Автоматизация тестирования на Python

Юрьева Ольга
Константин Волков

образование

Не забудьте отметиться на портале

Нагрузочное тестирование

образование

Виды тестирования:

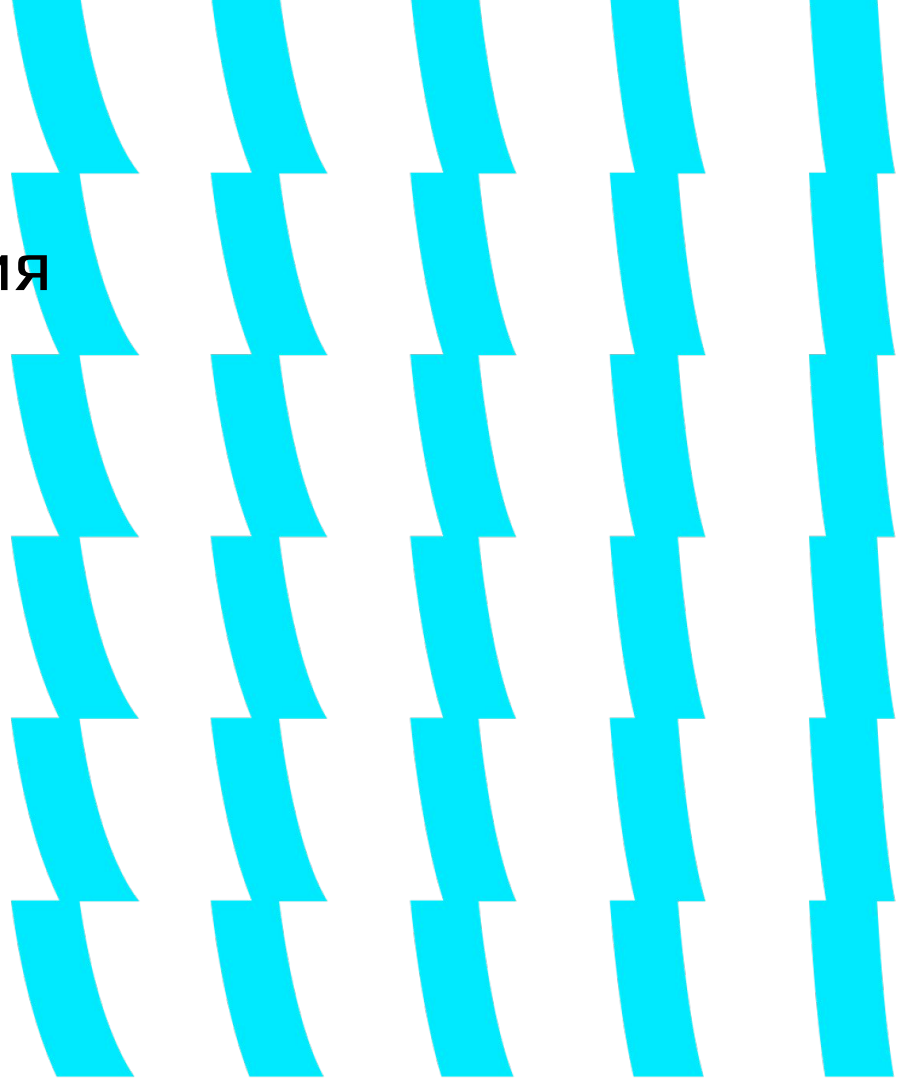
- Функциональное тестирование
- Тестирование безопасности
- Тестирование взаимодействия (интеграционное)

- Тестирование производительности:
  - нагрузочное тестирование
  - стрессовое тестирование
  - тестирование стабильности или надежности
  - объемное тестирование
- Тестирование установки
- Тестирование удобства пользования
- Тестирование на отказ и восстановление
- Конфигурационное тестирование
- Тестирование безопасности

Цели нагрузочного тестирования:

- Оценка производительности и работоспособности (времена отклика)
- Оптимизация производительности приложения/базы данных
- Подбор соответствующей для данного приложения аппаратной конфигурации сервера

Этапы проведения нагрузочного тестирования



Анализ требований:

- Время отклика
- Интенсивность (число запросов в секунду)
- Используемые ресурсы: загрузка процессора, кол-во используемой памяти, дисковое и сетевой I/O
- Максимальное количество пользователей
- Бизнес метрики: количество операций в единицу времени и время выполнения бизнес операции (SLA)

Определение профиля нагрузки:

Профиль это набор операций и интенсивность их выполнения для каждой пользовательской группы.

