МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о. завідувача кафедри
_____ Сафонова С.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Інформаційна технологія аналізу продуктивності WEB-систем

Освітній рівень"Магістр"
Спеціальність 122 "Комп'ютерні науки"

Науковий керівник роботи: _____ (підпис) В.М.Барбарук (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О.Критська (ініціали, прізвище)

Студент: _____ (підпис) В.Р.Макаренко (ініціали, прізвище)

Група: КН-18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри _____
_____ С.О. Сафонова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Макаренку Владиславу Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія аналізу продуктивності WEB-систем

керівник проекту (роботи) Барбарук Віктор Миколайович, к.т.н., доц.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «11» 10 2019 р. № 135/15.15
2. Строк подання студентом роботи 10.01.2020
3. Вихідні дані до роботи Матеріали науково-дослідної практики, перелік існуючих підходів до оптимізації показників продуктивності системи, список існуючих типів навантажувальних тестів, теоретичні відомості з існуючих метрик навантажувального тестування

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Навантажувальне тестування, підходи до оптимізації показників продуктивності , огляд «вузьких місць» тестувальної системи та вимоги до показників продуктивності, проведення навантажувального тестування та аналіз отриманих результатів, охорона праці та безпека в надзвичайних ситуаціях, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 14.10.2019

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання до магістерської роботи	02.09.2019-15.09.2019	
2	Аналіз завдання, робота з літературою	16.09.2019-22.09.2019	
3	Аналіз технічних засобів	23.09.2019-25.09.2019	
4	Створення тестів, використовуючи Gatling	26.09.2019-06.10.2019	
5	Розробка частини проекту "Охорона праці та безпеки в надзвичайних ситуаціях"	07.10.2019-25.11.2019	
6	Оформлення пояснювальної записки, автореферату та презентації	26.11.2019-9.01.2020	
7			

Студент _____ В.Р.Макаренко
(підпис) (прізвище та ініціали)

Науковий керівник _____ В.М.Барбарук
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Макаренко В.Р. Інформаційна технологія аналізу продуктивності WEB-систем.

Метою роботи є аналізу показників продуктивності програмного забезпечення з використання засобів навантажувального тестування.

Об’єктом дослідження є бібліотечна система Sierra.

Був проведений аналіз продуктивності системи первинної та актуальної версії,після впровадження методів оптимізації показників продуктивності, при навантаженні віртуальними користувачами. Також було дано рекомендації щодо подальшого розвитку продуктивності системи.

Для навантаження віртуальними користувачами був використаний фреймворк навантажувального тестування, Gatling. За допомогою режиму Recorder було записано навантажувальний тест та знайдено вузькі місця системи після запуску створеного тесту.

Ключові слова: час відгуку, навантажувальне тестування, кешування, оптимізація показників продуктивності, метрика.

ABSTRACT

Makarenko V.R. Information technology technology analysis of the productivity of WEB-systems.

The purpose of the work is to analyze the performance indicators of the software using load testing tools.

The object of the research is the Sierra library system.

System’s performance analysis of the primary and current versions was carried out after the introduction of methods to optimize performance indicators under load by virtual users. Recommendations were also made for further development of system performance.

For load by virtual users, a load testing framework, Gatling, was used. Using Recorder mode, a load test was recorded, and system bottlenecks were found after the launch of the created test.

Keywords: time of response, loading testing, cashing, optimization of indicators of productivity, metric.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

НТ – навантажувальне тестування

ПП – показники продуктивності

ВСТУП

У сучасному світі, коли обсяги оброблюваних даних, кількість користувачів і запитів до систем продовжують зростати, питання продуктивності і її тестування широко обговорюються в професійному середовищі.

Проблеми з продуктивністю призводять до відмови клієнтів від використання веб-систем. Система повинна бути доступна і виконувати свої завдання в будь-який момент часу, коли вона знадобиться клієнтові.

Тестування продуктивності може допомогти визначити характер або місце розташування проблеми, пов'язаної з програмним забезпеченням, підкреслюючи, де програма може вийти з ладу або відставання.

Продуктивність системи визначається кількістю операцій, які може обробити система за певний час, кількість операцій, які можуть бути виконані одночасно, час відгуку, кількість резервної потужності,яка потрібна системі для забезпечення зростання навантаження, кількість виключень, які система генерує під навантаженням.

Ці фактори можуть виявитися очевидними, але, звичайно, вони будуть змінюватися в залежності від рівня навантаження на систему. При достатній кількості ресурсів і невисокому навантаженні більшість систем виконує операції швидко. Критичний момент настає, коли навантаження збільшується до якогось граничного рівня.

За допомогою навантажувального тестування, а також внесення покращень та змін, заснованих на результатах цього тестування, ви зможете краще підготуватися до своїх користувачів та надати їм професійний сайт.

У зв'язку з цим, проведення якісного тестування навантаження повинне стати обов'язковим, для забезпечення стабільності роботи веб-систем.

1 НАВАНТАЖУВАЛЬНЕ ТЕСТУВАННЯ

1.1 Поняття навантажувального тестування та його роль

В даний час досить багато уваги приділяється зовнішньому вигляду веб-сайтів, їх функціональності, різним маркетинговим аспектам (реклама та просування). Ринки створення і просування сайтів мають свою сегментацію, замовники готові формувати вимоги за цими напрямками, бачать і розуміють результат при прийманні робіт. Однак, є два аспекти, які в основному не враховуються в процесі розробки і просування сайтів: зручність користування і швидкість роботи.

Навантажувальне тестування – це створення імітаційного моделювання у межах системи, максимально ближчої до продукту, готового до використання широким колом користувачів.Навантажувальне тестування дозволяє виявити рівень критичних навантажень при роботі з базою даних, інтернет-серверами та іншими ресурсами [8].

Навантажувальне тестування – це вид нефункціонального тестування, направлений на визначення показників швидкості реакції програми на зовнішні впливи при різній за характером і інтенсивності навантаження.

НТ може виявити системні затримки, проблеми із завантаженням сторінок та інше, що може відбуватися з системою, коли декілька користувачів звертаються до системи – цю проблему можна з легкістю пропустити на етапі розробки та тестування.

З точки зору технічної реалізації, навантажувальне тестування неможливо виконати до тих пір, доки проект не буде близьким до завершення свого виробничого циклу, у якому фактична взаємодія користувача та продуктивність системи можуть бути точно змодельовані та віддані на тестування.

1.2 Основні види навантажувальних тестів

Тестування продуктивності - вид тестування, призначений для визначення часу відгуку, пропускної здатності, надійності і масштабованості системи при даному рівні навантаження [2]. Об'єктом ТП є продуктивність програми за певних умов: робота заданої кількості користувачів на якомусь загальному ресурсі, їх дії в системі, тривалість роботи системи в цілому і т.д. У межах навантажувального тестування та тестування

продуктивності виконуються, як правило, наступні види тестів. На рисунку 1.1 зображено основні види тестів продуктивності.

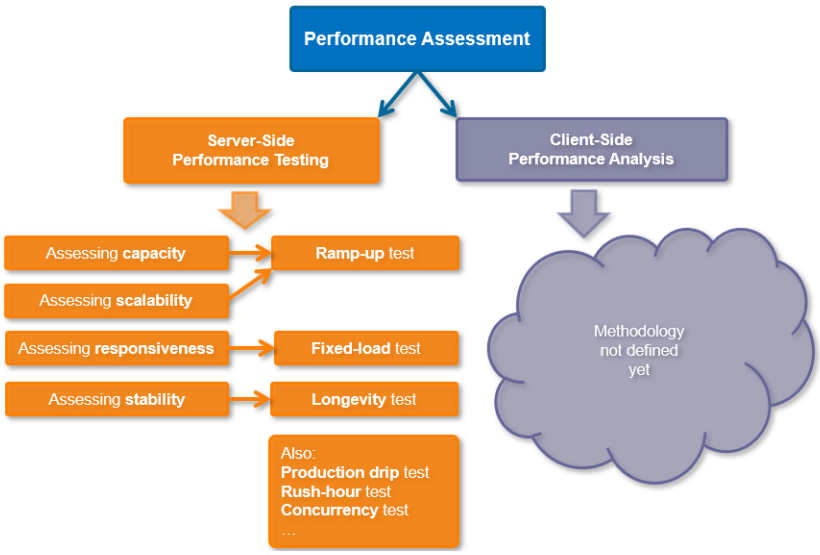


Рисунок 1.1 – Основні види тестів продуктивності

Тест на визначення максимальних можливостей системи (capacity test) дозволяє визначити так названу "точку насичення системи" (system saturation point) – рівень навантаження, при якому подальше збільшення кількості користувачів призводить до збільшення часу відклику системи або погіршенню стабільності системи, але не до збільшення кількості операцій, оброблених системою за одиницю часу. Capacity test спрямований на оцінку продуктивності системи апаратно-програмного комплексу, оскільки враховує доступні апаратні ресурси і ефективність їх використання.

Як можна побачити на рисунку 1.2 при 28 користувачах час відклику становить 3.5 секунди, а при збільшенні навантаження до 29 користувачів, час відклику починає перевищувати 3.5 секунди, що призводить до погіршення стабільності системи.

Перевагами тесту на визначення максимальних можливостей системи (capacity test) є:

– надання інформації про те, якими способами навантаження може бути оброблене для задоволення бізнес-вимоги;

– є можливість проведення різних тестів на порівняння нарощування навантаження;

– надає можливість визначення поточного використання і ємність існуючої систем для планування виробничих потужностей;

– забезпечення використання і тенденції ємності існуючої системи, для планування потужності.

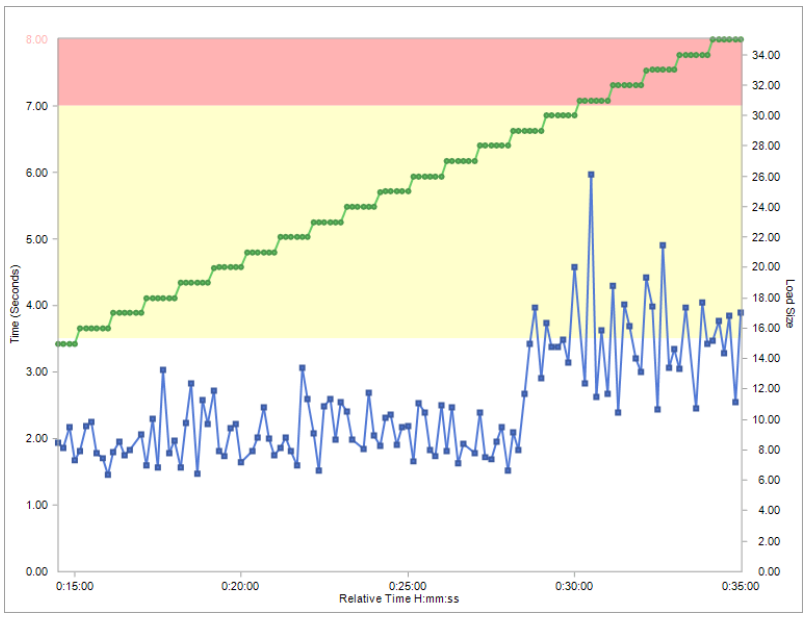


Рисунок 1.2 –Приклад capacity test

Проведення декількох тестів на визначення максимальних можливостей системи з додаванням апаратних ресурсів дозволяє визначити показники масштабованості (scalability) системи, яка визначається як здатність системи збільшувати продуктивність пропорційно додаванню апаратних ресурсів системи.

Перевагами тесту на масштабованість (scalability test) є:

– надає можливість визначити, як додаток масштабується зі збільшенням навантаження;

– є можливість визначити надійність на стороні сервера.

Volume test - це класичний тест продуктивності, де програмне забезпечення піддається великому обсягу даних. Об’ємне тестування проводиться для аналізу продуктивності системи шляхом збільшення обсягу даних в базі даних [9]. За допомогою об'ємного тестування вплив на час відгуку та поведінку системи можна досліджувати, коли система піддається великому обсягу даних.

Низько-, середньо- і високонавантажена робота (low-, mid-, high-load tests) – дозволяє оцінити час відгуку (response time) системи в деяких заданих діапазонах навантаження. Дана інформація може бути використана при складанні переліку вимог до умов експлуатації системи. Основними вимогами до тесту на час відгуку є:

– визначити, як довго система виконує транзакцію;

– порівняти поточний час відгуку системи з еталонним.

Тестування стабільності (Stability testing) - тест на можливість системи продовжувати функціонувати з часом та в усьому діапазоні використання, не викликаючи збої. Stability testing - це тривалий тест, який використовується для оцінки продуктивності

та / або стабільності програми у часі. Цей тест є популярним та корисним, тому що при проведенні цього виду тестування здійснюється спостереження за споживанням пам’яті для виявлення потенційних витоків. Крім того, таке тестування виявляє деградацію продуктивності, котра виражається у зниженні швидкості обробки інформації та / або збільшенні часу відповіді аплікації після тривалої роботи, порівняно з початком тесту. Основною метою Stability testing є:

– забезпечення впевненості в стабільності тестувальної системи;
– моніторинг ефективності системи;
– надання стабільності системи під тестуванням навантаження.

Тест на виживання (longevity test) показує здатність системи працювати тривалий час під високим навантаженням. Однією з найбільш небезпечних проблем, що виявляються даним тестом, є витік пам'яті та інше зниження ефективності використання апаратних ресурсів, через з часом накопичених помилок в роботі програми .

Тест "час пік" (rush hour test) дозволяє оцінити реакцію системи при різкій зміні навантаження. Під час тесту перевіряється здатність системи витримати стрибкоподібне збільшення навантаження під час "часу пік", а також здатність системи повернутися до початкових показників продуктивності після завершення "часу пік" (відновлення вихідних показників навантаження на систему). Такий тест дозволяє виявити проблеми з синхронізацією виконання окремих ділянок коду, а також проблеми з управлінням всіма видами міжкомпонентної взаємодії (в тому числі мережевих і локальних з'єднань) на всіх рівнях системи.

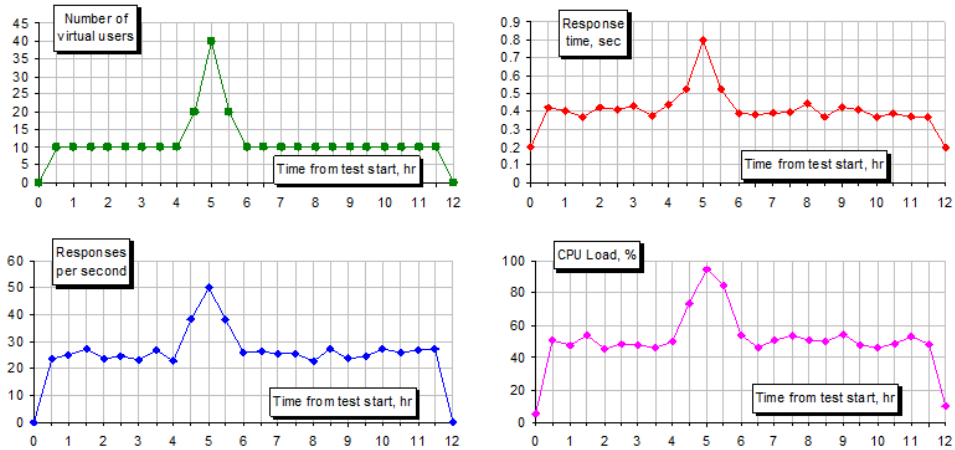


Рисунок 1.3 – Приклад “rush hour test” на системі з гарною продуктивністю

Як видно з рисунку 1.3, на якому зображено приклад rush hour test на системі з гарною продуктивністю, система витримала збільшення навантаження і після «часу пік» нормальна робота системи була відновлена.

На рисунку 1.4 зображено приклад rush hour test на системі з поганою продуктивністю. Як видно з графіків система витримала збільшення навантаження, але після «часу пік» продуктивність стала нестабільною, з великою кількістю коливань.

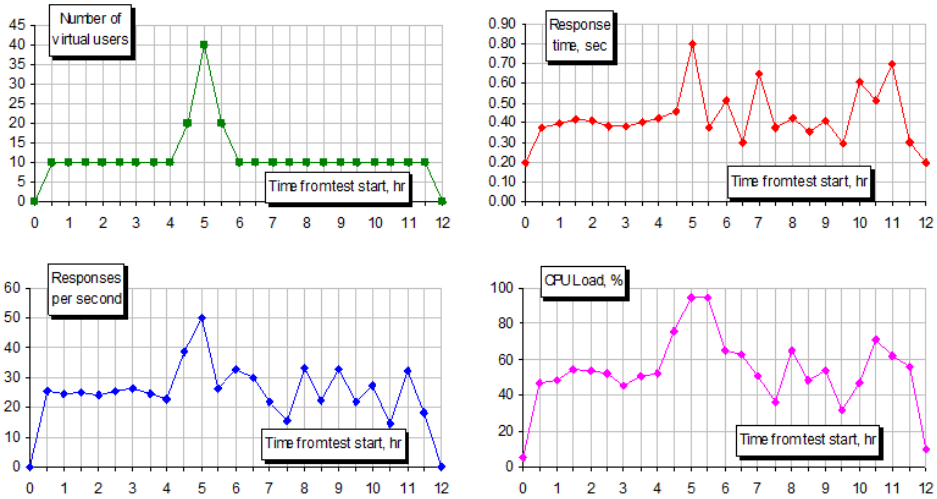


Рисунок 1.4 – Приклад “rush hour test” на системі з поганою продуктивністю

Тест "точки рандеву" - мається на увазі таке налаштування профілю навантаження і поведінки віртуальних користувачів, щоб в певний момент всі вони одночасно виконували одну і ту ж операцію: як правило, синхронну операцію збереження, запису, і тому подібне. На відміну від тесту "часу пік" цей тест не має на увазі збільшення числа одночасно працюючих з системою користувачів, а має на увазі дослідження ситуації конкуренції користувачів за деякі ресурси, спільне використання яких не представляється можливим або пов'язане з підвищеним навантаженням на системні ресурси. Зокрема, цей тест дозволяє виявити проблеми з поділом ресурсів на рівні баз даних.

Стрес тестування (Stress testing) − це найбільш популярний вид тестування, який характеризує систему з точки зору стійкості її роботи за умов, що перевищують нормальні. Стресом в даному контексті може бути перевищення інтенсивності виконання операцій до дуже високих значень або аварійна зміна конфігурації сервера.

Також одним із завдань при стресовому тестуванні може бути оцінка деградації продуктивності. Перевіряти працездатність системи в екстремальних умовах в першу чергу необхідно для програмного забезпечення, збої в роботі якого можуть призвести до великих і серйозних наслідків, як фінансових, так і більш серйозних, аж до людських життів.

Як можна побачити на рисунку 1.5 стрес-тест виявляє, що при навантаженні системи 41 користувачами, система працює, незважаючи на збільшення часу завантаження сторінки. Але, коли розмір навантаження досягає 42 користувачів, робота

системи починає погіршуватися, при цьому час завантаження сторінки становить 15-17 секунд.

