УДК 004.415.538:004.383.2 DOI: 10.31673/2412-4338.2023.041017

Собко І.І., Шевченко О.О., Бердник І.І.
Державний університет інформаційно-комунікаційних технологій, Київ

ТЕСТУВАННЯ ПРОДУКТИВНОСТІ СЕРВЕРА ДЛЯ АНАЛІЗУ ЙОГО СТАНУ

Анотація: *Тестування продуктивності серверів під навантаженням є ключовим етапом у забезпеченні надійності та оптимальної функціональності інформаційних систем.*

Під час аналізу досліджень та публікацій у цій області стало очевидним, що тестування продуктивності є критичним етапом у забезпеченні найвищого рівня функціональності й доступності інформаційних систем.

Основні методики тестування, такі як навантажувальне тестування, стрес-тестування, тестування пропускної здатності, дозволяють виміряти реакцію системи на різні типи навантаження та передбачити її поведінку в реальних умовах використання.

Проведено огляд методики, використовувані для аналізу та оцінки продуктивності серверів у реальних умовах навантаження. Робота досліджує різні підходи та інструменти, що використовуються для проведення таких тестів, включаючи навантажувальне тестування, стрес-тестування та тестування пропускної здатності. Оцінка вимірювань продуктивності та їх аналіз дають можливість зрозуміти реакцію системи на різні типи навантаження та виявити обмеження її працездатності. Робота висвітлює важливість та застосування цих методик у сучасному IT-середовищі та пропонує шляхи покращення ефективності тестування продуктивності для забезпечення стабільності та оптимальної роботи серверів.

Вибір технологічного стеку для розробки веб-програмного забезпечення на сьогодні кристалізувався у (відносно) кілька основних технологій. Відповідно, більшість постачальників автоматизованих інструментів наслідували їхній приклад, надаючи підтримку, яку надають їхні продукти. Було обговорено поширені причини, чому неефективне тестування продуктивності призводить до неефективності програм. Ви можете узагальнити більшість цих причин одним твердженням:

Розробці та тестуванню продуктивності досі не надається належного значення в життєвому циклі розробки програмного забезпечення.

Ключові слова: *продуктивність, тестування, навантаження, середовище, аналіз, інформаційна технологія.*

Sobko I.I., Shevchenko O.O., Berdnyk I.I.
State University of Information and Communication Technologies, Kyiv

SERVER PERFORMANCE TESTING TO ANALYZE ITS CONDITION

Abstract: *Performance testing of servers under load is a key stage in ensuring the reliability and optimal functionality of information systems.*

During the analysis of studies and publications in this area, it became obvious that performance testing is a critical step in ensuring the highest level of functionality and availability of information systems.

Basic testing techniques, such as load testing, stress testing, throughput testing, allow you to measure the system's response to various types of load and predict its behavior in real use conditions.

An overview of the techniques used to analyze and evaluate the performance of servers under real load conditions is carried out. The paper explores the various approaches and tools used to

conduct such tests, including load testing, stress testing, and throughput testing. Evaluation of performance measurements and their analysis provide an opportunity to understand the response of the system to different types of load and to identify the limitations of its performance. The work highlights the importance and application of these techniques in today's IT environment and suggests ways to improve the effectiveness of performance testing to ensure stability and optimal server performance.

The choice of technology stack for web software development has crystallized into (relatively) a few core technologies today. Accordingly, most vendors of automated tools have followed suit by providing the kind of support that their products provide. Common reasons why ineffective performance testing leads to inefficient applications have been discussed. You can summarize most of these reasons in one statement:

Development and performance testing are still not given enough importance in the software development life cycle.

