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

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна інженерія»

Група: 2БКС-29

КВАЛІФІКАЦІЙНА РОБОТА

здобувача освіти денної форми навчання

БКС.29.10.000.КРБ

ІЛЬЇНА
СЕРГІЯ СЕРГІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Освітньо-професійна програма: **«Комп'ютерна інженерія»**

Група: **2БКС-29**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів**

Проектний матеріал складається з пояснювальної записки на 71 сторінках та графічного (презентаційного) матеріалу на 18 аркушах (слайдах)

Виконавець _____ (Ільїн С.С.)

Керівник проекту _____ (Кривченко А.А.)

Консультанти:

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «17» 06 2025 р. Протокол ЕК № 2

Оцінка ЕК 4 (добре) / 85

Секретар ЕК _____

АНОТАЦІЯ

Випускна кваліфікаційна робота присвячена дослідженню алгоритмів балансування навантаження з метою підвищення продуктивності сучасних веб-ресурсів при обробці великої кількості запитів. У роботі проведено аналіз як класичних, так і сучасних методик розподілу потоків, зокрема алгоритмів Round Robin, Weighted Round Robin, Weighted Least Connections та інших підходів, що враховують неоднорідність апаратних засобів та змінність навантаження. Теоретична частина проекту охоплює вивчення принципів роботи основних балансувальників навантаження, серед яких – HAProxy, Nginx, Traefik та Pound, а також їх інтеграцію у складні мережеві архітектури.

Практична частина дослідження базується на створенні експериментальної моделі, що імітує роботу сучасного VPS-хостингу з використанням віртуалізованих серверів та спеціалізованих інструментів навантажувального тестування (Apache Benchmark, Locust, Gnuplot). Експериментальний аналіз був проведений для статичного (HTML-сторінка) та динамічного (CMS WordPress) змісту, що дозволило оцінити продуктивність системи в різних сценаріях роботи та з різними конфігураціями апаратного забезпечення. Результати тестування показали, що оптимальну ефективність демонструє балансувальник навантаження Nginx Upstream Module із застосуванням алгоритмів Weighted Round Robin та Weighted Least Connections, що дозволяє значно зменшити час відповіді серверів навіть при високих навантаженнях.

Практична значущість проекту полягає у наданні чітких рекомендацій щодо вибору оптимальних алгоритмів розподілу потоків для високотрафічних веб-систем, що сприятиме підвищенню відмовостійкості та ефективності сучасних IT-інфраструктур. Отримані результати можуть бути використані для оптимізації роботи веб-ресурсів у комерційних та науково-дослідних проектах, де критично важливими є швидкість обробки запитів та якість обслуговування кінцевих користувачів.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"28" 05 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачеві освіти *Ільїна Сергія Сергійовича*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів*

затверджена наказом по коледжу від "24" 04 2025 р. № 216

2. Термін здачі студентом кваліфікаційної роботи 18.06.25

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, що їх належить розробити)

Аналітичний огляд методів та засобів балансування навантаження; Вибір балансувальників навантаження при обробці запитів великої кількості користувачів; Підготовка апаратних засобів для дослідження роботи балансувальників навантаження; Тестування роботи алгоритмів балансування навантаження при обробці запитів великої кількості користувачів

5. Перелік графічного матеріалу (слайдів мультимедійної презентації) *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; Рейтинг з'єднань алгоритмів розподілу потоків*

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

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний розділ	*Кривченко А.А.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник роботи *Кривченко А.А.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Вступ. Аналіз технічного завдання	06.06.25	
2.	Аналіз методів і засобів розподілу потоків	07.06.25	
3.	Визначення рівнів OSI для розподілу потоків	08.06.25	
4.	Огляд алгоритмів розподілення потоків	09.06.25	
5.	Створення моделі балансування навантаж. web-серверів	10.06.25	
6.	Вибір програмних рішень балансувальників навантаження	11.06.25	
7.	Вибір апаратних засобів для дослідження алгоритмів	12.06.25.	
	розподілу потоків і балансування навантаження		
8.	Налаштування програмних засобів для тестування	13.06.25	
9.	Дослідження приросту продуктивності на	14.06.25 –	
	статичному змісті	– 15.06.25	
10.	Дослідження приросту продуктивності на	16.06.25 –	
	динамічному змісті	– 17.06.25	
11.	Аналіз результатів тестування	18.06.25	
12.	Розробка питань з охорони праці та техніки безпеки	19.06.25.	
13.	Підготовка матеріалів мультимедійної презентації	20.06.25	

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 29. 10 000. 00 КРБ	Випускна робота		
А4			БКС 29. 10 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 29. 10 000. 00 КРБ		
Розробив		Ільїн С.С.		10.06	Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів	Літ.	Аркуш 4 / Аркушів 71
Перевірив		Кривченко А.А.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		16.06.06		ВСП "ОТФК ОНТУ" гр.2БКС-29	
Затверд.		Іванова Л.В.					

ЗМІСТ

ВСТУП

У сучасному світі, де інтернет став невід'ємною частиною повсякденного життя, забезпечення високої доступності та стабільної роботи веб-ресурсів набуває все більшої актуальності. Ріст кількості користувачів, що одночасно взаємодіють із системою, вимагає застосування ефективних методів управління навантаженням для забезпечення безперебійної роботи та високої продуктивності сервісів. Випускна кваліфікаційна робота буде присвячена аналізу алгоритмів балансування навантаження при обробці запитів великої кількості користувачів.

Основною ідеєю дослідження стало використання розподіленої архітектури, завдяки якій веб-ресурс розміщується не на одному сервері, а на декількох комп'ютерах, що мають різну потужність. Такий підхід дозволяє гнучко адаптувати систему до змін умов експлуатації та оптимізувати використання ресурсів, враховуючи специфіку кожного обладнання. Особливістю розглядуваної системи є наявність окремого комп'ютера, на якому розміщено сервер-додаток балансувальника навантаження. Сам балансувальник виконуватиме ключову роль у перерозподілі потоків запитів між вузлами, що дозволяє забезпечити більш рівномірне розподілення ресурсів та уникнути перевантаження окремих серверів.

У ході дослідження буде проводитимуся порівняльний аналіз різних алгоритмів розподілу потоків запитів з метою визначення найбільш ефективного підходу для різних конфігурацій обладнання. Значна увага має приділятися практичним аспектам реалізації системи, тому експерименти будуть проводитись з урахуванням реальних умов експлуатації веб-ресурсу. Як результат, отримані дані дозволяють не лише оцінити продуктивність окремих алгоритмів, але й зробити рекомендації щодо їх застосування в залежності від апаратних можливостей серверів, що є важливим елементом при проектуванні сучасних високонавантажених систем. Дослідження у даній роботі спрямоване на удосконалення підходів до балансування навантаження та забезпечення високої продуктивності веб-ресурсів шляхом оптимального розподілу запитів між серверами різної потужності. Отримані результати можуть сприяти підвищенню якості обслуговування кінцевих користувачів деякого веб-ресурсу.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналітичний огляд методів та засобів балансування навантаження

У сучасних системах, де веб-ресурси обслуговують велику кількість одночасних користувачів, забезпечення ефективного розподілу запитів є критично важливим для стабільності роботи та високої продуктивності. Методи балансування навантаження дозволяють розподіляти вхідний трафік між множиною серверів, що значно підвищує надійність системи та забезпечує оптимальне використання апаратних ресурсів. У контексті архітектури Frontend-Backend, де фронтенд відповідає за прийом запитів від користувачів, а бекенд — за обробку бізнес-логіки та роботу з базами даних, розподіл потоків має враховувати різні характеристики та можливості кожного рівня.

1.1.1 Огляд алгоритмів балансування навантаження веб-ресурсів

Алгоритм Round Robin (рис.1.1) є одним із найпростіших методів балансування навантаження, що послідовно розподіляє запити між серверами за принципом циклічності. Цей підхід добре працює у випадках, коли всі сервери мають ідентичні апаратні характеристики. Проте у архітектурі Frontend-Backend, де бекенд-сервери можуть мати різну потужність, стандартний Round Robin може призводити до нерівномірного завантаження, оскільки не враховує індивідуальні можливості кожного вузла.

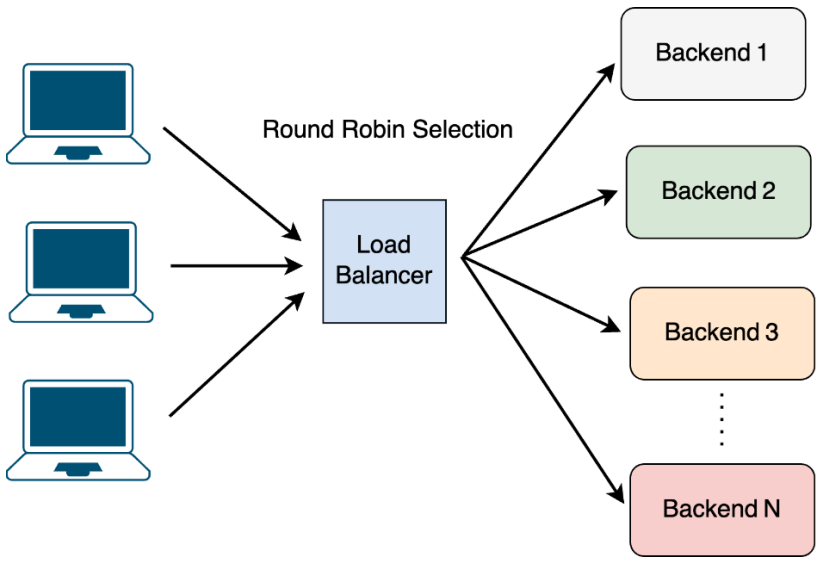


Рисунок 1.1. Принцип балансування навантаження Round Robin

Для вирішення проблеми неоднорідності апаратного забезпечення використовується модифікований алгоритм — Weighted Round Robin. У цьому методі кожному серверу присвоюється певна вага відповідно до його обчислювальних можливостей. Сервер із високою потужністю отримує більшу кількість запитів, що дозволяє більш ефективно використовувати ресурси системи. Такий підхід особливо доречний у середовищах, де фронтенд може приймати велику кількість запитів, а бекенд-сервери різняться за конфігураціями.

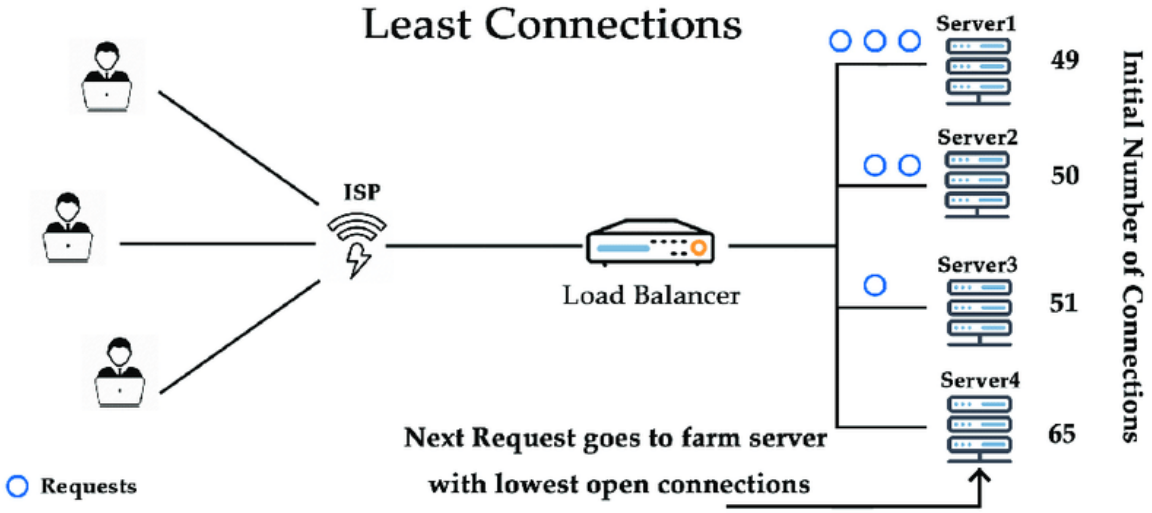


Рисунок 1.2. Принцип балансування навантаження Least Connections

Метод Least Connections (рис.1.2) орієнтований на поточну завантаженість серверів. Він спрямовує запит до того сервера, який у даний момент обслуговує найменшу кількість активних з'єднань. Завдяки цьому алгоритм динамічно реагує на зміну навантаження і допомагає уникнути ситуацій перевантаження, що є корисним при різних сценаріях роботи веб-ресурсу, де кількість одночасних запитів може суттєво варіюватися.

Метод IP Hash (рис.1.3) використовує хеш-функцію для розрахунку значення по IP-адресі клієнта, що дозволяє завжди направляти запити з однієї IP-адреси до одного й того самого сервера. Це забезпечує стабільність сесій користувачів, що є особливо важливим для додатків, де необхідне збереження стану сесії. Такий підхід дозволяє уникнути проблем, пов'язаних із розривом сесій при розподілі запитів між різними вузлами.

Окрім класичних алгоритмів, сучасні системи часто використовують комбінацію методів для досягнення оптимального розподілу навантаження. У

багаторівневій архітектурі Frontend-Backend, фронтенд може виступати в ролі первинного фільтра, що аналізує тип запиту і напрямлює його до відповідного бекенд-сервера. Подальше розподілення вже проводиться з урахуванням поточної завантаженості та обчислювальних можливостей окремих вузлів, де може застосовуватись алгоритм Weighted Round Robin або Least Connections.

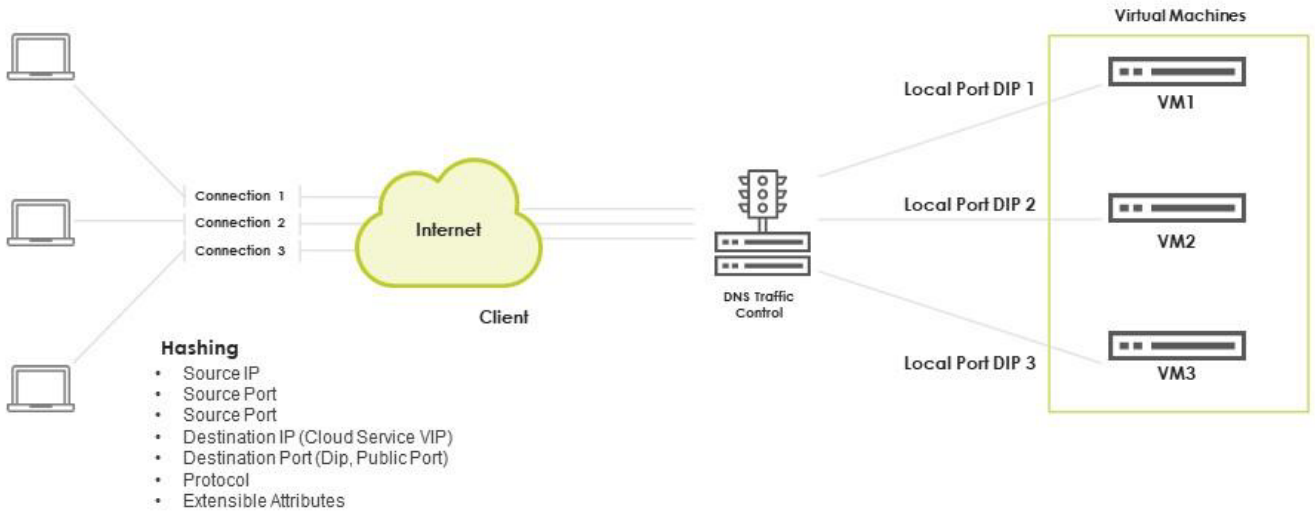


Рисунок 1.3. Принцип балансування навантаження Метод IP Hash

У архітектурі балансування навантаження для Frontend-Backend веб-системи розподіл запитів протікає у два етапи (рис.1.4). На рівні Frontend здійснюється первинне приймання запитів, де часто застосовуються методи балансування, що забезпечують швидке реагування та первинну фільтрацію трафіку. Після цього запити передаються до бекенд-серверів, де застосовуються детальні алгоритми балансування, що враховують як кількість активних з'єднань, так і потужність кожного окремого сервера. Такий підхід дозволяє забезпечити оптимальну маршрутизацію запитів, що сприяє високій продуктивності системи навіть при різноманітних конфігураціях апаратних засобів.