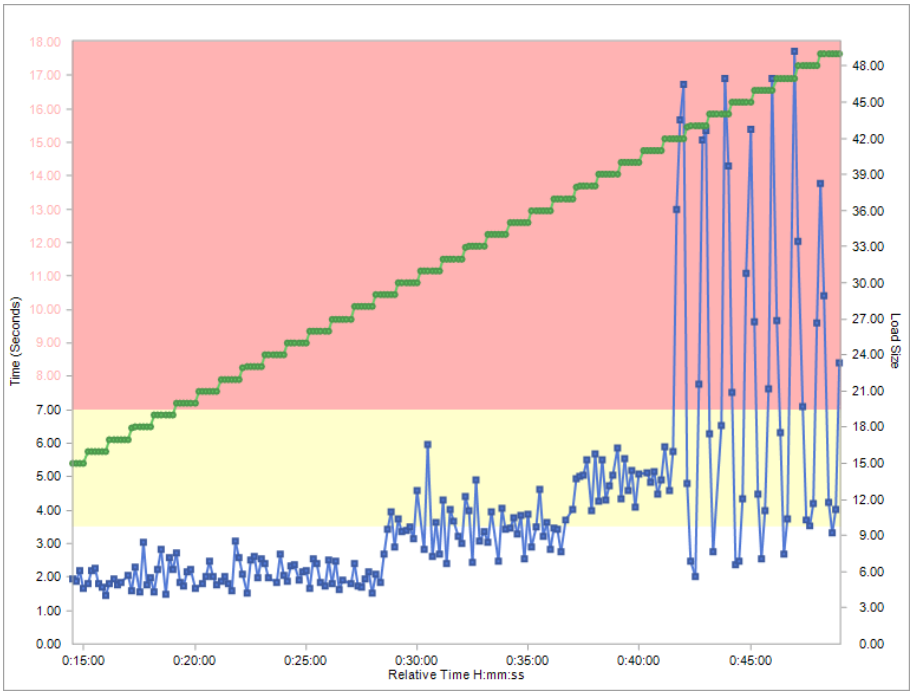


Рисунок 1.5 – Приклад стрес-тесту

Pipe-clean test - це підготовче завдання, яке служить для перевірки кожного сценарію тестування продуктивності в середовищі тестування продуктивності. Зазвичай тест виконується для одного сценарію та одного віртуального користувача протягом певного періоду часу або для заданої кількості ітерацій.

Проведення pipe-clean test, в ідеалі, повинно здійснюватися без будь-якої іншої діяльності в системі для забезпечення найкращого вимірювання. Потім ви можете використовувати показники, отримані в якості базової лінії, для визначення ступеня деградації продуктивності, яка виникає у зв'язку з збільшенням кількості користувачів, а також для визначення місця розташування сервера та мережі для кожного сценарію. Цей тест також забезпечує важливий внесок у модель навантаження.

1.3 Моделювання навантаження

Робоче навантаження – розподіл навантаження по всім ідентифікованим сценаріям. Модель робочого навантаження системи показує, як система буде використовуватися у виробничому середовищі. Для отримання таких результатів продуктивності, які б відображали справжню поведінку системи, необхідно протестувати її на моделі робочого навантаження, яка дуже схожа на робоче середовище. Робоче навантаження можна розділити на робоче навантаження бізнесу та навантаження на інфраструктуру .

Бізнес-навантаження – це набір дій користувача або бізнес-сценаріїв, виконаних у тесті для досягнення бізнес-цілей та задач. Кваліфікований аналітик повинен мати повне уявлення про усі основні бізнес-транзакції програми та їх вплив на загальну продуктивність системи.

Навантаження на інфраструктуру – кількість ресурсів/інфраструктури, яка використовується для досягнення бажаної цілі. Використання CPU, пам'яті та операцій введення-виведення під час тестування є прикладом моделі робочого навантаження інфраструктури, яка задається при визначенні цілей тестування.

Правильне імітування параметрів системи у тесті надасть вірогідність більш точних результатів тесту продуктивності. Етап планування, на якому аналітик продуктивності гарантує, що інформація усіх параметрів була отримана, щоб точно імітувати їх у тесті продуктивності.

Одним із етапів планування – є визначення моделі робочого навантаження. Модель робочого навантаження включає в себенаступну інформацію: типи дій користувача, що будуть протестовані при загрузці; бізнес-сценарії, які будуть розроблені для усіх користувачів; розділення користувачів по кожному сценарію. Інформація моделі робочого навантаження допоможевирішити команді тестувальників продуктивності наступні проблеми:

– ідентифікувати сценаріїв продуктивності (основна діяльність моделі навантаження полягає в розумінні додатка і визначенні його сценаріїв продуктивності);

– спростити зв'язок (модель робочого навантаження дозволяє повідомляти сценарії продуктивності та розподіл користувачів по ним з усіма зацікавленими сторонами програми);

– підготувати тестові дані (модель робочого навантаження допомагає у визначенні типу та обсягу тестових даних, які завжди потрібні для початку роботи з інструментом);

– з'ясувати кількість інфраструктури, необхідної для проведення навантажувального тестування (для успішного проведення навантажувального тестування

завжди потребується багато інфраструктур. Неправильні результати з'являються у тому випадку, коли система тестується з неадекватною інфраструктурою. Зазвичай навантаження користувачів моделюють за допомогою декількох машин для точного тестування).

1.4 Етапи моделювання навантаження

Навантажувальне тестування – складний процес, тому що складається з безлічі етапів, на кожному з яких виконується декілька дій. Моделювання робочого навантаження є однією з найголовніших складових процесу тестування продуктивності.

Модель навантаження є ключем до точного та релевантного тестування продуктивності. Вона визначає розподіл навантаження за сценаріями використання, і, що дуже важливо, цільові рівні паралельності та пропускної спроможності, необхідні для тестування. Модель навантаження спочатку створюється для тестування обсягу, і вона повинна відображати цілі ефективності, після чого вона змінюється, як це потрібно для інших типів тестів продуктивності.

Інший, дуже важливий, але часто забутий фактор - це вплив даних тесту на дизайн моделі навантаження. Наприклад, якщо веб-система представляє пошук продукту, тоді тестові дані для керування сценарієм спираються на повні назви продуктів. У такому випадку, перевірені функціональні можливості пошуку, одночасно узгоджені, не завжди є репрезентативними щодо того, як клієнти використовуватимуть сайт. Наприклад, фактичний результат, якщо клієнт виконує пошук усіх чорних суконь, а не конкретний предмет, це може викрити погано налаштований алгоритм пошуку [1].

Процес створення моделі робочого навантаження складається з наступних етапів: ідентифікація тестових цілей, розуміння тестової системи, ідентифікація основних сценаріїв, визначення шляхів навігації ключовими сценаріями, створення тестових даних, відносний розподіл навантаження по ідентифікованим сценаріям, визначення рівнів цільового навантаження та інші опціон альні налаштування (рис. 1.6).

Ідентифікація тестових цілей повинна бути не тільки під час тестування продуктивності, але і у будь-якій діяльності необхідне прикладання зусиль, які відповідають цілям виду тестування. Під ідентифікацією тестових цілей мається на увазі визначення дій, які необхідно вжити для успішного досягнення поставлених цілей тестування.

Прикладами цілей (під цілями можна вважати ПП, які необхідно протестувати), поставлених для проведення тестування продуктивності веб-системи електронної комерції є:

– час відгуку;
– пропускна здатність;
– використання ресурсів;
– максимальне навантаження користувачами.

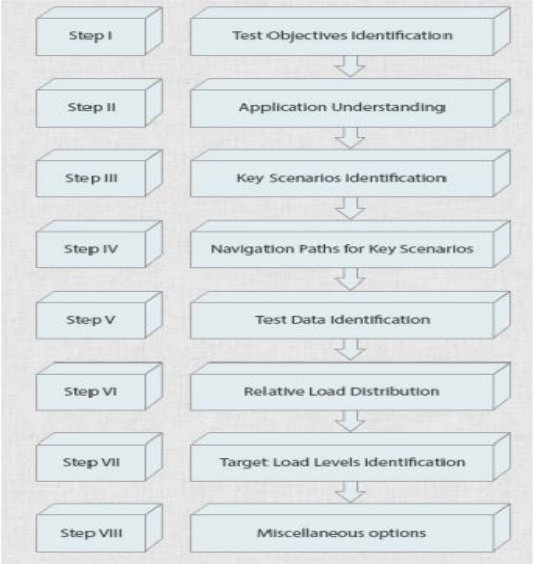


Рисунок 1.6 – Етапи моделювання робочого навантаження

Розуміння тестової системи є основною складовою процесу створення робочого навантаження. Важливою інформацією, якою повинні володіти команда тестувальників продуктивності є:

– кількість типів користувачів, що використовують програму;
– можливі бізнес-сценарії для кожного типу користувача;
– залежність навантаження користувачами від часу;
– час, необхідний для досягнення пікового навантаження;
– тривалість пікового навантаження.

На етапі ідентифікації ключових сценаріїв не потребується імітація усіх дій користувача через обмежену кількість бюджету та часу. Тому необхідно обрати обмежену кількість призначених користувачу дій, які надають найбільший вплив на роботу системи. Прикладами таких сценаріїв можуть бути наступні:

– вимірювані сценарії (Measureable Scenarios) – основним критерієм будь-якого користувальницького сценарію є його повна вимірюваність;

– найчастіше виконувані сценарії (Most Frequently Accessed Scenarios) – найбільш розповсюджені користувальницькі сценарії;

– бізнес критичні сценарії (Business Critical Scenarios) – основні сценарії системи, які містять бізнес-операції;

– сценарії інтенсивного використання ресурсів (Resource Intensive Scenarios) – користувальницькі сценарії, які споживають більше ресурсів у порівнянні з типовими сценаріями;

– залежні від часу часто використовувані сценарії (Time Dependent Frequently Accessed Scenarios) – такі сценарії системи, які доступні тільки у конкретних випадках.

Наступним етапом у процесі створення моделі навантаження є визначення шляхів навігації ключовими сценаріями. Існує безліч різноманітних шляхів для проходження обраних сценаріїв. Кожний користувач володіє різним рівнем технічних знань, і це цілком очевидно, тому буде слідувати до завершення певного сценарію різними кроками. Тому необхідною умовою буде ідентифікувати усі можливі шляхи, якими будуть слідувати користувачі, щоб успішно завершити ідентифікований сценарій, а також вирішити чи включати його у тестування навантаження або ні.

Реакція системи на той самий сценарій може сильно варіюватися в залежності від призначеного для користувача шляху навігації, тому настійно рекомендується перевірити усі основні шляхи обраного сценарію під навантаженням. Для визначення шляхів навігації по сценарію можна слідувати наступним рекомендаціям:

– використання шляхів, які можуть бути використані для успішного завершення більше одного ідентифікованого сценарію та мають значний вплив на продуктивність;

– використання керівництва для з’ясування усіх можливих шляхів ідентифікованого сценарію;

– дослідження системи шляхом виконання сценарію;

– інший підхід полягає у забезпеченні доступу до додатку для нових і досвідчених користувачів. Необхідно попросити їх виконати певні сценарії та спостерігати за цим. Таких підхід є важливим, тому що у експертів системи є необхідний досвід роботи у таких системах.

Ідентифікація унікальних тестових даних є наступним етапом створення моделі навантаження. Для успішного проходження сценаріїв, які були виявлені на попередніх етапах, необхідна підготовка тестових даних. Цей етап потрібний для успішної імітації проходження сценаріїв у тесті продуктивності. Для того, щоб досягти точних результатів у ході тестування, необхідно, щоб використовувалися ефективні та коректні тестові дані. Для того, щоб зрозуміти чи є обрані тестові дані відповідними до нашого тесту,

необхідно:зрозуміти, скільки часу користувач витрачає на сторінці під час навігації по певному шляху; виявити умови, що змушують користувача змінити шлях навігації; обрати конкретні тестові дані для кожного шляху сценарію.

Для того, щоб ефективно виконати тест, необхідно підготувати велику

кількість тестових даних та слідкувати за станом бази даних. Тестові дані - це дані, які будуть надані у якості вхідних даних для сценаріїв. Потрібно точно подивитися, що потрібно, скільки їх потрібно, а також - скільки роботи потрібно для її створення. Якщо, наприклад, було виділено два тижні для тестування продуктивності, і це займе місяць, щоб підготувати достатньо даних про тестування, потрібно ще раз подумати. При виконанні навантажувального тестування необхідно дотримуватись наступних правил:

– необхідно переконатися, що всі необхідні тестові дані підготовлені, тому що знадобляться додаткові тестові дані при тестуванні сценаріїв з усіма шляхами навігації;

– для декількох користувачів необхідно використовувати унікальні тестові дані, що допоможе обійти отримання неправильних результатів;

– під час проведення навантажувального тестування обов’язково необхідно періодично перевіряти стан бази даних для того, щоб бути впевненим у тому, що вона не перевантажена;

– винести у список позитивних сценаріїв негативні, тому що використання некоректних даних необхідне для імітації поведінки реальних користувачів, які можуть іноді помилятися та надавати невірні значення також.

Критично важливі три типи тестових даних: вхідні дані, цільові дані та дані сеансу. Нижче приведені деякі типові приклади вхідних даних.

Вхідні дані користувачів. Для програм, ці дані, як правило, складаються з логіна та пароля. Багато тестів продуктивності виконуються з обмеженою кількістю тестових облікових даних користувача. Це вносить елемент ризику стосовно декількох користувачів, з тими самими реєстраційними даними, які одночасно є активними, що в деяких випадках може призвести до помилок виконання. Коли це можливо, слід надати унікальні реєстраційні дані для кожного віртуального користувача, який буде включено в тест продуктивності.

Критерії пошуку. Майже у всіх веб-системах завжди використовуються випадки при тестуванні продуктивності, в яких проводяться різних видів пошуку. Щоб ці пошуки були реалістичними, необхідно надати різні дані, які формують критерії пошуку. До типових прикладів належать ім'я, адреса, номер рахунка-фактури та коди продукції. Можливо,також доведеться виконувати пошук за шаблоном, де в якості ключа пошуку надано лише певну кількість символів. Якщо програма дозволяє кінцевому користувачеві шукати всі записи, які починаються з А, то це повинно бути включено у дані тесту.

Асоційовані файли. Для деяких типів тестів продуктивності може виникнути необхідність пов'язати файли з випадками використання. Це загальна вимога до систем управління документами, де час завантаження та завантаження вмісту документа є критичним показником продуктивності. Ці документи можуть бути у різних форматах (наприклад, PDF, MS Word), тому необхідно забезпечити наявність достатньої кількості правильного розміру та типу.

Цільова база даних необхідна, ящо потрібно заповнити її реальними обсягами дійсних даних, щоб запити користувачів вимагали від бази даних реалістичних пошуків. Важливо забезпечити тестову базу даних реального розміру. Значно менші обсяги даних, ніж ті, що будуть присутні під час розгортання, дають потенційну можливість ввести в оману час відгуку на базі даних. Також можна використовувати snapshot існуючої виробничої бази даних, що має додаткову перевагу від реальних, а не тестових даних.

Відкат даних. Якщо будь-який з тестів продуктивності, який запускається під час тестування продуктивності, змінює вміст даних у тестовій базі даних, то в ідеалі - перед кожним виконанням тесту продуктивності - база даних повинна бути відновлена в тому ж стані, в якому вона була, до запуску тесту. Потрібно мати механізм для здійснення відкату даних у реальному часі. Якщо для відновлення бази даних потрібні дві години, то це необхідно враховувати в загальному часу, що допускається для проекту тестування продуктивності.

Дані сеансу. Під час виконання тестування продуктивності часто необхідно перехоплювати та використовувати дані, які повертаються з програми. Ці дані відрізняються від даних, введених користувачем. Типовим прикладом є інформація, пов'язана з поточною сеансом користувача, яка повинна бути повернена як частина кожного запиту, зробленого клієнтом. Якщо ця інформація не надана або невірна, сервер поверне помилку або відключить сеанс. Більшість інструментів тестування продуктивності забезпечують функціональність обробки такого роду даних. У таких ситуаціях, якщо немає змоги правильно обробляти дані сеансу, це зазвичай буде досить очевидним, оскільки скрипти не зможуть повторюватися. Проте будуть випадки, коли скрипт працює, але повторення буде неточним; у цій ситуації результати тестування продуктивності будуть недостовірними.

Безпека даних. При підготовці тестових даних потрібно також розглянути конфіденційність інформації, що вони містять. В деяких випадках, доведеться анонімувати такі дані, як імена, адреси та номери банківських рахунків, щоб не допустити, що особиста інформація буде скромпоментована кимось та хтось випадково зможе переглянути дані тесту.

Відносний розподіл навантаження по ідентифікованим сценаріям – це наступний етап після визначення основних сценаріїв тестування, шляхів навігації та створення тестових даних, необхідних для кожного ідентифікованого сценарію. Метою даного етапу є з'ясування розподілу усіх ідентифікованих сценаріїв у моделі робочого навантаження. Цілком очевидно, що в будь-якому додатку кілька сценаріїв виконуються частіше у порівнянні з іншими. Крім того, для деяких додатків, виконання сценаріїв залежить від відносного часу. Тому необхідно з'ясувати, які сценарії будуть виконуватися в який час та яким буде їх відсоток виконання у загальному робочому навантаженні. Для визначення відносного розподілу ідентифікованих сценаріїв необхідно :

– виконати перевірку файлів журналу сервера для того, щоб визначити тенденції користувачів у системі, якщо вони знаходяться у робочому середовищі;

– з'ясувати найбільш популярні серед користувачів функції, які, за думкою відділу продажу та маркетингу, будуть використовуватися;

– провести опитування у користувачів системи та з'ясувати, в яких функціях системи вони найбільш зацікавлені.

Наприклад, для сайту електронної комерції, ідентифікований розподіл сценаріїв може бути наступним (рис.1.7):

User Scenario	Percent Load Distribution
Browsing product catalog	40
Creating a user account	05
Searching for a product	30
Login to application	15
Order Placement	10
Total	**100**

Рисунок 1.7 – Приклад ідентифікованого розподілу сценаріїв для сайту електронної комерції

Визначення рівнів цільового навантаження – це наступний етап формування моделі робочого навантаження. Одним із останніх кроків, зв'язаних з виконанням робочої моделі навантаження після виконання усіх попередніх дій, є визначення нормального та пікового рівня навантаження для кожного обраного сценарію для тестування системи на очікувані та пікові навантаження. Головним чином в залежності від додатку необхідно визначити щомісячні, щотижневі та щогодинні середні цільові навантаження для кожного обраного

сценарію. Для ідентифікації цільових рівнів навантаження для системи необхідно володіти наступною інформацією:

– з’ясувати поточні та очікувані рівні звичайних та пікових запитів користувачів;

– мати інформацію про основні ключові тестові сценарії системи;

– необхідно з’ясувати розподіл користувальницьких запитів по визначеним сценаріям;

– створити навігаційні шляхи для усіх сценаріїв.

На рисунку 1.8 буде підведений підсумок, яка інформація буде потрібна при визначенні цільових рівнів навантаження для будь-яких сценаріїв :

Time Period	Normal Load Requests (Avg.)	Peak Load Requests (Avg.)	Peak Load Build up Time	Peak Load Duration
One Month	72,000	2,16,000	2 Hours	1 Hour
One Week	16,800	50,400	2 Hours	1 Hour
One day	2,400	7,200	2 Hours	1 Hour
One Hour	100	300	2 Hours	1 Hour

Рисунок 1.8 – Необхідна інформація при визначенні цільових рівнів навантаження для будь-яких сценаріїв

Після створення ключових сценаріїв та їх розподіл на рівні цільового навантаження останнім кроком створення робочого навантаження є визначення різних варіантів робочого навантаження, таких як поєднання браузеру, мережі, часу та паузи між ітераціями для імітації робочого навантаження відносно реального середовища. Усі інструменти тестування навантаження мають наступні опції:

– оснащення браузерами (необхідно вивести список усіх браузерів, які потрібно включити в тест та вказати розподіл навантаження для усіх браузерів для того,щоб перевірити, яка відповідь системи буде в усіх браузерах);

– оснащення мережею (включення основного списку мережевого підключення у свій тест та відповідним чином розподілити навантаження на них);

– час очікування (необхідно включити у тест час очікування, тому що реальні користувачі займають деякий час на роздуми при виконанні своїх дій, ігнорування часу очікування може призвести до недійсних результатів тесту);

– час паузи.

