.JMeter

Вячеслав Смирнов

Эксперт по тестированию,
Райффайзенбанк

Бываю на конференциях





APACHE
JMeter™



Apache.JMeter

- <https://jmeter.apache.org>, с 2003 по настоящее время, активен
- HTTP(S), JMS, JDBC, Java/Groovy, FTP, SMTP, ... 50+
- ClickHouse, InfluxDB, Graphite, Grafana, HTML
- Распределённый запуск, CI
- Maven, IntelliJ IDEA
- Сообщество



Рассмотрим

1. HTTP Request. Максимальная интенсивность
2. HTTP Request. Скачивание и отправка
3. PostProcessor
4. PreProcessor
5. Секретное оружие



Цели

Разрабатывать такие тесты,
чтобы система тормозила, а не **Apache.JMeter**.

Экономить на ресурсах нагрузочных агентов.

Разрабатывать тесты быстро и просто.

HTTP Request

Максимальная интенсивность

Samplers.HTTP.Request.X. Простой тест

10

Несколько GET-запросов на локальный сервер

Samplers.HTTP.Request.X.jmx (/home/x1337/Project/Apache.JMeter.Benchmark.NG/src/test/jmeter/Samplers.HTTP.Request.X.jmx) - Apache JMeter (5.1.1 r1855137)

File Edit Search Run Options Tools Help

00:00:00 0 0/0

- Samplers.HTTP.Request.X
 - setUp Thread Group
 - OS Process Sampler: mkdir
 - JSR223 PostProcessor: set ignore
 - OS Process Sampler: nginx.start
 - JSR223 PostProcessor: set ignore
 - Thread Group
 - Loop Controller
 - HTTP Request**
 - tearDown Thread Group
 - OS Process Sampler: nginx.stop
 - JSR223 PostProcessor: set ignore
 - Backend Listener

HTTP Request

Name: HTTP Request

Comments:

Basic **Advanced**

Web Server

Protocol [http]: **Server Name or IP:** \${RequestHost} **Port Number:** stPort

HTTP Request

Method: GET **Path:** \${RequestPath} **Content encoding:**

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters **Body Data** **Files Upload**

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?

Detail **Add** **Add from Clipboard** **Delete** **Up** **Down**

Запуск и остановка Nginx в тесте, используя OS Process Sampler

11

The screenshot displays the Apache JMeter GUI. The left sidebar shows a test plan named 'Samplers.HTTP.Request.X'. It contains a 'setUp Thread Group' with two 'OS Process Sampler' elements (one for 'mkdir' and one for 'nginx.start'), followed by a 'Thread Group' with a 'Loop Controller' and an 'HTTP Request' sampler, and finally a 'tearDown Thread Group' with two 'OS Process Sampler' elements (one for 'nginx.stop' and one for 'JSR223 PostProcessor: set ignore'). The 'HTTP Request' sampler is selected and its configuration is shown on the right.

File Edit Search Run Options Tools Help

00:00:00 0 0/0

Samplers.HTTP.Request.X

- setUp Thread Group
 - OS Process Sampler: mkdir
 - JSR223 PostProcessor: set ignore
 - OS Process Sampler: nginx.start
 - JSR223 PostProcessor: set ignore
- Thread Group
 - Loop Controller
 - HTTP Request**
- tearDown Thread Group
 - OS Process Sampler: nginx.stop
 - JSR223 PostProcessor: set ignore
- Backend Listener

HTTP Request

Name: HTTP Request

Comments:

Basic **Advanced**

Web Server

Protocol [http]: ☐ Server Name or IP: \${RequestHost} Port Number: stPort

HTTP Request

Method: GET Path: \${RequestPath} Content encoding: ☐

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters **Body Data** **Files Upload**

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?

Detail **Add** **Add from Clipboard** **Delete** **Up** **Down**

Для скорости — два процесса вместо одного

```
01.worker_processes 2; # default 1
02.worker_connections 512;
03.sendfile off;
04.tcp_nodelay on;
05.tcp_nopush off;
06.keepalive_timeout 75s;
07.keepalive_requests 100;
08.keepalive_disable msie6;
09.gzip off;
```

Настраиваются параметры HTTP-запроса

13

Параметризованы хост, порт, путь. Все настройки — по умолчанию.

The screenshot displays the Apache JMeter interface. On the left, the 'Tree' view shows a hierarchy of samplers. The 'Thread Group' is selected, and its children are expanded. The 'HTTP Request' sampler is highlighted with a red box. The main panel on the right shows the configuration for this sampler. The 'Basic' tab is active, and the 'Web Server' section is expanded. The 'Protocol' is set to 'http'. The 'Server Name or IP' is set to '\${RequestHost}', and the 'Port Number' is set to 'stPort'. The 'HTTP Request' section shows the 'Method' as 'GET' and the 'Path' as '\${RequestPath}'. The 'Content encoding' is set to an empty field. Below this, there are checkboxes for 'Redirect Automatically', 'Follow Redirects', 'Use KeepAlive', 'Use multipart/form-data', and 'Browser-compatible headers'. The 'Parameters' tab is also visible, showing a table for 'Send Parameters With the Request'.

File Edit Search Run Options Tools Help

Samplers.HTTPRequest.X.jmx (/home/x1337/Project/Apache.JMeter.Benchmark.NG/src/test/jmeter/Samplers.HTTPRequest.X.jmx) - Apache JMeter (5.1.1 r1855137)

00:00:00 0 0/0

Tree View:

- Samplers.HTTPRequest.X
 - setUp Thread Group
 - OS Process Sampler: mkdir
 - JSR223 PostProcessor: set ignore
 - OS Process Sampler: nginx.start
 - JSR223 PostProcessor: set ignore
 - Thread Group**
 - Loop Controller
 - HTTP Request**
 - tearDown Thread Group
 - OS Process Sampler: nginx.stop
 - JSR223 PostProcessor: set ignore
 - Backend Listener

HTTP Request Configuration:

Name: HTTP Request

Comments:

Basic Advanced

Web Server

Protocol [http]: ☐ **Server Name or IP:** \${RequestHost} **Port Number:** stPort

HTTP Request

Method: GET **Path:** \${RequestPath} **Content encoding:**

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

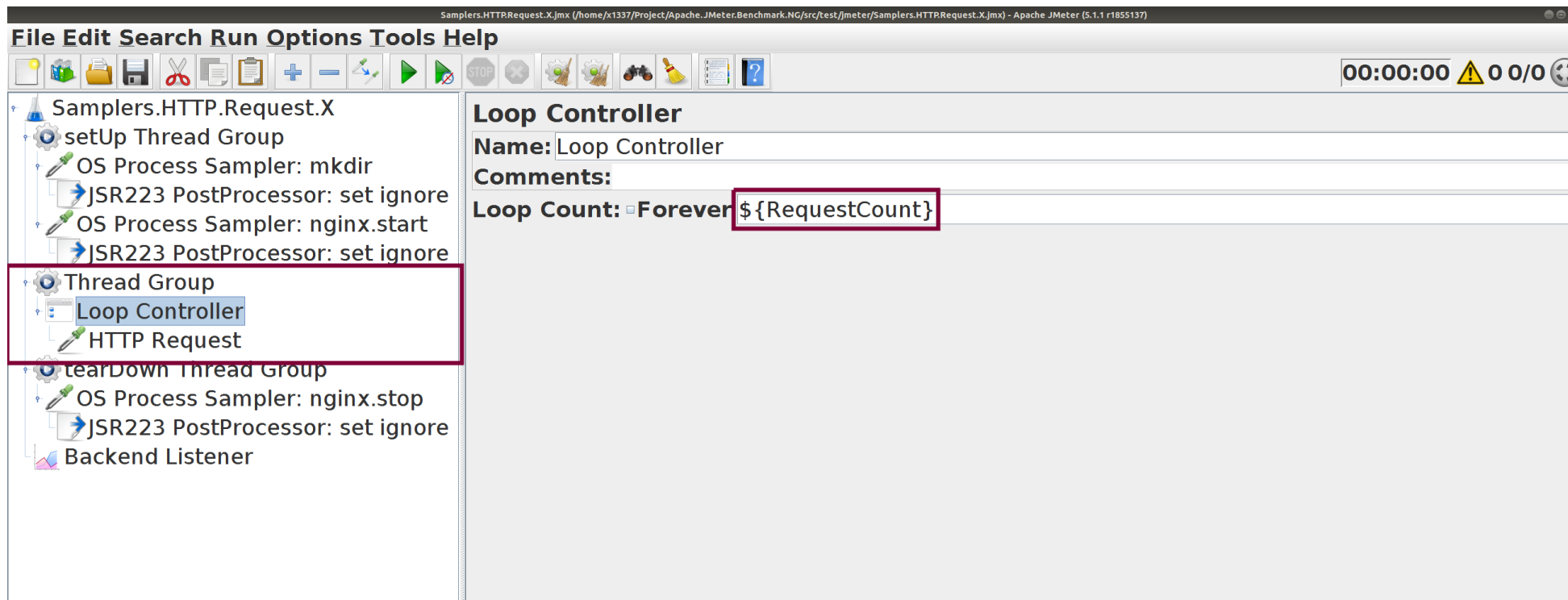
Name:	Value	URL Encode?	Content-Type	Include Equals?

Detail Add Add from Clipboard Delete Up Down

Используется цикл для цепочки запросов

14

`${RequestCount}` последовательных запросов в каждой итерации



Количество потоков и итераций настраивается

15

Профиль — стабильная высокая нагрузка от **`${Threads}`** потоков

The screenshot displays the Apache JMeter interface. On the left, the 'Tree' view shows a test plan structure: 'Samplers.HTTP.Request.X' (expanded) contains 'setUp Thread Group', 'OS Process Sampler: mkdir' (with JSR223 PostProcessor: set ignore), 'OS Process Sampler: nginx.start' (with JSR223 PostProcessor: set ignore), 'Thread Group' (highlighted with a red box), 'tearDown Thread Group', 'OS Process Sampler: nginx.stop' (with JSR223 PostProcessor: set ignore), and 'Backend Listener'. The 'Thread Group' is selected, and its configuration is shown on the right. The 'Thread Group' configuration includes: 'Name: Thread Group', 'Comments:', 'Action to be taken after a Sampler error' (with radio buttons for Continue, Start Next Thread Loop, Stop Thread, Stop Test, and Stop Test Now), 'Thread Properties' (with 'Number of Threads (users):' set to `${Threads}` and 'Ramp-Up Period (in seconds):' set to `${RampUp}`), 'Loop Count' (set to 'Forever' with `${LoopCount}`), 'Delay Thread creation until needed' (checked), 'Scheduler' (checked), 'Scheduler Configuration' (with 'If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count * iteration)'), 'Duration (seconds)', and 'Startup delay (seconds)'.

Samplers.HTTP.Request.X.jmx (/home/x1337/Project/Apache.JMeter.Benchmark.NG/src/test/jmeter/Samplers.HTTP.Request.X.jmx) - Apache JMeter (5.1.1 r1855137)

File Edit Search Run Options Tools Help

00:00:00 ⚠ 0 0/0

Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

- Continue • Start Next Thread Loop • Stop Thread • Stop Test • Stop Test Now

Thread Properties

Number of Threads (users): `${Threads}`

Ramp-Up Period (in seconds): `${RampUp}`

Loop Count: ☒ Forever `${LoopCount}`

☒ Delay Thread creation until needed

☒ Scheduler

Scheduler Configuration

▲ If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count * iteration)

Duration (seconds)

Startup delay (seconds)

HTTP Request. Серия экспериментов

16

Количество потоков **\${Threads}**:

-  1, 2, 3

Количество последовательных запросов **\${RequestCount}**:

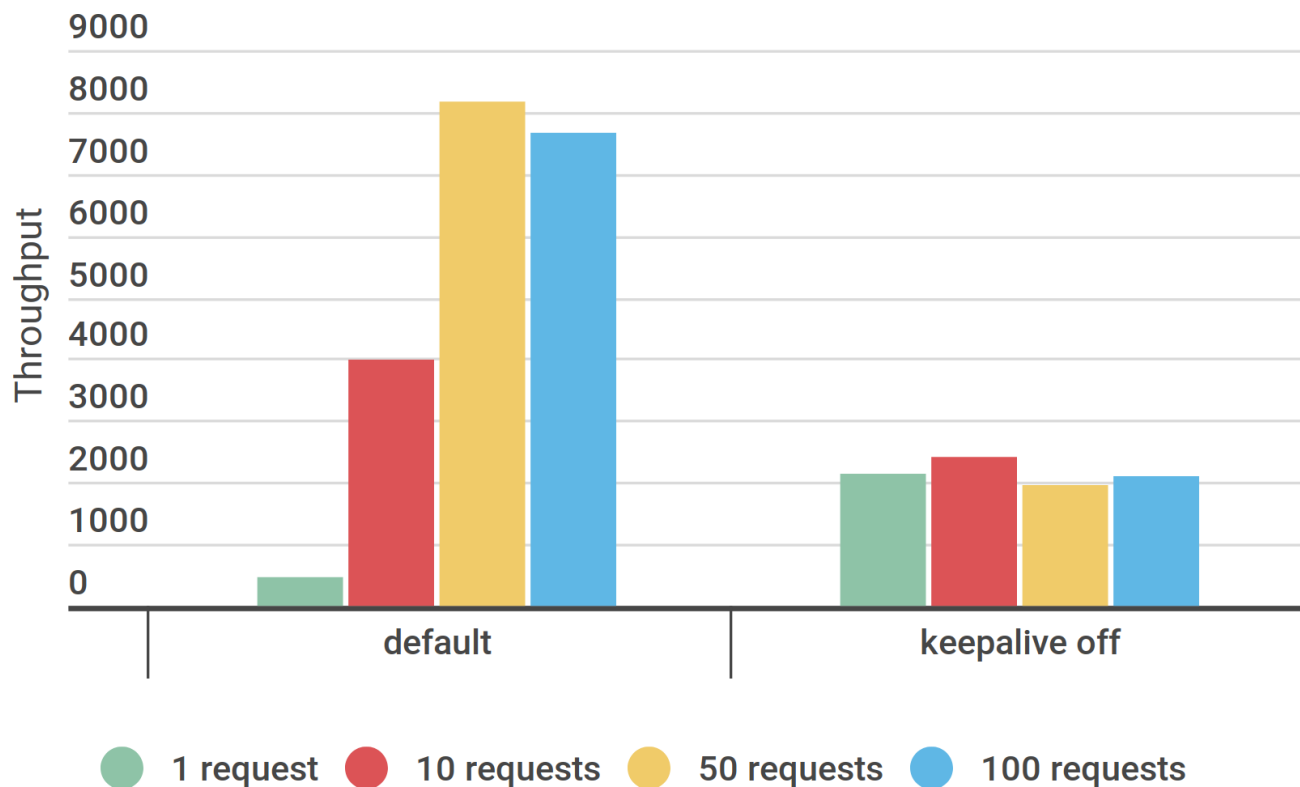
-  1, 10, 50, 100

HTTP Request **Keep-Alive**:

- ☒ включить (по умолчанию)
- ☐ отключить

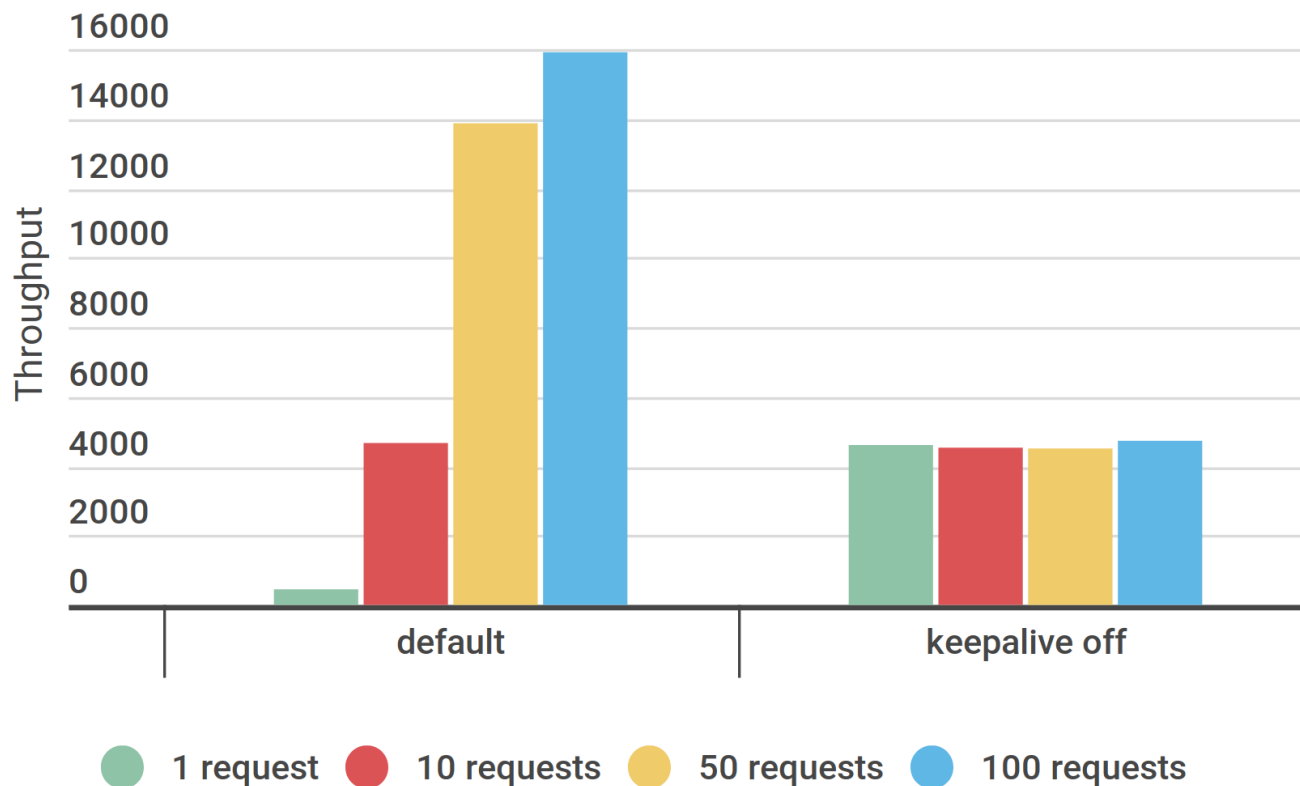
Средняя интенсивность для одного потока

17



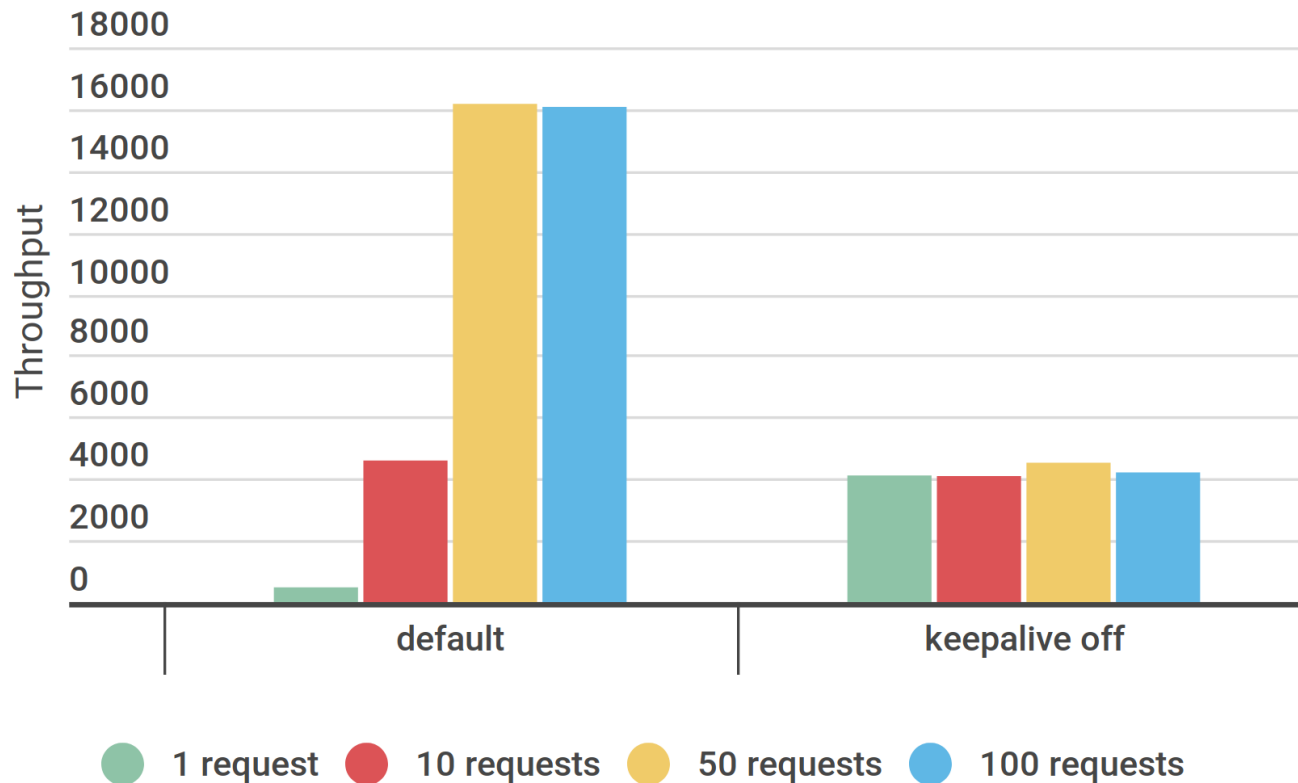
Средняя интенсивность для двух потоков

18



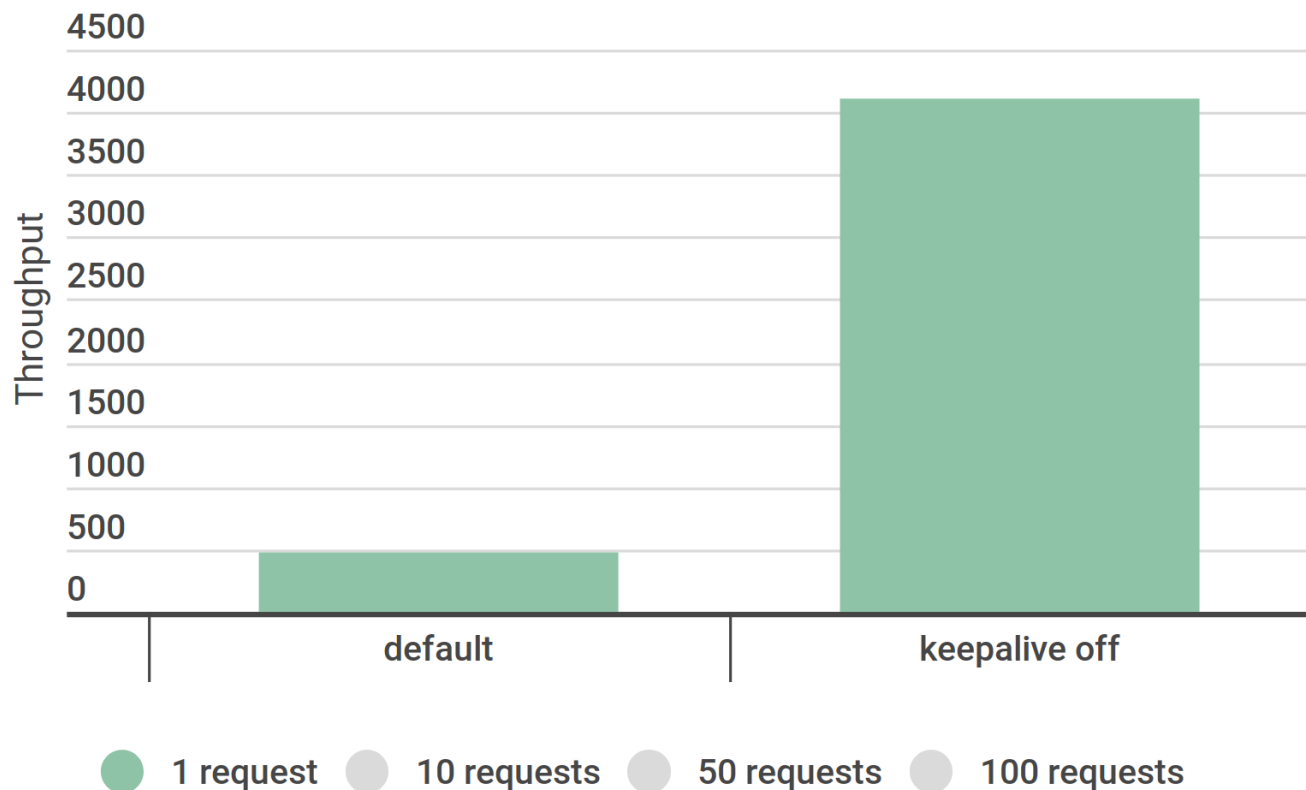
Средняя интенсивность для трёх потоков

19



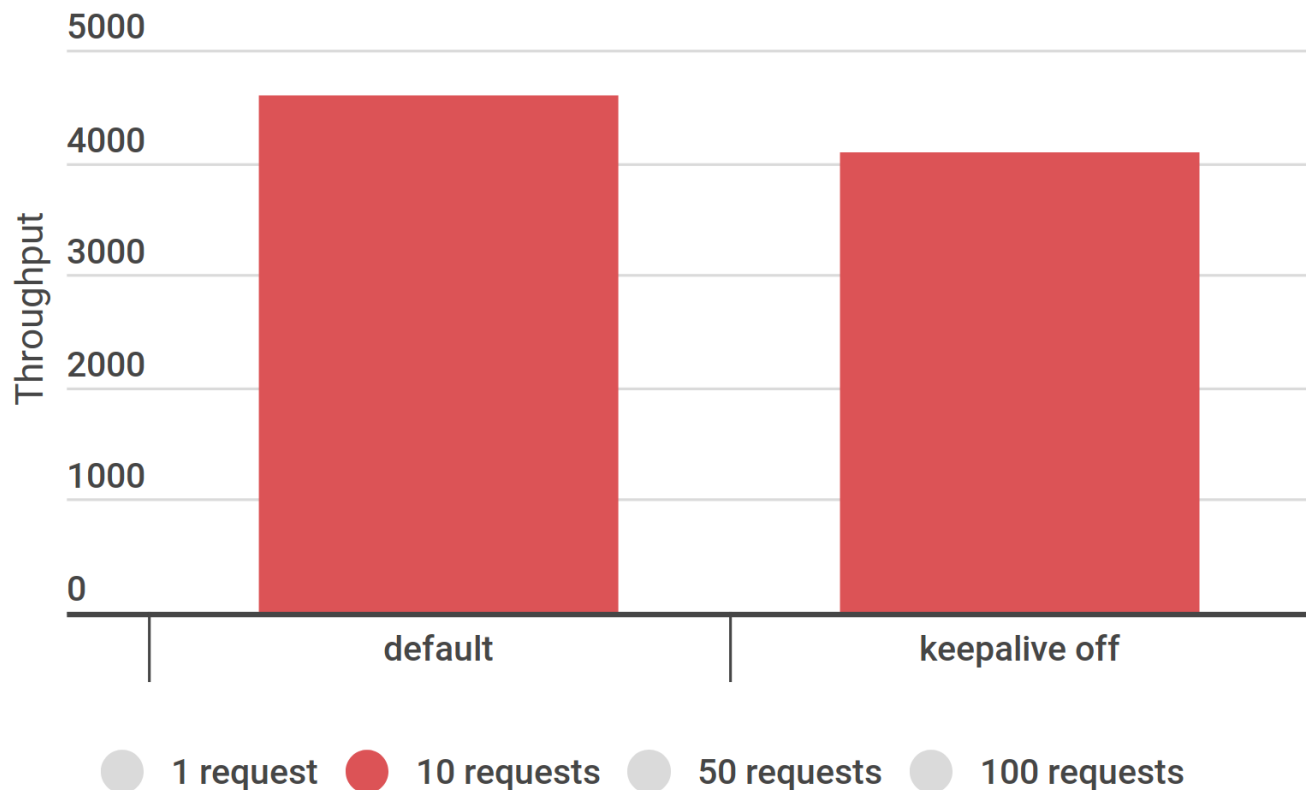
На коротком сценарии лучше без Keep-Alive