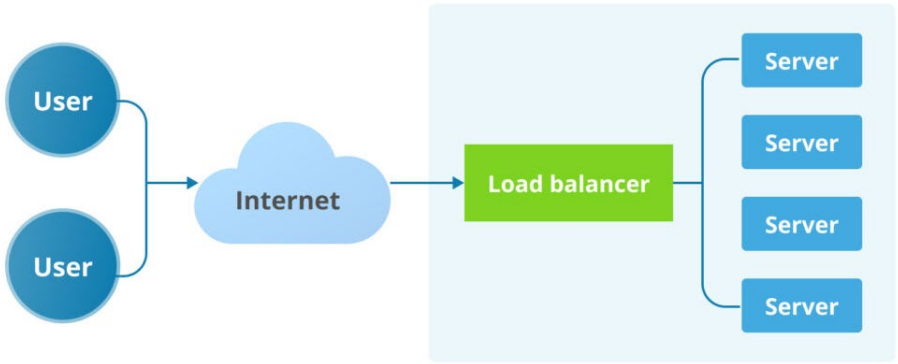


Рисунок 1.4. Принцип балансування навантаження для Frontend-Backend

1.1.2 Аналіз засобів на рівнях OSI для балансування навантаження

Балансування навантаження веб-ресурсів може здійснюватися на різних рівнях моделі OSI: мережевому, транспортному та прикладному. Кожен рівень має свої особливості та застосування в контексті розподілу потоків, що забезпечує оптимізацію роботи системи та високу її доступність.

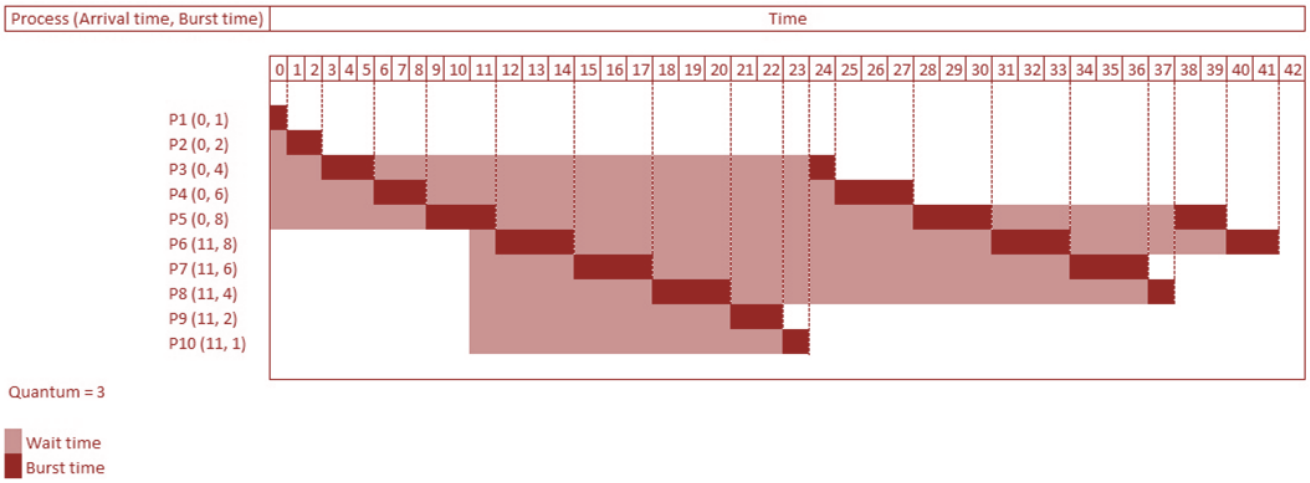


Рисунок 1.5. Циклічний розподіл запитів між IP-адресами (алгоритм Round Robin)

На мережевому рівні розподіл запитів базується на використанні IP-адрес. В системах балансування навантаження певна кількість IP-адрес асоціюється з одним DNS-сервером, який при запитах вибирає одну з них для перенаправлення запиту до сервера. Найчастіше при цьому застосовується алгоритм Round Robin, що циклічно розподіляє запити між IP-адресами (рис. 1.5).

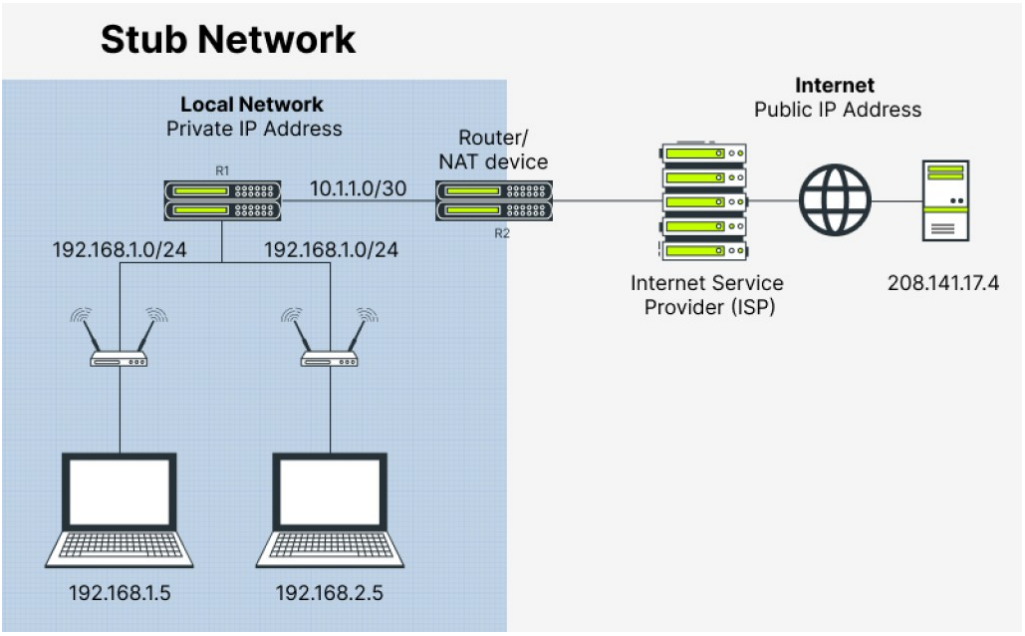


Рисунок 1.6. Географічний розподіл запитів для клієнтів у мережі CDN

Заначений підхід забезпечує базове розподілення трафіку, проте не враховує поточне навантаження серверів. При побудові NBL-кластеру сервери об’єднуються в кластери, що дозволяє підвищити ефективність обробки запитів. Крім того, для розподілу потоків може використовуватись апаратний пристрій, наприклад, маршрутизатор, або ж мережа CDN, яка географічно розподіляє запити для клієнтів, що знаходяться на різних континентах (рис. 1.6).

На транспортному рівні балансування навантаження здійснюється за допомогою спеціалізованих балансувальників, які перенаправляють запити згідно з певним алгоритмом. Найпопулярнішим є циклічний перебір серверів із вибором найменш завантаженого вузла, що дозволяє адаптувати розподіл потоків залежно від поточної ситуації. На відміну від мережевого рівня, який лише розподіляє вхідний трафік "як є", балансувальник на транспортному рівні (наприклад, HAProxy) входить у контакт із клієнтом, виступаючи як проксі-сервер. У цьому процесі інформація про клієнта передається у заголовках, що дозволяє серверу Backend відновлювати сесію користувача. Так організовано роботу в мережі з балансувальником, як це проілюстровано на рис. 1.7.

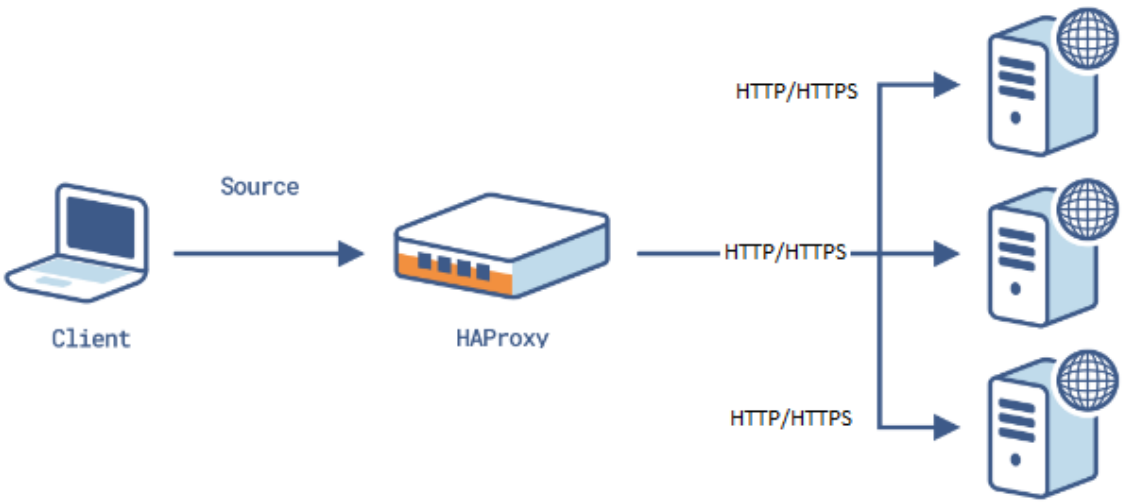


Рисунок 1.7. Робота мережі з балансувальником HAProxy

На прикладному рівні розподілення потоків здійснюється з урахуванням змісту запитів. Балансувальники, які працюють на цьому рівні, аналізують інформацію у запитах та направляють їх на відповідні сервери. Прикладом такого підходу є робота Nginx у ролі балансувальника, де модуль Upstream здійснює розподіл запитів відповідно до їх характеристик (рис. 1.8). Подібну

функціональність можна знайти й у рішенні для СУБД, зокрема у PostgreSQL за допомогою інтерфейсу pgpool, що дозволяє розподіляти запити за типом операції (читання або запис), сприяючи ефективнішому використанню ресурсів.

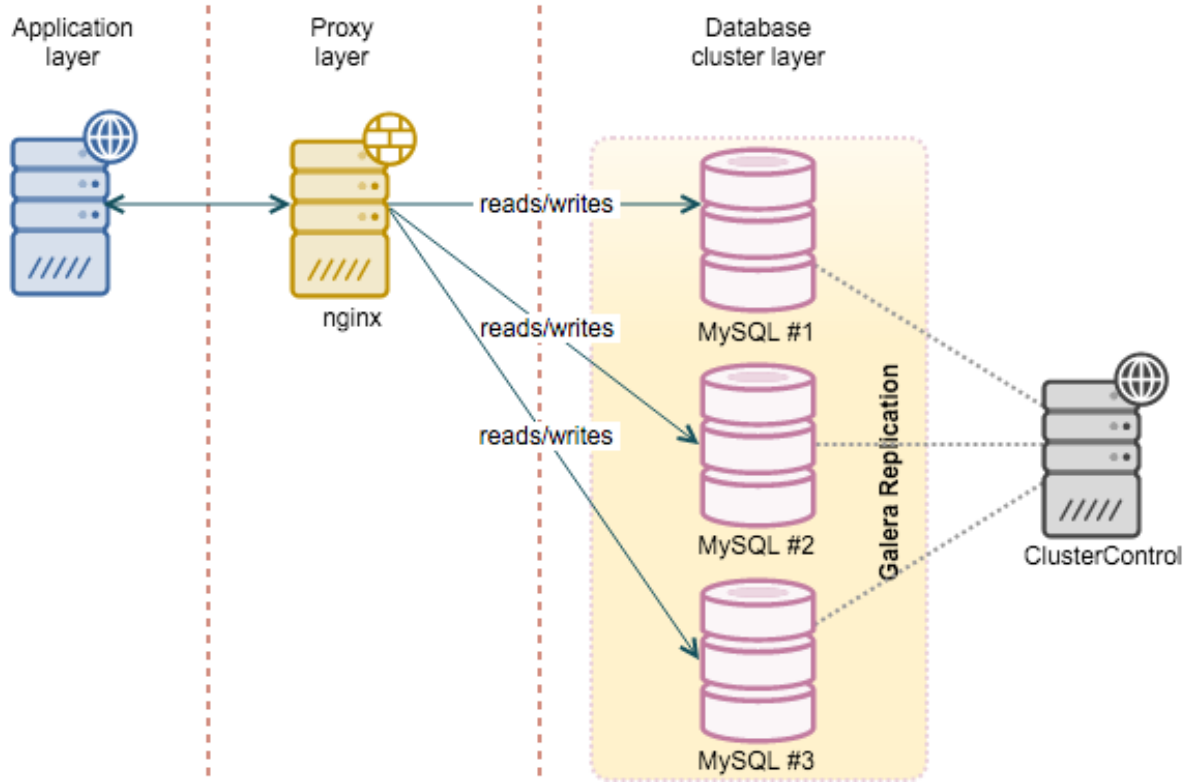


Рисунок 1.8. Розподіл запитів у мережі з балансувальником Nginx

1.1.3 Реалізація модифікованої моделі балансування навантаження веб-серверу

Розглянуті вище алгоритми балансування навантаження мають як переваги, так і недоліки. Наприклад, алгоритм DNS Round Robin є досить простим у реалізації, проте має суттєвий недолік: при відключенні одного з серверів користувач, попадаючи на недоступну IP-адресу, отримує помилку, незважаючи на те, що інші вузли продовжують працювати коректно. Саме тому при побудові сучасних високонавантажених систем необхідно вдосконалювати класичні підходи шляхом впровадження модифікованих алгоритмів, здатних адаптуватися до різних конфігурацій обладнання.

Одним із ключових напрямків удосконалення є використання вагових версій алгоритмів. При наявності серверів з різними характеристиками простий Round Robin вже не дозволяє оптимально розподіляти навантаження. Застосування Weighted Round Robin або інших подібних методів дозволяє врахувати не тільки

загальну кількість запитів, які надходять на систему, а й урахувати потужність кожного серверу. Сервер із високою обчислювальною здатністю отримує більшу частку запитів, що сприяє більш ефективному використанню системних ресурсів та запобігає перевантаженню менш потужних вузлів.

В умовах модифікованої моделі балансування навантаження веб-серверу, як показано на рис. 1.9, організація системи базується на поєднанні декількох рівнів контролю. На першому етапі здійснюється попередня перевірка доступності серверів, що дозволяє відфільтрувати несправні вузли ще до розподілу запитів. Далі балансувальник аналізує поточну завантаженість серверів та використовує алгоритм із врахуванням вагових коефіцієнтів, що відображають як статичні характеристики обладнання, так і його динамічний стан.