1.5 Метрики навантажувального тестування

Метрика - одиниця виміру, яка обчислює результат. Метрики продуктивності використовуються для обчислення параметрів продуктивності та виявлення слабких областей програми [4].

Метрики використовуються для відстеження прогресу проекту. Вони використовуються для створення базової лінії для всіх тестів. Використовуючи метрики, можна визначити основну причину проблеми, порівняти результати різних тестів і з'ясувати вплив будь-яких змін, внесених до програми. Це допомагає поліпшити якість продукту та виявити області, які потребують уваги.

Нижче будуть описані метрики, засновані на наступних характеристиках:

- метрики продуктивності, які є важливі для тестувальної системі;
- метрики, які легко прорахувати;
- метрики продуктивності, повні дані яких можна виявити та розрахувати.

Першим кроком є з’ясування метрик шляхом збору повного набору вимог до продуктивності. Це важливий крок для правильного та чіткого розуміння вимог. Вимоги можна зібрати, звернувшись до різних клієнтів, або це може бути зроблено за допомогою журналу активності сервера.

Метрики тестування навантаження, які будуть описані далі, є основними показниками ефективності веб-системи. Метрики відповіді показують вимірювання продуктивності з точки зору користувача .

Середній час відгуку (Average Response Time). При вимірюванні початку кожного запиту та завершенні кожної відповіді на ці запити, ми будемо мати дані про те, скільки часу потрібно цільовій веб-системі для доставки того запиту, який був відправлений (рис. 1.9). Середній час відгуку бере до уваги кожний цикл запиту/відповіді до кінця навантажувального тесту та вираховує середнє математичне усіх відгуків для цього інтервалу. Результуюча метрика відображає швидкість веб-системи – це найкращий показник роботи системи з точки зору користувача. Середній час відгуку включає доставку HTML, зображень, CSS, XML, Javascript-файлів та будь-яких інших використовуваних ресурсів. Таким чином, на середню швидкість будуть впливати у значній мірі будь-які повільні компоненти.

Час відгуку може бути виміряний як час до першого байту та як час до останнього байту. Час до першого байту показує, скільки часу потрібно для отримання запиту та скільки часу потрібно серверу для початку відповіді. Але це тільки частина реального рівняння. Корисніше знати весь цикл відповіді, який охоплює тривалість завантаження

ресурсу. Користувач бажає побачити HTML-сторінку, котра потребує отримання повного документу. Тому, час до останнього байту буде краще в якості ключового показника ефективності з плином часу першого байту.

Утилізація (Utilization) - відсоток теоретичної ємності ресурсу, що використовується. Приклади включають в себе, скільки пропускної здатності мережі споживається трафіком додатків або обсягом пам'яті, що використовується на веб-сервері, коли активується 1000 відвідувачів.

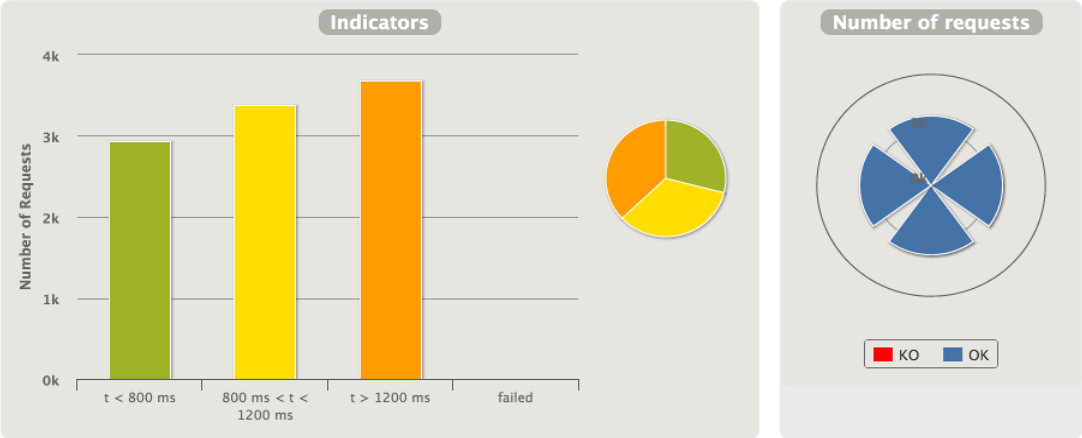


Рисунок 1.9 – Графік середнього часу відгуку

Піковий час відгуку (Peak Response Time). Як і попередній показник, метрика пікового часу відгуку вимірює кожний цикл запиту/відповіді. Тим не менш, піковий час показує, який найдовший цикл був на даному хвилинному інтервалі тесту (рис.1.10). Час пікового відгуку показує, що принаймні один із ресурсів проблематичний.

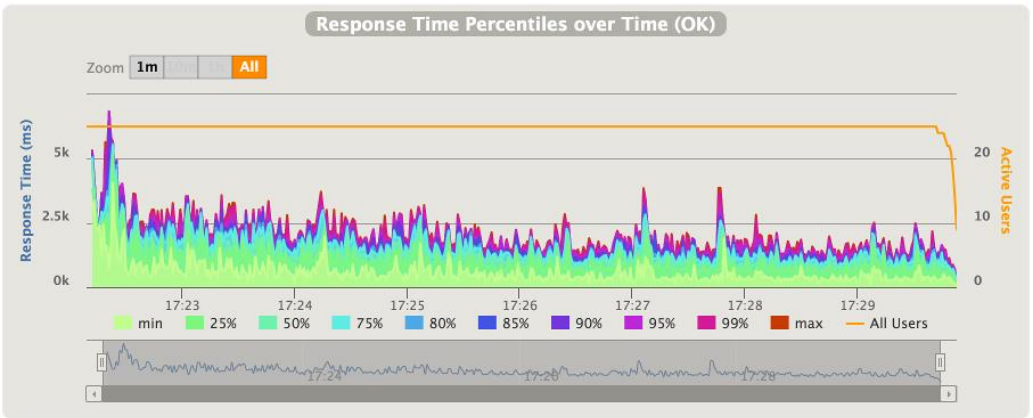


Рисунок 1.10 – Графік часу відгуку з плином часу

Це може відображати аномалію у системі, де цільовий запит неправильно обробляється цільовою системою. У веб-системі процес динамічного створення HTML-

документу із логіки додатку та запитів до бази даних зазвичай є найбільш довготривалою частиною системи. Великі за розміром файли можуть давати повільні відповіді, які будуть відображатися у піковому часі відгуку.

Частота помилок (Error Rate). Слід очікувати, що при обробці запитів, особливо під навантаженням, можуть виникати деякі помилки. У більшості випадків можна побачити, що помилки починають виникати, коли навантаження досягає межі, котра перевищує можливість веб-системи доставляти необхідне вчасно.

Коефіцієнт помилок – це математичне обчислення, котрий виробляє відсоток запитів із проблемами у порівнянні з усіма запитами. Відсоток відображає, скільки відповідей – коди стану HTTP, що вказують на помилку на сервері, а також будь-який запит, що спливає раніше отримання або завершення відповіді.

Веб-сервер повертає HTTP Status Code в заголовку відповіді. Звичайні коди, як правило, 200 (OK) або щось в діапазоні 3xx вказує на переадресацію на сервері. Загальний код помилки 500, який означає, що веб-сервер знає, що є проблема з виконанням цього запиту. Це, звичайно, не говорить вам, що викликало проблему, але це дає змогу зрозуміти, що сервер знає про існуючу технічну проблему у системі.

Коефіцієнт помилок є важливою метрикою, тому що обчислює «провал продуктивності» у системі. Він показує, скільки невдалих запитів виконується в певний момент навантаження. Значення цієї метрики є найбільш очевидним при умовах великого навантаження, так як коли відсоток проблем значно зростає, так як більше навантаження створює більше помилок. У багатьох навантажувальних тестах це перевищення це зростання частоти помилок буде радикальним.

Одночасні користувачі (Concurrent Users). Одночасні користувачі – це найбільш розповсюджений спосіб вираження навантаження, що використовується під часу тесту. Цей показник обчислює кількість активних користувачів в певний момент часу (рис. 1.11). Віртуальний користувач виконує те, що робить «реальний» користувач, як вказано у коді, створеному в інструменті навантажувального тестування.

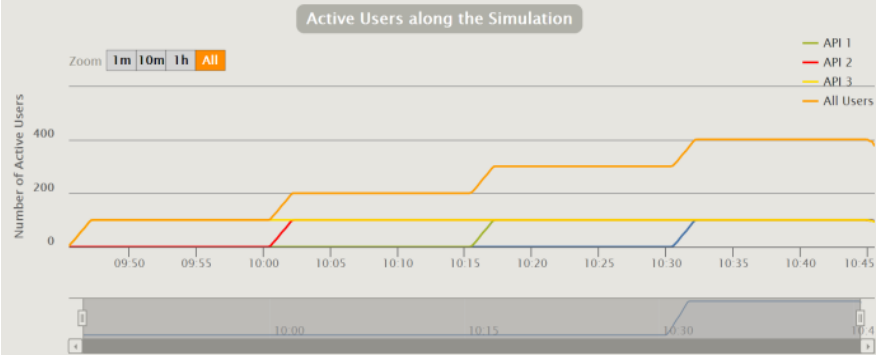


Рисунок 1.11 – Графік активних користувачів

Запит у секунду (Request Per Second). Запит в секунду - метрика, що обчислює, скільки запитів направляються на цільової сервер у секунду часу. Він включає в себе запити на HTML-сторінці, CSS таблицю стилів, XML-документи, бібліотеки JavaScript, зображення, Flash / мультимедійні файли, а також будь-який інший запитаний ресурс як можна побачити на рисунку нижче (рис. 1.12). Ця метрика залежить від кількості ресурсів, що визиваються із сторінок веб-системи.

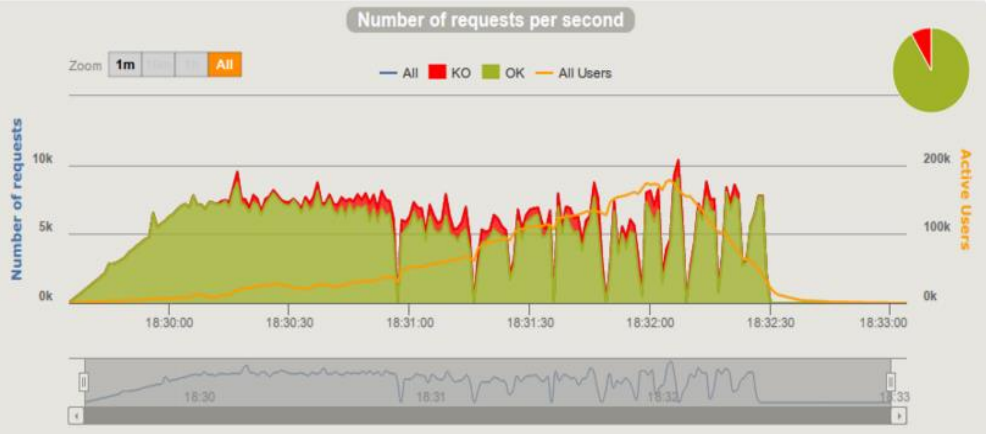


Рисунок 1.12 – Графік кількості запитів у секунду

Пропускна здатність (Throughput) вимірюється в одиницях кілобайт у секунду та цей вимір пропускної здатності, що споживається у ході роботи тесту. Ця метрика показує, скільки даних приходить із серверів. Гарним прикладом може бути кількість звернень на веб-сторінці протягом певного періоду часу.

Пропускна здатність буде часто змінюватися від тесту до тесту по зрівнянню з одночасними користувачами, але можуть бути й інші причини цих змін. Якщо пропускна здатність дуже висока це може означати, що сайт успішно передає велику кількість даних відповідей, але це також може бути сигналом того, що сайт має кілька ресурсів, таких як зображення, які можуть бути стиснуті для економії трафіку. Дуже низька пропускна здатність може означати, що сайт не був в змозі відповісти, перш ніж час очікування минув

1.6 Постановка задачі дослідження

Метою даної атестаційної роботи є порівняння показників продуктивності системи до та після оптимізації ПП системи. Аналіз показників продуктивності допоможе виявити

«вузькі місця» системи, тобто зробити висновок, які функції системи впливають на зниження рівня продуктивності, виявити потенційні ризики та можливі підходи для оптимізації продуктивності системи.

Буде проведено навантажувальне тестування, направлене на аналіз показників продуктивності проблемних функцій системи, які в свою чергу є найчастіше використовуваними користувачами та являються ключовими функціями системи. Тобто буде проведена імітація одночасного перебування на сайті великої кількості користувачів, які здійснюватимуть аналогічну послідовність дій у тестових функціях, і дослідження показників продуктивності, таких як - середній час відгуку, потрібний для вчинення таких дій, як, наприклад, процес завантаження сторінки, процес відкриття пункту меню; кількість запитів та відгуків, відправлених та отриманих у секунду.

Результатом завершення процесу тестування та аналізу даних будуть звіти, які відображатимуть показники продуктивності сайту до та після впровадження підходів оптимізації продуктивності системи.

Звіти будуть слугувати помічником для прийняття рішення подальших шляхів покращення роботи системи. Для проведення навантажувального тестування передбачається використання такого інструменту, як Gatling.

2 ПІДХОДИ ДО ОПТИМІЗАЦІЇ ПОКАЗНИКІВ ПРОДУКТИВНОСТІ ВЕБ-СИСТЕМ

2.1 Кешування даних

Кешування (або кеш) - це якийсь проміжний буфер, в якому зберігаються дані. Завдяки кешуванню сторінка сайту не відтворюється заново для кожного користувача. Кешування дозволяє здійснювати роботу з великою кількістю даних у максимально стислі терміни і при обмежених ресурсах (серверних і призначених для користувача). Необхідно розуміти, що роботу з даними можна проводити як на стороні клієнта, так і на сервері. Притому, серверна обробка даних централізована і має ряд безсумнівних переваг (особливо для служби підтримки).

Кешування дозволяє збільшувати продуктивність веб-систем за рахунок використання збережених раніше даних, на кшталт відповідей на мережеві запити або результатів обчислень. Завдяки кешу, при черговому зверненні клієнта за одними і тими ж даними, сервер може обслуговувати запити швидше. Кешування - ефективний архітектурний патерн, так як більшість програм часто звертаються до одних і тих же даних і інструкцій [3].

Кешування може виконуватися на двох рівнях:

– клієнтське кешування;

– серверне кешування.

Браузерне або клієнтське кешування представляє собою створення для браузера команди використовувати наявну кешовану копію. Робота такого кешування заснована на тому, що при повторному відвідуванні, браузеру віддається заголовок 304 Not Modified, а сама сторінка або картинка завантажуються з локального користувача кеша. Виходить, що ви економите на трафіку між браузером відвідувача і хостингом сайту. Відповідно, сторінка вашого сайту починає завантажуватися швидше.

Кешування файлів та зображень є одним із видів клієнтського кешування. Браузерні кешування як не можна краще підходить для сайтів, що містять велику кількість зображень: картинка не скачується кожен раз при відкритті сайту, а просто завантажується через кеш браузера. Однак в даному випадку є важливий нюанс: якщо зображення на сайті змінюється, то браузер дізнається про це не відразу, а тільки якщо почекати або скинути кеш в самому браузері. Це не дуже ефективно, якщо файл постійно змінюється і необхідно постійно віддавати його актуальну версію.

Кешування сторінок веб-системи – ще один вид клієнтського кешування. Коли сторінка вже створена, потрібно постійно відслідковувати її актуальність. Для цього ви повинні використовувати серверний кеш з відстеженням часу зміни окремих частин сторінки (якщо сторінка будується з безлічі динамічно генеруючих блоків).

При такому підході в кожній відповіді від сервера встановлені спеціальні заголовки, що позначають час зміни сторінки, які потім відправляються браузером користувача при повторному зверненні до сторінки сайту. Сервер при отриманні таких заголовків може проаналізувати поточний стан сторінки (можливо, навіть відмалювати її), але замість вмісту сторінки віддати заголовок «304 Not Modified», що для призначеного для користувача браузера буде означати, що можна показати сторінку зі свого (браузера користувача) кеша.

Звичайно, можна відправляти відповідні заголовки без використання серверного відстеження кеша, але в такому випадку більшість користувачів отримають оновлення контенту сторінки досить пізно. При такому підході браузер іноді опитує сервер для отримання оновлень, але періодичність і правила для кожного браузера налаштовуються його розробником, тому сподіватися на те, що ваші користувачі отримають оновлення вчасно, не доводиться.

Як правило, кеш підрозділяється за типом користувачів:

- для авторизованих;
- для неавторизованих.

Цей поділ обумовлено унікальністю контенту, для кожного авторизованого користувача і спільністю контенту для гостьових користувачів. У більшості сайтів не авторизований користувач не може змінювати вміст сайту, а значить і впливати на його вміст.

Браузерний кеш дозволяє економити трафік і час, що витрачається на завантаження сторінок. Але для досягнення ефекту економії, користувач повинен хоча б один раз відвідати нашу сторінку, а це означає, що навантаження на серверні ресурси зменшиться, але не значно.

Під серверним кешуванням розуміються всі види кешування, при якому дані зберігаються на серверній стороні. Ці дані не доступні клієнтським браузерам.

Кеш створюється і зберігається за принципом «один до багатьох» . Багато, в даному випадку, - це клієнтські пристрої (рис. 2.1).

Кешування сторінок цілком має перевагу в тому, що віддача сторінки відбувається практично в момент звернення, як наслідок - це можливість обробки мільйонів запитів навіть на самому слабкому сервері зі швидкістю роботи пам'яті і з незначним залученням процесора.

Але і у цього типу кеша є свої мінуси: наприклад, неможливість кешувати сторінки для авторизованого користувача, або користувача, вміст сторінки якого залежить від поточних змінних користувача.

Рисунок 2.1 – Ілюстрація серверного кешування

Використання цього кеша можливе у випадку, якщо серверу відомі всі статичні стану зовнішніх даних, такі як: uri, get (без додаткових параметрів), користувач не авторизований - тобто, фактично, це ідеальний стан сторінки для гостьових користувачів.

Враховуйте той факт, що при такому кешуванні архітектура сайту або програми завжди повинна обробляти вхідні запити та віддавати однотипні відповіді. Такий стан є в будь-якому додатку або сайті, його потрібно лише відстежити і застосувати до нього кеш.

Кешування сторінок цілком, найчастіше, застосовують в якихось екстрених випадках, при цьому кеш сторінок зберігається на заздалегідь зазначений час (від 2 хвилин), протягом якого відповіді від сервера однотипні.

Кешування окремих блоків сторінки є найскладнішим видом. Проте, він теж може бути ефективним, і на його прикладі найлегше пояснити принципи кешування в цілому.

Необхідно відстежувати: стан таблиць, стан сесії користувача, вимикати чи кешування при POST або GET запитах (http query), залежність від поточної адреси, сталість кешування (при зміні попередніх умов) або його динамічну підстроювання.

Кешування окремих блоків сторінок краще за інших типів кешування підійде, якщо вам потрібно, наприклад, зменшити кількість запитів до бази даних від реальних

(авторизованих) користувачів. До речі, при правильно заданих залежностях, він буде працювати навіть ефективніше, ніж всі наступні види кешування.

2.2 Асинхронна обробка даних

Комп'ютерні програми часто мають справу з тривалими процесами. Наприклад, отримують дані з бази або виробляють складні обчислення. Поки виконується одна операція, можна було б завершити ще кілька. А бездіяльність призводить до зниження продуктивності і збитків. Асинхронне програмування збільшує ефективність, тому що не дозволяє блокувати основний потік виконання.