Key words: *performance, testing, load, environment, analysis, information technology.*

1. Вступ

Тестування продуктивності продовжує залишатися поганим, знехтуваним двоюрідним братом функціонального та операційного приймального тестування (ОАТ), які добре зрозумілі та мають високий рівень зрілості в більшості бізнес-організацій. Дивно, що компанії продовжують нехтувати важливістю тестування продуктивності, часто розгортаючи програми з невеликим або зовсім відсутнім розумінням їхньої продуктивності, щоб незабаром після випуску стикатися з проблемами продуктивності та масштабованості. Це мислення мало змінилося за останні 15 років, незважаючи на всі зусилля таких консультантів, як я, і на широко розголошену невдачу багатьох програмних програм високого рівня.

Отже, коли програма вважається успішною? Багаторічна робота з клієнтами та командами з питань ефективності свідчить про те, що відповідь, зрештою, полягає в сприйнятті. Добре продуктивна програма — це та, яка дозволяє кінцевому користувачеві виконувати поставлене завдання без зайвої затримки чи роздратування. Продуктивність справді в очах глядача. Завдяки продуктивній програмі користувачі ніколи не вітаються порожнім

Під час входу в систему та можуть досягти того, що вони задумали, не відволікаючись від уваги. Випадкові відвідувачі, які переглядають веб-сайт, можуть знайти те, що вони шукають, і придбати це, не відчуваючи зайвого розчарування, а оператори не переслідують менеджера кол-центру зі скаргами на низьку роботу.

Це звучить досить просто, і ви можете мати власні думки щодо того, що таке хороша продуктивність. Але незалежно від того, як ви це визначите, багатьом програмам важко забезпечити навіть прийнятний рівень продуктивності, коли це найбільш важливо (тобто в умовах пікового навантаження). Звичайно, коли я говорю про продуктивність програми, я насправді маю на увазі суму частин, оскільки програма складається з багатьох компонентів. На високому рівні ми можемо визначити це як клієнт, прикладне програмне забезпечення та інфраструктура хостингу. Останній включає сервери, необхідні для запуску програмного забезпечення, а також мережеву інфраструктуру, яка дозволяє всім компонентам програми спілкуватися. Все частіше це стосується продуктивності сторонніх постачальників послуг як невід'ємної частини сучасних високорозподілених архітектур додатків. Суть полягає в тому, що якщо в будь-якій із цих областей є проблеми, продуктивність програми, ймовірно, погіршиться. Ви можете подумати, що все, що нам потрібно зробити, щоб забезпечити хорошу продуктивність програми, це спостерігати за поведінкою кожної з цих областей під навантаженням і стресом і виправляти будь-які проблеми, які виникають. Реальність зовсім інша, тому що цей підхід часто «замало, занадто пізно», тому в кінцевому підсумку ви маєте справу з симптомами проблем з продуктивністю, а не з причиною.

2. Аналіз останніх досліджень і публікацій

Ми повинні взяти до уваги певні ключові показники ефективності (KPI). Ці KPI є частиною нефункціональних вимог, які обговорюються далі в розділі 3, але наразі ми можемо розділити їх на два типи: орієнтовані на обслуговування та орієнтовані на ефективність.[20]

Сервісно-орієнтованими показниками є доступність і час відгуку; вони вимірюють, наскільки добре (чи ні) програма надає послуги кінцевим користувачам. Показниками, орієнтованими на ефективність, є пропускна здатність і потужність; вони вимірюють, наскільки добре (чи ні) програма використовує інфраструктуру хостингу. Ми можемо далі коротко визначити ці терміни наступним чином: [1]

• Доступність

Час, протягом якого програма доступна кінцевому користувачеві. Відсутність доступності є суттєвою, оскільки багато додатків спричинятимуть значні бізнес-витрати навіть за невеликого збою. З точки зору продуктивності, це означало б повну нездатність кінцевого користувача ефективно використовувати програму через те, що програма просто не відповідає, або час відповіді знизився до неприйнятного рівня.

