МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна інженерія»

Група: 2БКС-27

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

здобувача освіти денної форми навчання

БКС.27.11.000.КРБ

ЗАДВОРНОГО МАКСИМА ЄВГЕНОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітня програма: **«Комп'ютерна інженерія»**

Група: **2БКС-27**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **«Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів»**

Проектний матеріал складається з пояснювальної записки на 65 сторінках та графічного (презентаційного) матеріалу на 22 аркушах (слайдах)

Виконавець _____ (Задворний М.Є.)

Керівник проекту _____ (Кривченко Ю.В.)

Консультанти:

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «24» 06 2023 р. Протокол ДКК № 2

Оцінка ДКК 5 (відмінно)

Секретар ДКК _____

АНОТАЦІЯ

У випускній кваліфікаційній роботі досліджено алгоритми розподілу потоків для підвищення продуктивності Web-ресурсів. Визначено, описано та досліджено проблему підвищення продуктивності Web-ресурсу шляхом розміщення його не на одному, а на декількох комп'ютерах різної потужності. Передбачено розміщення сервера-додатка балансувальника навантаження на окремому комп'ютері. Обрано оптимальний алгоритм розподілу потоків і найкращий варіант при додаванні до кластеру нових серверів.

Розглянуто програмні засоби реалізації, використані в процесі проведення експериментів. Проведено порівняльні експерименти різних методів балансування навантаження на різних конфігураціях обладнання.

Отримано графіки поведінки системи у різних ситуаціях. Отримані результати поведінки системи при часткових несправностях. Отримано результати, які свідчать про зменшення часу відповіді на запит до Web-ресурсу при використані обраного алгоритму балансування навантаження та додатку-серверу.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення Комп'ютерних систем Кафедра Комп'ютерної інженерії

Спеціальність 123 «Комп'ютерна інженерія»

Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР Беркань І.В.

"_____" _____ 202___ р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

Здобувачеві (здобувачці) освіти Задворному Максиму Євгенович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів

затверджена наказом по коледжу від "17" жовтня 202 2 р. № 235-А2-ОД

2. Термін здачі кваліфікаційної роботи 12.06.2023

3. Вихідні дані до роботи 1. Архітектура системи балансування навантаження на статичному та динамічному змісті; 2. Результати огляду та аналізу проблем високонавантажених комп'ютерних систем та Web-ресурсів; 3. Використовувати два комп'ютера у якості серверів та комп'ютер балансувальника навантаження; 4. Проаналізувати зміни продуктивності в залежності від часткової відмови системи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз методів і засобів розподілу потоків Web-ресурсів

Вибір програмних рішень балансувальників навантаження

Вибір апаратних засобів для дослідження алгоритмів розподілу потоків

Тестування роботи алгоритмів розподілу потоків та балансувальників навантаження

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Схема розподілу потоків навантаження; Блок-схема роботи системи розподілу потоків навантаження Web-серверів; Принцип роботи алгоритму Round Robin; Механізм балансувальника навантаження з використанням алгоритму Weighted Least Connections; Організація мережі з балансувальником навантаження HAProxy; Організація мережі з балансувальником Nginx; Схема експериментальних досліджень; Графіки продуктивності балансувальників навантаження на статичному змісті; Результати тестування продуктивності балансувальників навантаження на динамічному змісті; Максимальний рейтинг з'єднань алгоритмів розподілу потоків

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів роботи, що стосується їх

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Технологічний	Кривченко Ю.В.		8.06
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Кривченко Ю.В.		

7. Дата видачі завдання 12.06.2023

Керівник роботи Кривченко Ю.В. _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів кваліфікаційної роботи	Термін виконання етапів кваліфікаційної роботи	Примітка
1.	Вступ. Аналіз технічного завдання	4.05.23	Виконано
2.	Аналіз методів і засобів розподілу потоків	7.05.23	Виконано
3.	Визначення рівнів OSI для розподілу потоків	9.05.23	Виконано
4.	Огляд алгоритмів розподілення потоків	10.05.23	Виконано
5.	Створення моделі розподілу потоків		Виконано
	навантаження web-серверів	12.05.23	Виконано
6.	Вибір програмних рішень		Виконано
	балансувальників навантаження	13.05.23	
7.	Вибір апаратних засобів для дослідження		Виконано
	алгоритмів розподілу потоків	14.05.23	
8.	Налаштування програмних засобів для		Виконано
	тестування	16.05.23	
9.	Дослідження приросту продуктивності на		Виконано
	статичному змісті	18.05.23	
10.	Дослідження приросту продуктивності на		Виконано
	динамічному змісті	19.05.23	
11.	Аналіз результатів тестування	1.06.23	Виконано
12.	Розробка питань з охорони праці	3.06.23	Виконано
13.	Підготовка матеріалів мультимедійної	6.06.23	Виконано
	презентації		

Виконавець _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 27. 11 000. 00 КРБ	Випускна робота		
А4			БКС 27. 11 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 27. 11 000. 00 КРБ		
Розробив		Задворний М.Є.			Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів	Літ.	Аркуш
Перевірив		Кривченко Ю.В.				Н Д П	5 / 65
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.2БКС-27	
Затверд.		Іванова Л.В.					

ЗМІСТ

ВСТУП

Web-додатки сучасного типу не зберігають готовий HTML-код, а генерують його під час запиту. Це зумовлено тим, що на ресурсах постійно відбуваються зміни. Саме цей динамічний зміст підставляється у визначені місця Web-сервером під час генерування сторінки.

На сьогоднішній день будь-який популярний ресурс має тенденцію зростання кількості користувачів. Коли серверної потужності буде не вистачати при більшій кількості користувачів, можна модифікувати серверну машину, збільшити обсяг її оперативної пам'яті, змінивши процесор на більш потужний, встановити ємкий і швидкий SSD-накопичувач. Але у такого шляху модифікування теж є межа: навіть якщо повністю замінити серверну машину на найпотужнішу на цей час, продуктивність більшу, ніж дозволяють ресурси однієї машини, отримати буде неможливо. Іншим шляхом нарощування продуктивності обробки запитів є збільшення кількості серверних машин. Такому нарощуванню майже немає меж. Більша кількість серверних машин, очевидно, матиме більшу обчислювальну потужність, але їх треба поєднати, щоб розмістити на них один ресурс.

Для підвищення продуктивності обробки запитів використовується балансувальник навантаження Web-серверів. Застосовувати розподіл потоків навантаження доцільно, коли постає питання розширення можливості опрацьовувати системою більших обсягів запитів. У випадку розміщення ресурсу на більш, ніж на одному сервері, це стає необхідним заходом [1].

Існують програми для імітації навантаження, які допомагають дізнаватись реакцію серверу на велику кількість користувачів.

В даній випускній кваліфікаційній роботі буде досліджена проблема підвищення продуктивності Web-ресурсу шляхом розміщення його не на одному, а на декількох комп'ютерах різної потужності. На окремому комп'ютері буде розміщено сервер-додаток балансувальника навантаження. Треба обрати алгоритм розподілу потоків для підвищення продуктивності web-ресурсів і найкращий варіант при додаванні до кластеру нових серверів.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Аналіз методів і засобів розподілу потоків Web-ресурсів

Для економічно ефективного масштабування відповідно з великими об'ємами запитів треба додавати більшу кількості серверів, тобто виконувати горизонтальне масштабування. Для реалізації такого масштабування системи необхідно використовувати додатковий вузол, який забезпечить регулювання трафіку між групою серверів – балансувальник навантаження. Балансування навантаження (load balancing) забезпечує розподілення робочих навантажень по кільком обчислювальним ресурсам, таким як комп'ютери, кластери, мережеві з'єднання, центральні процесори чи диски зберігання даних. Балансування навантаження полягає у ефективному розподіленню вхідного мережевого трафіку між групою Backend-серверів. Зазвичай для задачі розподілення потоків використовується архітектура, показана на рис. 1.1.

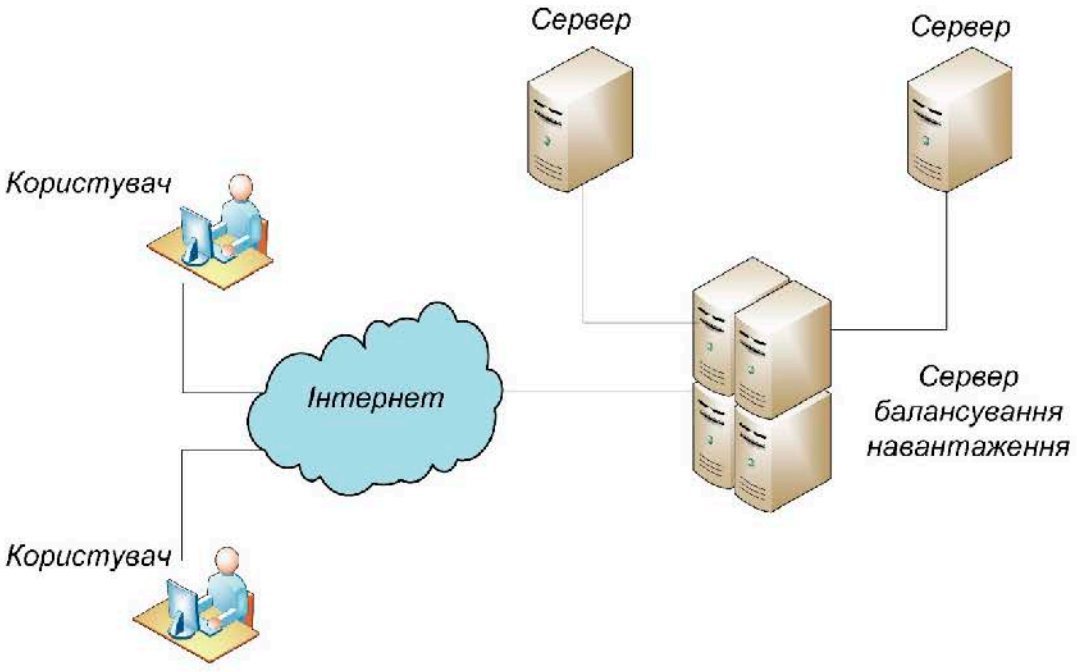


Рисунок 1.1. Схема розподілу потоків навантаження
для архітектури Frontend-Backend

Сервер балансування навантаження є Frontend-сервером, тобто сервером, на який надходять всі запити користувачів. У нього обов'язково має бути зовнішня IP-адреса у випадку, коли потрібно мати доступ до ресурсу через глобальну мережу Internet. Адреси самих Web-серверів можуть бути і «не видні» у інтернеті.

Це може бути у випадку, коли всі сервери та сервер з балансувальником знаходяться у спільній локальній мережі, і тільки порт та адреса «прокинуті» зовні. Задача перенаправлення трафіку не така складна і довга, як обчислення складних математичних формул, доступ до бази даних або очікування даних від стороннього API. Тому на це виділяється один сервер, на якому розміщується, наприклад, HAProxy, і конфігурується згідно до множини Backend-серверів, розташованих «поза» балансувальником. Сервери, зокрема сервер із встановленим балансувальником, віртуальні виділені сервери, можуть розташовуватися в локальній мережі [2].

1.1.1 Визначення рівнів OSI для розподілу потоків

Розподіл потоків web-ресурсів для балансування навантаження можливий на мережевому, транспортному та прикладному рівнях моделі OSI.

На мережевому рівні IP-адреси в заданій кількості призначаються одному DNS-серверу. При запиті на DNS-сервері відбувається вибір та перенаправлення запиту на одну із зазначених IP-адрес. Зазвичай використовується алгоритм Round Robin (рис.1.2).

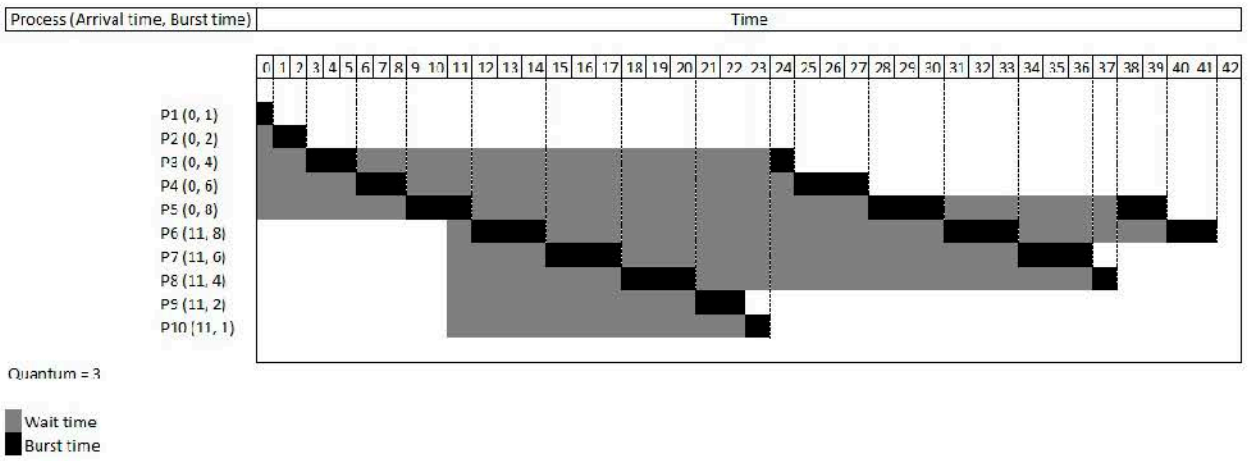


Рисунок 1.2. Принцип роботи алгоритму Round Robin

При побудові NBL-кластеру сервери об'єднуються в кластери. Ця технологія використовується компанією Майкрософт. Також може використовуватись додатковий апаратний пристрій для розподілу потоків, а саме маршрутизатор. При територіальному розподіленні запитів також використовують CDN – мережу, розподілену за геолокацією (рис.1.3). Так, якщо

треба збільшити здатність швидко завантажувати інформацію клієнтам, які знаходяться на іншому континенті, постає задача також налаштувати CND-сервер на відповідному континенті.

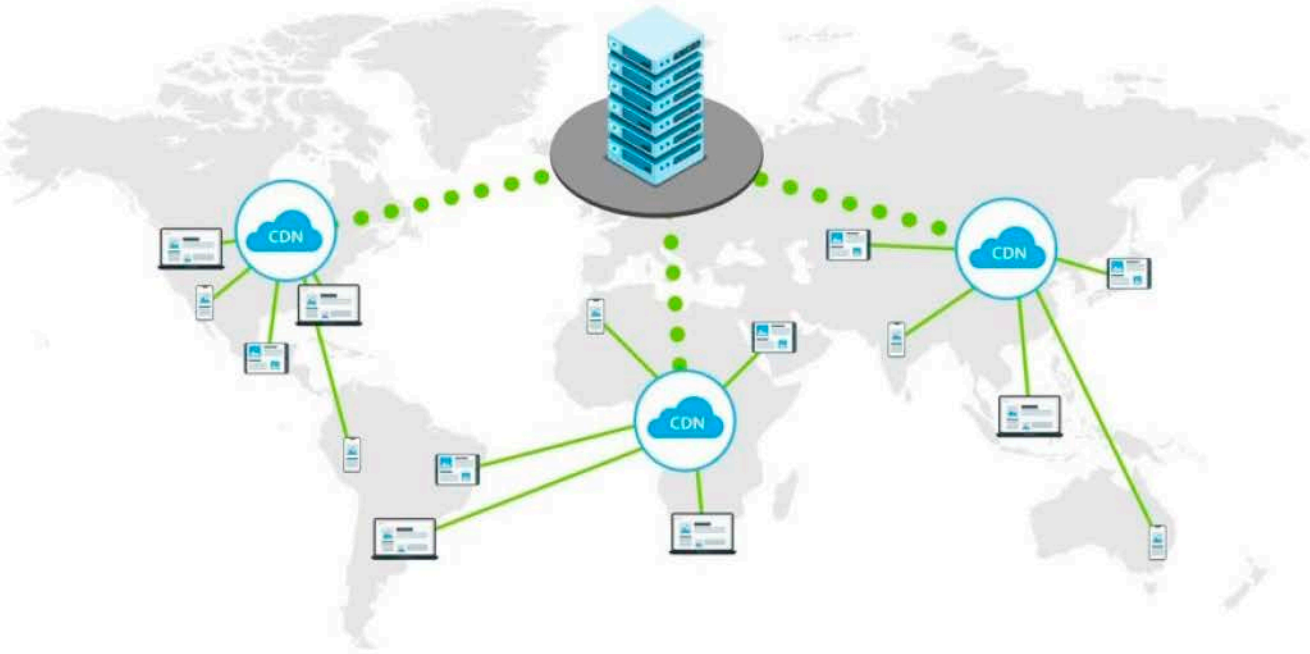


Рисунок 1.3. Організація CDN-мережі

На транспортному рівні розподіл потоків передбачає перенаправлення запиту на сервер згідно з алгоритмом. Перенаправлення може здійснюватися згідно з алгоритмом циклічного перебору всіх серверів шляхом вибору зі списку наймеш завантаженого [3].

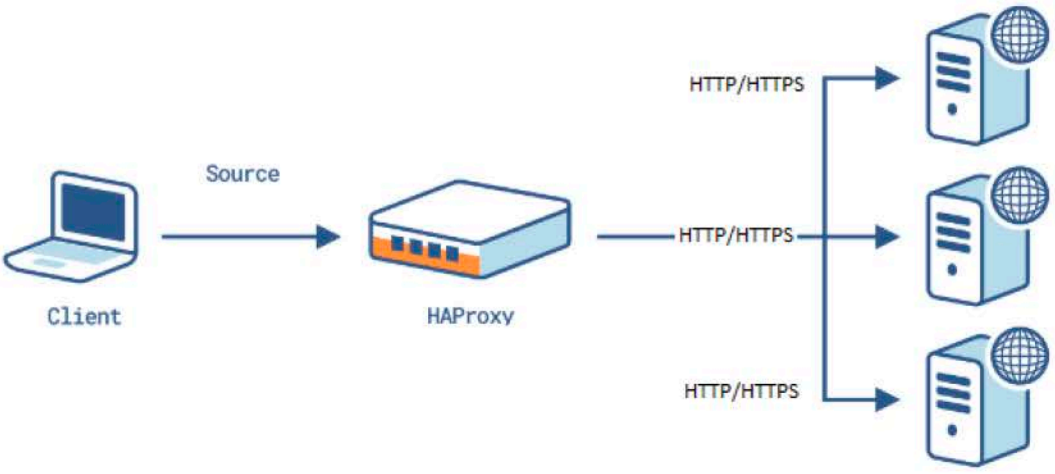


Рисунок 1.4. Організація мережі з балансувальником HAProxy

Відмінність від мережевого рівня, який не втручається у сесії користувачів, тобто перенаправляє трафік «як є», і сесію з клієнтом здійснює сервер, у тому, що на транспортному рівні клієнт контактує з балансувальником, а балансувальник, в

свою чергу, спілкується з Backendom. Фактично балансувальник виступає у ролі проксі-серверу, а інформація про клієнта передається у заголовках. HAProxy є гарним прикладом цього рівня розподілу потоків (рис.1.4).

На прикладному рівні розподілу потоків в залежності від запитуваного змісту балансувальник перенаправляє запити на потрібний сервер. Так працює Nginx у ролі балансувальника (рис.1.5). Без модуля Upstream це неможливо. У PostgreeSQL є pgpool – проміжна інстанція, яка може розподіляти запити за змістом. Зчитування та запис потраплятимуть на різні сервери, зокрема.

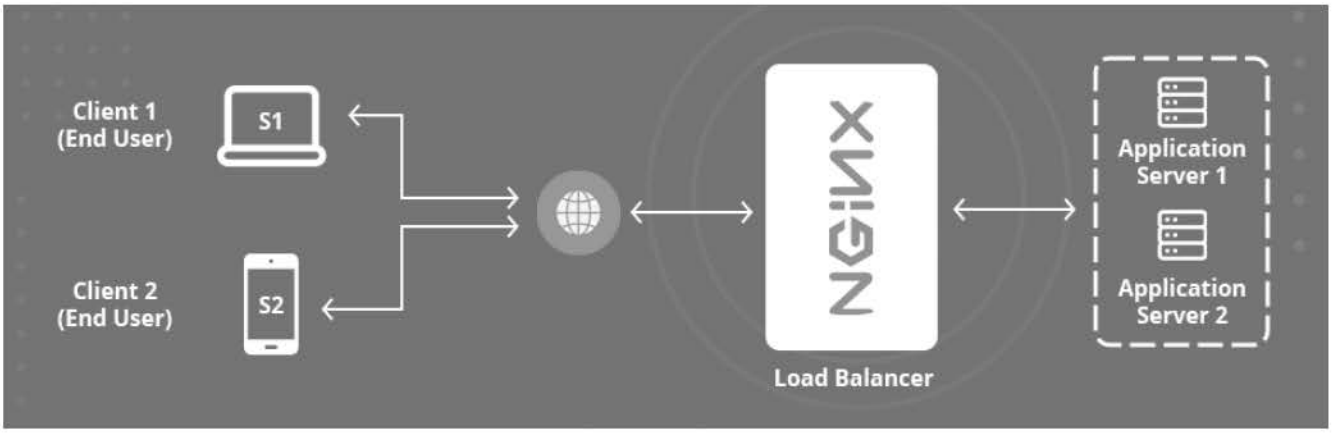


Рисунок 1.5. Організація мережі з балансувальником Nginx

1.1.2 Огляд алгоритмів і методів розподілення потоків

Наразі існує немало алгоритмів, що пропонують широкий спектр вирішення проблеми розподілу потоків. Відповідно до кінцевої мети треба ретельно підібрати потрібний алгоритм. Балансуванню мають бути притаманні такі якості:

- ефективність: жодна одиниця комп'ютерної техніки не має простоювати в той час, коли інші навантажені на сто відсотків;
- справедливість: кожен запит має бути оброблений. Не повинно виникати ситуації, у якій запит не обробиться або всі інші будуть чекати обробку запиту;
- скорочення часу відгуку;
- скорочення часу виконання запиту;
- передбачуваність: алгоритми мають бути чітко підібрані для того чи іншого випадку. Також треба знати, як кожен із них буде себе проявляти у різних ситуаціях;

– алгоритм повинен бути легко масштабованим за необхідності;

– однакове завантаження обчислювальних одиниць.