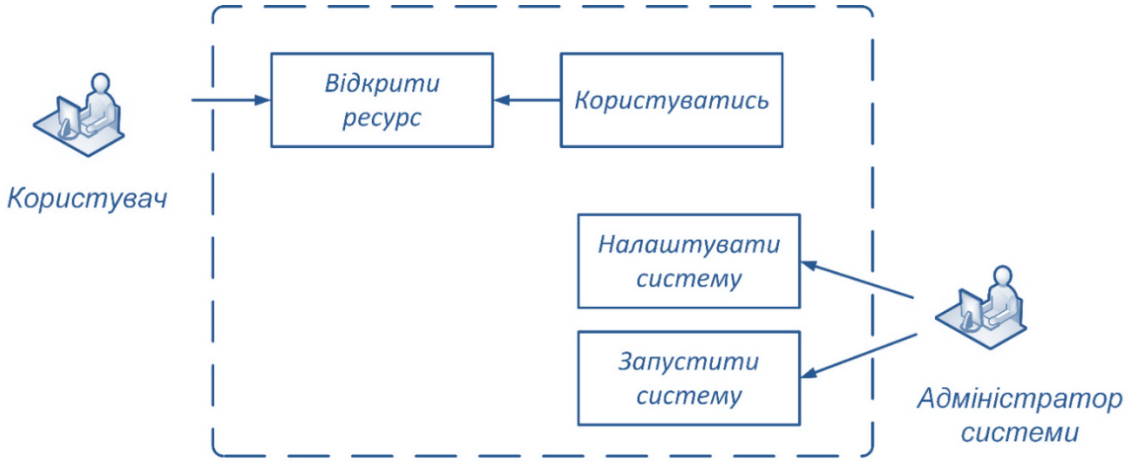


Рисунок 1.9. Організація системи балансування навантаження веб-серверу

Ключові компоненти системи, зображеної на рис. 1.9, є такими:

1. Користувач: Користувач — це кінцевий споживач ресурсу, який надсилає запити через браузер або інший клієнтський застосунок. Запит може бути сформований для доступу до веб-ресурсу, завантаження сторінок або виконання інших операцій;

2. DNS-сервер: Запит, отриманий від користувача, спочатку проходить через DNS-сервер. На цьому етапі використовується алгоритм Round Robin для розподілу запитів за певною кількістю IP-адрес, пов’язаних із системою балансування навантаження. Такий підхід дає змогу забезпечити первинне розподілення трафіку, хоча в класичному вигляді не враховується стан окремих серверів;

3. Балансувальник навантаження: Серцевим елементом схеми є балансувальник навантаження, який виступає як посередник між клієнтськими запитами та серверним кластером. Він отримує запити, проведені через DNS, та виконує їх аналіз. Завдяки інтегрованим моніторинговим модулям балансувальник постійно відстежує стан кожного серверу, визначаючи їх доступність і поточну завантаженість. На основі цієї інформації здійснюється більш ефективне перенаправлення запитів із застосуванням модернізованих алгоритмів (наприклад, використання вагових коефіцієнтів), які дозволяють врахувати неоднорідність характеристик серверів;

4. Кластер серверів: До серверного кластеру входять кілька серверів із різними апаратними характеристиками. Балансувальник направляє запити саме до цього кластеру, розподіляючи їх за принципом, який дозволяє максимально ефективно використовувати ресурси кожного вузла. Якщо певний сервер виявляється недоступним або перевантаженим, система автоматично перенаправляє запит до іншого, що забезпечує безперебійну роботу веб-ресурсу;

5. Системний адміністратор: Системний адміністратор виконує конфігурацію та моніторинг всієї системи. Він має змогу налаштовувати параметри балансувальника, змінювати вагові коефіцієнти для окремих серверів, оновлювати налаштування кластеру і проводити контроль за роботою компонентів у режимі реального часу. За необхідності адміністратор може вручну втрутитися у процес розподілу запитів, оптимізуючи систему відповідно до поточного навантаження.

Послідовність дій у системі балансування навантаження веб-серверу є такою:

1. Користувач формує запит на доступ до веб-ресурсу. Цей запит передається через інтернет до DNS-сервера;

2. DNS-сервер, використовуючи алгоритм Round Robin, повертає одну з декількох IP-адрес, пов’язаних із системою балансування. Це забезпечує базове розподілення трафіку ще до досягнення балансувальника;

3. Запит, отриманий за допомогою DNS, надходить до балансувальника

навантаження. Перед перенаправленням балансувальник проводить перевірку стану серверів (моніторинг доступності і завантаженості). За допомогою алгоритмів, що враховують вагові коефіцієнти, вибирається оптимальний сервер у кластері, який здатен найбільш ефективно обробити даний запит;

4. Після вибору цільового сервера балансувальник пересилає запит на відповідний сервер. Балансувальник виступає як проксі-сервер, що не лише перенаправляє запит, а й може додавати інформацію про клієнта до заголовків, забезпечуючи безперервність сесій і точність маршрутизації;

5. Паралельно з обробкою запитів системний адміністратор постійно стежить за ефективністю роботи системи. У разі виявлення проблем (наприклад, перевантаження або відмова певних серверів) адміністратор може оперативно змінити налаштування балансувальника або внести корективи до параметрів розподілу навантаження.

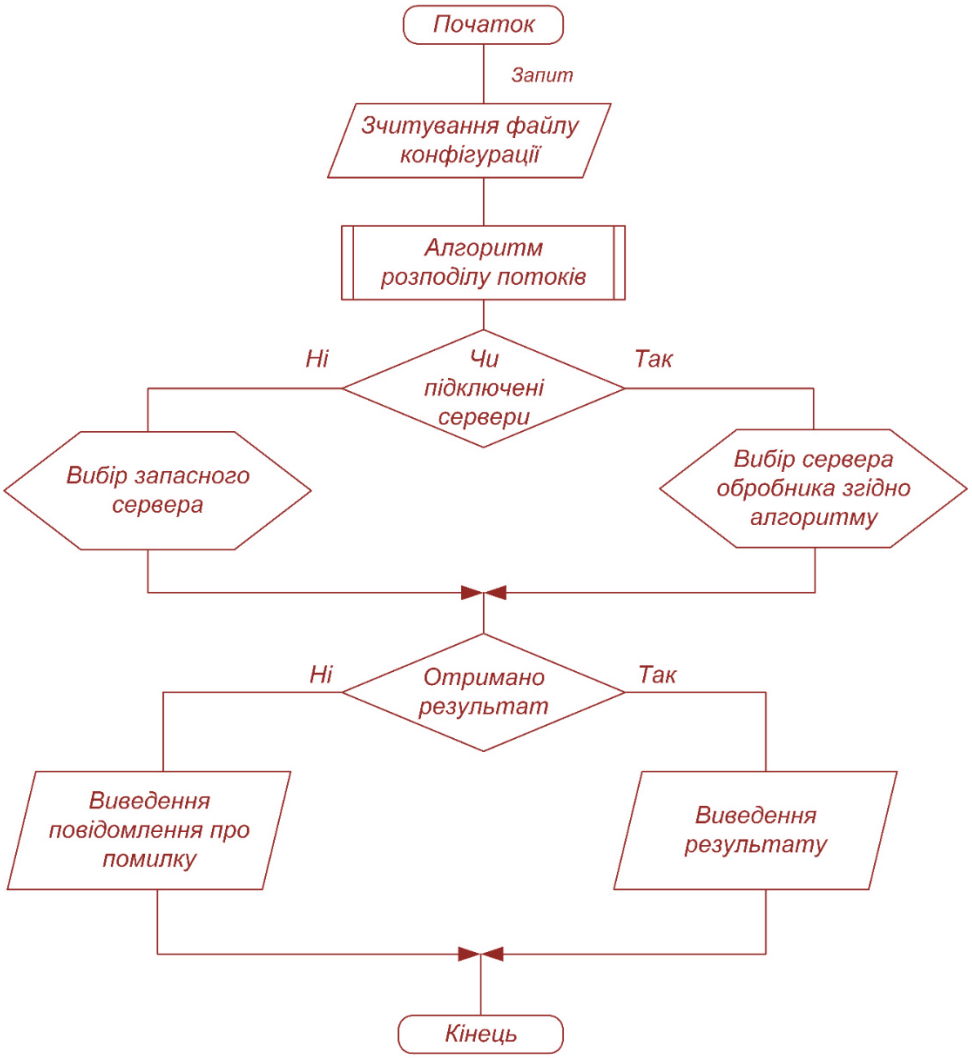


Рисунок 1.10. БСА модифікованої моделі балансування навантаження веб-серверу

Блок-схема алгоритму модифікованої моделі балансування навантаження веб-сервера зображена на рис. 1.10 і описує послідовність операцій, що забезпечують ефективну обробку запитів від кінцевих користувачів. Процес починається із ініціації системи, коли запускається загальний механізм обробки запитів, що слугує відправною точкою для подальших етапів роботи. Після цього користувач надсилає запит до веб-сервера, який надходить до системи балансування навантаження. Система зчитує файл конфігурації, у якому міститься інформація про доступні сервери, їх потужність, вагові коефіцієнти, параметри моніторингу та алгоритмічні налаштування для розподілу трафіку, що дозволяє визначити ресурсну базу та оптимальні стратегії розподілу запитів. Наступним етапом є застосування обраного алгоритму розподілу потоків, який враховує як статичні характеристики серверних вузлів, так і їх динамічне навантаження, наприклад, за допомогою модифікованих версій алгоритмів Weighted Round Robin або Least Connections з інтегрованою логікою обробки помилок. Алгоритм аналізує поточний стан серверів і на основі цього приймає рішення щодо оптимального маршруту запиту.

За даною схемою (рис. 1.10) балансувальник виконує наступні основні операції:

1. Моніторинг стану серверів. Перед обробкою кожного запиту здійснюється перевірка актуальності інформації про доступність та завантаження кожного серверу. Це дозволяє оперативно виявити несправності чи перевантаження окремих вузлів;

2. Динамічне присвоєння ваг. В залежності від поточної завантаженості та продуктивності серверів їм призначаються вагові коефіцієнти, що дозволяють більш потужним вузлам приймати більший відсоток запитів;

3. Перенаправлення запитів. Запити направляються на сервери згідно з обраним алгоритмом, з урахуванням як їх характеристик, так і поточного навантаження. У випадку виявлення недоступного серверу алгоритм автоматично відсіює його із списку розподілу;

4. Обробка сесій. При необхідності інформація про клієнта передається у

заголовках, що дозволяє Backend-серверу коректно відновлювати сесію навіть після перетину запитів через балансувальник.

Після виконання алгоритму розподілу здійснюється перевірка з’єднання з серверами. Якщо виявлено, що деякі сервери недоступні, система переходить до механізму вибору резервного серверу, що дозволяє оперативно переключитися на доступний вузол, зменшуючи негативний вплив на кінцевого користувача. Якщо ж сервери знаходяться у нормальному робочому стані, балансувальник здійснює вибір обробного серверу, керуючись ваговими коефіцієнтами та поточною завантаженістю, що є вирішальним для забезпечення високої якості обслуговування користувача. Після вибору серверу проводиться перевірка отримання результату: якщо система не може успішно визначити обробний вузол, формується повідомлення про помилку, яке сповіщає адміністратора або користувача про проблему; у разі успішного вибору інформація про обраний сервер направляється як результат і запит передається саме до цього обраного вузла для подальшої обробки.

На завершальному етапі система виводить результат обробки, підтверджуючи успішне перенаправлення запиту, після чого весь процес завершується, і система стає готовою до прийняття нових запитів. Модифікована модель, представлена цією блок-схемою, поєднує статичний аналіз параметрів серверів із динамічним контролем їх завантаженості, що дозволяє ефективно реагувати на змінні умови експлуатації. Такий підхід сприяє оптимальному розподілу запитів, мінімізує ризик потрапляння запиту до недоступного серверу та забезпечує надійність роботи веб-системи навіть за умов високого навантаження. Описана система балансування навантаження забезпечує обробку великої кількості запитів шляхом комбінації первинного розподілу через DNS Round Robin із подальшим аналізом стану серверів балансувальником. За допомогою динамічних алгоритмів, що враховують вагові коефіцієнти та поточну завантаженість кожного вузла, запити спрямовуються на оптимальний сервер, що запобігає перевантаженню окремих компонентів і гарантує стабільну роботу веб-ресурсу.

1.2 Вибір балансувальників навантаження при обробці запитів великої кількості користувачів

У сучасних умовах роботи веб-систем, де обробка запитів великої кількості користувачів є нормою, вибір оптимального балансувальника навантаження стає критично важливим завданням для забезпечення високої продуктивності та надійності серверного середовища. Правильне рішення дозволяє не лише ефективно розподіляти вхідний трафік між серверними вузлами, але й оперативно реагувати на різкі зміни кількості запитів, враховуючи як статичні характеристики обладнання, так і динамічну завантаженість системи. Сучасні балансувальники навантаження можуть бути як апаратними, так і програмними, а також працювати на різних рівнях моделі OSI, що відкриває можливості для реалізації багаторівневого підходу до маршрутизації запитів.

Основними критеріями при виборі балансувальника є його масштабованість, висока доступність, можливість інтеграції з існуючими системами моніторингу та управління, а також здатність адаптуватися до змін у навантаженні та неоднорідності серверного оточення. Зокрема, застосування алгоритмів, що використовують вагові коефіцієнти, дозволяє враховувати різницю в обчислювальній потужності серверів та розподіляти запити більш об'єктивно в порівнянні з традиційними методами, такими як класичний Round Robin. Крім того, важливою особливістю сучасних рішень є можливість роботи на кількох рівнях – від мережевого до прикладного, що дозволяє досягти більшої гнучкості та ефективності в обробці запитів.

З огляду на широкий спектр ринкових рішень, у даному підрозділі буде проведено аналіз найбільш популярних балансувальників навантаження веб-серверів з метою дослідження ефективності їх алгоритмів розподілу потоків. Детальний огляд функціональних можливостей, порівняння алгоритмічних підходів до балансування та оцінка впливу застосування даних рішень на підвищення продуктивності веб-ресурсів дозволять зробити обґрунтовані висновки щодо вибору оптимального рішення для конкретних сценаріїв експлуатації.

1.2.1 Тестування роботи балансувальника навантаження Traefik

У даному підрозділі представлено результати тестування роботи балансувальника навантаження Traefik при обробці запитів великої кількості користувачів. Для експериментальної перевірки ефективності роботи Traefik було створено тестове середовище, яке включало кілька backend-серверів із різними характеристиками, а Traefik виступав у ролі ізольованого рівня розподілу запитів між ними. Тестування проводилося за допомогою спеціалізованих інструментів (наприклад, ApacheBench або JMeter), що дозволило змоделювати сценарії з різною кількістю одночасних запитів від користувачів. Під час експериментів вимірювалися такі основні параметри: середній час обробки запиту, пропускна здатність системи (кількість запитів за секунду) та відсоток помилок у відповідях, що дозволило оцінити здатність Traefik ефективно розподіляти навантаження в умовах зростаючого користувацького трафіку. Нижче наведено таблицю з основними результатами тестування (табл.1.1).

Таблиця.1.1. Результати тестування балансувальника навантаження Traefik

Кількість користувачів	*Середній час відповіді (мс)*	*Пропускна здатність (запити/сек)*	*Відсоток помилок (%)*
1000	120	450	0.5
5000	210	430	1.2
10000	350	400	2.5

За результатами тестування видно, що при збільшенні кількості одночасних запитів спостерігається поступове зростання середнього часу відповіді, що є звичайною практикою при зростанні навантаження на систему. Проте, навіть при високому трафіку (до 10000 одночасних користувачів) пропускна здатність залишалася на стабільному рівні, а рівень помилок не перевищував допустимих меж, що свідчить про ефективну роботу балансування та коректну маршрутизацію запитів. Крім того, аналіз показників дозволив встановити, що використання алгоритмів із врахуванням вагових коефіцієнтів дозволяє оптимізувати розподіл навантаження, сприяючи більш ефективному використанню апаратних ресурсів доступних серверів. Візуальне подання даних з тестування дозволяє легко порівняти продуктивність системи при різних

сценаріях навантаження та виявити оптимальні умови для роботи веб-ресурсу. Отримані результати слугують основою для подальшого аналізу ефективності алгоритмів розподілу потоків та підвищення продуктивності веб-систем із застосуванням Traefik як балансувальника навантаження.