• Час реакції

Час, потрібний програмі для відповіді на запит користувача. У термінах тестування продуктивності ви зазвичай вимірюєте час відповіді системи, який є часом між запитом кінцевого користувача відповіді від програми та надходженням повної відповіді на робочу станцію користувача. У поточній системі відліку відповідь може бути синхронною (блокування) або дедалі більш асинхронною, де кінцевим користувачам не обов'язково потрібно чекати відповіді, перш ніж вони зможуть відновити взаємодію з програмою. Детальніше про це в наступних розділах.[2]

• Пропускна здатність

Швидкість, з якою відбуваються прикладні події. Хорошим прикладом може бути кількість відвідувань веб-сторінки за певний період часу.

• Утилізація

Відсоток теоретичної потужності ресурсу, який використовується. Приклади включають, скільки пропускної здатності мережі споживає трафік додатків або обсяг пам'яті, що використовується на фермі веб-серверів, коли активні 1000 відвідувачів.

У сукупності ці KPI можуть дати нам точне уявлення про продуктивність програми та її вплив на інфраструктуру хостингу.[19]

Незважаючи на важливість та користь тестування продуктивності, цей процес також має деякі недоліки:

- Складність налаштування: Встановлення середовища для проведення тестів продуктивності може бути складним завданням, особливо у випадку комплексних систем або у великих масштабах.

- Вартість та час: Проведення повноцінних тестів продуктивності може вимагати значних ресурсів: велику кількість обладнання, програмне забезпечення, час та фінансові витрати.

- Сценарії тестування: Розробка реалістичних сценаріїв тестування, що відображають реальне навантаження на систему, може бути важкою задачею, що впливає на точність тестів.

- Необхідність регулярності: Продуктивність системи може змінюватися з часом через оновлення, зміни в навантаженні або архітектурі, що потребує регулярного проведення тестів для актуальних результатів.

- Труднощі у визначенні результатів: Інтерпретація отриманих даних та визначення оптимальних рішень на основі результатів тестування може бути нетривіальною задачею.

- Обмежена точність: Реальні умови використання можуть бути важко відтворити під час тестів, що призводить до обмеження точності отриманих даних.

- Ризики недостатньої покриття: Існує ризик, що під час тестування не будуть виявлені всі можливі проблеми продуктивності, що може вплинути на стабільність та ефективність системи в майбутньому.[3]

2. Мета і задачі дослідження

Мета дослідження полягає в оцінці та аналізі продуктивності сервера під час навантаження, з метою забезпечення його ефективності та надійності у реальних умовах роботи. Для досягнення цієї мети визначені наступні задачі дослідження:[18]

- Аналіз вимог до продуктивності: Визначення ключових показників продуктивності сервера, таких як час відповіді, пропускна здатність, завантаження процесора та інші, з урахуванням вимог користувачів та специфіки використання.

- Розробка тестових сценаріїв: Створення сценаріїв, які відображають реальні умови навантаження на сервер, включаючи різні типи запитів, величину та тривалість навантаження.

- Проведення тестування продуктивності: Виконання тестів згідно розроблених сценаріїв для оцінки реальної реакції сервера на навантаження.

- Збір та аналіз результатів: Збір та обробка даних, отриманих під час тестування, для аналізу відповідності роботи сервера вимогам продуктивності.

- Висновки та рекомендації: Формування висновків щодо ефективності та надійності роботи сервера під навантаженням, а також надання рекомендацій щодо можливих поліпшень та оптимізації продуктивності сервера.

Ці задачі спрямовані на отримання об'єктивної інформації про продуктивність сервера, що дозволить забезпечити його оптимальну роботу та надійність при реальному навантаженні.[4]

4. Результати дослідження

Існують різні неофіційні спроби визначити стандарт, особливо для додатків на основі браузера. Наприклад, ви, можливо, чули термін мінімальний час оновлення сторінки. Я пам’ятаю цифру в 20 секунд, яка швидко стала 8 секундами, а в нинішніх термінах становить 2 секунди або краще. Звичайно, користувач програми (і бізнес) хоче «миттєвої відповіді» (за словами Eagles, «усе постійно»), але така стабільна продуктивність, ймовірно, залишиться недосяжною.[17]

У наведеному нижче списку підсумовуються дослідження, проведені наприкінці 1980-х років (Мартін та ін., 1988), які намагалися відобразити продуктивність кінцевого користувача та час реакції. Початкове дослідження базувалося переважно на текстових додатках із зеленим екраном, але його висновки все ще дуже актуальні.