Асинхронність була завжди, але в останні роки цей стиль розробки став особливо популярним. Всі сучасні мови мають інструменти для його реалізації і постійно покращують їх.

У синхронному коді кожна операція чекає закінчення попередньої. Тому вся програма може зависнути, якщо одна з команд виконується дуже довго.

Асинхронний код прибирає блокуючу операцію з основного потоку програми, так що вона залишиться активною, але десь в іншому місці, а обробник може йти далі. Простіше кажучи, головний «процес» ставить завдання і передає її іншому незалежному «процесу».

Асинхронне програмування успішно вирішує безліч завдань. Одна з найважливіших - доступність інтерфейсу користувача.

Візьмемо для прикладу систему, яка підбирає фільм за зазначеними критеріями. Після того як користувач вибрав параметри, програма відправляє запит на сервер. А там відбувається підбір відповідних картин. Обробка може тривати досить довго. Якщо система працює синхронно, то користувач не зможе взаємодіяти зі сторінкою, поки не прийде результат. Він не зможе навіть прокрутити сторінку.

Асинхронний код дозволяє приховати від користувача ці неприємні ефекти і зберегти активність сторінки. Після того як дані завантажаться, програма виведе їх на екран. В цьому випадку головний потік виконання поділяється на дві гілки. Одна з них продовжує займатися інтерфейсом, а інша виконує запит. (рис.2.2)

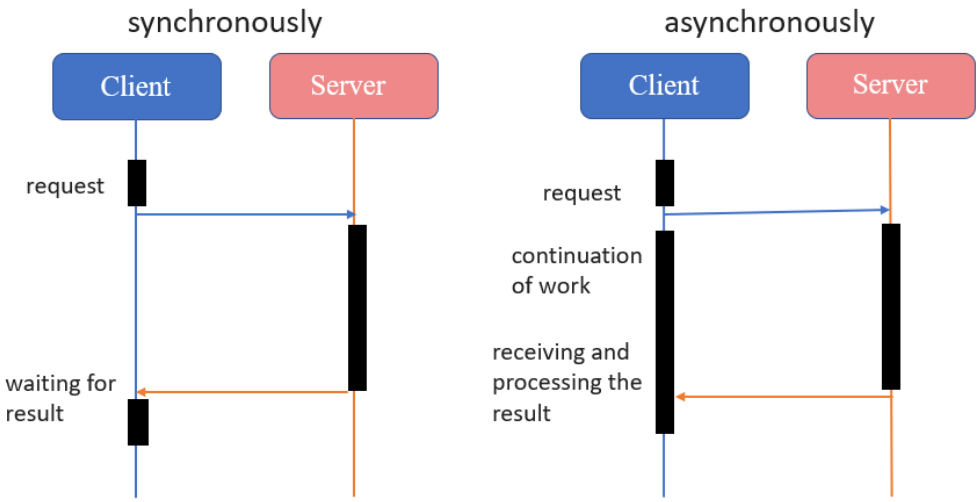


Рисунок 2.2 – Схема асинхронної та синхронної обробки даних

Тут виникає проблема. Коли запит завершиться в додатковій гілці, як про це дізнається головна? Як повернути отримане значення в основний потік, якщо це необхідно? Для цього існують події та механізм зворотного виклику. Якщо запит виконується асинхронно, то він може оповістити всіх охочих про своє закінчення. Програма підписується на це повідомлення і реєструє для нього обробник. Коли прийде час, запит створить подію і повідомить. Оброблювач продовжує виконувати наступний код, поки не отримає повідомлення. Тоді він перерветься і обробить його.

Асинхронних операцій в програмі може бути кілька. Щоб розібратися з численними подіями, існує спеціальна черга. Вона працює за принципом «перший прийшов - перший пішов».

Обробник події сам по собі може блокувати потік виконання коду. Наприклад, якщо всередині нього синхронно виконуються складні операції. Це повертає нас до проблеми очікування і зависання програми. Щоб уникнути блокування, можна зробити ще один зворотний виклик.

Події виконання і зворотні виклики - це класична схема асинхронної моделі. Так вона реалізована в більшості мов. Однак у неї є ряд недоліків. Зараз існують більш зручні інструменти для роботи з асинхронністю. Їх можна розділити на дві групи. Перша з них повертає «обіцянки». Сюди відносяться deferred, promises і futures. Друга реалізує асинхронність обчислень. Це конструкції з ключовими словами async / await. Вперше ця архітектура виникла в C #, але її переваги швидко оцінили в інших мовах.

Коли мова заходить про асинхронність,необхідно розглянути ще три близькі поняття. Це конкурентність (concurrency), паралелізм (parallel execution) і багатопоточність (multithreading). Всі вони пов'язані з одночасним виконанням завдань, проте це не одне і те ж.

Поняття конкурентного виконання саме загальне. Воно буквально означає, що безліч завдань вирішуються в один час. Можна сказати, що в програмі є кілька логічних потоків - по одному на кожну задачу. При цьому потоки можуть фізично виконуватися одночасно, але це не обов'язково. Завдання при цьому не пов'язані один з одним. Отже, не має значення, яка з них завершиться раніше, а яка пізніше.

Паралельне виконання зазвичай використовується для поділу одного завдання на частини для прискорення обчислень. Наприклад, потрібно зробити кольорове зображення чорно-білим. Обробка верхньої половини не відрізняється від обробки нижньої. Отже, можна розділити цю задачу на дві частини і роздати їх різним потокам, щоб прискорити виконання в два рази. Наявність двох фізичних потоків тут принципово важлива, так як на комп'ютері з одним обчислювальним пристроєм (процесорним ядром) такий прийом провести неможливо.

У багатопоточності потік є абстракцією, під якою може ховатися окреме ядро процесора. Деякі мови навіть мають власні об'єкти потоків. Таким чином, ця концепція може мати принципово різну реалізацію.

Ідея асинхронного виконання полягає в тому, що початок і кінець однієї операції відбуваються в різний час в різних частинах коду. Щоб отримати результат, необхідно почекати, причому час очікування може виявитися непередбачуваним.

Можна виділити три найпопулярніші схеми асинхронних запитів: послідовне виконання, паралельне виконання та конкурентне виконання (рис.2.3).

Послідовне виконання використовується для пов'язаних завдань, які потрібно запускати один за одним. Наприклад, перший запит отримує назви фільмів, а другий - інформацію про них.

Паралельне виконання застосовується для вирішення незалежних завдань, коли важливо, щоб виконалися всі запити. Наприклад, дані веб-сторінки вантажаться з трьох серверів, а після цього починається рендеринг.

Конкурентне виконання використовується для вирішення незалежних завдань, коли важливо, щоб виконався хоча б один запит. Наприклад, відправка ідентичних запитів на різні сервера.

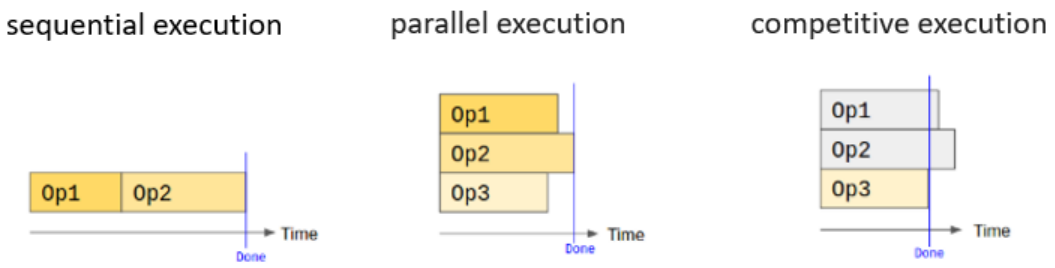


Рисунок 2.3 – Схема асинхронних запитів

2.3 Динамічне завантаження контенту як підхід для оптимізації показників продуктивності

Ajax-завантаження даних при прокрутці сторінки дає можливість позбутися від необхідності завантажувати з бази даних відразу весь контент, що значно збільшить швидкість завантаження всієї сторінки і зменшить навантаження на сервер баз даних.

Звичайно, як альтернативу цьому, можна розбити контент на сторінки і реалізувати посторінкову навігацію, але при цьому кожен раз доведеться перезавантажувати сторінку повністю, разом з усіма js файлами і файлами стилів. Більш того, користувачам доведеться виконувати більше дій, що хоч і трохи, але все ж знижує рівень зручності використання.

Переваги такого методу очевидні. Для того, щоб отримати додаткову порцію контенту вам немає необхідності переходити на іншу сторінку, що також збиває вашу увагу і змушує дивитися в іншу область сторінки. Додавши можливість динамічного завантаження ви утримуєте увагу користувача на одній і тій же області в процесі читання.

Динамічне завантаження ефективне не у всіх ситуаціях. Наприклад, у випадках, коли користувач переходить за посиланням, а потім хоче повернутися на попередню сторінку за допомогою історії браузера, він втрачає позицію в тій частині, яка була довантажена за допомогою AJAX. Одним із способів вирішення подібної проблеми - відкривати всі посилання в новому вікні або вкладці.

Недолік, який випливає з вищесказаного - у користувача немає можливості запам'ятати свою позицію в частині документа, яка була довантажена за допомогою AJAX. Уявімо, що ви хочете поділитися чимось зі сторінки з динамічним завантаженням зі своїм другом по електронній пошті. Ви не зможете цього зробити, так як посилання

приведе вас назад до вашої початкової позиції. Так що, перш ніж впроваджувати динамічне завантаження на вашому сайті, необхідно подумати про зручність використання.

2.4 Посторінкова розбивка як підхід до оптимізації показників продуктивності

Сьогодні майже всі дані, доступні в Інтернеті, зберігаються в базах даних. Розглянемо абстрактний інтернет-магазин: в ньому список категорій і товарів магазину зберігається у вигляді таблиці записів, кожен рядок якої відповідає одній категорії або товару. І якщо в разі категорій така таблиця, як правило, не перевищує 20-30 рядків, то товарів можуть бути сотні, тисячі і навіть десятки тисяч. При таких обсягах даних виникає ряд проблем:

– дуже великі обсяги сторінок;

– час завантаження сторінки навіть при швидкісних з'єднаннях вимірюється хвилинами;

– навантаження на базу даних при відображенні списку записів стає невиправдано високою.

Саме ці проблеми і повинна вирішувати посторінкова розбивка. Однак, якщо не приділити цьому елементу достатньо уваги, то він може серйозно зіпсувати зручність використання сайту, і користувач більше ніколи не повернеться до такого сайту.

Найбільш популярними варіантами посторінкової розбивки є: необмежений список сторінок (рис. 2.4), навігаційні посилання (рис. 2.5), обмежений список сторінок з навігаційними посиланнями (рис. 2.6), обмежений список сторінок з плаваючою групою (рис. 2.7) та фонове завантаження наступної сторінки.

Необмежений список сторінок є найбільш простим з точки зору програмування та використання, доки кількість сторінок не сильно велика. Користувач може бачити скільки всього доступно сторінок та легко перейти на будь-яку з них. Але якщо кількість сторінок досягає, наприклад, 200, тоді не можна сказати, що такий спосіб є зручним у використанні. Перевагами даного методу посторінкової розбивки є :

– користувач має можливість бачити, скільки всього сторінок доступно;

– користувач з легкістю може перейти на будь-яку сторінку;

– можна отримати посилання на будь-яку сторінку списку.

Недоліком можна вважати складність використання для великої кількості сторінок.

Рисунок 2.4 – Приклад посторінкової розбивки «Необмежений список сторінок»

Навігаційні посилання застосовується для сайтів, де не стоїть завдання вибору однієї записи з представленого списку, а кількість записів не грає ролі. Прикладом такого сайту є блог.

Слід зауважити, що навігаційні посилання також часто застосовуються разом з попереднім варіантом посторінкової розбивки, але в даному випадку мається на увазі саме їх самостійне застосування.

Перевагами даного варіанту посторінкової розбивки є:

– можна отримати посилання на будь-яку сторінку списку;
– простота навігації за допомогою кнопок «Назад» та «Далі».

Недоліками даного варіанту посторінкової розбивки є:

– незрозуміло, скільки усього сторінок;
– користувач не в змозі перейти на певну сторінку.

Рисунок 2.5 – Приклад посторінкової розбивки «Навігаційні посилання»

Обмежений список сторінок із навігаційними посиланнями є комбінацією попередніх двох варіантів посторінкової розбивки: необмежений список сторінок та навігаційні посилання. Номери сторінок представлені групами по 5, 10, 15 посилань, а посилання навігації дозволяють переміщатися між групами. Недолік у тому, що, перебуваючи на першій сторінці, неможливо відразу перейти, наприклад, на сторінку 231 без додаткових навігаційних елементів. Такий варіант розбивки вважається застарілим.

Перевагами даного варіанту посторінкової розбивки є:

– дозволяє вмістити будь-яку кількість сторінок;
– можна перейти на будь-яку сторінку списку;
– можна отримати посилання на будь-яку сторінку списку.

Недоліками даного варіанту посторінкової розбивки є:

– важкий перехід на довільну сторінку списку;

– перебуваючи в межах однієї групи неможливо оцінити розміри інших груп, та й кількість сторінок в цілому.

Рисунок 2.6 – Приклад посторінкової розбивки «Обмежений список сторінок із навігаційними посиланнями»

Обмежений список сторінок з плаваючою групою є удосконаленим варіантом попередньої посторінкової розбивки. Тут також є група сторінок, однак вона відраховується від активної в даний момент сторінки. Якщо ви перебуваєте на першій сторінці, то бачите сторінки від 1 до 10, якщо ж ви перебуваєте на сторінці 10, то бачите сторінки від 5 до 15.

Перевагами даного варіанту посторінкової розбивки є:

– дозволяє вмістити будь-яку кількість сторінок;

– можна перейти на будь-яку сторінку списку;

– можна отримати посилання на будь-яку сторінку списку.

Недоліками даного варіанту посторінкової розбивки є:

– важко перейти на певну сторінку списку.

Рисунок 2.7 – Приклад посторінкової розбивки «Обмежений список сторінок з плаваючою групою»

З приходом ери веб-систем з'явилися нові види посторінкових разбивок. У варіанті фонового завантаження наступної сторінки немає ніяких номерів сторінок, тільки кнопка «завантажити більше» в кінці списку. Коли користувач натискає на цю кнопку, наступна сторінка завантажується у фоновому режимі та нові записи додаються в кінець списку. У випадку досягнення кінця списку, кнопка зникає, а користувач отримує повний список

записів. Перевагами даного варіанту посторінкової розбивки є навігація у списку істотно спрощена в порівнянні з традиційними рішеннями.

Недоліками даного варіанту розбивки є:

- незрозуміло, скільки всього сторінок;
- неможливо перейти на довільну сторінку списку;
- неможливо отримати посилання на довільну сторінку списку.

3 ОГЛЯД «ВУЗЬКИХ МІСЦЬ» ТЕСТУВАЛЬНОЇ СИСТЕМИ ТА ВИМОГИ ДО ПОКАЗНИКІВ ПРОДУКТИВНОСТІ

3.1 Предметна область для проведення навантажувального тестування

Для проведення навантажувального тестування була обрана web-система бібліотека. Sierra – це бібліотечна система, яка забезпечує автоматизовані робочі потоки, інтегроване управління ресурсами та відкритий доступ до даних. Вона модернізує традиційні бібліотечні робочі процеси з революційним дизайном та зручним веб-доступом. Sierra вирішує багато проблем, з якими сьогодні стикаються бібліотеки, з її здатністю безперешкодно взаємодіяти з іншими системами. Sierra підключається до інтерфейсів електронної торгівлі, систем управління курсовою програмою та широкого кола сторонніх постачальників цифрового контенту, які істотно розширюють її функціональність. Sierra – система для бібліотекарів, яка забезпечує облік та статистику виданих та зданих користувачами бібліотеки матеріалів.

Основними функціями системи є:

– Catalog - функція, що дозволяє бібліотекарю виконати пошук бажаної книги (або іншого бібліотечного матеріалу), створити список необхідних матеріалів, шляхом застосування запиту або додативласну книгу до списку каталогу. На рисунку 3.1 можна побачити інтерфейс функції Catalog: які поля має бібліотечний матеріал та які дії можна з ним проводити. У верхній секції можна побачити поле пошуку, яке дозволяє проводити пошук по наступним критеріям: Title, Author, Record No і таке інше. В Sierra існують наступні основні типи бібліотечних матеріалів, які можуть бути створені користувачем: bibliographic record та item record.

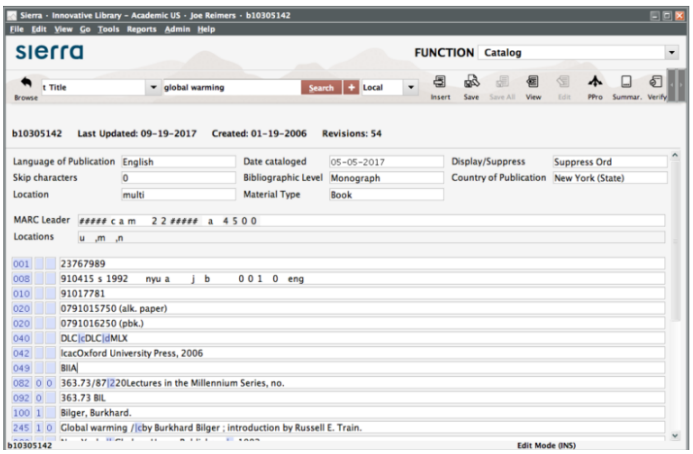


Рисунок 3.1 – Інтерфейс функції Catalog тестувальної системи Sierra

– Circulation - функція, що дає можливість бібліотекарю виконувати пошук користувача бібліотеки, заходити на його особисту сторінку та реєструвати видачу та здачу бібліотечних матеріалів. Також є можливість нарахування штрафів за нездану в час книгу і можливість бронювання матеріалу на певний час. На рисунку 3.2 можна побачити інтерфейс функції Circulation.

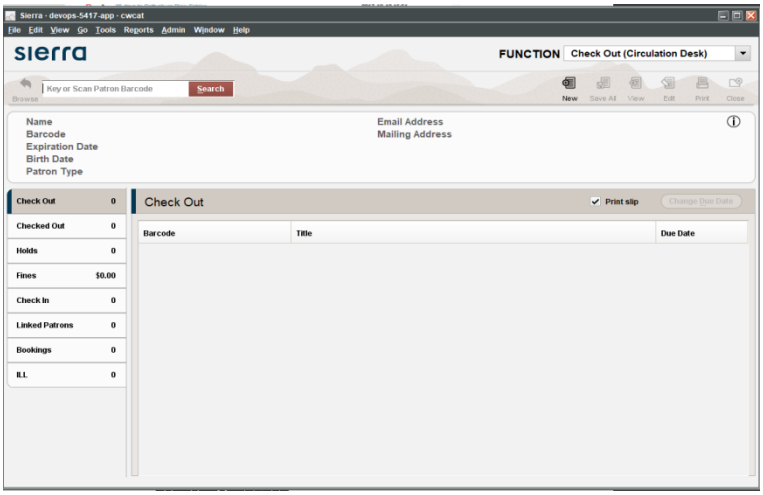


Рисунок 3.2 – Інтерфейс функції Circulation тестувальної системи Sierra

– Acquisition - функція, пов'язана з придбанням матеріалів, дозволяє бібліотекарям керувати всіма аспектами замовлення матеріалів, в тому числі введення та відправлення замовлень, отримання замовлених товарів, заява або скасування замовлення, і обробка рахунків-фактур. Ці функції також дозволяють керувати та відстежувати фінансові ресурси вашої бібліотеки, у тому числі можливість генерування та друку бухгалтерських звітів, коригування коштів, підтримка фонду та інформації постачальника, і створювати статистичні звіти про постачальників.

