Результаты тестирования серверной части приложения Petstore API

Цель проведения тестирования: Проанализировать возможные уязвимые места и проблемы, выполняя тесты под разной нагрузкой

Конфигурация теста:

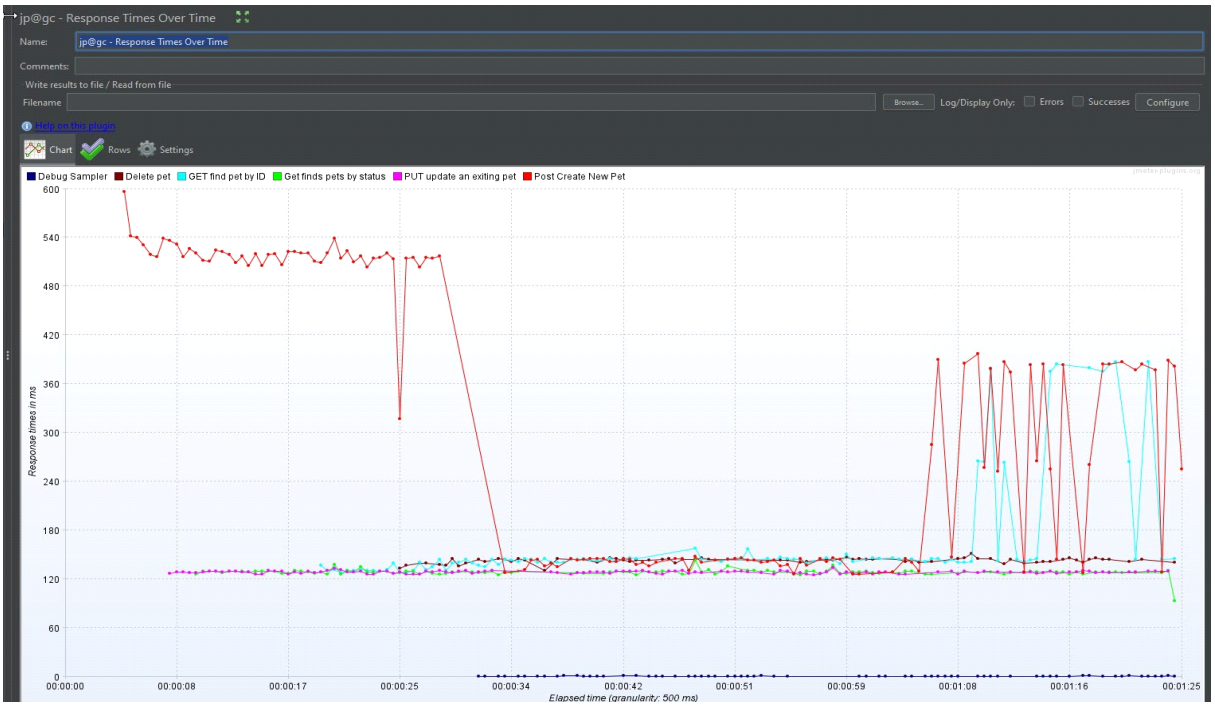
Thread Group:

1. Количество пользователей (Numbers pf Threads): 50
2. В ремя, за которое все потоки будут постепенно запущены (Ramp-up preriod) seconds: 25
3. Включен бесконечный цикл выполнения запросов каждым потоком (Loop Count: Infinity)