Необходимо определить:

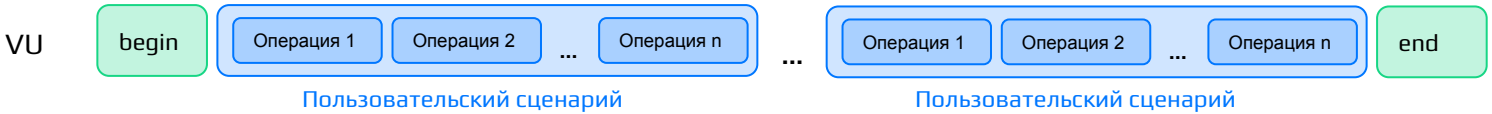
- список тестируемых операций критичных для производительности и для бизнеса
- интенсивность выполнения операций

Разработка моделей нагрузки:

Моделью нагрузки является набор профилей нагрузки, где каждый профиль отличается от другого или набором операций или интенсивностями выполнения этих операций.

Подходы:

- на основе потоков: VU (виртуальный пользователь, программный процесс, эмулирующий действия физического пользователя) циклически производит выполнение пользовательского сценария

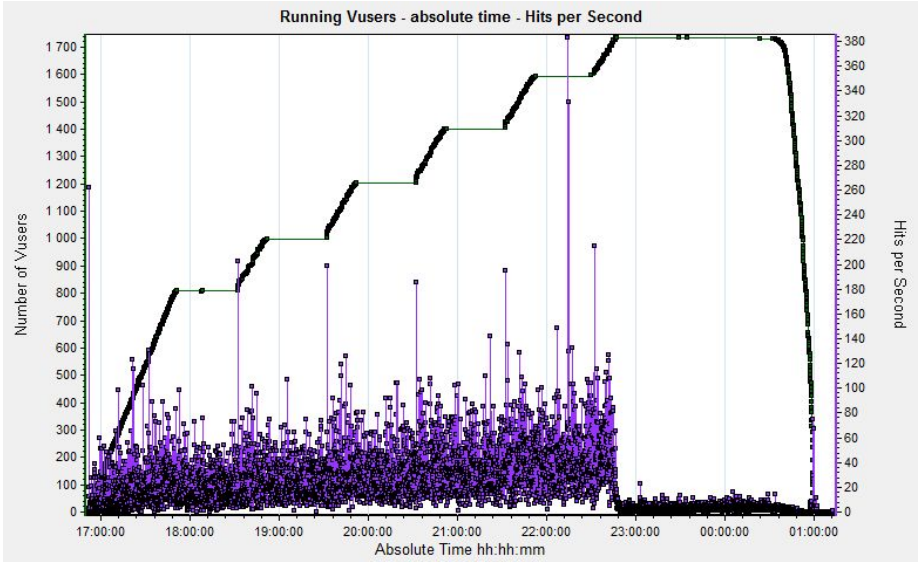


- на основе запросов в секунду: RPS (асинхронный подход, при котором не дожидаемся ответа от предыдущего запроса)

Определение максимальной производительности

Постепенное увеличение нагрузки, добавляя новых пользователей с некоторым интервалом времени, позволяет нам определить:

- измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций
- количество пользователей, способных одновременно работать с приложением
- границы приемлемой производительности при увеличении нагрузки
- производительность при разных нагрузках