3.2 Огляд проблемних частин функціоналу тестувальної системи, що погіршують показники продуктивності

У ході тестування бібліотечної системи Sierra було виявлено ряд проблемних місць системи, що погіршують показники продуктивності та це призводить до втрати користувачів. Вузькі місцями системи є наступні функції.

Однією з проблемних функцій система є пошук у каталозі. Під час виконання пошуку бібліотечного матеріалу по якомусь індексу (наприклад, заголовку), що поверне

дуже велику кількість результатів, сторінка буде завантажуватись дуже довго, приблизно досягає 10+ секунд. Ця проблема призводить до втрати користувачів системи через довгу затримку при завантаженні сторінок, адже пошук у бібліотечних системах грає найважливішу роль.

Наступне вузьке місце системи - перегляд інформації про штрафи користувачів бібліотеки на сторінці Fines. Велика кількість рядків у таблиці про штрафи користувачів бібліотеки не дозволяє бібліотекарю за допомогою прокрутки сторінки побачити результати нижніх рядків сторінки, тому що виконується оновлення сторінки кожну секунду.

Ще однією функцією, що має безліч проблем є - створення файлу з відібраним бібліотечним матеріалом по введеному запиту. Ця функція системи є важливою та спрощує роботу бібліотекарю при пошуку необхідних матеріалів. Але створення файлу за запитом, який містить більше одної умови, призводить до уповільнення роботи системи.

Бронювання бібліотечного матеріалу на певний проміжок часу – важлива функція, що має ряд недоліків. Під час виконання бронювання книги на певний час, сторінка з календарем оновлюється деякий час і користувач не має можливості провести бронювання до кінця швидко.

3.3 Вимоги до показників продуктивності системи

Існують різні неофіційні спроби визначити стандарти продуктивності системи, особливо для браузерних систем. У наступному списку підсумовуються дослідження, проведені наприкінці 1980-х років, які намагалися відображати продуктивність кінцевого користувача до часу відгуку. Оригінальне дослідження було в основному засноване на системах із зеленим екраном, але його висновки все ще дуже актуальні [10].

Очікування більше 15 секунд. Для деяких типів систем певні кінцеві користувачі можуть знаходитися у системі більше 15 секунд, очікуючи відповіді на один простий запит. Проте, для зайнятого оператора call-центру затримки більше 15 секунд можуть здаватися нестерпними. Якщо такі затримки можуть виникнути, система повинна бути спроектована таким чином, щоб кінцевий користувач міг звернутися до інших заходів і виконати запит пізніше.

Очікування більше 4 секунд. Ці затримки зазвичай є занадто довгими, вимагаючи від кінцевого користувача зберігати інформацію в короткочасній пам'яті (пам'ять кінцевого користувача, а не комп'ютер!). Такі затримки перешкоджатимуть активній

роботі з вирішення проблем та перешкоджають введенню даних. Проте після завершення транзакції може бути допустима затримка від 4 до 15 секунд.

Очікування від 2 до 4 секунд. Затримка довша, ніж 2 секунди, може перешкоджати операціям, які вимагають високого рівня концентрації. Очікування від 2 до 4 секунд може здатися довгим, коли кінцевий користувач поглинається і емоційно прагне до виконання поставленого завдання. Знову ж таки, затримка в цьому діапазоні може бути прийнятною після незначного закриття. Можливо, покупці чекають від 2-4 секунд після введення їхньої адреси та номера кредитної картки, але не на більш ранній стадії, коли вони можуть порівнювати різні характеристики продукту.

Очікування менше ніж 2 секунди. Коли користувач програми повинен запам'ятовувати інформацію протягом кількох відповідей, час відповіді має бути коротким. Чим точніше інформація, яку слід запам'ятати, тим більша потреба у відповідях менше 2 секунд. Таким чином, для складних дій, таких як проглядання продуктів, які варіюються в різних параметрах, 2 секунди являє собою важливий час відповіді.

Очікування до 1 секунди. Деякі типи інтенсивної роботи (наприклад, написання книги), особливо з системами, багатими графікою, вимагають дуже короткого часу відгуку, щоб тримати тривалий час інтерес та увагу кінцевих користувачів.

Відповідь на натискання клавіші (наприклад, побачення символу, що відображається на екрані) або натискання об'єкта екрана за допомогою миші має бути майже миттєвим: менше 0,1 секунди після дії. Багато комп'ютерних ігор вимагає надзвичайно швидкої взаємодії.

Як видно, критичний бар'єр реагування, як видається, становить 2 секунди, що, цікаво, є місцем очікуваного часу реакції веб-системи. Час відгуку більше 2 секунд має певний вплив на продуктивність для середнього кінцевого користувача.

Історія вивчення впливу швидкості починається з досліджень, пов'язаних з розробкою традиційних інтерфейсів. В результаті були отримані орієнтовні показники реакції користувача програмного засобу в залежності від швидкості реакції на його дії.

Затримка реакції,мс	Сприйняття користувачем
0-100	Миттєва реакція
100-300	Невелика, але помітна затримка
300-1000	Система працює, але навантажена
1000-10000	Ймовірність переключення думок на інші задачі
10000 та більше	Задача відміняється (система не працює)

Рисунок 3.3 – Сприйняття затримки часу відгуку користувачем системи

Як видно з рисунку 3.3, швидкість реакції веб-сайту на дії користувача повинна бути обмежена рамками в 300 мс для збереження відчуття залученості в процес. Для електронного бізнесу процесом може вибір і покупка товару. У разі, якщо цей процес переривається, з'являється ризик зниження коефіцієнта конверсії на сайті. Наприклад, покупець може перейти на магазин конкурента, або вирішити відкласти покупку.

Якою має бути швидкість роботи сайтів електронного бізнесу в реальності є складним питанням. Вимірювання часу відтворення сторінки в браузері користувача залежить від декількох факторів:

- клієнтська продуктивність;
- можливості та стан інтернет-каналу від клієнта до сервера;
- серверна продуктивність.

Кожен з цих факторів включає в себе безліч компонентів, в результаті чого картина швидкості завантаження сайту у реальних користувачів відрізняється крайньою різноманітністю. Зміна може відбуватися в довільні моменти часу в залежності від завантаження каналів і мережевого устаткування.

Перед командою розробки бібліотечного продукту Sierra стояли наступні вимоги до оптимізації системи:

- відсоток запитів, час відгуку яких більше 1200 мс не повинен перевищувати 5%;
- максимальний час відгуку не повинен перевищувати 5000 мс;
- збільшити середню кількість оброблюваних запитів у секунду сервером з 100 до 200 при паралельній роботі 20 користувачів.

3.4 Описання сценаріїв smoke тесту системи Sierra

Smoke test – це тестування, направлене на перевірку працездатності основних, найчастіше використовуючих користувачами функцій. Це короткий цикл тестів, що підтверджує (або заперечує) факт того, що система стартує успішно і виконує свої основні функції. Даний тип тестування дозволяє на початковому етапі виявити основні критичні дефекти.

Smoke test системи Sierra складається з наступних сценаріїв:

- авторизація у систему:
    1) ввести логін та пароль бібліотекаря у форму авторизації;
    2) натиснути кнопку «Login»;

3) переконатися, що користувач авторизувався у систему успішно.

– пошук книги, використовуючи компактний режим каталогу:

1) виставити компактний режим перегляду у налаштуваннях;

2) виконати пошук по заголовку бібліотечного матеріалу;

3) переглянути вміст знайденого матеріалу (заповнених полів книги).

– пошук книги, використовуючи розширений режим каталогу:

1) виставити компактний режим перегляду у налаштуваннях;

2) виконати пошук по заголовку бібліотечного матеріалу;

3) переглянути вміст знайденого матеріалу (заповнених полів книги).

– видання книги користувачу та зміна дати здачі:

1) виконати пошук користувача бібліотеки, який бажає отримати книгу та перейти на особистий кабінет користувача;

2) виконати пошук бажаного бібліотечного матеріалу;

3) виконати операцію видання книги шляхом сканування id-номеру матеріалу;

4) змінити дату здачі книги.

– повернення книги на сторінці користувача:

1) виконати пошук користувача бібліотеки, який бажає отримати книгу та перейти на особистий кабінет користувача;

2) виконати пошук бажаного бібліотечного матеріалу;

3) виконати операцію повернення книги шляхом сканування id-номеру виданого матеріалу на сторінці користувача;

4) переконатися, що користувач повернув книгу вчасно та штраф не був нарахований (у випадку, якщо користувач повернув книгу не вчасно, переконатися, що штраф був нарахований правильно відносно виставлених налаштувань).

– повернення книги у режимі «без користувача»:

1) перейти у функцію «Check in (No Patron)» (здача книги без користувача);

2) виконати операцію повернення книги шляхом сканування id-номеру виданого матеріалу.

– бронювання бібліотечного матеріалу на певний проміжок часу:

1) перейти у функцію «Booking» (бронювання);

2) знайти необхідну книгу для бронювання шляхом пошуку по id-номеру;

3) обрати у списку користувача, який бажає забронювати книгу;

4) обрати дати початку та завершення бронювання;

5) звернути увагу, що книга не заброньована іншим користувачем на цю дату (якщо вона зайнята на цей період часу, необхідно обрати інший екземпляр книги, або відмінити бронювання у іншого користувача).

Для проведення навантажувального тестування було обрано наступні сценарії із smoke-тесту, які є набільш розповсюдженими серед бібліотекарів:

- авторизація у систему;
- пошук книги, використовуючи розширений режим каталогу;
- видання книги користувачу та зміна дати здачі;
- повернення книги у режимі «без користувача»;
- бронювання білбіотечного матеріалу на певний проміжок часу.

4 ПРОВЕДЕННЯ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Для проведення навантажувального тестування системи буде використовуватися фреймворк Gatling. Цей інструмент підтримує мову розробки Scala і створений для навантажувального та стрес-тестування. Gatling реалізував повністю нову архітектуру для інструменту тестування продуктивності, щоб бути більш ресурсомістким [5]. Це дає змогу імітувати велику кількість запитів за секунду за допомогою однієї машини.

Звіт Gatling є цінним джерелом інформації для читання даних про продуктивність системи, надаючи деякі подробиці про запити та час відгуку. Основною із цілей використання Gatling – це виявлення bottlenecks (вузьких місць) системи.

У Gatling існує дуже гарна опція recorder, яка забезпечує створення сценаріїв.

Recorder працює як проксі, захоплюючи весь трафік і перетворюючи йогов сценарій Gatling. Як тільки симуляція записується, вона може бути змінена відповідними значеннями, кількістю користувачі та інше.

Під «симуляцією» розуміється фактичний тест. При моделюванні об’єкт протоколу HTTP створюється та конфігурується відповідними значеннями як URL, параметрами заголовку запиту, автентифікацією, кешуванням і т.д. Моделювання має один або декілька «сценаріїв».Сценарій представляє собою серію запитів HTTP з різними параметрами дій (POST / GET) та запиту. Сценарій налаштовується з рахунком кількості користувачів навантаження та шаблонів нарощування. Це налаштування описується у методі «setUp» Simulation. Декілька сценаріїв можуть сформувати одну симуляцію. Існують інші елементи, такі як Feeders, які створюють вхідні дані та перевірки, які використовуються для перевіркиresponses (відповідей).

Зазвичай розробка тестового сценарію - це двоетапна робота:

1) записати скрипт тесту, враховуючи трафік між клієнтом та сервером;

2) налаштувати тестовий скрипт (додати складну логіку, перевірки, параметризацію тощо).

На рисунку 4.1 наведено типовий випадок зв'язку між користувацьким браузером та деяким веб-сайтом, доступним через мережу.

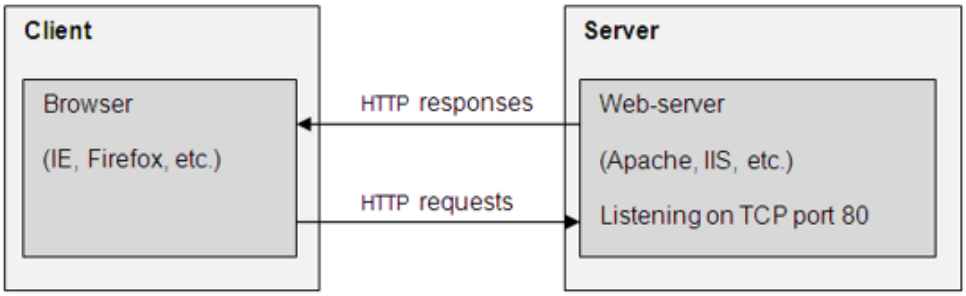


Рисунок 4.1 – Приклад взаємозв'язку між між користувацьким браузером та деяким веб-сайтом

Для запису тестового скрипту необхідно додати модуль взаємодії між клієнтом і сервером. Деякі інструменти реалізують це через проксі-сервери, деякі інші інструменти роблять це автоматично (рис.4.2).

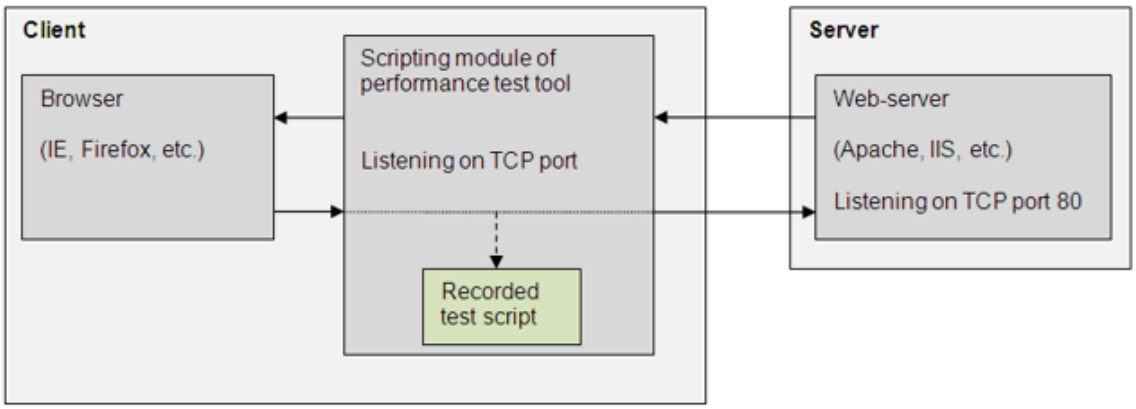


Рисунок 4.2 – Приклад взаємозв'язку між між користувацьким браузером, веб-сайтом та тестовим скриптом

Було проведено навантажувальне тестування на системі ранньої версії та на актуальній, після впровадження підходів для оптимізації показників продуктивності.

Нижче будуть приведені звіти тестування при використанні сценаріїв із smoke тесту та проведений аналіз показників продуктивності на системі двох версій : первинній та після оптимізації. Навантажувальне тестування буде проведено при імітуванні навантаження у 20 та 60 користувачів.

4.1 Результати показників продуктивності при імітуванні паралельної роботи 20 користувачів

Після створення сценарію, використовуючи опцію recorder та змінивши 1 віртуального користувача (значення по замовчуванню) на 20 у функції Setup, запускаємо навантажувальний тест та отримуємо звіти, які будуть описані нижче.

Звіт Indicators Graphic містить розподіл responses (відповідей) у наступних інтервалах часу: менше 800мс, 800 мс − 1200 мс, більше ніж 1200 мс та збої. Цей звіт дає загальний огляд продуктивності системи. Якщо найвищий відсоток відповідей менше 800 мс − це досить гарна індикація продуктивності. Як можна побачити на графіку, що відповідає системі первинної версії (рис. 4.3), найбільша кількість запитів, 70%, потрапила у інтервал менше ніж 800мс, але є певна кількість запитів, 19%, що потрапили в інтервал 800мс<t<1200мс та 11% запитів, які знаходяться у інтервалі >1200мс. Тому можна зробити висновок, що система до оптимізації при навантаженні у 20 користувачів, має рівень якості продуктивності середній.

Рисунок 4.3 – Звіт «Indicators Graphic» на первинній версії системи при імітуванні паралельної роботи 20 користувачів

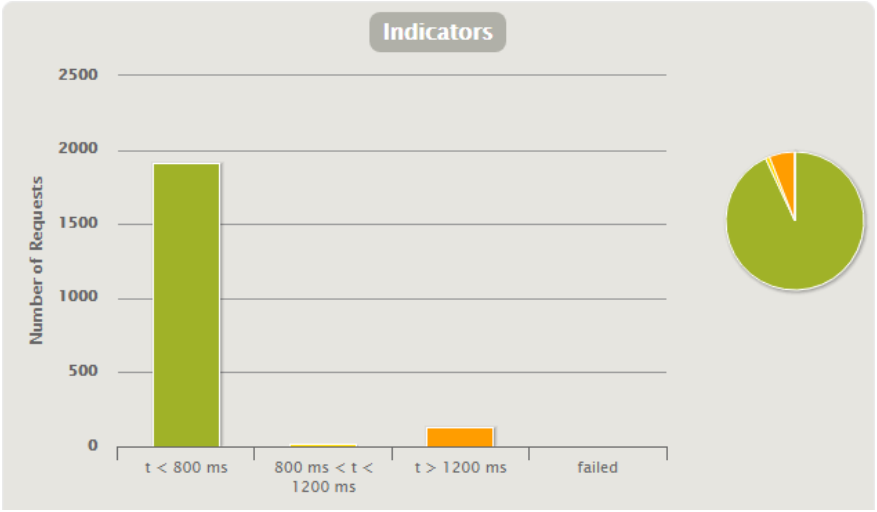


Рисунок 4.4 – Звіт «Indicators Graphic» на актуальній версії системи при імітуванні паралельної роботи 20 користувачів

Щодо графіку (рис.4.4), який відповідає системі актуальної версії, можна побачити, що найбільша кількість запитів, 93%, потрапила у інтервал менше ніж 800мс, але є певна кількість запитів, 1%, що потрапили в інтервал 800мс<t<1200мс та 6% запитів, які у інтервалі >1200мс. Це говорить про те, що рівень якості продуктивності системи актуальної версії високий при навантаженні у 20 користувачів по зрівнянню з попереднім графіком.

Indicator Graphic показує, що, вимога: «відсоток запитів, час відгуку яких більше 1200 мс, не повинен перевищувати 5%» при одночасному використанні системи 20 користувачами була виконана командою розробників.

Звіт «Responsive time distribution» відображає детальний розподіл відповідей в невеликих тимчасових інтервалах. Цей звіт схожий на звіт «Indicators», але є більш докладним, оскільки інтервали часу дуже малі.

Цей звіт дає краще уявлення про продуктивність, тому що є можливість побачити, який відсоток запитів знаходиться у заданому інтервалі часу.

Графік також включає в себе запити про помилки. Відсоток усіх запитів, зроблених під час виконання тесту відображається по осі Y. Він буде включати в себе як запити, що були відправлені успішно, так і запити із збоями. Всі значення Y повинні складати 100%.

Час відгуку (час, який потрібен, щоб запросити сторінку і відправити дані назад на сервер, щоб визнати, чи отримали ви його) на осі x. По мірі збільшення навантаження на сервер буде видно, як дані на цьому графіку переміщаються далі вправо (час відповіді буде повільніший).

Нижче будуть представлені звіти для систем первинної версії та актуальної версії.