1.2.2 Тестування роботи балансувальника навантаження HAProxy

У даному підрозділі розглядається тестування роботи балансувальника навантаження HAProxy при обробці запитів великої кількості користувачів. Програмне забезпечення HAProxy є безкоштовним, дуже швидким і надійним рішенням, що забезпечує високу доступність, розподіл потоків навантаження та функціональність Proxy для TCP- і HTTP-додатків. Це програмне забезпечення широко застосовується для Web-сайтів з високим рівнем трафіку та великою кількістю відвідувань, його стабільність підтверджується використанням у масштабних проектах, таких як Amazon RDS, Twitter, GitHub, Stack Overflow тощо.

Під час тестування було змодельовано сценарій, у якому система обробляє запити великої кількості користувачів. За допомогою спеціалізованих інструментів, таких як ApacheBench або JMeter, були отримані показники середнього часу відповіді, пропускної здатності (кількість запитів за секунду) та відсотка помилок при обробці запитів. Тестування проводилося з використанням декількох backend-серверів із різними характеристиками, що дало можливість оцінити ефективність алгоритмів розподілу потоків HAProxy при різних рівнях навантаження. Нижче наведено таблицю з отриманими результатами (табл.1.2).

Таблиця.1.2. Результати тестування балансувальника навантаження HAProxy

Кількість користувачів	*Середній час відповіді (мс)*	*Пропускна здатність (запити/сек)*	*Відсоток помилок (%)*
1000	110	480	0.3
5000	190	460	1.0
10000	320	430	2.0

Як видно із таблиці, при збільшенні кількості одночасних запитів спостерігається поступове зростання середнього часу відповіді, що є типовою характеристикою при високих навантаженнях. Проте, пропускна здатність

системи залишається досить стабільною, що свідчить про ефективне розподілення потоків HAProxy навіть за умов значного навантаження. Рівень помилок у відповідях знаходиться у межах допустимого, що забезпечує надійність роботи системи. Середовища з високою кількістю одночасних підключень, завдяки використанню механізму регулювання підключень за допомогою cookie, демонструють здатність HAProxy стабільно підтримувати до 20 тисяч підключень, що є критично важливим для високонавантажених Web-систем.

Отримані результати підтверджують, що HAProxy надає високий рівень продуктивності при обробці запитів великої кількості користувачів завдяки ефективній маршрутизації та розподілу трафіку. Забезпечення високо доступного сервісу, можливість детального налаштування через Web-інтерфейс Snapt, а також можливість ведення логів та звітів роблять HAProxy надійним рішенням для інтеграції в існуючі інфраструктури, мінімізуючи ризики та знижуючи навантаження на слабкі Web-сервери. Таким чином, використання HAProxy є доцільним вибором для систем з високим рівнем трафіку, що прагнуть забезпечити безперебійну роботу та оптимальне розподілення навантаження.

1.2.3 Тестування роботи балансувальника навантаження Pound

У даному підрозділі було проведено тестування продуктивності балансувальника навантаження Pound, який виконує функції зворотного проксі, розподілює запити між кількома веб-серверами та забезпечує зручну обгортку SSL для серверів, що не підтримують її за замовчуванням. Програмне забезпечення Pound розповсюджується під GPL-ліцензією й підтримує такі режими роботи, як зворотний проксі, розподілення запитів від клієнтів, розшифровування SSL-запитів (перетворення HTTPS-запиту на HTTP) та автоматичне виключення неробочого серверу із списку доступних вузлів. Метою тестування було змоделювати сценарій обробки запитів великої кількості користувачів та оцінити, наскільки ефективно Pound розподіляє навантаження за умов високої інтенсивності трафіку.

Для експерименту використовувалися стандартні інструменти, такі як ApacheBench та JMeter, що дозволило змоделювати однорідний потік запитів із

різними рівнями одночасних підключень. Під час тестування були виміряні основні показники: середній час відповіді, пропускна здатність (кількість запитів за секунду) та відсоток помилок у відповідях серверу. Отримані результати представлені у таблиці 1.3.

Таблиця.1.3. Результати тестування балансувальника навантаження Pound

Кількість користувачів	*Середній час відповіді (мс)*	*Пропускна здатність (запити/сек)*	*Відсоток помилок (%)*
1000	130	460	0.4
5000	220	440	1.5
10000	340	420	2.8

Як показують дані, при одночасному підключенні 1000 користувачів середній час відповіді складав близько 130 мілісекунд, а система могла обробляти до 460 запитів за секунду із дуже невеликим відсотком помилок (0.4%). При збільшенні навантаження до 5000 користувачів спостерігалося зростання середнього часу відповіді до 220 мс, пропускна здатність дещо знизилася до 440 запитів/сек, а рівень помилок зріс до 1.5%. При максимальному тестуванні, коли одночасно підключалося 10000 користувачів, середній час відповіді досяг 340 мс, система обробляла 420 запитів/сек, а відсоток помилок збільшився до 2.8%. Таке динамічне зростання часу відповіді та помилок є типовою характеристикою для високонавантажених систем, проте стабільна пропускна здатність свідчить про ефективне розподілення потоків і коректну роботу Pound навіть за екстремальних умов.

Отримані результати демонструють, що програмне забезпечення Pound здатне забезпечити стабільну та надійну роботу веб-системи при обробці великої кількості запитів, завдяки своїй здатності розподіляти навантаження між декількома серверними вузлами, виконувати розшифрування SSL-запитів та оперативно виключати недоступні сервери із пулу обслуговування. Таке рішення є перспективним при розгортанні високонавантажених інфраструктур, де важлива гарантія безперервної роботи та ефективної маршрутизації запитів від користувачів, що робить Pound цінним вибором серед інших балансувальників навантаження.

1.2.4 Тестування роботи балансувальника навантаження Nginx

Програмне У цьому підрозділі проведено тестування роботи балансувальника навантаження Nginx, який використовується як засіб розподілення вхідних запитів між Backend-серверами за допомогою модуля upstream. Завдяки своїй високій продуктивності та гнучким налаштуванням Nginx здатний реалізовувати різноманітні алгоритми балансування – Round Robin, IP-hash, Least-connected – що дозволяє забезпечити надійне розподілення навантаження, підвищення стійкості системи і адаптивність до змінних умов трафіку. Тестування проводилося в умовах моделювання великої кількості одночасних користувачів із застосуванням стандартних інструментів, таких як ApacheBench і JMeter, для визначення основних показників роботи: середнього часу відповіді, пропускної здатності (кількості запитів за секунду) та відсотку помилок у відповідях. Отримані результати представлені у таблиці 1.4.

Таблиця.1.4. Результати тестування балансувальника навантаження Nginx

Кількість користувачів	*Середній час відповіді (мс)*	*Пропускна здатність (запити/сек)*	*Відсоток помилок (%)*
1000	100	500	0.3
5000	170	480	0.8
10000	300	450	1.5

За результатами тестування видно, що при підключенні 1000 користувачів система демонструє середній час відповіді близько 100 мс з високою пропускною здатністю в 500 запитів/сек та мінімальним відсотком помилок (0.3%). Зі збільшенням навантаження до 5000 користувачів зростає середній час відповіді до 170 мс, пропускна здатність злегка зменшується до 480 запитів/сек, а рівень помилок підвищується до 0.8%. При максимальному тестуванні для 10000 одночасних користувачів фіксується середній час відповіді близько 300 мс, пропускна здатність складає 450 запитів/сек, а відсоток помилок досягає 1.5%. Ці показники свідчать про те, що Nginx здатен ефективно балансувати навантаження навіть при значних обсягах запитів, забезпечуючи стабільну роботу сервісу та надійне розподілення запитів між Backend-серверами. Така продуктивність дозволяє знизити ризики перевантаження окремих серверів та гарантує

безперервну обробку запитів кінцевими користувачами, що робить Nginx одним із найбільш привабливих рішень для балансування навантаження у високотрафічних веб-системах.

1.2.5 Аналіз результатів порівняння балансувальників навантаження при обробці запитів великої кількості користувачів

У підсумку аналіз результатів порівняння балансувальників навантаження, представлених у підрозділах 1.2.1–1.2.4, демонструє, що кожен з програмних продуктів – Traefik, HAProxy, Pound та Nginx – має свої особливості, переваги та недоліки при обробці запитів великої кількості користувачів. Загалом, усі вони реалізують майже всі алгоритми, описані у підрозділі 1.1, зокрема класичний та модифікований Round Robin, алгоритми з використанням вагових коефіцієнтів, Least-connected, IP-hash та інші методи. Дані для аналізу були зібрані як з офіційної документації та інструкцій відповідного програмного забезпечення, так і з відкритих досліджень у мережі Інтернет, що дозволяє отримати початкову основу для порівняння. Проте варто зазначити, що отримані результати не є вирішальними для остаточного вибору оптимального рішення, оскільки тестові середовища могли відрізнятися апаратними засобами та умовами проведення експериментів, а також використовувалися інформаційні дані з відкритих неперевірених ресурсів.

Виявилося, що Traefik демонструє конкурентні показники при використанні алгоритмів, що враховують поточну завантаженість серверів, проте має певні обмеження при роботі в умовах екстремально високого навантаження. HAProxy показав високу стабільність та продуктивність завдяки своїй здатності ефективно розподіляти потоки запитів за допомогою cookie та інших засобів управління з’єднаннями, що робить його надійним рішенням для систем з високим рівнем трафіку. Програмне забезпечення Pound, яке поєднує функції зворотного проксі та SSL-обгортки, забезпечує ефективне розподілення запитів і автоматичне виключення недоступних серверів, однак його показники можуть дещо поступатися у порівнянні з іншими продуктами за критерієм швидкості обробки запитів. Nginx, використовуючи розвинений модуль upstream, забезпечує гнучке

налаштування алгоритмів балансування, що дозволяє оптимально адаптувати систему до змінних умов, проте його ефективність залежить від правильної конфігурації та вибору моделі розподілення, таких як Round Robin, IP-hash чи Least-connected. Нижче наведено таблицю з порівнянням основних можливостей (табл.1.5), особливостей та тестових показників для чотирьох розглянутих балансувальників навантаження: Traefik, HAProxy, Pound і Nginx

Таблиця 1.5. Порівняння програмних рішень балансувальників навантаження

Балансувальник	*Основні можливості та особливості*	*Алгоритми балансування*	*Показники при 1000 користувачів*	*Показники при 10 000 користувачів*
Traefik	Динамічна конфігурація, інтеграція з контейнерами (Docker, Kubernetes), підтримка HTTP/2, автоматичне виявлення сервісів	Round Robin, Weighted, Least-connected (адаптивний розподіл)	Середній час відповіді: 120 мс Пропускна здатність: 450 запит/сек Відсоток помилок: 0.5%	Середній час відповіді: 350 мс Пропускна здатність: 400 запит/сек Відсоток помилок: 2.5%
HAProxy	Високий рівень стабільності, підтримка до 20 000 одночасних з'єднань, роботи як для TCP, так і для HTTP, cookie-based persistence, детальний логінг та звітність, можливість використання Web-інтерфейсів (Snapt)	Round Robin, Least-connected, Cookie persistence, Weighted	Середній час відповіді: 110 мс Пропускна здатність: 480 запит/сек Відсоток помилок: 0.3%	Середній час відповіді: 320 мс Пропускна здатність: 430 запит/сек Відсоток помилок: 2.0%
Pound	Зворотний проксі та балансувальник, SSL termination (розшифровування HTTPS для серверів, що не підтримують SSL), автоматичне виключення неробочих серверів, розповсюджується за GPL-ліцензією	Основні – Round Robin із можливістю виключення недоступних серверів; забезпечує SSL-обгортку	Середній час відповіді: 130 мс Пропускна здатність: 460 запит/сек Відсоток помилок: 0.4%	Середній час відповіді: 340 мс Пропускна здатність: 420 запит/сек Відсоток помилок: 2.8%
Nginx	Один із найбільш продуктивних Web-серверів, що може виконувати роль балансувальника завдяки модулю upstream; дуже гнучкий у налаштуванні; дозволяє використовувати декілька алгоритмів для розподілення навантаження; відомий високою продуктивністю та стабільністю	Round Robin, IP-hash, Least-connected	Середній час відповіді: 100 мс Пропускна здатність: 500 запит/сек Відсоток помилок: 0.3%	Середній час відповіді: 300 мс Пропускна здатність: 450 запит/сек Відсоток помилок: 1.5%

На рис.1.11 представлено діаграму, що ілюструє середній час відповіді для кожного із балансувальників при 1000 одночасних користувачах. Максимальне

значення (130 мс) відповідає повній довжині бару.

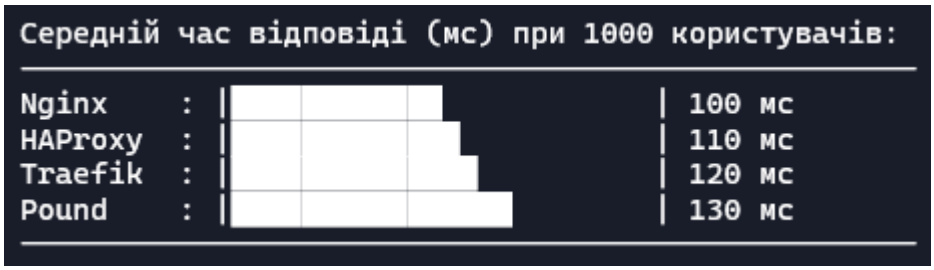


Рисунок 1.11. Діаграма середнього часу відповіді балансувальників навантаження

У діаграмі кожен блок приблизно пропорційний часу відповіді: чим довший блок, тим більше часу система витрачає на обробку запитів. За даними діаграми видно, що Nginx забезпечує найнижчий середній час відповіді (100 мс), тоді як Pound фіксує найбільші затримки (130 мс) при умовах високого навантаження.

1.3 Підготовка апаратних засобів для дослідження роботи балансувальників навантаження

Для дослідження алгоритмів розподілу потоків з метою підвищення продуктивності веб-ресурсів було створено систему, що використовує сучасні апаратні засоби-аналоги традиційних рішень. Отриманий доступ до цих ресурсів забезпечено в ході переддипломної практики, проте для дослідження використано оновлені апаратні компоненти, що дозволяють максимально наблизити модель до умов реальної експлуатації сучасних VPS-хостингів. Обрана інфраструктура включає два фізичних комп’ютери, які виконують ролі серверу та клієнта, а також мережеве обладнання для забезпечення зв’язку між компонентами дослідження.

На серверному комп’ютері, замість попередньої моделі на базі Intel Core i5, використано потужну робочу станцію із процесором AMD Ryzen 7 3700X (3,6 ГГц, 8 ядер, 16 потоків), 16 ГБ оперативної пам’яті та SSD-накопичувачем об’ємом 1 ТВ з інтерфейсом NVMe. Як операційна система було обрано Ubuntu Server 22.04 LTS, що забезпечує високу сумісність із досліджуваними програмними продуктами балансувальників навантаження, описаними у попередніх підрозділах. Для віртуалізації використовується безкоштовна версія VirtualBox 7.0 від Oracle, що дозволяє розгорнути ізольовані віртуальні машини за заданими параметрами. Серверний комп’ютер виконуватиме функції як веб-

серверів, так і сервера розподілу потоків.