Список запросов: POST, PUT, GET, DELETE

Общий анализ графиков и результатов

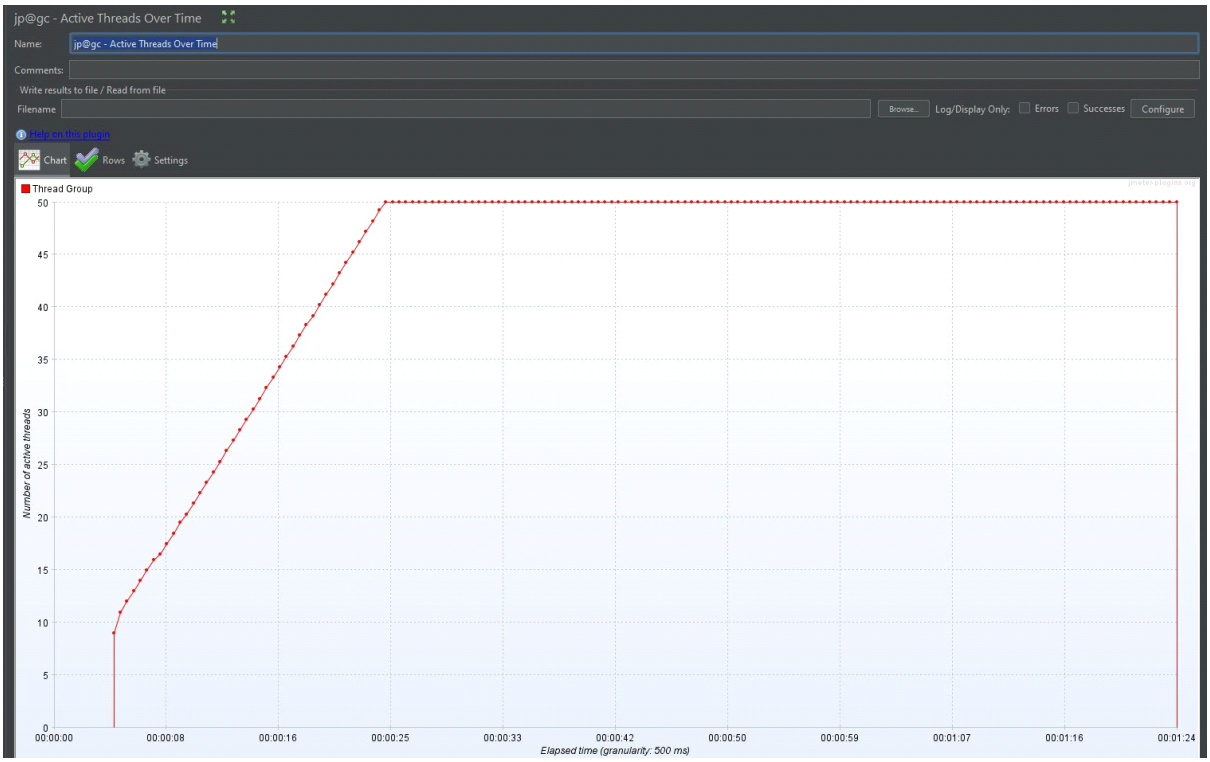
1. *Response Times Over Time*



- Запросы POST (Create New Pet) — имели наибольшее время отклика: до 600 мс, но затем стабилизировались.

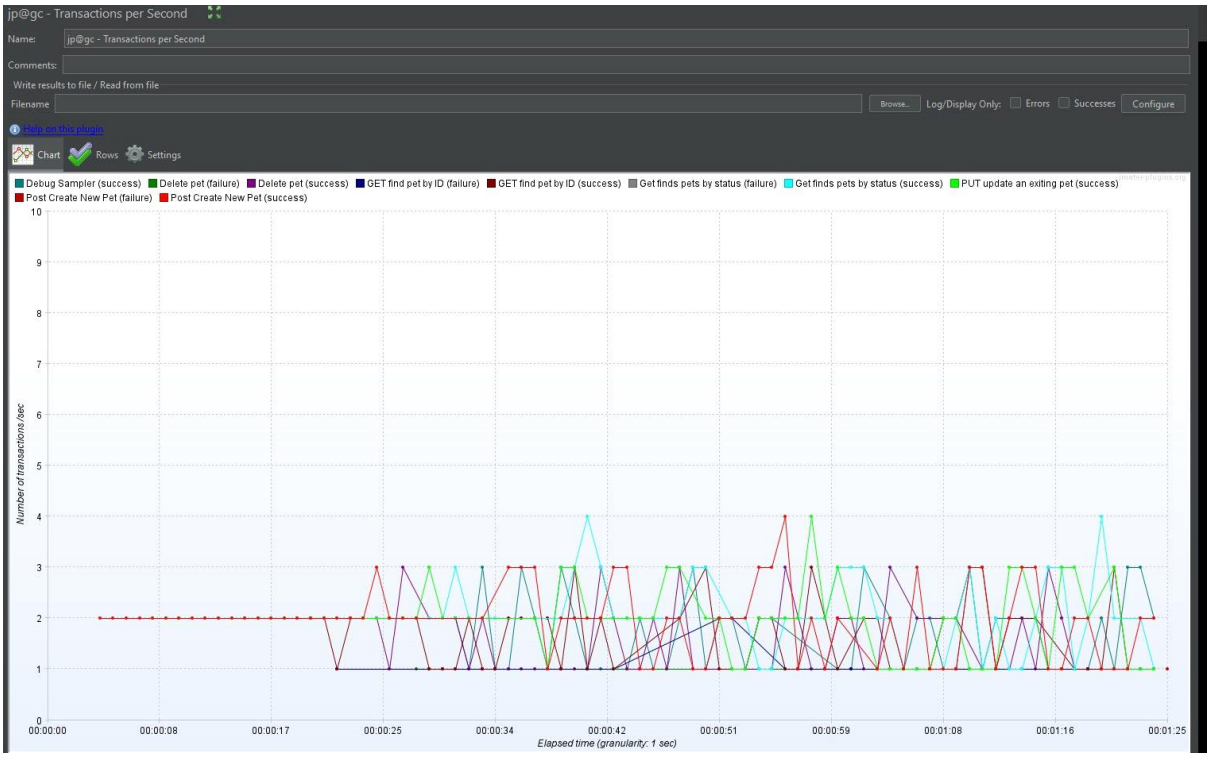
- GET, PUT, DELETE — в среднем ~130–150 мс.
- В середине теста видно, как увеличивалась нагрузка, но отклики оставались стабильными, что говорит о хорошей устойчивости API.

