Тестування, верифікація та валідація програмного забезпечення

Лекція №7

Тема: Тестування продуктивності та надійності програмного забезпечення

ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Лектор: асистент кафедри комп'ютерних наук Українець Микола Олександрович

Питання лекції

1. Тестування продуктивності. Тестування надійності.
2. Метрики для оцінки продуктивності та надійності програмного забезпечення.
3. Інструменти для тестування продуктивності.

Тестування продуктивності та його важливість

Продуктивність (Performance Efficiency) - здатність ПЗ при заданих умовах забезпечувати необхідну працездатність стосовно виділених для цього ресурсів. Можна визначити її і як відношення одержуваних за допомогою ПЗ результатів до затрачуваних на це ресурсів усіх типів.

ISO/IEC 25010 включає performance efficiency серед головних характеристик якості.

Тестування продуктивності та його важливість

Тестування продуктивності (Performance Testing) – яке проводиться для визначення наскільки стабільно і як швидко працює програма або її частина під деяким навантаженням. Метою тестування є вузьких місць (bottlenecks) в системі, визначення швидкості завантаження даних і їх обробки, надійності програми.

Цілі тестування продуктивності:

- Перевірити, як система поводиться під різними навантаженнями.
- Виявити вузькі місця (bottlenecks).
- Виміряти швидкість реакції, стабільність, масштабованість.
- Порівняти реальні показники з очікуваними.
- Підготувати систему до production-навантаження.

Тестування продуктивності та його важливість

Чому тестування продуктивності важливе?

- У сучасному світі швидкодія = конкурентна перевага
- Забезпечує хороший користувацький досвід
- Знижує ризики збою в production-середовищі
- Виявляє "вузькі місця" на ранніх етапах процесу розробки.
- Перевіряє готовність до сценаріїв, таких як різкі стрибки трафіку або тривале використання.

Види тестування продуктивності

Вид тестування	Мета	Приклад сценарію
Load Testing	Виміряти поведінку при типовому навантаженні	100 користувачів одночасно оформлюють замовлення
Stress Testing	Перевірити межі системи при перевищенні ліміту	10 000 користувачів роблять запит одночасно
Spike Testing	Оцінити реакцію системи на раптовий стрибок навантаження	Різкий підйом кількості запитів до 10 разів за секунду
Endurance (Soak) Testing	Перевірити стабільність протягом тривалого часу	Сервіс працює 24 години з постійним навантаженням
Scalability Testing	Визначити, як система масштабується	Додавання серверів при рості трафіку
Volume Testing	Перевірити ефективність роботи з великими обсягами даних	Обробка мільйона записів у БД

Метрики для оцінки продуктивності

Метрика	Зміст	Одиниця виміру
Response Time	Загальний час комунікації між клієнтом і сервером	мс, с
Throughput	Кількість операцій за одиницю часу	запитів/с
Latency	Затримка між запитом і відповіддю	мс
Error Rate	Частка невдалих запитів	%
CPU / Memory Usage	Ресурсне навантаження системи	%
Concurrency Level	Кількість одночасних користувачів	к-сть користувачів

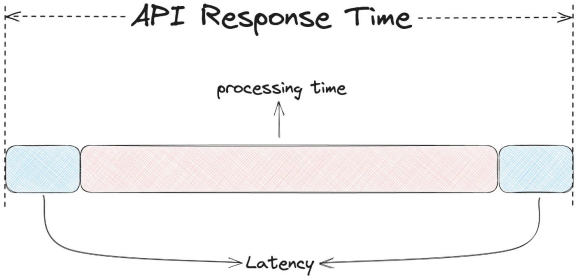
Метрики для оцінки продуктивності

Час відповіді API (API Response Time) — це загальний час клієнт-серверної взаємодії. Він показує, скільки часу потрібно API, щоб відповісти на запит клієнта. До цього часу входить як обробка запиту на сервері, так і передача відповіді назад клієнту.
Іншими словами, це проміжок від моменту, коли клієнт надсилає запит, до моменту отримання відповіді.
Якщо запит виконується повільно — це означає, що час відповіді занадто великий. Будь-який фактор, який спричиняє уповільнення, впливає на збільшення цього показника.