20



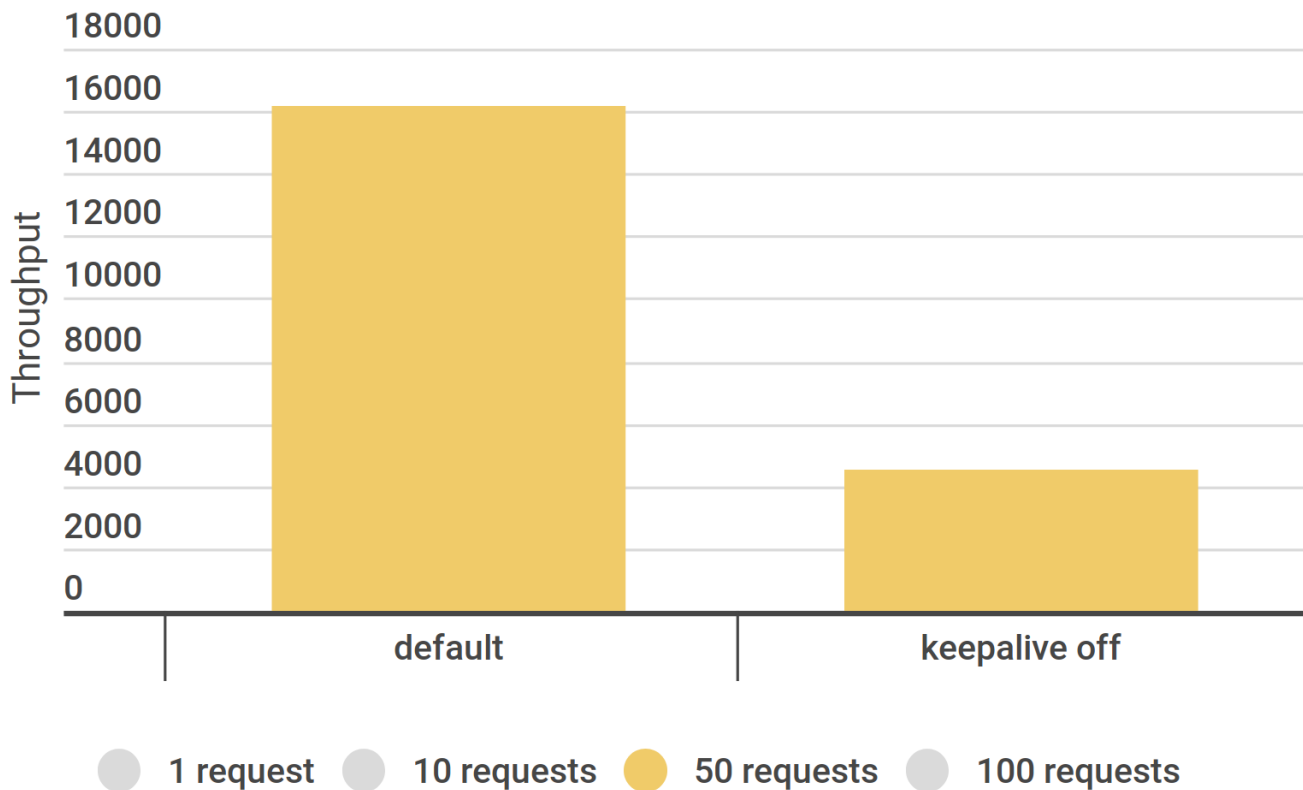
На среднем сценарии — влияние Keep-Alive небольшое

21



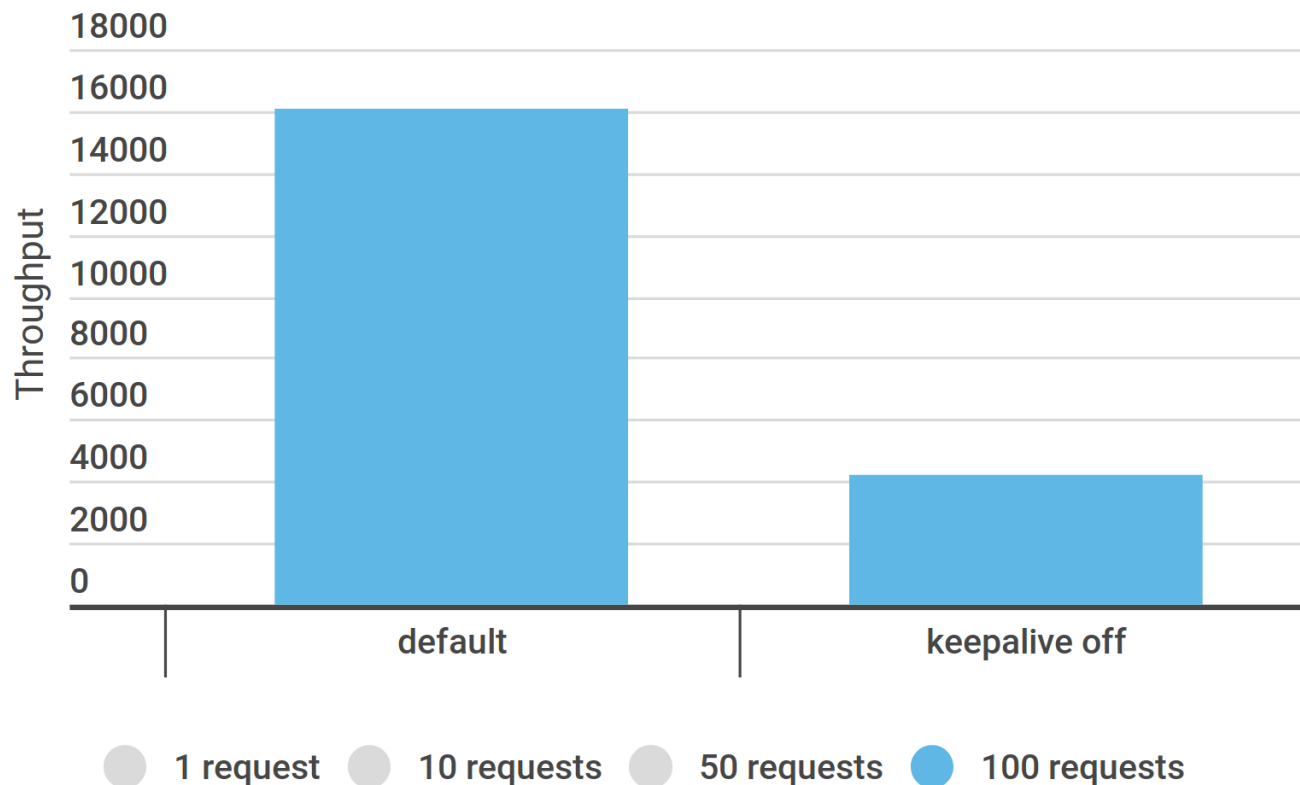
На больших сценариях — Keep-Alive ускоряет работу

22



На больших сценариях — Keep-Alive ускоряет работу

23



Очевидно, что

Keep-Alive полезен, не нужно отключать его.

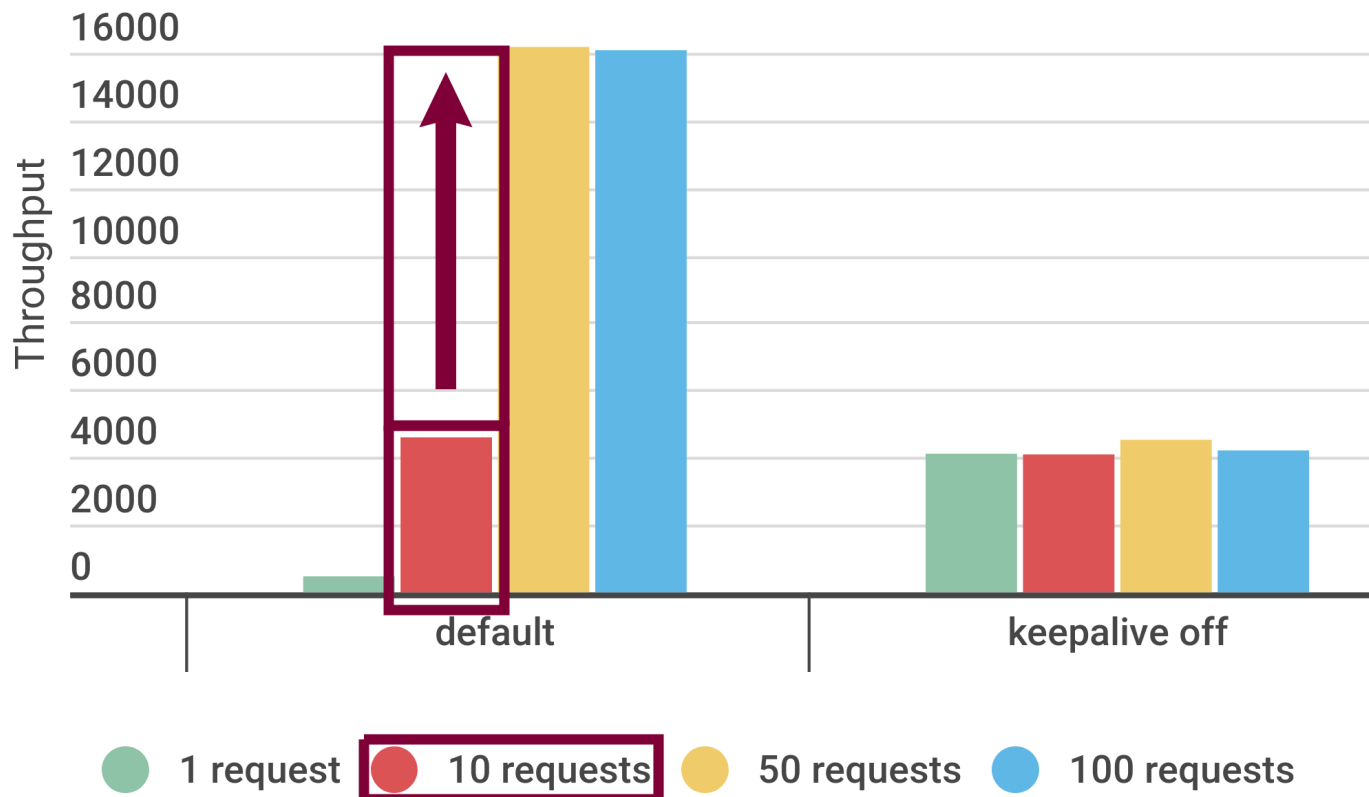
Лучший результат —

с большим сценарием (**50+**) и **Keep-Alive**.

Не делать benchmark на **Apache.JMeter**
из одного **HTTP Request** в сценарии
при настройках по умолчанию.

Сценарий на 10 запросов самый частый







26



Ускорим тест с 10 запросами в сценарии
на 3 потока/пользователя в катушке

с 4k req/s до 16k req/s
и больше.

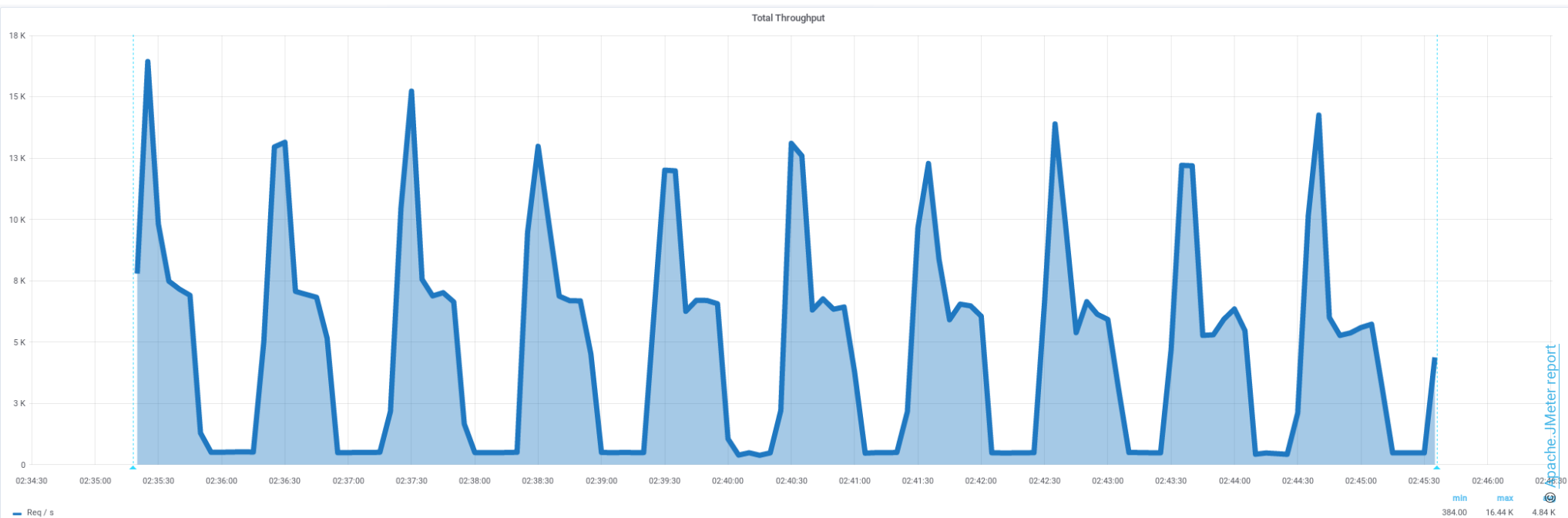
Понадобится

-  telegraf, influxdb, grafana
-  netstat, bash
-  /proc/sys/net/*
-  документация на ядро Linux
-  SJK, Java Flight Recorder
-  настройки Apache.JMeter

Средняя интенсивность 4855/s,

должна быть стабильная нагрузка,

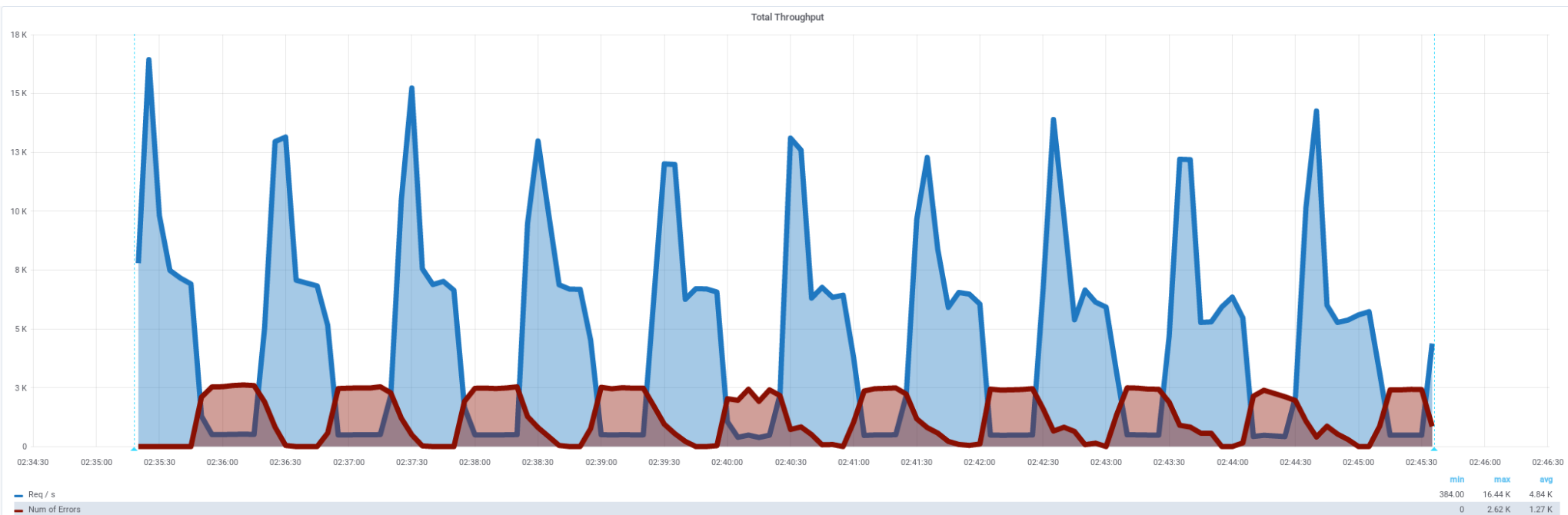
а видим всплески от 500/s до 16000/s



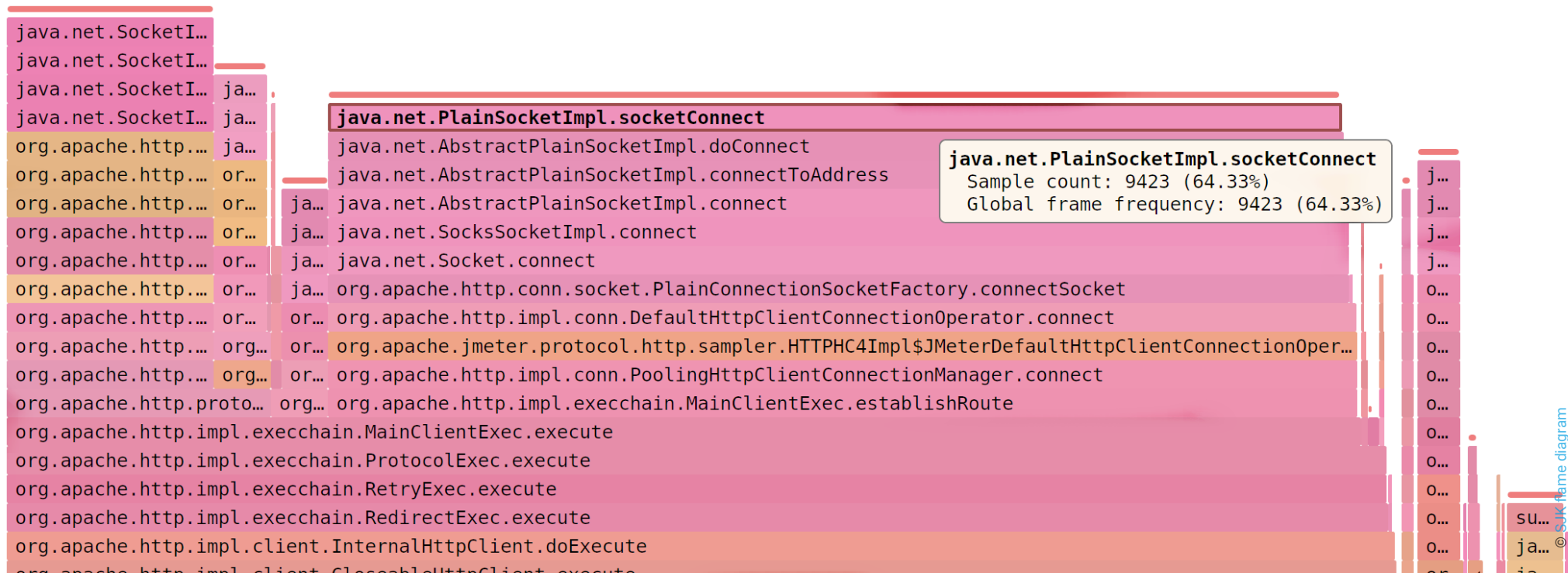
Есть ошибки 5.24%

30

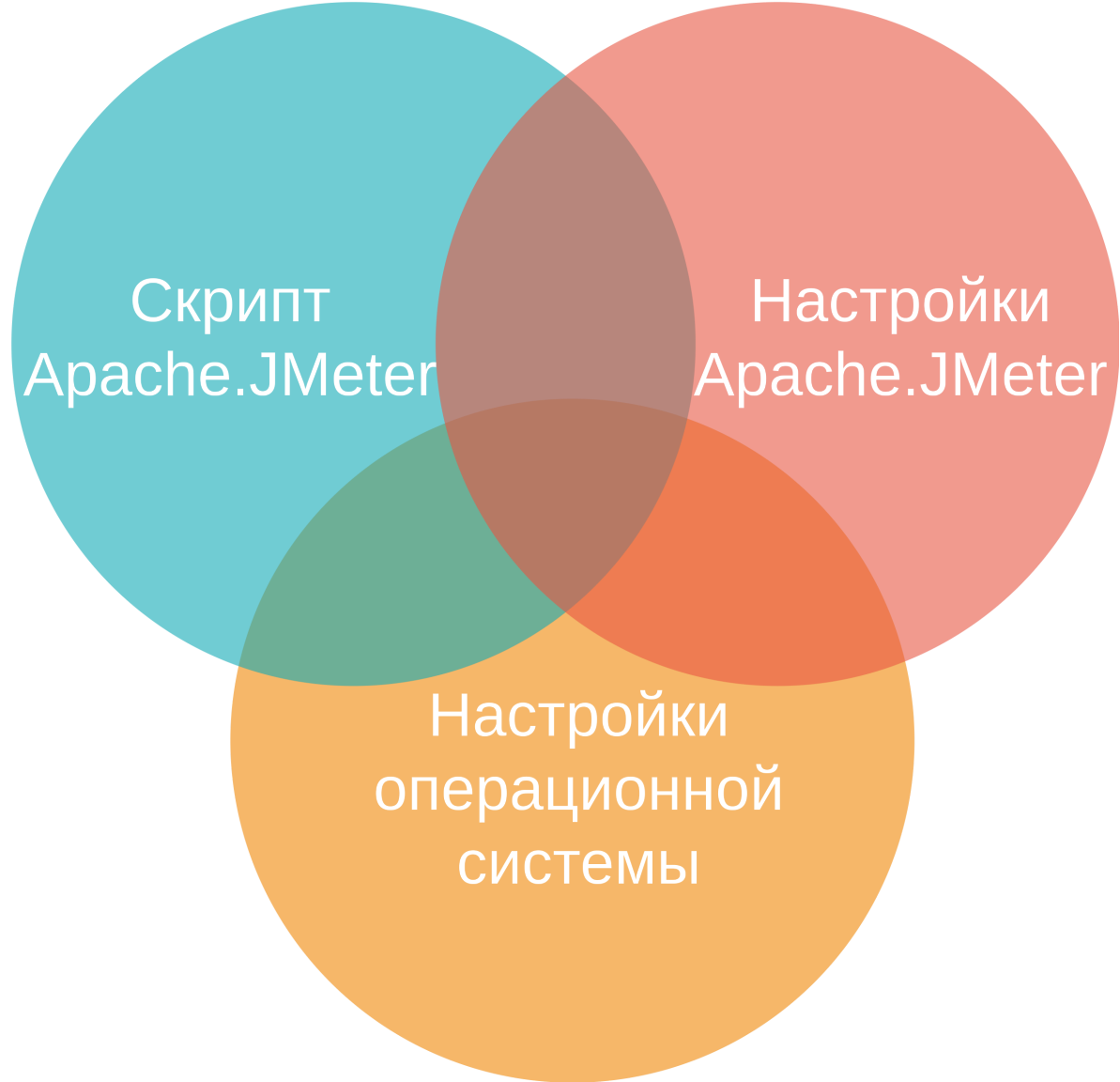
Non HTTP response code: java.net.NoRouteToHostException/Non HTTP
response message: Невозможно назначить запрошенный адрес
(Address not available)

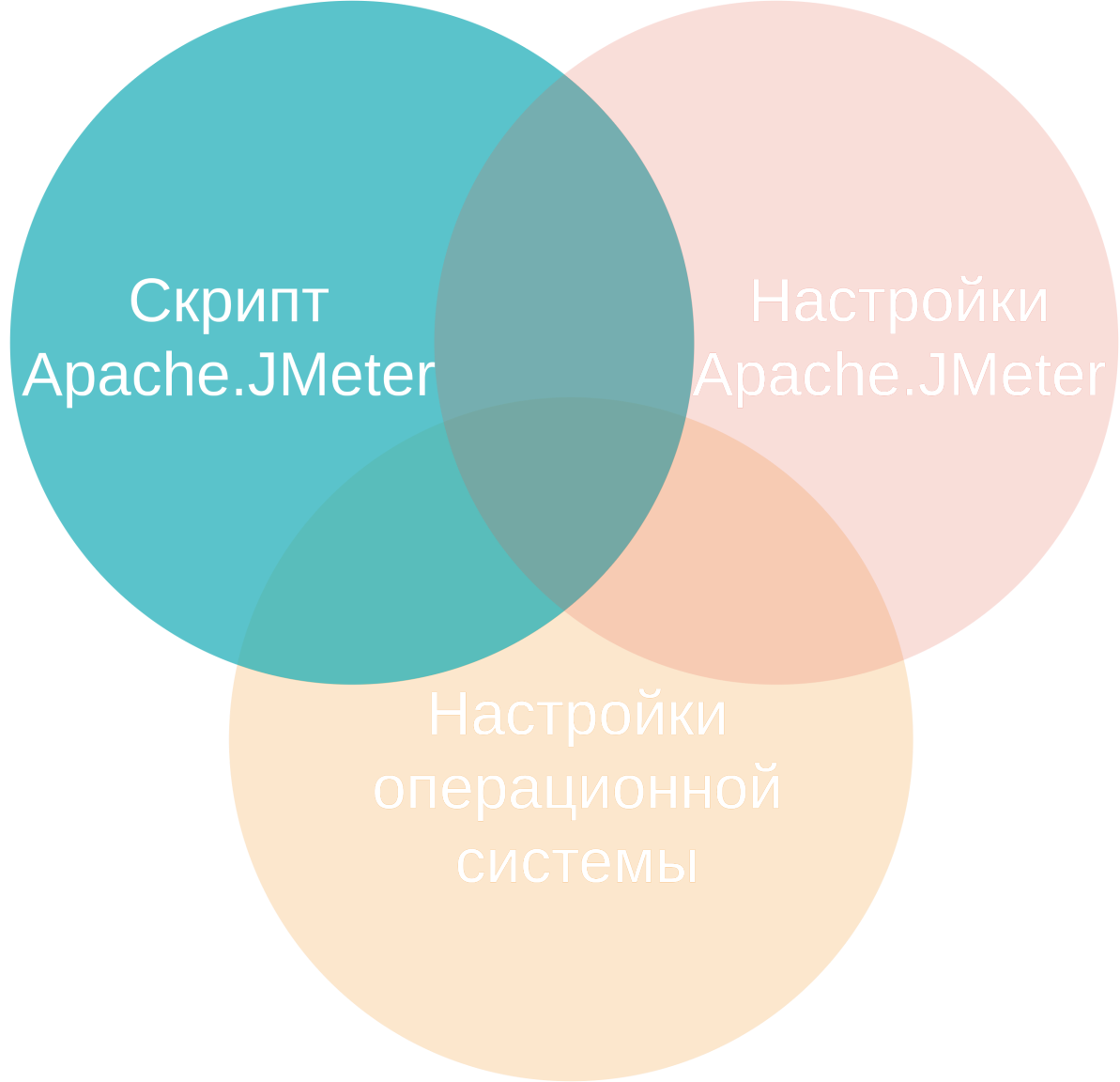


Apache.JMeter ждёт соединения









Скрипт JMeter: увеличить количество запросов (RequestCount) до 50

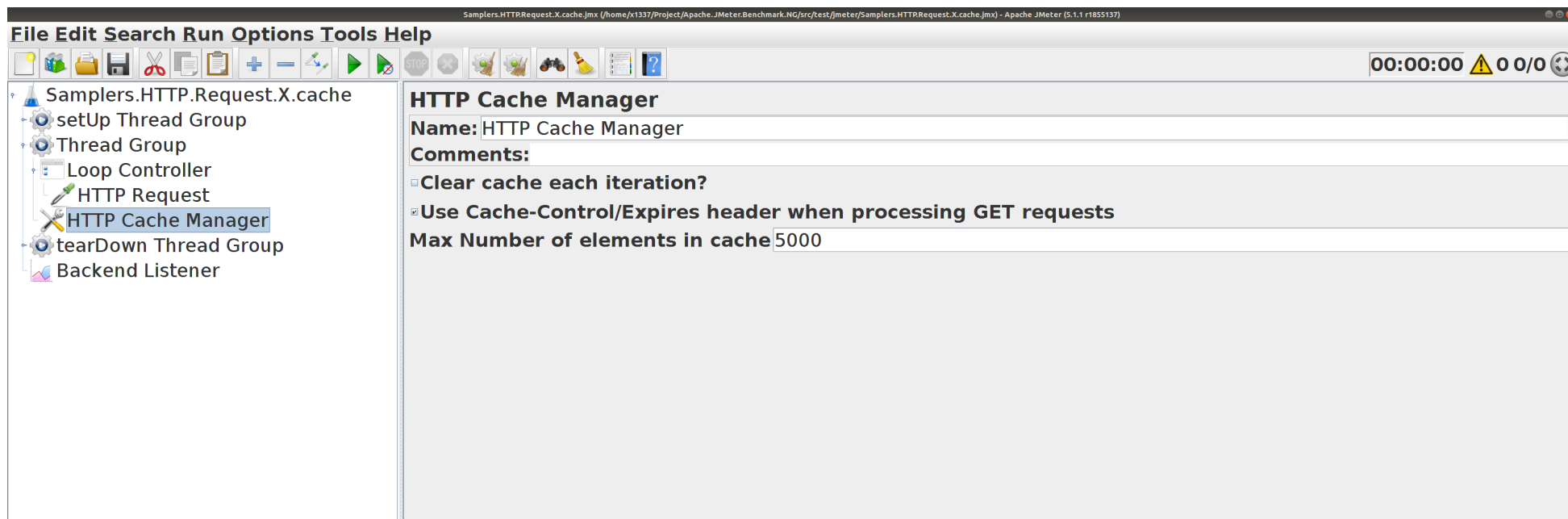
Интенсивность 16500/s (x 3,3) и без ошибок

Нужно удлинить сценарий — запрещённый вариант

Скрипт JMeter: добавить HTTP Cache Manager

36

Может ускорит?