2. *Active Threads Over Time*



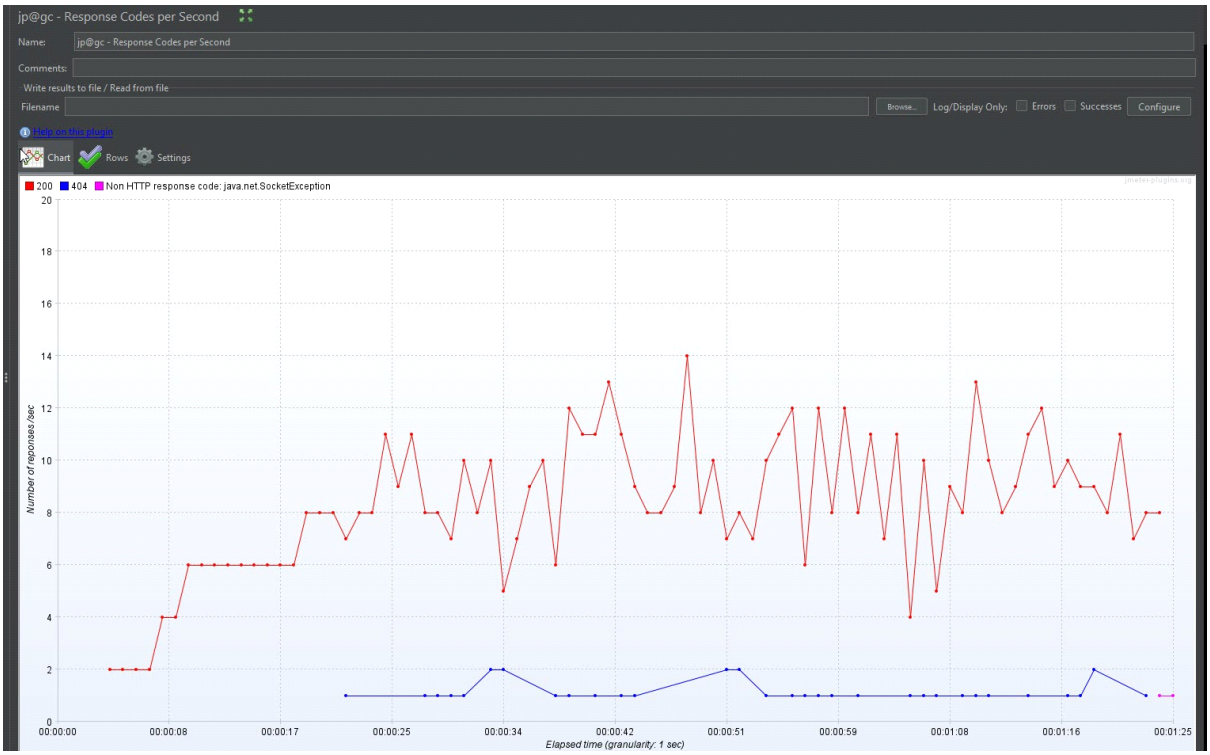
- Потоки запускались плавно: все 50 были активны примерно через 25 секунд, как и задано в Ramp-up.
- После этого все 50 потоков одновременно работали, что подтверждает корректную конфигурацию Thread Group.