Далі розглянуто особливості найбільш популярних алгоритмів розподілення потоків.

Алгоритм Round Robin є алгоритмом кругового обігу. Запити циклічно передаються доступним серверам. Цей алгоритм можна застосовувати з комп'ютерними одиницями приблизно однакової потужності. Round Robin DNS – найпоширеніша імплементація цього алгоритму. DNS-сервер зберігає пару «ім'я хоста – IP-адреса» для кожної машини в певному домені. Сервер доменних назв проходить циклічно таблицю. З кожним наступним запитом буде наступна адреса. Перевагою цього алгоритму є незалежність від протоколу. У всіх випадках, де звернення до ресурсу виконується через ім'я, можна застосувати алгоритм Round Robin DNS. У алгоритмі Round Robin сервери не обов'язково мають бути поєднані між собою, що дозволяє використовувати його для глобального та локального розподілу потоків. Для того, щоб алгоритм запрацював, треба лише додати потрібні записи до серверу доменних назв, що зумовлює його низьку вартість, але також призводить до проблем. Серед недоліків є те, що в наявності потрібно мати лише однакові сервери, на які будуть надходити запити. В іншому випадку критерії ефективності та справедливості не будуть досягнуті. Іншим недоліком є те, що має бути використано однакову кількість ресурсів, що на практиці не виконується. Ще одним недоліком є неврахування навантаження на кожному із серверів. У алгоритмі навіть не передбачено ситуацію, у якій один із серверів за певних причин навантажений на 95 %, а інший на 5%. Вузол з навантаженням 95% все одно буде отримувати запити. Відповідно, сфера застосування цього алгоритму є вельми обмеженою [4].

Алгоритм Weighted Round Robin є покращеною версією алгоритму Round Robin. Його головна особливість полягає у тому, що можна встановлювати такий параметр, як вага. Сервери з меншою вагою оброблятимуть менше запитів, у свою чергу сервери з більшою вагою – більшу. Якщо є два сервери з потужністю 1 і 2, їх вага буде відповідно дорівнювати 1 і 2. Перший запит відправлятиметься на

обробку першому серверу, а 2 і 3 – другому серверу. Так вирішено проблему з тим, що потрібно мати серверні машини однакової потужності. Однак це вдосконалення не вирішує проблем з відмовостійкістю, яка залишається на тому самому рівні, що і у алгоритмі Round Robin.

Алгоритм Least Connections покликаний вдосконалити алгоритм Round Robin та врахувати кількість одночасно активних підключень. Наприклад, якщо перший сервер має 200 підключених користувачів, а другий – 2000 підключень, то перший сервер може бути завантажений навіть більше другого за рахунок того, що операції, які він має виконувати, потребують більше часу, тобто складніші. Зазначена проблема вирішується за допомогою алгоритму Least Connections, який враховує кількість активних підключень до кожного серверу. Таким чином кожний наступний запит потрапляє на сервер, у якого кількість підключень є мінімальною [5].

Алгоритм Weighted Least Connections є вдосконаленим варіантом попереднього алгоритму. Працює він таким самим чином, але враховує ще й потужність серверів. У вдосконаленого алгоритму існує ще кілька варіантів реалізації: Locality-Based Least Connection Scheduling і Locality-Based Least Connection Scheduling with Replication Scheduling. Метод Locality-Based Least Connection Scheduling був розроблений для кешуючих proxy-серверів. Працює він таким чином: до серверів з найменшою кількістю підключень передається більша частина запитів. Кожний такий сервер закріплює за собою та обслуговує групу адрес. Запити з цих адрес направляються на головний сервер, якщо його потужність не вичерпана. Якщо ж це сталося, запит перенаправляється на другий сервер, що менш завантажений, а якщо і його обчислювальна здатність вичерпана, то на третій і т.д.

У варіанті реалізації алгоритму Locality-Based Least Connection Scheduling with Replication Scheduling кожна IP-адреса або група IP-адрес закріплюється не за окремим сервером, а за цілою групою серверів. Запити передаються найменш завантаженому серверу з групи. Коли всі сервери із своєї групи перевантажені, виділяється новий сервер. Сервер додається до групи, яка в той час обслуговувала

певний IP. Найбільш завантажений сервер з групи буде знищено, тобто видалено зі списку, що дає можливість уникнути великої кількості реплікацій.

Алгоритм Destination Hash Scheduling створений для роботи із кластерами кешуючих proxy-серверів. Цей алгоритм використовується не тільки за призначенням. Існує статична таблиця IP-адрес, з якої обирається потрібний сервер. Алгоритм Source Hash Scheduling базується на тій самій ідеї, що і Алгоритм Weighted Least Connections, але з особливістю того, що ресурс, котрий призначається для підготовки відповіді, призначається згідно інтернет-адреси відправника та таблиці.

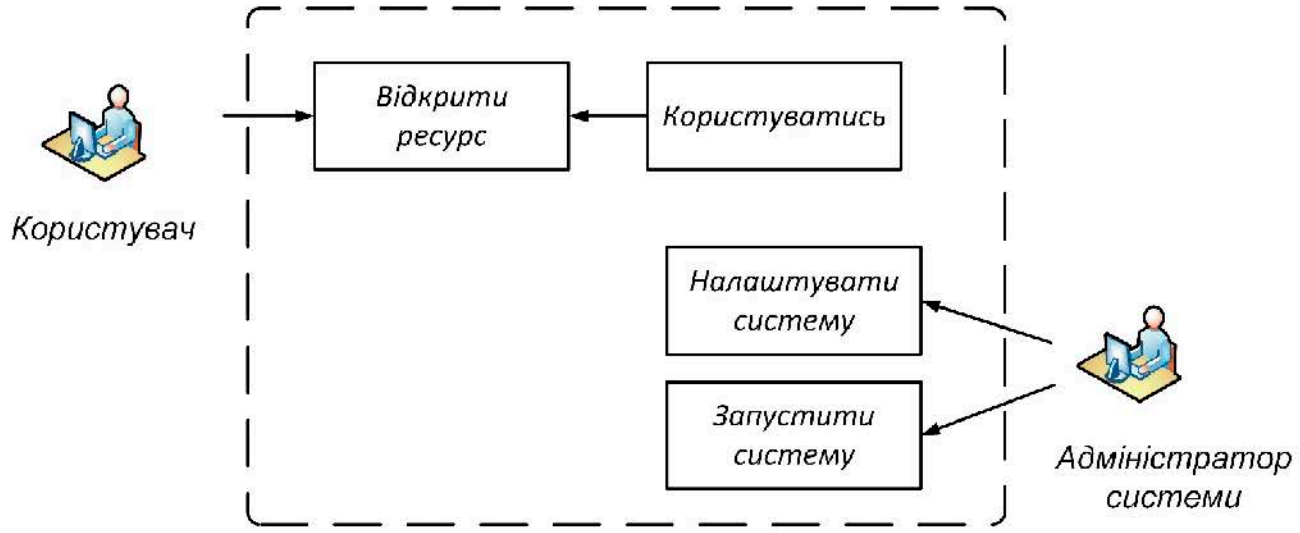


Рисунок 1.6. Схема роботи системи розподілу потоків навантаження Web-серверів

Прикладом реалізації алгоритму Sticky Sessions є Web-сервер Nginx. Запити направляються одному й тому самому серверу групи. IP-адреса є маркером для закріплення сесій до певного серверу. Як зазначено в документації, метод гарантує, що запити одного і того самого клієнта передаються на один і той самий сервер. Якщо закріплений за конкретною адресою сервер недоступний, запит буде перенаправлено на інший сервер. В Nginx для кожного сервера можна вказувати вагу (Weight) починаючи з версії 1.2.2.

1.1.3 Створення моделі розподілу потоків навантаження web-серверів

Кожен з розглянутих у п.1.1.2 алгоритмів має свої переваги та недоліки. Алгоритм DNS Round Robin має суттєвий недолік: при вимкненні одного з серверів та потраплянні цього серверу у вибір балансувальника користувач

отримує помилку як при вимкненому сервері, не дивлячись на те, що сервери будуть працювати правильно. Немає очевидної відповіді, який алгоритм використовувати оптимальніше, але вже зараз можна сказати, що Weighted-версії алгоритмів незамінні при різних потужностях комп'ютерів. При наявності серверів з різними характеристиками необхідно використовувати алгоритми, що враховують різні потужності комп'ютерів. На рис.1.6 показано схему роботи системи розподілу потоків навантаження Web-серверів, а на рис.1.7 – пропоновану блок-схему алгоритму системи розподілу потоків навантаження Web-серверів.

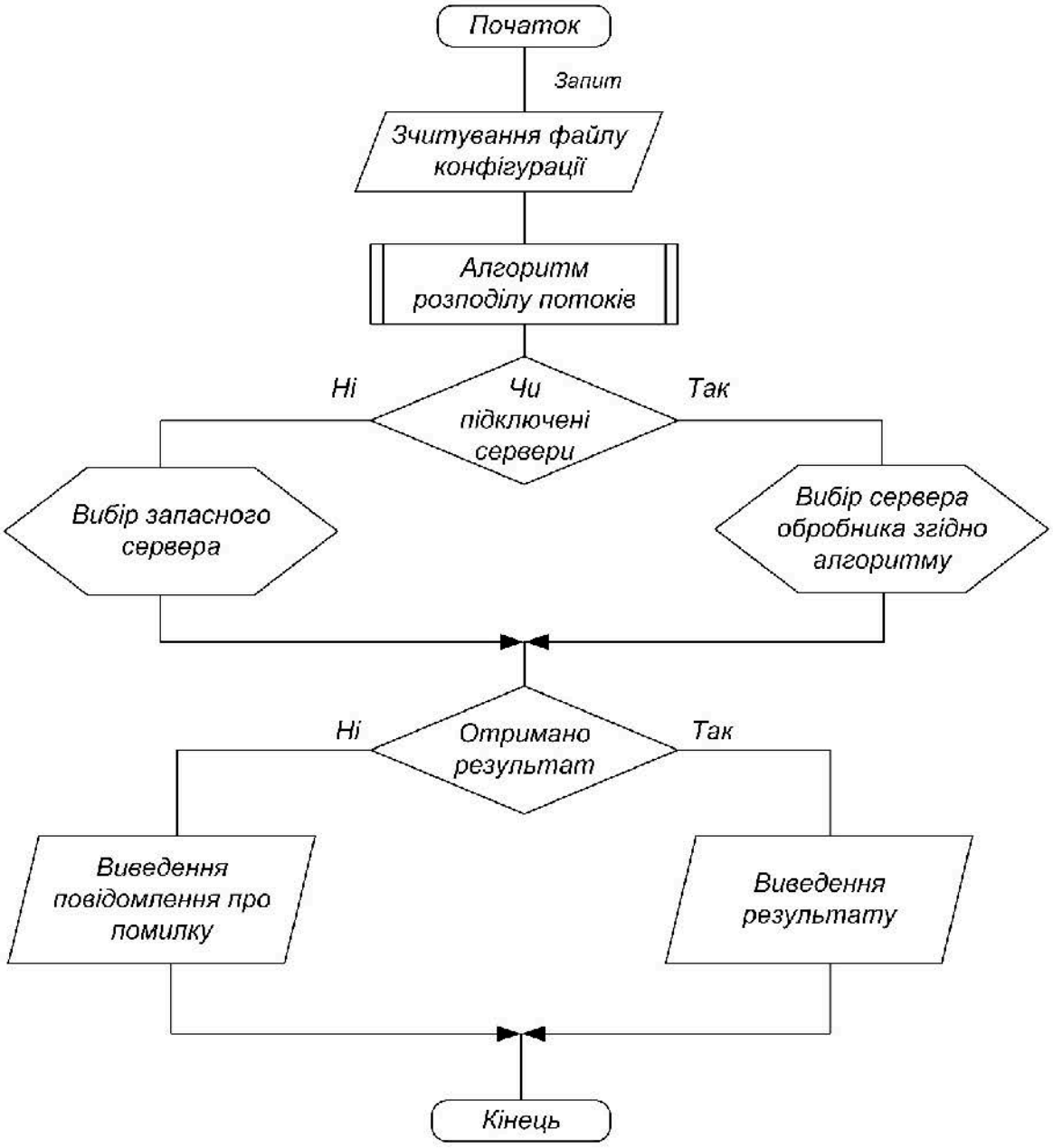


Рисунок 1.7. Блок-схема роботи системи розподілу потоків навантаження Web-серверів

1.2 Вибір програмних рішень балансувальників навантаження

У даному підрозділі серед присутніх на ринку балансувальників навантаження Web-серверів будуть виділені найбільш популярні рішення для подальшого дослідження ефективності їх алгоритмів розподілу потоків з метою підвищення продуктивності web-ресурсів.

1.2.1 Балансувальник навантаження Traefik

Програмне забезпечення Traefik є сучасним HTTP-зворотним proxy-сервером і балансувальником навантаження, що полегшує розгортання мікросервісів. Traefik інтегрується з існуючими компонентами інфраструктури (Docker, Swarm, Kubernetes, Marathon, Consul, Etcd, Rancher, Amazon ECS) і налаштовується автоматично і динамічно. Для того, щоб налаштувати Traefik, треба вказати на нього у контейнері Docker. Traefik постійно оновлює свою конфігурацію без перезапусків, швидко працює, підтримує багато алгоритмів розподілу потоків навантаження, підтримує HTTPS. В ньому присутній Web-інтерфейс, підтримується Websocket, HTTP/2, GRPC.

1.2.2 Балансувальник навантаження HAProxy

Програмне забезпечення HAProxy є безкоштовним, дуже швидким і надійним рішенням, що пропонує високу доступність, розподіл потоків навантаження і Proxy для TCP- і HTTP-додатків. Особливо HAProxy підходить для Web-сайтів з високим рівнем трафіку і великою кількістю відвідувань сайтів. Протягом багатьох років HAProxy став де-факто стандартним балансувальником навантаження з відкритим кодом та поставляється з більшістю основних дистрибутивів Linux, часто розгортається за замовчуванням у хмарних платформах. Передбачені режими роботи роблять його інтеграцію в існуючі архітектури дуже легкою і мінімізують ризики, пропонуючи можливість не виставляти у мережу слабкі Web-сервери. Програмне забезпечення HAProxy дозволяє виконувати розподіл потоків на рівні додатків. Особливість полягає у використанні cookie для регулювання підключень. HAProxy дозволяє стабільно підтримувати 20 тис. одночасних підключень, передбачає запис логів, звітів. Для

налаштування можна використовувати Web-інтерфейс Snapt. HAProxy використовують такі масштабні проекти як Amazon RDS, Twitter GitHub, Stack, Server Fault, Overflow.

1.2.3 Балансувальник навантаження Pound

Програмне забезпечення Pound є зворотним Proxy, балансувальником навантаження та інтерфейсом HTTPS для Web-серверів. Pound був розроблений для того, щоб дозволити розподілити навантаження між декількома Web-серверами і дозволити зручну обгортку SSL для тих Web-серверів, які не пропонують її за замовчуванням. Програмне забезпечення Pound поширюється під GPL-ліцензією. Pound передбачає такі режими роботи: зворотний проксі-сервер, розподілення запитів від клієнтів між серверами, розшифровування SSL: передання HTTPS-запиту за протоколом HTTP до серверу, виключення неробочого серверу зі списку готових приймати запити серверів [6].

1.2.4 Балансувальник навантаження Nginx

Програмне забезпечення Nginx є одним із найбільш продуктивних Web-серверів. Його можна використовувати у ролі балансувальника навантаження. Для цього використовується модуль upstream module. Цей модуль має безліч можливих налаштувань, надаючи дуже гнучкий, а головне – надійний інструмент для розподілення запитів між Backend-ами. Реалізуються такі алгоритми, як Round Robin, IP-hash, Least-connected.

1.2.5 Порівняння програмних рішень балансувальників навантаження

Кожен з описаних вище програмних продуктів має свої недоліки та переваги. Загалом вони реалізують майже всі алгоритми, описані у підрозділі 1.1. Ці дані зібрані або з інструкцій відповідного програмного забезпечення, або з відкритих досліджень у мережі Інтернет. Наведена інформація не є вирішальною для вибору оптимального рішення (зокрема, через відмінності у апаратних засобах, що були використані у дослідженнях, а також через те, що використовувалися дані відкритих неперевірених ресурсів), але дозволить мати деякі початкові дані для досліджень у наступних підрозділах.

У дослідженні, що буде виконано, треба проілюструвати як і які саме алгоритми працюють у складі тих чи інших програмних засобів. Необхідно визначити, який саме алгоритм розподілу потоків та який програмний продукт для більшої кількості конкурентних користувачів будуть забезпечувати мінімальний час відгуку.

Таблиця 1.1. Характеристики програмних рішень балансувальників навантаження

	HAProxy	**Nginx**	**Traefik**	**Pound**
Операційна система	Linux 2.6.32+ (рекомендовано для максимальної продуктивності) та 2.4, Solaris 8–10, FreeBSD, OpenBSD	Windows, CentOS, Debian, Ubuntu, SLES, Apline	Docker-контейнер	Linux, Solaris, OpenBSD
Алгоритми	Round Robin, Random, Weighted Random, Least Session, Least Bandwidth, Hash, Agent і Randomized Agent	Weighted Round Robin, Weighted Least-connections, IP-hash	Weighted Round Robin, Dynamic Round Robin	Least Session, Round Robin
Протоколи	HTTP, HTTPS	HTTP, HTTPS, FastCGI, SCGI, gRPC	HTTP, HTTPS	SSL, HTTP(s)

1.3 Вибір апаратних засобів для дослідження алгоритмів розподілу потоків

З метою дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів застосовані достатньо потужні апаратні засоби, доступ до яких отримано протягом переддипломної практики. До складу апаратних засобів входять мережеві маршрутизатори та робочі станції з встановленими програмними засобами, описаними у п.1.2. Досліджувана система складається з декількох Web-серверів та серверу розподілу потоків, а також програмного забезпечення для тестування навантаження. Два комп'ютера виконують роль серверів та клієнтів. Маршрутизатор виконує роль пристрою зв'язку. Така мережева система представляє собою спрощену версію VPS-хостингів, що є сучасним видом надання послуг хостинг-провайдерами. Хостинг-

провайдери не надають VPS безкоштовно, тому було запущено декілька віртуальних серверів, які загалом не відрізняються від тих, що на хостингу та тих, які умовна IT-компанія придбала б для розміщення свого ресурсу. Це максимально наближує досліджувану систему до тієї, яку можна використовувати у реальності.

У складі досліджуваної мережі два комп'ютера використані для того, щоб максимально відділити віртуальні сервери від будь-яких чинників. Комп'ютер-сервер буде містити тільки віртуальні машини зі встановленими Web-серверами, по одному на кожній, та один балансувальник навантаження. Комп'ютер-клієнт буде містити тільки віртуальну машину із засобами для тестування навантаження. Віртуальні машини використовуються для обмеження обчислювальної здатності. Конфігуратор віртуальних машин дозволяє надати стільки обчислювальної здатності кожній віртуальній машині, скільки потрібно. Проблемою розподілу потоків є те, що різні за потужністю сервери викликають проблеми для деяких алгоритмів, тому у роботі виконується моделювання саме цього випадку [7].

В якості операційної системи використовується ОС Red Hat Enterprise Linux. Це пов'язано з тим, що на серверах підприємства, де проходила переддипломна практика, встановлена саме операційна система Red Hat Enterprise Linux. Крім того, всі розглянуті у п.1.2 програмні засоби балансувальників навантаження працюють під керуванням цієї операційної системи. Для створення віртуальних машин була використана безкоштовна версія системи віртуалізації Virtual Box 7. Virtual Box є віртуальною машиною від компанії Oracle. Серверний та клієнтський комп'ютери були підключені до маршрутизатору мережевим кабелем UTP. Швидкість портів маршрутизатору сягає 100 Мбіт/с, що є цілком достатнім для проведення таких досліджень. У якості Web-серверу віртуальних машин використано потужний Web-сервер Nginx. Майже всі хостинг-провайдери пропонують Nginx у ролі Web-серверу.

Нижче наведені основні характеристики комп'ютера-сервера:

- центральний процесор: Intel Core i5 655k (4,5GHz, 2cores, 4threads);
- оперативна пам'ять: 8Gb;

- накопичувач: жорсткий диск 2TB (7200 RPM);
- мережева карта: D-Link DFE-520TX;
- основна операційна система Microsoft Windows 10;
- віртуальна машина: Virtual Box 7.

Нижче наведені основні характеристики комп'ютера-клієнта:

- центральний процесор: Intel Core i3 4200m (2,5GHz, 2cores, 4threads);
- оперативна пам'ять: 8Gb;
- накопичувач: SSD-диск 240 GB.

Нижче наведені основні характеристики екземплярів віртуальних машин на комп'ютері-сервері:

Комп'ютер 1:

- операційна система: Red Hat Enterprise Linux;
- об'єм пам'яті: 512 МБ;
- завантаження процесорних ядр: 1 ядро на 50%;
- програмне забезпечення: Nginx із стандартною конфігурацією;

Комп'ютер 2:

- операційна система: Red Hat Enterprise Linux;
- об'єм пам'яті: 2048 МБ;
- завантаження процесорних ядр: 2 ядра на 100%;
- програмне забезпечення: Nginx із стандартною конфігурацією.

Комп'ютер балансувальника навантаження:

- операційна система: Red Hat Enterprise Linux;
- об'єм пам'яті: 512 МБ;
- завантаження процесорних ядр: 1 ядро на 50%;
- програмне забезпечення: змінюється відповідно до етапу тестування.

Нижче наведені основні характеристики екземплярів віртуальних машин на комп'ютері-клієнті:

Комп'ютер для створення навантаження:

- операційна система: Red Hat Enterprise Linux;

– об’єм пам’яті: 5 ГБ;
– завантаження процесорних ядр: 4 ядра на 90%;
– програмне забезпечення: змінюється відповідно до етапу тестування.

Нижче наведені основні характеристики мережевого обладнання:

WiFi-маршрутизатор NETIS WF2411I:

– швидкість портів: 100 Мбіт/с;
– кількість LAN-портів: 4 од.;
– кількість WAN-портів: 1 од.;
– чипсет: Realtek RTL8196C.