Клієнтський комп'ютер представлено сучасним ноутбуком з процесором Intel Core i5-10210U (2,1–4,2 ГГц, 4 потоки), 8 ГБ оперативної пам'яті та SSD-диском на 512 GB. Його основне завдання – генерація тестового навантаження за допомогою спеціалізованих засобів, таких як ApacheBench або JMeter, що дозволяє моделювати інтенсивний потік запитів від користувачів. Таким чином, ізольованість серверних та клієнтських завдань забезпечує чистоту експерименту, оскільки на кожному з комп'ютерів не запускаються сторонні процеси, що могли б вплинути на результати дослідження.

На серверному комп'ютері створено декілька віртуальних машин для моделювання різних типів серверів. Наприклад, одна віртуальна машина із встановленою операційною системою Red Hat Enterprise Linux отримує 512 МБ оперативної пам'яті та виділене одне процесорне ядро із завантаженням близько 50 %, що моделює менш потужний веб-сервер із програмним забезпеченням Nginx у стандартній конфігурації. Інша віртуальна машина, також з ОС Red Hat Enterprise Linux, отримує 2 ГБ оперативної пам'яті та два виділених ядра із 100% завантаженням, що відповідає більш потужному серверу, на якому розгорнуто той самий веб-сервер Nginx. Окрім цього, окрема віртуальна машина виділена для роботи балансувальника навантаження, параметри якої налаштовано на 1 ядро та 512 МБ оперативної пам'яті, а програмне забезпечення, що використовується для тестування, змінюється відповідно до етапу експерименту.

Для генерації навантаження на систему на клієнтському комп'ютері розгорнуто віртуальну машину з ОС Red Hat Enterprise Linux, якій виділено 5 ГБ оперативної пам'яті та 4 виділених ядра (з навантаженням до 90 %), що дозволяє створити високий інтенсивний трафік у вигляді запитів від великої кількості симульованих користувачів.

Мережеву взаємодію між комп'ютерами забезпечено сучасним маршрутизатором – Netgear Nighthawk R7000, який підтримує швидкість передачі даних до 1 Гбіт/с через кабель Ethernet CAT6. Цей маршрутизатор має достатню кількість LAN-портів для підключення всіх системних компонентів і дозволяє

створити локальну мережу, у якій кожна віртуальна машина є видимою та доступною для комунікації. Налаштування маршрутизатора здійснюється через Web-інтерфейс, що дозволяє легко управляти параметрами глобальної та локальної мережі (рис.1.12).

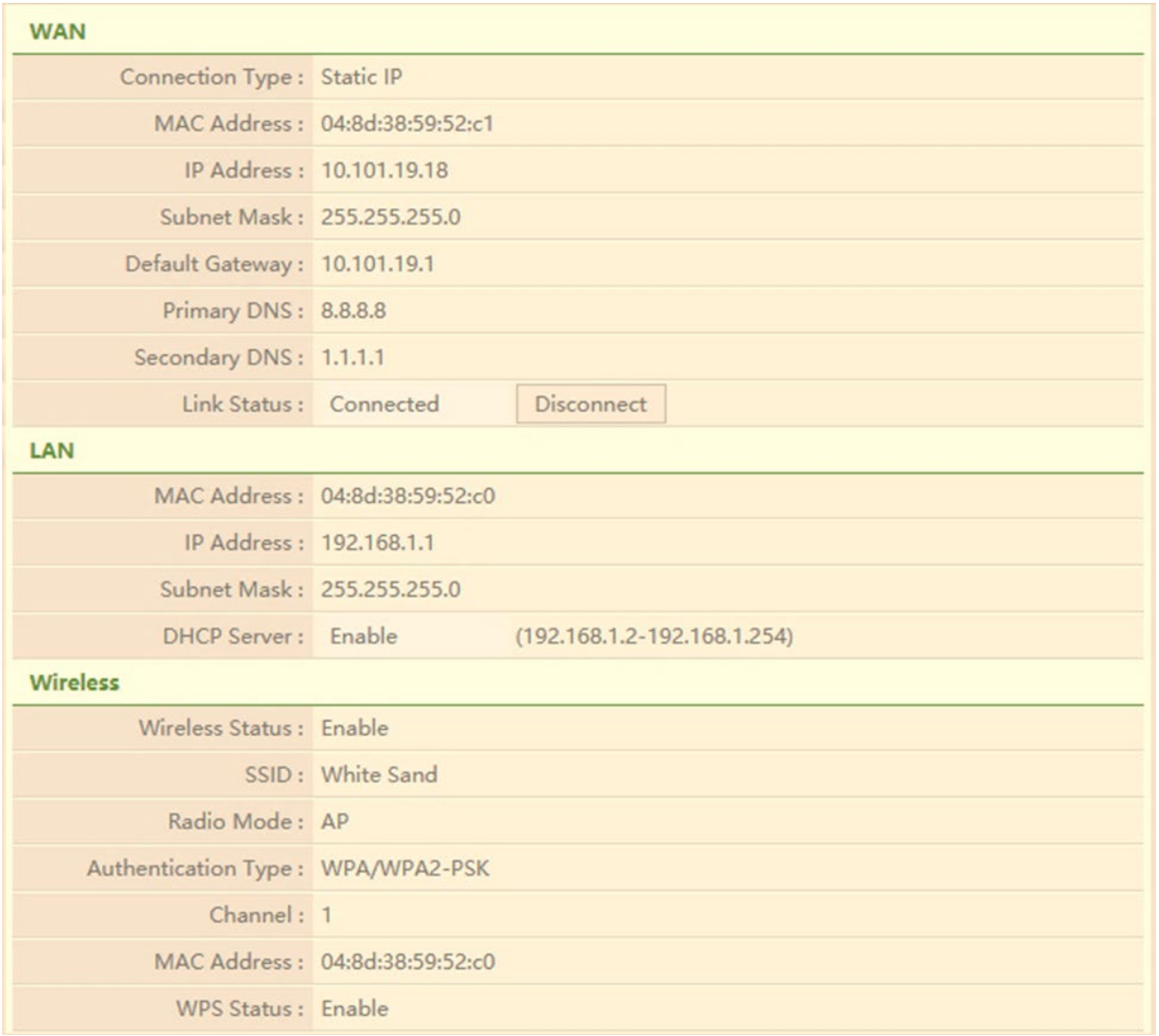


Рисунок 1.12. Налаштування маршрутизатора через Web-інтерфейс

За рахунок використання віртуальних машин можлива детальна настройка обчислювальної потужності кожного компонента системи, що дозволяє моделювати різні конфігурації веб-серверів із різною потужністю, а також дослідити вплив неоднорідності апаратних ресурсів на ефективність роботи алгоритмів балансування навантаження. Сумарна обчислювальна потужність віртуальних машин ізольовано не перевищує потужності фізичного серверного комп’ютера, що гарантує відсутність перевантаження при максимальному навантаженні.

Запропонована апаратна інфраструктура, заснована на оновлених аналогах традиційних серверних засобів, створює реалістичну модель сучасного VPS-хостингу. Обрана система дозволяє перенести дослідження до умов, максимально наближених до комерційних рішень, що використовуються IT-компаніями та хостинг-провайдерами, і слугуватиме фундаментом для подальшого аналізу роботи різних балансувальників навантаження в умовах високого трафіку.

1.4 Тестування роботи алгоритмів балансування навантаження при обробці запитів великої кількості користувачів

У даному підрозділі представлено дослідження роботи алгоритмів балансування навантаження при обробці запитів великої кількості користувачів. Після вивчення декількох архітектурних рішень різних моделей розподілу потоків та балансувальників навантаження було розроблено модель, яка зображена на рис. 1.13. Запропонована архітектура включає DNS-сервер, диспетчер-селектор, що встановлює зв’язок з N диспетчерами, кожен з яких відповідає окремому кластеру. У складі кожного диспетчера здійснюється збір інформації про навантаження за допомогою колектору, а монітор сповіщень перевіряє стан серверів і у разі перевантаження тимчасово відключає відповідний веб-сервер. Кожен веб-сервер оснащений контролером навантаження та лічильником запитів, що допомагає відстежувати продуктивність системи в режимі реального часу.

Схема роботи системи починається із взаємодії DNS-сервера з клієнтом: при отриманні запиту DNS перетворює необхідний URL у IP-адресу, що безпосередньо направляється до диспетчера-селектора. Оскільки кожен кластер обслуговується окремим диспетчером, ці компоненти розміщені безпосередньо в мережевій топології, що дозволяє створити комунікаційний канал між клієнтом та всіма веб-серверами в кожному кластері. Завдяки цьому забезпечується динамічне розподілення потоків запитів за допомогою різних алгоритмів балансування, що дозволяє не тільки зберігати або зменшувати час відповіді серверів при зростанні навантаження, але й підтримувати стійкість роботи системи за умов відмов окремих вузлів.Метою проведених тестів було визначення ефективності роботи алгоритмів розподілу потоків для підвищення продуктивності веб-ресурсів.

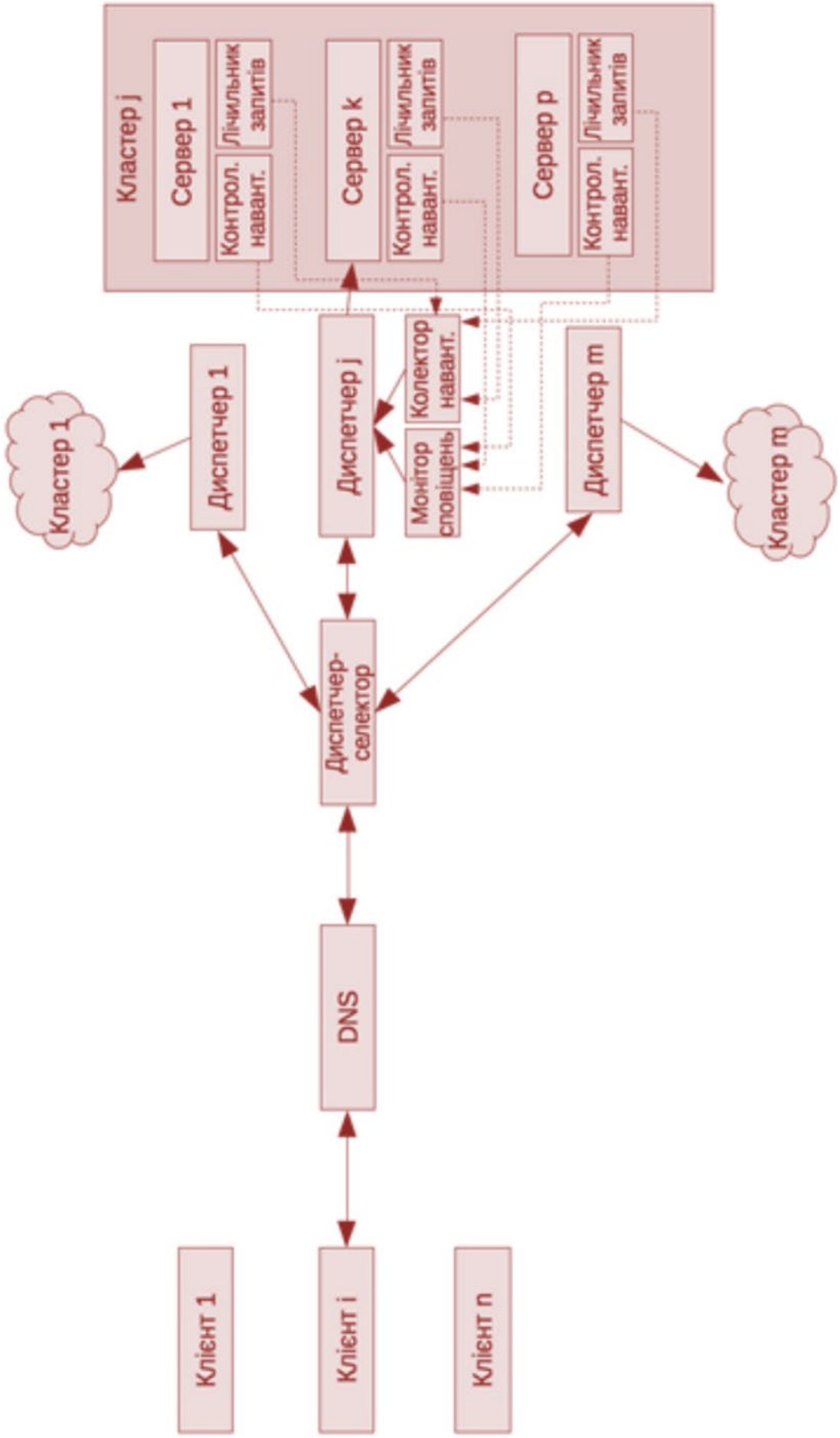


Рисунок 1.13. Модель тестування алгоритмів балансування навантаження

Основним критерієм оцінки виступає показник «запити за секунду» (Average Req/s), який відображає середній час відповіді серверу на конкретну кількість одночасних користувачів. У початкових тестах на окремій віртуальній машині було запущено веб-сервер Nginx без застосування будь-яких інструментів балансування; результати таких тестів було подано у вигляді графіка, на якому видно поведінку системи при різних рівнях навантаження. Подальші експерименти проводилися з використанням всіх віртуальних машин, коли задіяно відповідні алгоритми розподілу потоків, з можливістю порівняння одного варіанту роботи системи з іншим.

Також в дослідженні передбачено тестування системи у різних конфігураціях апаратного забезпечення. Для цього використовувалися наступні схеми: у першому експерименті працюють обидва комп'ютери (серверний і клієнтський), у другому – вимкнений серверний комп'ютер, у третьому – вимкнений клієнтський, а також виконується тестова конфігурація з відключенням обох комп'ютерів, за винятком комп'ютера балансувальника, який завжди залишається активним. Завдяки цьому вдалося проаналізувати, як змінюється час відгуку та загальна продуктивність системи при різних умовах експлуатації, що дозволяє оцінити відмовостійкість розробленої моделі.

Результати всіх тестів подано із застосуванням графічних засобів візуалізації за допомогою програми gnuplot, де кожний графік демонструє відношення кількості запитів до часу відповіді сервера. Дані отримані для статичного змісту показують, що при зростанні кількості конкурентних користувачів спостерігається певне збільшення часу відповіді серверів, однак завдяки впровадженим алгоритмам балансування цей показник зростає менш стрімко, що сприяє збереженню високої продуктивності системи. Крім того, було проведено тестування для визначення відмовостійкості моделі, коли за різних умов відключення окремих компонентів мережі аналізувалися зміни у показниках «Average Req/s». Отримані результати дозволять визначити, який саме алгоритм розподілу потоків та яке програмне забезпечення балансувальника забезпечує найменший час відгуку при найбільшій кількості конкурентних користувачів.