3. Transactions per Second



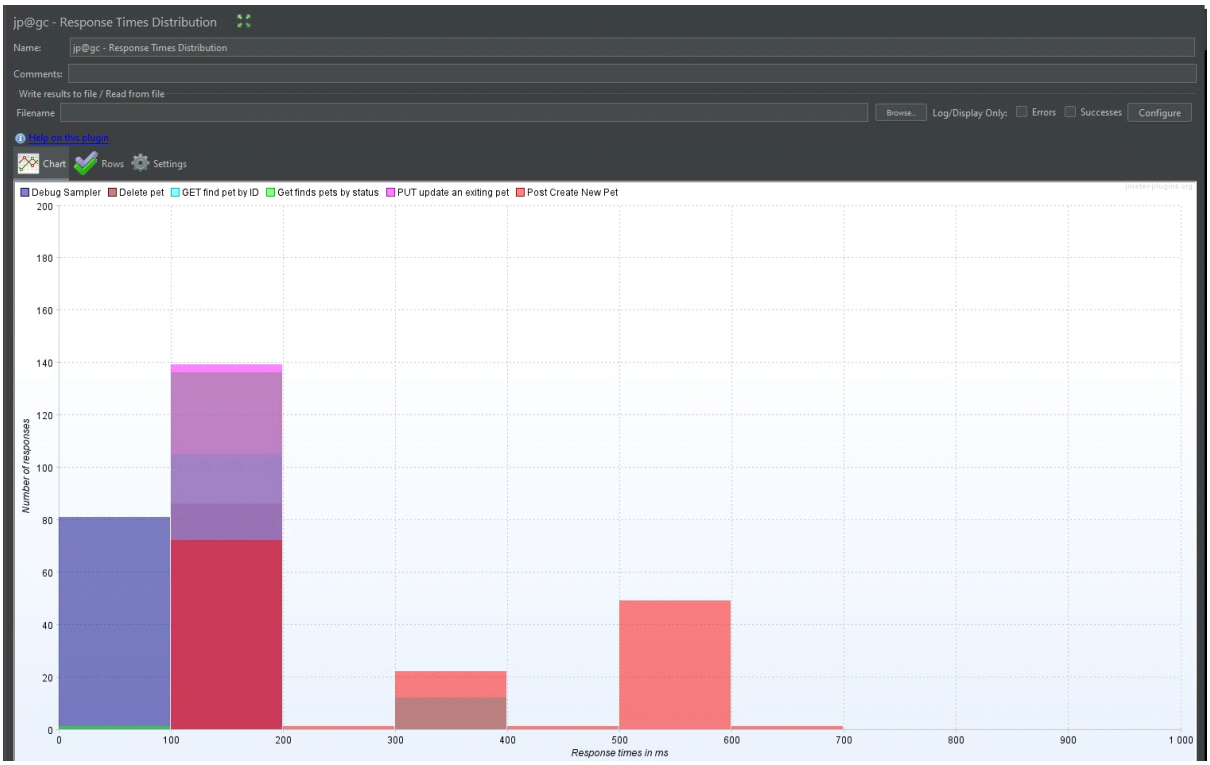
- Все типы запросов успешно выполнялись, в среднем 1–3 транзакции в секунду на каждый тип.
- Видны пики активности, что указывает на естественные колебания нагрузки.
- Появление ошибок на некоторых GET и DELETE связано с удалением питомца до получения — API логика, не баг.

4. *Response Codes per Second*



- Основной ответ: 200 OK — почти весь поток.
- Иногда появлялся 404 Not Found — это ожидаемо, когда GET запрашивает уже удалённого питомца.
- Один раз зафиксирован java.net.SocketException — скорее всего, кратковременное соединение не успело установиться из-за нагрузки.