Скрипт JMeter: добавить HTTP Cache Manager

37

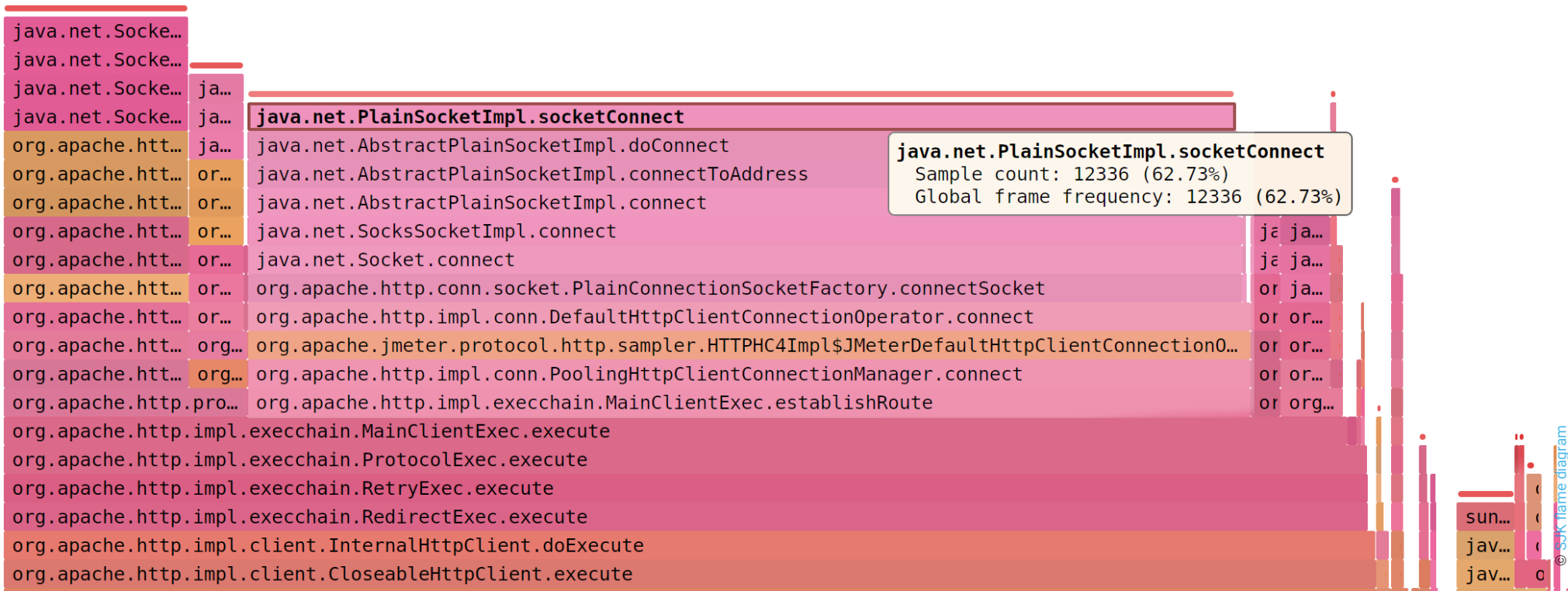
Интенсивность 4800/с и 4.9% ошибок (не повлиял)



SJK: socketConnect (62,73%)

38

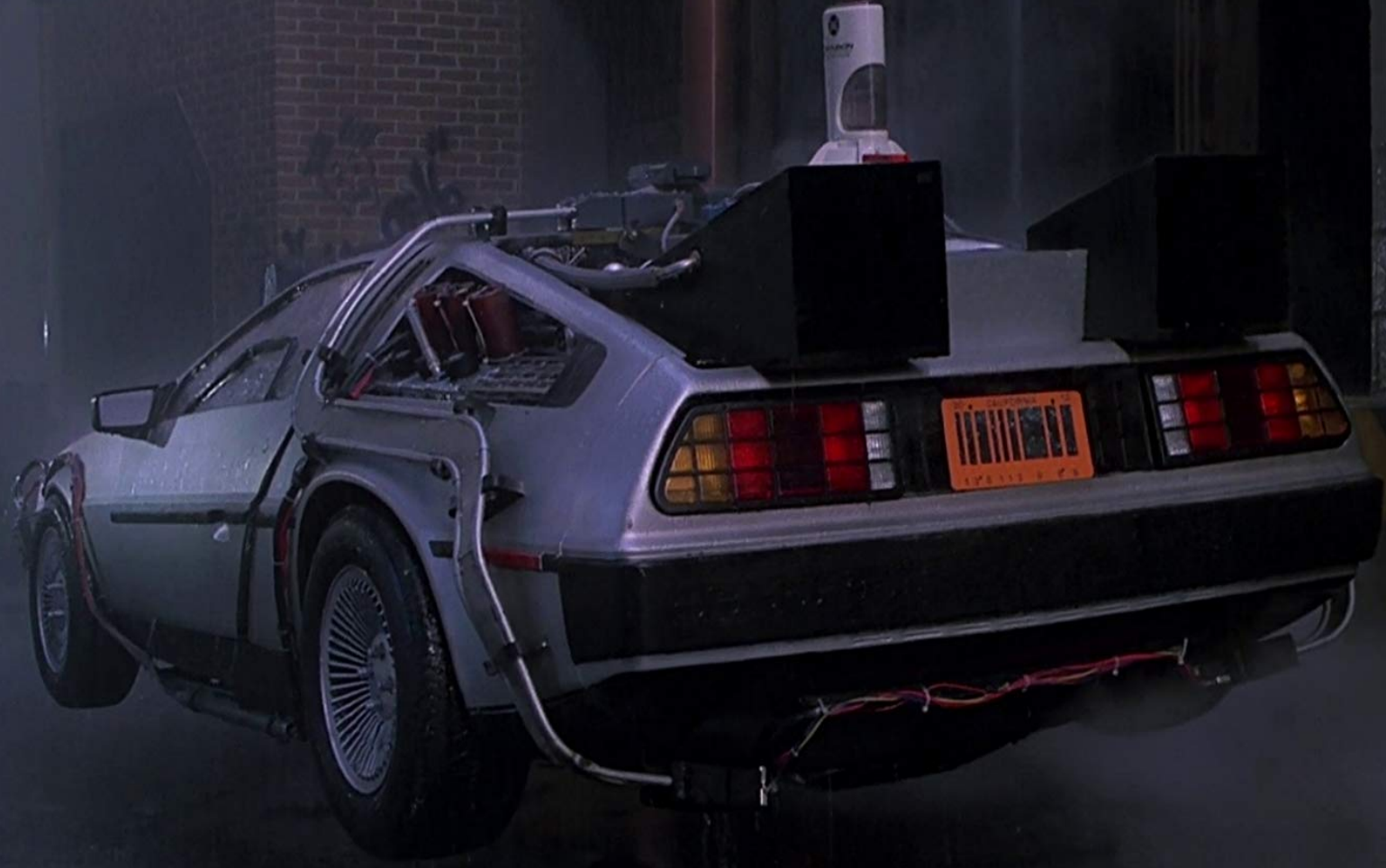
Apache.JMeter ждёт соединения

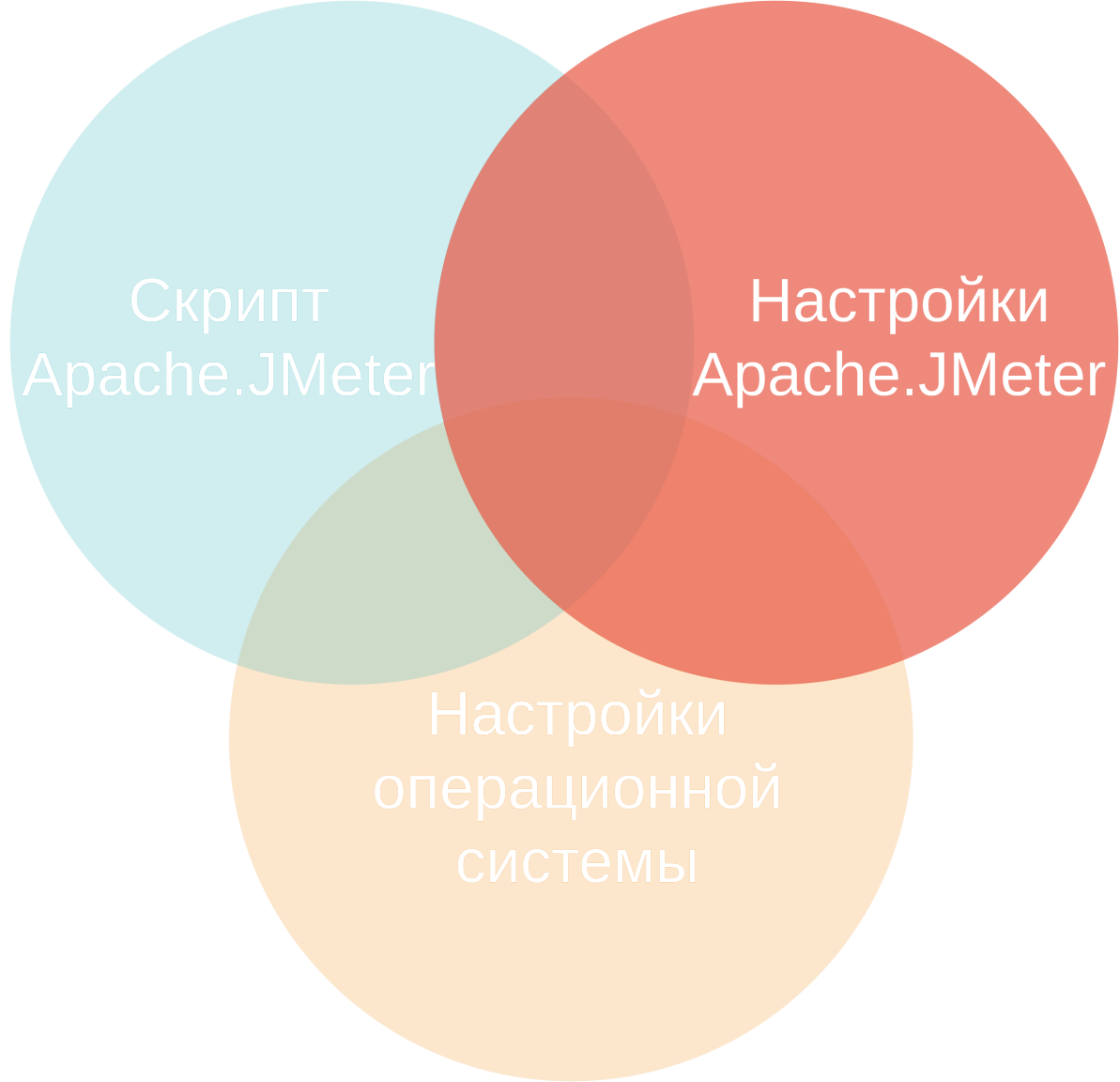


Модификацией скрипта интенсивность не поднять.

Решили оставить **10** запросов в итерации.

А HTTP Cache Manager не ускорил
получение маленького ответа.





Настройки JMeter: jmeter.httpsampler=Java

42

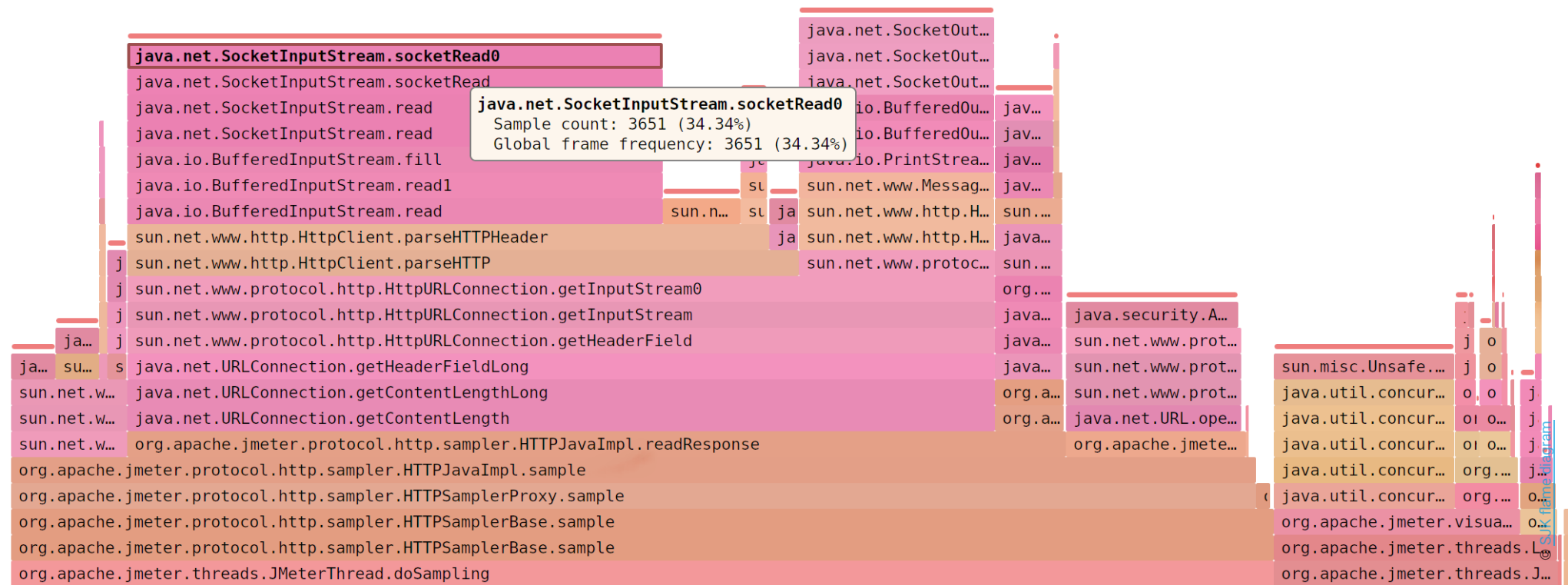
Интенсивность 16300/s (x 3,3) и без ошибок

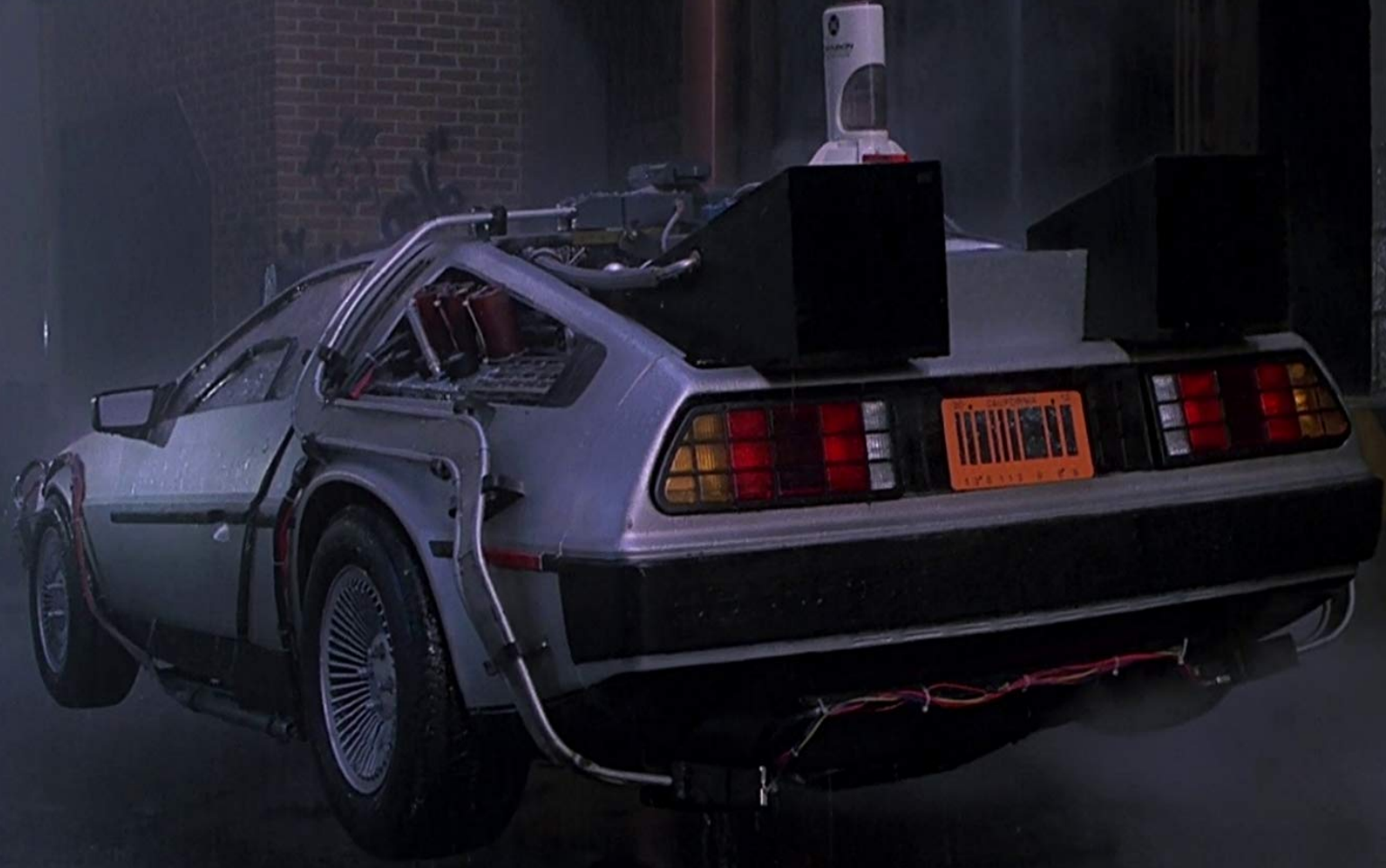


SJK: socketRead (34,34%)

43

Apache.JMeter читает заголовок ответа





httpClient.reset_state_on_thread_group_iteration (false)

```
01. #-----  
02. # SSL configuration  
03. #-----  
04. httpClient.reset_state_on_thread_group_iteration=false
```

Настройки JMeter: не сбрасывать состояние

46

Интенсивность 19240/s (x 3,9) и без ошибок



Настройки JMeter

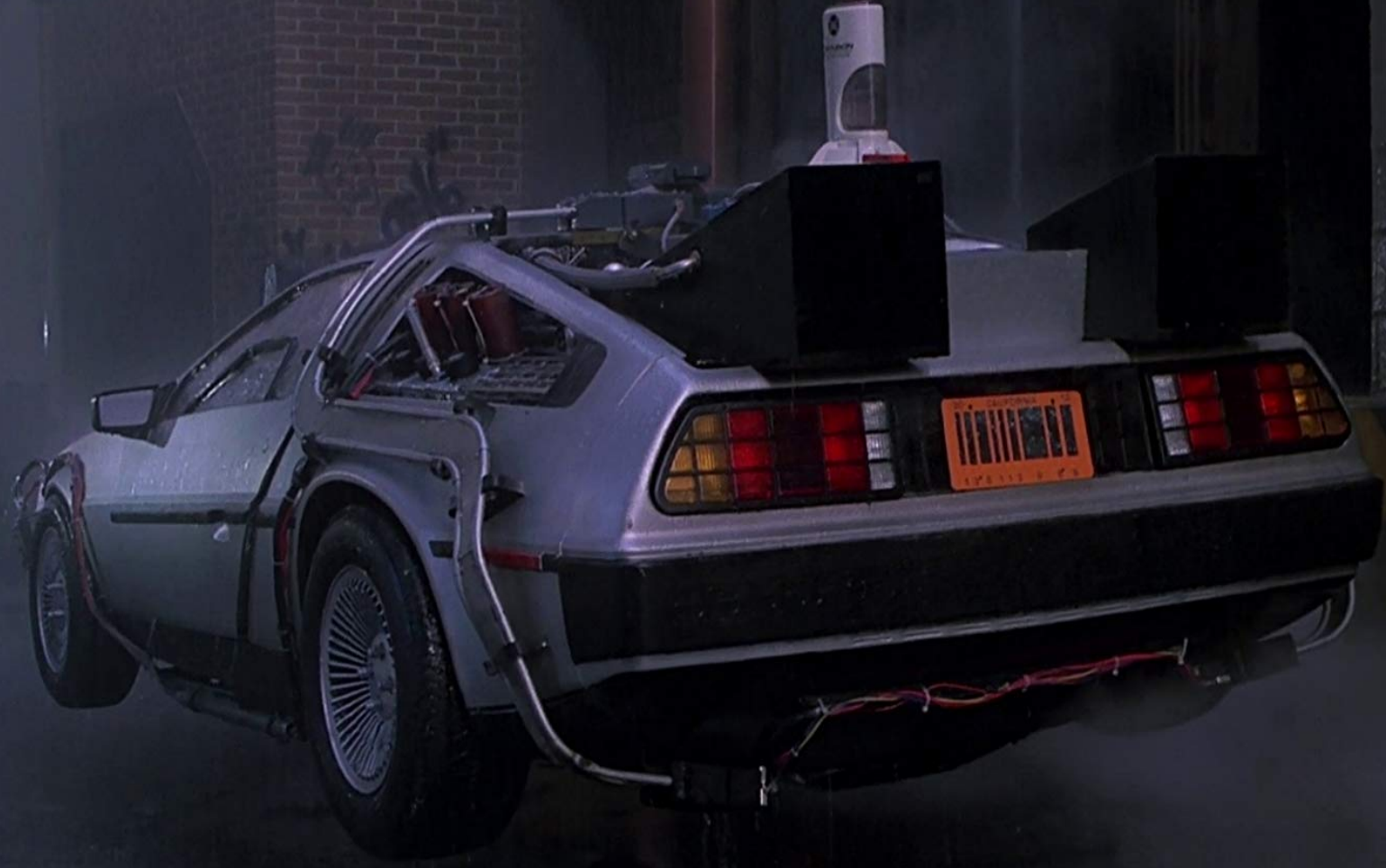
Если интенсивность скачет (500/s - 16000/s), есть ошибки (Address not available) и JMeter занят socketConnect (> 50%)

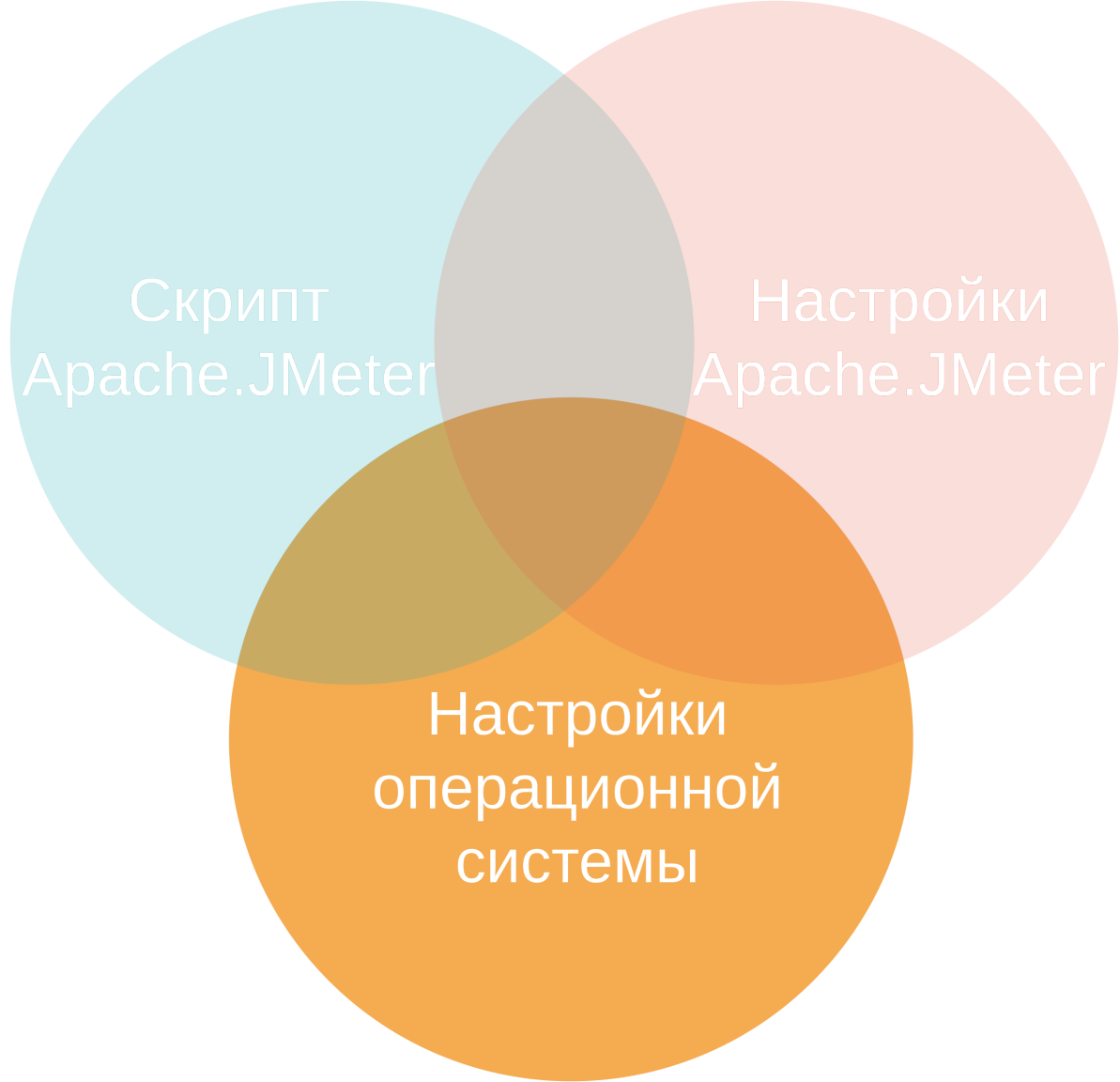
Можно перейти с HTTPClient4 на Java-клиент (x 3,3)

```
jmeter.httpsampler=Java
```

Или кешировать соединение между итерациями (x 3,9)

```
httpClient.reset_state_on_thread_group_iteration=false
```

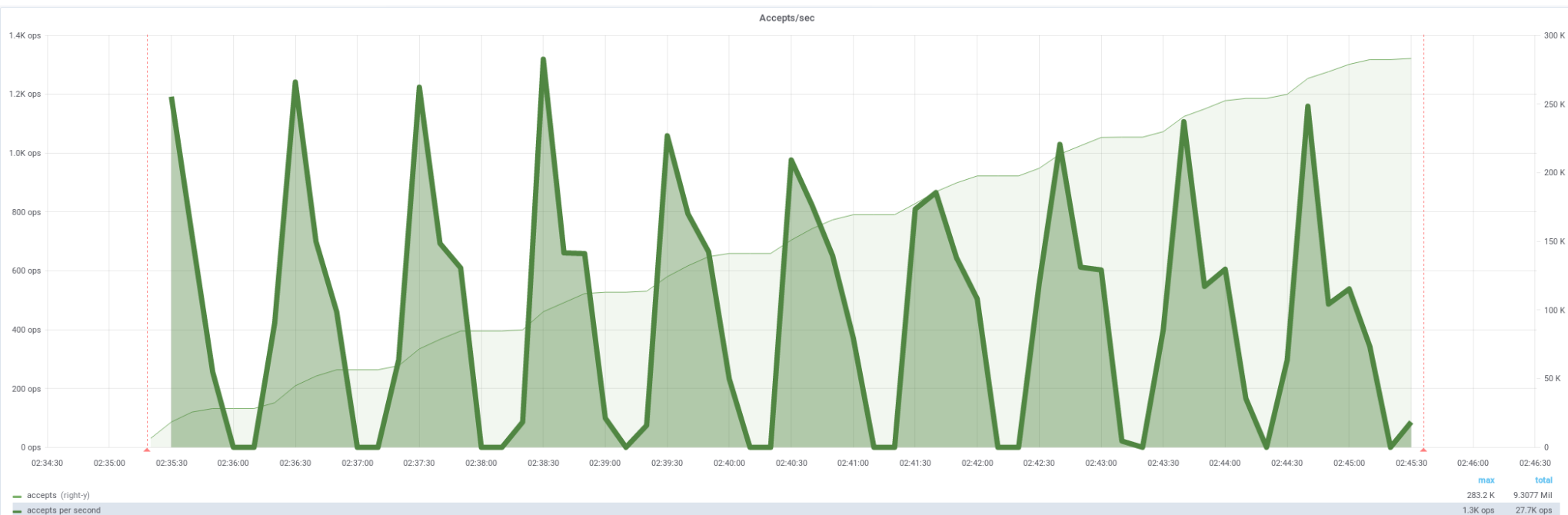





Подключения к NGinx идут волнами

50

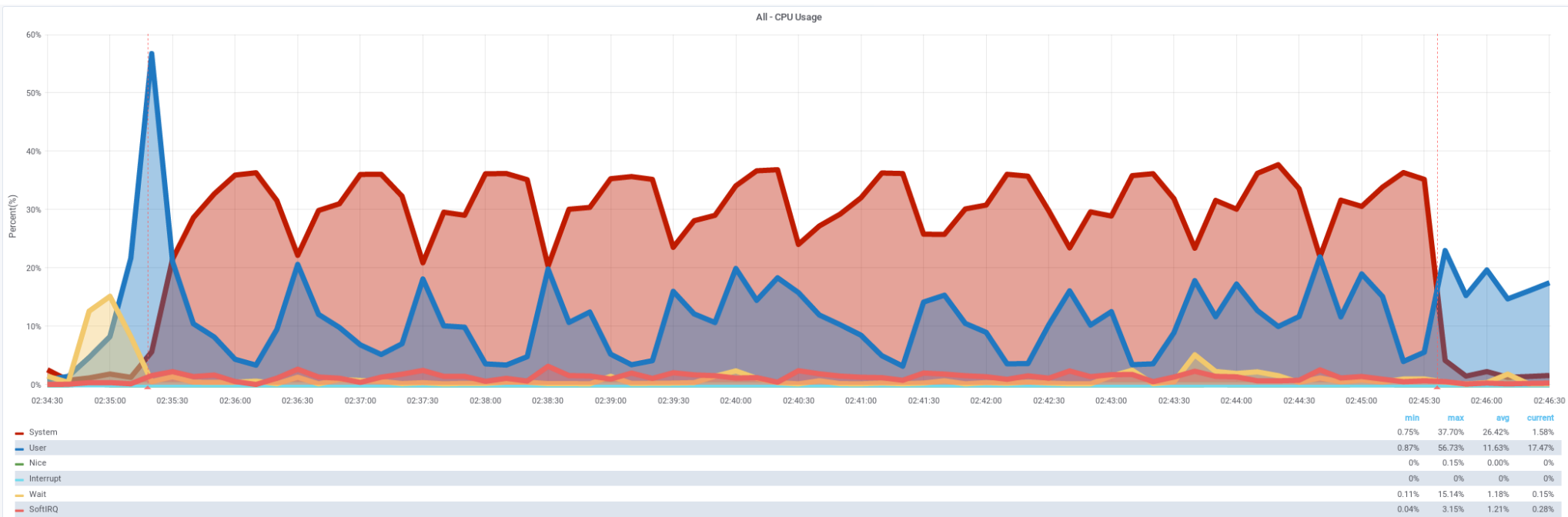
от 0/s до 1200/s (x 0.1 от интенсивности)