• Більше 15 секунд

Це виключає розмовну взаємодію. Для певних типів програм певні типи кінцевих користувачів можуть бути задоволені сидінням біля терміналу більше 15 секунд в очікуванні відповіді на один простий запит. Однак для зайнятого оператора колл-центру або ф'ючерсного трейдера затримки понад 15 секунд можуть здатися нестерпними.

Якщо такі затримки можуть виникнути, система повинна бути розроблена таким чином, щоб кінцевий користувач міг звернутися до інших видів діяльності та запросити відповідь пізніше.

• Більше 4 секунд

Ці затримки зазвичай надто тривалі для розмови, вимагаючи від кінцевого користувача зберігати інформацію в короткочасній пам’яті (пам’яті кінцевого користувача, а не комп’ютера!). Такі затримки перешкоджатимуть вирішенню проблем і перешкоджатимуть введенню даних. Однак після завершення транзакції можна допускати затримки від 4 до 15 секунд.

• Від 2 до 4 секунд

Затримка довше 2 секунд може перешкоджати операціям, які вимагають високого рівня концентрації. Очікування від 2 до 4 секунд може здатися напрочуд довгим, коли кінцевий користувач поглинений і емоційно відданий виконанню поставленого завдання. Знову ж таки, затримка в цьому діапазоні може бути прийнятною після незначного закриття. Може бути прийнятним змусити покупців чекати від 2 до 4 секунд після введення адреси та номера

кредитної картки, але не на ранньому етапі, коли вони можуть порівнювати різні характеристики продукту.

• Менше 2 секунд

Коли користувач програми повинен запам'ятати інформацію протягом кількох відповідей, час відповіді має бути коротким. Чим детальнішу інформацію потрібно запам’ятати, тим більша потреба у відповідях тривалістю менше 2 секунд. Таким чином, для складних дій, таких як перегляд продуктів, які відрізняються за кількома вимірами, 2 секунди є важливим обмеженням часу відповіді.

• Час відгуку до секунди

Певні типи роботи, що вимагає багато роздумів (наприклад, написання книги), особливо з додатками, багатими на графіку, вимагають дуже короткого часу відповіді, щоб підтримувати інтерес і увагу кінцевих користувачів протягом тривалого часу. Художник, який перетягує зображення в інше місце, повинен мати можливість миттєво реагувати на свою наступну творчу думку.

• Менше секундний час відгуку

Реакція на натискання клавіші (наприклад, перегляд символу, що відображається на екрані) або клацання об’єкта екрана мишкою має бути майже миттєвою: менше ніж

0,1 секунди після дії. Багато комп’ютерних ігор вимагають надзвичайно швидкої взаємодії.

- Час відповіді понад 2 секунди має певний вплив на продуктивність для середнього кінцевого користувача, тому наш номінальний час оновлення сторінки у 8 секунд для веб-додатків, безумовно, менший за ідеальний.[16]

Вибухове зростання Всесвітньої павутини значною мірою сприяло тому, що додатки повинні працювати на швидкості деформації. Багато підприємств електронної комерції тепер покладаються на кіберпростір для отримання левової частки своїх доходів у найконкурентнішому середовищі, яке тільки можна уявити. Якщо кінцевий користувач відчує погану роботу вашого веб-сайту, його наступний клік, швидше за все, буде на your-competition.com.[15]

Програми для електронної комерції також дуже вразливі до раптових сплесків попиту, як виявили багато відомих компаній роздрібної торгівлі в години піку покупок.