5. *Response Times DistributionResponse Times Distribution*



- Большинство откликов укладываются в диапазон:

A. 0–200 мс для GET, PUT, DELETE

B. 500–700 мс — для POST (создание новых питомцев)

- Это показывает высокую производительность сервиса при текущей нагрузке.

Возможные узкие места:

- Создание новых питомцев (POST) — самый медленный запрос. При увеличении количества пользователей он может стать бутылочным горлышком.
- При высоких нагрузках (50+ потоков) возможны конфликты удаления и чтения: DELETE → GET → 404.

Рекомендации:

- Для продакшн-тестов — добавить Random Timer, чтобы потоковая нагрузка была менее синхронной.
- Разделить GET и DELETE во времени, чтобы исключить неправильный порядок выполнения запросов
- Провести повторный тест в разное время суток, чтобы исключить временное влияние

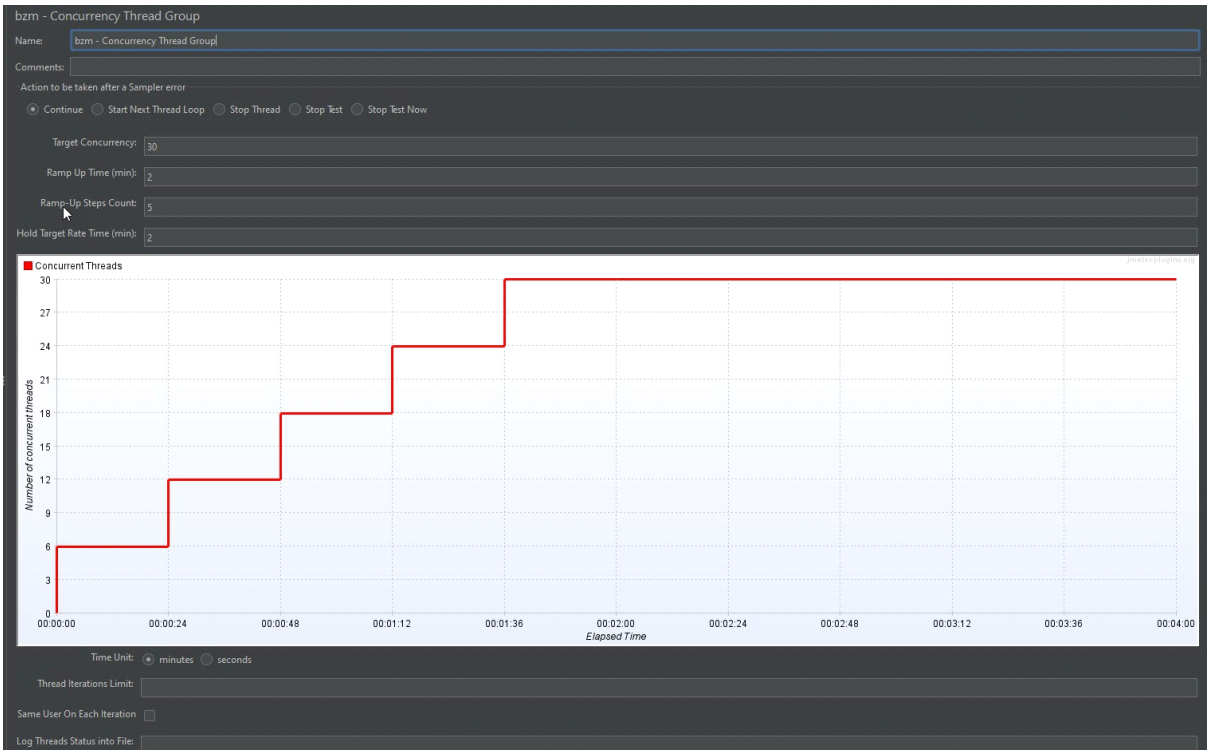
внешней нагрузки.