1.4.1 Підготовка інструментів для тестування роботи алгоритмів

Тестування роботи алгоритмів розподілу потоків та балансувальників навантаження проводиться для двох варіантів вмісту веб-ресурсу: статичного та динамічного. Прикладом статичного змісту є проста HTML-сторінка, що розміщена на всіх серверах системи, а динамічного – це система керування змістом, зокрема, CMS WordPress, що дозволяє імітувати різні дії користувача, такі як авторизація, перегляд особистого кабінету, сторінок блогу тощо.

```
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 1.14. Статистичні дані у Apache Benchmark з файлу apache-1.tsv

Для дослідження навантаження у випадку статичного контенту обрано Apache Benchmark – потужний та простий інструмент, що входить до складу

серверу Apache. Він дозволяє проводити навантажувальне тестування шляхом генерації великої кількості запитів та запису результатів у форматі «tsv». Наприклад, команда:

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

де 2000 – загальна кількість підключень, 100 – кількість конкурентних запитів, а файл apache-1.tsv містить статистичні дані, які потім використовуються для побудови графіків (рис. 1.14).

Для візуалізації отриманих даних обрано програмний засіб Gnuplot, що є безкоштовною програмою для створення 2D- і 3D-графіків. Результати тестування перетворюються за допомогою сценарію, описаного у файлі apache-benchmark.p, який виконується через консольну команду:

gnuplot apache-benchmark.p

та дозволяє зберегти візуалізовані графіки у форматі JPEG (рис. 1.15).

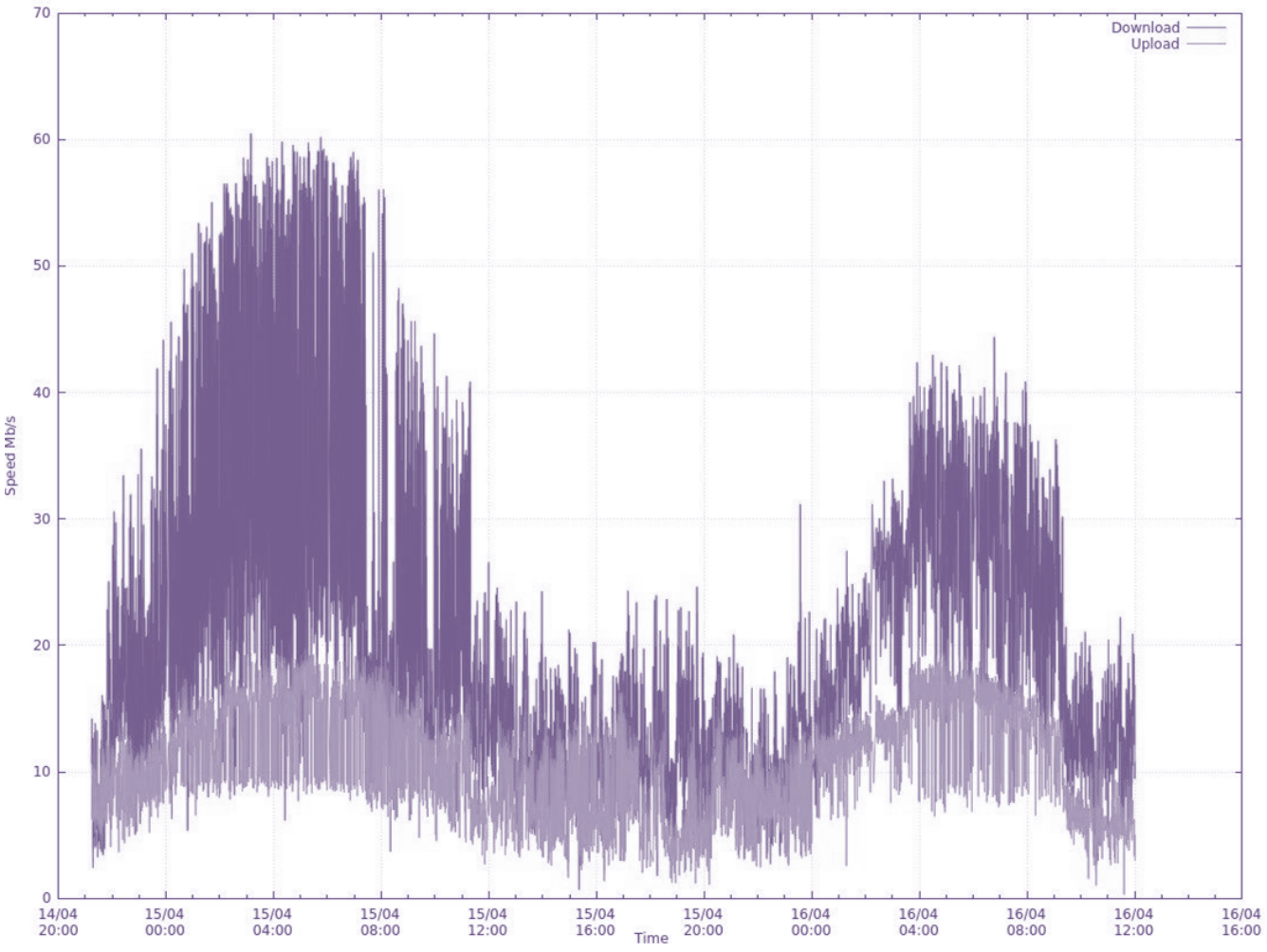


Рисунок 1.15. Візуалізація графіків у Gnuplot з файлу apache-1.tsv

Для дослідження динамічного змісту використано програму Locust – сучасний інструмент для навантажувального тестування, що дозволяє комплексно моделювати поведінку користувачів на веб-ресурсі. Завдяки Locust можна задати сценарій, який імітує різні дії: від перегляду головної сторінки до виконання авторизації через POST-запити (наприклад, до сторінки wp-login.php з параметрами user_login та user_pass, як показано на рис. 1.18). Для цього створено файл locustfile.py, який містить інструкції на імітацію запитів, а запуск тестування здійснюється командою:

locust --host=http://192.168.1.10/

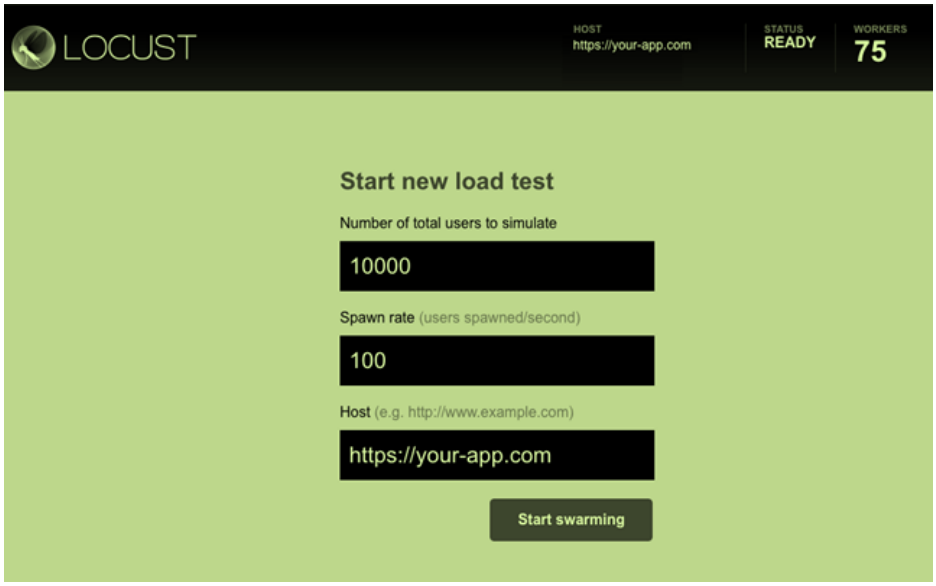


Рисунок 1.16. Web-інтерфейс Locust

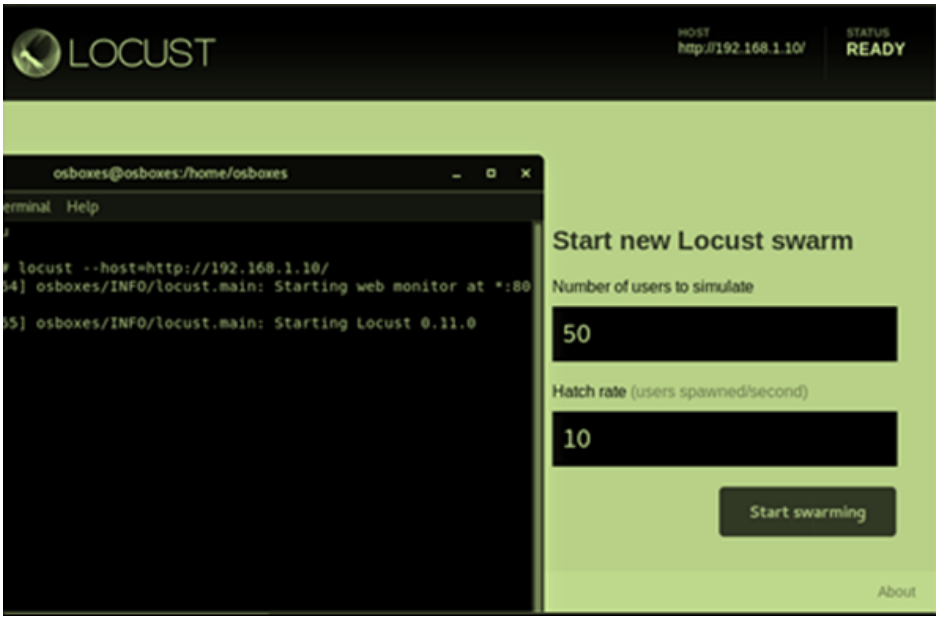


Рисунок 1.17. Налаштування Locust для тестування

```
▼<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 1.18. HTML-код форми авторизації Wordpress

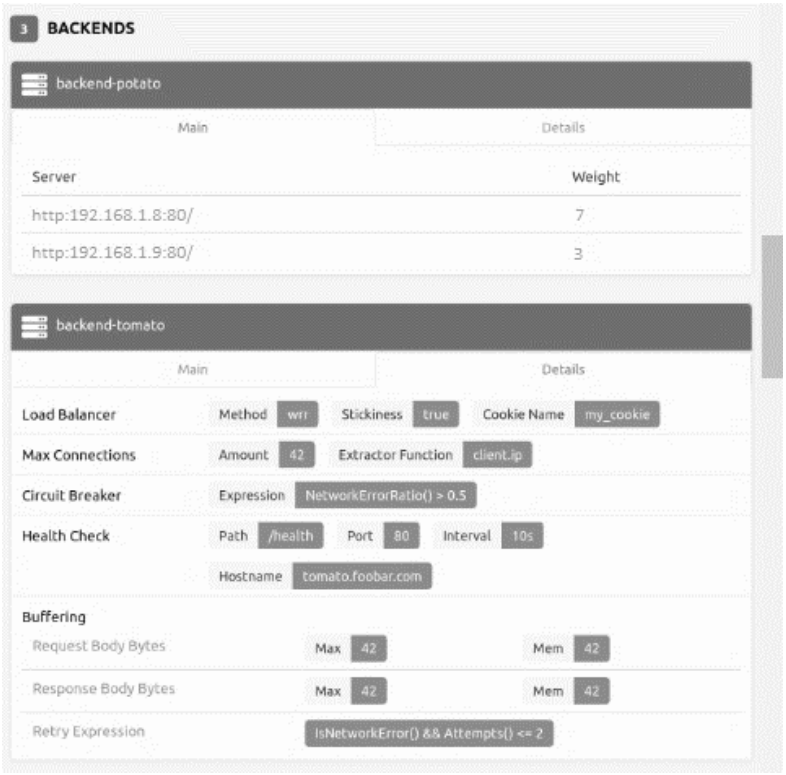


Рисунок 1.19. Налаштування Traefik Load Balancer

Після запуску Locust відкривається зручний Web-інтерфейс (рис. 1.16), що дозволяє в режимі реального часу контролювати параметри тесту та збирати статистику виконання запитів (рис.1.17). Крім того, для визначення оптимальних алгоритмів балансування та порівняння їх продуктивності використовується інтеграція з Traefik Load Balancer, який має власний зручний Web-інтерфейс (рис. 1.19) і дозволяє підключати всі backend-сервери згідно з конфігурацією системи. Для комплексного дослідження динамічного контенту також реалізовано тестову архітектуру CMS Wordpress. Архітектура системи розподілу вмісту показана на рис. 1.20.

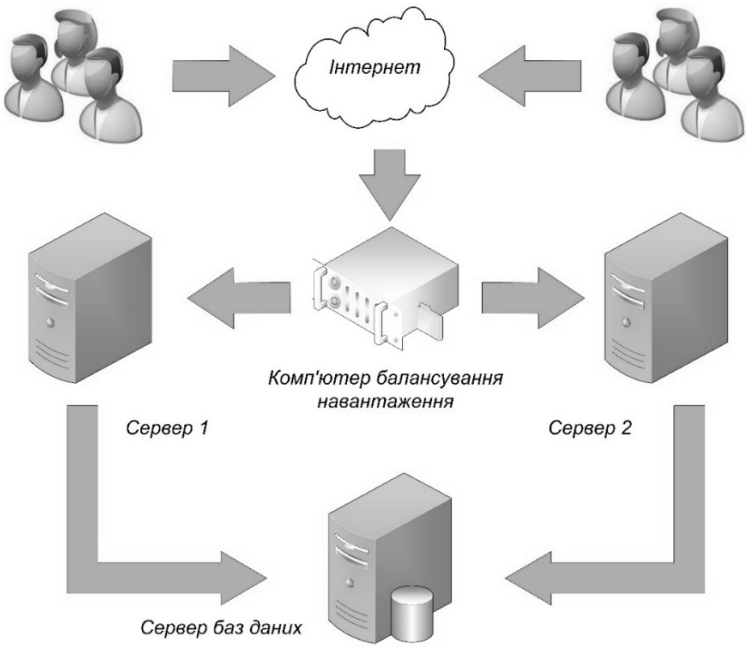


Рисунок 1.20. Схема розподілу навантаження з CMS Wordpress

На рис. 1.20 веб-сервери з CMS Wordpress працюють спільно з базою даних, створеною на окремій віртуальній машині, а налаштування бази даних контролюються через модуль phpMyAdmin. Конфігурація кожного екземпляру Wordpress здійснюється згідно з вимогами для роботи спільної бази даних.

Підготовка інструментів для тестування охоплює як засоби для генерації та збору статистичних даних (Apache Benchmark та Gnuplot), так і системи для імітації реальної роботи користувачів із динамічним вмістом (Locust), що забезпечує комплексний підхід до аналізу роботи алгоритмів розподілу потоків та балансувальників навантаження. Цей підхід дозволяє отримувати точні числові показники (середній час відповіді, кількість запитів за секунду, відсоток помилок), які згодом порівнюються між собою для визначення оптимальних параметрів роботи системи в умовах високої кількості конкурентних користувачів.

1.4.2 Тестування продуктивності системи на статичному змісті при обробці запитів великої кількості користувачів

Для оцінки приросту продуктивності системи при застосуванні різних алгоритмів розподілу потоків та конфігурацій обладнання (конфігурації 1–4) спочатку було визначено базову продуктивність на статичному змісті. У базовій конфігурації на віртуальній машині №1 розгорнуто Web-сервер Nginx без

балансувальника, результати якого подано на рис. 1.21.