Кожна віртуальна машина підключена через мережевий міст адаптера фізичного комп’ютера, тобто кожна віртуальна машина є видимою у локальній мережі, створеній маршрутизатором. На рис. 1.8, 1.9 проілюстровано етапи налаштування маршрутизатору через Web-інтерфейс.

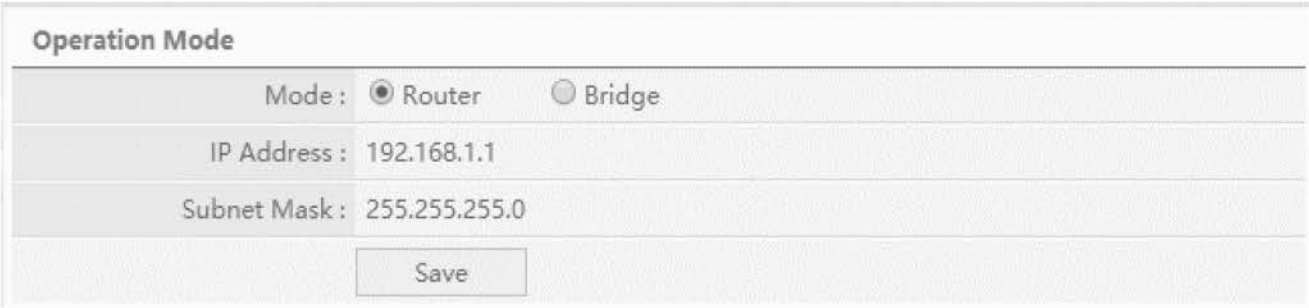


Рисунок 1.8. Вибір режиму функціонування маршрутизатору

Кожна віртуальна машина споживає ресурси фізичного комп’ютера-сервера. Було створено зазначену кількість віртуальних машин з зазначеними вище характеристиками. Їх сумарна обчислювальна потужність укладається у потужність фізичного комп’ютера-сервера та не виходить за її межі. При завантаженні всіх віртуальних машин на 100% не буде перевантаження фізичного комп’ютера-сервера. Під час проведення всіх досліджень, для чистоти експерименту, на фізичному комп’ютері-сервері та на кожній віртуальній машині не було завантажено жодного стороннього програмного додатку [8].

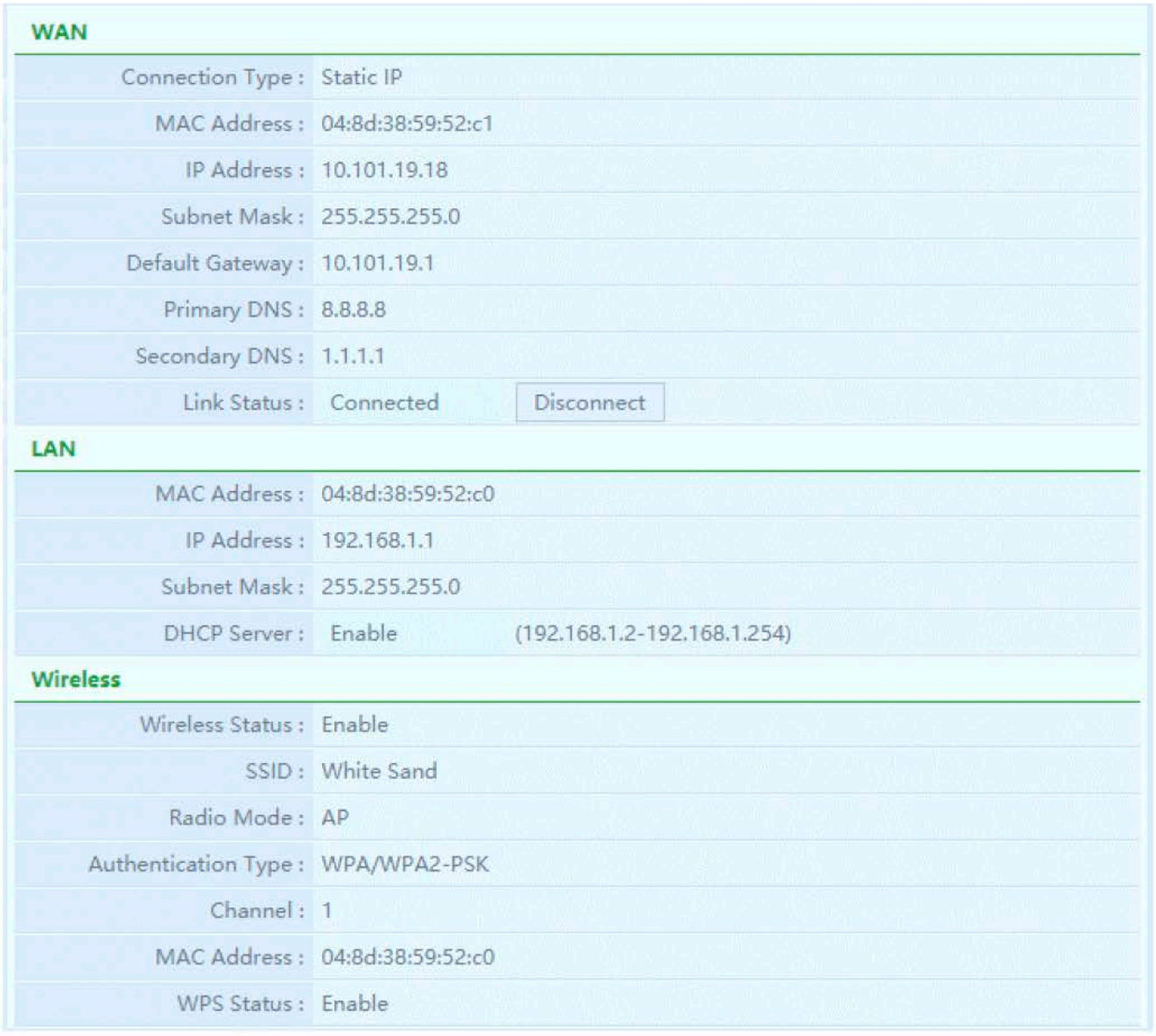


Рисунок 1.9. Налаштування глобальної та локальної мережі на маршрутизаторі

1.4 Тестування роботи алгоритмів розподілу потоків та балансувальників навантаження

Після вивчення декількох архітектурних рішень різних моделей розподілу потоків та балансувальників навантаження було розроблено модель, показану на рис. 1.10. Показана архітектура складається з таких компонентів:

- DNS-сервер, диспетчер-селектор;
- N диспетчерів (відповідають N кластерам, тобто по одному для кожного
- кластера);
- колектор навантаження і монітор сповіщень для кожного диспетчера;
- контролер навантаження і лічильник запитів для кожного сервера.

DNS-сервер спочатку спілкується з клієнтом, отримує запит і перетворює необхідний URL у IP-адресу. Оскільки кожен кластер має одного диспетчера, диспетчер-селектор напряму підключений до кожного. Кожен диспетчер безпосередньо пов'язаний з усіма web-серверами у своєму кластері. Кожен диспетчер складається з одного колектору навантаження, який збирає інформацію про навантаження кожного сервера. Кожен диспетчер також складається з одного монітора сповіщень, який перевіряє навантаження на кожен сервер і тимчасово припиняє сервіс web-сервера, якщо навантаження стає зависоким. Кожен сервер складається з контролера навантаження і лічильника запитів, які розраховують і передають інформацію про навантаження на сервері [9].

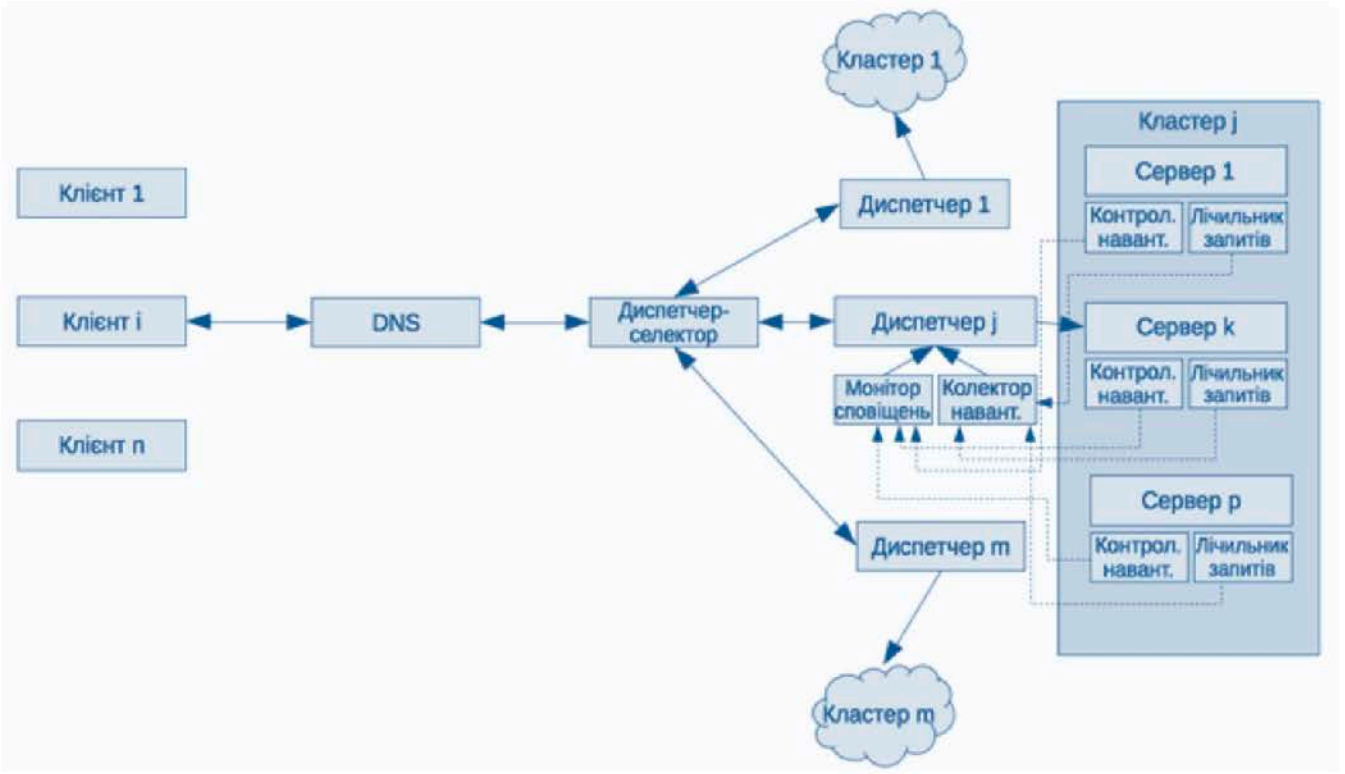


Рисунок 1.10. Архітектурне рішення алгоритму балансування навантаження

У цьому підрозділі описано результати проведеного дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів за допомогою різних інструментів та конфігурацій. Метою є збереження часу відповіді сервера при зростанні кількості конкурентних користувачів web-ресурсу або зменшення часу відповіді для тієї ж кількості користувачів. Тому для кожного алгоритму (відповідно програмному забезпеченню) проведено тести та зібрано статистичний матеріал, який буде подано далі у вигляді графіків. Перший графік буде

відображати, як поводить себе одна віртуальна машина без розподілу потоків, запущений Nginx Web-сервер на віртуальній машині №1. Наступні графіки будуть відображати поведінку системи за різними алгоритмами розподілу потоків, коли задіяні всі віртуальні машини. У результаті треба визначити, який алгоритм та при яких налаштуваннях надає найменший час відповіді при більшому числі користувачів. Також проведено тест відмовостійкості системи.

Під час тестування роботи алгоритмів розподілу потоків та балансувальників навантаження будуть використані такі конфігурації обладнання:

– комп'ютер 1 працює, комп'ютер 2 працює;
– комп'ютер 1 вимкнений, комп'ютер 2 працює;
– комп'ютер 1 працює, комп'ютер 2 вимкнений;
– комп'ютер 1 вимкнений, комп'ютер 2 вимкнений.

В усіх зазначених конфігураціях комп'ютер балансувальника не вимикається, адже при його відмові переривається робота всієї системи та дослідження не мають сенсу.

Визначення продуктивності системи Web-серверів виконується за показником «запити за секунду». Для статичного змісту вихідні дані програми gnuplot будуть демонструвати відношення кількості запитів до часу відповіді сервера, що буде показано на графіках. Головним показником порівняння є стовпчик «Average Req/s», тобто середнє значення часу відповіді серверу. Кожен сервер налаштований віддавати однакову HTML-сторінку. Прикладом такого сайту є цільова сторінка, яка заздалегідь сформована та завантажена на сервери, і призначена для відображення інформації без динамічного змісту. Для динамічного змісту, у програмі locust, вихідними даними є таблиця після завершення тесту [10].

1.4.1 Налаштування програмних засобів для тестування

Тестування роботи алгоритмів розподілу потоків та балансувальників навантаження буде проводитися для двох випадків, а саме статичного та динамічного змісту. Прикладом статичного змісту є проста HTML-сторінка, розміщена на всіх серверах системи, а динамічного – система керування змістом

CMS Wordpress.

Для дослідження навантаження випадку статичного змісту використано простий, але потужний інструмент для тестування Apache Benchmark, який є частиною серверу Apache. Не менш важливим критерієм вибору програмних засобів для тестування є можливість збирати статистичні дані у вигляді графіків. Apache Benchmark дозволяє створювати файл з розширенням «tsv», що містять статистичні дані (рис.1.11). Для початку навантажувального тестування потрібно сформувати команду запуску тестування. Прикладом команди є рядок:

ab –n 2000 –c 100 –g apache-1.tsv HTTP://192.168.1.10

де 2000 – кількість підключень, 100 – кількість конкурентних запитів, apache-1.tsv – файл, з якого формується графік, 192.168.1.10 – адреса тестованого сервера.

```
λ leebrandt [~] → ab -n 100 -c 10 https://google.com/
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking google.com (be patient).....done


Server Software:        gws
Server Hostname:        google.com
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
TLS Server Name:        google.com

Document Path:          /
Document Length:        220 bytes

Concurrency Level:      10
Time taken for tests:   2.652 seconds
Complete requests:      100
Failed requests:        0
Non-2xx responses:      100
Total transferred:      65600 bytes
HTML transferred:       22000 bytes
Requests per second:    37.71 [#/sec] (mean)
Time per request:       265.203 [ms] (mean)
Time per request:       26.520 [ms] (mean, across all concurrent requests)
Transfer rate:          24.16 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       71  155 150.8     93     946
Processing:    44   72  63.8     48     321
Waiting:       43   71  63.2     48     321
Total:        121  228 155.9    148     993

Percentage of the requests served within a certain time (ms)
  50%    148
  66%    225
  75%    346
  80%    354
  90%    386
  95%    407
  98%    992
  99%    993
 100%    993 (longest request)
```

Рисунок 1.11. Виведення результатів тестування у Apache Benchmark

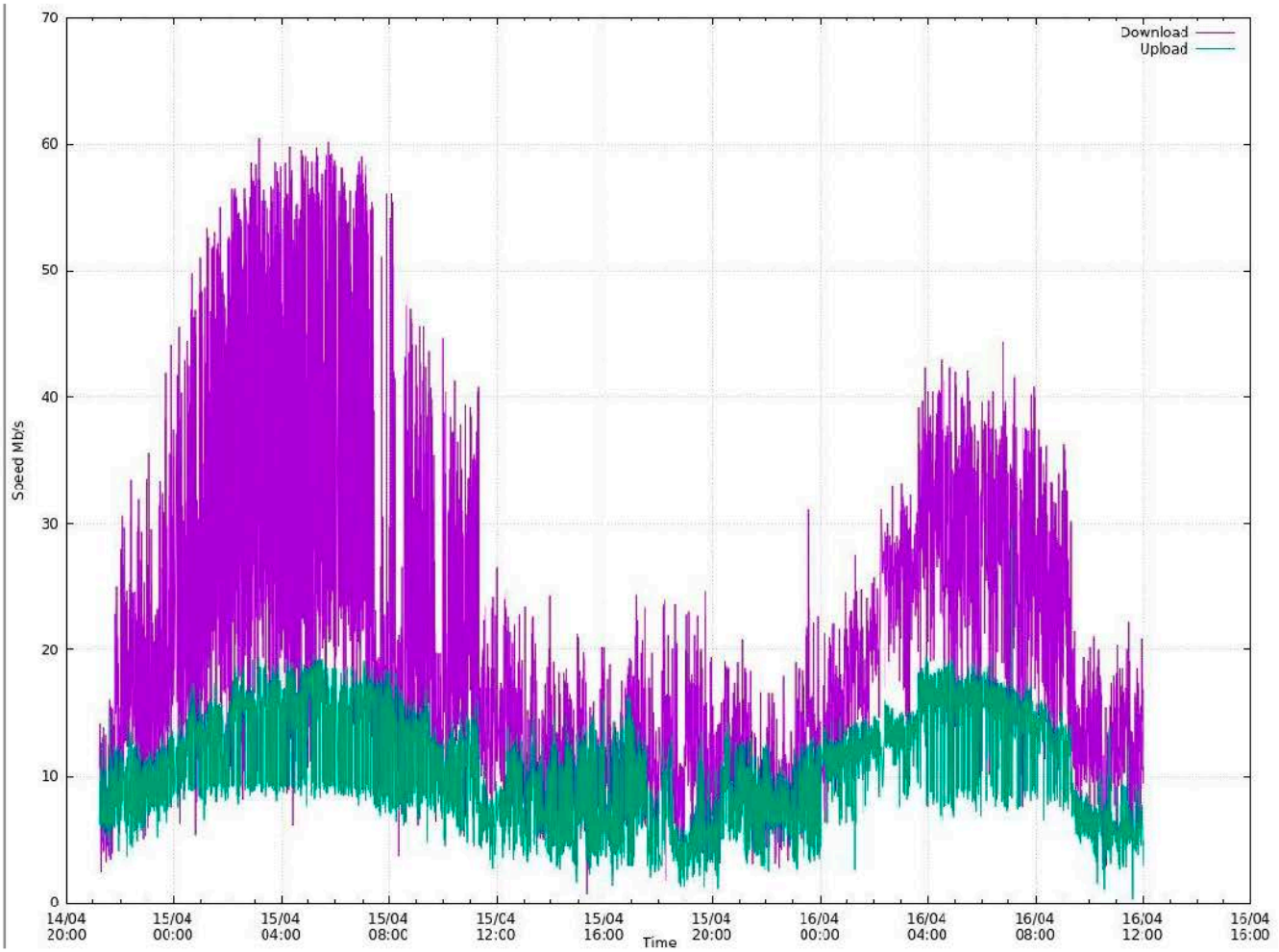


Рисунок 1.12. Виведення графіків у Gnuplot

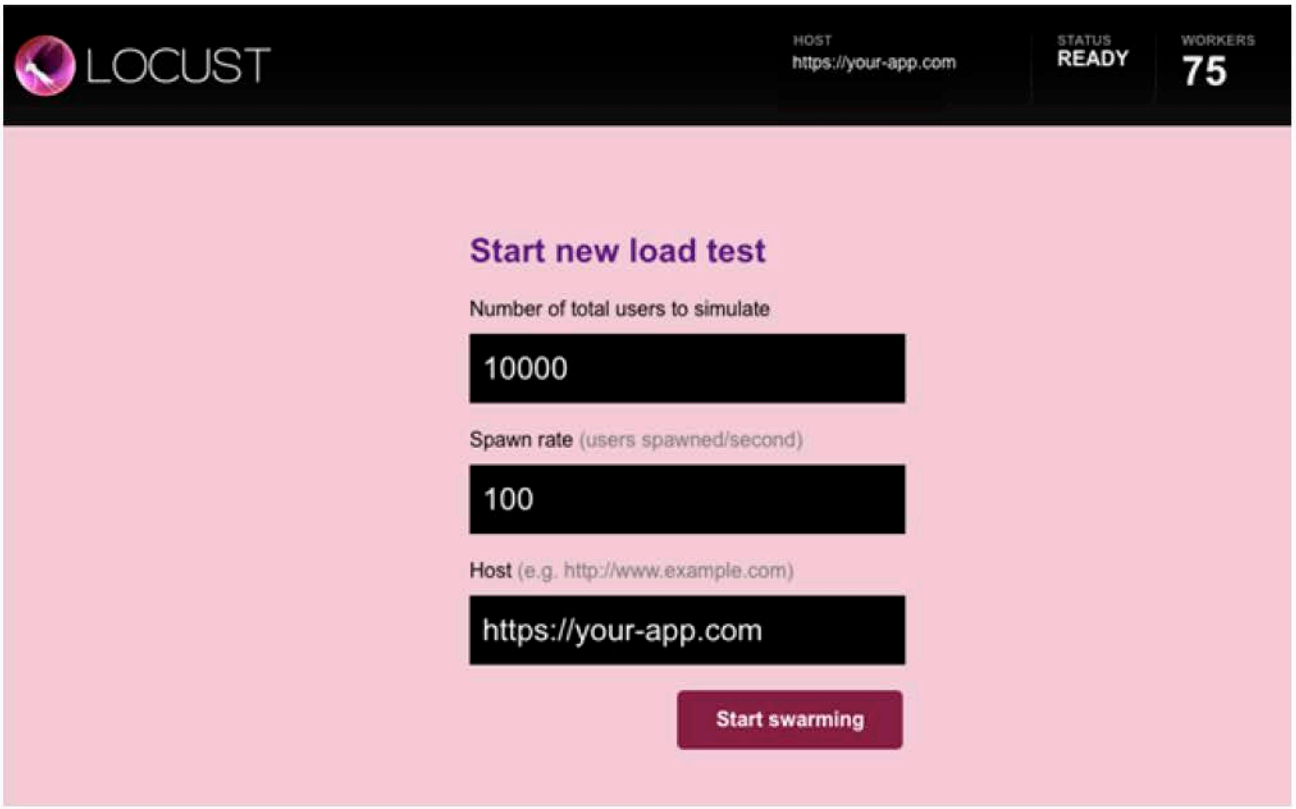


Рисунок 1.13. Web-інтерфейс програми для навантажувального тестування Locust

Іншим інструментом для отримання кінцевих графіків обрано програмний засіб Gnuplot. Gnuplot (рис.1.12) є безкоштовною програмою для створення 2D- та 3D-графіків і має свою систему програмування вихідних даних. Вихідні зображення будуть виводитися і зберігатися у форматі JPEG. Вхідними даними для програми Gnuplot є файл apache-1.tsv та файл apache-benchmark.p. Перший файл являє собою вихідні дані програми Apache Benchmark. Для створення графіку у даному проекті реалізовано програму-сценарій, лістинг якого наведено у додатку А. У ньому зазначені директорії та назви вхідного і вихідного файлів. Для запуску перетворення сирих даних з файлу apache-1.tsv до вигляду графіку треба запустити консольну команду gnuplot apache-benchmark.p, де apache-benchmark.p є заздалегідь сформованим файлом [11].