Час відповіді складається з двох основних компонентів:
Затримка (latency) — час, необхідний для передачі даних між клієнтом і сервером.
Час обробки (processing time) — час, який потрібен серверу для обробки запиту та формування відповіді.

На latency можуть впливати такі чинники:

- швидкість мережі,
- навантаження на сервер,
- продуктивність балансувальника навантаження,
- розмір переданих даних,
- архітектура та якість дизайну API.



Методика проведення тестування продуктивності.

1. Аналіз вимог — що саме хочемо перевірити.
2. Планування сценаріїв — визначення користувацьких потоків (login, search, checkout).
3. Підготовка середовища — створення окремої ізольованої тестової інфраструктури.
4. Запуск тестів — у контрольованих умовах.
5. Збір та аналіз метрик — CPU, пам’ять, latency, throughput.
6. Виявлення вузьких місць — наприклад, повільні SQL-запити або блокування потоків.
7. Оптимізація та повторне тестування.

Тестування надійності

Надійність (Reliability) – здатність ПЗ виконувати необхідні завдання у зазначених умовах протягом заданого проміжку часу або вказаної кількості операцій. Атрибути даної характеристики – це завершеність і цілісність всієї системи, здатність самостійно і коректно відновлюватися після збоїв у роботі, відмовостійкість.

Тестування надійності — це вид тестування програмного забезпечення, який перевіряє, чи може програмне забезпечення виконувати безвідмовну роботу в певному середовищі протягом визначеного періоду часу.

Види тестування надійності

Тип	Мета
Stability Testing	Перевірити роботу системи при тривалому використанні
Recovery Testing	Перевірити, як система відновлюється після збою
Failover Testing	Тестування поведінки при відмові одного з серверів
Chaos Testing	Імітація випадкових збоїв компонентів системи

Метрики тестування надійності

Метрика	Зміст
MTBF (Mean Time Between Failures)	Середній час між відмовами. MTBF = MTTF + MTTR
MTTF (Mean Time To Failures):	Різниця в часі між двома послідовними збоями.
MTTR (Mean Time To Repair)	Середній час відновлення
Availability	Доступність системи, наприклад 99.9%
Error Resilience	Здатність системи продовжувати роботу при часткових збоях

Інструменти для тестування продуктивності

Під час тестування швидкодії важливо правильно обрати інструмент, який відповідає типу системи, навантаженню та вашим цілям. Нижче наведено популярні рішення, які допомагають автоматизувати перевірку продуктивності:

Apache JMeter — класичний open-source інструмент для тестування вебдодатків, API та баз даних. Підтримує різні типи запитів і має зручний графічний інтерфейс.
Gatling — високопродуктивний інструмент, написаний на Scala. Добре інтегрується в CI/CD конвеєри, забезпечує детальну аналітику результатів.
Locust — гнучкий інструмент на Python, що дозволяє створювати сценарії тестування у вигляді коду. Зручний для розробників, які хочуть автоматизувати навантажувальні тести.
k6 — сучасний інструмент командного рядка з підтримкою JavaScript-сценаріїв. Часто використовується у DevOps середовищі, легко інтегрується з CI/CD.
LoadRunner — комерційне рішення корпоративного рівня, яке підтримує різні протоколи та забезпечує високу точність моделювання навантаження.
Artillery.io — простий інструмент для Node.js, зручний для тестування API та мікросервісів, має зрозумілий синтаксис і легку інтеграцію з CI/CD.

Корисні посилання

1. https://www.testrail.com/blog/performance-testing-types/
2. https://blog.sentry.io/whats-the-difference-between-api-latency-and-api-response-time/
3. https://training.qatestlab.com/blog/technical-articles/performance-testing/

Дякую за увагу!