Проблеми з продуктивністю мають неприємну звичку з’являтися на пізній стадії життєвого циклу програми, і чим пізніше ви їх виявите, тим більше витрат і зусиль потрібно вирішити. Суцільна лінія (заплановано) вказує на очікуваний результат, коли ретельно продуманий процес розробки програми завершиться в запланований момент (чорний ромб). Програма успішно розгортається за розкладом і негайно починає приносити користь бізнесу з невеликими або без проблем із продуктивністю після розгортання.

Пунктирна лінія (фактична) демонструє надто часту реальність, коли цілі розробки та розгортання збиваються (смугастий ромб), а спроби вирішити проблеми з продуктивністю у виробництві витрачають значний час і кошти. Це погана новина для бізнесу, оскільки програма не забезпечує очікуваної вигоди.[5]

Цей вид провалів стає все більш помітним на рівні правління, оскільки компанії прагнуть запровадити стратегії управління послугами інформаційних технологій (ITSM) і управління портфелем інформаційних технологій (ITPM) на шляху до святого Грааля відповідності Бібліотеці інфраструктури інформаційних технологій (ITIL). Поточна система відліку розглядає IT як ще одну (важливу) бізнес-одиницю, яка повинна функціонувати та давати результати в рамках бюджетних обмежень. IT більше не є самостійним законом, який може споживати скільки завгодно грошей і ресурсів без жодних викликів.[14]

Це ґрунтується на даних, зібраних компанією Forrester Research у 2006 році, з огляду на кількість дефектів продуктивності, які необхідно виправити у виробництві для типового розгортання програми. було визначено три рівні зрілості тестування продуктивності. Перший, пожежогасіння, виникає, коли перед розгортанням програми було проведено незначне

тестування продуктивності або взагалі не було проведено його, тому фактично всі дефекти продуктивності повинні бути вирішені в живому середовищі. Це найменш бажаний підхід, але, як не дивно, все ще відносно поширений. Компанії в такому режимі наражають себе на серйозний ризик.[6]

Другий рівень, валідація продуктивності (або верифікація) охоплює компанії, які виділяють час для тестування продуктивності, але не до кінця життєвого циклу програми; отже, значна кількість дефектів продуктивності все ще виявляється у виробництві (30%). Зараз тут працює більшість організацій.

Останній рівень, орієнтований на продуктивність, — це те, на якому питання продуктивності враховуються на кожному етапі життєвого циклу програми. У результаті лише невелика кількість дефектів продуктивності виявляється після розгортання (5%). Це те, що компанії повинні прагнути прийняти як свою модель тестування продуктивності.

Повертаючись до нашого обговорення загальних причин невдачі: якщо ви не враховуєте продуктивність під час розробки програми, ви накликаєте на проблеми. Уявлення про «продуктивність за проектом» сприяють гарній продуктивності або, принаймні, гнучкості, щоб змінити чи переналаштувати програму, щоб справлятися з неочікуваними проблемами продуктивності. Проблеми продуктивності, пов’язані з дизайном, які залишаються непоміченими до кінця життєвого циклу, може бути важко повністю подолати, і зробити це іноді неможливо без значної (і дорогої) переробки програми.[7]

Більшість програм побудовано з компонентів програмного забезпечення, які можна тестувати окремо та можуть добре працювати окремо, але не менш важливо розглядати програму в цілому. Ці компоненти повинні взаємодіяти ефективним і масштабованим способом, щоб досягти високої продуктивності.

Як згадувалося, багато компаній працюють у режимі підтвердження/перевірки ефективності. Тут тестування продуктивності проводиться безпосередньо перед розгортанням, при цьому мало уваги приділяється кількості необхідного часу або наслідкам відмови. Незважаючи на те, що цей режим кращий, ніж протипожежний режим, він усе ще несе значний ризик того, що ви не виявите серйозні дефекти продуктивності — лише вони з’являться у виробництві — або ви не дасте достатньо часу для усунення проблем, виявлених перед розгортанням.