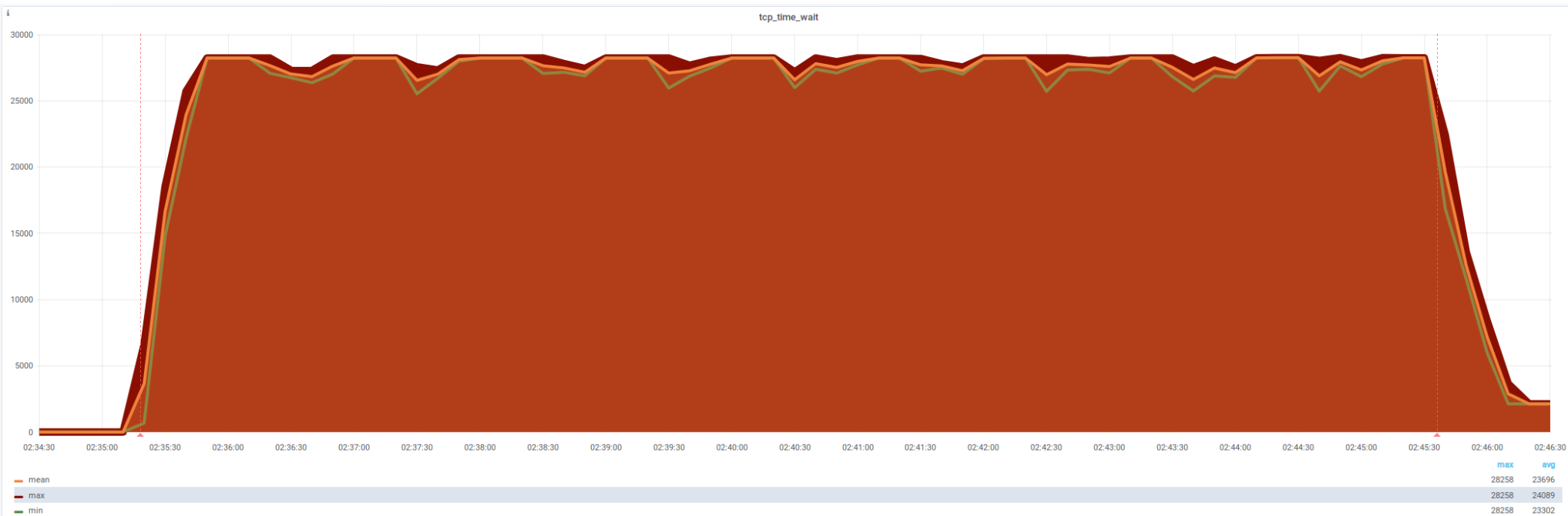
CPU: преимущественно системное время

51

%System в 3,5 раза выше, чем %User



Открывается слишком много соединений, лимит портов



Закрытые соединения с nginx (localhost:5555)

53

Одно **ESTABLISHED** и 27500 **TIME_WAIT**

```
01. netstat | grep -oP \
```

```
02. "^\w+\s+\d+\s+\d+\s+(\w+:\w+)\s+(localhost:5555)\s.*"
```

```
01. tcp    0      0  localhost:*****    localhost:5555    ESTABLISHED
```

```
02. tcp6   0      0  localhost:*****    localhost:5555    TIME_WAIT
```

```
03. tcp6   0      0  localhost:*****    localhost:5555    TIME_WAIT
```

```
04. ...
```

Больше диапазон — больше соединений

Изначально диапазон на 28231 портов:

```
01. cat /proc/sys/net/ipv4/ip_local_port_range
```

```
02. 32768 60999
```

Больше диапазон — больше соединений

Расширим с 28231 до 59974 (x 2,1):

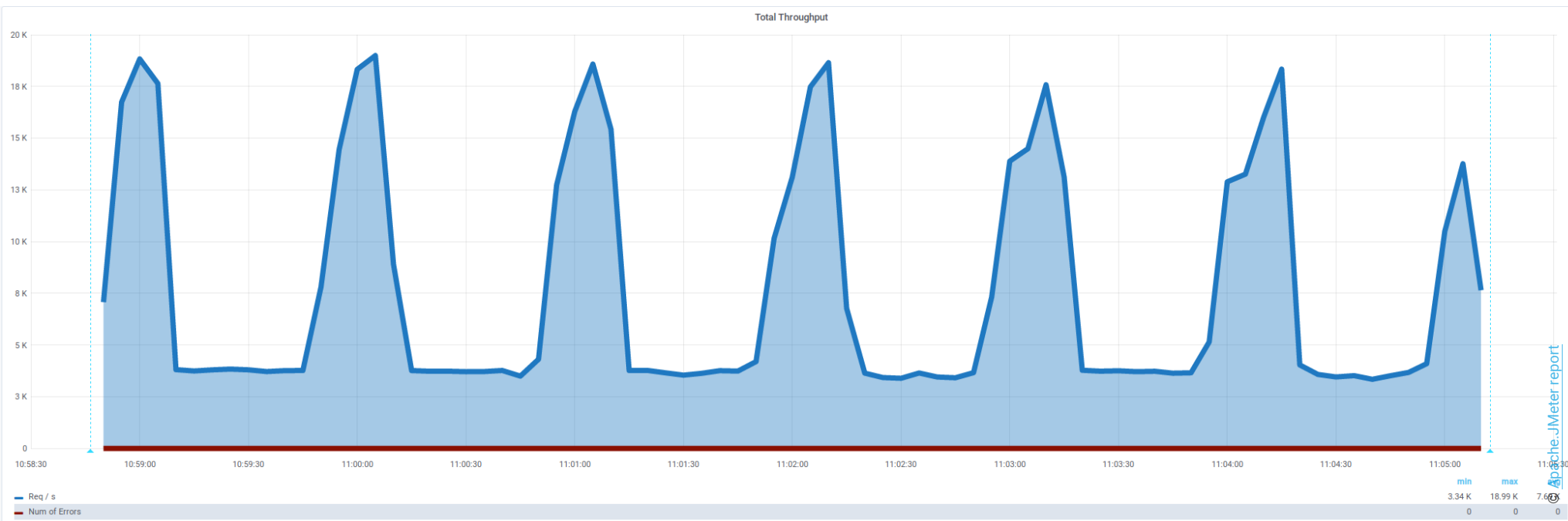
```
echo 1025 60999 > /proc/sys/net/ipv4/ip_local_port_range
```

[Проблемы с очередью TIME_WAIT \(@blog.kireev.pro\)](http://blog.kireev.pro)

Средняя интенсивность 7772/s (x 1,6)

56

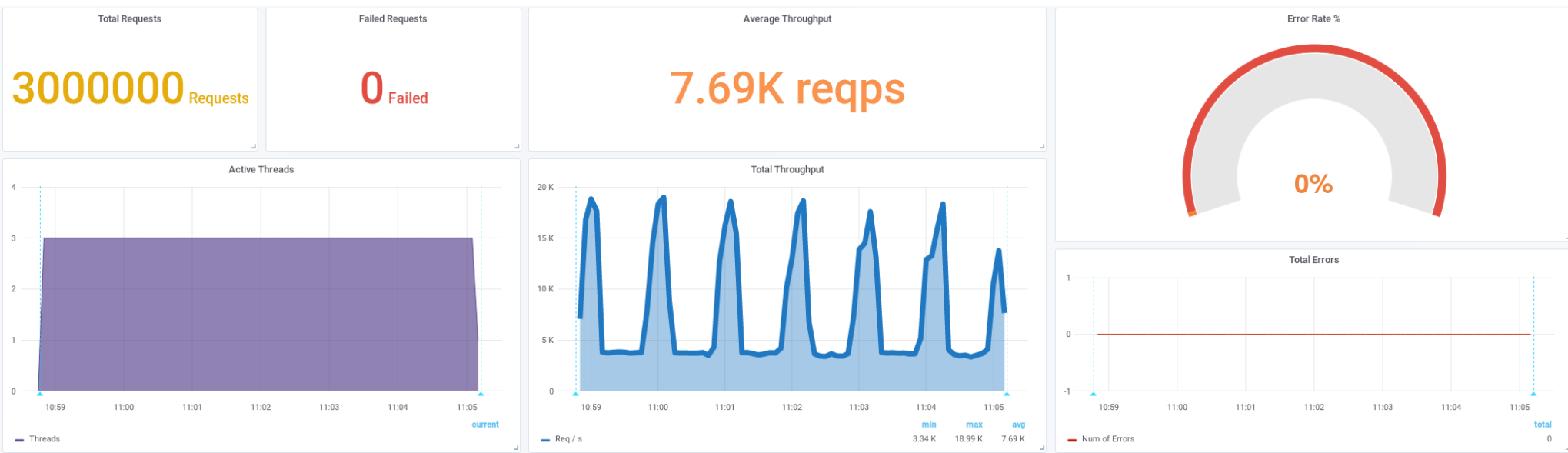
нестабильная: 3500/s ... 19000/s



Теперь без ошибок

57

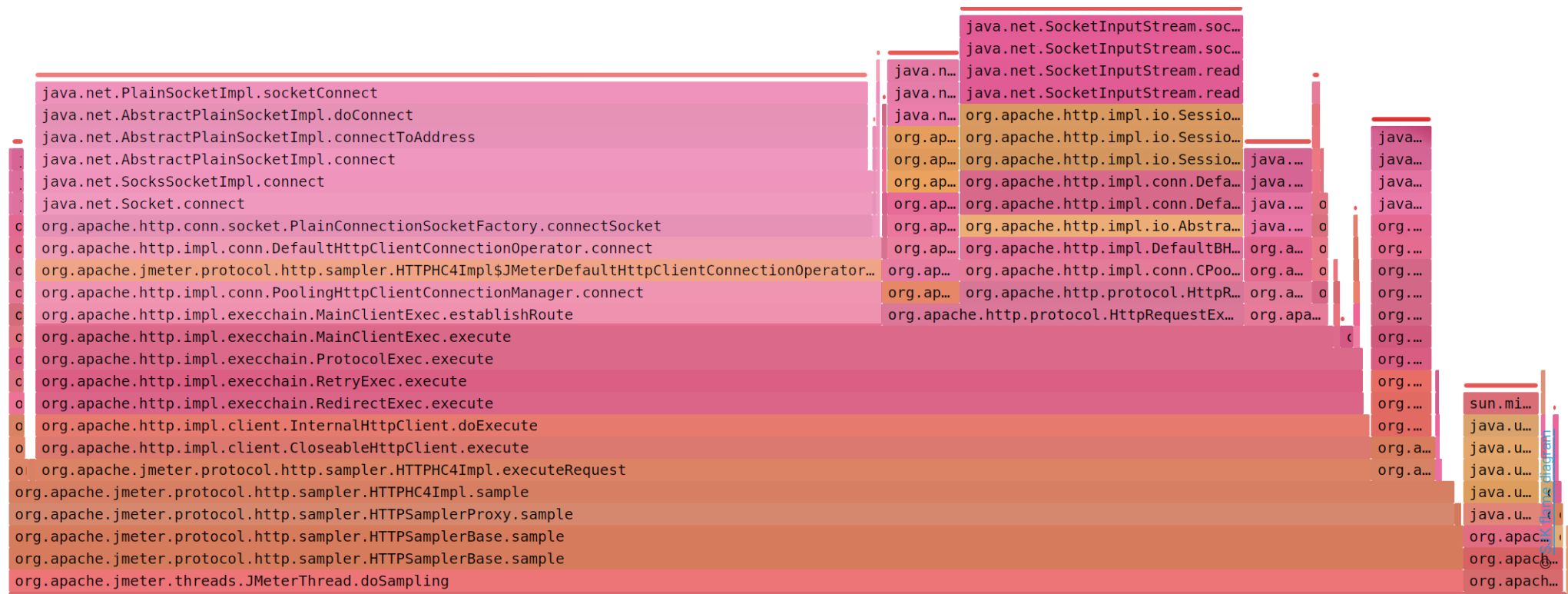
ошибки ушли



SJK: socketConnect (53,28%)

58

Apache.JMeter ждёт соединения



Ускорение в **1,6** раза — с 4800/s до 7700/s

и избавление от ошибок **Address not available**

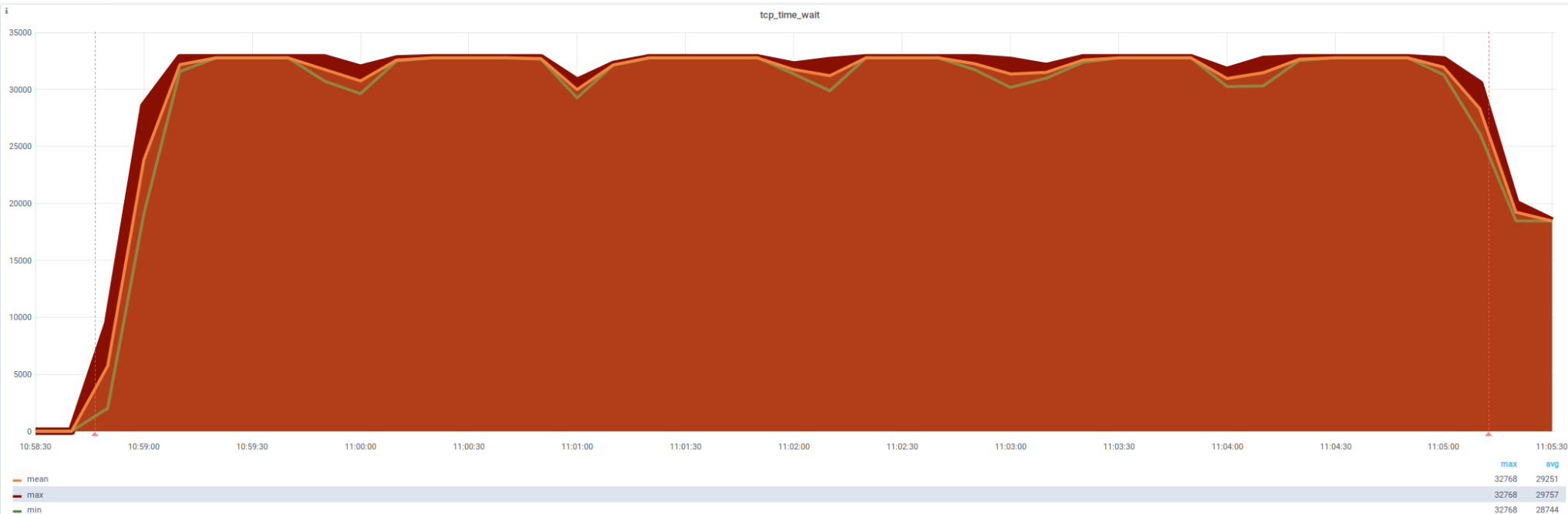
при расширении диапазона портов

net.ipv4.ip_local_port_range (1025 60999)

netstat: tcp_time_wait (32 768 , не 59 975)

60

НОВЫЙ ЛИМИТ — СИМПТОМЫ ПОЛЕЧЕНЫ, ПРОБЛЕМА ОСТАЛАСЬ



Расширим размер очереди TIME_WAIT

Значение в Linux можно посмотреть с помощью команды **cat** так:

01. `cat /proc/sys/net/ipv4/tcp_max_tw_buckets`

02. `32768`

Расширим размер очереди TIME_WAIT (временно)

Увеличим в два раза до значения **65536**. Временную настройку можно выполнить так:

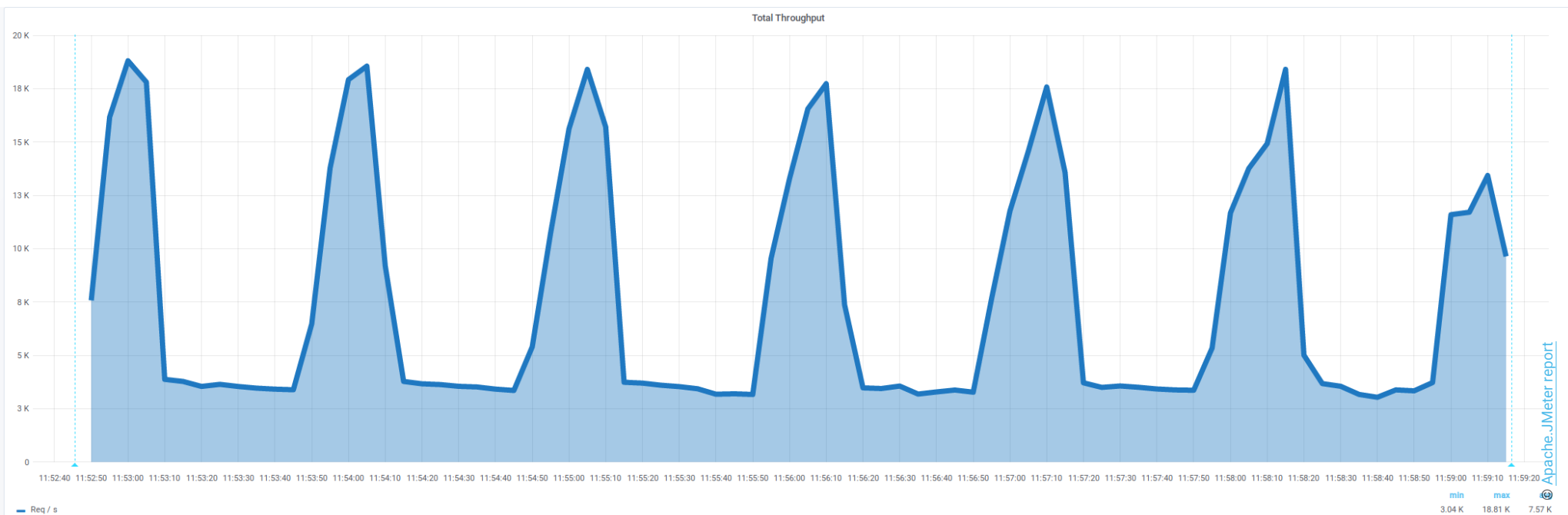
```
01. echo 65536 > /proc/sys/net/ipv4/tcp_max_tw_buckets
```

```
02.
```

Средняя интенсивность 7700/s (та же)

63

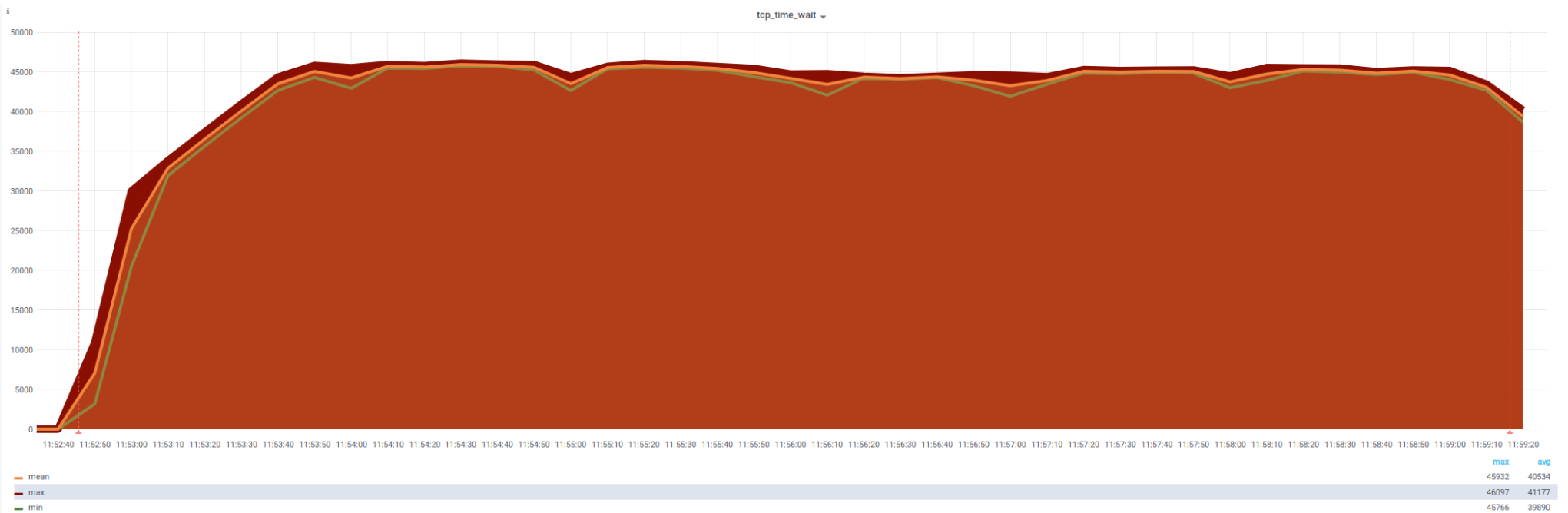
нестабильная: 3500/s ... 19000/s



netstat: tcp_time_wait (40 000 - 46 000)

64

лимита в 32768 больше нет — но не помогло



Увеличение размера очереди TIME_WAIT

net.ipv4.tcp_max_tw_buckets (65536)

не дало положительного эффекта для скорости,
но нет больше полки для netstat: tcp_time_wait

Method	Count	Frequency (%)
java.net.SocketInputStr...	11295	54.85%
java.net.SocketInputStr...	11295	54.85%
java.net.SocketInputStr...	11295	54.85%
java.net.SocketInputStr...	11295	54.85%
org.apache.http.impl.io...	11295	54.85%
org.apache.http.impl.io...	11295	54.85%
org.apache.http.impl.io...	11295	54.85%
org.apache.http.impl.co...	11295	54.85%
org.apache.http.impl.co...	11295	54.85%
org.apache.http.impl.io...	11295	54.85%
org.apache.http.impl.De...	11295	54.85%
org.apache.http.impl.co...	11295	54.85%
org.apache.http.protoco...	11295	54.85%
org.apache.http.protocol.HttpR...	11295	54.85%
org.apache.http.impl.execchain.MainClientExec.execute	11295	54.85%
org.apache.http.impl.execchain.ProtocolExec.execute	11295	54.85%
org.apache.http.impl.execchain.RetryExec.execute	11295	54.85%
org.apache.http.impl.execchain.RedirectExec.execute	11295	54.85%
org.apache.http.impl.client.InternalHttpClient.doExecute	11295	54.85%
org.apache.http.impl.client.CloseableHttpClient.execute	11295	54.85%

Разрешим переиспользовать TIME_WAIT

По умолчанию для исходящих подключений нельзя использовать TIME_WAIT-соединения.