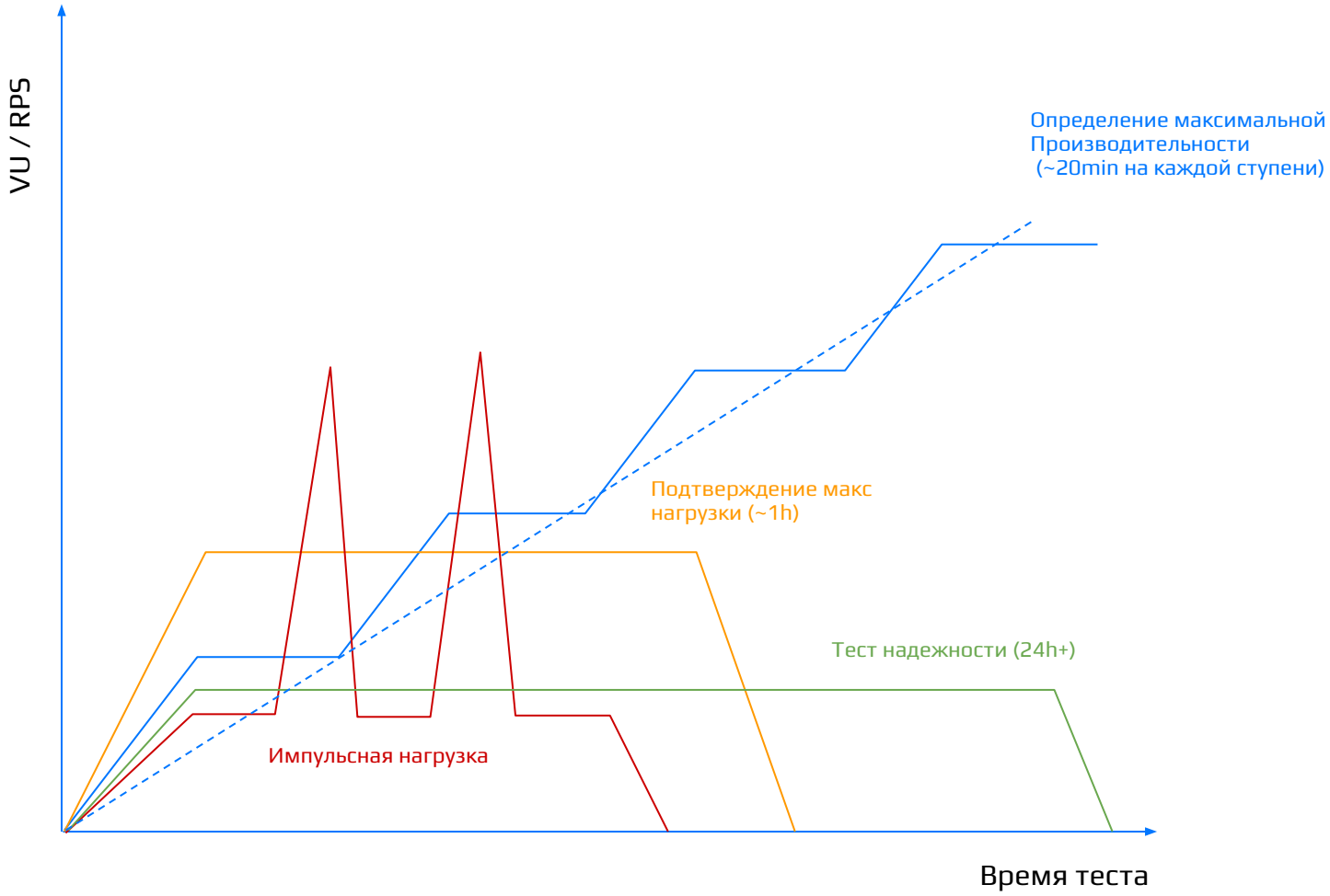
Стрессовое тестирование

Увеличиваем интенсивность операций выше пиковых (максимально разрешенных) значений либо увеличивая количество пользователей, до тех пор пока нагрузка не станет выше максимально допустимых значений, проверяем, что система работоспособна в условиях стресса.

Далее, опустив нагрузку до средних значений, проверяем (способность системы к регенерации), что система вернулась к нормальному состоянию (основные нагрузочные характеристики не превышают базовые).

Тестирование стабильности или надежности

Используя базовый нагрузочный профиль, запускаем тест длительностью от нескольких часов до нескольких дней, с целью выявления утечек памяти, перезапуска серверов и других аспектов влияющих на нагрузку.


VU / RPS
Определение максимальной
Производительности
(~20min на каждой ступени)
Подтверждение макс
нагрузки (~1h)
Тест надежности (24h+)
Импульсная нагрузка
Время теста

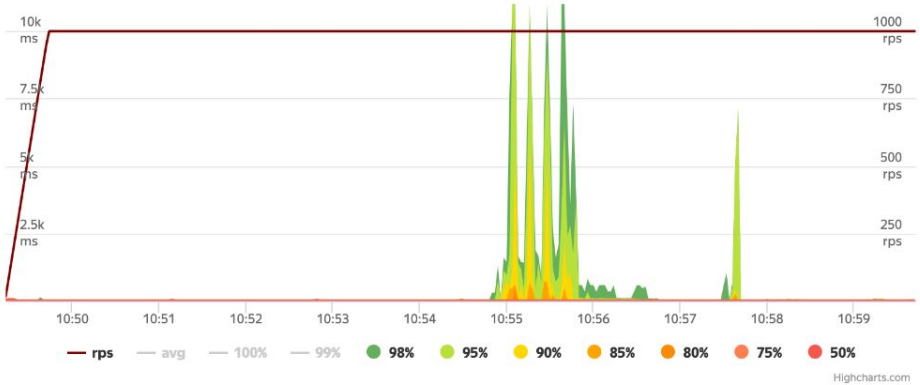
Инструменты для нагрузочного тестирования:

- Apache JMeter (https://jmeter.apache.org/)
- Gatling (https://gatling.io/)
- K6 (https://k6.io/)
- Locust (https://locust.io/)
- MF LoadRunner (https://www.microfocus.com/)
- Yandex.Tank(https://yandextank.readthedocs.io/en/latest/)

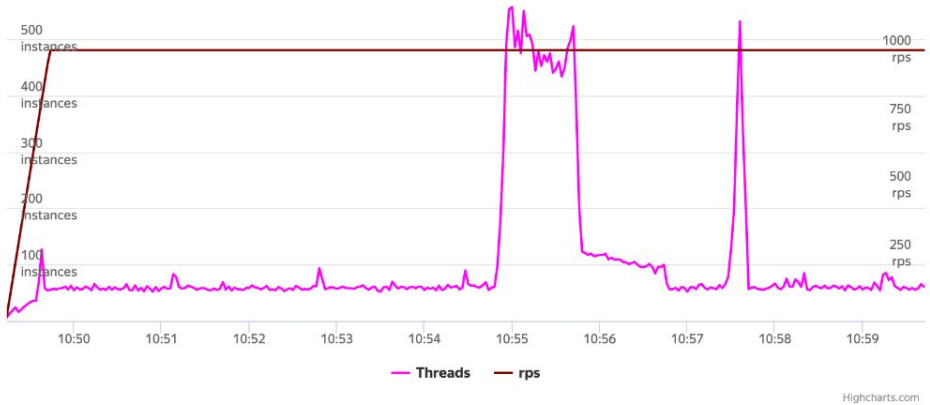
Анализ результатов

Тип	VU	Response Time	Requests	Errors	Throughput
VU	Позволяет сравнить планируемую нагрузку с реальной	Показывает, как увеличение пользователей сказывается на росте времени отклика при закрытой модели	Можно увидеть увеличение RPS/TPS/HITS с увеличением количества пользователей, а также уменьшение в связи с выходом пользователей или стабилизацией подаваемой нагрузки	Показывает, при каком числе VU происходит рост ошибок. Не всегда показателен, так как в открытой модели сложно коррелировать графики	Метрики должны полностью коррелировать, так как рост пользователей приводит к росту объема данных, отправляемых ими. Расхождения или резкие выбросы являются поводами для дальнейшего изучения
Response Time		Показывает, как быстро сервер отвечает на наши запросы. Основная метрика	Показывает среднее время, за которое обрабатываются все транзакции или запросы на всем протяжении теста	Можно увидеть, как увеличение ошибок влияет на рост времени ответа приложения	Рост времени ответа часто связан с увеличением количества отправленных данных (Throughput). Если на графике замечен рост Response Time, а Throughput при этом остается прежним, это указывает на проблемы с сетью или с приложением: где-то начинает копиться очередь запросов
Requests			Показывает, сколько запросов выдерживает система в секунду, минуту и т. д.	Позволяет найти значение интенсивности, после которой появляются ошибки. Удобно для определения SLA	Позволяет контролировать рост объема данных и количества запросов. Эти метрики должны коррелировать. Появление проблем может свидетельствовать, что мы отправляем неверные запросы