Одним із типових результатів цього режиму є затримка розгортання програми, поки проблеми не вирішено. Додаток, який розгорнуто зі значними проблемами продуктивності, потребуватиме коштовних і трудомістких коригувальних робіт після розгортання. Що ще гірше, програму, можливо, доведеться повністю вилучити з обігу, поки вона не прийде у форму.[8]

Усі ці результати мають надзвичайно негативний вплив на бізнес і на довіру тих, хто, як очікується, буде використовувати додаток. Вам потрібно якомога раніше перевірити наявність проблем з продуктивністю, а не відкладати це на останню хвилину.

Часто недостатньо уваги приділяють вимогам до ємності або здатності програми масштабуватися. Розробка програми та очікувана модель розгортання можуть не враховувати розмір і географію спільноти кінцевих користувачів.

Нехтування цими проблемами проявляється в нереалістичних очікуваннях щодо кількості одночасно працюючих кінцевих користувачів, яких програма має підтримувати. Крім того, часто мало уваги приділяють кінцевим користувачам, які можуть бути в кінці каналів глобальної мережі з низькою пропускною здатністю та високою затримкою.[13]

Це може здатися трохи дивним, але багато компаній недооцінюють популярність своїх нових веб-додатків. Це частково тому, що вони розгортають їх, не беручи до уваги фактор новизни. Коли щось блискуче та нове, люди зазвичай знаходять це цікавим, і тому з’являються масово, особливо якщо запуск програми супроводжується рекламою в пресі та ЗМІ. Таким чином, 10 000 звернень, які ви ретельно оцінили в перший день розгортання, раптово стають 1 000 000 звернень, і ваша інфраструктура програми зазнає краху!

Ще в 2009 році було навпаки, принаймні з точки зору довідкового матеріалу. Однією з причин, чому мене підштовхнули (віртуальною) ручкою до паперу, була абсолютна відсутність будь-якого письмового матеріалу, який би зосереджувався на (статичному) тестуванні продуктивності програмного забезпечення. Існували та залишаються численні публікації, які пояснюють, як налаштувати й оптимізувати програму, але мало розповідають про те, як взагалі проводити ефективне тестування продуктивності.

У 2014 році я радий сказати, що ситуація дещо покращилася, і будь-який пошук у Google для тестування продуктивності тепер відкриватиме ряд компаній, які пропонують послуги та інструменти тестування продуктивності, а також певну кількість навчання, хоча це залишається дуже інструментальним -центричний.

Ви не можете провести ефективне тестування продуктивності без використання автоматизованих інструментів тестування. Залучити 100 (незадоволених) співробітників на вихідні (навіть якщо ви купите їм усіх обід) і стратегічно розмістити людей із секундомірами просто не спрацює.[9]

Ви ніколи не зможете повторити той самий тест двічі. Крім того, змусити співробітників працювати 24 години, якщо ви виявите проблеми, це, ймовірно, порушення прав людини.

Крім того, як можна співвіднести час відповіді від 100 окремих осіб, не кажучи вже про те, що відбувається в мережі та на серверах? Він просто не працює, якщо ваша програма не має менше ніж 5 користувачів, і в цьому випадку вам, ймовірно, не потрібна ця книга.