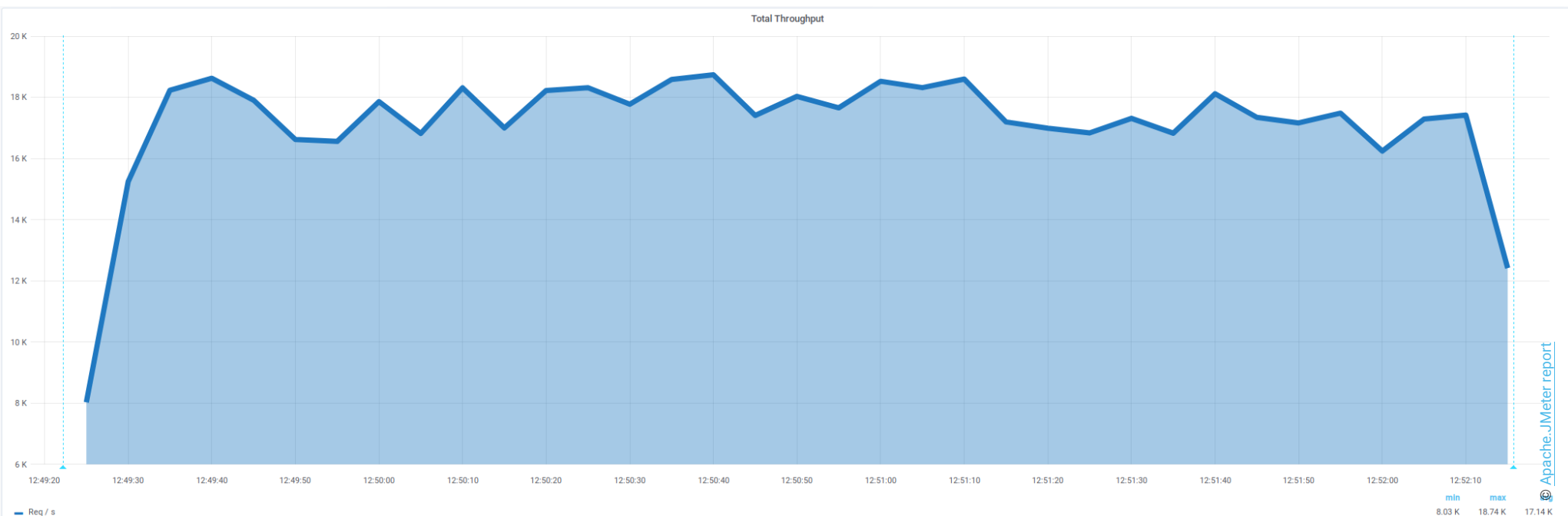
Разрешим:

```
echo 1 > /proc/sys/net/ipv4/tcp_tw_reuse
```

Средняя интенсивность 17300/s (x 3,6)

68

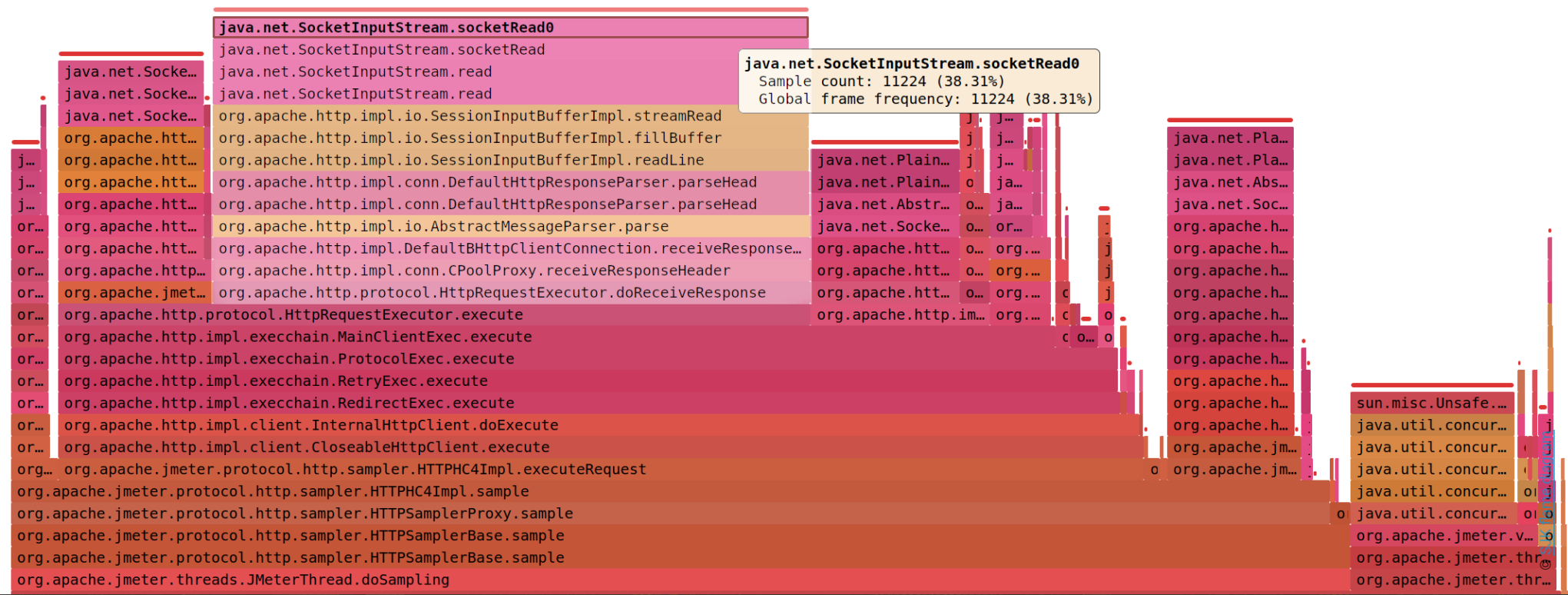
более стабильная: 16500/s ... 19000/s



SJK: socketRead (38,31%)

69

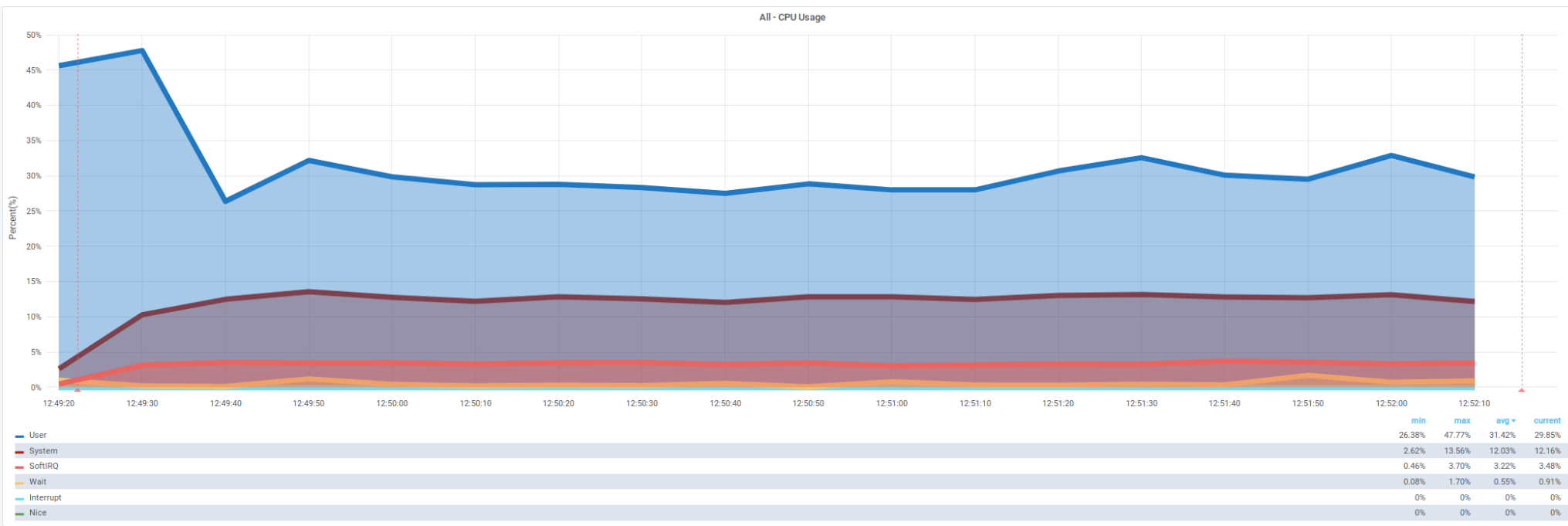
Apache.JMeter читает заголовок ответа



CPU: больше не упирается в систему

70

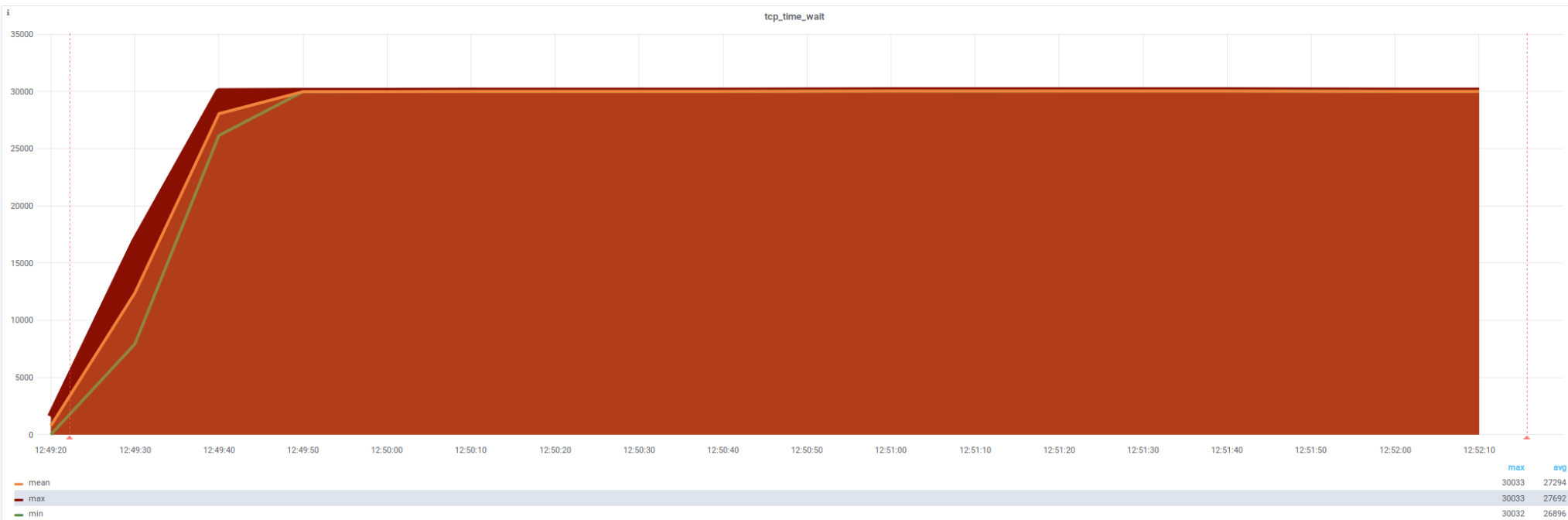
%User в 2 раза выше, чем %System



netstat: tcp_time_wait (30 000 , снижение)

71

подозрительно ровная полка



Gatling documentation / 3.1 / General / Operations

- Java Version
- IPv4 vs IPv6
- OS Tuning
 - Open Files Limit
 - Kernel and Network Tuning

Рекомендации для Gatling

73

```
01. -Djava.net.preferIPv4Stack=true
02. -Djava.net.preferIPv6Addresses=false
03. ulimit -n 65536
04. sudo sysctl -w net.ipv4.ip_local_port_range="1025 65535"
05. echo 300000 | sudo tee /proc/sys/fs/nr_open
06. echo 300000 | sudo tee /proc/sys/fs/file-max
07. net.ipv4.tcp_max_syn_backlog = 40000
08. net.core.somaxconn = 40000
09. net.core.wmem_default = 8388608
10. net.core.rmem_default = 8388608
11. net.ipv4.tcp_sack = 1
```

Рекомендации для Gatling

74

```
01.net.ipv4.tcp_window_scaling = 1
02.net.ipv4.tcp_fin_timeout = 15
03.net.ipv4.tcp_keepalive_intvl = 30
04.net.ipv4.tcp_tw_reuse = 1
05.net.ipv4.tcp_moderate_rcvbuf = 1
06.net.core.rmem_max = 134217728
07.net.core.wmem_max = 134217728
08.net.ipv4.tcp_mem = 134217728 134217728 134217728
09.net.ipv4.tcp_rmem = 4096 277750 134217728
10.net.ipv4.tcp_wmem = 4096 277750 134217728
11.net.core.netdev_max_backlog = 300000
```

И для Yandex.Tank (bit.ly/tank-tuning)

75

```
01. ulimit -n 30000
02. net.ipv4.tcp_max_tw_buckets = 65536
03. net.ipv4.tcp_tw_recycle = 1
04. net.ipv4.tcp_tw_reuse = 0
05. net.ipv4.tcp_max_syn_backlog = 131072
06. net.ipv4.tcp_syn_retries = 3
07. net.ipv4.tcp_synack_retries = 3
08. net.ipv4.tcp_retries1 = 3
09. net.ipv4.tcp_retries2 = 8
10. net.ipv4.tcp_rmem = 16384 174760 349520
11. net.ipv4.tcp_wmem = 16384 131072 262144
12. net.ipv4.tcp_mem = 262144 524288 1048576
```

Ускорение в **3,6** раза — с 4800/s до 17300/s

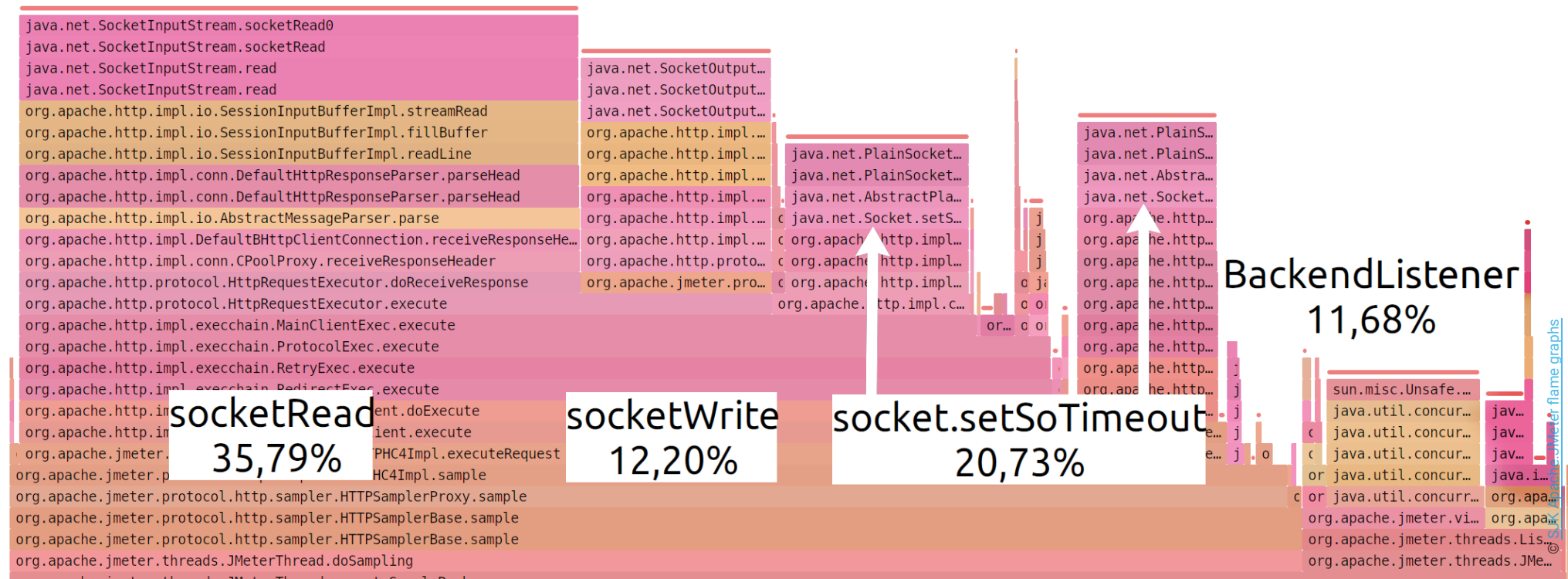
при разрешении переиспользовать TIME_WAIT

net.ipv4.tcp_tw_reuse (1)

Опция	Значение	Ускорение
httpClient.reset_state_on_thread_group_iteration	false	x3.9
jmeter.httpsampler	Java	x3.3
Linux: net.ipv4.tcp_tw_reuse	1	x3.6
Linux: net.ipv4.tcp_max_tw_buckets	65536	
Linux: net.ipv4.ip_local_port_range	1025 60999	x1.6
NGinx: worker_processes	2 (+1)	
Настройки Gatling bit.ly/gatling-tuning и Yandex.Tank bit.ly/tank-tuning		

Если попробовать смешать?

Apache.JMeter читает, пишет, собирает статистику





HTTP Request

Скачивание файлов

Последовательное скачивание файла,
используя Thread Group на один поток и 200
итераций, имея 4 ГБайт Heap Size

Создадим файл на 1 ГБайт и скачаем его 200 раз

```
01. dd if=/dev/urandom of=/tmp/data/1g.img \
02.     bs=1 count=0 seek=1G
```

200 ГБайт скачались достаточно быстро

82

Параметр

Значение

Среднее время

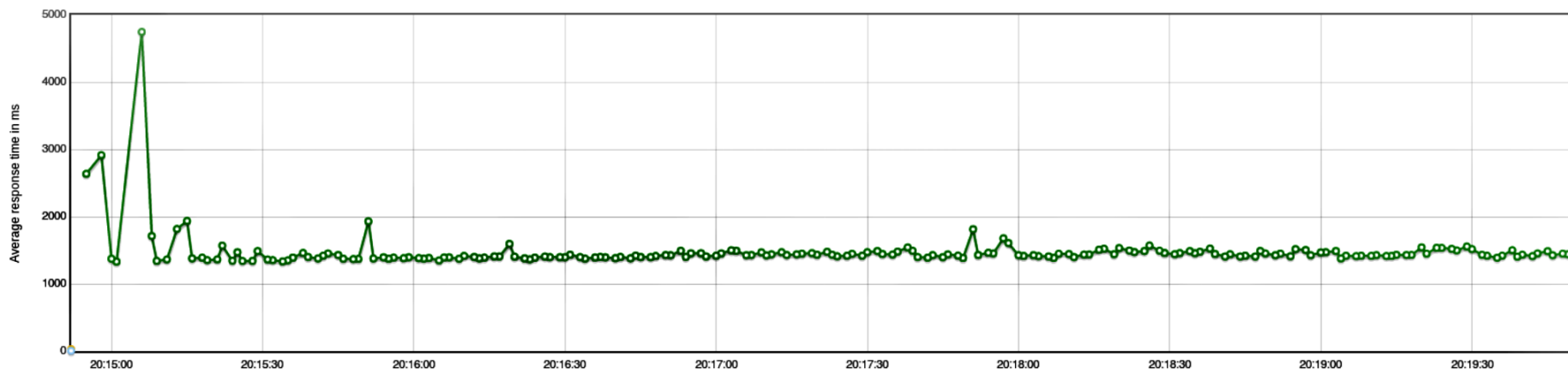
1460 ms

Длительность

5 минут

Необходимая память

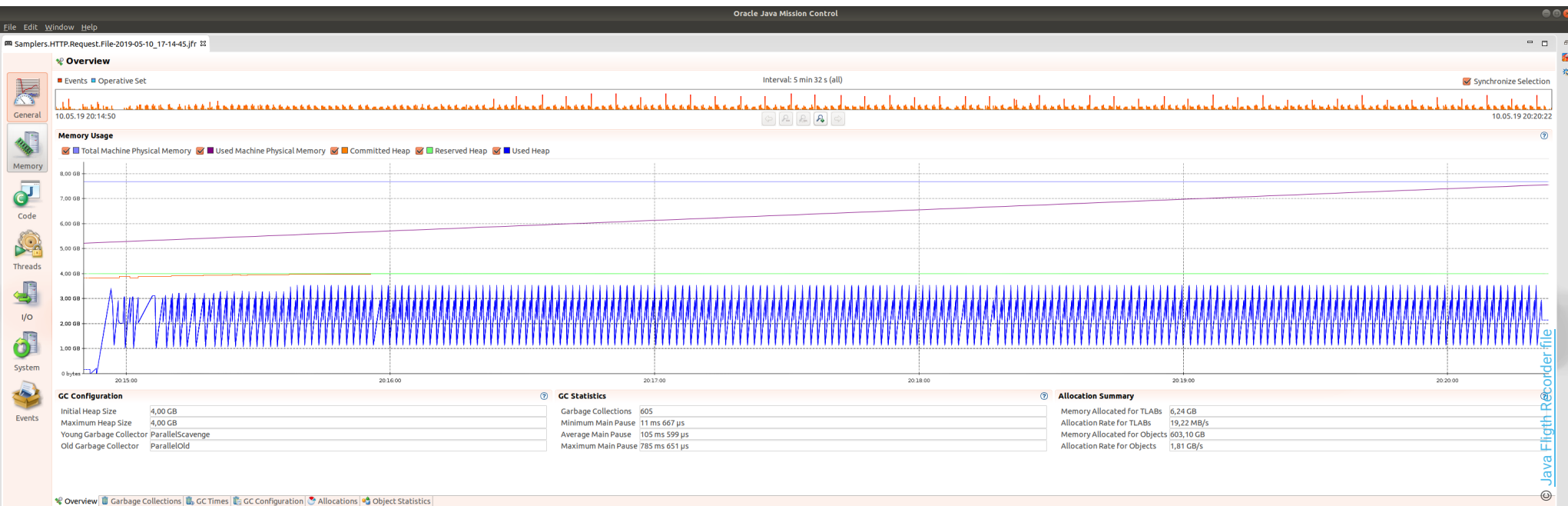
4096 МБайт



Трафик памяти огромный 1,81 ГБайт/сек

83

Память используется постоянно. 1 минута на GC



Apache.JMeter читает тело ответа

java.net.SocketInputStream.socketRead0

java.net.SocketInputStream.socketRead

java.net.SocketInputStream.read

java.net.SocketInputStream.read

org.apache.http.impl.io.SessionInputBufferImpl.streamRead

org.apache.http.impl.io.SessionInputBufferImpl.read

org.apache.http.impl.io.ContentLengthInputStream.read

org.apache.http.conn.EofSensorInputStream.read

org.apache.http.conn.EofSensorInputStream.read

org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.readResponse

org.apache.jmeter.protocol.http.sampler.HTTPAbstractImpl.readResponse

org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.sample

org.apache.jmeter.protocol.http.sampler.HTTPSamplerProxy.sample

org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample

org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample

org.apache.jmeter.threads.JMeterThread.doSampling

org.apache.jmeter.threads.JMeterThread.executeSamplePackage

org.apache.jmeter.threads.JMeterThread.processSampler

org.apache.jmeter.threads.JMeterThread.run

java.lang.Thread.run

java.net.SocketInputStream.socketRead

Sample count: 3622 (94.72%)

Global frame frequency: 3622 (94.72%)

Больше 1 ГБайт не скачать для Непар 4 ГБайт

85

Ошибка **OutOfMemoryError** уже для файла в 1100 МБайт

```
2 java.lang.OutOfMemoryError: Java heap space
3   at java.util.Arrays.copyOf(Arrays.java:3236) ~[?:1.8.0_181]
4   at java.io.ByteArrayOutputStream.grow(ByteArrayOutputStream.java:118) ~[?:1.8.0_181]
5   at java.io.ByteArrayOutputStream.ensureCapacity(ByteArrayOutputStream.java:93) ~[?:1.8.0_181]
6   at java.io.ByteArrayOutputStream.write(ByteArrayOutputStream.java:153) ~[?:1.8.0_181]
7   at org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.readResponse(HTTPSamplerBase.java:1869) ~[ApacheJMeter_
8   at org.apache.jmeter.protocol.http.sampler.HTTPAbstractImpl.readResponse(HTTPAbstractImpl.java:477) ~[ApacheJMeter
9   at org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.sample(HTTPHC4Impl.java:600) ~[ApacheJMeter_http-5.1.1.jar:
10  at org.apache.jmeter.protocol.http.sampler.HTTPSamplerProxy.sample(HTTPSamplerProxy.java:67) ~[ApacheJMeter_http-5
11  at org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample(HTTPSamplerBase.java:1231) ~[ApacheJMeter_http-5
12  at org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample(HTTPSamplerBase.java:1220) ~[ApacheJMeter_http-5
13  at org.apache.jmeter.threads.JMeterThread.doSampling(JMeterThread.java:622) ~[ApacheJMeter_core-5.1.1.jar:5.1.1 r1
14  at org.apache.jmeter.threads.JMeterThread.executeSamplePackage(JMeterThread.java:546) ~[ApacheJMeter_core-5.1.1.jar:
15  at org.apache.jmeter.threads.JMeterThread.processSampler(JMeterThread.java:486) ~[ApacheJMeter_core-5.1.1.jar:5.1.
16  at org.apache.jmeter.threads.JMeterThread.run(JMeterThread.java:253) ~[ApacheJMeter_core-5.1.1.jar:5.1.1 r1855137]
17  at java.lang.Thread.run(Thread.java:748) [?:1.8.0_181]
```