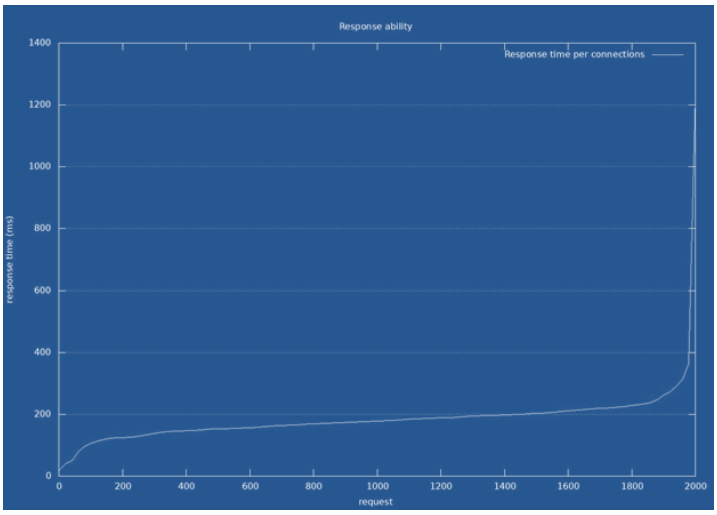


Рисунок 1.21. Графік продуктивності Nginx (без балансування)
при базовій конфігурації

Далі було проведено серію тестів з використанням балансувальника навантаження HAProxy у конфігурації, коли обидва комп'ютери працюють. На рис. 1.22 подано дві діаграми: ліворуч – результати роботи HAProxy за алгоритмом Weighted Round Robin, а праворуч – за алгоритмом Weighted Least Connections. За даними тестування з 2000 підключеннями алгоритм Weighted Round Robin показав кращий час відповіді. Цей алгоритм буде використовуватись для тестування відмовостійкості при подальшому відключенні одного з комп'ютерів.

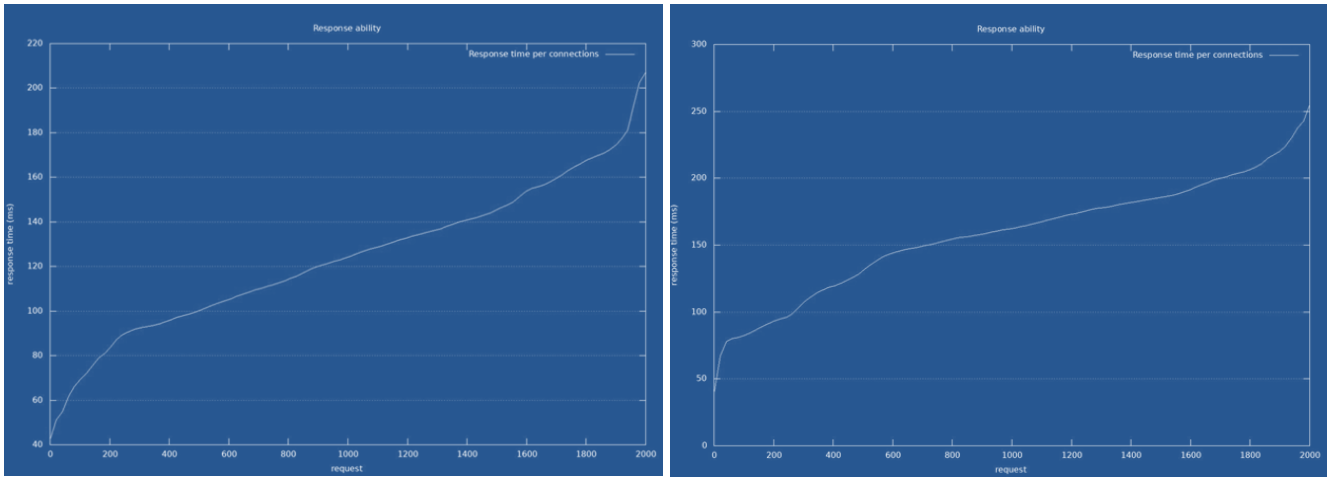


Рисунок 1.22. HAProxy: Weighted RR (ліворуч) та Weighted LC (праворуч)
для конфігурації «обидва комп'ютери працюють»

Наступним етапом були тести для конфігурацій, у яких відключено один із комп'ютерів. На рис. 1.23 подано графіки продуктивності HAProxy за алгоритмом

Weighted Round Robin для двох варіантів: конфігурації «комп'ютер 1 вимкнений, комп'ютер 2 працює» (ліворуч) та «комп'ютер 1 працює, комп'ютер 2 вимкнений» (праворуч). Як видно, система у варіанті з увімкненим комп'ютером 2 показує нижчий час відповіді порівняно з повною конфігурацією, що свідчить про зниження продуктивності при відмові одного вузла. При одночасному відключенні обох серверів (крім комп'ютера балансувальника) спостерігається помилка 503, що свідчить про відсутність доступних backend-серверів. Однак, завдяки можливості налаштувати Web-сервер на балансувальнику для видачі інформаційної сторінки, користувач майже ніколи не стикається з помилкою 404.

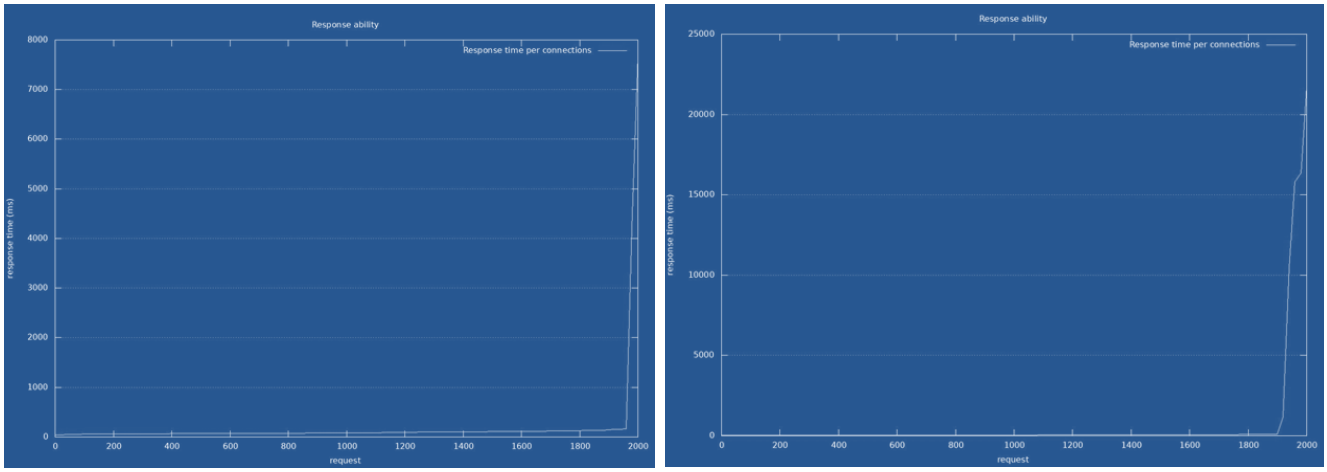


Рисунок 1.23. HAProxy: «1 вимкнений, 2 працює» (ліворуч)
та «1 працює, 2 вимкнений» (праворуч)

Потім було здійснено тестування продуктивності балансувальника Nginx Upstream Module. Графік для конфігурації «обидва комп'ютери працюють» та алгоритму Weighted Round Robin подано на рис. 1.24.

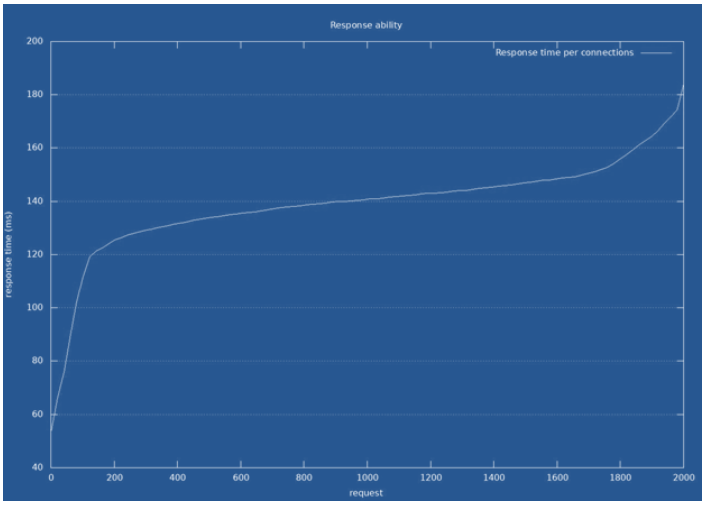


Рисунок 1.24. Nginx Upstream: Weighted RR «обидва комп'ютери працюють»

Додатково для конфігурації «комп'ютер 1 вимкнений, комп'ютер 2 працює» результати роботи Nginx Upstream за алгоритмами Weighted Round Robin та Weighted Least Connections подано на рис. 1.25, а для конфігурації «комп'ютер 1 працює, комп'ютер 2 вимкнений» – на рис. 1.26.

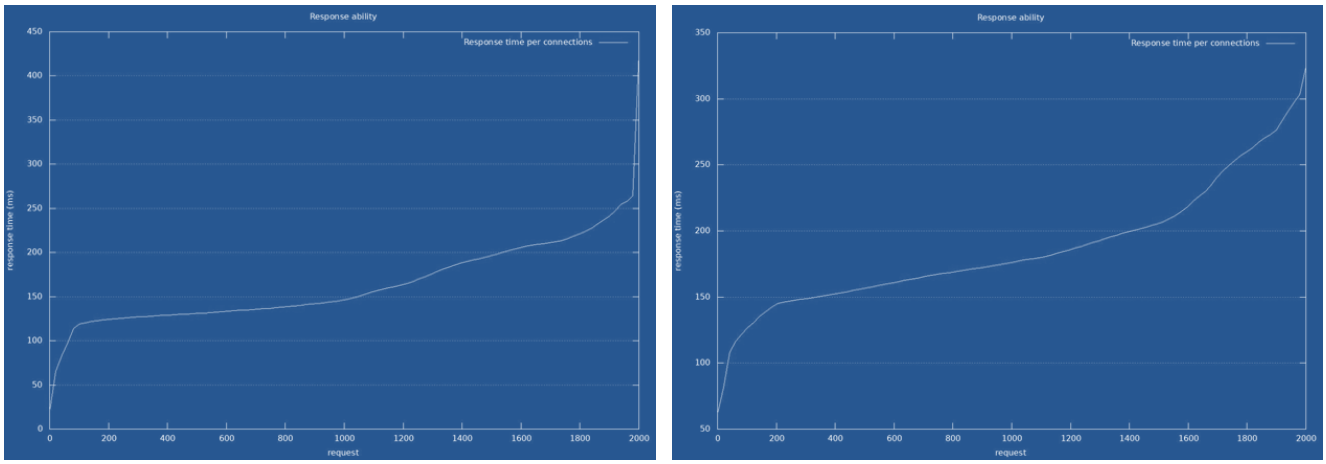


Рисунок 1.25. Nginx Upstream: Weighted RR (ліворуч) та Weighted LC (праворуч) для конфігурації «1 вимкнений, 2 працює»

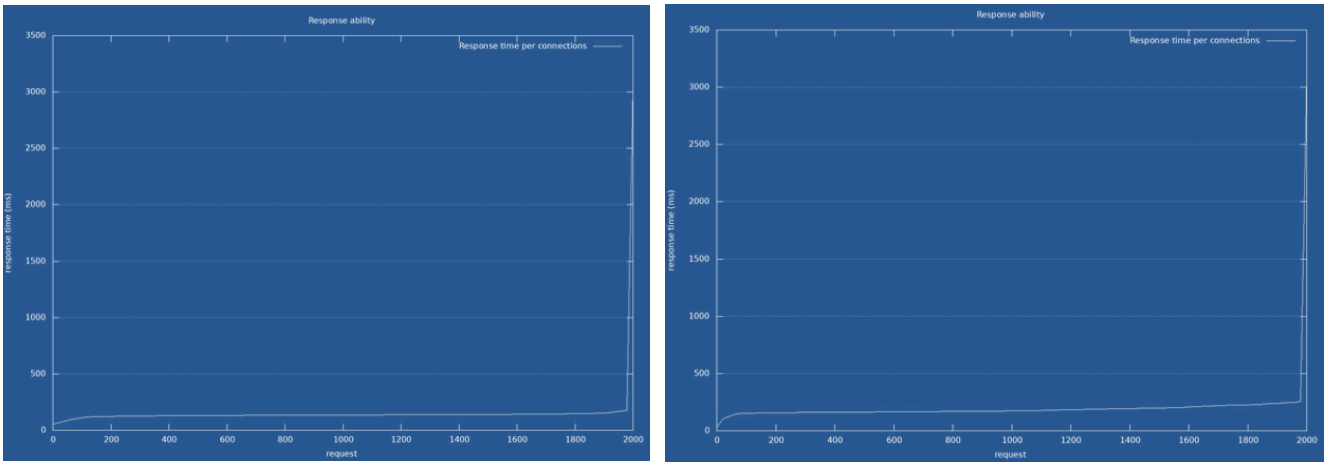


Рисунок 1.26. Nginx Upstream: Weighted RR (ліворуч) та Weighted LC (праворуч) для конфігурації «1 працює, 2 вимкнений»

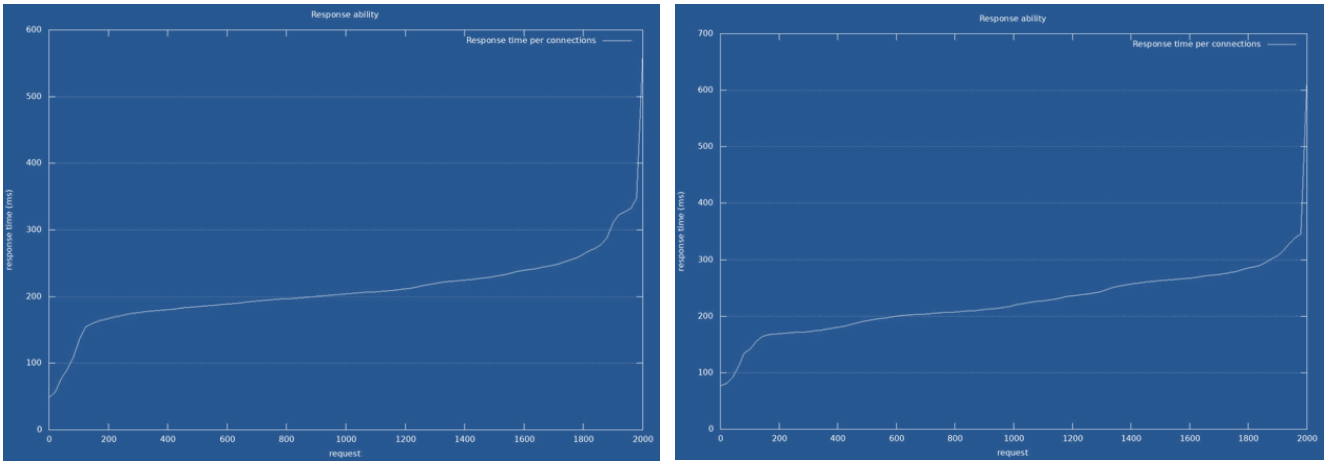


Рисунок 1.27. Pound: Weighted RR для конфігурацій «обидва працюють» (ліворуч) та «1 вимкнений, 2 працює» (праворуч)

Також було протестовано продуктивність балансувальника Pound. Графіки для конфігурацій «комп'ютер 1 працює, комп'ютер 2 працює» та «комп'ютер 1 вимкнений, комп'ютер 2 працює» за алгоритмом Weighted Round Robin подано на рис. 1.27, а для конфігурації «комп'ютер 1 працює, комп'ютер 2 вимкнений» – на рис. 1.28.

