УДК 004.738:004.9
Коцюба А.Ю., Цяпич Я.П., Лавренчук С.В.
Луцький національний технічний університет

ПРО МЕТОДИКУ ОПТИМІЗАЦІЇ ВІДМОВСТІЙКОСТІ ВЕБ-СЕРВЕРІВ НА ОДНОЧАСНУ КІЛЬКІСТЬ ЗАПИТІВ

Коцюба А.Ю., Цяпич Я.П., Лавренчук С.В. Про методику оптимізації відмовостійкості веб-серверів на одночасну кількість запитів. Проведено дослідження ефективності та відмовостійкості програмної частини веб-сервера. Було проведено навантажувальне тестування веб-серверів синхронної та асинхронної моделі обробки запитів. Окрім тестування та порівняння веб-серверів, було здійснено пошук та налаштування більш оптимізованого конфігураційного рішення для роботи веб-сервера, що дає можливість витримувати більшу кількість запитів.
Ключові слова: відмовостійкість, веб-сервер, асинхронна модель опрацювання запитів, навантажувальне тестування, Node.js, Nginx, Apache, Siege.

Коцюба А.Ю., Цяпич Я.П., Лавренчук С.В. О методике оптимизации отказоустойчивости веб-серверов на одновременное количество запросов. Проведено исследование эффективности и отказоустойчивости программной части веб-сервера. Было проведено нагрузочное тестирование веб-серверов синхронной и асинхронной модели обработки запросов. Кроме тестирования и сравнения веб-серверов, было осуществлено поиск и настройка более оптимизированного конфигурационного решения для работы веб-сервера, что позволяет выдерживать большее количество запросов.
Ключевые слова: отказоустойчивость, веб-сервер, асинхронная модель обработки запросов, нагрузочное тестирование, Node.js, Nginx, Apache, Siege.

Kotsyuba A.Y., Tsyapych Y.P., Lavrenchuk S.V., On the method of optimizing the web-server failover for simultaneous amount of requests. The research of the web-server efficiency and resiliency is made in the article. The web-servers testing with loading them have been made in agreement of synchronous and asynchronous query processing model. Furthermore the Web-servers testing and comparison, there have been carried out some search and setting of configuration to get more optimized solutions for the web server that allows to withstand the more number of queries.
Keywords: fault tolerance, web-server, an asynchronous request processing model, stress testing, Node.js, Nginx, Apache, Siege.

Вступ. На сьогоднішній день ми спостерігаємо значний ріст інтернет-технологій та комунікацій, які все більше і більше масштабуються та нарощуються в своїй масі високими темпами. Зокрема така тенденція відчутна при розробці веб-додатків чи веб-сайтів із великою кількістю користувачів. При проектуванні багатокористувацьких веб-проектів слід враховувати сам підхід і технологію вибрану для їх реалізації, адже від цього залежить, яким чином вібуватиметься функціонування, масштабування та відмовостійкість даного веб-рішення в майбутньому.

Актуальність проблеми. Розглянемо основні аспекти та проблематику, що пов'язана з розробкою веб-проектів із великою кількістю одночасних запитів та їх безвідмовною обробкою. Як правило, професійні веб-додатки використовують клієнт-серверну програмно-архітектурну модель. Клієнтом виступає браузер користувача, а сервером – наприклад, одна з машин в дата-центрі деякого хостера. Браузер запитує якийсь ресурс у сервера, а той, у свою чергу, його віддає клієнту. Таким чином, у нас є модель найпростішого веб-сервера – програмна модель, яка приймає запити від браузера, обробляє їх і повертає відповідь. Звучить зрозуміло, але такий найпростіший сервер вміє одночасно спілкуватися лише з одним користувачем. Якщо в момент обробки запиту до нього звернеться ще один клієнт, то користувачу доведеться чекати поки сервер відповість першому. А в разі одночасного звернення до сервера, наприклад, десятків користувачів, може відбутися збій у його роботі і, як результат, ніхто не отримає відповідь на свій запит. Для того, щоб уникнути цього, необхідно налаштувати обробку запитів між користувачами так, щоб кожен з користувачів безвідмовно одержав відповідь на свій запит. Очевидне рішення: обробляти запити користувачів в окремих потоках або процесах операційної системи. На сьогоднішній день у деякому розумінні даного підходу дотримуються майже всі сучасні веб-сервери. Найбільш відомими серед них є такі, як, наприклад, Apache або IIS.