HTTP Request с настройками по умолчанию,

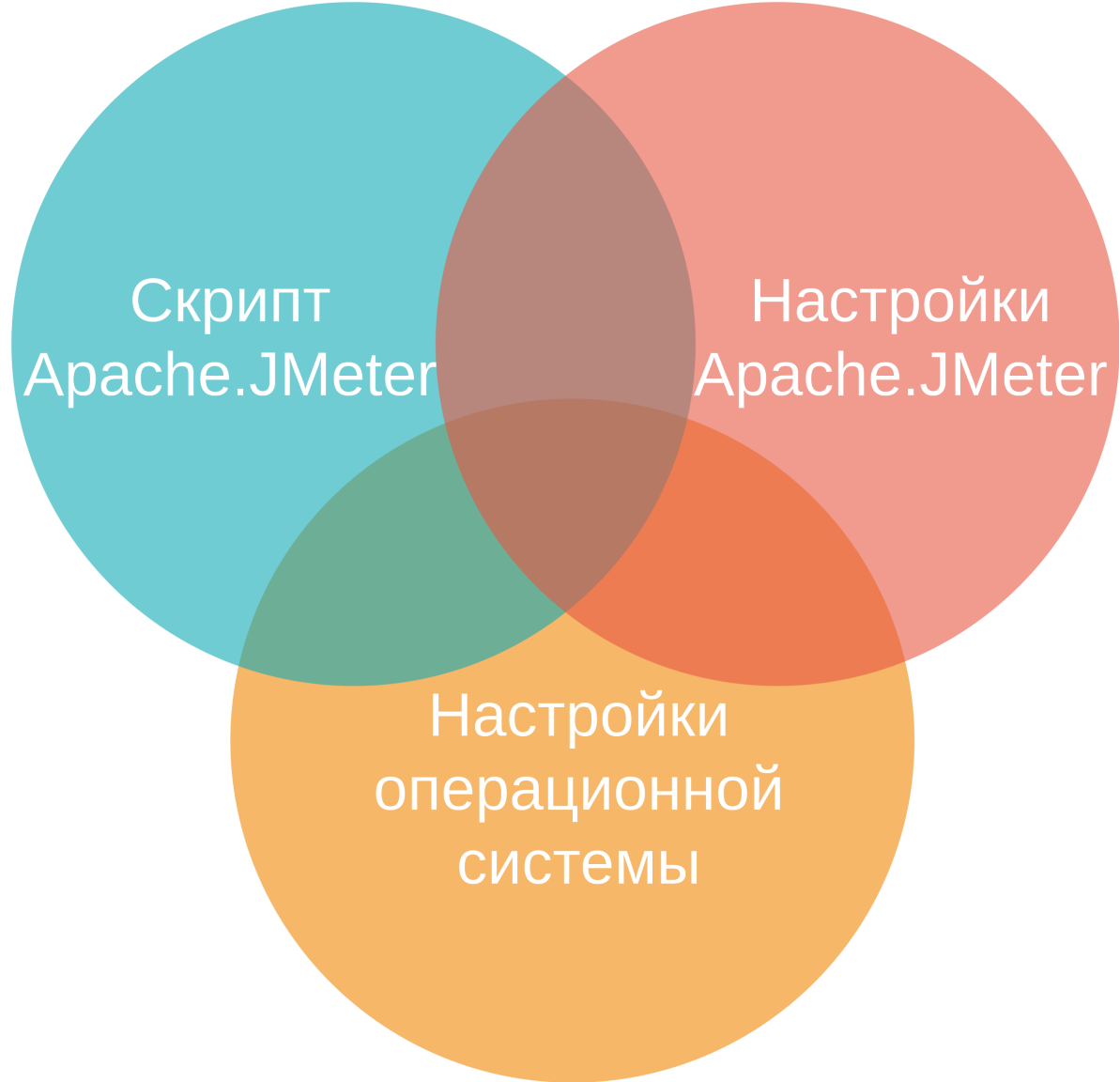
расходует очень много памяти

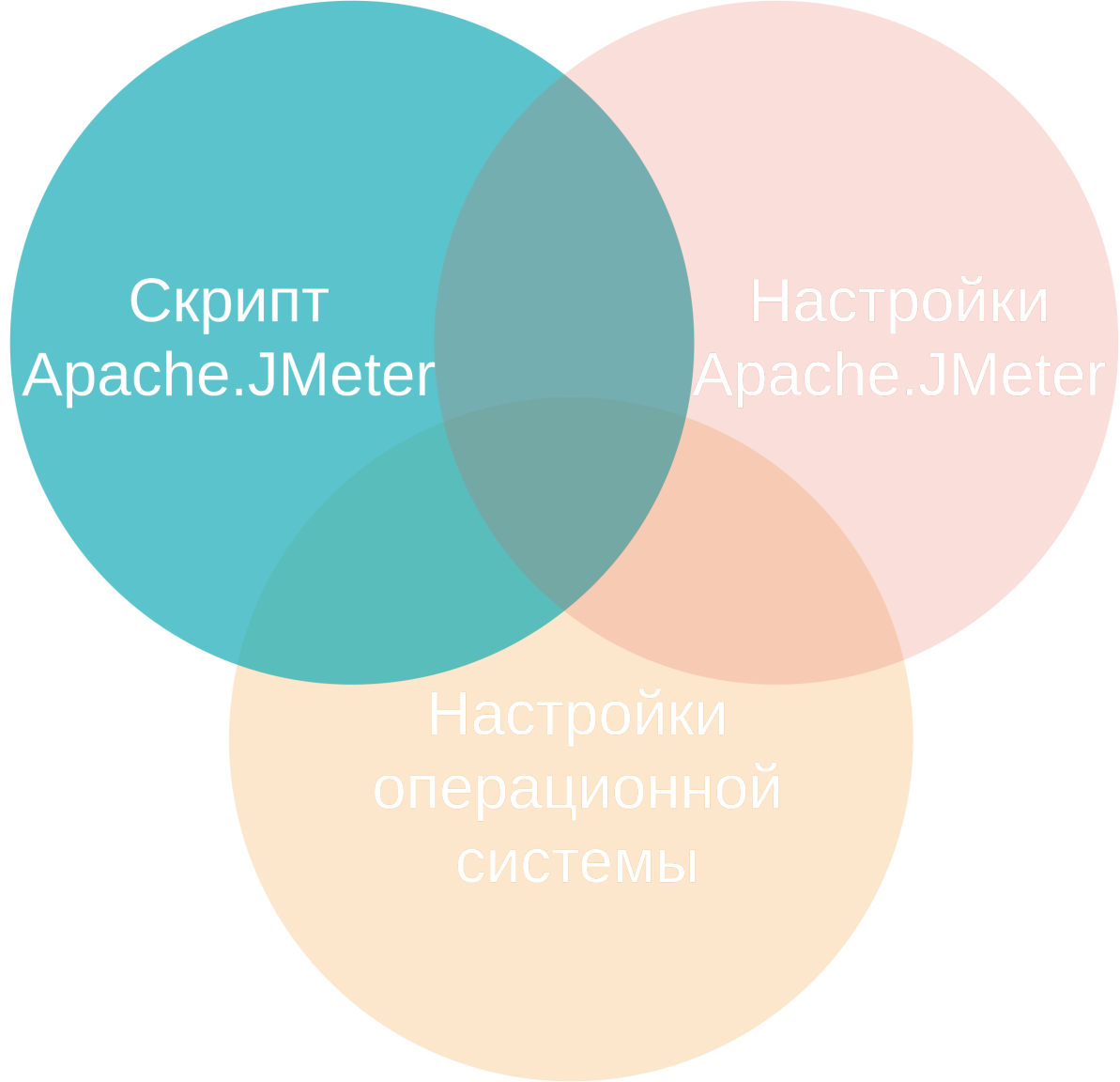
4-х кратный буфер от размера файла

генерируя объекты по 1,8 ГБайт/с,

но качает 200 ГБайт за 5 минут





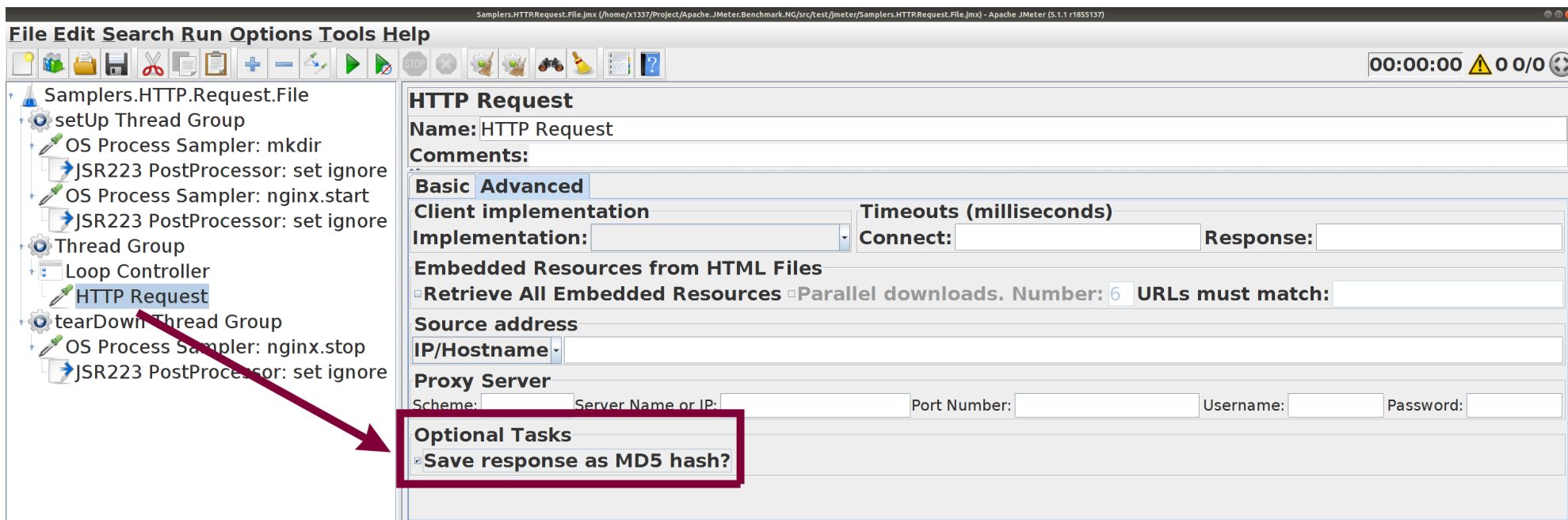




Save response as MD5 Hash

90

экономит память при скачивании больших файлов



200 ГБайт с MD5 скачивались дольше

91

Параметр

Значение

Среднее время

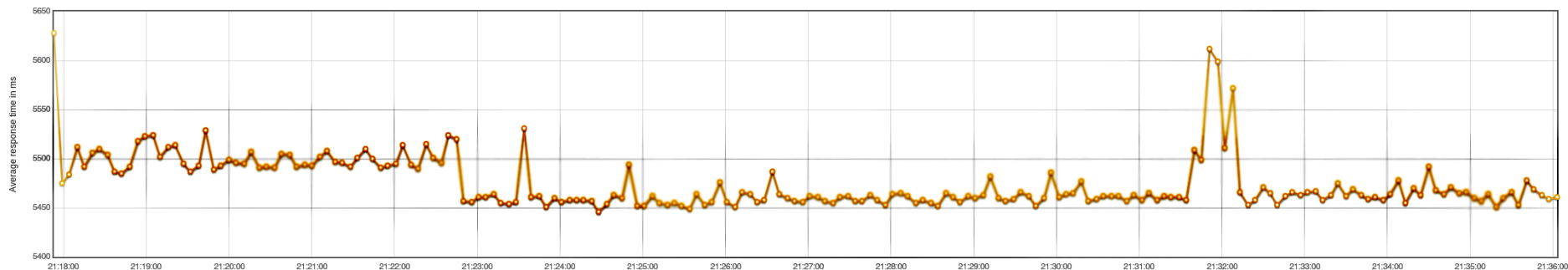
5475 ms

Длительность

18 минут 17 сек

Необходимая память

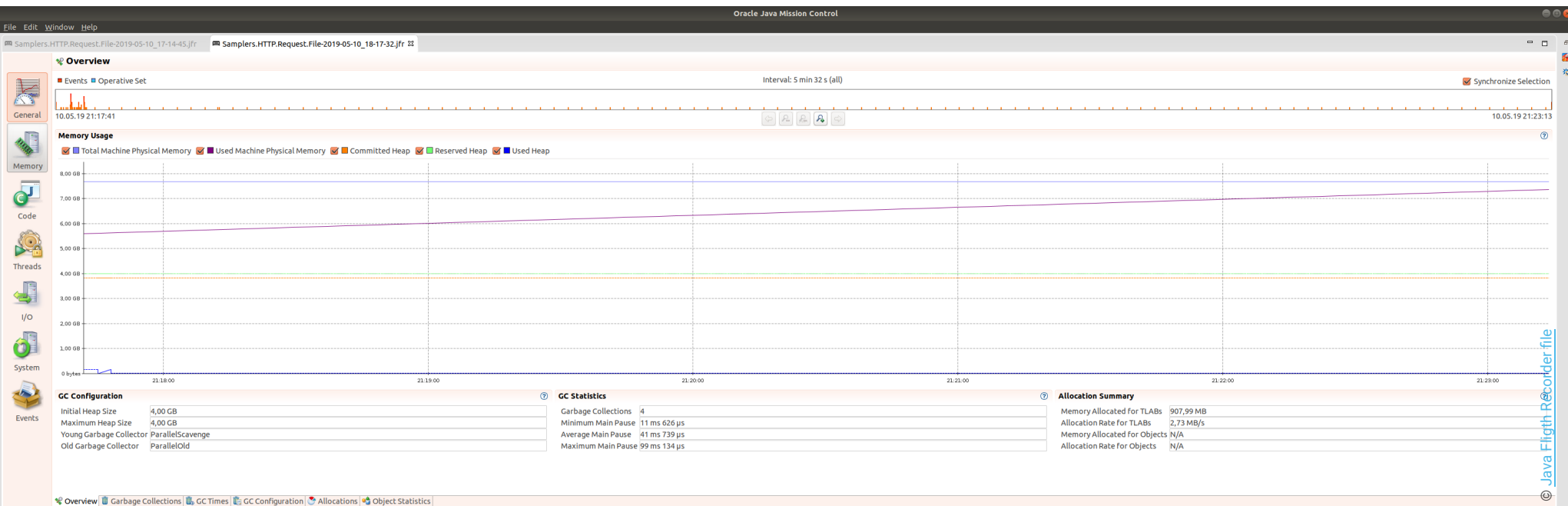
0 МБайт (минимум)



Трафик памяти при включенном MD5 нулевой

92

Память не использовалась вообще, CPU тоже меньше



JMeter считает MD5 в два раза дольше, чем читает

```
java.net.SocketInputStream.socketRead0  
java.net.SocketInputStream.socketRead  
java.net.SocketInputStream.read  
java.net.SocketInputStream.read
```

```
org.apache.http.impl.io.SessionInputBufferI...
```

```
sun....
```

```
org.apache.http.impl.io.SessionInputBufferI...
```

```
sun.security.provider.DigestBase.implCompressMultiBlock
```

```
org.apache.http.impl.io.ContentLengthInputS...
```

```
sun.security.provider.DigestBase.engineUpdate
```

```
org.apache.http.conn.EofSensorInputStream.r...
```

```
java.security.MessageDigest$Delegate.engineUpdate
```

```
org.apache.http.conn.EofSensorInputStream.r...
```

```
java.security.MessageDigest.update
```

```
org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.readResponse
```

```
java.security.MessageDigest.update
```

```
org.apache.jmeter.protocol.http.sampler.HTTPAbstractImpl.readResponse
```

```
Sample count: 7904 (66.90%)
```

```
org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.sample
```

```
Global frame frequency: 7904 (66.90%)
```

```
org.apache.jmeter.protocol.http.sampler.HTTPSamplerProxy.sample
```

```
org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample
```

```
org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample
```

```
org.apache.jmeter.threads.JMeterThread.doSampling
```

```
org.apache.jmeter.threads.JMeterThread.executeSamplePackage
```

```
org.apache.jmeter.threads.JMeterThread.processSampler
```

```
org.apache.jmeter.threads.JMeterThread.run
```

```
java.lang.Thread.run
```



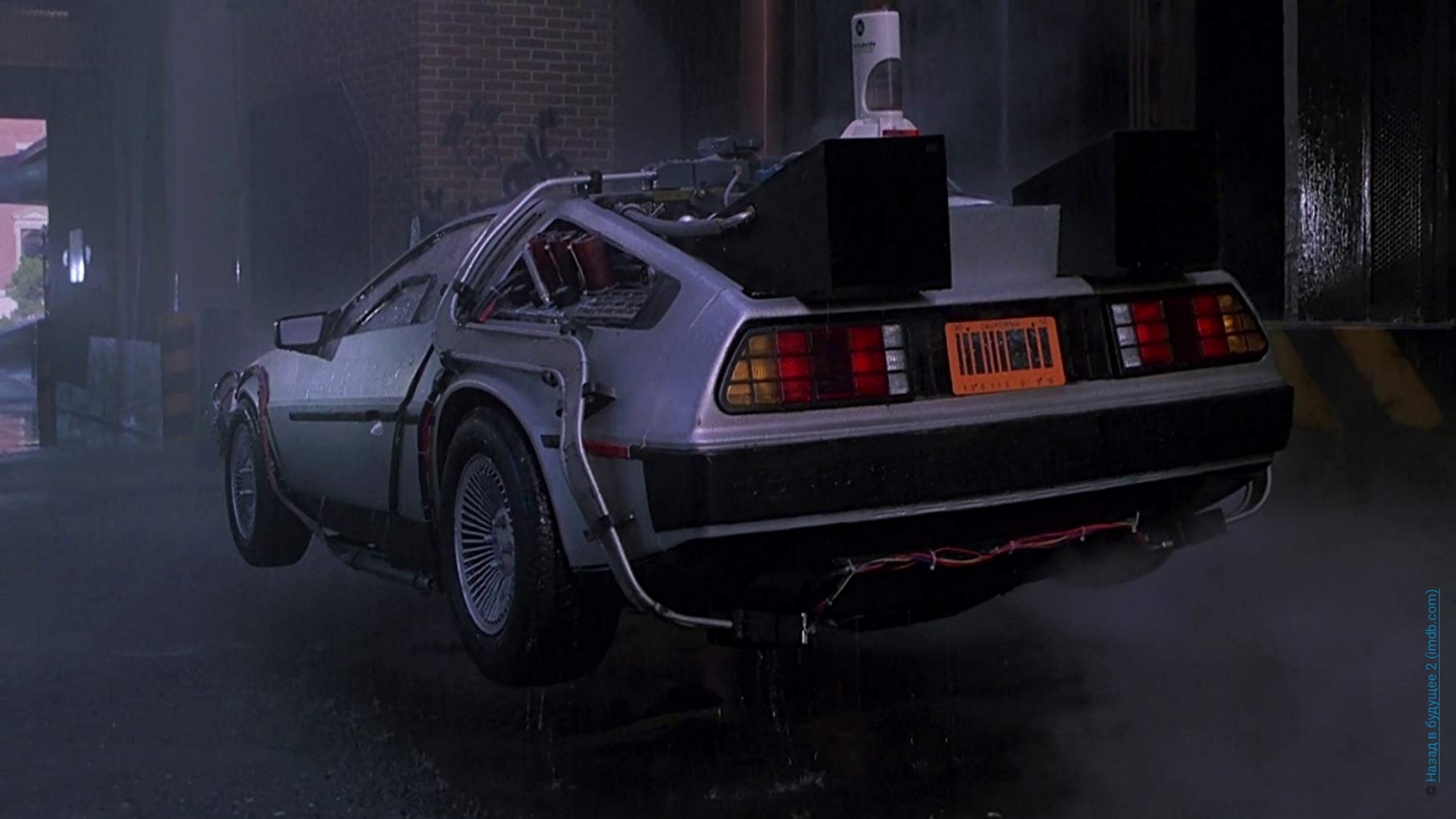
Save response as MD5 Hash

экономит память до 0,

экономит процессорное время,

не имеет ограничений на размер файла,

но замедляет скачивание до 3-х раз



Как скачивать быстро

с минимальными расходами памяти

из Apache.JMeter?

Может быть **curl/wget** и **bash** помогут?

wget есть в Linux

wget-download.sh url

```
01. wget -S \  
02.      --progress=dot:mega \  
03.      --output-document=/dev/null \  
04.      $1
```

wget есть и в Microsoft Windows (MinGW)

98

wget-download.bat url

```
01. wget -S ^
02.      --progress=dot:mega ^
03.      --output-document=- ^
04.      %1 ^
05.      1>NUL
```

Можно ограничить максимальную скорость

99

wget-download.bat **url**

```
01. wget -S ^  
02.     --limit-rate=20m ^  
03.     --progress=dot:mega ^  
04.     --output-document=- ^  
05.     %1 ^  
06.     1>NUL
```

Что превосходит аналог (cps)

100

по удобству использования

```
01. # Define characters per second > 0 to emulate slow connection
```

```
02. httpclient.socket.http.cps=0
```

```
03. httpclient.socket.https.cps=0
```

Прикинуться браузером с поддержкой gzip

101

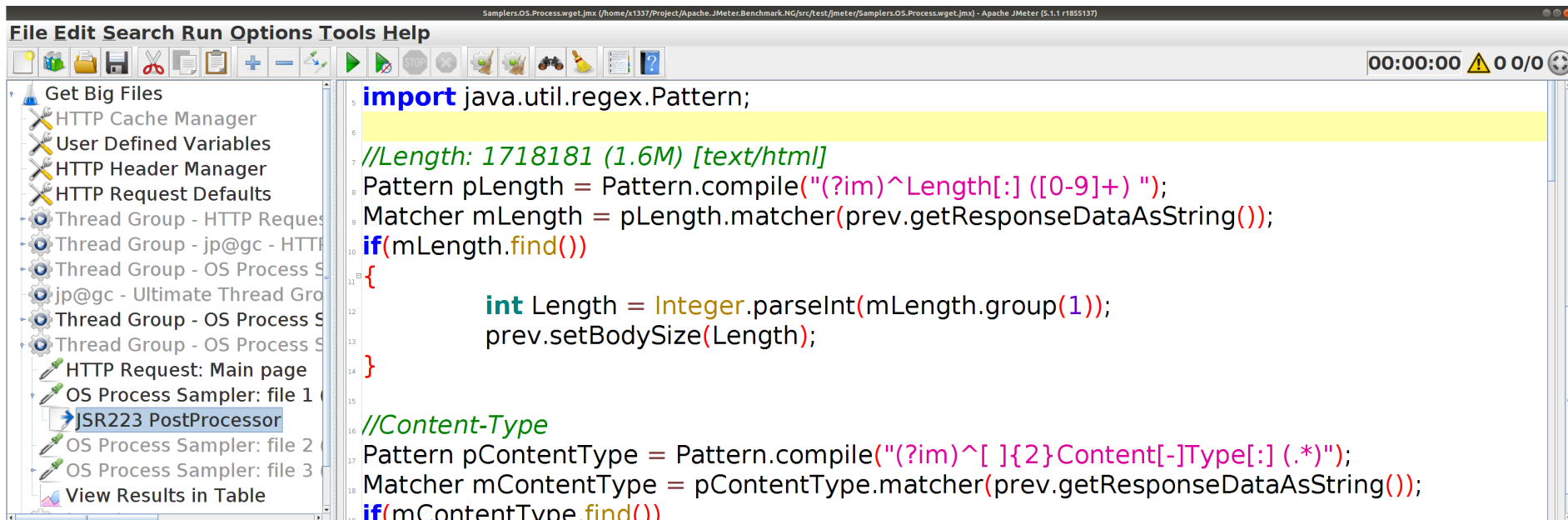
wget-download.gzip.bat url

```
01. wget -S --progress=dot:mega ^
02.     --header "User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64
03.     --header "Accept: text/html,application/xhtml+xml,application/
04.     --header "Accept-Encoding: gzip, deflate" ^
05.     --output-document=- %1 1>NUL
```

wget-download.gzip.head.bat url

```
01. wget -S --progress=dot:mega ^
02.     --header "User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64
03.     --header "Accept: text/html,application/xhtml+xml,application/
04.     --header "Accept-Encoding: gzip, deflate" ^
05.     --output-document=- %1 | ^
06.     head -c 1000 | gzip -d -c -
```

Программное заполнение параметров **SampleResult**



The screenshot displays the Apache JMeter interface. On the left, the 'Tree' view shows a test plan with several samplers and a 'JSR223 PostProcessor' selected. The 'Main' view on the right shows the Java code for the post-processor. The code uses the 'java.util.regex' package to parse the response length and content type from the previous sampler's response data.

```
File Edit Search Run Options Tools Help
00:00:00 0 0/0

Get Big Files
HTTP Cache Manager
User Defined Variables
HTTP Header Manager
HTTP Request Defaults
Thread Group - HTTP Request
Thread Group - jp@gc - HTTP
Thread Group - OS Process S
jp@gc - Ultimate Thread Gro
Thread Group - OS Process S
Thread Group - OS Process S
HTTP Request: Main page
OS Process Sampler: file 1
JSR223 PostProcessor
OS Process Sampler: file 2
OS Process Sampler: file 3
View Results in Table

import java.util.regex.Pattern;

//Length: 1718181 (1.6M) [text/html]
Pattern pLength = Pattern.compile("(?im)^Length[:] ([0-9]+) ");
Matcher mLength = pLength.matcher(prev.getResponseDataAsString());
if(mLength.find())
{
    int Length = Integer.parseInt(mLength.group(1));
    prev.setBodySize(Length);
}

//Content-Type
Pattern pContentType = Pattern.compile("(?im)^[ ]{2}Content[-]Type[:] (.*)");
Matcher mContentType = pContentType.matcher(prev.getResponseDataAsString());
if(mContentType.find())
```


JSR-223 PostProcessor встраивает wget

104

Сохраняется Response Body

The screenshot shows the Apache JMeter 5.1.1 interface. The left sidebar displays the test plan structure: Samplers.OS.Process.wget.new, setUp Thread Group, Thread Group, tearDown Thread Group, and View Results Tree (selected). The main window is titled 'View Results Tree' and shows the 'Response Body' tab for the 'OS Process Sampler'. The response body contains a table with 8 rows of data, showing the output of the JSR-223 PostProcessor.

Text	Sampler result	Request	Response data
	Response Body	Response headers	
✓ OS Process Sampler	OK	0%	610M 2s
✓ OS Process Sampler	3072K	0%	563M 2s
✓ OS Process Sampler	6144K	0%	533M 2s
✓ OS Process Sampler	9216K	1%	528M 2s
✓ OS Process Sampler	12288K	1%	535M 2s
✓ OS Process Sampler	15360K	1%	667M 2s
✓ OS Process Sampler	18432K	2%	833M 2s

JSR-223 PostProcessor встраивает wget

105

Сохраняется Response Headers

The screenshot displays the Apache JMeter 5.1.1 user interface. The top menu bar includes File, Edit, Search, Run, Options, Tools, and Help. Below the menu is a toolbar with various icons for file operations and execution. The main window is divided into several panels. On the left is the 'Samplers.OS.Process.wget.new' tree, which includes 'setUp Thread Group', 'Thread Group', 'tearDown Thread Group', and 'View Results Tree'. The 'View Results Tree' panel is active, showing a list of 'OS Process Sampler' entries, each with a green checkmark. The right panel shows the 'View Results Tree' details for a selected entry. It includes fields for 'Name' (View Results Tree), 'Comments', and 'Write results to file / Read from file'. Below these is a 'Search' field with 'Case sensitive' and 'Regular exp.' checkboxes, and 'Search' and 'Reset' buttons. The 'Text' tab is selected, showing the 'Response data' section. The 'Response Body' and 'Response headers' tabs are also visible. The 'Response headers' section lists the following headers:

```
1 Server: nginx/1.14.0 (Ubuntu)
2 Date: Fri, 10 May 2019 23:04:53 GMT
3 Content-Type: application/octet-stream
4 Content-Length: 1073741824
5 Last-Modified: Fri, 10 May 2019 16:31:30 GMT
Connection: keep-alive
```

200 ГБайт с wget скачивались очень быстро

106

Параметр

Значение

Среднее время

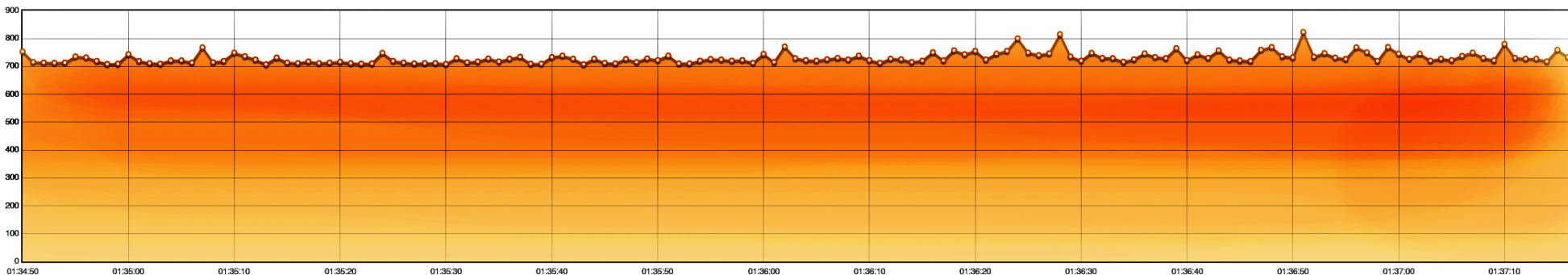
727 ms

Длительность

2 минуты 27 сек

Необходимая память

0 МБайт (минимум)



Скачать 200 ГБайт

107

Способ

Длительность

HTTP Request

5 минут



Save response as MD5 Hash

18 минут 17 сек

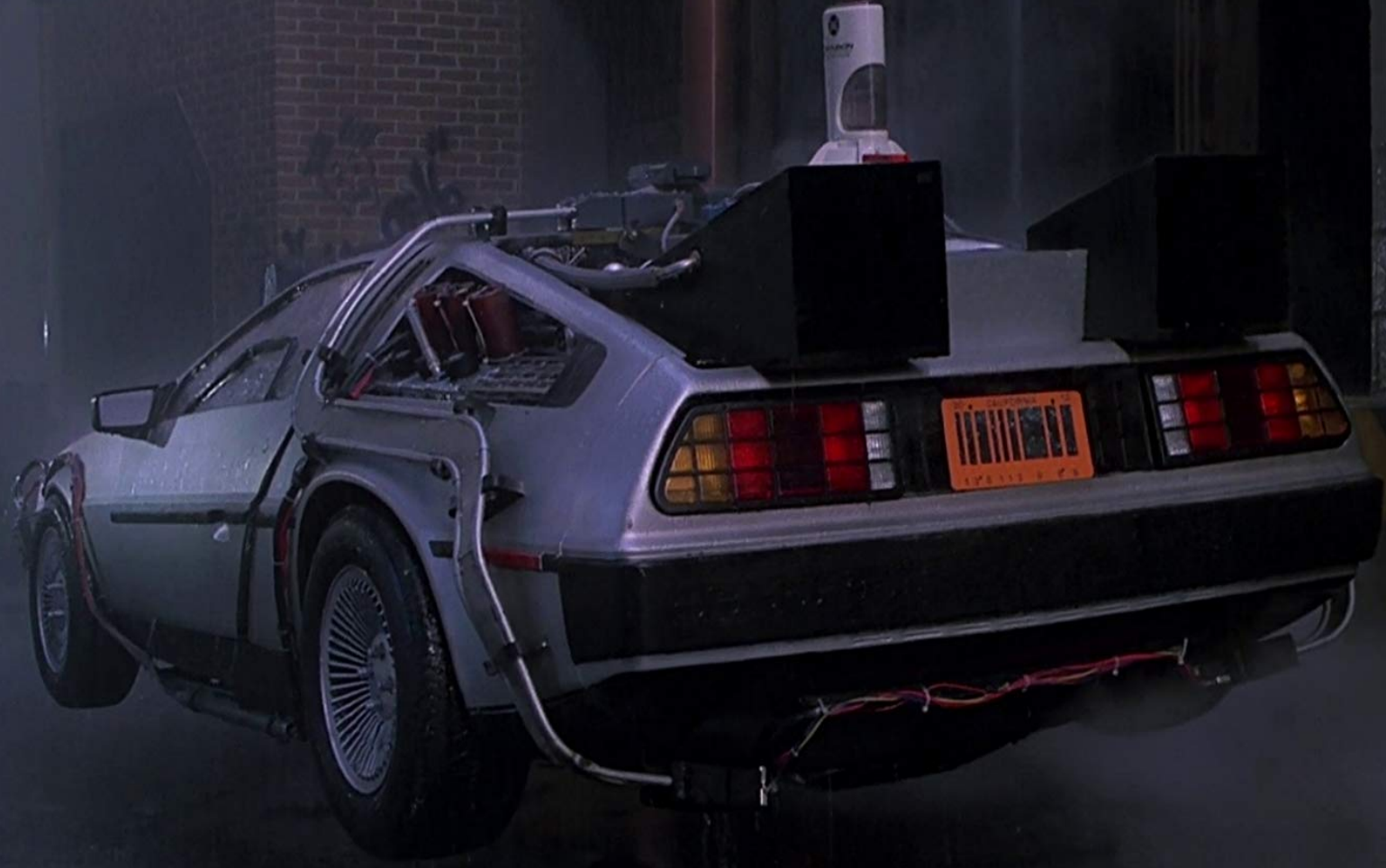
OS Process Sampler и wget

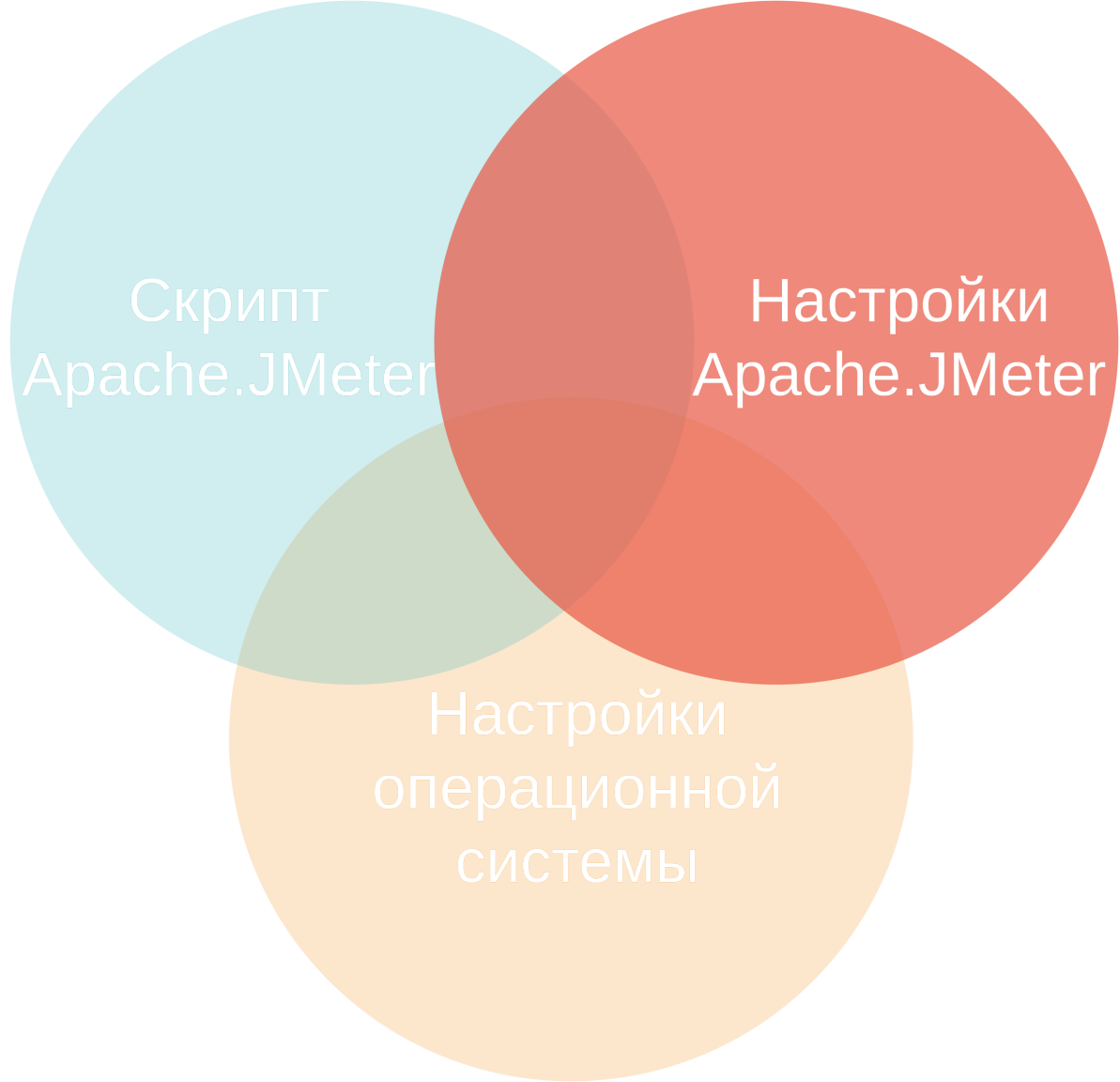
2 минуты 27 сек

OS Process Sampler + JSR-223 PostProcessor

позволяют вызывать консольные инструменты, которые хорошо работают с большими файлами.

