Ускоряем Apache.JMeter

> «Ты можешь быстрее. Предела нет. Знай: ты можешь! Будь уверен.» (Морфеус, «Матрица»).

О себе

Вячеслав Смирнов

Эксперт по тестированию в Райффайзенбанк

Занимаюсь тестированием производительности

Читаю почту:

✉️ owasp@yandex.ru

Публикую код:

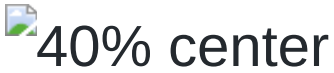
🔗 https://github.com/polarnik/

<h1 class="test">Apache.JMeter можно и нужно ускорять</h1>

В этой презентации

- Когда нужна оптимизация
- Известные рекомендации
- Производительность стандартных компонентов
    - результаты исследования

Варианты подачи нагрузки из JMeter


40% center

Варианты подачи нагрузки из JMeter

По частоте использования выделю (субъективно) следующие варианты подачи нагрузки из `Apache.JMeter`:

- Встроенные компоненты
- Плагины
- Библиотеки
  - `JSR223 Sampler` и дополнительные библиотеки
  - `JUnit Sampler` и дополнительные библиотеки
- Внешние приложения
  - через `OS Process Sampler`
- Внешние системы
  - по отношению к которым `Apache.JMeter` выступает, как контрольный центр

Варианты подачи нагрузки из JMeter

Прежде всего поговорим о производительности встроенных компонентов и популярных плагинов для `Apache.JMeter`:

- **Встроенные компоненты**
- **Плагины**
- Библиотеки
  - `JSR223 Sampler` и дополнительные библиотеки
  - `JUnit Sampler` и дополнительные библиотеки
- Внешние приложения
  - через `OS Process Sampler`
- Внешние системы
  - по отношению к которым `Apache.JMeter` выступает, как контрольный центр

С чего всё начинается

- недовес:
  - нужно 1000 сценариев в минуту
  - видим 800 сценариев в минуту
- перегруз:
  - нужно уложиться в 4 ГБайта ОЗУ
  - видим `java.lang.OutOfMemoryError: Java heap space`

И есть скрипт на `Apache.JMeter`, и, возможно, причина в нём.

Некоторые причины недовеса и перегруза

- Профиль нагрузки:
  - расчитан неверно
- Ограничения:
  - выставленные ограничения не позволяют использовать доступные системные ресурсы
- Ошибки:
  - ошибки компонентов и скрипта не позволяют нагрузить систему
- Неоптимальное использование компонентов:
  - неоптималные настройки `Apache.JMeter` или неоптимальное использование компонентов

Некоторые причины недовеса и перегруза

- Профиль нагрузки:
  - ошибка расчёта профиля нагрузки или шага нагрузки
- Ограничения:
  - сработали ограничения JVM или cgroups
- Ошибки:
  - ошибки `Apache.JMeter` или разработки скрипта
- Неоптимальное использование компонентов:
  - недостаточно оперативной памяти
  - недостаточно ресурсов процессора
  - недостаточно соединений

Что будем делать?

Известные рекомендации

- Будет корректно рассчитывать профиль нагрузки
- Понимая ограничения JVM, cgroups, системы
- Обходя и не допуская ошибок

Неизвестные рекомендации

- Исследуем
  - потребление памяти компонентами `Apache.JMeter`
  - потребление ресурсов процессора компонентами
  - общее влияние компонентов `Apache.JMeter` на производительность
- Выберем подходы для типовых задач

Когда нужна оптимизация

Опимизация оправдана, когда:

- Применены известные (очевидные) рекомендации
  - Профиль нагрузки известен и расчитан корректно
  - Известны и настроены ограничения JVM, cgroups, системы
  - Скрипт написан без ошибок (без функциональных ошибок)

> Сначала нужно делать то, что нужно делать с начала
>
> (капитан)

Профиль нагрузки

- Закрытая модель нагрузки (RPS и потоки ограничены сверху)
- Шаг нагрузки расчитан верно, потоки расчитаны верно
- Зависающие запросы прерываются
- Ошибочные тесты прерываются

Ограничения JVM и системы

- операционная система имеет ограничения и настройки
  - в операционной системе выполняются процессы
- для процессов есть ограничения и настройки
  - java-машина выполняет Apache.JMeter и его компоненты
  - некоторые компоненты Apache.JMeter запускают новые процессы
- для JVM есть ограничения и настройки
  - настройки размера кучи

Ошибка расчёта профиля нагрузки

Thread Group и одноразовые пользователи без таймеров

Дано:

- нужна интенсивность 16 в секунду

Ошибиться можно с математикой или пониманием

- Thread Properties
  - Number of Threads (users): 16000
  - Ramp-Up Period (in seconds): 1000

> Здесь нет пула пользователей, пул неограниченно большой. Чем медленее система отвечает, тем большее количество потоков будут ждать ответа от сервера.