Для дослідження навантаження випадку динамічного змісту використано програму для тестування навантаження Locust. Ця програма дозволяє комплексно дослідити поведінку ресурсу у різних випадках, зокрема, імітувати різні дії користувачів, такі як перегляд особистого кабінету, авторизація на сайті, перегляд блогу та інше. Окрім того, програма Locust має зручний і простий інтерфейс (рис.1.13). На рис. 1.14 показані налаштування Locust для тестування.

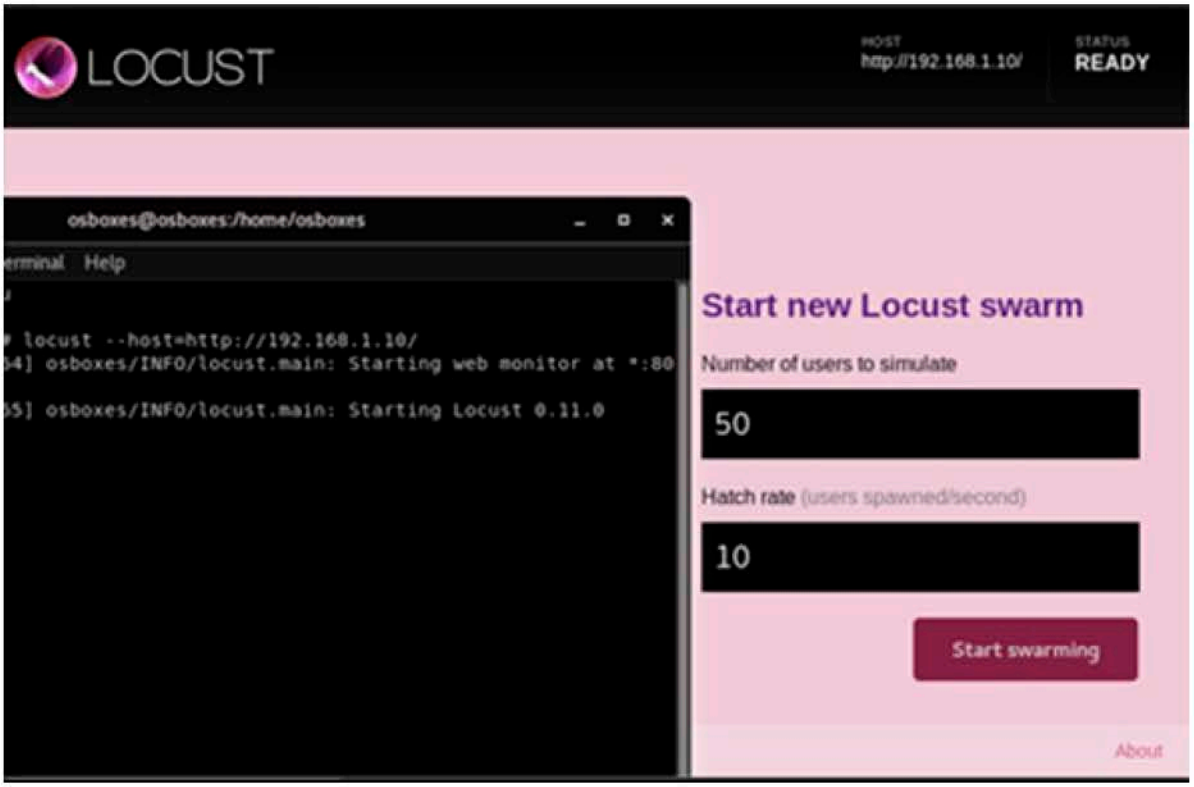


Рисунок 1.14. Налаштування програми Locust для тестування

```
▼<form name="loginform" id="loginform" action="/wp-login.php" method=
"post"> == $0
  ▼<p>
    ▼<label for="user_login">
       "Имя пользователя или e-mail"
       <br>
       <input type="text" name="log" id="user_login" class="input" value
       size="20">
     </label>
   </p>
  ▼<p>
    ▼<label for="user_pass">
       "Пароль"
       <br>
       <input type="password" name="pwd" id="user_pass" class="input" value
       size="20">
     </label>
```

Рисунок 1.15. Форма авторизації Wordpress у вигляді HTML-коду

Сервіс Locust виконує тестування згідно зі сценарієм. Щоб запустити Locust, треба виконати консольну команду locust --host= HTTP://192.168.1.10/. У такому випадку Locust буде шукати файл із сценарієм у директорії, в якій запущена команда. За замовчуванням він шукає файл locustfile.py, тому буде створено цей файл, який описує, які саме адреси будуть відвідувати створені користувачі та які дії будуть виконувати.

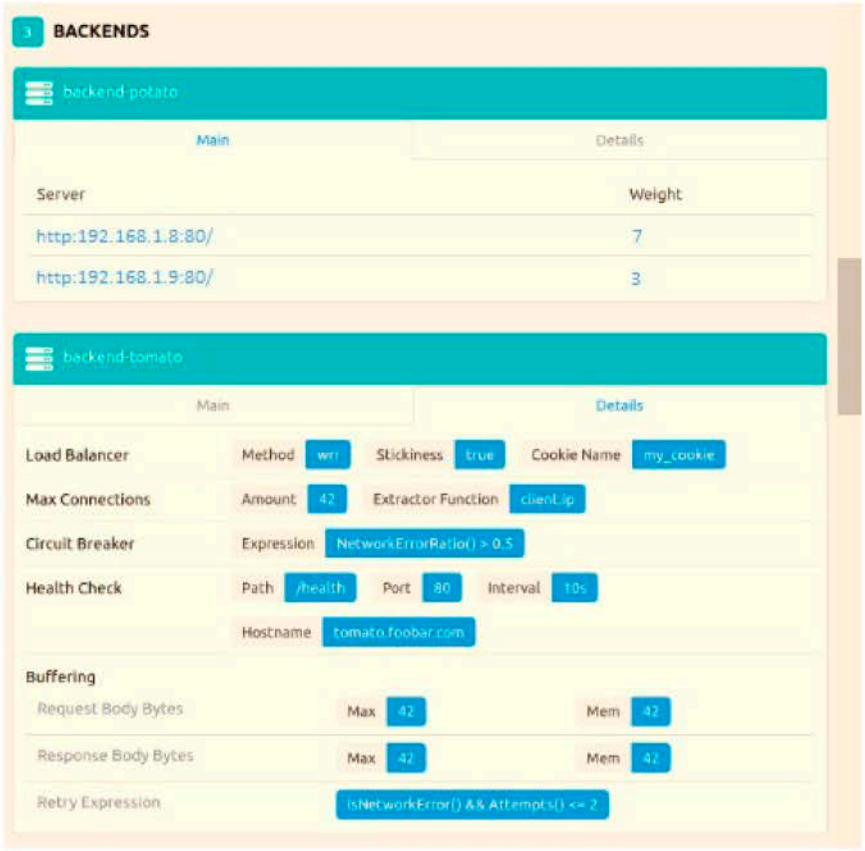


Рисунок 1.16. Web-інтерфейс балансувальника навантаження Traefik Load Balancer

Код цього файлу розміщено у додатку А. Авторизація є post-запитом до сторінки wp-login.php з параметрами user_login та user_pass (рис.1.15). У файл locustfile.py будуть записані відповідні рядки коду, щоб імітувати цей post-запит. Перегляд головної сторінки та сторінки запису блогу реалізується get-запитами.

Сервіс Traefik Load Balancer є також балансувальником навантаження, що має зручний Web-інтерфейс. Backend-сервери включені були включені до конфігурації при налаштуванні (рис.1.16).

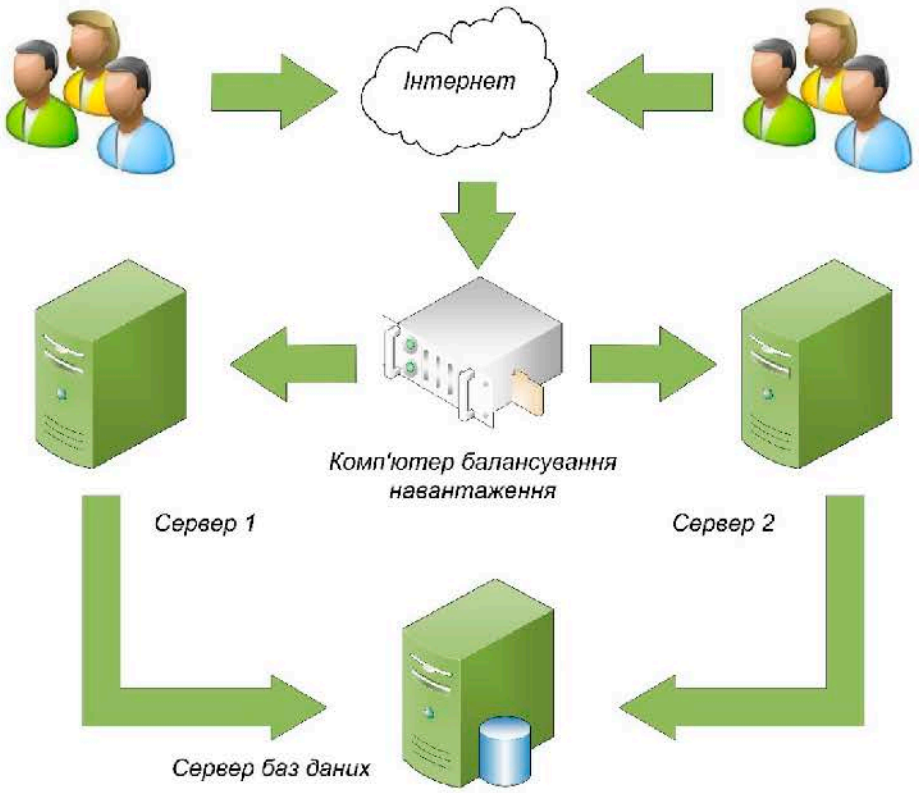


Рисунок 1.17. Архітектура балансування навантаження для CMS Wordpress

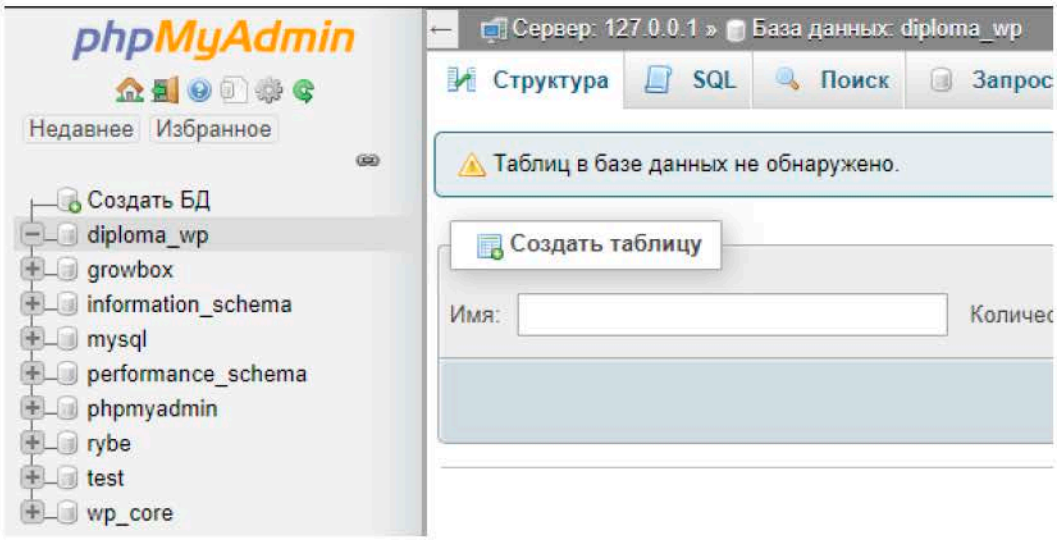


Рисунок 1.18. Створення бази даних у модулі phpMyAdmin

Організація роботи CMS Wordpress пов'язана з архітектурою, зображеною на рис. 1.17. Для того, щоб розмістити екземпляр спільної бази даних, виділено ще одну Віртуальну машину 3 з аналогічною Віртуальній машині 2 обчислювальній потужністю. Віртуальна машина 3 має IP-адресу 192.168.1.11. На віртуальних машинах 1 і 2 розміщені екземпляри CMS Wordpress, а на віртуальній машині 3 розміщено спільну базу даних diploma_wp. Початок створення бази даних у модулі phpMyAdmin показаний на рис.1.18. Щоб система керування змістом Wordpress мала змогу працювати за даних умов, кожен екземпляр для роботи з однією базою даних налаштовано, як показано на рис. 1.19.

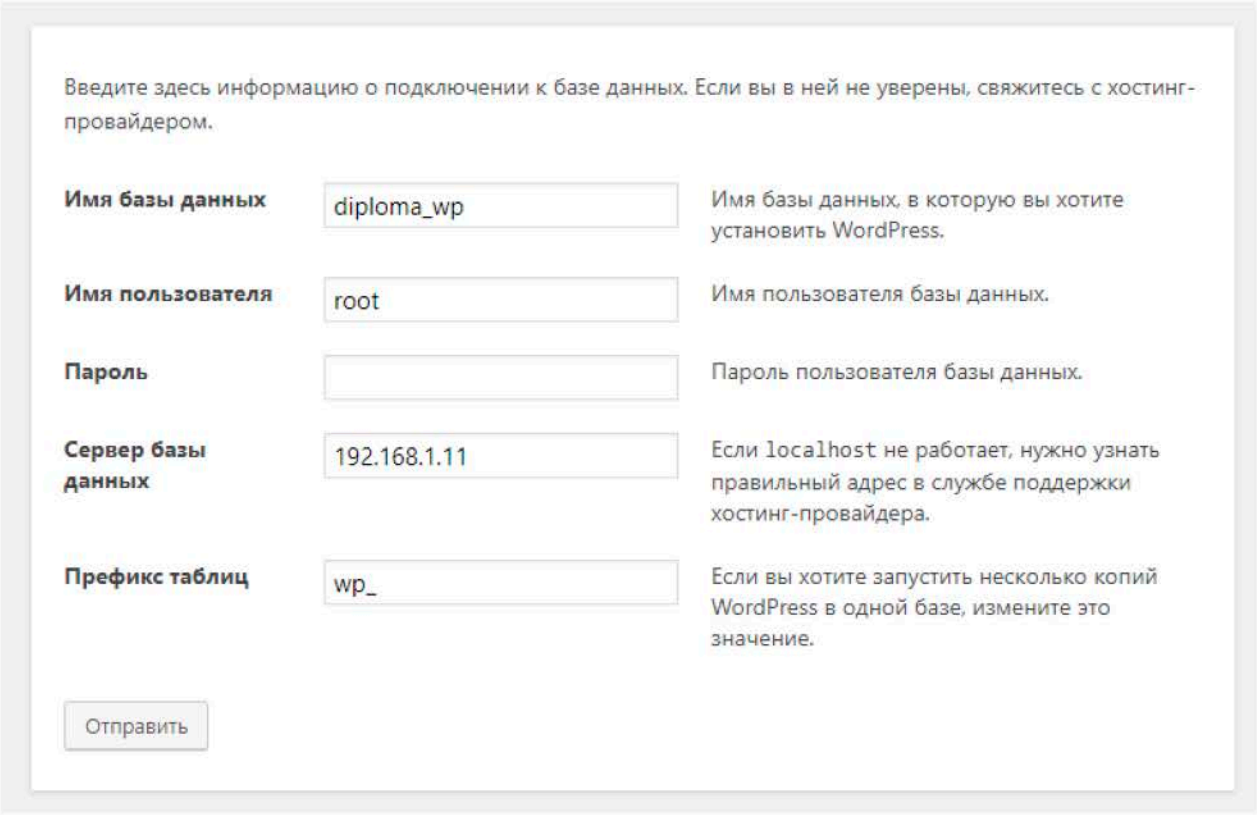


Рисунок 1.19. Налаштування Wordpress на Віртуальній машині 1

1.4.2 Дослідження приросту продуктивності на статичному змісті

Для оцінки приросту продуктивності системи при використанні різних алгоритмів розподілу потоків та різних конфігурацій обладнання (конфігурація 1 ÷ конфігурація 4), необхідно спочатку визначити продуктивність системи в початковій конфігурації на статичному змісті з запущеним Web-сервером Nginx на віртуальній машині №1 (рис.1.20).

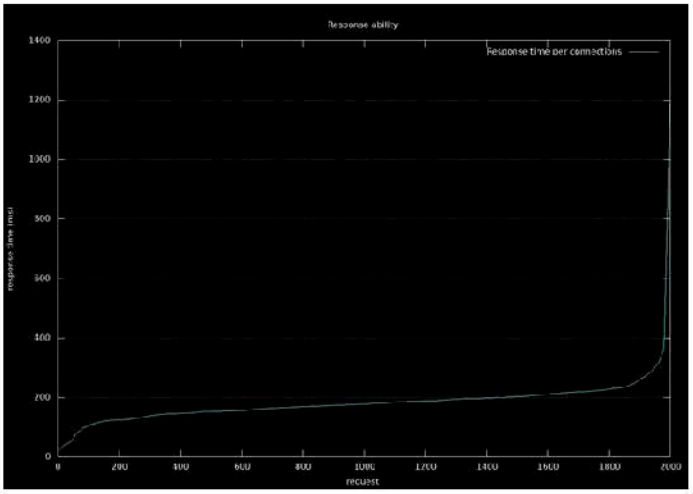


Рисунок 1.20. Графік продуктивності без балансувальника навантаження для початкової конфігурації на статичному змісті з запущеним Web-сервером Nginx на віртуальній машині №1

Графіки продуктивності балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin та Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведені на рис.1.21. Алгоритм Weighted Round Robin показав найкращий результат на 2000 підключеннях. Подальше випробування у конфігураціях, де одна з машин відмовляє, буде проведено саме для цього алгоритму – це дасть можливість визначити відмовостійкість балансувальника навантаження HAProxy та алгоритму Weighted Round Robin.

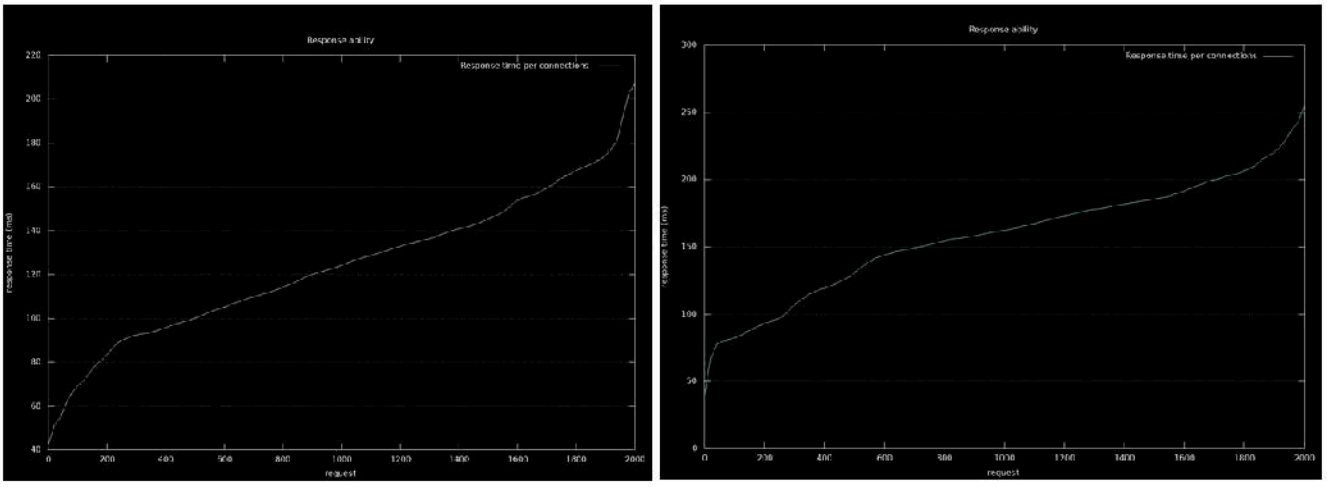


Рисунок 1.21. Графіки продуктивності балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Weighted Least Connections (справа) для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”

Графіки продуктивності балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin для конфігурацій “комп’ютер 1 вимкнений, комп’ютер 2 працює” та “комп’ютер 1 працює, комп’ютер 2 вимкнений” наведені на рис.1.22. Система конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” є працездатною. Усі запити обробляв комп’ютер 2, який залишився увімкненим. Для визначення продуктивності системи у випадку відмови її складової частини, тобто віртуальної машини 1, виконувався тест у Apache Benchmark. Як видно за графіком, система обробляла запити не так швидко, як при функціонуванні усіх її компонентів. Система конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” залишилась працездатною і у випадку з вимкненим комп’ютером 2 (сервером).