Так **wget** даёт ускорение в **2** раза, и в **7** раз по сравнению с MD5.





Попробуем ограничить читаемый размер до 100 байт

httpsampler.max_bytes_to_store_per_request=100

```
01. # Max size of bytes stored in memory per SampleResult
02. # Ensure you don't exceed max capacity of a Java Array and re
03. # that the higher it is, the higher JMeter will consume heap
04. # Defaults to 0, which means no truncation
05. #httpsampler.max_bytes_to_store_per_request=0
```


Быстрее с max_bytes_to_store_per_request=100

112

Параметр

Значение

Среднее время

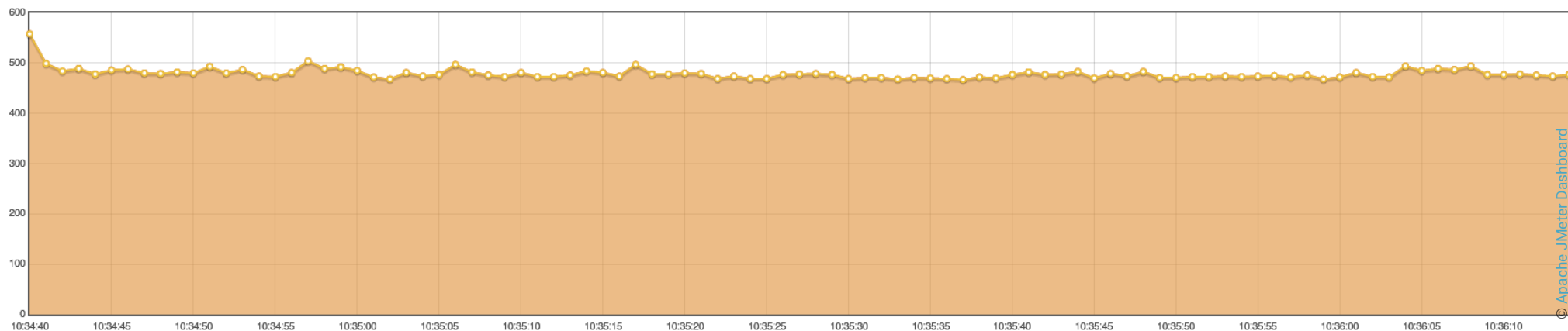
475 ms

Длительность

1 минута 36 сек

Необходимая память

180 МБайт (немного)



SJK: socketRead (99.35%)

113

JMeter читает, ни на что не отвлекается

java.net.SocketInputStream.socketRead0

java.net.SocketInputStream.socketRead

java.net.SocketInputStream.read

java.net.SocketInputStream.read

org.apache.http.impl.io.SessionInputBufferImpl.streamRead

org.apache.http.impl.io.SessionInputBufferImpl.read

org.apache.http.impl.io.ContentLengthInputStream.read

org.apache.http.conn.EofSensorInputStream.read

org.apache.http.conn.EofSensorInputStream.read

org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.readResponse

org.apache.jmeter.protocol.http.sampler.HTTPAbstractImpl.readResponse

org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.sample

org.apache.jmeter.protocol.http.sampler.HTTPSamplerProxy.sample

org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample

org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample

org.apache.jmeter.threads.JMeterThread.doSampling

org.apache.jmeter.threads.JMeterThread.executeSamplePackage

org.apache.jmeter.threads.JMeterThread.processSampler

org.apache.jmeter.threads.JMeterThread.run

java.lang.Thread.run

java.net.SocketInputStream.socketRead0

Sample count: 1999 (99.35%)

Global frame frequency: 1999 (99.35%)

Скачать 200 ГБайт

114

Способ	Длительность
HTTP Request	5 минут
<input checked="" type="checkbox"/> Save response as MD5 Hash	18 минут 17 сек
OS Process Sampler и wget	2 минуты 27 сек
max_bytes_to_store_per_request=100	1 минута 36 сек



HTTP Request

Отправка файлов

HTTP Request. Отправка файлов

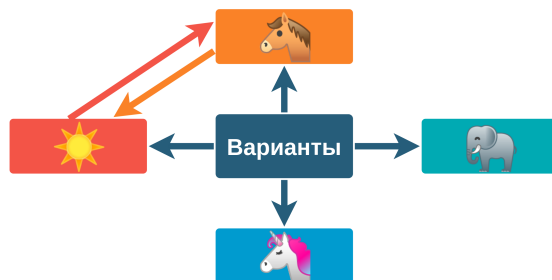
Загрузка файлов быстрая  и экономная

Пользуйтесь вкладкой **Files Upload**.

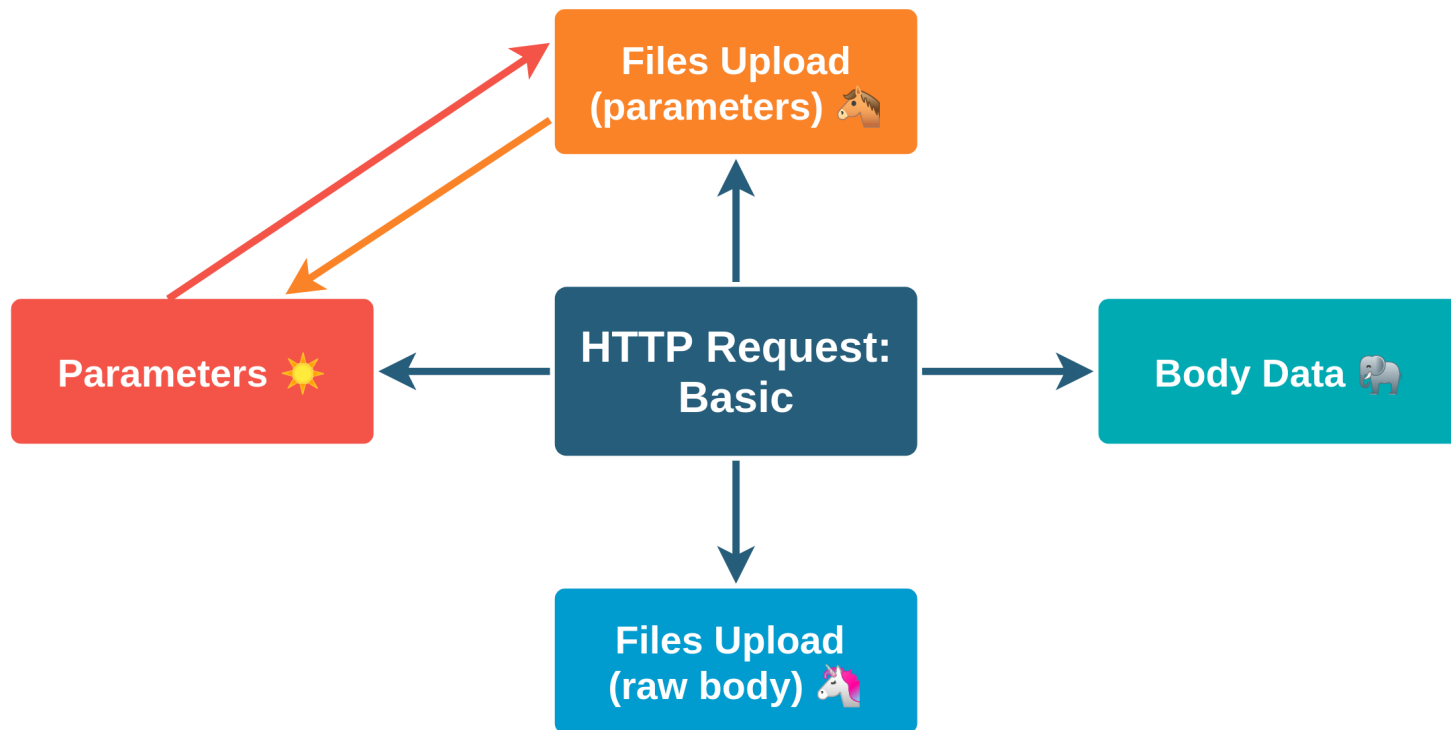
Комбинации вариантов передачи параметров

118

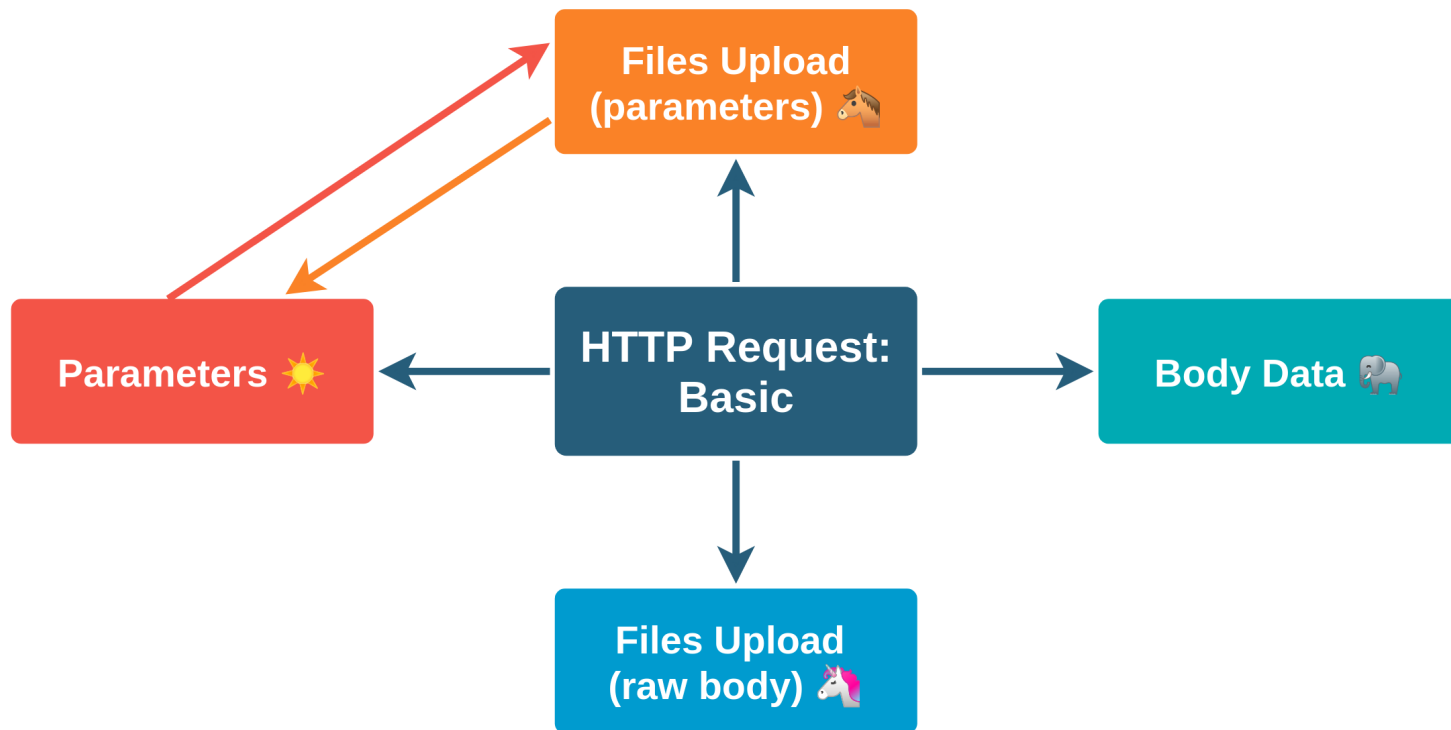
Вариант передачи	Один	С 	С 	С 	С 
Parameters					
Body Data					
Files Upload (All)					
Files Upload (File Path)					



☀ Parameters и 🐎 Files Upload (parameters) дружат



🦄 Files Upload (body data) и 🐘 Body Data сами по себе



HTTP POST один параметр

121

 **multipart/form-data**

File Edit Search Run Options Tools Help

00:00:00 0 0/0

Samplers.HTTP.Request.Upload
setUp Thread Group
Thread Group
Loop Controller
HTTP Request: File param
HTTP Header Manager
HTTP Request: File params
HTTP Header Manager
HTTP Request: Raw Body
HTTP Header Manager
tearDown Thread Group
View Results Tree
Property Display

HTTP Request

Name: HTTP Request: File param

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: \${RequestHost} Port Number: testPort}

HTTP Request

Method: POST Path: \${RequestPath} Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

File Path	Parameter Name	MIME Type
\${UploadFile}	userfile	\${UploadFileMIME}

Add Browse... Delete

HTTP POST один параметр

122

 **multipart/form-data**

File Edit Search Run Options Tools Help

00:00:00 0 0/0

Samplers.HTTP.Request.Upload

setUp Thread Group

Thread Group

Loop Controller

HTTP Request: File param

HTTP Header Manager

HTTP Request: File param

HTTP Header Manager

HTTP Request: Raw Body

HTTP Header Manager

tearDown Thread Group

View Results Tree

Property Display

HTTP Request

Name: HTTP Request: File param

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: \${RequestHost} Port Number: testPort}

HTTP Request

Method: POST Path: \${RequestPath} Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☒ Use multipart/form-data ☐ Browser-compatible headers

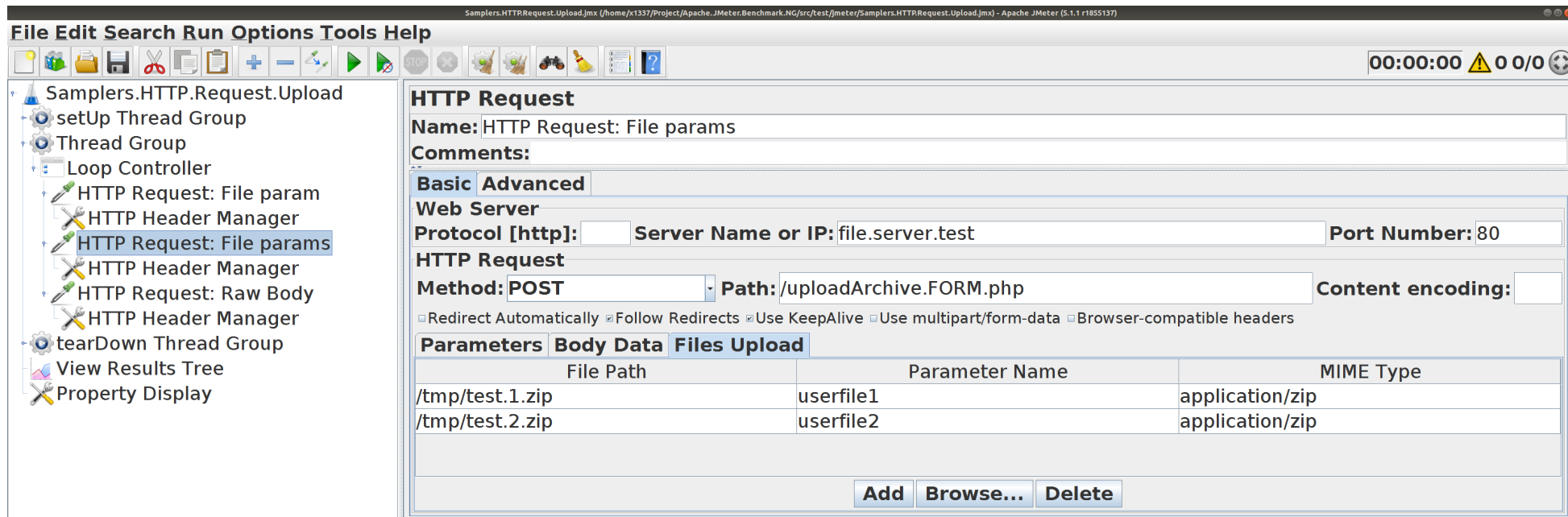
File Path	Parameter Name	MIME Type
\${UploadFile}	uploadfile	\${UploadFileMIME}

Add Browse... Delete

HTTP POST несколько параметров

123

 multipart/form-data



The screenshot shows the Apache JMeter GUI. The left sidebar contains a tree view of the test plan, with 'HTTP Request: File params' selected. The main panel displays the configuration for this sampler. The 'Basic' tab is active, showing the following settings:

- Name:** HTTP Request: File params
- Comments:**
- Web Server:** Protocol [http], Server Name or IP: file.server.test, Port Number: 80
- HTTP Request:** Method: POST, Path: /uploadArchive.FORM.php, Content encoding:
- ☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

The 'Files Upload' tab is also visible, showing a table of files to be uploaded:

File Path	Parameter Name	MIME Type
/tmp/test.1.zip	userfile1	application/zip
/tmp/test.2.zip	userfile2	application/zip

At the bottom of the 'Files Upload' tab, there are buttons for 'Add', 'Browse...', and 'Delete'.

HTTP POST (raw body)

124



Загрузка тела запроса из файла

Samplers.HTTP.Request.Upload.jmx (/home/x1337/Project/Apache.JMeter.Benchmark.NG/src/test/jmeter/Samplers.HTTP.Request.Upload.jmx) - Apache JMeter (5.1.1 r1855137)

File Edit Search Run Options Tools Help

00:00:00 0 0/0

- Samplers.HTTP.Request.Upload
 - setUp Thread Group
 - Thread Group
 - Loop Controller
 - HTTP Request: File param
 - HTTP Header Manager
 - HTTP Request: File params
 - HTTP Header Manager
 - HTTP Request: Raw Body**
 - HTTP Header Manager
 - tearDown Thread Group
 - View Results Tree
 - Property Display

HTTP Request

Name: HTTP Request: Raw Body

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: file.server.test Port Number: 80

HTTP Request

Method: POST Path: /uploadArchive.FORM.php Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data **Files Upload**

File Path	Parameter Name	MIME Type
/tmp/HTTP.POST.Body.content.txt		

Add Browse... Delete

HTTP POST (raw body)

125



Нужно заполнить только File Path

File Edit Search Run Options Tools Help

00:00:00 0 0/0

Samplers.HTTP.Request.Upload

setUp Thread Group

Thread Group

Loop Controller

HTTP Request: File param

HTTP Header Manager

HTTP Request: File params

HTTP Header Manager

HTTP Request: Raw Body

HTTP Header Manager

tearDown Thread Group

View Results Tree

Property Display

HTTP Request

Name: HTTP Request: Raw Body

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: file.server.test Port Number: 80

HTTP Request

Method: POST Path: /uploadArchive FORM.php Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

File Path	Parameter Name	MIME Type
/tmp/HTTP.POST.Body.content.txt		

Add Browse... Delete

Всё просто прекрасно,
но загружаемые файлы должны быть готовы.

Если нужна параметризация содержимого, то
чем заменить вкладку **Body Data**?

HTTP Request. Отправка файлов

Как готовить файлы на 10 ГБайт?

Во время теста! С минимальными затратами!

Подготовить архив, распаковать, заменить в нём строки



Archive.tar.gz



tar



Templates



sed, bash



Profit!

HTTP Request. Отправка файлов

Большие файлы лучше читать с диска.

Большие тела запросов лучше читать с диска.

Готовить большие файлы удобно с **gzip**, **sed**, ...

Пользуйтесь вкладкой **Files Upload**,




где также можно загружать и тело запроса.



PostProcessor-ы

Как отличается их скорость?

Найдём заголовок страницы



About

- Overview
- License



Download

- Download Releases
- Release Notes

Documentation

- Get Started
- User Manual
- Best Practices
- Component Reference
- Functions Reference
- Properties Reference

Apache JMeter™



The **Apache JMeter™** application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

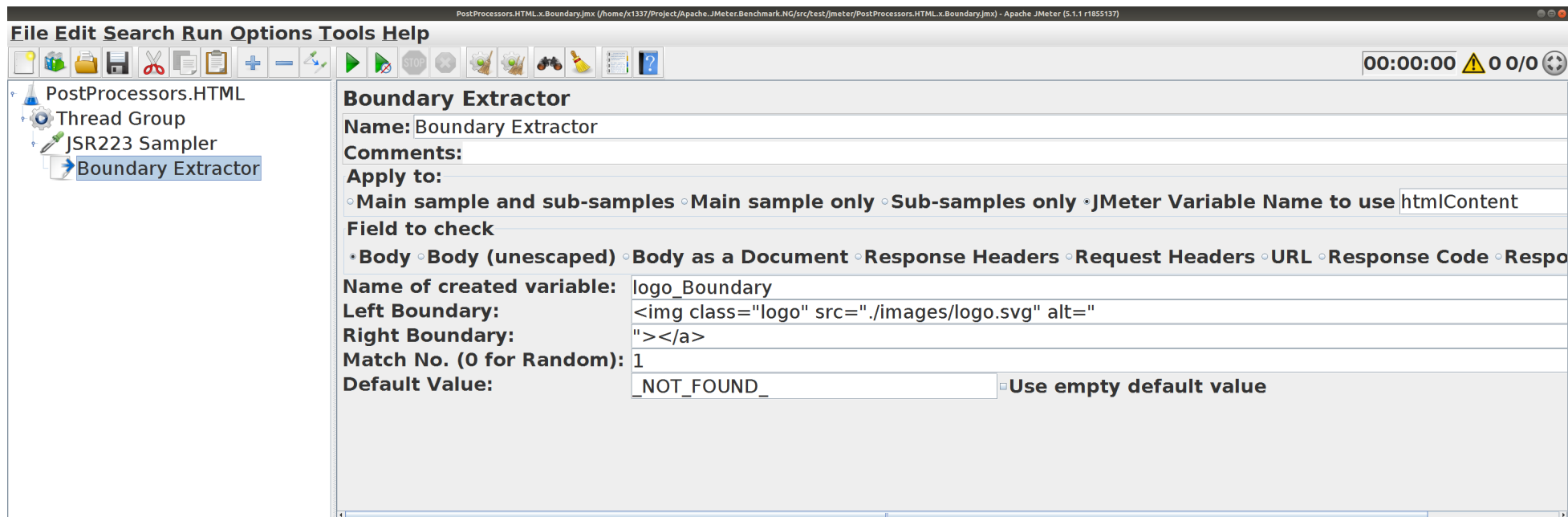
What can I do with it?

Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications. It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load types.

Boundary Extractor

133

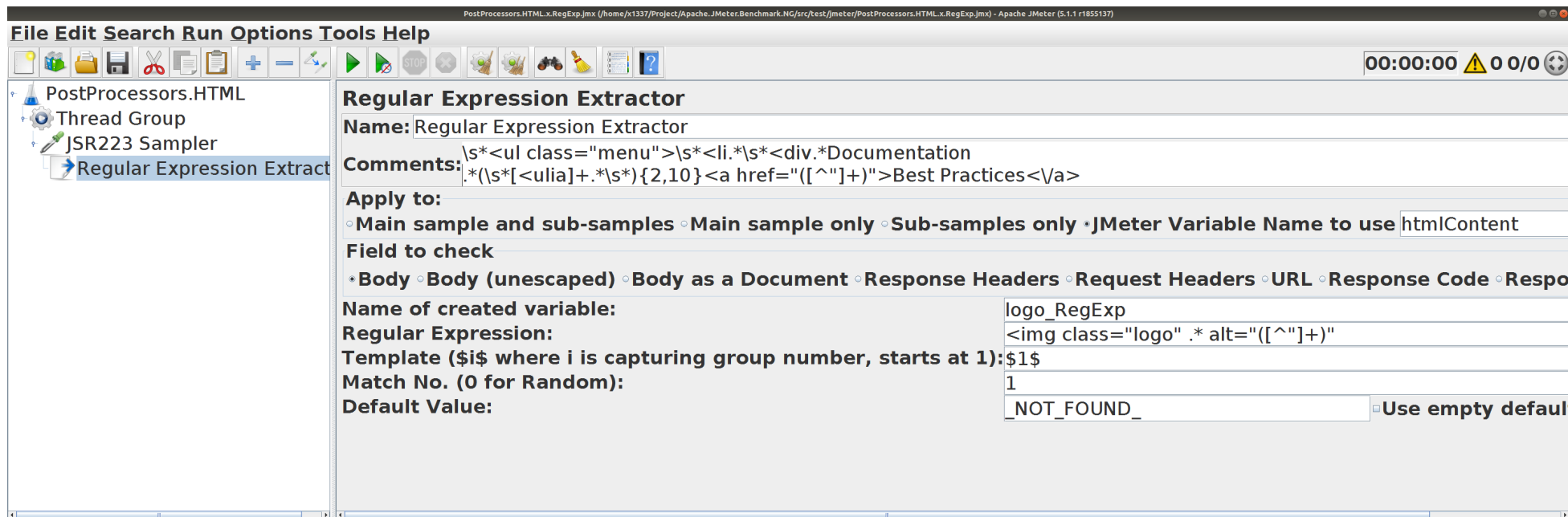
для **htmlContent**



Regular Expression Extractor

134

для **htmlContent**



JSR-223 PostProcessor

135

The screenshot displays the Apache JMeter user interface. The top menu bar includes 'File', 'Edit', 'Search', 'Run', 'Options', 'Tools', and 'Help'. Below the menu is a toolbar with various icons for file operations and execution. The status bar at the top right shows a timer at '00:00:00', a warning icon, and '0 0/0'. The left sidebar shows a tree view of the test plan: 'PostProcessors.HTML' (expanded), 'Thread Group', 'JSR223 Sampler', and 'JSR223 PostProcessor' (selected). The main configuration area for the 'JSR223 PostProcessor' is visible, containing the following fields and options:

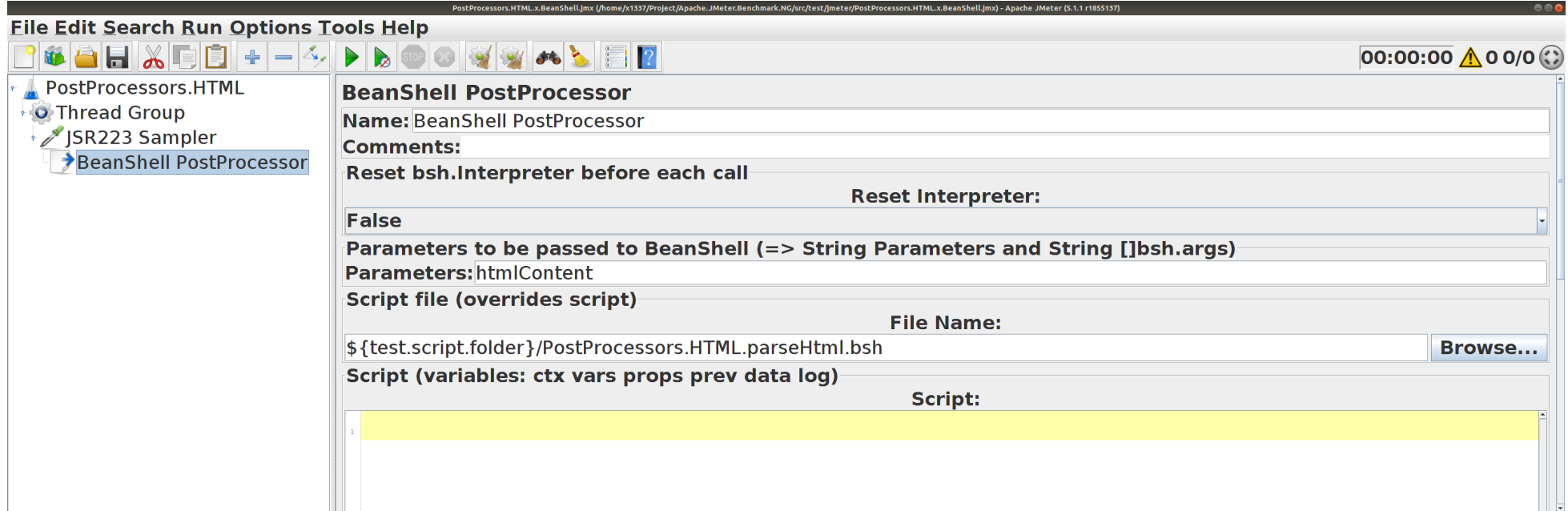
- Name:** JSR223 PostProcessor
- Comments:** (empty text area)
- Script language (e.g. groovy, jexl, javascript (avoid for performances), ...):** groovy (Groovy 2.4.16 / Groovy Scripting Engine 2.0)
- Parameters to be passed to script (=> String Parameters and String []args):** Parameters: htmlContent
- Script file (overrides script):** File Name: \${test.script.folder}/PostProcessors.HTML.parseHtml.groovy (with a 'Browse...' button)
- Script compilation caching:** Cache compiled script if available: ☒
- Script (variables: ctx vars props prev sampler log Label Filename Parameters args[] OUT):** (empty text area)