Але якщо виникає необхідність обробляти одночасно тисячі запитів, а той більше, то для ефективності необхідно повністю змінити ідеологію обробки запитів. І причин для цього є декілька. По-перше, як уже згадувалось вище, у синхронному середовищі запити та їх обробка відбувається послідовно. І навіть перерозподіл цієї послідовності на велику кількість потоків не дасть очікуваної ефективності на таку кількість одночасних запитів. Це буде призводити до неефективного використання серверних ресурсів. І як результат, під час передачі даних по мережі, веб-сервер більшу частину свого часу буде простоювати. По-друге, якщо пробувати вирішувати дану проблему створенням розподілених потоків, то доведеться зустрітися з проблемою неефективного використан-

ня оперативної пам'яті (тобто, якщо створювати окремий потік для кожного з тисячі користувачів, то незабаром можемо опинитися в ситуації, коли на сервері пам'яті просто не залишиться).

Рішенням в даній ситуації – це використовувати асинхронну обробку запитів на серевері. Для того, щоб реалізувати такий підхід, необхідно організувати усе введення-виведення і роботу з базою даних за допомогою неблокуйочої архітектури, побудованої на подійно-орієнтованії парадигмі, так званій "evented I/O". Одною із не багатьох платформ, які пропонують такі можливості є "Node.js".

Аналіз досліджень. Дослідження проблематики асинхронності обробки запитів на сервері залишаються актуальними на сьогоднішній день. Результатом пошуку компромісу між багатопоточністю та однопоточністю обробки даних на стороні сервера стала розробка web-платформи під назвою "Node.js", переваги якої наведені у статті [1].

У публікації [2] наведено модель реалізації такого типу сервера в поєднані з CMF Drupal та короткий опис процесу його розробки, що у свою чергу було підкріплено результатами тестування даної платформи.

У статті [3] наведено приклад порівняння двох серверів, а саме Apache і на базі платформи Node.js. Було проведено дослідження відмовостійкості серверів та здійснена порівняльна характеристика, яка виявила перевагу "Node.js" над "Apache". Під час спроб взлому (DOS-атаки) web-сервера Apache ситуація виглядала наступним чином: початок атаки сервер не відчуває ніякої завантаженості, але після великої кількості запитів у розмірі *1000000* було виявлено зниження працездатності. Тест показав, що завантаження жорсткого диска дорівнює *97*%, але сервер Apache через короткий проміжок часу (*7-10* с) знову нормалізовувався. Далі відбувався збій пов'язаний з блокуванням (перевантаженням) каналу і маршрутизатор не відповідав, відповідно до сервера не приходили запити і він був не доступний. Стандартний захист маршрутизатора пропускав *100*% пакетів протоколу TCP/IP і відповідно канал перевантажувався помилковими запитами, після чого пристрій перестав функціонувати і починав перезапускатися. Як висновок – атака пройшла успішно. В той час як у Node.js ситуація більш оптимістична. Початок атаки проходить доволі успішно, потік пакетів транслюється на сервер, але протягом *5-10* с пакети починають блокуватися маршрутизатором і вже пропускати лише *5-10*% всіх вхідних пакетів на сервер. У підсумку сервер обробляє запити на достатньому по надійності рівні. Підсумок – атака пройшла безуспішно.

В дослідженні встановлено, що успішність захисту від DOS і DDOS залежить не тільки від поставленого серверного обладнання а й від серверної програми. Не маючи потужного устаткування, у разі використання сервера на базі платформи Node.js спостерігається висока працездатність навіть при навантаженні вище середнього на відміну від сервера Apache.

Опис способу використання хмарних обчислень з використанням асинхронних web-серверів наведено у праці [4], де було описано алгоритм та методи реалізації асинхронної моделі, що використовують асинхронний ввід/вивід за допомогою зворотніх викликів, в результаті якої було побудовано навчальний чат-сервер.

Виклад основного матеріалу та обґрунтування отриманих результатів. Практична частина дослідження відмовостійкості веб-серверів розпочалось із навантажувального тестування. Тестуючим інструментом виступала утиліта під назвою "Siege". За допомогою цієї утиліти відбувалось генерування запитів до сервера та формувались дані про проведене навантажувальне тестування, серед яких:

- загальна кількість запитів;
- надійність обробки запитів;
- час потрачений обробку усіх запитів;
- кількість даних передана сервером для усіх запитів;
- середній час відклику сервера на запит;
- середнє число запитів, яке сервер зумів обробити за секунду;
- середнє число даних передане від сервера до клієнта за секунду;
- кількість конкурентних запитів, які сервер зумів обробити без затримок;
- кількість вдало оброблених запитів;
- кількість невдало оброблених запитів;
- найдовший час обробки запиту;
- найменший час обробки запиту.