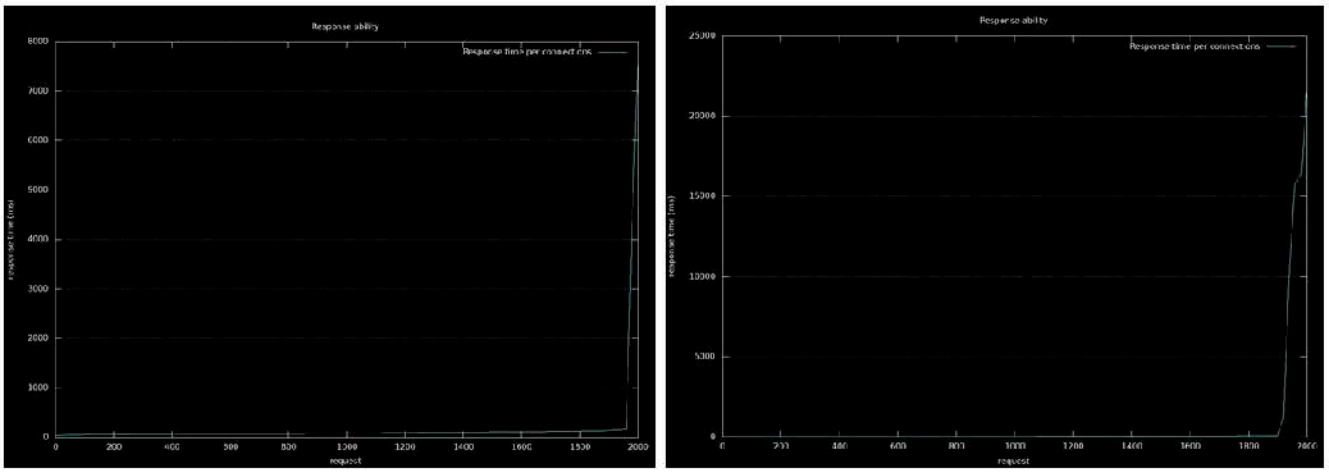


Рисунок 1.22. Графіки продуктивності балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” (зліва) та “комп’ютер 1 працює, комп’ютер 2 вимкнений” (справа)

При тестуванні конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 вимкнений” було отримано помилку 503 (сервіс недоступний) при спробі доступу до ресурсу. Ця сторінка повідомляє про те, що немає доступних серверів для обробки запиту. У конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 вимкнений” усі сервери вимкнені, але у конфігурації балансувальника навантаження є можливість встановити адресу обробника у випадку, якщо жоден сервер зі списку недоступний. Також на комп’ютері балансувальника можна розмістити Web-сервер, який буде тільки віддавати сторінку, яка повідомляє про

помилку на ресурсі і пропонує почекати. Тому, можна зробити висновок про те, що користувач майже ніколи не побачить помилку 404 (сервер не знайдений), що говорить про високу надійність системи.

Графік продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведений на рис.1.23.

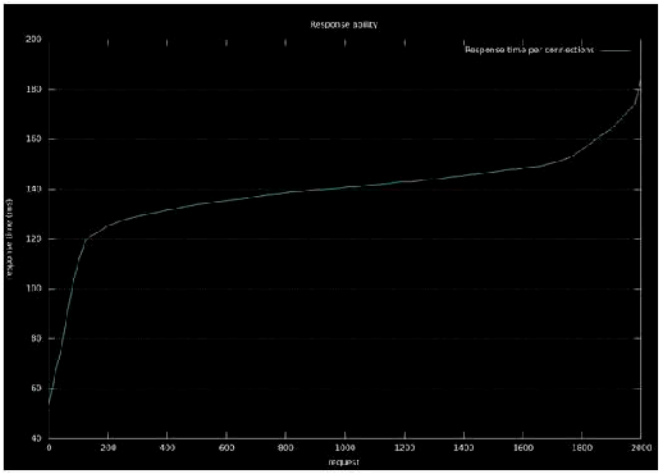


Рисунок 1.23. Графік продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”

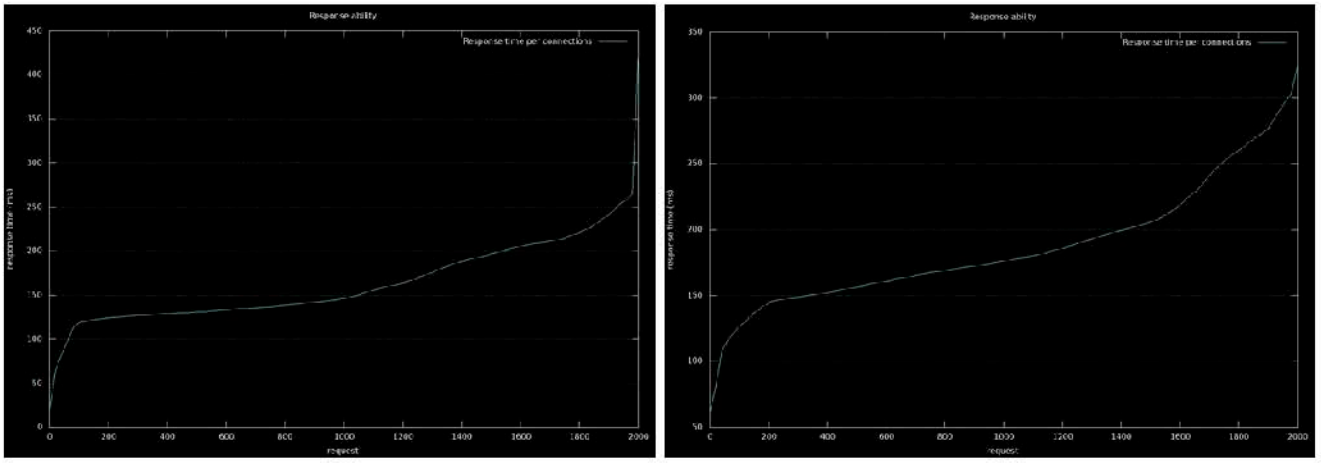


Рисунок 1.24. Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Weighted Least Connections (справа) для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin та Weighted Least Connections для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” наведені на рис.1.24.

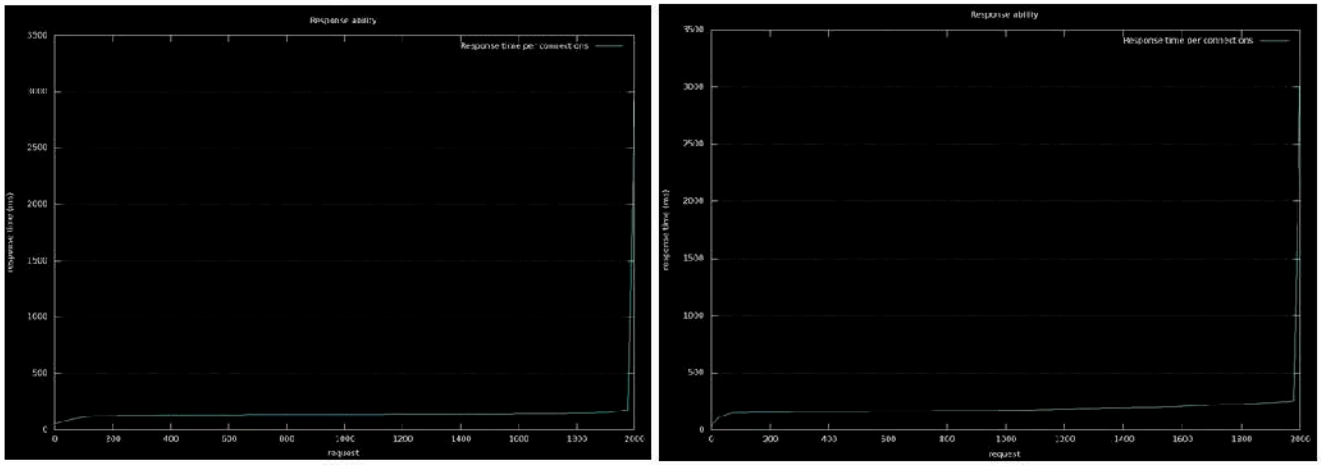


Рисунок 1.25. Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Weighted Least Connections (справа) для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin та Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” наведені на рис.1.25. При тестуванні конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 вимкнений” у разі відмови всіх backend-серверів було отримано помилку 502 (помилка шлюзу) при спробі доступу до ресурсу. При цьому система відповіла помилкою, а не вимкнулась повністю. Балансувальник навантаження Nginx, як і HAProxy, має обробник ситуації, коли всі сервери, окрім серверу-балансувальника, вимкнені. Комп’ютер балансувальника навантаження також можна налаштувати на Web-сервер.

Графіки продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурацій “комп’ютер 1 працює, комп’ютер 2 працює” та “комп’ютер 1 вимкнений, комп’ютер 2 працює” наведені на рис.1.26. Графік продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” наведений на рис.1.27. Графіки продуктивності балансувальника навантаження Traefik Load Balancer на статичному змісті за алгоритмами Weighted Round Robin та Dynamic Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведені на рис.1.28.

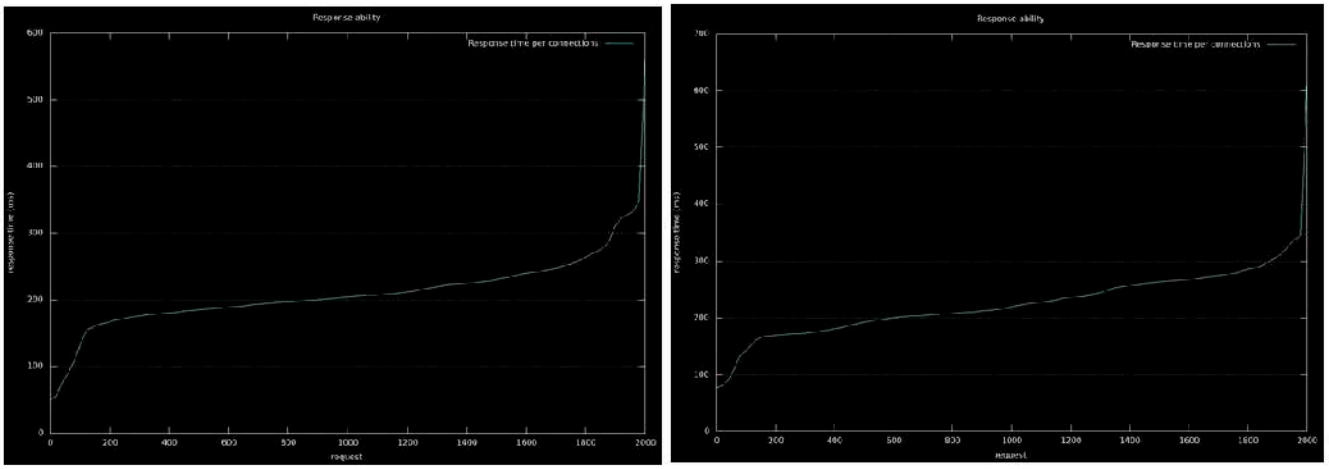


Рисунок 1.26. Графіки продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” (зліва) та “комп’ютер 1 вимкнений, комп’ютер 2 працює” (справа)

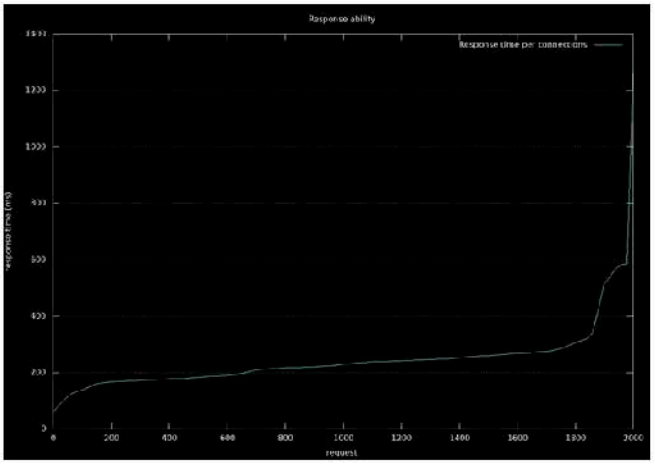


Рисунок 1.27. Графік продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

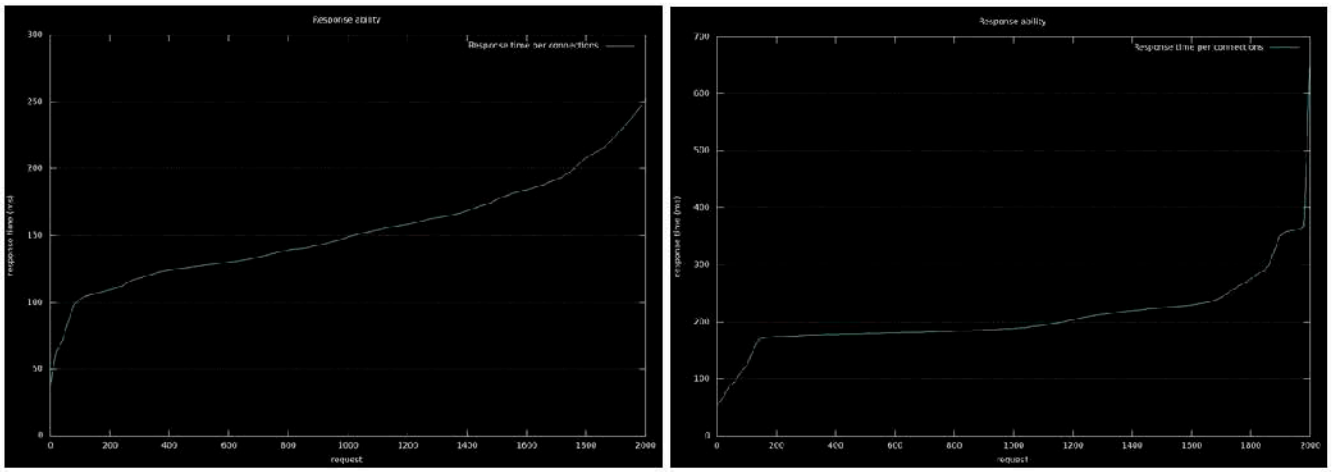


Рисунок 1.28. Графіки продуктивності балансувальника навантаження Traefik на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Dynamic Round Robin (справа) для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”

Графік продуктивності балансувальника навантаження Traefik Load Balancer на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” наведений на рис.1.29.

Графіки продуктивності балансувальника навантаження Traefik Load Balancer на статичному змісті за алгоритмами Weighted Round Robin та Dynamic Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” наведені на рис.1.30.

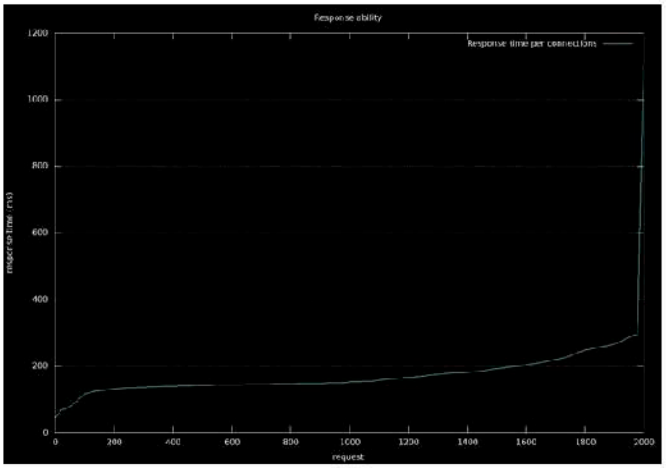


Рисунок 1.29. Графік продуктивності балансувальника навантаження Traefik Load Balancer на статичному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

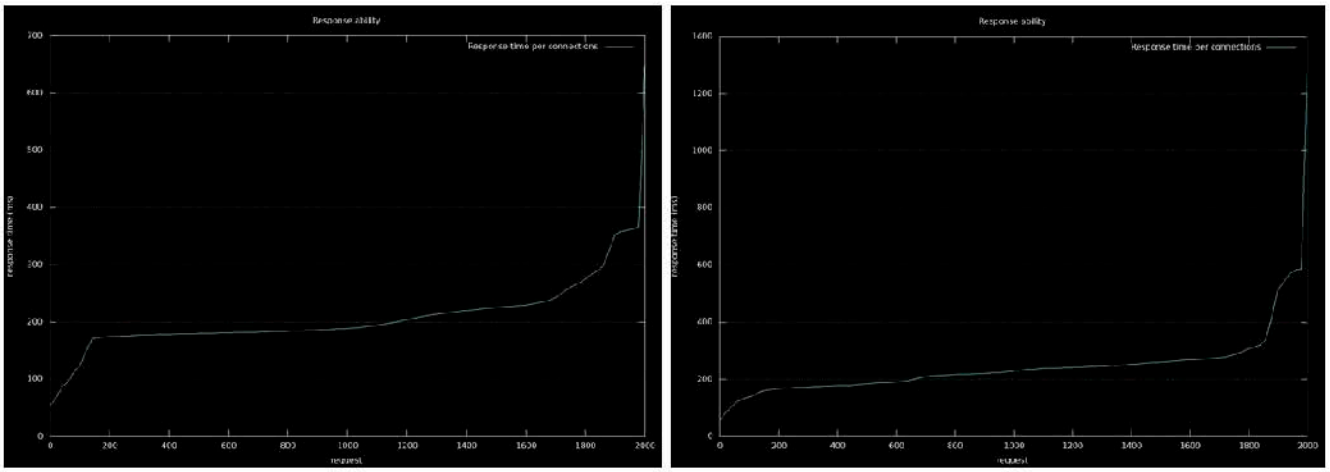


Рисунок 1.30. Графіки продуктивності балансувальника навантаження Traefik на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Dynamic Round Robin (справа) для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

1.4.3 Дослідження приросту продуктивності на динамічному змісті

Для оцінки приросту продуктивності системи при використанні різних алгоритмів розподілу потоків та різних конфігурацій обладнання (конфігурація 1

÷ конфігурація 4), необхідно спочатку визначити продуктивність системи в початковій конфігурації на динамічному змісті для Wordpress (рис.1.31).

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	//	34	0	2000	4689	626.2259483337402	20907.031059265137	51996	0.2
GET	//?p=1	11	0	1600	1559	794.3689823150635	2591.9852256774902	56520	0.2
POST	//wp-login.php	5	0	14000	13480	12625.993967056274	14144.117832183838	2347	0
	Total	**50**	**0**	**2000**	**4880**	**626.2259483337402**	**20907.031059265137**	**48026**	**0.4**

Рисунок 1.31. Результати тестування продуктивності без балансувальника навантаження на динамічному змісті у початковій конфігурації для Wordpress

Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмами Weighted Round Robin та Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведені на рис.1.32 та рис.1.33; для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” – на рис.1.34 та рис.1.35; для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” – на рис.1.36 та рис.1.37.

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	//	28	1	830	1005	380.0849914550781	2343.1370258331 3	50162	0.4
GET	//?p=1	20	1	890	1144	373.2089996337 8906	3127.8300285339355	53722	0.1
POST	//wp-login.php	5	0	460	500	184.6339702606 2012	947.2730159759 521	2351	0
	Total	**53**	**2**	**840**	**1010**	**184.6339702606 2012**	**3127.8300285339355**	**46994**	**0.5**

Рисунок 1.32. Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	30	0	640	929	401.0269641876 2207	2445.942163467 407	52010	0.5
GET	/?p=1	16	0	540	881	413.0570888519 287	1886.216878890 9912	56552	0.3
POST	/wp-login.php	5	0	210	338	184.1509342193 6035	892.5020694732 666	2351	0
	Total	**51**	**0**	**570**	**856**	**184.1509342193 6035**	**2445.942163467 407**	**48566**	**0.8**

Рисунок 1.33. Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	32	0	940	2260	353.7411689758 301	12090.08312225 3418	52024	0.3
GET	/?p=1	14	1	610	1164	368.0179119110 1074	3315.400123596 1914	52520	0.1
POST	/wp-login.php	5	0	470	675	167.1259403228 7598	1413.018941879 2725	2352	0
	Total	**51**	**1**	**910**	**1804**	**167.1259403228 7598**	**12090.08312225 3418**	**47290**	**0.4**

Рисунок 1.34. Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	29	0	370	1992	349.6458530426 0254	9643.318891525 269	52024	0.2
GET	/?p=1	17	0	400	2205	358.6168289184 5703	17218.45412254 3335	56560	0.4
POST	/wp-login.php	5	0	170	3572	167.8099632263 1836	17161.28993034 3628	2352	0
	Total	**51**	**0**	**380**	**2218**	**167.8099632263 1836**	**17218.45412254 3335**	**48666**	**0.6**

Рисунок 1.35. Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	27	4	7000	7738	835.6750011444092	16521.929025650024	44292	0
GET	/?p=1	13	1	6000	7550	633.5608959197998	15644.397020339966	52172	0.1
POST	/wp-login.php	5	0	8300	8216	6742.2850131988525	9524.54686164856	2347	0
	Total	**45**	**5**	**7100**	**7737**	**633.5608959197998**	**16521.929025650024**	**41908**	**0.1**

Рисунок 1.36. Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

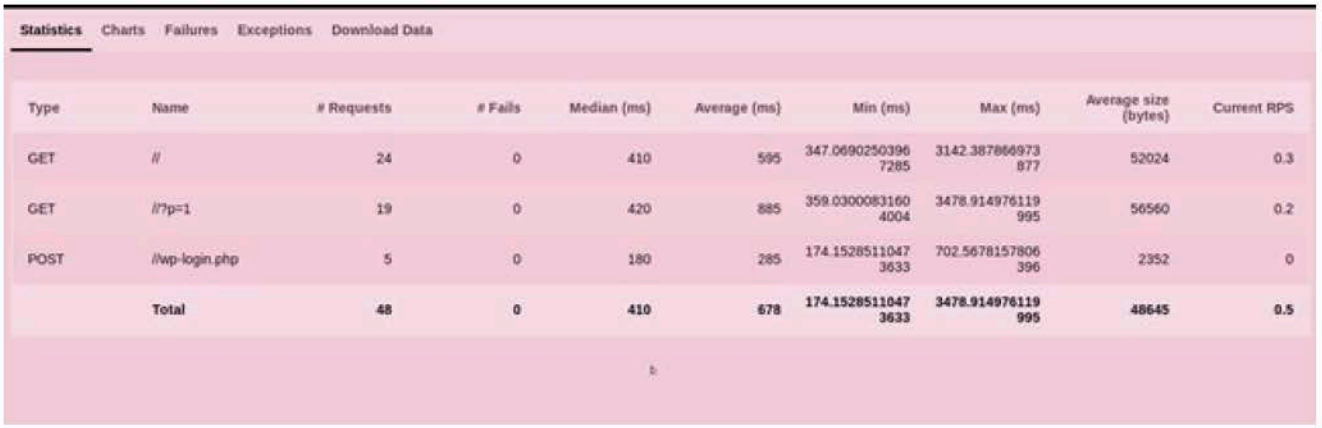
Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	30	1	1100	5332	570.0039863586426	19186.876058577849	50262	0.2
GET	/?p=1	13	1	690	3697	586.9147777557373	17123.921871185303	52172	0.1
POST	/wp-login.php	5	0	14000	8871	370.21398544311523	15561.010122299194	2347	0
	Total	**48**	**2**	**1100**	**5258**	**370.21398544311523**	**19186.876058577849**	**45788**	**0.3**

Рисунок 1.37. Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмами Weighted Round Robin та Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведені на рис.1.38 та рис.1.39; для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” – на рис.1.40 та рис.1.41; для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” – на рис.1.42 та рис.1.43.