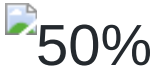
Ошибка расчёта профиля нагрузки

Thread Group и бесконечно много потоков

```
summary +    498 in 00:00:30 =   16.6/s Avg:     0 Min:     0 Max:     2 Err:     0 (0.00%) Active: 3 Started: 389 Finished: 386
summary =    773 in 00:01:02 =   12.4/s Avg:     0 Min:     0 Max:    12 Err:     0 (0.00%)
summary +    738 in 00:00:30 =   24.6/s Avg:     0 Min:     0 Max:    19 Err:     0 (0.00%) Active: 2 Started: 757 Finished: 755
summary =   1511 in 00:01:32 =   16.4/s Avg:     0 Min:     0 Max:    19 Err:     0 (0.00%)
summary +    976 in 00:00:30 =   32.6/s Avg:     0 Min:     0 Max:    17 Err:     0 (0.00%) Active: 2 Started: 1245 Finished: 1243
summary =   2487 in 00:02:02 =   20.3/s Avg:     0 Min:     0 Max:    19 Err:     0 (0.00%)
summary +   1214 in 00:00:30 =   40.4/s Avg:     0 Min:     0 Max:     1 Err:     0 (0.00%) Active: 4 Started: 1854 Finished: 1850
summary =   3701 in 00:02:32 =   24.3/s Avg:     0 Min:     0 Max:    19 Err:     0 (0.00%)
summary +   1454 in 00:00:30 =   48.5/s Avg:     0 Min:     0 Max:     2 Err:     0 (0.00%) Active: 3 Started: 2580 Finished: 2577
summary =   5155 in 00:03:02 =   28.3/s Avg:     0 Min:     0 Max:    19 Err:     0 (0.00%)
summary +   1684 in 00:00:30 =   56.1/s Avg:     0 Min:     0 Max:     2 Err:     0 (0.00%) Active: 8 Started: 3427 Finished: 3419
summary =   6839 in 00:03:32 =   32.2/s Avg:     0 Min:     0 Max:    19 Err:     0 (0.00%)
summary +    488 in 00:00:30 =   16.1/s Avg:     2 Min:     0 Max:   361 Err:     0 (0.00%) Active: 155 Started: 3818 Finished: 3663
summary =   7327 in 00:04:03 =   30.2/s Avg:     0 Min:     0 Max:   361 Err:     0 (0.00%)
java.lang.OutOfMemoryError: Java heap space
Dumping heap to java_pid21386.hprof ...
```

TransactionsPerSecond

На седьмой ступени при интенсивности 28 транзакций в сек всё было хорошо, а дальше `OutOfMemoryError`:


50%

Ошибка выставления шага нагрузки

> Использовать пул потоков и шаг нагрузки надежнее, надо только расчитать верно

https://loadtestweb.wordpress.com/2017/08/23/pacing/

Известные рекомендации

Производительность стандартных компонентов

Постпроцессоры

- Regular Expression Extractor
- CSS Selector Extractor (was: CSS/JQuery Extractor)
- XPath2 Extractor
- XPath Extractor
- Result Status Action Handler
- BeanShell PostProcessor
- JSR223 PostProcessor
- JDBC PostProcessor
- JSON Extractor
- Boundary Extractor
- Debug PostProcessor

http://jmeter.apache.org/usermanual/component_reference.html

Постпроцессоры для извлечения значений из ответа

Для HTML

- Regular Expression Extractor
- CSS Selector Extractor (was: CSS/JQuery Extractor)
- XPath2 Extractor
- XPath Extractor
- BeanShell PostProcessor
- JSR223 PostProcessor
- Boundary Extractor

Постпроцессоры для извлечения значений из ответа

Для XML

- Regular Expression Extractor
- XPath2 Extractor
- XPath Extractor
- BeanShell PostProcessor
- JSR223 PostProcessor
- Boundary Extractor

Постпроцессоры для извлечения значений из ответа

Для JSON

- Regular Expression Extractor
- BeanShell PostProcessor
- JSR223 PostProcessor
- Boundary Extractor

Известные проблемы производительности

Решение долгих и больших задач

Отправка больших HTTP-запросов

Получение больших HTTP-ответов

Спасибо за внимание!

Вячеслав Смирнов

Эксперт по тестированию в Райффайзенбанк

owasp@yandex.ru
https://github.com/polarnik/
@SmirnovQA