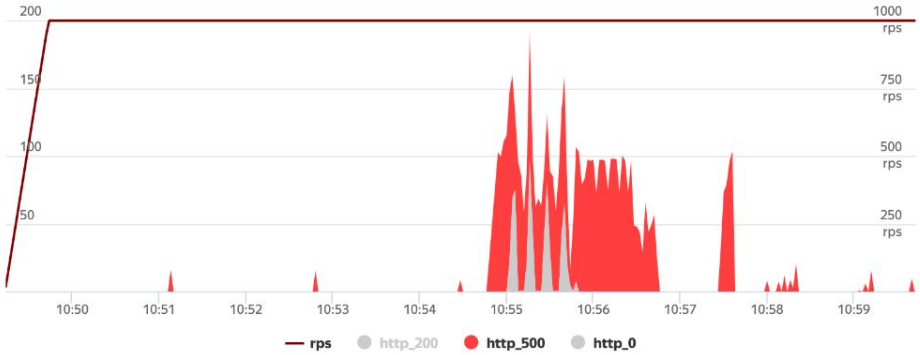
Quantiles



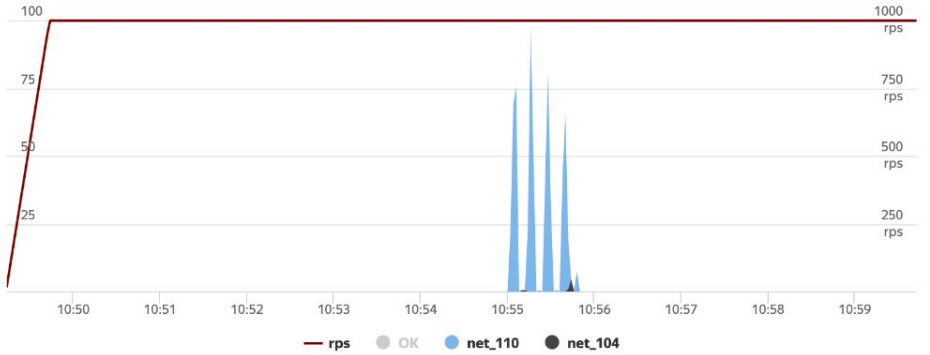
Threads



HTTP codes



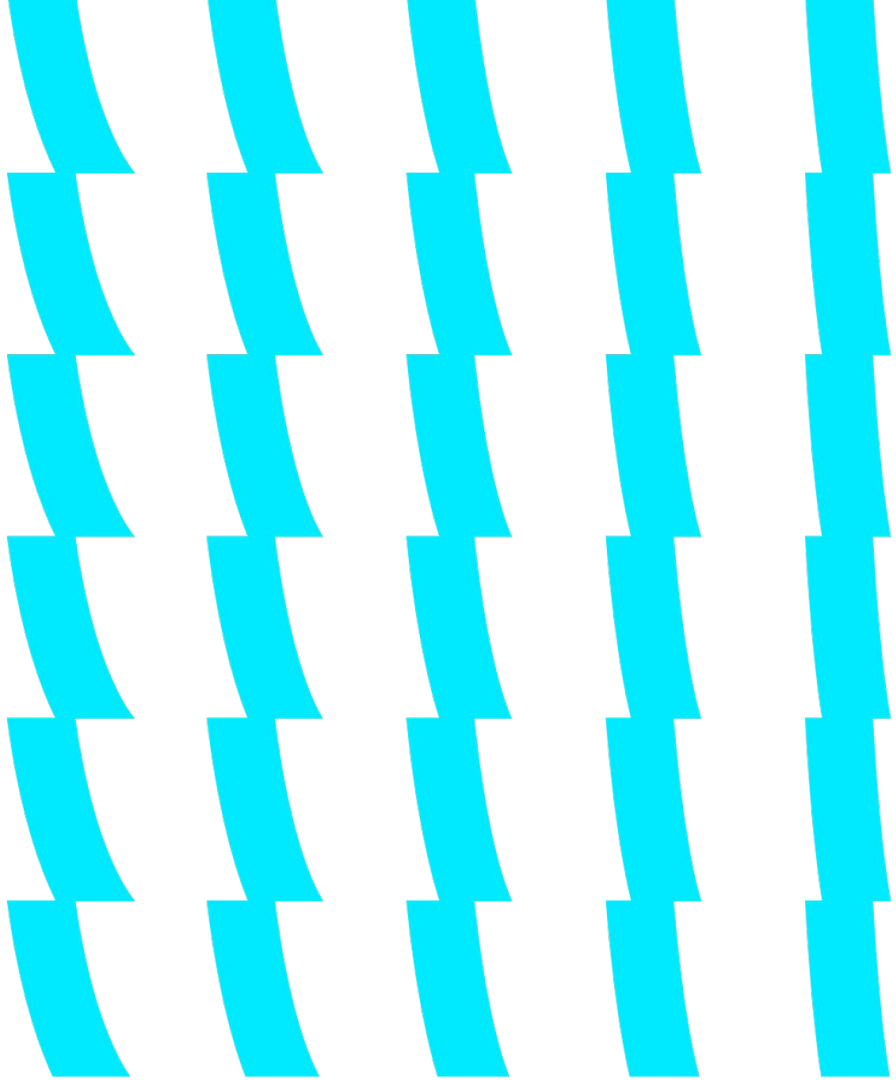
Net codes



https://overload.yandex.net/111#tab=test_data&tags=&plot_groups=extended&machines=8&metrics=&slider_start=1471258146&slider_end=1471258777&compress_ratio=1

Большое спасибо!

Буду ждать ваших отзывов



Что почитать:

https://www.performance-lab.ru/blog/luchshie-instrumenty-dlya-nagruzochnogo-testirovaniya

https://habr.com/ru/post/649295/

https://habr.com/ru/company/tinkoff/blog/514314/

https://habr.com/ru/company/jetinfosystems/blog/470726/