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	31	0	1300	1531	359.6298694610 5957	4337.743043899 536	52016	0.4
GET	/?p=1	19	0	980	1264	366.3511276245 117	4096.921920776 367	56553	0.4
POST	/wp-login.php	5	0	1300	1115	170.7839965820 3125	2028.173923492 4316	2352	0
	Total	**55**	**0**	**1100**	**1401**	**170.7839965820 3125**	**4337.743043899 536**	**49069**	**0.8**

Рисунок 1.38. Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”



Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	24	0	410	595	347.0690250396 7285	3142.387866973 877	52024	0.3
GET	/?p=1	19	0	420	885	359.0300083160 4004	3478.914976119 995	56560	0.2
POST	/wp-login.php	5	0	180	285	174.1528511047 3633	702.5678157806 396	2352	0
	Total	**48**	**0**	**410**	**678**	**174.1528511047 3633**	**3478.914976119 995**	**48645**	**0.5**

Рисунок 1.39. Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	33	0	1100	2000	594.0721035003 662	6486.274957656 86	51996	0.5
GET	/?p=1	13	0	900	1514	624.3319511413 574	4061.661958694 458	56520	0
POST	/wp-login.php	5	0	1100	1093	270.3030109405 5176	1972.776889801 0254	2347	0
	Total	**51**	**0**	**1100**	**1787**	**270.3030109405 5176**	**6486.274957656 86**	**48281**	**0.5**

Рисунок 1.40. Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	26	0	410	801	355.7698726654053	2996.4728355407715	52016	0.3
GET	/?p=1	21	0	800	1054	357.0899963378906	2476.346969604492	56552	0.3
POST	/wp-login.php	5	0	190	196	178.08008193969727	226.96590423583984	2352	0
	Total	**52**	**0**	**550**	**845**	**178.08008193969727**	**2996.4728355407715**	**49072**	**0.6**

Рисунок 1.41. Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	27	0	1700	3799	411.6530418395996	18887.799978256226	52024	0.2
GET	/?p=1	17	0	1400	1797	415.5578613281225	4372.474908828735	56560	0.5
POST	/wp-login.php	5	0	210	3831	177.85191535949707	18354.665994644165	2352	0
	Total	**49**	**0**	**1400**	**3108**	**177.85191535949707**	**18887.799978256226**	**48529**	**0.7**

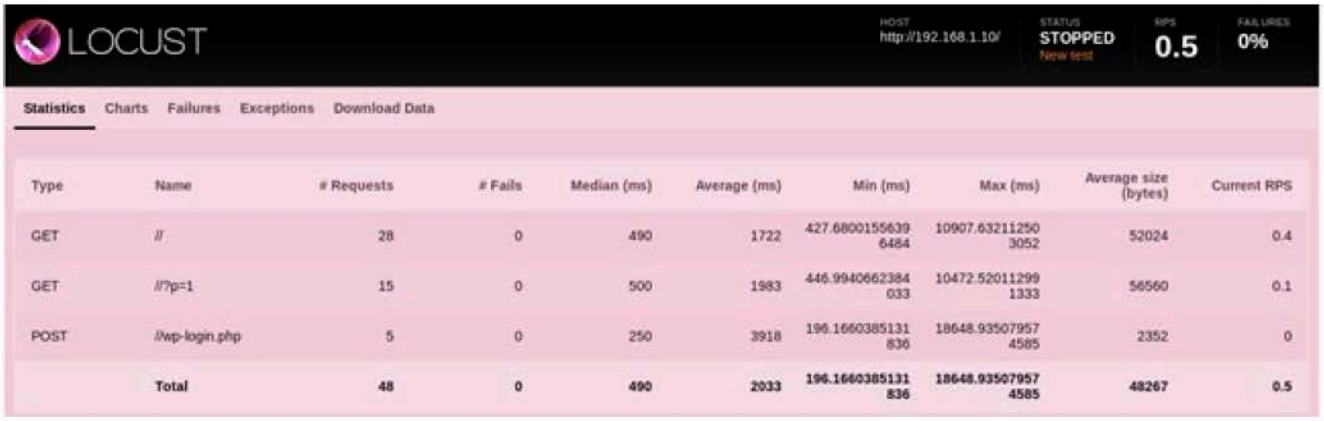
Рисунок 1.42. Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	33	0	2500	2683	587.5980854034424	6962.255001068115	51996	0.5
GET	/?p=1	13	0	2700	2994	671.623945236206	4912.293910980225	56520	0
POST	/wp-login.php	5	0	9700	9350	490.8871650695801	18338.788986206055	2347	0
	Total	**51**	**0**	**2700**	**3416**	**490.8871650695801**	**18338.788986206055**	**48281**	**0.5**

Рисунок 1.43. Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	42	0	660	741	476.794958114624	1923.2239723205566	52013	0.6
GET	/?p=1	15	0	680	820	485.4149818442041	2048.344135284424	56549	0.1
POST	/wp-login.php	5	0	2600	2976	2044.4509983062744	4691.068172454834	2351	0
	Total	**62**	**0**	**700**	**940**	**476.794958114624**	**4691.068172454834**	**49105**	**0.7**

Рисунок 1.44. Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”



Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	28	0	490	1722	427.68001556396484	10907.632112503052	52024	0.4
GET	/?p=1	15	0	500	1983	446.9940662384033	10472.520112991333	56560	0.1
POST	/wp-login.php	5	0	250	3918	196.16603851318836	18648.935079574585	2352	0
	Total	**48**	**0**	**490**	**2033**	**196.16603851318836**	**18648.935079574585**	**48267**	**0.5**

Рисунок 1.45. Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	30	2	1800	6216	693.7940120697021	34198.22192192078	48529	0.2
GET	/?p=1	14	2	3000	6918	687.3719692230225	21504.547119140625	48445	0
POST	/wp-login.php	5	0	17000	11429	524.3470668792725	18637.557029724121	2347	0
	Total	**49**	**4**	**2400**	**6949**	**524.3470668792725**	**34198.22192192078**	**43793**	**0.2**

Рисунок 1.46. Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

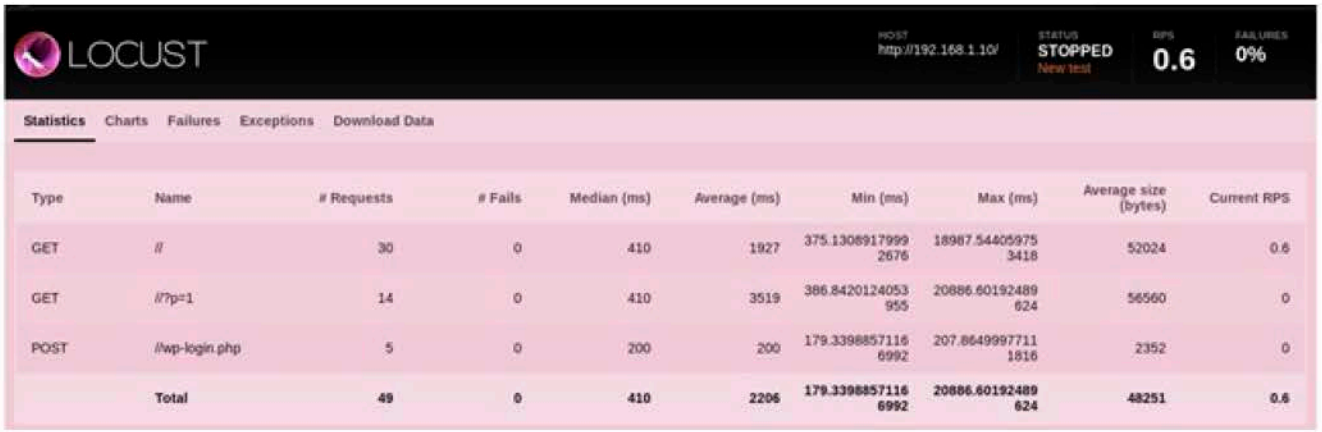
Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведені на рис.1.44;

для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” – на рис.1.45;

для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” – на рис.1.46.

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	28	0	560	852	422.5049018859863	2612.7030849456787	52013	0.3
GET	/?p=1	17	0	640	887	423.6338138580322	2445.8439350128174	56541	0.4
POST	/wp-login.php	5	0	670	1195	248.84319305419922	3634.1309547424316	2351	0
	Total	**50**	**0**	**590**	**898**	**248.84319305419922**	**3634.1309547424316**	**48586**	**0.7**

Рисунок 1.47. Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює”



Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	30	0	410	1927	375.13089179992676	18987.544059753418	52024	0.6
GET	/?p=1	14	0	410	3519	386.8420124053955	20886.601924896240	56560	0
POST	/wp-login.php	5	0	200	200	179.33988571166992	207.86499977111816	2352	0
	Total	**49**	**0**	**410**	**2206**	**179.33988571166992**	**20886.601924896240**	**48251**	**0.6**

Рисунок 1.48. Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	24	0	11000	9253	579.2930126190186	26318.600893020063	51996	0.2
GET	/?p=1	21	0	750	4621	584.1560363769531	21331.197977060604	56520	0
POST	/wp-login.php	5	0	1600	7912	966.6740894317627	18293.133974075317	2347	0
	Total	**50**	**0**	**1000**	**7174**	**579.2930126190186**	**26318.600893020063**	**48931**	**0.2**

Рисунок 1.49. Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 працює” наведені на рис.1.47; для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює” – на рис.1.48; для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений” – на рис.1.49.

Найкращий результат для статичної HTML-сторінки показаний на рис. 1.23. Цей результат забезпечений балансувальником навантаження Nginx Upstream Module із застосованим алгоритмом Weighted Round Robin. Кінцевий стовпчик графіків, де аргумент сягає відмітки 2000, дає час відповіді 184 мс. Конкурентним рішенням став балансувальник навантаження HAProxy із застосованим алгоритмом Weighted Round Robin. На рис. 1.21 максимальний час відповіді сягає 208 мс, що стало другим результатом у проведеному тестуванні.

Найкращий результат для динамічного змісту показаний на рис. 1.39, де середній час відповіді склав 678 мс, що є найменшим показником затримки серед усіх досліджених для динамічного змісту. Найкращий результат, таким чином, забезпечений балансувальником навантаження Nginx Upstream Module.

Застосовані конфігурації обладнання “комп’ютер 1 вимкнений, комп’ютер 2 працює” та “комп’ютер 1 працює, комп’ютер 2 вимкнений” дозволяють продемонструвати зміни продуктивності в залежності від часткової відмови системи. При цьому система, хоча і втрачає продуктивність, залишається працездатною.

2 ОХОРОНА ПРАЦІ

Закон України «Про охорону праці» визначає основні положення по охороні праці і регулює взаємини між працівниками і адміністрацією. В Україні законодавство по охороні праці складається із Закона України «Про охорону праці», Кодексу законів про працю і інших нормативних актів.

Широке впровадження комп'ютерної техніки, що дозволяє автоматизувати багато рутинних операцій, дістати доступ до численних джерел інформації істотно підвищує продуктивність праці користувачів відеотерміналу електронно-обчислювальної машини (ВДТ ЕОМ).

В дипломному проекті проводиться дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів. Тому даному розділі дипломного проекту розглядається питання охорони праці програміста.

2.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці даного програмного комплексу

Програміст як і користувач персонального комп'ютера випробовує значне навантаження, як фізичне (сидяче положення, навантаження на очі), так і розумове, що приводить до зниження його працездатності до кінця робочого дня.

На робочому місці під час роботи програміст піддається впливу наступних несприятливих факторів:

- недостатнє освітлення;
- шум від працюючих машин;
- електромагнітне випромінювання;
- виділення надлишків теплоти.

Тому необхідно розробити засоби захисту від цих шкідливих факторів

2.2 Гігієнічні вимоги до виробничого середовища

2.2.1 Вимоги до приміщення

Приміщення в яких планується установка та подальша робота з комп'ютером, повинні відповідати проектній документації будинку, погодженій з державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи.

Об’ємно-планувальні рішення будівель та приміщень для роботи з ВДТ мають відповідати вимогам ДСанПІН 3.3.2.007-98. Розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено. Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп’ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів. Приміщення укомплектоване системами центрального опалення Площа на одне робоче місце становить не менше 6,0 м^2, а об’єм – не менше ніж 20,0 м^3. У приміщеннях слід щоденно робити вологе прибирання. Вони повинні бути оснащені аптечками першої медичної допомоги.

2.2.2 Мікроклімат

У наслідок досліджень про взаємодію користувачів комп'ютерів з мікрокліматичними умовами на робочих місцях, а вірніше з повітряним середовищем виробничих приміщень, спостерігається низка фізичних відхилень організму, це пов'язано з наступними небезпечними факторами:

1. Збільшення концентрації: позитивних іонів, аерозолів, мікробних тіл в повітрі приміщень з ВДТ.
2. Перевищення температури повітря що спричиняє висихання слизових оболонок, їх пересихання і розтріскування;
3. Забруднень хвороботворними мікробами на комп’ютеризованих робочих місцях в теплий період року, що спричиняє до зміни терморегуляції робітників і зниження працездатності.
4. Відносна вологість повітря часто є нижче за встановлені норми.

Для забезпечення оптимальних мікрокліматичних умов в будь-який період року для приміщень в яких розташовані комп'ютеризовані робочі місця повинно бути виконано: опалювання і приміщеннях, кондиціонування повітря (найпоширеніші способи нормалізації мікроклімату);

– раціоналізація режимів праці і відпочинку (досягається скороченням тривалості робочого часу за рахунок додаткових перерв, створенням умов

для ефективного відпочинку в приміщеннях з нормальними метеорологічними умовами);

- теплоізоляція обладнання і захисних екранів (як теплоізоляційні матеріали широко використовують: азбест, азбоцемент, мінеральну вату, склотканина, керамзит, пінопласт);
- для підтримки допустимих значень мікроклімату і концентрації позитивних і негативних іонів необхідно передбачити установки або прилади зволоження та / або штучної іонізації, кондиціонування повітря.

У виробничих приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря – ГОСТ 12.1.005-88, СН 4088-86.

Таблиця 2.1

Параметри мікроклімату	**Значення параметру**	**Значення параметру**
	Взимку	влітку
Температура, C^0	22-24	23-25
Відносна вологість, %	40-60	40-60
Швидкість руху повітря, м/с	0,1	0,1-0,2

2.2.3 Освітлення

Освітлення приміщення має природне та штучне походження. Природне освітлення подається через віконні прорізі, бокове. Для штучного освітлення у приміщенні використовуються люмінесцентні лампи, які в порівнянні з лампами розжарювання мають ряд істотних переваг. Так за спектральним складом світла вони близькі до природного світла, мають підвищену світлову віддачу, триваліший термін служби. Норма освітленості на робочих місцях складає 300-500лк.

2.2.4 Вимоги до організації робочого місця працівника

Робочі місця повинні бути розташовані так, щоб у поле зору працюючого не попадали поверхні, що мають властивість віддзеркалювання, вікна освітлювальні прилади. Відеотермінали повинні встановлюватися під кутом 90-100 градусів від

вікон, так, щоб світло падало з боку. Робочі місця з ВДТ доцільно розміщати в глибині приміщення. Розташування відео термінала, при якому працюючий звернений обличчям або спиною до вікон, неприпустимо при будь-якому способі реалізації загального висвітлення, як прямим, так і відбитим світлом

Робочий стіл повинен регулюватися по висоті в границях 680-800 мм, а ширина – забезпечувати можливість виконання операцій в зоні досяжності моторного поля. Рекомендовані розміри столу: висота 725 мм, ширина 600-1400 мм, глибина 800-1000 мм. Робочий стілець повинен бути оснащений підйомно-поворотним пристроєм для регулювання висоти сидіння і спинки, а також кута її нахилу. Регулювання кожного параметра повинне вироблятися легко, бути незалежним і надійно фіксуватися.

2.2.5 Електробезпека

Ураження струмом може виникнути при роботі під напругою і при несправному стані електроустановок, а саме при дотику до оголених проводів, незаземлених металевих корпусах електричного обладнання, при відкритих рубильниках і других струмоведучих частинах.

Для захисту працюючих від ураження електричним струмом передбачені наступні заходи:

- недоступність струмоведучих частин;
- захисне заземлення (занулення) корпусів електрообладнання;
- передбачені рубильники закритого типу;
- блокіровка, надписи, плакати, засоби індивідуального захисту (калоші і боти діелектричні (ГОСТ 13385-78), рукавиці резинові діелектричні, коврики резинові діелектричні (ГОСТ 4997-75).

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном) мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу.

У приміщенні, де одночасно експлуатуються понад п’ять ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення недоступність струмоведучих частин;

2.3 Пожежна безпека

Протипожежний захист приміщення забезпечується застосуванням автоматичної установки пожежної сигналізації, наявністю засобів пожежогасіння, застосуванням основних будівельних конструкцій будинку з регламентованими межами вогнестійкості, організацією своєчасної евакуації людей.

Для ліквідації пожеж використовують первинні засоби пожежогасіння, які призначенні для гасіння пожеж у початковій стадії їх розвитку. Вони є у всіх виробничих приміщеннях, цехах.

Необхідну кількість первинних засобів пожежогасіння визначають окремо для кожного поверху та приміщення. Якщо в одному приміщенні знаходяться декілька різних за пожежною небезпекою виробництв, не відділених одне від одного протипожежними стінами, усі ці приміщення забезпечують вогнегасниками, пожежним інвентарем та іншими видами засобів пожежогасіння за нормами найбільш небезпечного виробництва.

Пожежні щити (стенди) встановлюють на території об’єкта з розрахунку один щит (стенд) на площу 5000м^2. До комплекту засобів пожежогасіння, які розміщуються на ньому, слід включати: вогнегасники – 3шт., ящик з піском – 1шт., покривало з негорючого теплоізоляційного матеріалу або повсті розміром 2м х 2м – 1шт., гаки – 3шт., лопати – 2шт., ломи – 2шт., сокири – 2шт.

Ящики для піску повинні мати місткість 0.5, 1.0 або 3.0м^2 та бути укомплектованими совковою лопатою. Вмістилище для піску, що є елементом конструкції пожежного стенду, повинні бути місткістю не менше 0.1м^3. Конструкція ящика (вмістилище) повинна забезпечувати зручність діставання піску та усування попадання опадів.

ВИСНОВКИ

Досліджена у даній випускній роботі проблема підвищення продуктивності Web-ресурсу шляхом розміщення його не на одному, а на декількох комп'ютерах різної потужності може бути вирішена шляхом розміщення на окремому комп'ютері сервера-додатка балансувальника навантаження. Розподіл потоків підвищує продуктивність Web-ресурсів.

Проведений аналіз сучасних високонавантажених систем показав, що впродовж останніх років навантаження на Web-сайти, Web-додатки та інші типи програмного забезпечення, що використовується в мережі Інтернет, невпинно зростає. При аналізі головних проблем масштабування високонавантажених систем виявлено найпоширеніші підходи в вирішенні цих проблем з використанням балансувальників навантаження.

Визначено відгуки та фактори проведених експериментів. Відгуками експерименту є: час з моменту посилання запиту до його отримання користувачем, пропускна здатність серверу з використанням різного методу балансування навантаження, кількість одночасно встановлених з'єднань та кількість невдалих з'єднань, продуктивність системи при одночасному зверненні сотні користувачів. Факторами, що впливають на результати, є: модель процесора сервера, на якому розгорнута система, модель процесора балансувальника навантаження, тип материнської плати, обсяг оперативної пам'яті та обсяг пам'яті накопичувача, тип бази даних та операційної системи, кількість одночасних користувачів, що здійснюють запити до системи.

У роботі проведені дослідження ефективності різних алгоритмів розподілу потоків з різними конфігураціями обладнання, в результаті чого обрано оптимальний алгоритм Weighted Round Robin та балансувальник навантаження Nginx Upstream Module для статичного та динамічного змісту Web-ресурсів.

Впровадження серверу розподілу потоків не тільки підвищує загальну продуктивність багатосерверної системи шляхом правильного розподілення її ресурсів, а і підвищує відмовостійкість. Балансування навантаження є концепцією, яка все ще перебуває в стані досліджень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волкова В.Н. Системный анализ информационных комплексов. Учебное пособие. М.: Лань, 2017. 336 с.