За допомогою отриманої інформації, побудуємо графічні залежності та обґрунтуємо висновки щодо роботи веб-серверів під навантаженням.

Для початку слід зазначити, що тестування відбувалось із використанням локального веб-сервера під управлінням операційної системи Ubuntu 16.04. Практична частина навантажувального тестування складалась із трьох тестів:

1) Навантажувальне тестування веб-серверів, які генеруватимуть і віддаватимуть html-сторінку, на якій буде виводитись лише привітання.

2) Навантажувальне тестування веб-серверів, які обраховуватимуть *20*-й член послідовності Фібоначчі та віддаватимуть результат клієнту.

3) Навантажувальне тестування конфігураційного рішення, яке складалось із Nginx та Node.js, які разом реалізовували один веб-сервер з покращеними навантажувальними характеристиками.

Процес тестування відбувався ітеративно. За допомогою тестувальної утиліти, було згенеровано *100* одночасних запитів *100* разів, що у кінцевому результаті дало 100×100 запитів. Протягом усього тесту число одночасних запитів збільшувалось на *50* одиниць. Дана навантажувальнеа процедура відбувалась до того часу, доки сервер не перестав відповідати на запити.

Першим навантажувальним тестуванням для веб-серверів була html-сторінка, яка виводила привітання, згенероване на серверній стороні. Результат тестування, який полягає у знаходженні усередненого часу обробки запитів у секундах, при *100*-кратному повторенні експериментів наведено у таблиці 1 та у вигляді графіка на рис. 1.

Таблиця 1 – Результат тестування на основі сторінки-привітання веб-серверів (Тест 1)

Кількість запитів	**Apache, с**	**Node.js, с**
100	*6.9*	*4.94*
150	*11.37*	*7.85*
200	*14.16*	*11.46*
250	*18.8*	*13.4*
300	*29.72*	*17.16*
350	*30.64*	*21.03*
400	*32.87*	*23.96*
450	*34.09*	*28*
500	-	*34.85*
550	-	*33.35*
600	-	*35.88*
650	-	*44.38*
700	-	*66*
750	-	*46.25*
800	-	*46.98*
850	-	*51.14*

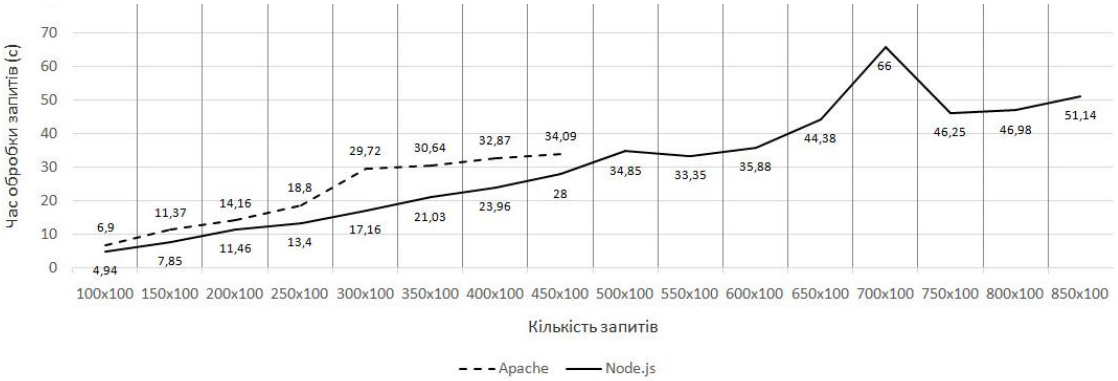

Рис. 1 – Залежність між кількістю запитів та усередненим часом затраченим на їх виконання (Тест 1)

У разі хоча б одного збою вважатимемо, що усередненого часу обробки запитів немає. Таким чином, з таблиці та рисунку видно, що веб-сервер на базі платформи Node.js витримує майже в два рази більше навантаження ніж Apache, а саме *850* одночасних з'єднань. Також даний веб-сервер зумів обробити аналогічну кількість запитів, в середньому, на *6.34* с швидше.

Другим навантажувальним тестуванням для веб-серверів була html-сторінка, згенерована на серверній стороні, яка виводила результат обчислення *20*-го члену послідовності Фібоначчі. Результат тестування при аналогічних вищеописаним умовах наведено у таблиці 2 та у вигляді графіка на рис. 2.