Concurrency Thread Group

Конфигурация:

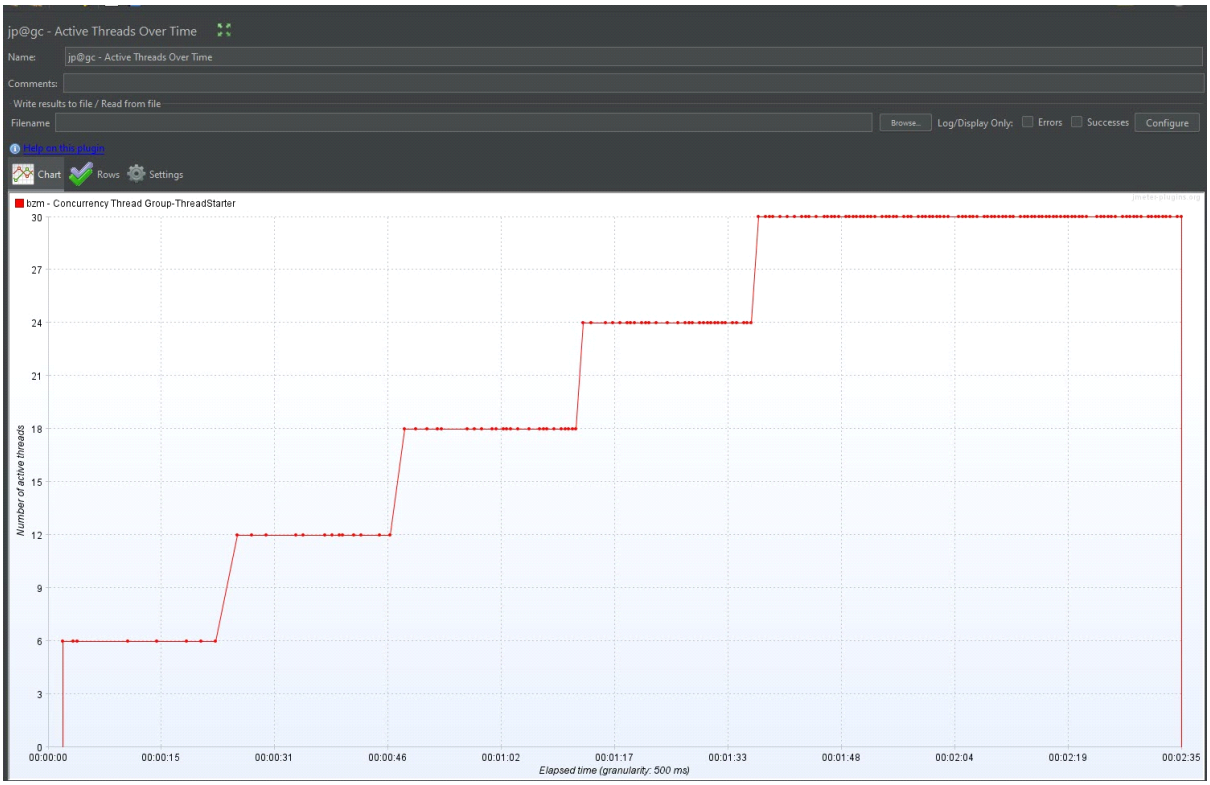
- Target Concurrency = 30
- Ramp Up Time = 2 min
- Ramp-Up Steps Count = 5

График (ожидаемый) — построен в настройках Concurrency Thread Group:



Ожидание: каждую ~24 секунды прибавляется ~6 пользователей, всего 5 ступенек по 6 потоков, пока не наберётся 30.

График (фактический результат):



- Всё происходит ровно как запланировано:
- Потоки добавляются ступенчато: 6 → 12 → 18 → 24 → 30
- Последние ~2 минуты поддерживается пиковая нагрузка из 30 пользователей (ты задала "Hold Target Rate Time = 2 мин").

Выводы:

- План загрузки реализован корректно, JMeter правильно увеличивал число пользователей по шагам.
- Это говорит о том, что настройка нагрузки стабильна.