2. Чжоу Т. Системы балансировки нагрузки Web-серверов [Електронний ресурс] http://www.osp.ru/win2000/2000/03/174228/

3. Емельянов А. Балансировка нагрузки: основные алгоритмы и методы [Електронний ресурс] http://blog.selectel.ru/balansirovka-nagruzkiosnovnye-algoritmy-i-metody/

4. Балансировка нагрузки: основные алгоритмы и методы [Електронний ресурс] https://habr.com/ru/company/selectel/blog/250201/

5. HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer [Електронний ресурс] http://www.haproxy.org/

6. Docker Guide: Installing Traefik [Електронний ресурс] https://www.howtoforge.com/tutorial/ubuntu-docker-traefik-proxy/

7. Digital 2019: global internet use accelerates [Електроний ресурс] https://wearesocial.com/blog/2019/01/digital-2019-globalinternet-use-accelerates

8. Load Balancing 101: Nuts and Bolts [Електроний ресурс] https://f5.com/resources/white-papers/load-balancing-101-nuts-andbolts-31392

9. ZEN Load balancer часть №1. Теория и практика балансировки нагрузки [Електроний ресурс] https://ivirt-it.ru/zen-loadbalancer-theory/

10. Comparing Load Balancing Algorithms [Електроний ресурс] https://www.jscape.com/blog/load-balancing-algorithms

11. Test the Performance of Web Applications Under Load [Електроний ресурс] https://www.loadtestingtool.com/

12. Проектирование и внедрение высоконагруженных систем [Электронный ресурс] http://dotrunet.ru/hiload.html

13. Архитектурные особенности высоконагруженных систем [Электронный ресурс] http://www.highload.ru/2017/abstracts/1076.html

ДОДАТОК А. Лістинги конфігурації програмного забезпечення для балансування навантаження

Лістинг файлу apache-benchmark.p для gnuplot

```
# Setting output to a jpeg file
set terminal jpeg size 1280,720
# Graph aspect ratio setting
set size 1, 1
# Filename to write
set output "graph.jpg"
# Name of the chart
set title "Apache benchmark"
# Legend location
set key left top
# Draw gridlines oriented on the y axis
set grid y
# Specify that the x-series data is time data
set xdata time
# Specify the *input* format of the time data
set timefmt "%s"
# Specify the *output* format for the x-axis tick labels
set format x "%S"
# Label the x-axis
set xlabel 'reuests'
# Label the y-axis
set ylabel "response time (ms)"
# Tell gnuplot to use tabs as the delimiter instead of spaces (default)
set datafile separator '\t'
# Plot the data
plot "result.dat" every ::2 using 2:5 title 'response time' with points
```

Лістинг конфігурації Weighted Round Robin для HAProxy

```
frontend my_HTTP_front
        bind *:80
        default_backend my_HTTP_back
backend my_HTTP_back
        balance roundrobin
        server myweb1 192.168.1.8:80 weight 70
        server myweb2 192.168.1.9:80 weight 30
```

Лістинг конфігурації Weighted Least Connections для HAProxy

```
frontend my_HTTP_front
        bind *:80
        default_backend my_HTTP_back
backend my_HTTP_back
        least_conn
        server myweb1 192.168.1.8:80 weight 70
        server myweb2 192.168.1.9:80 weight 30
```

Лістинг конфігурації Weighted Round Robin для Nginx

```
user  nginx;
worker_processes  3;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
   worker_connections  1024;
}
HTTP {
  upstream backend {
     server 192.168.1.8 weight=70;
     server 192.168.1.9 weight=30;
  }
  server {
     listen 80;
     location / {
        proxy_pass HTTP://backend;
     }
  }
}
#HTTP {
#    include       /etc/nginx/mime.types;
#    default_type  application/octet-stream;
#    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
#    '$status $body_bytes_sent "$HTTP_referer" '
#    '"$HTTP_user_agent" "$HTTP_x_forwarded_for"';
#    access_log  /var/log/nginx/access.log  main;
#    sendfile        on;
#    #tcp_nopush     on;
#    keepalive_timeout  65;
```

Лістинг конфігурації Weighted Least Connections для Nginx

```
user  nginx;
worker_processes  3;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
   worker_connections  1024;
}
HTTP {
  upstream backend {
     least_conn;
     server 192.168.1.8 weight=70;
     server 192.168.1.9 weight=30;
  }
  server {
     listen 80;
     location / {
        proxy_pass HTTP://backend;
     }
  }
}
#HTTP {
```

```
#    include       /etc/nginx/mime.types;
#    default_type  application/octet-stream;
#    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
#    '$status $body_bytes_sent "$HTTP_referer" '
#     '"$HTTP_user_agent" "$HTTP_x_forwarded_for"';
#    access_log  /var/log/nginx/access.log  main;
#    sendfile        on;
#    #tcp_nopush     on;
#    keepalive_timeout  65;
```

Лістинг конфігурації Weighted Round Robin для Pound

```
ListenHTTP
   Address 192.168.1.10
   Port 8080
End
Service
   BackEnd
      Address 192.168.1.8
      Port    80
     Priority 7
   End
  BackEnd
      Address 192.168.1.9
      Port    80
      Priority 3
   End
EndP
```

Лістинг файлу locustfile.py

```
from locust import HttpLocust, TaskSet
def login(l):
   l.client.post("/wp-login.php", {"user_login":"benchuser",
" user_pass ":"benchpassword"})
def index(l):
   l.client.get("/")
def post(l):
   l.client.get("/?p=1")
class UserBehavior(TaskSet):
   tasks = {index: 2, post: 1}
   def on_start(self):
      login(self)
class WebsiteUser(HttpLocust):
   task_set = UserBehavior
   min_wait = 5000
   max_wait = 9000
```

ДОДАТОК Б

Слайди мультимедійної презентації

Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів

Задворний Максим Євгенович, "ОТФК ОНТУ"

Схема розподілу потоків навантаження для архітектури Frontend-Backend

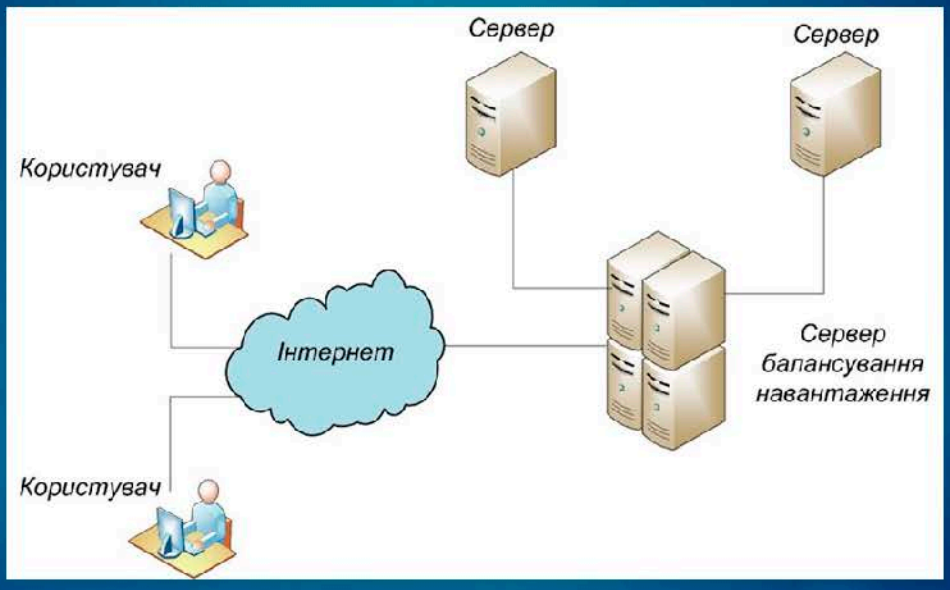
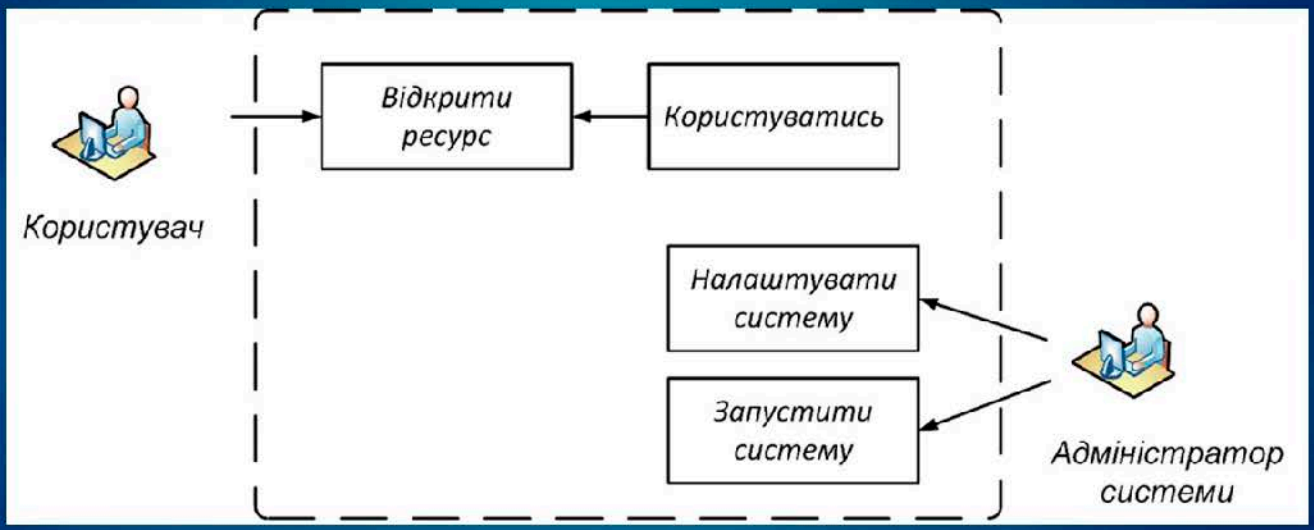
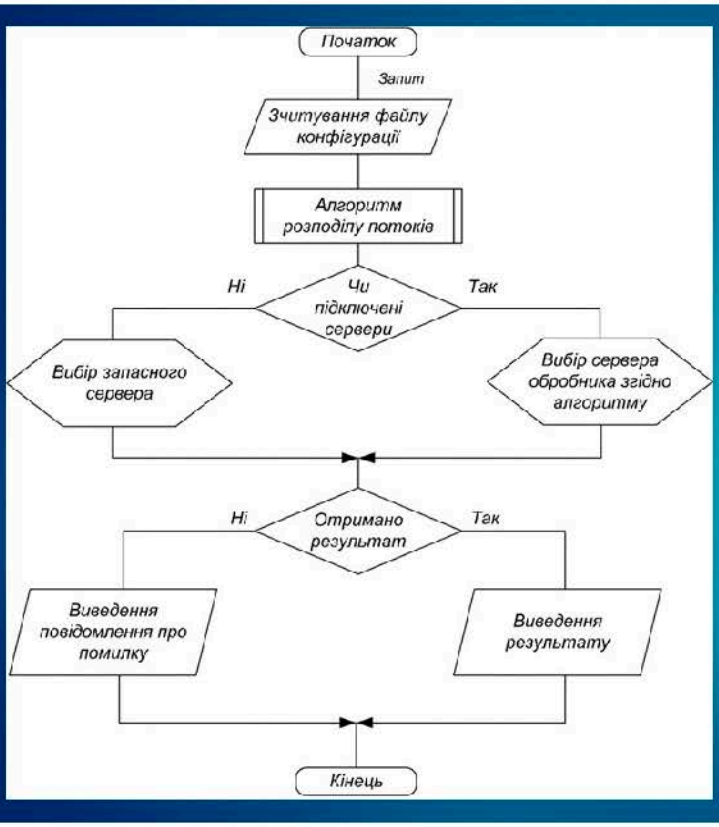


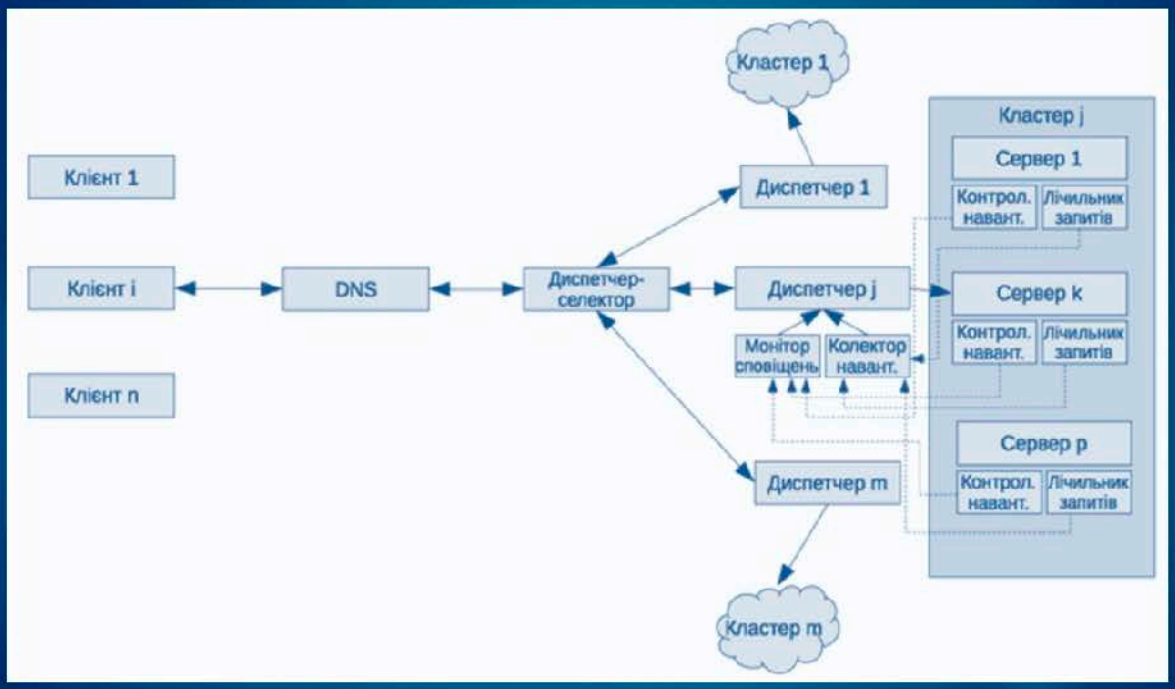
Схема роботи системи розподілу потоків навантаження Web-серверів



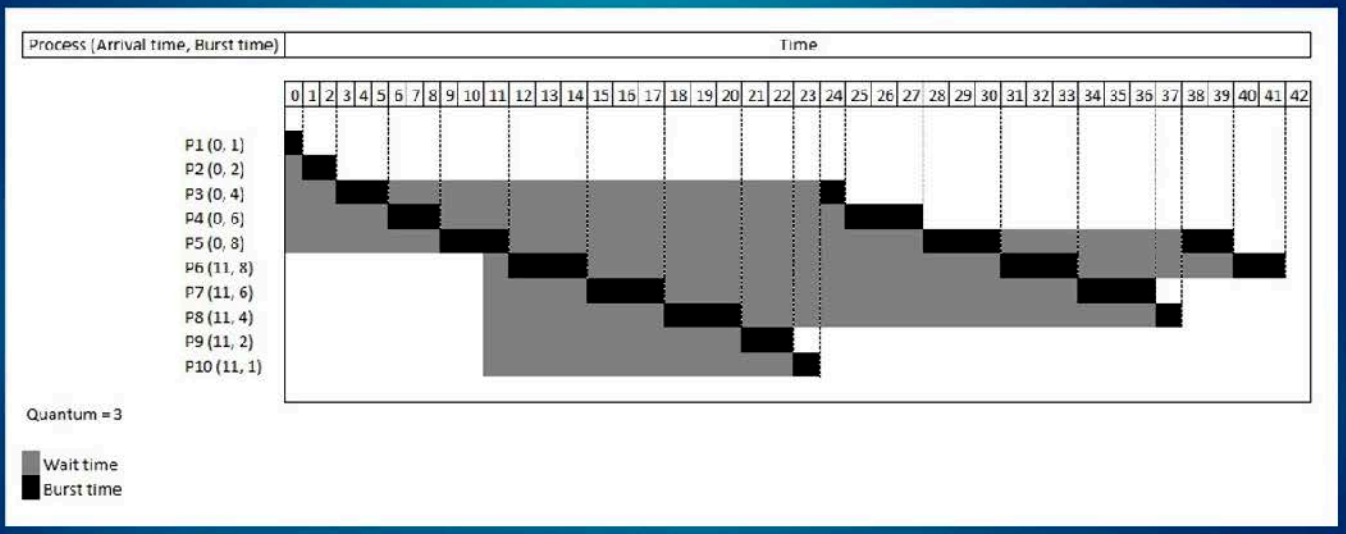


Блок-схема роботи системи розподілу потоків навантаження Web-серверів

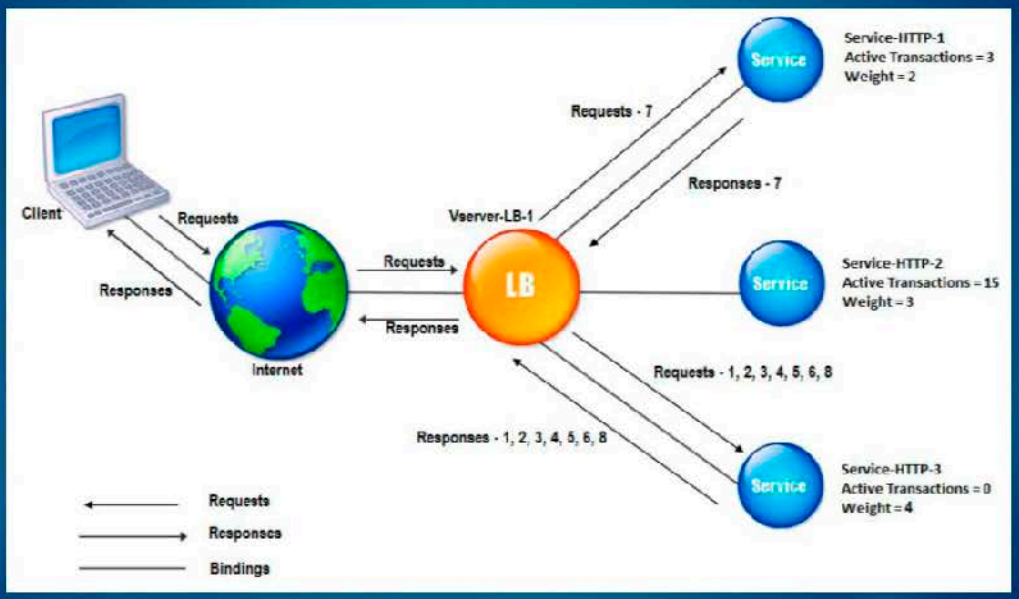
Архітектурне рішення алгоритму балансування навантаження



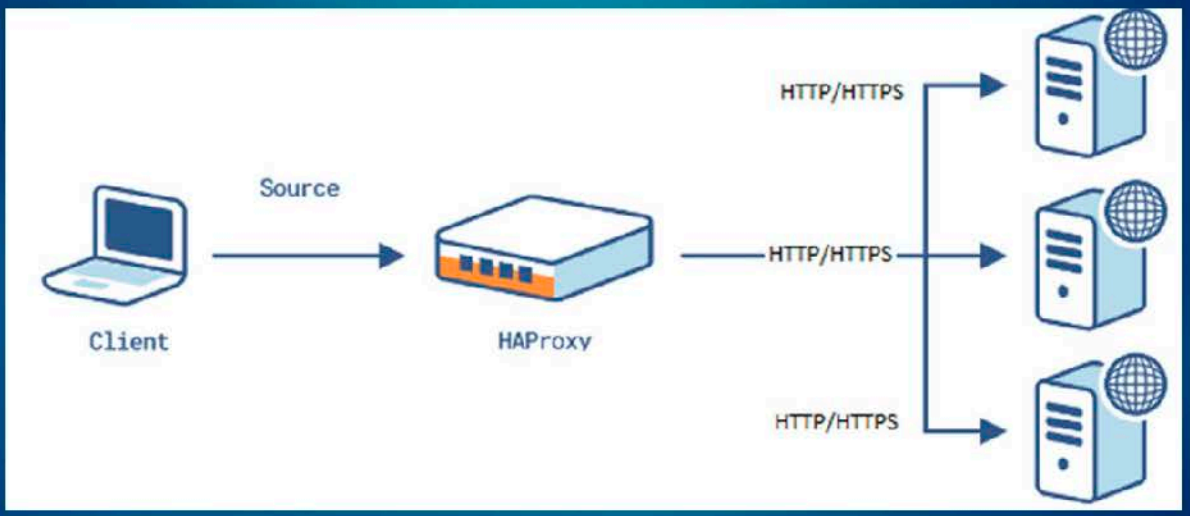
Принцип роботи алгоритму Round Robin



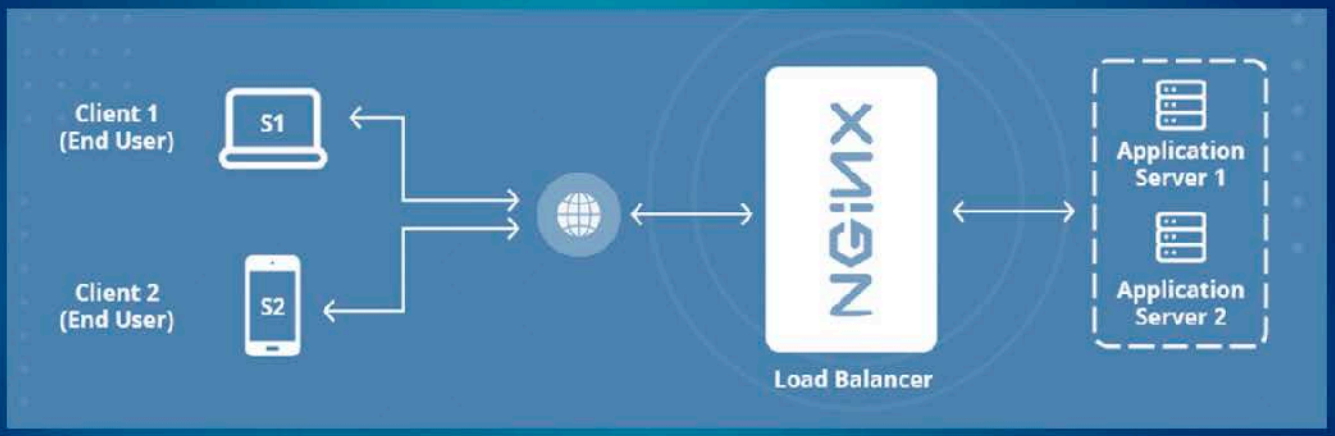
Механізм балансувальника навантаження з використанням алгоритму Weighted Least Connections



Організація мережі з балансувальником навантаження HAProxy



Організація мережі з балансувальником Nginx



Характеристики програмних рішень балансувальників навантаження

	HAProxy	Nginx	Traefik	Pound
Операційна система	Linux 2.6.32+ (рекомендовано для максимальної продуктивності) та 2.4, Solaris 8–10, FreeBSD, OpenBSD	Windows, CentOS, Debian, Ubuntu, SLES, Apline	Docker-контейнер	Linux, Solaris, OpenBSD
Алгоритми	Round Robin, Random, Weighted Random, Least Session, Least Bandwidth, Hash, Agent і Randomized Agent	Weighted Round Robin, Weighted Least-connections, IP-hash	Weighted Round Robin, Dynamic Round Robin	Least Session, Round Robin
Протоколи	HTTP, HTTPS	HTTP, HTTPS, FastCGI, SCGI, gRPC	HTTP, HTTPS	SSL, HTTP(s)