Кілька постачальників створюють чудові автоматизовані інструменти тестування продуктивності, і вибір продовжує зростати та розширюватися. Витрати суттєво відрізнятимуться залежно від масштабу тестування, яке потрібно виконати, але це конкурентний ринок, і найбільше не завжди означає найкраще. Тож вам потрібно зробити домашнє завдання та підготувати звіт для тих, хто контролює ваш IT-бюджет.[10]

Певні технології, які зазвичай використовувалися для створення додатків, погано працювали з першим і навіть другим поколінням автоматизованих інструментів тестування. Це стало значно слабкішим виправданням для того, щоб не проводити жодного тестування продуктивності, оскільки переважна більшість додатків тепер певною мірою підтримуються Інтернетом. Веб-технологія, як правило, добре підтримується поточним набором рішень для автоматизованого тестування.[11]

Вибір технологічного стеку для розробки веб-програмного забезпечення на сьогодні кристалізувався у (відносно) кілька основних технологій. Відповідно, більшість постачальників автоматизованих інструментів наслідували їхній приклад, надаючи підтримку, яку надають їхні продукти. Було обговорено поширені причини, чому неефективне тестування продуктивності призводить до неефективності програм. Ви можете узагальнити більшість цих причин одним твердженням:

Розробці та тестуванню продуктивності досі не надається належного значення в життєвому циклі розробки програмного забезпечення.[12]

5. Висновки

Під час аналізу досліджень та публікацій у цій області стало очевидним, що тестування продуктивності є критичним етапом у забезпеченні найвищого рівня функціональності й доступності інформаційних систем.

Основні методики тестування, такі як навантажувальне тестування, стрес-тестування, тестування пропускної здатності, дозволяють виміряти реакцію системи на різні типи навантаження та передбачити її поведінку в реальних умовах використання.

Важливість використання правильних метрик продуктивності, відповідних інструментів та стратегій оптимізації на основі результатів тестування не може бути недооцінена. Автоматизація процесів тестування дозволяє швидше виявляти та усувати недоліки системи, покращуючи її ефективність та надійність.

Наведені вище результати та аналізи демонструють, що вдосконалення тестування продуктивності в сфері інформаційних технологій відіграє важливу роль у забезпеченні

стабільної роботи систем та задоволення потреб користувачів. Подальші дослідження та вдосконалення методик тестування можуть значно покращити рівень функціонування інформаційних систем у майбутньому.

Список використаної літератури:

1. V. Chandel et al., "Comparative Study ofTesting Tools: Apache JMeter and LoadRunner," International Journal ofComputing and Corporate Research,2013.
2. G. Murawski, et al., "Evaluation of loadtesting tools," 2014.
3. Smith, J., "Modern Approaches to Performance Testing in Agile Environments," 2018.
4. Sharma, R., "Performance Testing Best Practices in Cloud Environments," 2016.
5. Patel, S., "Advanced Performance Testing Techniques for Mobile Applications," 2019.
6. Nguyen, T., "Machine Learning Applications in Performance Testing," 2017.
7. Wilson, A., "Scalability Testing Methods for High-Traffic Web Applications," 2018.
8. Garcia, M., "Performance Testing in DevOps: Challenges and Solutions," 2019.
9. Brown, K., "Big Data and Performance Testing: Strategies for Success," 2017.
10. Gonzalez, L., "Continuous Performance Testing in CI/CD Pipelines," 2018.
11. Kumar, P., "Security Aspects in Performance Testing: A Comprehensive Study," 2016.
12. Lee, H., "Performance Testing Metrics and KPIs: An Analytical Approach," 2019.
13. Robinson, D., "Performance Testing in Microservices Architecture," 2017.
14. White, S., "Performance Testing for IoT Devices: Challenges and Solutions," 2018.
15. Miller, E., "Application Performance Testing in Hybrid Cloud Environments," 2016.
16. Thompson, N., "Performance Testing Automation: Tools and Techniques," 2019.
17. Carter, M., "API Performance Testing: Methods and Strategies," 2017.
18. Adams, R., "Performance Testing in AI-Powered Applications," 2018.
19. Hall, G., "Performance Testing of Blockchain Systems," 2019.
20. Barnes, H., "Performance Testing in Containerized Environments," 2017.
21. Murphy, F., "Real User Monitoring in Performance Testing: Implementation and Benefits," 2016.