Таблиця 2 – Результат тестування на основі сторінки, що нараховує *20*-й член послідовності Фібоначчі, веб-серверів (Тест 2)

Кількість запитів	**Apache, с**	**Node.js, с**
100	*34.43*	*5.29*
150	*43.91*	*8.14*
200	*58.84*	*10.58*
250	*73.63*	*13.68*
300	*102.43*	*17.99*
350	*103.76*	*20.52*
400	*132.95*	*24.97*
450	-	*34.14*
500	-	*34.53*
550	-	*32.59*
600	-	*39.1*
650	-	*41.01*
700	-	*47.48*
750	-	*46.66*
800	-	*70.39*
850	-	*51.39*

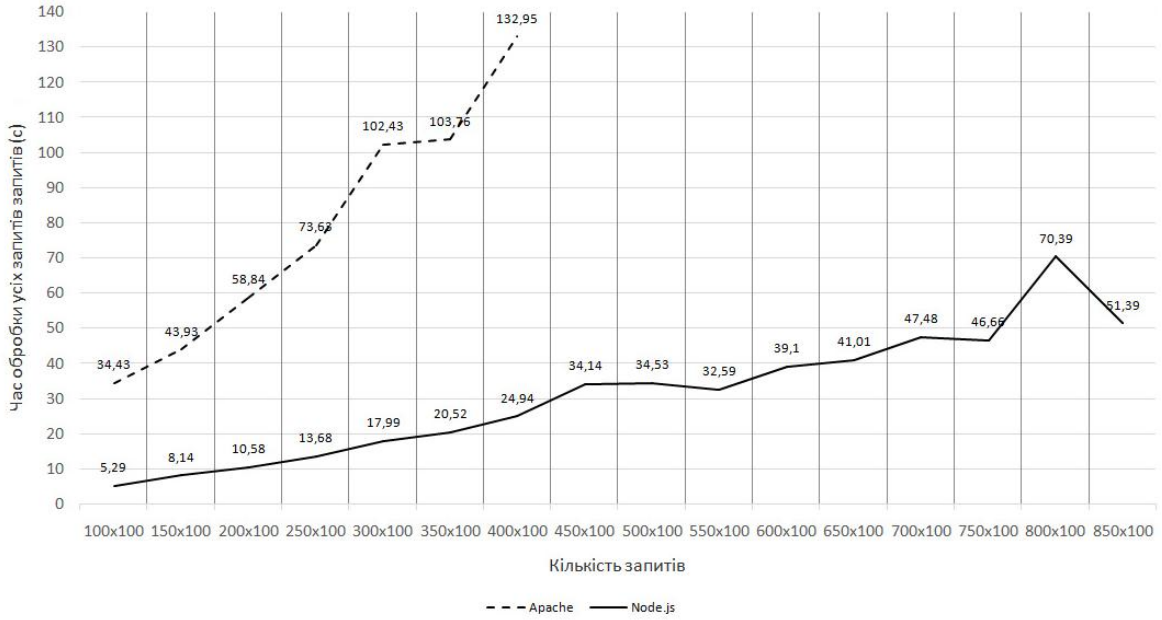

Рис. 2 – Залежність між кількістю запитів та усередненим часом затраченим на їх виконання (Тест 2)

Як видно із результату другого тесту, при виконанні більш трудомісткого процесу обчислення, сервер на базі платформи Node.js залишається в робочому стані і стабільно витримує високу кількість з'єднань, ніж веб-сервер Apache. Одержані з 2-го тесту результати демонструють значну перевагу роботи асинхронної архітектури обробки запитів. Щодо багатопоточної технології обробки запитів, то вона не лише дає збій при одночасній кількості запитів більшій за *400*, але і демонструє набагато більший час обробки запитів, що пов'язані з незначними математичними розрахунками.

Для більш оптимізованої обробки запитів була налагоджена взамодія веб-сервера Nginx, задача якого обробляти та віддавати статичні файли, з асинхронною технологією Node.js, на яку запропоновано покласти основну функцію роботи з базами даних, та інші динамічні операції. Спочатку було протестовано вищеописану конфігурацію сервера, який під час запиту зчитував дані із бази даних, та паралельно виводив статичний контент у вигляді картинок. Після цього були виконані певні налаштування Nginx для того, щоб він по вищеописаній схемі взаємодіяв з асинхронною технологією Node.js. В даному випадку Nginx виступав в ролі проксі-сервера, який обробляє запити, результатами яких є віддача статичних файлів клієнту. А технологія Node.js використовувалася для виконня певних обчислень (наприклад, реалізація асинхонності, взаємодія із базами даних тощо). Запропоновано "легенький" веб-ресурс, який має у контенті *10* статичних картинок та інформацію, що витягується з бази даних. Під поняттям "легенький" мається на увазі, що розроблено ресурс, у якому на обробку одного запиту, який пов'язаних із підтягуванням

картинок та взаємодією з БД, повинно витрачатися якомога менше часу. Результат тестування при аналогічних вищеописаним умовах наведено у таблиці 3 та у вигляді графіка на рис. 3.