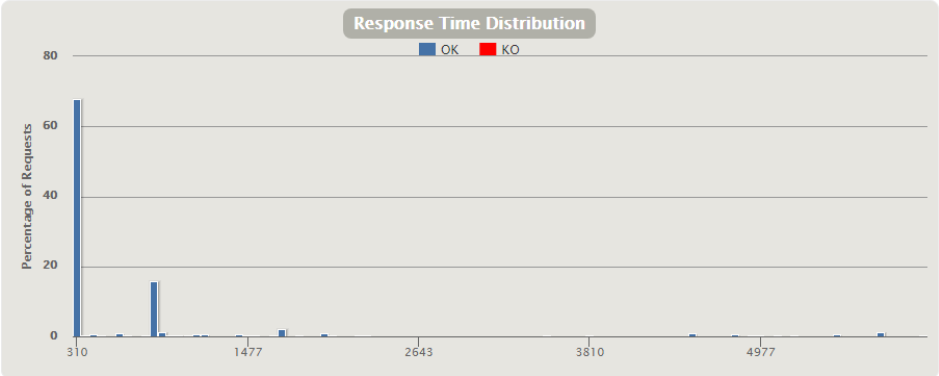


Рисунок 4.5 – Звіт «Response time distribution» на первинній версії системи при імітуванні паралельної роботи 20 користувачів

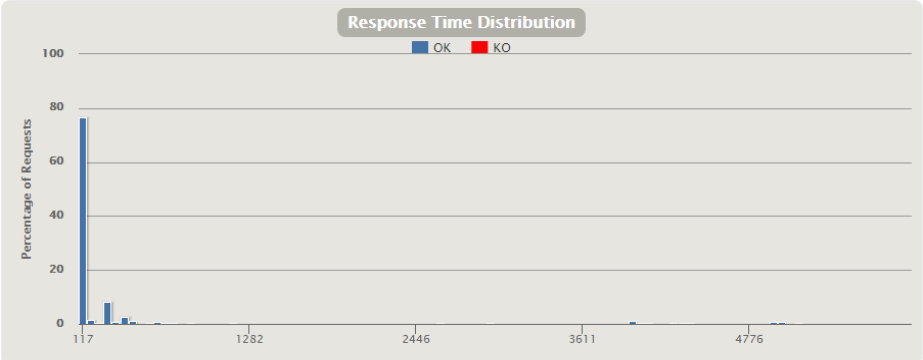


Рисунок 4.6 – Звіт «Response time distribution» на актуальній версії системи при імітуванні паралельної роботи 20 користувачів

Як видно з графіку для первинної версії системи (рис.4.5) найбільша кількість запитів була відправлена, майже 20% у перші 835мс, це означає, що 20% запитів було відправлено при проходженні сценарію «пошук книги у розширеному режимі». На відміну від системи актуальної версії, де кількість запитів рівномірно розподілена при проходженні сценаріїв, на графіку Response time distribution не видно великих перепадів. (рис. 4.6).

Звіт «Active users along the simulation» показує кількість віртуальних користувачів, що відправляли запити в кожен момент часу протягом моделювання. Для системи було обрано навантаження в 20 користувачів, тому для системи первинної версії (рис. 4.7) та актуальної версії (рис. 4.8) на графіках показано 20 активних віртуальних користувачів.

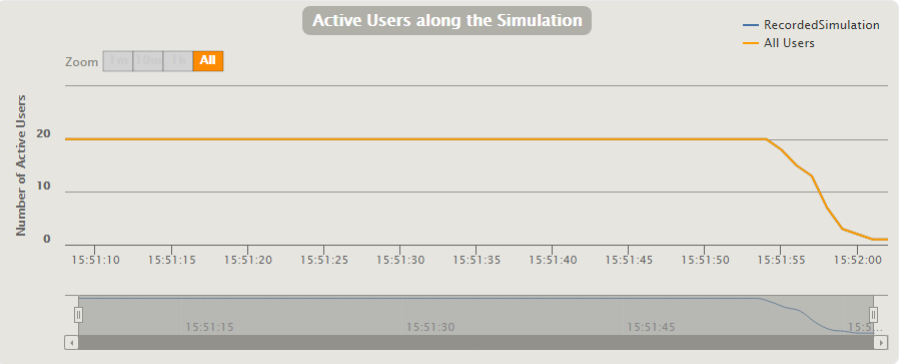


Рисунок 4.7 – Звіт «Active users along the simulation» на первинній версії системи при імітуванні паралельної роботи 20 користувачів

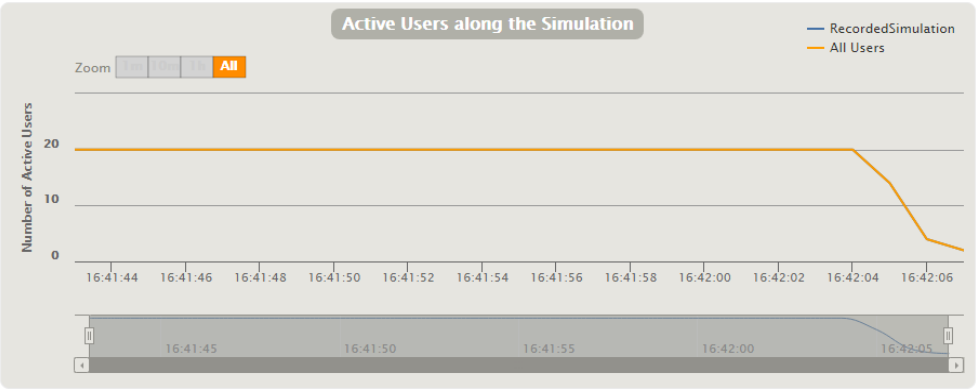


Рисунок 4.8 – Звіт «Active users along the simulation» на первинній версії системи при імітуванні паралельної роботи 20 користувачів

Звіт «Response Time Percentiles over Time» відображає мінімальний та максимальний час запиту в кожен момент при моделюванні тільки для успішних запитів. Існує також групування запитів у відсотках, що показує, який відсоток запитів займає певну кількість часу. Крім того, цей графік відображає кількість користувачів в кожен момент під час моделювання. Якщо подивитись на графік системи первинної версії (рис. 4.9), мінімальним часом запиту є 284 мс, максимальним – 6,098 мс. Для системи актуальної версії (рис. 4.10), мінімальним часом запиту є 90 мс, максимальним – 5,051мс.

Було виявлено, що при проходженні наступних сценаріїв на системі первинної версії виникали затримки:

– авторизація у систему;
– пошук книги у розширеному режимі;
– бронювання бібліотечного матеріалу на певний проміжок часу.

Щодо проходження сценаріїв smoke-тесту на системі актуальної версії, невеликі затримки виникали під час авторизації у систему та пошуку матеріалу у розширеному режимі.

Але, як видно на звіті «Response Time Percentiles over Time» для актуальної версії, він не має великих перепадів у часу відгуку та має досить гарні показники часу відгуку, на відміну від первинної системи.

Далі будуть проаналізовані показники часу відгуку відносно пройдених сценаріїв:

1) Авторизація у систему:

– максимальний час відгуку на системі первинної версії – 6,098 мс;

– максимальний час відгуку на системі актуальної версії - 5,011мс.

2) Пошук книги у розширеному режимі:

– максимальний час відгуку на системі первинної версії – 5,934 мс;

– максимальний час відгуку на системі актуальної версії - 4,051мс.

3) Видання книги користувачу та зміна дати здачі:

– максимальний час відгуку на системі первинної версії – 1,934 мс;

– максимальний час відгуку на системі актуальної версії – 1,715 мс.

4) Повернення книги у режимі «без користувача»:

– максимальний час відгуку на системі первинної версії – 1,934 мс;

– максимальний час відгуку на системі актуальної версії – 1,722 мс.

5) Бронювання бібліотечного матеріалу на певний проміжок часу:

– максимальний час відгуку на системі первинної версії –2,812 мс;

– максимальний час відгуку на системі актуальної версії – 1,010 мс.

З описаного вище детального аналізу показника часу відгуку можна зробити висновок, що затримки при проходженні сценаріїв «пошук книги у розширеному режимі» та «бронювання бібліотечного матеріалу на певний проміжок часу» було вирішено шляхом впровадження посторінкової розбивки.

Проаналізувавши отримані показники часу відгуку можна зробити висновок, що вимога, «максимальний час відгуку не повинен перевищувати 5000 мс», успішно виконана командою розробників.

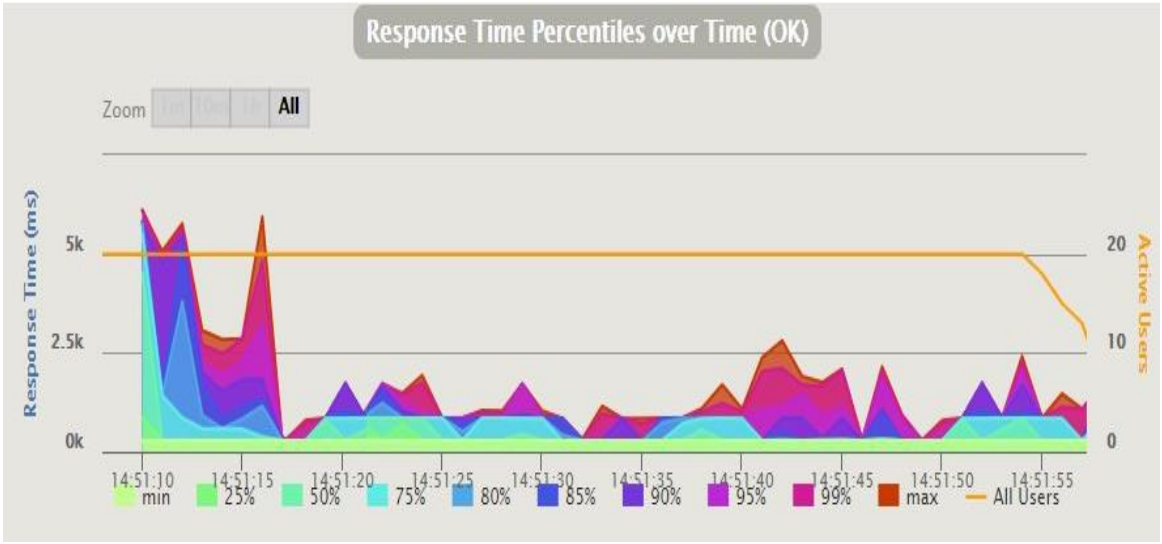


Рисунок 4.9 – Звіт «Response Time Percentiles over Time» для первинної версії системи при імітуванні паралельної роботи 20 користувачів

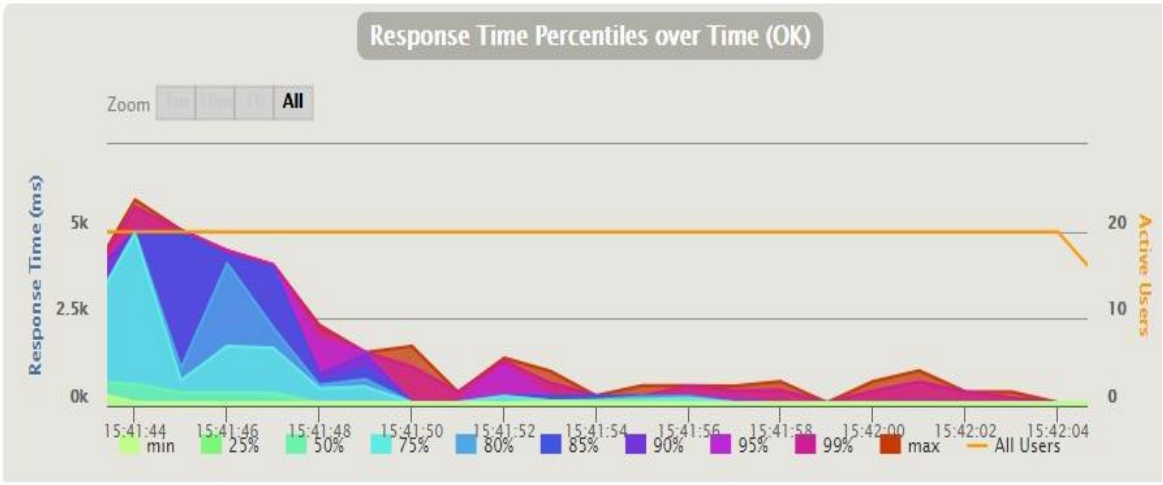


Рисунок 4.10 – Звіт «Response Time Percentiles over Time» для актуальної версії системи при імітуванні паралельної роботи 20 користувачів

Звіт «Number of requests per second» описує кількість запитів, які виконуються на сервері в кожен момент часу в процесі моделювання. Також графік показує кількість користувачів в кожен момент під час моделювання. Нижче представлені звіти для системи первинної версії (рис. 4.11) та актуальної версії (рис. 4.12).

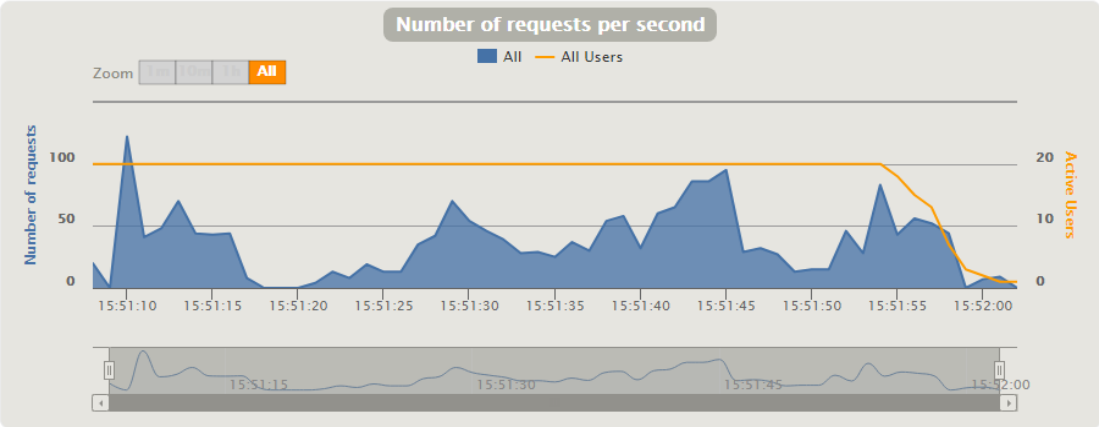


Рисунок 4.11 - Звіт «Number of requests per second» для первинної версії системи при імітуванні паралельної роботи 20 користувачів

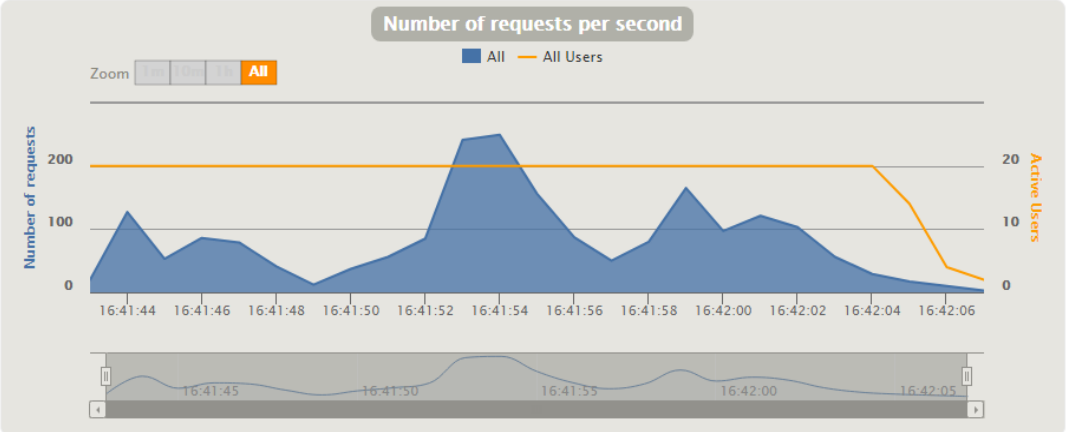


Рисунок 4.12 - Звіт «Number of requests per second» для актуальної версії системи при імітуванні паралельної роботи 20 користувачів

Звіт «Number of responses per second» відображає кількість відповідей, які виконуються на сервері в кожен момент часу в процесі моделювання. Крім того, цей графік показує кількість користувачів в кожен момент часу. Нижче представлені звіти для системи первинної версії (рис. 4.13) та актуальної версії (рис. 4.14). Як можна побачити на графіку для актуальної версії, вимога щодо збільшення середньої кількості оброблюваних запитів у секунду сервером з 50 до 100 була виконана командою розробників.

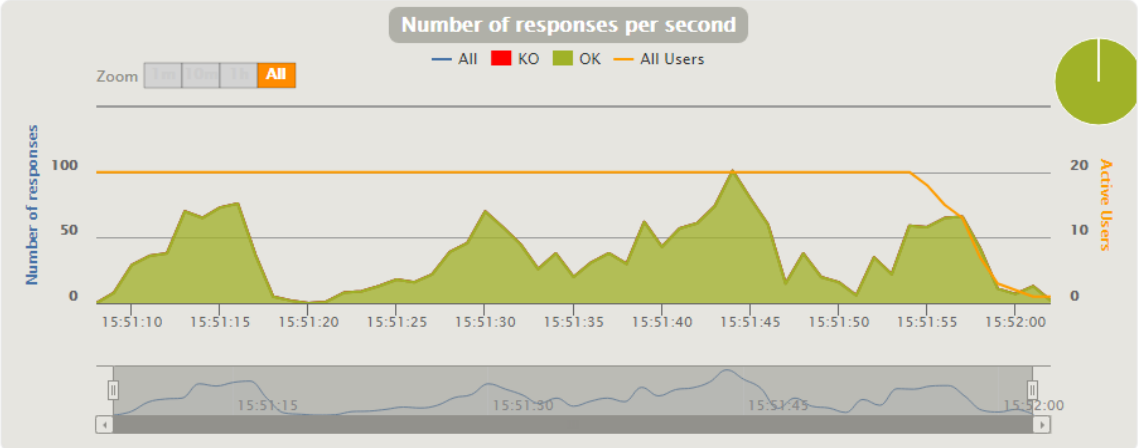


Рисунок 4.13 - Звіт «Number of requests per second» для первинної версії системи при імітуванні паралельної роботи 20 користувачів

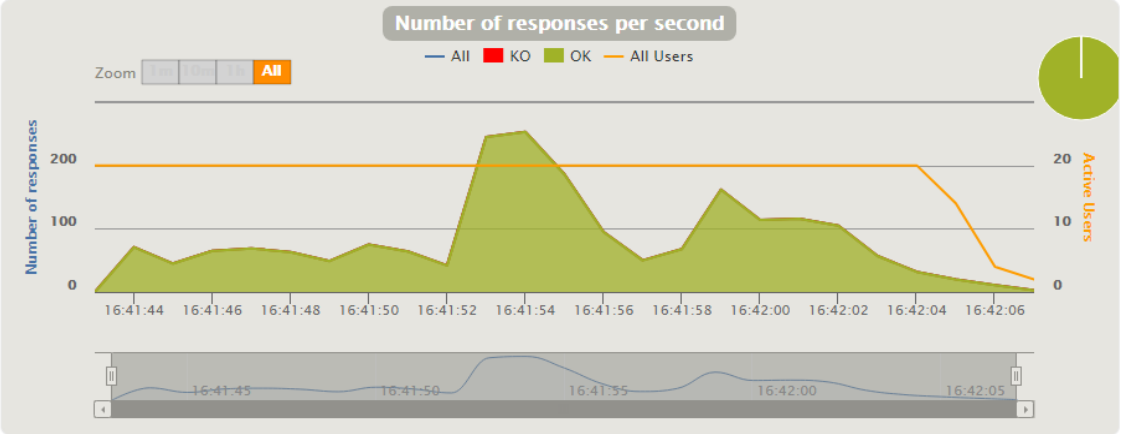


Рисунок 4.14 - Звіт «Number of requests per second» для актуальної версії системи при імітуванні паралельної роботи 20 користувачів

4.2 Результати показників продуктивності при імітуванні паралельної роботи 60 користувачів

Після створення сценарію, використовуючи опцію recorder та змінивши 1 віртуального користувача (значення по замовчуванню) на 60 у функції Setup, запускаємо навантажувальний тест та отримуємо звіти, які будуть описані нижче.