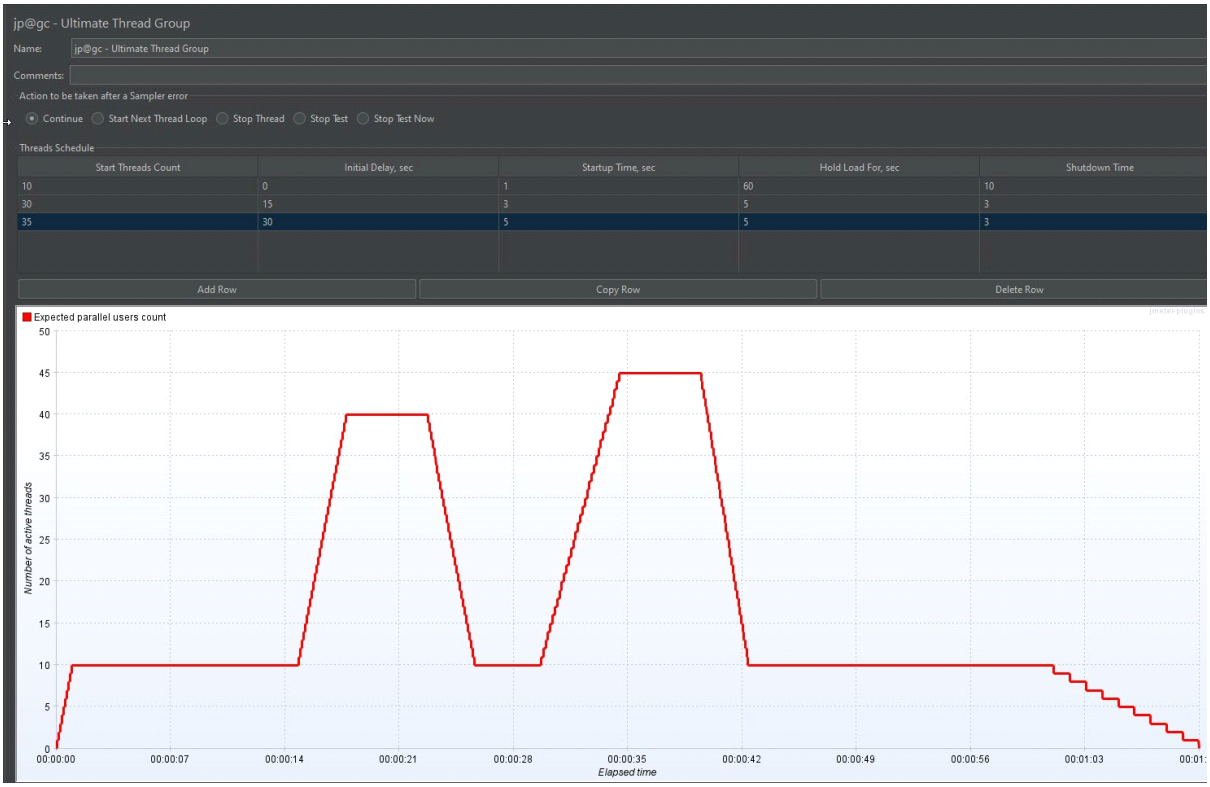
Ultimate Thread Group

Конфигурайции:

- постепенное увеличение числа потоков с 10 до 40;
- удержание пикового значения 40 потоков;
- резкое снижение до 10 и повторное увеличение до 45;

- снова падение до 10 и постепенное завершение.

График (ожидаемый)



Ожидаемый план нагрузки:

- 10 пользователей сначала,
- затем добавляются 30, потом 35;
- каждый этап стартует с задержкой и держится заданное время.

График (фактический результат):

На реальном графике видно:

1. Общая форма графика соответствует плану нагрузки, заданному в Ultimate Thread Group:

- Видны три чётко выраженные ступени с ростом и спадом количества пользователей.
- Нагрузка держалась стабильно в нужные промежутки времени, что говорит о корректной работе Thread Schedule.

2. Пиковое значение пользователей достигнуто, но с небольшим смещением по времени:

- Это может указывать на небольшую задержку в запуске потоков — вполне допустимо, особенно если машина не высокопроизводительная.
- Потоки, возможно, стартуют с небольшой задержкой из-за ресурсов или особенностей выполнения скриптов.

3. Резкие спады после пиков:

- Видно, что после фазы нагрузки потоки быстро завершаются — это подтверждает корректную работу параметров Shutdown Time.
- Нет «зависших» пользователей, что говорит о стабильном завершении всех сценариев.

4. Низкие значения до старта и после окончания пиков:

- Начальные и конечные участки графика на уровне 10 потоков соответствуют настройкам

первых строк сценария.

- Это подтверждает, что начальный и финальный этапы нагрузки работают как задуман

Выводы:

Тестовый сценарий с Ultimate Thread Group работает стабильно и хорошо воспроизводит запланированную нагрузку. Незначительные отклонения по времени не критичны и скорее всего вызваны особенностями работы JMeter и нагрузкой на машину.