аналогичный работе Boundary Extractor

```
1
2  def varName = 'htmlContent'
3  def htmlContent = vars.get(varName)
4
5  def logo_jsr223 = '_NOT_FOUND_'
6
7  def logo = ''
8  def boundaryStart = '</a>'
15      def indexEnd = logo.indexOf(boundaryEnd)
16      if(indexEnd != -1) {
17          logo_jsr223 = logo.substring(0, indexEnd)
18      }
19  }
20
21  vars.put('logo_jsr223', logo_jsr223)
22
```

BeanShell PostProcessor

137



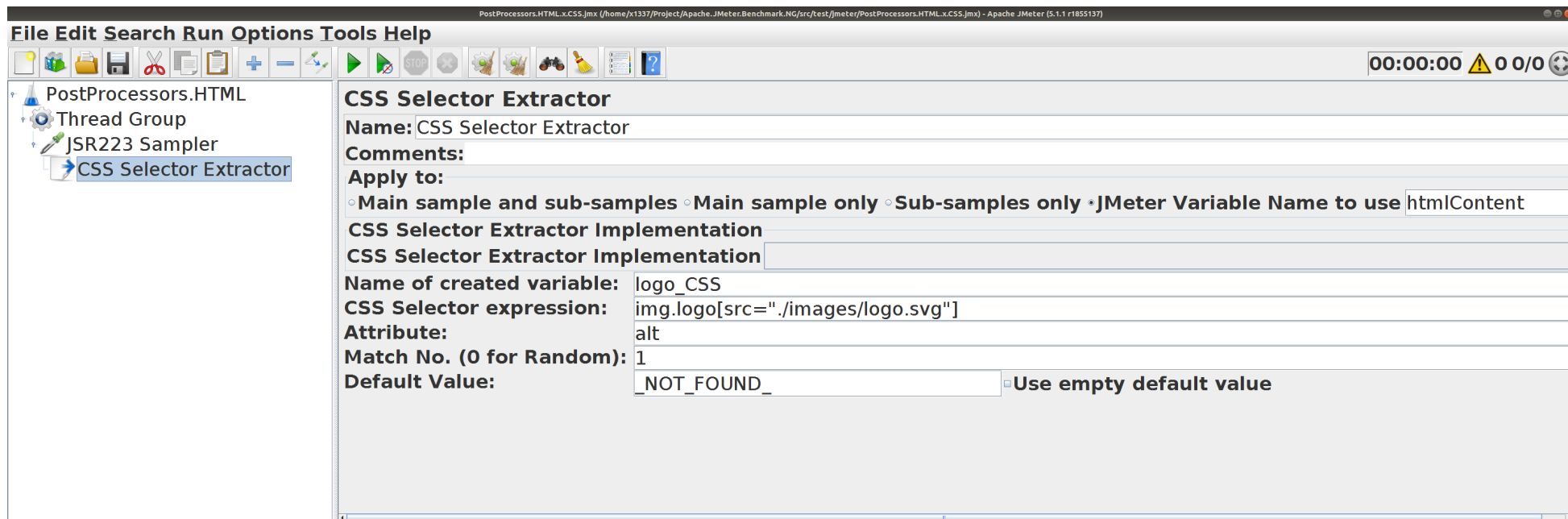
Аналогичный работе Boundary Extractor

```
1
2  var varName = Parameters;
3
4  var htmlContent = String.valueOf(vars.get(varName));
5
6
7  var logo_bsh = "_NOT_FOUND_";
8
9  var logo = "";
10 var boundaryStart = "<img class=\"logo\" src=\"./images/logo.svg\" alt=\"\"";
11 var boundaryEnd = "></a>";
12
13 var indexStart = htmlContent.indexOf(boundaryStart) + boundaryStart.length();
14 var indexEnd;
15 if (indexStart != -1) {
16     logo = htmlContent.substring(indexStart);
17
18     indexEnd = logo.indexOf(boundaryEnd);
19     if(indexEnd != -1) {
20         logo = logo.substring(0, indexEnd);
21         logo_bsh = logo;
22     }
23 }
24
25 vars.put("logo_bsh", logo_bsh);
```

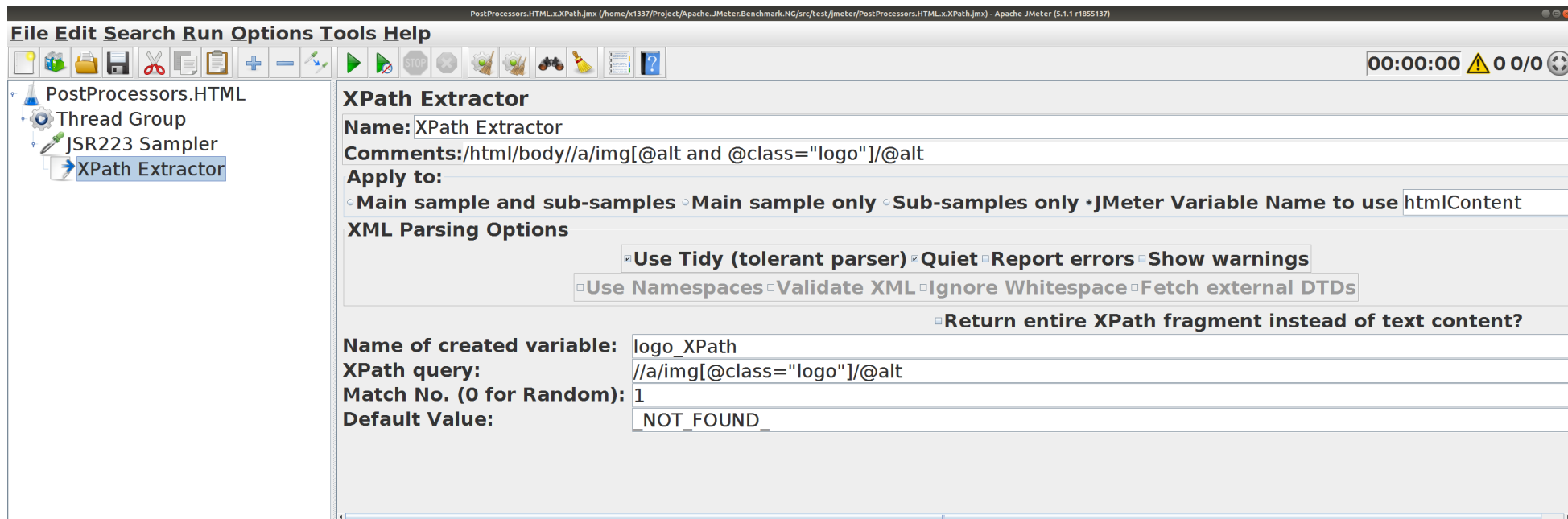
CSS Selector Extractor

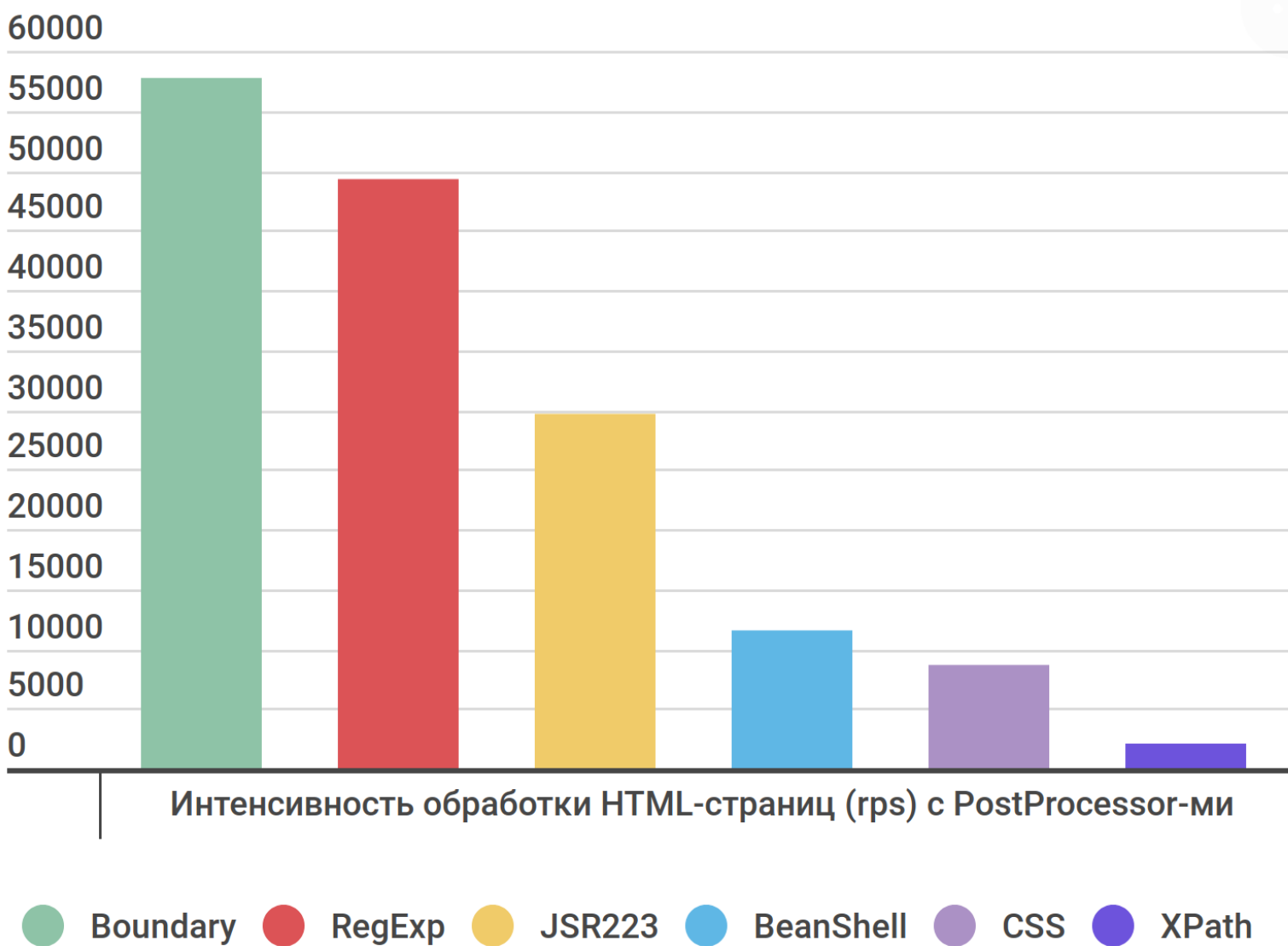
139

для **htmlContent**



XPath (HTML) для **htmlContent**

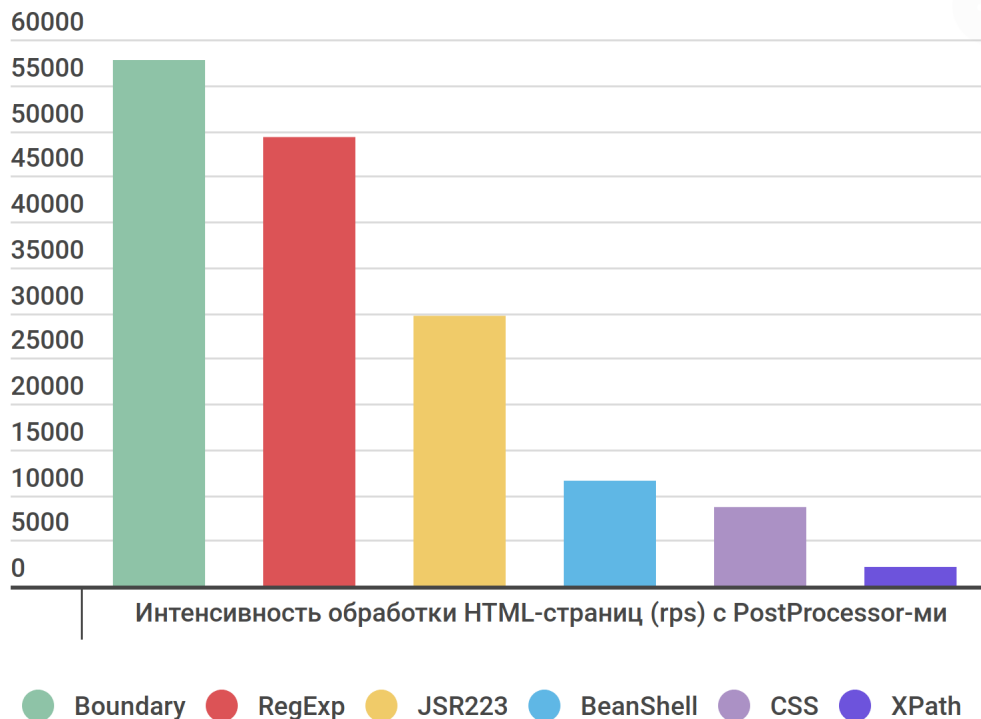




Boundary (58k) самый быстрый

142

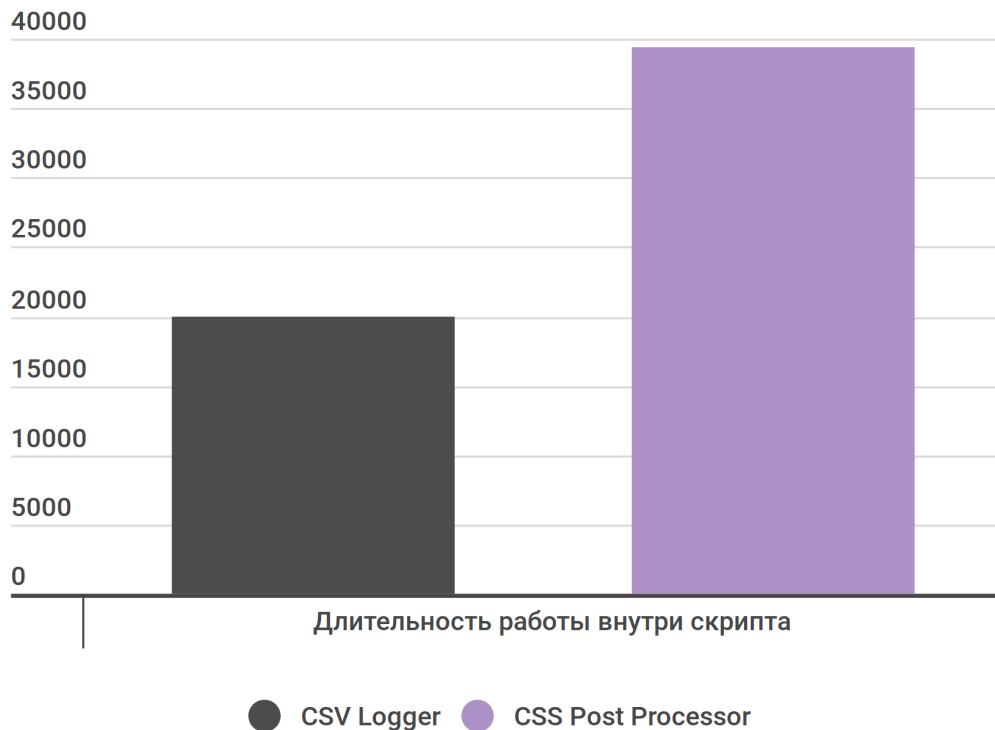
RegExp (50k) почти такой же быстрый



Общее влияние PostProcessor-ов небольшое

143

Так **CSS Selector** выполняется дольше **CSV Logger** лишь в 2 раза



PostProcessor для HTML

Точно не использовать **XPath** для HTML.

Лидеры **RegExp** и **Boundary Extractor**.

Но и **CSS Selector** неплох — лишь x2 от логгера.



PreProcessor-ы

Как быстро подготовить XML?

Как быстро подготовить XML?

147

groovy.xml.MarkupBuilder x5 от SimpleTemplateEngine

Script (variables: ctx vars props prev sampler log Label Filename Parameters args[] OUT)
Script:

```
1 def writer = new StringWriter()
2 def builder = new groovy.xml.MarkupBuilder(writer)
3 builder.setDoubleQuotes(true)
4 builder.mkp.xmlDeclaration(version: '1.0', encoding: 'utf-8')
5 builder.ROOT(attribute1:'RUR'){
6     SUB_1 {
7         for (int index = 0; index < 200; index++) {
8             SUB_2(attribute2: index) {
9                 SUB_3 'Text ' + index + ' Hello, World'
10            }
11        }
12    }
13 }
14 vars.put("XML", writer.toString())
```

PreProcessor-ы

В Groovy **groovy.xml.MarkupBuilder** быстрее

SimpleTemplateEngine, быстрее сериализации.

И экономит много времени.

Секретное оружие

Как быстро разрабатывать скрипты на JMeter?

Как быстро скачивать расширения и библиотеки?

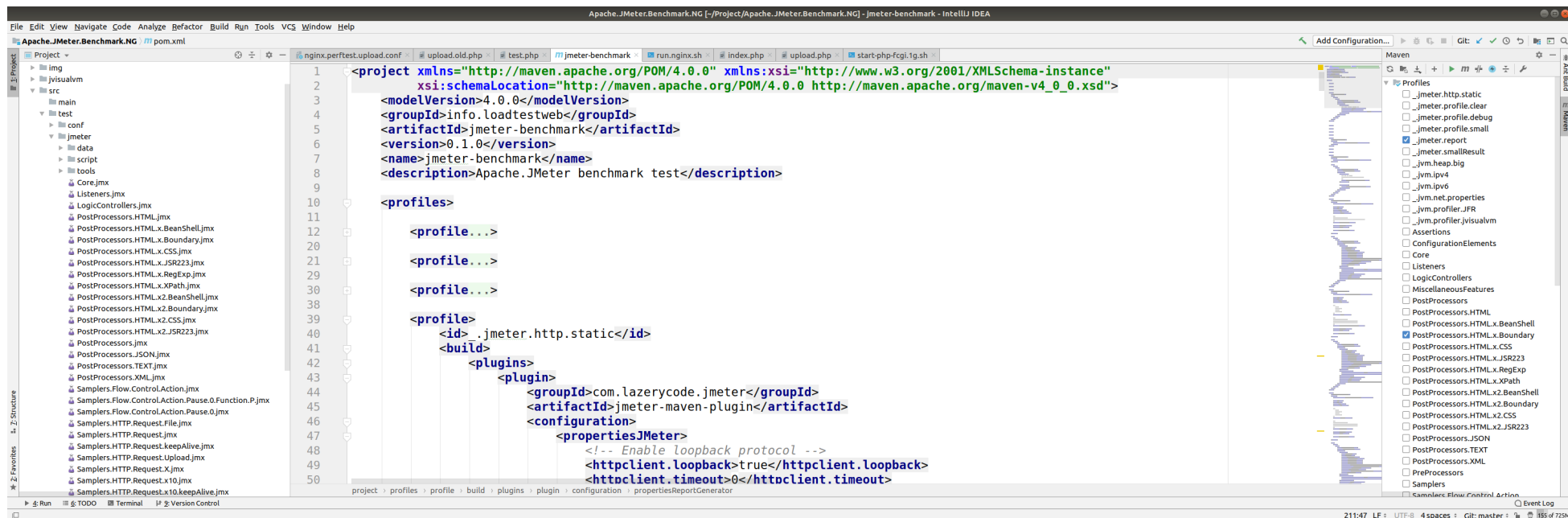
Как запустить профилирование теста?

Как не терять изменения в скрипте?

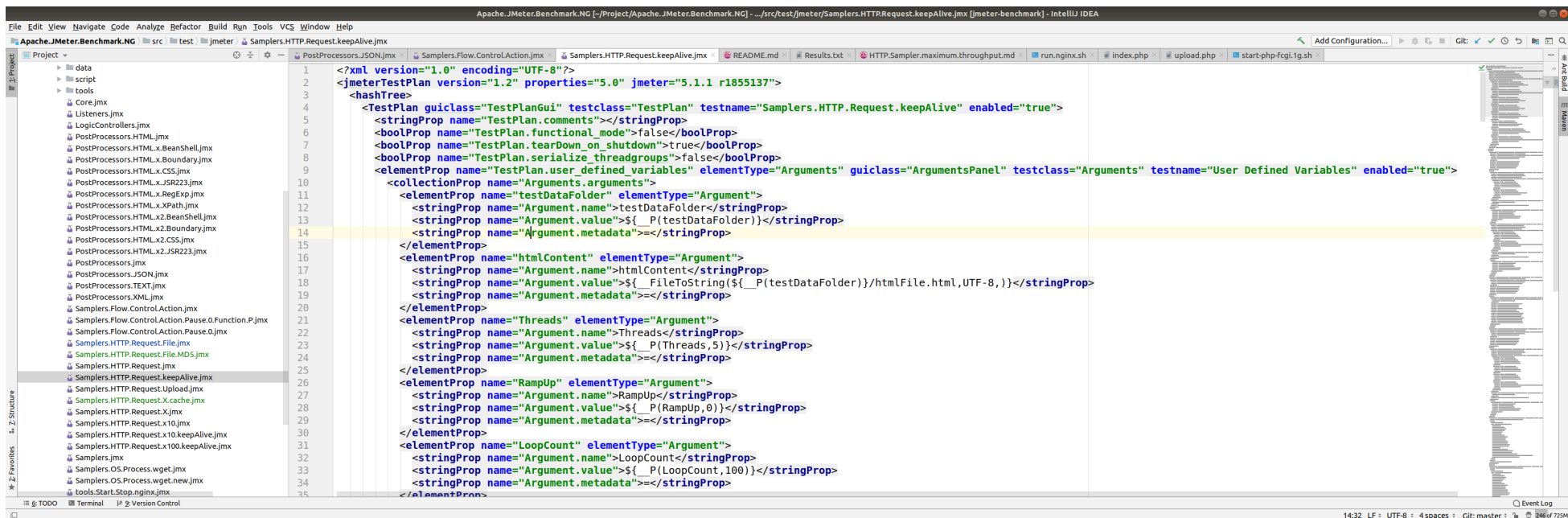
Maven, jmeter-maven-plugin и IntelliJ IDEA

150

Ускорение работы в 10 раз, не меньше



Подсветка синтаксиса, горячие клавиши, git



Плагины хранятся в git (не бинарники)

```
<jmeterExtensions>
```

```
  <!-- Custom Thread Groups
```

```
  https://jmeter-plugins.org/?search=jpgc-casutg
```

```
  http://repo.maven.apache.org/maven2/kg/apc/jmeter-plugins-casutg/
```

```
  -->
```

```
  <artifact>kg.apc:jmeter-plugins-casutg:2.6</artifact>
```

```
  <!-- Parallel Controller & Sampler
```

```
  https://jmeter-plugins.org/?search=parallel
```

```
  https://github.com/Blazemeter/jmeter-bzm-plugins/blob/master/parallel/Parallel.md
```

```
  https://www.blazemeter.com/blog/how-to-use-the-parallel-controller-in-jmeter
```

```
  -->
```

```
  <artifact>com.blazemeter:jmeter-parallel:0.9</artifact>
```

```
</jmeterExtensions>
```

Библиотеки хранятся в git (не бинарники)

```
<testPlanLibraries>
```

```
  <!-- JMeter Plugins Common Classes: Various utility classes to ease development of plugins  
  http://repo.maven.apache.org/maven2/kg/apc/jmeter-plugins-cmn-jmeter/  
  -->
```

```
  <artifact>kg.apc:jmeter-plugins-cmn-jmeter:0.6</artifact>
```

```
  <!-- InfluxDB java client + dependencies -->
```

```
  <artifact>org.influxdb:influxdb-java:2.15</artifact>
```

```
  <artifact>com.squareup.okhttp3:okhttp:3.13.1</artifact>
```

```
  <artifact>com.squareup.okhttp3:logging-interceptor:3.13.1</artifact>
```

```
  <artifact>com.squareup.retrofit2:retrofit:2.5.0</artifact>
```

```
  <artifact>com.squareup.retrofit2:converter-moshi:2.5.0</artifact>
```

```
  <artifact>org.msgpack:msgpack-core:0.8.16</artifact>
```

```
  <artifact>com.squareup.okio:okio:1.17.2</artifact>
```

```
  <artifact>com.squareup.moshi:moshi:1.8.0</artifact>
```

```
</testPlanLibraries>
```

Зависимости скачиваются сами

```
<testPlanLibraries>
  <!-- JMeter Plugins Common Classes: Various utility classes to ease development of plugins
  http://repo.maven.apache.org/maven2/kg/apc/jmeter-plugins-cmn-jmeter/
  -->
  <artifact>kg.apc:jmeter-plugins-cmn-jmeter:0.6</artifact>

  <!-- InfluxDB java client + dependencies -->
  <artifact>org.influxdb:influxdb-java:2.15</artifact>
</testPlanLibraries>

<downloadExtensionDependencies>>false</downloadExtensionDependencies>
<downloadLibraryDependencies>true</downloadLibraryDependencies>
<downloadOptionalDependencies>>false</downloadOptionalDependencies>
<downloadJMeterDependencies>>false</downloadJMeterDependencies>
```

Любые параметры модифицируются

Создадим файл на 1 ГБайт и скачаем его 200 раз

```
01. dd if=/dev/urandom of=/tmp/data/1g.img \  
02.     bs=1 count=0 seek=1G  
03. mvn jmeter:jmeter \  
04.     -P Samplers.HTTP.Request.File \  
05.     -DThreads=1 -DLoopCount=200 -DRequestCount=1 \  
06.     -DRequestPath=/tmp/data/1g.img
```

Профили комбинируются между собой

156

Выполним тестирование с большим размером Heap

```
01. dd if=/dev/urandom of=/tmp/data/1g.img \  
02.     bs=1 count=0 seek=1G \  
03. mvn jmeter:jmeter \  
04.     -P Samplers.HTTP.Request.File,_.jvm.heap.big \  
05.     -DThreads=1 -DLoopCount=200 -DRequestCount=1 \  
06.     -DRequestPath=/tmp/data/1g.img
```

И через 10 минут готов отчёт

01. # Запуск sjk-профайлера

02. `mvn exec:exec@sjk`

01. # Запуск профилируемого теста

02. `mvn jmeter:jmeter \`

03. `-P Samplers.HTTP.Request.File, \`

04. `-DThreads=1 -DLoopCount=200 -DRequestCount=1 \`

05. `-DRequestPath=/tmp/data/1g.img`

Профилирование Java Flight Recorder в одну строку

И через 10 минут готов отчёт

```
01. # Запуск профилируемого теста
02. mvn jmeter:jmeter \
03.     -P Samplers.HTTP.Request.File,_.jvm.profiler.JFR \
04.     -DThreads=1 -DLoopCount=200 -DRequestCount=1 \
05.     -DRequestPath=/tmp/data/1g.img
```

И через 10 минут готов отчёт

```
01. # Запуск профилируемого теста
02. mvn jmeter:jmeter \
03.     -P Samplers.HTTP.Request.File,_.jvm.profiler.jvisualvm \
04.     -DThreads=1 -DLoopCount=200 -DRequestCount=1 \
05.     -DRequestPath=/tmp/data/1g.img
```


github.com/polarnik/Apache.JMeter.Benchmark.NG

```
<?xml version="1.0" encoding="UTF-8" ?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>info.loadtestweb</groupId>
    <artifactId>jmeter-benchmark</artifactId>
    <version>0.1.0</version>
    <name>jmeter-benchmark</name>
    <description>Apache.JMeter benchmark test</description>

    <profiles>
        <profile...>
        <profile...>
        <profile...>

        <profile>
            <id>.jmeter.http.static</id>
            <build>
                <plugins>
                    <plugin>
                        <groupId>com.lazerycode.jmeter</groupId>
                        <artifactId>jmeter-maven-plugin</artifactId>
                        <configuration>
                            <propertiesJMeter>
                                <!-- Enable loopback protocol -->
                                <httpClient.loopback>true</httpClient.loopback>
                                <httpClient.timeout>0</httpClient.timeout>
                            </propertiesJMeter>
                        </configuration>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
</project>
```

Как ускорить Apache.JMeter

Скрипт.

Настройки Apache.JMeter.

Настройки операционной системы.

Maven, jmeter-maven-plugin, IntelliJ IDEA.



**Райффайзен
БАНК**

Вячеслав Смирнов

github.com/polarnik

Telegram: @qa_load

Спасибо!