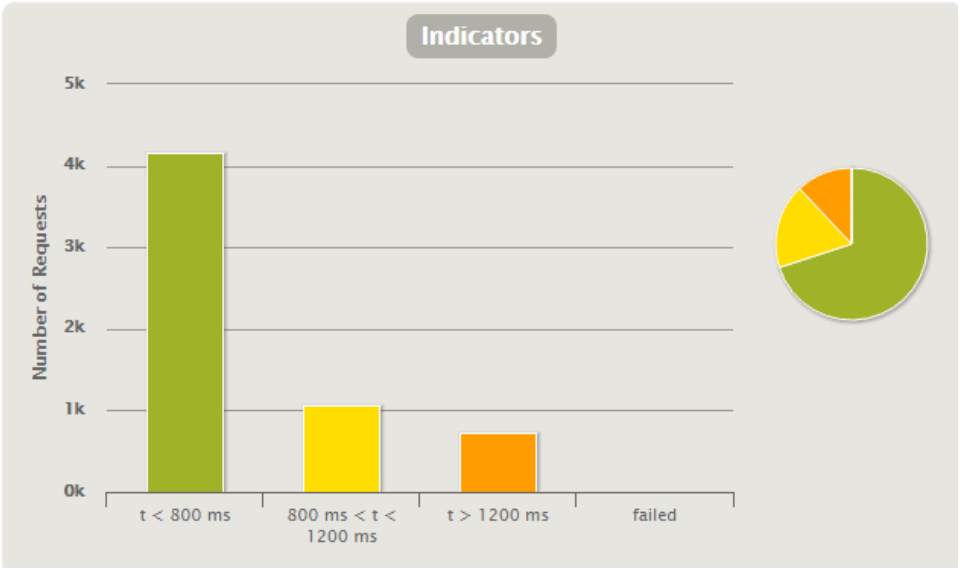


Рисунок 4.15 – Звіт «Indicators Graphic» на первинній версії системи при імітуванні паралельної роботи 60 користувачів

Як можна побачити на графіку, що відповідає системі первинної версії (рис. 4.15), найбільша кількість запитів, 70% потрапила у інтервал менше ніж 800мс, але є певна кількість запитів, 18%, що потрапили в інтервал 800мс<t<1200мс та 12% запитів, які знаходяться у інтервалі >1200мс. Тому можна зробити висновок, що система до оптимізації при навантаженні у 20 користувачів, має рівень якості продуктивності середній.

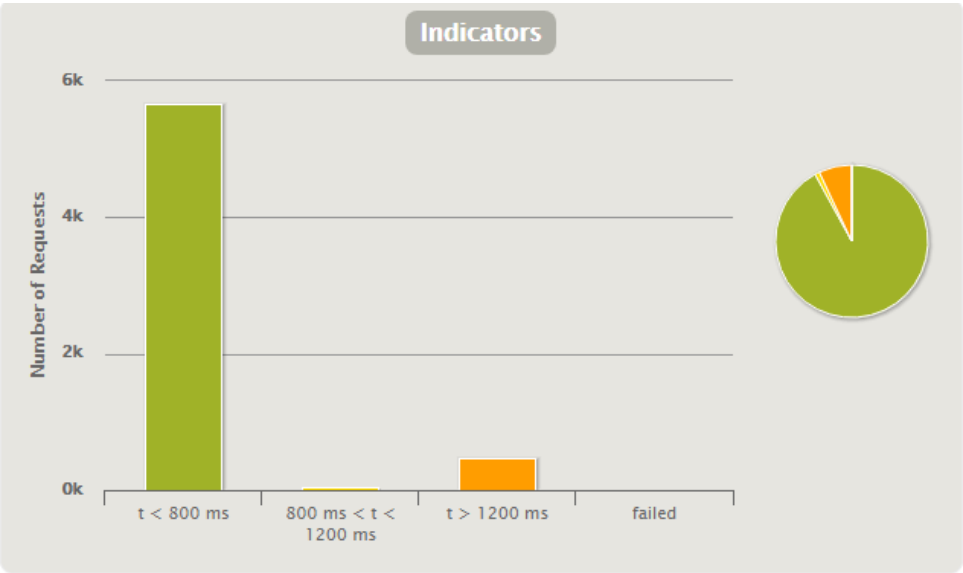


Рисунок 4.16 – Звіт «Indicators Graphic» на актуальній версії системи при імітуванні паралельної роботи 60 користувачів

Щодо графіку, що відповідає системі актуальної версії (рис. 4.16), найбільша кількість запитів, 92% потрапила у інтервал менше ніж 800мс, але є певна кількість запитів, 1%, що потрапили в інтервал 800мс<t<1200мс та 7% запитів, які знаходяться у інтервалі >1200мс. Тому можна зробити висновок, що вимога: «відсоток запитів, час відгуку яких більше 1200 мс, не повинен перевищувати 5%» при одночасному використанні системи 60 користувачами була виконана командою розробників.

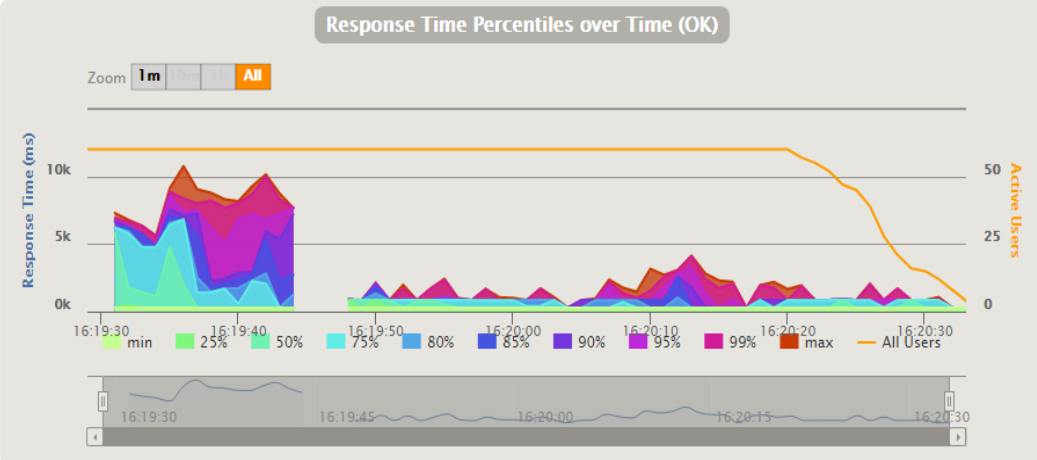


Рисунок 4.17 – Звіт «Response Time Percentiles over Time» для первинної версії системи при імітуванні паралельної роботи 60 користувачів

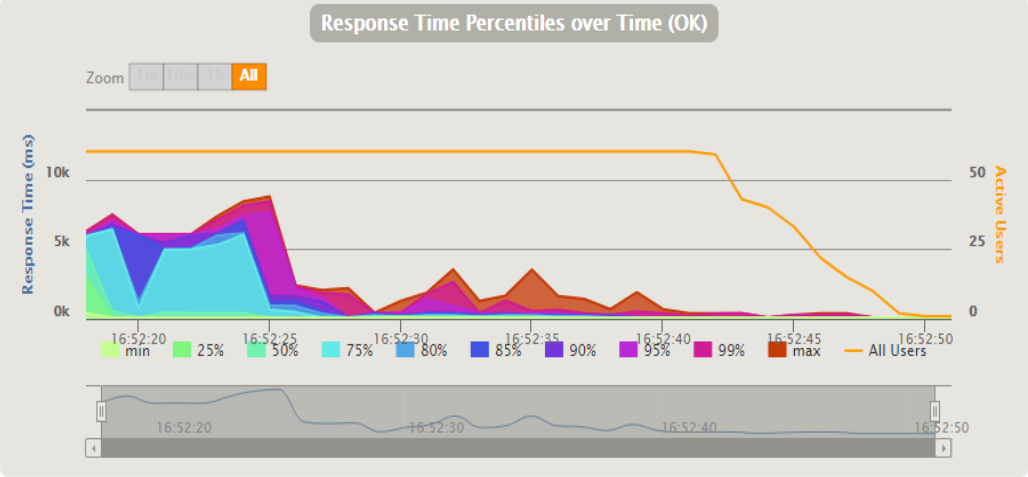


Рисунок 4.18 – Звіт «Response Time Percentiles over Time» для актуальної версії системи при імітуванні паралельної роботи 20 користувачів

Якщо подивитись на графік системи первинної версії (рис.4.17), мінімальним часом запиту є 281 мс, максимальним – 10,761 мс. Для системи актуальної версії (рис. 4.18), мінімальним часом запиту є 88 мс, максимальним – 8,790мс. Проаналізувавши отримані показники часу відгуку можна зробити висновок, що вимога, «максимальний час відгуку не повинен перевищувати 5000 мс», успішно виконана командою розробників.

На наступних графіках можна відстежити кількість запитів, які виконуються на сервері в кожен момент часу протягом тесту. На рисунку 4.19 представлений звіт для первинної системи при імітуванні паралельної роботи 60 користувачів, а на рисунку 4.20 – для актуальної системи.

На рисунку 4.21, що відповідає звіту «Number of responses per second» для первинної версії системи при імітуванні паралельної роботи 60 користувачів, та на рисунку 4.22 – для актуальної версії, можна відстежити кількість відповідей, які виконуються на сервері, та кількість користувачів в кожен момент часу.

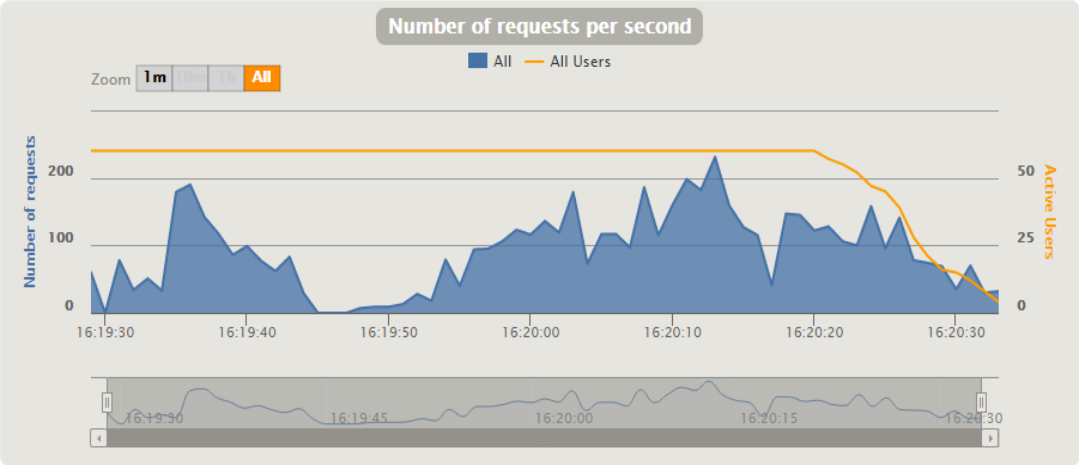


Рисунок 4.19 - Звіт «Number of requests per second» для первинної версії системи при імітуванні паралельної роботи 60 користувачів

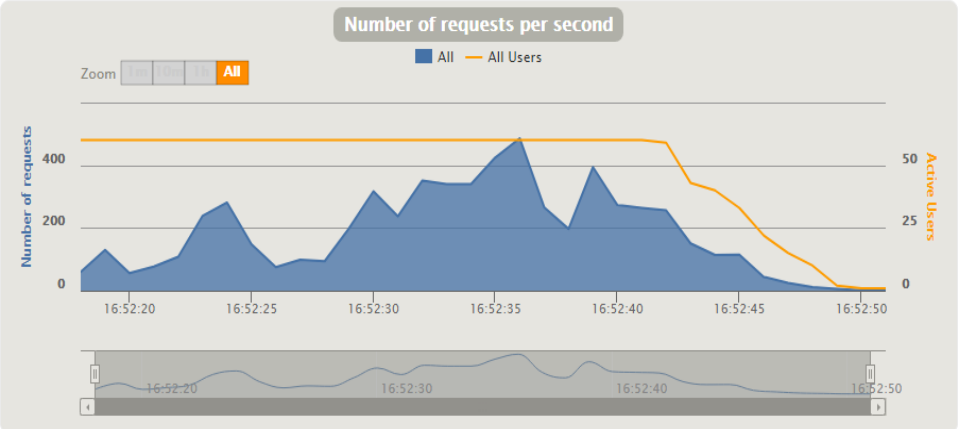


Рисунок 4.20 - Звіт «Number of requests per second» для первинної версії системи при імітуванні паралельної роботи 60 користувачів

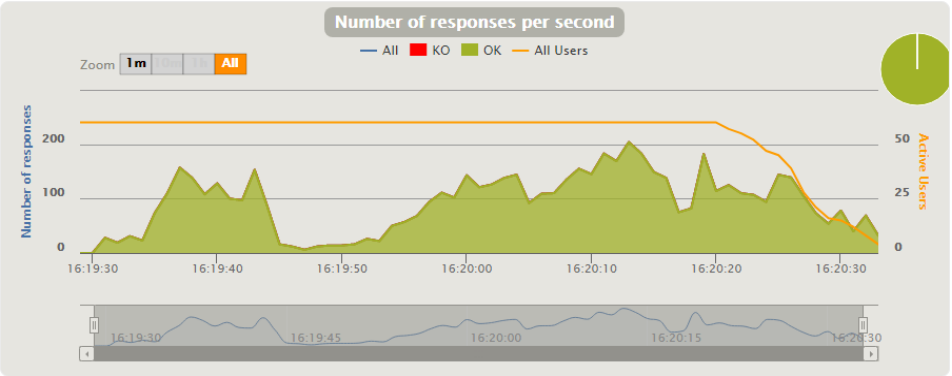


Рисунок 4.21 - Звіт «Number of responses per second» для первинної версії системи при імітуванні паралельної роботи 60 користувачів

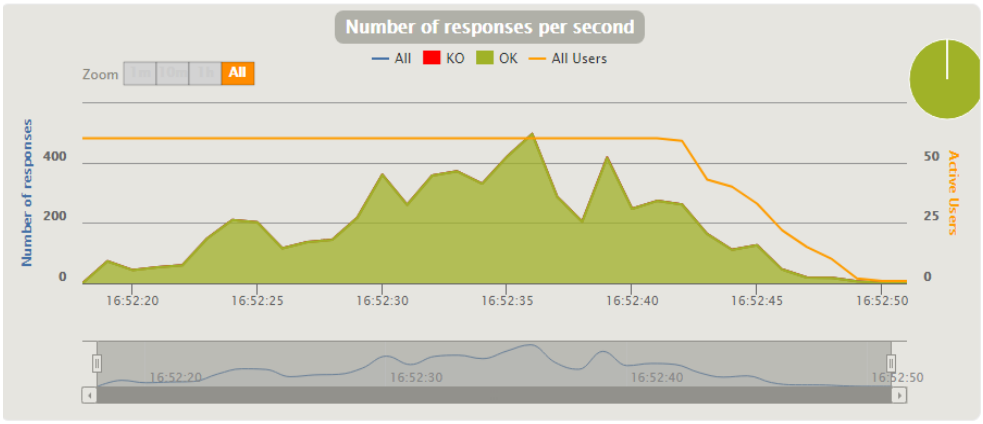


Рисунок 4.22 - Звіт «Number of responses per second» для первинної версії системи при імітуванні паралельної роботи 60 користувачів

4.3 Рекомендації щодо подальшого вдосконалення рівня продуктивності системи

Для подальшого вдосконалення рівня продуктивності системи Sierra було знайдено наступні методи:

– оптимізувати запити до бази даних;

– перехід на Java 8.

Оптимізація бази даних є однією із методів, що дозволить покращити показники продуктивності системи [6]. Найбільш популярними техніками оптимізації бази даних є:

– створення індексів для швидкості доступу;

– заміна корельованих підзапитів запитами з використанням Joins;

– уникання використання loops;

Індекс - це, в основному, структура даних, яка допомагає прискорити процес пошуку даних в цілому. Унікальний індекс - це своєрідна індексація, яка створює окремі стовпці даних, не перекриваючи один одного. Правильне індексування забезпечує швидший доступ до бази даних. Надмірна індексація або відсутність індексації взагалі є помилковими. Без будь-якої індексації взагалі, обробка буде дуже повільною, тоді як індексація все зробить вставку та тригери оновлення неефективними.

Корельований підзапит в основному залежить від батьківського або зовнішнього запиту. Цей вид пошуку виконується по рядку. Це означає, що це зменшує загальну швидкість процесу. Ця проблема, як правило, полягає в команді WHERE з зовнішнього запиту, який застосовує, який виконується підзапит для кожного рядка, який повертається за допомогою батьківського запиту, а отже, сповільнюється весь процес і зменшується ефективність бази даних. Таким чином, кращим способом настроювання бази даних в цьому випадку є команда INNER JOIN, а не корельований підзапит. Але в деяких випадках використання корельованого підзапиту має важливе значення.

Щоб уникнути уповільнення всієї послідовності, дуже важливо уникати використання loops. Цього можна досягти, використовуючи унікальні команди UPDATE або INSERT з окремими рядками, і гарантуючи, що команда WHERE не оновлює збережені дані, якщо він знаходить відповідні попередньо наявні дані.

Перехід системи із Java 6 на Java 8 надасть можливість використовувати нову колекцію, Stream, для швидкості та легкості керування даними. Stream має можливість застосовувати функції filter, map, reduce для його обробки [7]. Для Stream є два режими: послідовний і паралельний. Це дозволяє використовувати можливості багатоядерних процесорів, що є великим досягненням. Замість використання традиційного for-циклу, використання Stream в паралельному режимі теоретично прискорюється, зі збільшенням числа ядер, задіяних в обробці.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Конституція України, а саме ч.3, ст.43 [11] - гарантує нам безпечні і здорові умови праці. Детальніше щодо вимог охорони праці містить Кодекс законів про працю.Згідно вимог ст. 153 Кодексу законів про працю України [12] та ст. 6 Закону України «Про охорону праці» [13] на всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці. Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. Правила поширюються на всіх суб’єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов’язану з персональними комп’ютерами, у тому числі на тих, які мають робочі місця, обладнані персональними комп’ютерами і периферійними пристроями. Зазначені нормативно-правові акти встановлюють санітарно-гігієнічні вимоги до приміщення, в якому розташоване робоче місце, власне до робочого місця, освітлення, рівнів вібрації і шуму, мікроклімату в приміщенні тощо. В законі України «Про охорону праці» [14] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Під час роботи с обчислюваною технікою можуть змінюватися хімічні та фізичні фактори навколишнього середовища, в результаті чого виникає статична електрика, змінюється температура та вологість приміщення, змінюються рівень вмісту кисню та озону в повітрі.

Неправильна організація робочого приміщення негативно сприяє на працівника, сприяє підвищенню напрузі м’язів шиї, та шкідливо впливає на спинний хребет сприяюче розвитку остеохондроза, та інших захворювань.

Зазвичай робота з проектами проводитися в у кабінетах або інших приміщеннях, в яких використовують різне електрообладнання, таке як персональні ком’ютери та інші периферійні пристрої.

Основними робочими характеристиками персонального комп'ютера є наступні:

– робоча напруга U = +220В ± 5%;

– робочий струм I = 2А;

– споживана потужність P = 350 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 [15].