Таблиця 3 – Результат тестування веб-сервера оптимізованої конфігурації (Тест 3)

Кількість запитів	**Apache, с**	**Node.js, с**
100	*9.97*	*7.1*
150	*15.4*	*9.64*
200	*19.57*	*13.37*
250	*25.48*	*16.32*
300	*30.21*	*20.02*
350	*37.3*	*23.39*
400	*66.12*	*27.33*

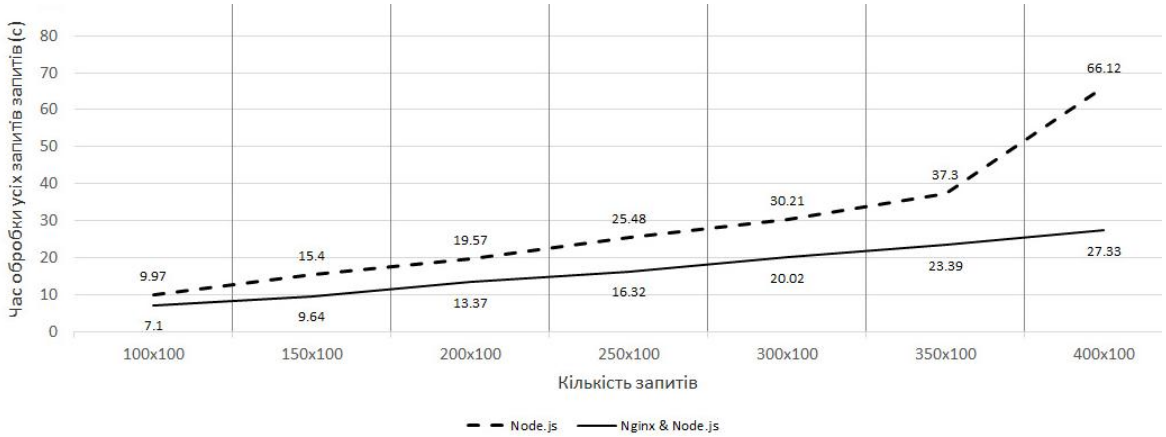

Рис. 3 – Залежність між кількістю запитів та усередненим часом затраченим на їх виконання (Тест 3)

Отриманий результат третього тесту демонструє приріст продуктивності роботи технології Node.js в парі з проксі-сервером Nginx. При кількості одночасних запитів меншій за *400* час їхньої обробки в даному програмно-архітектурного конфігураційного рішенні суттєво зменшився, що є дуже важливим для досягнення максимальної відмостійкості в умовах великої кількості запитів та навантажень.

Висновки. Дослідивши та протестувавши роботу веб-серверів із синхронною та асинхронною моделлю обробки запитів, було отримано дані та залежності, з яких можна зробити наступні висновки. Спочатку було з'ясовано, що асинхронна технологія обробки запитів працює ефективніше (швидше і безвідмовніше), ніж багатопоточна. При цьому вона витримує більшу кількість одночасних запитів. Після цього було побудовано програмно-архітектурне конфігураційне рішення, яке полягає у перерозподілі часу обробки запитів між синхронною та асинхронною технологіями: на першу частину роботи, що пов'язана з підтягуванням статичної інформації, не було задіяно технологію асинхронності; а на другу, що пов'язана зі взаємодією з базами даних та з іншими динамічними операціями, відповідно задіяно дану технологію. Результат останнього тестування є позитивним і він показує, що розроблене в даній роботі конфігураційне рішення може використовуватися при побудові високонавантажувальних систем таких як: чат-сервери, соціальні мережі, багатопоточні веб-додатки, та інші веб-проекти, де необхідно опрацьовувати велику кількість запитів одночасно і в реальному часі.

1. Книга О.О. Аналіз переваг використання серверу Node.js для високонавантажених веб-додатків / Збірник тез Харківського національного економічного університету.
2. Пех П.А. Дослідження ефективності синхронних та асинхронних технологій програмування у процесі розробки веб-додатків / П.А. Пех, В.О. Бондарчук // Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництвто". – Луцьк, 2015. – № 18 – С. 73-76.
3. http://www.sworld.com.ua/index.php/ru/conference/the-content-of-conferences/archives-of-individual-conferences/march-2013
4. http://www.ibm.com/developerworks/ru/library/cl-nodejscloud