Архітектура системи балансування навантаження

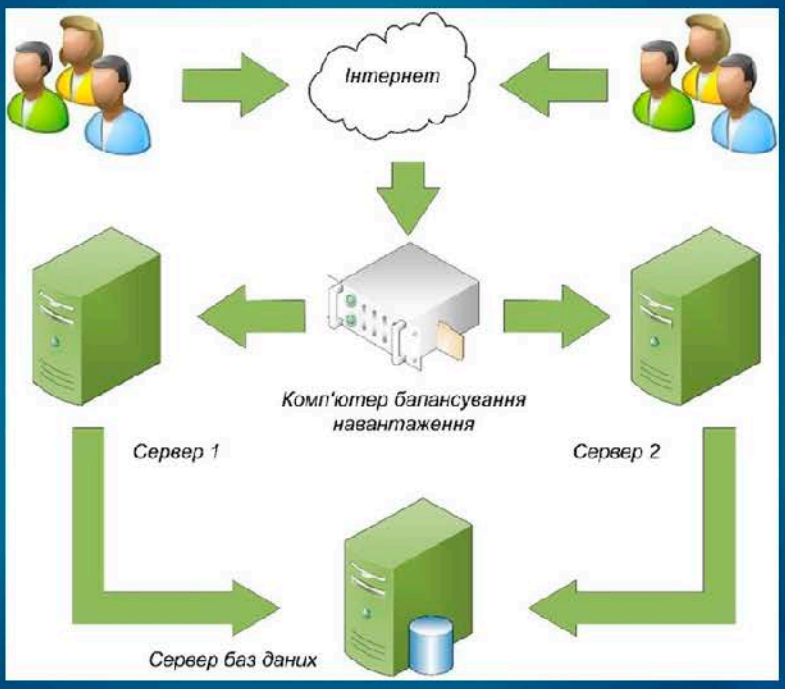
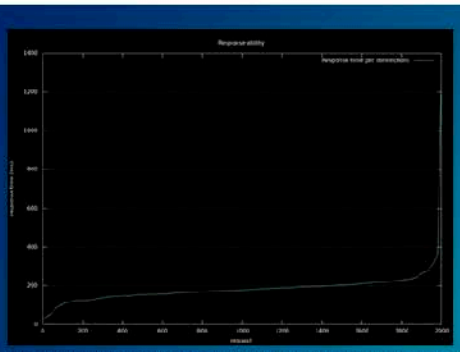
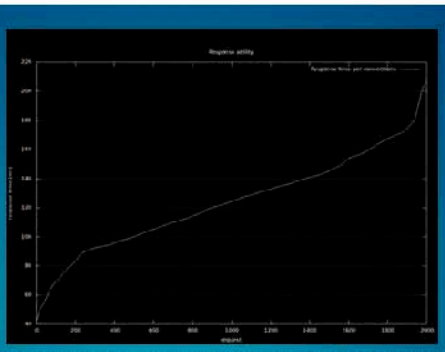
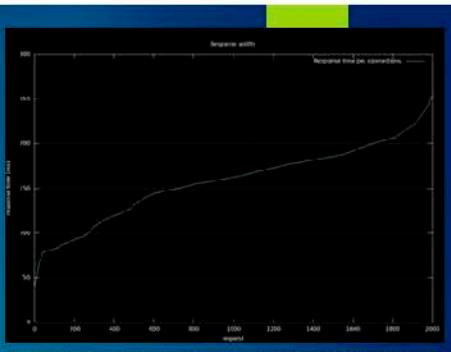
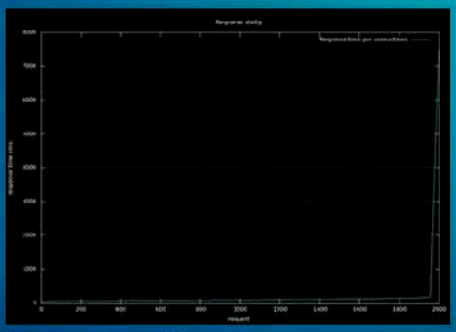
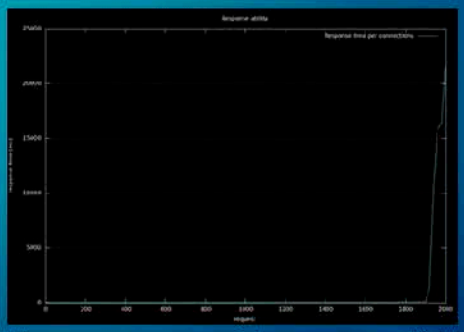


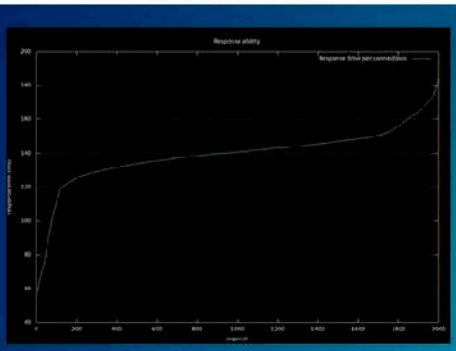
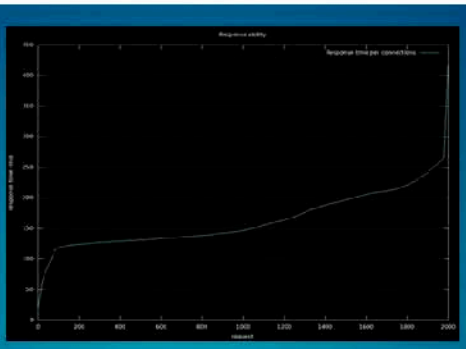
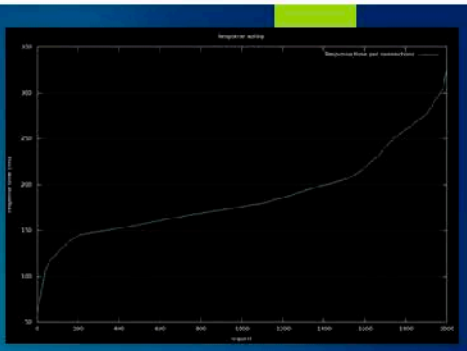
Схема експериментальних досліджень

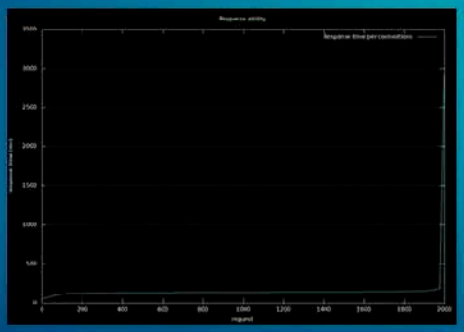
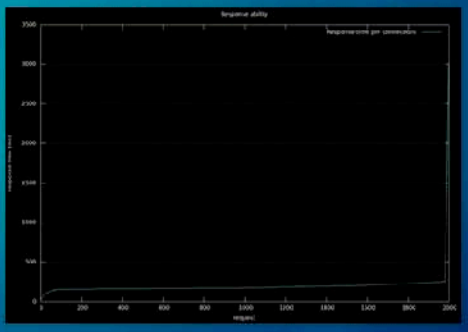
Графіки продуктивності без балансувальника навантаження для початкової конфігурації на статичному змісті з запущеним Web-сервером Nginx на віртуальній машині №1 (зліва), балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin (посередині) та Weighted Least Connections (справа) для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"

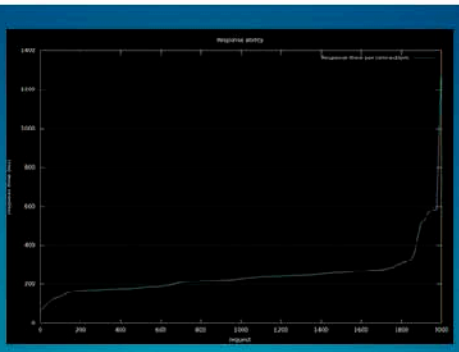
Графіки продуктивності балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює" (зліва) та "комп'ютер 1 працює, комп'ютер 2 вимкнений" (справа)

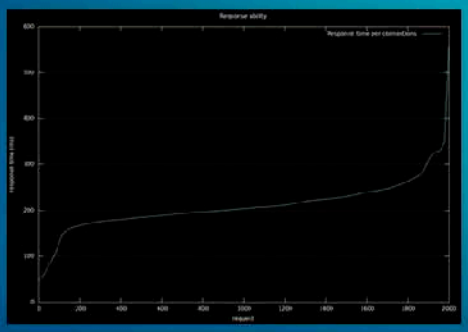
Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює" (зліва), за алгоритмом Weighted Round Robin (посередин) та Weighted Least Connections (справа) для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"

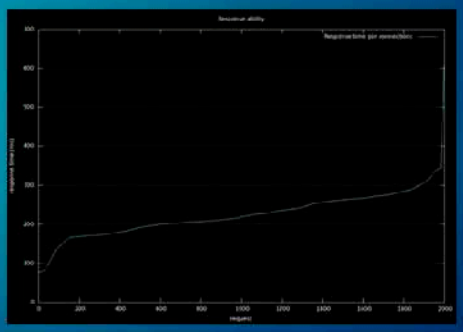



Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Weighted Least Connections (справа) для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

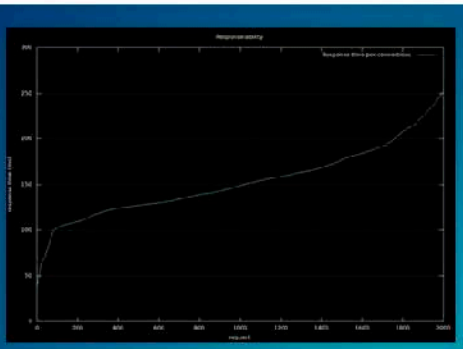


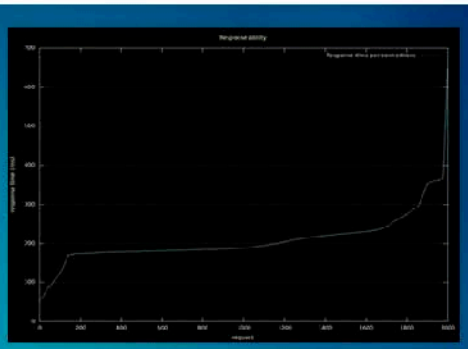
Графік продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"



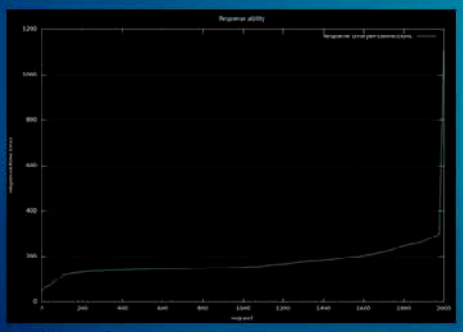


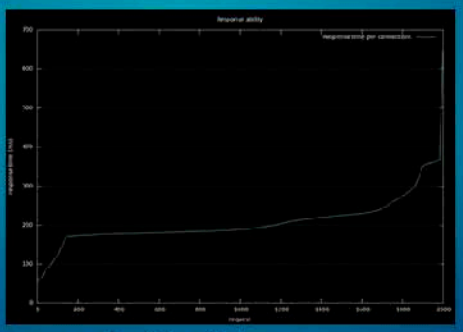
Графіки продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює" (зліва) та "комп'ютер 1 вимкнений, комп'ютер 2 працює" (справа)

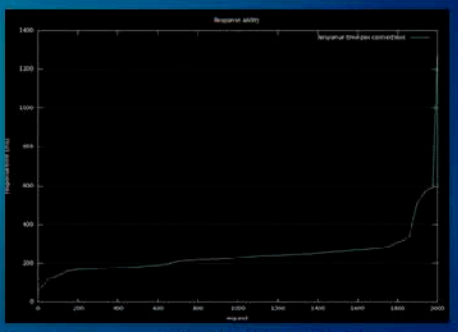




Графіки продуктивності балансувальника навантаження Traefik на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Dynamic Round Robin (справа) для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"







Графіки продуктивності балансувальника навантаження Traefik Load Balancer на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює" (зліва), для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений" (посередині) та за алгоритмом Dynamic Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений" (справа)

Результати тестування продуктивності без балансувальника навантаження на динамічному змісті у початковій конфігурації для Wordpress

Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"

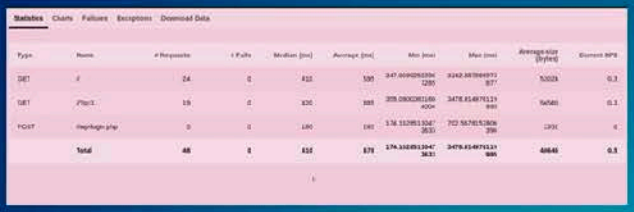
Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

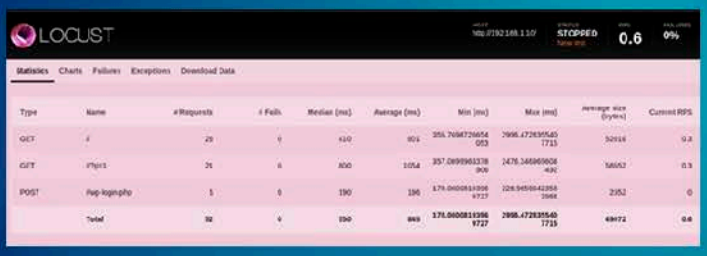
Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"



Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"

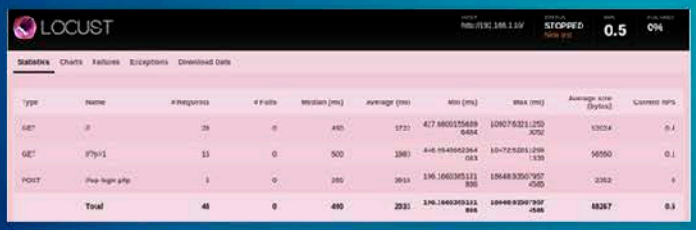


Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"

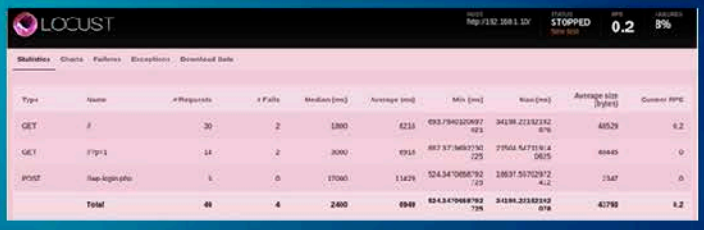
Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Least Connections для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"



Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"



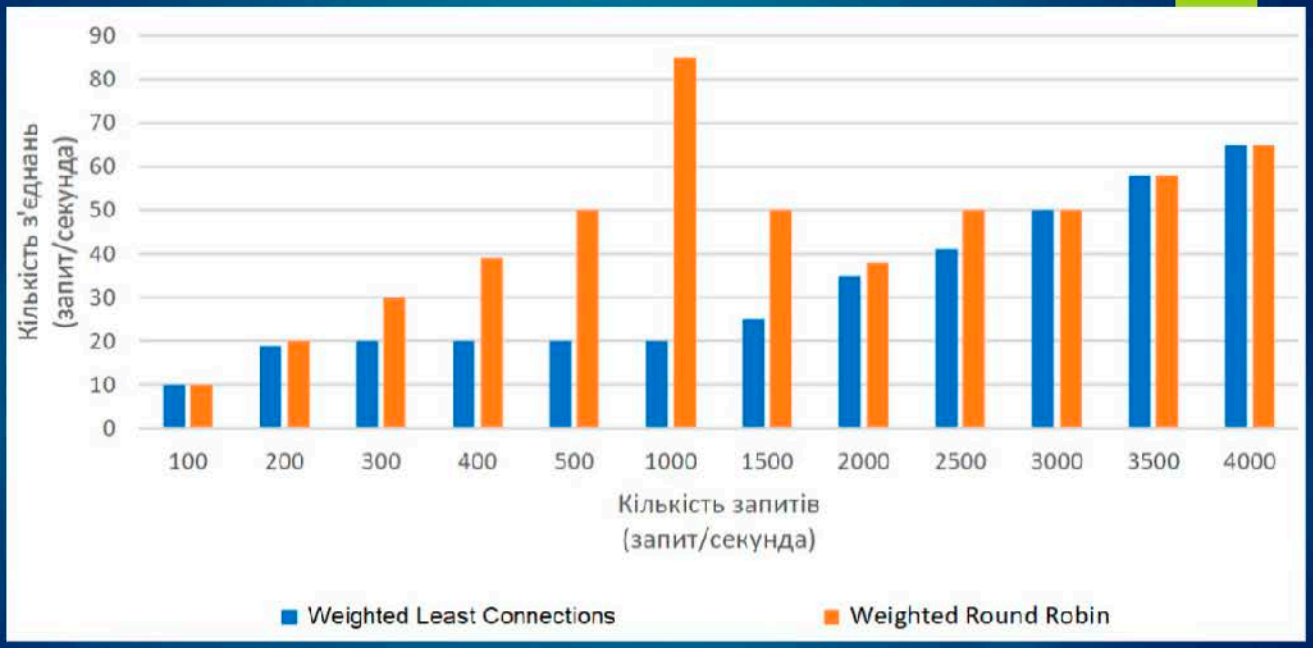
Результати тестування продуктивності балансувальника навантаження Pound на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

Максимальний рейтинг з'єднань алгоритмів розподілу потоків



UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015241500

Дата перевірки:
25.05.2023 10:28:15 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2023 10:34:22 EEST

ID користувача:
100011688

Назва документа: 2БКС-27 Задворний М.Є.

Кількість сторінок: 46 Кількість слів: 7514 Кількість символів: 57405 Розмір файлу: 3.70 MB ID файлу: 1014917307

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

16.8%
Схожість

Найбільша схожість: 7.29% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/28519/1/Reshchenko_bakalavr...

16.8% Джерела з Інтернету | 619 ..... Сторінка 48

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	4

Підозріле форматування	15 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Задворний Максим Євгенович,
здобувач освіти гр. 2БКС-27, та

Кривченко Юрій Вікторович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи бакалавра на тему:

«Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів» (автор роботи – Задворний М.Є., керівник роботи – Кривченко Ю.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Задворний М.Є. /

Керівник _____ / Кривченко Ю.В./

«15» червня 2023 р.

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

керівника про випускну роботу бакалавра

Задворного Максима Євгеновича

(прізвище, ім`я та по батькові)

Спеціальність 123 "Комп'ютерна інженерія"

Тема випускної роботи Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) Обсяг і якість виконання роботи (графічного матеріалу і розрахунково-пояснювальної записки) Випускна робота виконана відповідно технічному завданню. Пояснювальна записка до випускної роботи містить 65 сторінок. У пояснювальній записці розглянуті алгоритми розподілу потоків для підвищення продуктивності web-ресурсів, а також програмні засоби реалізації балансувальників навантаження. Графічна частина складається з 22 слайди, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра, розробку виконано у повному обсязі.

б) Самостійність роботи

Протягом виконання випускної бакалаврської роботи Задворний М.Є. поступово та послідовно виконувала всі етапи, проявила ініціативу у створенні загальної концепції та реалізації випускної роботи. Всі роботи вона виконувала самостійно, з оглядом на рекомендації керівника.

в) Теоретична підготовка здобувача освіти

Задворний М.Є. під час роботи над випускною бакалаврською роботою вивчив достатню кількість літературних джерел за даною тематикою.

Вважаю, що теоретична підготовка здобувача освіти добра і він готовий до захисту роботи.

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

Під час виконання роботи Задворний М.Є. мав змогу самостійно приймати окремі рішення з виконання програмної частини роботи та показав вміння організовано працювати над поставленою задачею, користуючись сучасними комп'ютерними програмними засобами, такими як Apache Benchmark, Gnuplot, Locust та ін.

Оцінка розрахункової частини Добре

Оцінка графічної частини Відмінно

Загальна оцінка Добре

Прізвище, ім`я, по батькові Кривченко Юрій Вікторович

Місце роботи і посада керівника роботи ВСП "Одеський технічний фаховий коледж ОНТУ", викладач кафедри комп'ютерної інженерії

Підпис

« 8 » червня 2023 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти
відділення комп'ютерних систем

Задворного Максима Євгеновича

(прізвище, ім`я та по батькові)

Спеціальність 123 Комп'ютерна інженерія

Освітня програма Комп'ютерна інженерія

Керівник кваліфікаційної роботи Кривченко Юрій Вікторович

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *Дослідження алгоритмів розподілу потоків для підвищення продуктивності web-ресурсів*

Обсяг розрахунково-пояснювальної записки 65 сторінок

Обсяг графічної (презентаційної) частини 22 аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію випускна кваліфікаційна робота відповідає затвердженій темі та виконаний відповідно до ТЗ. Випускна робота має актуальну тематику щодо розподілу потоків для підвищення продуктивності web-ресурсів, а також ПЗ реалізації балансувальників навантаження

б) характеристика виконання кожного розділу кваліфікаційної роботи

Пояснювальна записка складається з технологічного розділу, розділу охорони праці та додатку. Технологічний розділ пояснювальної записки містить підрозділи, що поетапно охоплюють аналітичну частину, реалізацію суті роботи, дослідження ефективності прийнятих рішень. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора ЕОТ

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Графічна частина складається з 22 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять креслення та ілюстративні схеми, таблиці, графіки, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів. Якість виконання графічної частини роботи та пояснювальної записки висока, розробку виконано у повному обсязі

г) перелік позитивних якостей кваліфікаційної роботи
У роботі проведені дослідження ефективності різних алгоритмів розподілу потоків з різними конфігураціями обладнання, в результаті чого обрано оптимальний алгоритм Weighted Round Robin та балансувальник навантаження Nginx Upstream Module для статичного та динамічного змісту Web-ресурсів

д) основні недоліки кваліфікаційної роботи
У пояснювальній записці багато уваги приділено огляду методів і засобів розподілу потоків Web-ресурсів.
Не проведено посилання на використану літературу

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, к.ф.н., доцент кафедри КБ та ТЗІ, пом.декану факультету інформаційних технологій та кібербезпеки

Підпис:

« 16 » червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ЗВ'ЯЗКУ * 43997335 * ВІДДІЛ КАДРІВ