Зазвичай при роботі с ПК виникають наступні небезпечні та шкідливі чинники:

– забруднення повітря шкідливими речовинами (гранично-допустима концентрація (ГДК) шкідливих речовин у повітрі робочої зони повинна відповідати нормам, зазначеним [16]);

– електромагнітні випромінювання (рівні випромінювання і полів повинні відповідати [17]);

– освітлення (умови освітленості виробничих приміщень повинні відповідати нормам, зазначеним у [18]);

– шум (припустимі рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, повинен відповідати санітарним нормам допустимих рівнів шуму на робочих місцях [19])

– електростатичне поле (напруженість електричних полів промислової частоти на робочих місцях повинна відповідати нормам, зазначеним у [20]);

5.1 Освітлення

Світло є необхідною природною умовою для існування та праці людини. Хороше освітлення приміщення позитивно сприяє на працівника, та поліпшує його працездатність.

При поганому освітленні людина швидко втомлюється, загальній рівень продуктивності падає, виникає потенційна небезпека помилок та нещасних випадків.

Робота с ЕОМ має здійснюватися тільки при таких видах освітлення:

– загальному штучному освітленні, коли монітори встановленні по периметру приміщення у два ряди, звернені в протилежні боки;

– суміщене освітлення тільки при одному і трьох рядному розташуванні робочих місць, коли екран екрані поверхня робочого столу знаходяться перпендикулярно світла несучій стіні;

У приміщенні де розташовані ЕОМ передбачається бічне освітлення зліва, джерелом природного освітлення є сонячне світло. Не допускається спрямування природного світлового потоку ззаду, спереду або праворуч від працівника. Штучне освітлення в приміщеннях де розташовані ЕОМ та ПЕОМ, має здійснюватися рівномірним освітленням.

Світильники загального освітлення мают бути над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400LM кожна.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/8 - 1/10:

$$\sqrt{a^2 + b^2} \cdot S_b = (1/8 \div 1/10) \cdot S_n \qquad (5.1)$$

$$S_n = a \cdot b = 5 \cdot 5 = 25 \text{ м}^2$$

$$S_{вік} = 1/8 \cdot 25 = 3{,}125 \text{ м}^2$$

де Sn – площа віконних прорізів, м2; Sвік– площа підлоги, м2.

Приймаємо 2 вікна площею $S = 1{,}6 \text{ м}^2$ кожне

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників п виробляється по формулі (5.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \quad (5.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м2; S = 25 м2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2; F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (5.1), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1,1 \cdot 1,5}{5400 \cdot 0,575 \cdot 2} \approx 2.$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

5.2 Гігієнічні вимоги до параметрів виробничого середовища

Під час проведення будь-яких робіт, де обробка отриманих даних здійснюється за допомогою комп’ютерів, потрібно дотримуватися гігієнічних норм, правил і вимог техніки безпеки при роботі з персональним комп’ютером. Користувачі персональних комп’ютерів мають бути забезпечені відповідними робочими місцями, які відповідатимуть гігієнічним нормам. У приміщенні на робочому місці має забезпечуватися оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до ДСН 3.3.6.042-99 [21], рівні позитивних і негативних іонів у повітрі мають відповідати цьому ДСН [21]. Для забезпечення

комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби, чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціонування повітря, опалювальна система).

5.3 Вентилювання

У приміщенні де знаходиться ЕОМ повітрообмін має реалізовуватись з допомогою спеціально організованої вентиляції (вентиляційних шахт) і автономних кондиціонерів. Це має забезпечувати притоки потрібного свіжого повітря що визначається в СНіП (30 м3 на годину на одного працюючого). Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

5.4 Шум та вібрація, електромагнітне випромінювання

Шум часто є причиною зниження рівня працездатності, підвищення рівня загальної та професійної захворюваності, частоти виробничих травм..

Рівень шуму, що супроводжує роботу користувачів персональних комп’ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [22]. Шум такої інтенсивності на тлі високого ступеня напруженості праці негативно впливає на функціональний стан користувачів.

Віброізоляція можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам ДСН 3.3.6.037-99 [22]. Допустимий рівень вібрацій на робочому місці: для 1 ступеня шкідливості до 3 дБ; для 2-3 - 1-6 дБ; для 3 - більше 6 дБ.

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

1) застосування нових плазмових моніторів, LG W2271TC,
2) віддалення робочого місця не менше, ніж на 0,4-0,5 м, оскільки
3) напруженість електричного поля зменшується при віддаленні від джерела поля,

4) встановлення раціональних режимів роботи персоналу (обмеження часу перебування),

5) раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

5.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Загальний опір захисного заземлення визначається за формулою (5.3) :

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п}, \quad (5.3)$$

де Rз - опір заземлення, якими когут бать труби, опори, кути і т.п., Ом; Rш - опір опори, яке з'єднує заземлювачі, Ом; п - кількість заземлювачів; ηз - коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; ηз = 0,7 ηш - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; Опір заземлення визначається за формулою (5.4):

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(\ln \frac{2 \cdot l}{d} + \frac{1}{2} \ln \frac{4 \cdot t + l}{4 \cdot t - l} \right) \quad (5.4)$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м; для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м; l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м; d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м; t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 2 м

$$R_з = \frac{400}{2 \cdot 3{,}14 \cdot 3} \left(\ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} \ln \frac{4 \cdot 2 + 3}{4 \cdot 2 - 3} \right) = 110 \text{, Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою(5.5):

$$R_{ш} = \frac{\rho}{2\pi \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot t^1} \quad (5.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: Пбуд. = 42·2 + 38·2 = 160 м; L = 160 м;

b - ширина смуги, м; b = 0,03 м; t1 - глибина заземлення від рівня землі, м; t1 = 0,5 м.

$$R_{п} = \frac{400}{2 \cdot 3{,}14 \cdot 160} \cdot \ln \frac{2 \cdot 160^2}{0{,}03 \cdot 0{,}5} = 5{,}99 \text{ , Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою (5.6):

$$n = \frac{2 \cdot R_{з}}{4 \cdot \eta_{з}} \quad (5.6)$$

де 4 - допустимий загальний опір, Ом; 2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{ззп} = \frac{110 \cdot 5{,}99}{5{,}99 \cdot 79 \cdot 0{,}7 + 110 \cdot 0{,}5} = 1{,}7 \text{ Ом}$$

Данне захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: Rззп< 4 Ом.

При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

5.6 Екологія

Діяльність за темою дипломної роботи « » в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [23], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення»[24], Законом України «Про відходи»[25].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на віддалення (знешкодження), утилізацію.

В процесі діяльності за системою автоматизації природоохоронної діяльності промислових підприємств виникають процеси поводження з відходами. Види відходів, що утворюються в процесі:

- макулатура - IV клас небезпеки;
- матеріали пакувальні, що вміщують п/ет, п/пр - IV клас небезпеки;
- побутові відходи - IV клас небезпеки;
- змінні носії інформації;

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення Зазначені для зберігання відходів місця чи об'єкти повинні використовуватися лише для заявлених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи»[25] повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів.

Висновки до розділу 5

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також було розглянуто вплив на навколишнє природне середовище.

ВИСНОВКИ

У результаті виконання атестаційної роботи було проведено навантажувальне тестування для бібліотечної системи Sierra на первинній та актуальній версії, а також проведений аналіз показників продуктивності. Для цього був використаний фреймворк для НТ: Gatling.

Було проведено навантажувальне тестування, направлене на аналіз показників продуктивності проблемних функцій системи, які в свою чергу є найчастіше використовуваними користувачами та являються ключовими функціями системи. Були перевірені наступні характеристики продуктивності програмних засобів:

- максимальний та мінімальний час відповіді;
- розподіл відповідей відносно інтервалів часу;
- відсоток успішних та відповідей зі збоями протягом виконання тесту;
- кількість відправлених запитів у секунду.

Тестування було проведено під навантаженням 20 та 60 користувачами. Під час аналізу графіків, отриманих на системі актуальної версії, було виявлено, що після оптимізації системи командою були вирішені наступні вимоги замовника:

- відсоток запитів, час відгуку яких більше 1200 мс не повинен перевищувати 5%;
- максимальний час відгуку не повинен перевищувати 5000 мс;
- збільшити середню кількість оброблюваних запитів у секунду сервером з 100 до 200 при паралельній роботі 20 користувачів.

Під час аналізу отриманих графіків, було виявлено, що система первинної версії має середній рівень продуктивності, на відміну від системи актуальної версії, що має рівень продуктивності вище середнього.

У ході проведення аналізу продуктивності системи різних версій можна зробити висновок, що проведення навантажувального тестування повинно стати обов’язковим для виявлення проблемних місць під час роботи програмного забезпечення.

Був проведений аналіз методів оптимізації показників продуктивності системи та дані рекомендації щодо подальшого вдосконалення рівня продуктивності системи.

Таким чином, тестування продуктивності дозволяє виявити і усунути безліч потенційних проблем, які можуть виникнути у веб-системі на стадії експлуатації.

Отже, проведення навантажувального тестування допоможе визначити чи відповідає ПЗ вимогам швидкості, масштабованості та стабільності при очікуваних робочих навантаженнях та при перевищенні цього навантаження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Molino A.A. The Art of Application Performance Testing: Help for Programmers and Quality Assurance [Text] /A.A. Molino. – Proc.: O'Reilly Media, 2009. –158p.

2) Канер, С. Тестування програмного забезпечення. Фундаментальні концепції менеджменту бізнес–додатків [Текст] / С. Канер, Д. Фолк, Енг. – К.: Видавництво «Диасофт», 2001. – 544 с.

3) Техніки оптимізації продуктивності веб-систем [Электронний ресурс] – Режим доступу: www/URL:https://apiumhub.com/tech-blog-barcelona/web-performance-optimization-techniques/–Загл.с экрана

4) Метрики продуктивності веб-систем [Электроний ресурс] – Режим доступу: https://stackify.com/application-performance-metrics/ - Загл. з екрану.

5) Намиот Д.Е. Инструменты нагрузочного тестирования [Текст] / Д.Е. Намиот, Мясников С.О. – М.: Синергия, 2018. – 102 с.

6) Шляхи оптимізації продуктивності бази даних веб-систем [Электроний ресурс] – Режим доступу: www/URL: https://jaxenter.com/6-ways-optimize-sql-database-136448.html/ - Загл. з екрану.

7) Java 8 performance tutorial [Электроний ресурс] – Режим доступу: www/URL: https://jaxenter.com/java-performance-tutorial-how-fast-are-the-java-8-streams-118830.html - Загл. з екрану.

8) Performance Goal-Setting tutorial [Электроний ресурс] – Режим доступу: www/URL:https://www.dartmouth.edu/~hrs/profldev/performance_management/performance_objective.html - Загл. з екрану.

9) Volume Testing Tutorial [Электроний ресурс] – Режим доступу: www/URL: https://www.guru99.com/volume-testing.html - Загл. з екрану.

10) Анализ ключевых показателей производительности [Электронный ресурс] – Режим доступа:www/URL: https://msdn.microsoft.com/ru-ru/mt613109.aspx

11) Конституція України, стаття 43 https://kodeksy.com.ua/konstitutsiya_ukraini/statja-43.ht m - 08.12.200 4

12) ст. 153 Кодексу законів про працю https://kodeksy.com.ua/kodeks_zakoniv_pro_pratsyu_ukraini/statja-153.htm -10.12.1971

13) ст. 6 Закону України «Про охорону праці» https://kodeksy.com.ua/pro_ohoronu_pratsi283_new/statja-6.htm - 14.10.1992

14) Закон України «Про охорону праці» https://dnaop.com/html/3428/doc-zakon-ukrajini-pro-ohoronu-praci - 02.09.2008

15) Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин https://zakon.rada.gov.ua/rada/show/v0007282-98 - 10.12.98

16) ГОСТ 12.1.007-76 про концентрацію шкідливих речовин у повітрі http://legalexpert.in.ua/standarty-i-normativi/ministerstva-i-vedomstva/gost/8840-gost-1210 07-76.html - 03.02.99

17) Нормування та захист від електромагнітних випромінювань ГОСТ 12.1.006-84 - https://cpo.stu.cn.ua/Oksana/posibnik/910.html - 01.01.1986

18) ДБН В.2.5-28-2006 Природне і штучне освітлення http://www.gorsvet.kiev.ua/wp-content/uploads/2016/08/ДБН-В.2.5-28-2006.pdf 10.01.2006

19) ДСТУ 2867-94 про рівні звуку та еквівалентні рівні звуку на робочих місцях, повинен відповідати санітарним нормам допустимих рівнів шуму на робочих місцях https://dnaop.com/html/43864/doc-ДСТУ_2867-94 - 01.01.1996

20) Державний стандарт України ГОСТ 13109-97 http://docs.cntd.ru/document/120000603 4 - 01.01.1999

21) ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» https://dnaop.com/html/31678/doc-ДСН_3.3.6.042-99 - 01.12.1999

22) Санітарні норми шуму у робочих приміщеннях https://zakon.rada.gov.ua/rada/show/va037282-99 - 01.12.1999

23) Закон України про охорону навколишнього середовища https://zakon.rada.gov.ua/laws/show/1264-12 - 06.04.2000

24) Закон України про забезпечення санітарного та епідемічного благополуччя населення https://zakon.rada.gov.ua/laws/show/4004-12 - 24.02.1994

25) Закон України про відходи https://zakon.rada.gov.ua/laws/show/187/98 - 13.03.1998

ДОДАТОК А.
Електронні плакати

Інформаційна технологія аналізу продуктивності WEB-систем

Роботу виконав:
ст.гр. КН-18дм
Макаренко В.Р.
Керівник: доц. Барбарук В.М.

Актуальність навантажувального тестування

Тестування продуктивності може допомогти визначити характер або місце розташування проблеми, пов'язаної з програмним забезпеченням, підкреслюючи, де програма може вийти з ладу або відставання.

За допомогою навантажувального тестування, а також внесення покращень та змін, заснованих на результатах цього тестування, ви зможете надати користувачам професійну веб-систему, що буде доступна і виконуватиме свої задачі у будь-який момент часу, коли вона знадобиться клієнтам.

Постановка задачі дослідження

Метою даної атестаційної роботи є порівняння показників продуктивності на системі первинної та актуальної версії, після впровадження методів оптимізації показників продуктивності.

Gatling як фреймворк для навантажувального тестування

Фреймворк Gatling підтримує мову розробки Scala і створений для навантажувального та стрес-тестування. Gatling реалізував повністю нову архітектуру для інструменту тестування продуктивності, щоб бути більш ресурсомістким. Це дає змогу імітувати велику кількість запитів за секунду за допомогою однієї машини.

blog.webnet.fr

Підходи до оптимізації показників продуктивності Web-систем

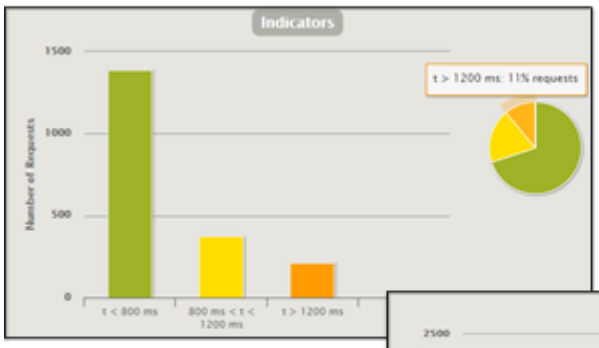
- кешування даних;
- асинхронна обробка даних;
- динамічне завантаження контенту;
- посторінкова розбивка.

Вимоги до показників продуктивності тестової системи

Перед командою розробки бібліотечного продукту Sierra стояли наступні вимоги до оптимізації показників продуктивності системи при паралельній роботі 20 користувачів:

- відсоток запитів, час відгуку яких більше 1200 мс не повинен перевищувати 5%;
- максимальний час відгуку не повинен перевищувати 5000 мс;
- збільшити середню кількість оброблюваних запитів у секунду сервером з 100 до 200.

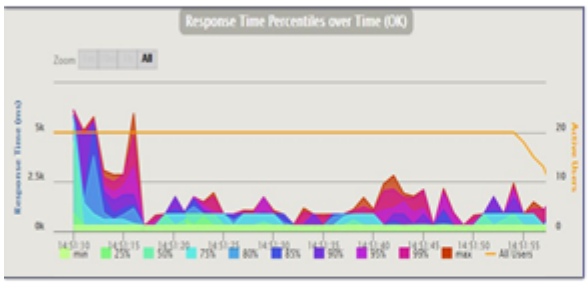
Звіт “Indicators”



первинна версія

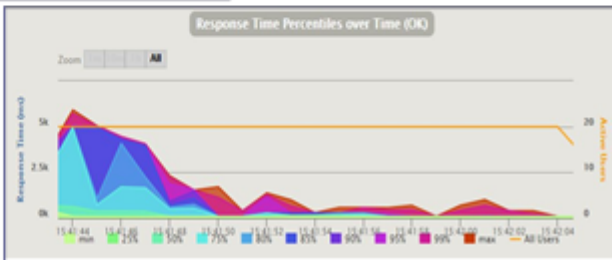
актуальна версія

Звіт “Response time Percentiles over Time”

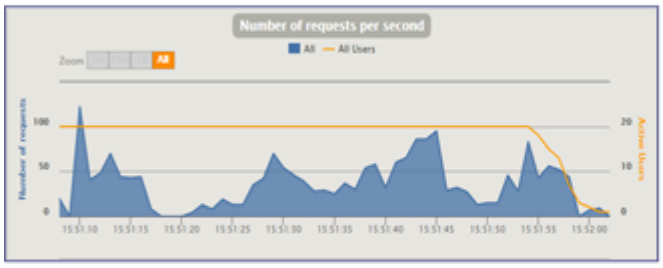


первинна версія

актуальна версія

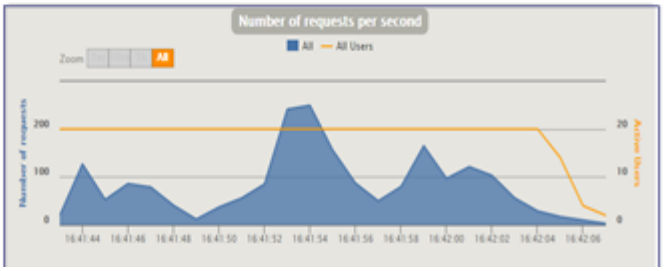


Звіт “Number of requests per second”

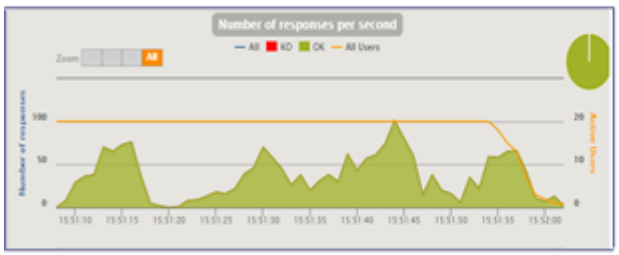


актуальна версія

первинна версія

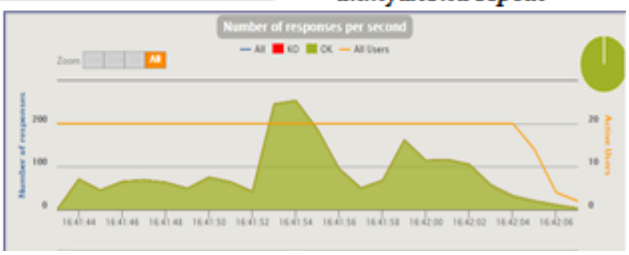


Звіт “Number of responses per second”



актуальна версія

первинна версія



Висновки

Були перевірені наступні характеристики продуктивності :

- максимальний та мінімальний час відповіді;
- розподіл відповідей відносно інтервалів часу;
- відсоток успішних та відповідей зі збоями протягом виконання тесту;
- кількість оброблених запитів у секунду;
- результати роботи обговорювалися на конференції «Майбутній науковець 2019».

Дякую за увагу!