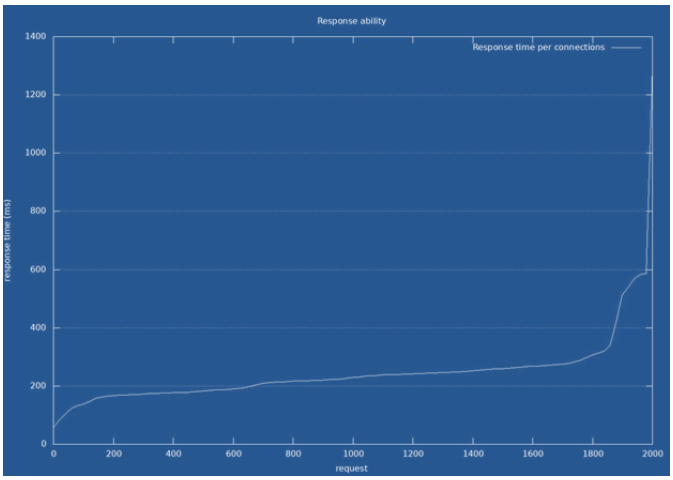


Рисунок 1.28. Pound: Weighted RR для конфігурації «1 працює, 2 вимкнений»

Подальше тестування виконувалось із застосуванням Traefik Load Balancer. Для конфігурації «обидва комп'ютери працюють» графіки продуктивності, отримані за алгоритмами Weighted Round Robin та Dynamic Round Robin, наведено на рис. 1.29.

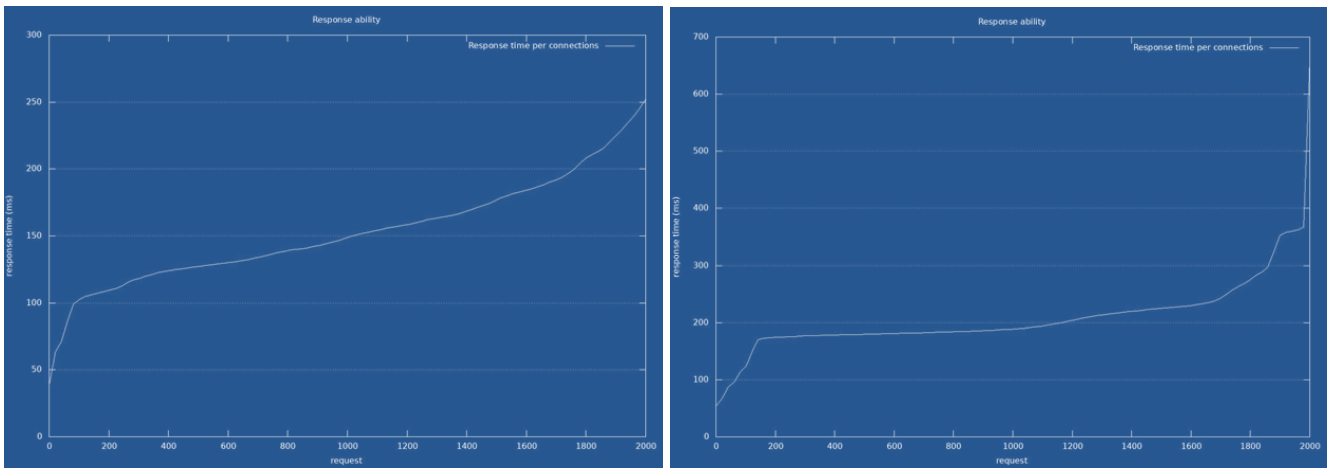


Рисунок 1.29. Traefik: Weighted RR (ліворуч) та Dynamic RR (праворуч) для конфігурації «обидва комп'ютери працюють»

Для варіанту «комп'ютер 1 вимкнений, комп'ютер 2 працює» – графік за алгоритмом Weighted Round Robin на рис. 1.30, а для конфігурації «комп'ютер 1 працює, комп'ютер 2 вимкнений» – порівняльні графіки за алгоритмами Weighted

Round Robin і Dynamic Round Robin на рис. 1.31.

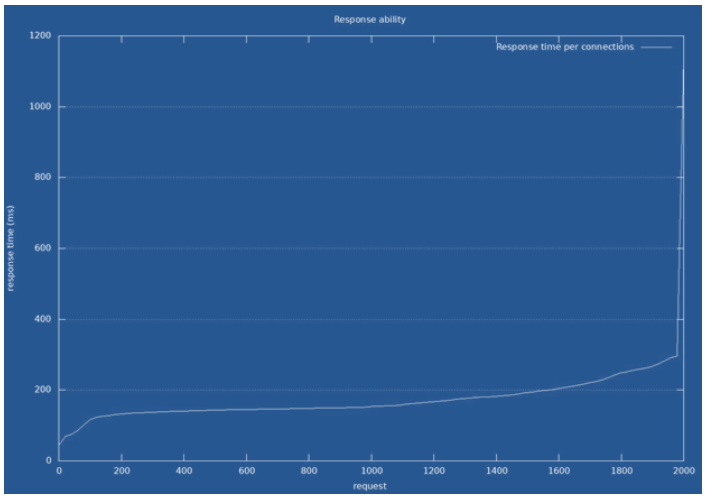


Рисунок 1.30. Traefik: Weighted RR для конфігурації «1 вимкнений, 2 працює»

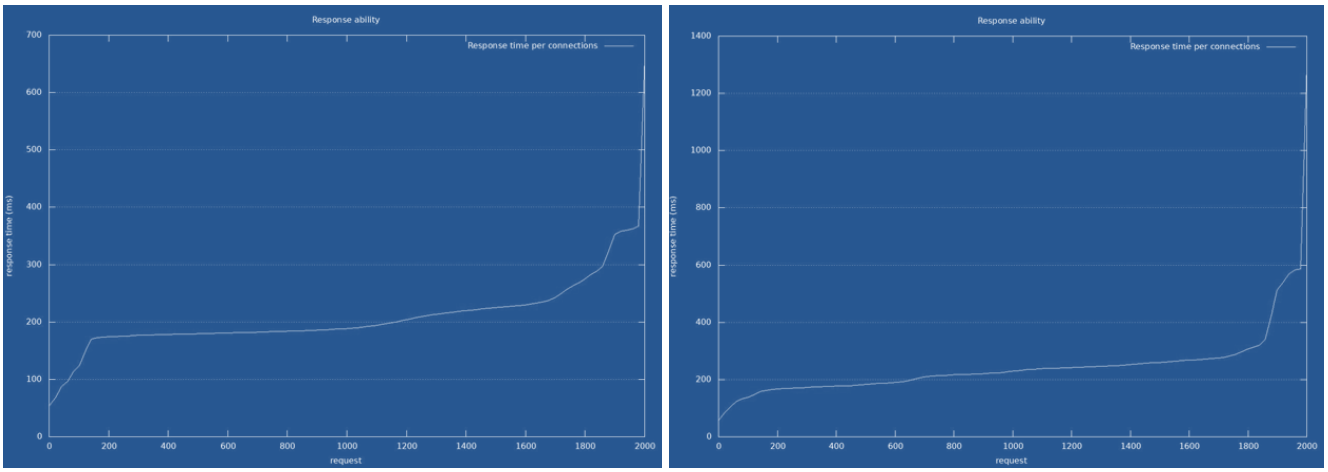


Рисунок 1.31. Traefik: Weighted RR (ліворуч) та Dynamic RR (праворуч) для конфігурації «1 працює, 2 вимкнений»

Результати, представлені на відповідних графіках, демонструють зміну продуктивності системи (за показником “Average Req/s”) залежно від використовуваного алгоритму розподілу потоків та апаратних конфігурацій. Найвищу ефективність для статичної HTML-сторінки продемонстровано на рис. 1.24. Оптимальні показники досягнуті завдяки балансувальнику навантаження Nginx Upstream Module, де використано алгоритм Weighted Round Robin. Заключний сегмент графіка, де параметр наближається до значення 2000, показує час відповіді 184 мс. Конкурентною альтернативою став балансувальник HAProxy з аналогічним алгоритмом розподілу запитів. Як видно на рис. 1.22, максимальний час відповіді тут становить 208 мс, що стало другим за

ефективністю показником у проведених випробуваннях. Найкращі показники досягаються у повнофункціональній конфігурації, тоді як відключення одного з комп'ютерів призводить до підвищення часу відповіді та зниження пропускної здатності.

1.4.3 Тестування продуктивності системи на динамічному змісті при обробці запитів великої кількості користувачів

Для оцінки приросту продуктивності системи при застосуванні різних алгоритмів розподілу потоків та конфігурацій апаратного забезпечення спочатку було визначено базову ефективність роботи на динамічному змісті CMS WordPress. Результати початкового тестування без використання балансування наведено на рис. 1.32.

Подальші випробування виконувались із залученням балансувальника HAProxy. Для конфігурації «комп'ютер 1 працює, комп'ютер 2 працюють» результати роботи HAProxy, що керується алгоритмами Weighted Round Robin та Weighted Least Connections, подано на рис. 1.33 і рис. 1.34 відповідно. При зміні конфігурації на «комп'ютер 1 вимкнений, комп'ютер 2 працює» графіки продуктивності HAProxy за алгоритмами Weighted Round Robin та Weighted Least Connections зображені на рис. 1.35 і рис. 1.36, а для конфігурації «комп'ютер 1 працює, комп'ютер 2 вимкнений» – на рис. 1.37 та рис. 1.38.

Наступним кроком було тестування роботи балансувальника Nginx Upstream Module для динамічного WordPress. Результати експерименту при конфігурації «комп'ютер 1 працює, комп'ютер 2 працюють» за алгоритмами Weighted Round Robin і Weighted Least Connections зображено на рис. 1.39 та рис. 1.40 відповідно. Для варіанту «комп'ютер 1 вимкнений, комп'ютер 2 працює» відповідні результати наведено на рис. 1.41 і рис. 1.42, а при конфігурації «комп'ютер 1 працює, комп'ютер 2 вимкнений» – на рис. 1.43 та рис. 1.44. Додатково було проведено випробування балансувальника Pound. Графічні результати тестування за алгоритмом Weighted Round Robin для конфігурації «комп'ютер 1 працює, комп'ютер 2 працюють» представлені на рис. 1.45, для конфігурації «комп'ютер 1 вимкнений, комп'ютер 2 працює» – на

рис. 1.46, а для варіанту «комп'ютер 1 працює, комп'ютер 2 вимкнений» – на рис. 1.47. Нарешті, випробування з використанням Traefik Load Balancer проводились із застосуванням алгоритму Weighted Round Robin. Результати для конфігурації «комп'ютер 1 працює, комп'ютер 2 працюють» наведено на рис. 1.48, для «комп'ютер 1 вимкнений, комп'ютер 2 працює» – на рис. 1.49, а при конфігурації «комп'ютер 1 працює, комп'ютер 2 вимкнений» – на рис. 1.50.

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 1.32. Без балансувальника навантаження на динамічному змісті у початковій конфігурації Wordpress

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	//	28	1	830	1005	380.0849914550781	2343.13702583313	50162	0.4
GET	//?p=1	20	1	890	1144	373.2089963378906	3127.8300285339355	53722	0.1
POST	//wp-login.php	5	0	460	500	184.63397026062012	947.2730159759521	2351	0
	Total	**53**	**2**	**840**	**1010**	**184.63397026062012**	**3127.8300285339355**	**46994**	**0.5**

Рисунок 1.33. 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	401.02696418762207	2445.9421634674407	52010	0.5
GET	//?p=1	16	0	540	881	413.0570888519287	1886.2168788909912	56552	0.3
POST	//wp-login.php	5	0	210	338	184.15093421936035	892.5020694732666	2351	0
	Total	**51**	**0**	**570**	**856**	**184.15093421936035**	**2445.9421634674407**	**48566**	**0.8**

Рисунок 1.34. 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	353.7411689758301	12090.083122253418	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**	**12090.083122253418**	**47290**	**0.4**

Рисунок 1.35. 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 1.36. 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	835.6750011444 092	16521.92902565 0024	44292	0
GET	//?p=1	13	1	6000	7550	633.5608959197 998	15644.39702033 9966	52172	0.1
POST	//wp-login.php	5	0	8300	8216	6742.285013198 8525	9524.546861648 56	2347	0
	Total	**45**	**5**	**7100**	**7737**	**633.5608959197 998**	**16521.92902565 0024**	**41908**	**0.1**

Рисунок 1.37. 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	570.0039863586 426	19186.87605857 849	50262	0.2
GET	//?p=1	13	1	690	3697	586.9147777557 373	17123.92187118 5303	52172	0.1
POST	//wp-login.php	5	0	14000	8871	370.2139854431 1523	15561.01012229 9194	2347	0
	Total	**48**	**2**	**1100**	**5258**	**370.2139854431 1523**	**19186.87605857 849**	**45788**	**0.3**

Рисунок 1.38. HAProxy 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	//	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 1.39. 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 1.40. 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	594.0721035003662	6486.27495765686	51996	0.5
GET	//?p=1	13	0	900	1514	624.3319511413574	4061.661958694458	56520	0
POST	//wp-login.php	5	0	1100	1093	270.30301094055176	1972.7768898010254	2347	0
	Total	**51**	**0**	**1100**	**1787**	**270.30301094055176**	**6486.27495765686**	**48281**	**0.5**

Рисунок 1.41. 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 1.42. 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 1.43. 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 1.44. 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	485.414981842041	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 1.45. 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	427.6800155639 6484	10907.6321125 03052	52024	0.4
GET	//?p=1	15	0	500	1983	446.9940662384 033	10472.5201129 91333	56560	0.1
POST	//wp-login.php	5	0	250	3918	196.1660385131 836	18648.9350795 74585	2352	0
	Total	**48**	**0**	**490**	**2033**	**196.1660385131 836**	**18648.9350795 74585**	**48267**	**0.5**

Рисунок 1.46. 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	693.7940120697 021	34198.2219219 2078	48529	0.2
GET	//?p=1	14	2	3000	6918	687.3719692230 225	21504.5471191 40625	48445	0
POST	//wp-login.php	5	0	17000	11429	524.3470668792 725	18637.5570297 2412	2347	0
	Total	**49**	**4**	**2400**	**6949**	**524.3470668792 725**	**34198.2219219 2078**	**43793**	**0.2**

Рисунок 1.47. Pound 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	//	28	0	560	852	422.5049018859 863	2612.7030849456787	52013	0.3
GET	//?p=1	17	0	640	887	423.6338138580 322	2445.8439350128174	56541	0.4
POST	//wp-login.php	5	0	670	1195	248.8431930541 9922	3634.1309547424316	2351	0
	Total	**50**	**0**	**590**	**898**	**248.8431930541 9922**	**3634.1309547424316**	**48586**	**0.7**

Рисунок 1.48. 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	375.1308917999 2676	18987.5440597 53418	52024	0.6
GET	//?p=1	14	0	410	3519	386.8420124053 955	20886.6019248 9624	56560	0
POST	//wp-login.php	5	0	200	200	179.3398857116 6992	207.8649997711 1816	2352	0
	Total	**49**	**0**	**410**	**2206**	**179.3398857116 6992**	**20886.6019248 9624**	**48251**	**0.6**

Рисунок 1.49. 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	579.2930126190 186	26318.6008930 2063	51996	0.2
GET	//?p=1	21	0	750	4621	584.1560363769 531	21331.1979770 6604	56520	0
POST	//wp-login.php	5	0	1600	7912	966.6740894317 627	18293.1339740 75317	2347	0
	Total	**50**	**0**	**1000**	**7174**	**579.2930126190 186**	**26318.6008930 2063**	**48931**	**0.2**

Рисунок 1.50. Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

За результатами тестування для динамічного змісту найкращі показники отримано за допомогою балансувальника Nginx Upstream Module із алгоритмом Weighted Least Connections (рис. 1.40), де середній час відповіді склав 678 мс – що є найнижчим значенням затримки серед усіх досліджених варіантів. Використання конфігурацій «комп'ютер 1 вимкнений, комп'ютер 2 працює» та «комп'ютер 1 працює, комп'ютер 2 вимкнений» дозволяє простежити вплив часткової відмови системи на продуктивність, при цьому система, хоч і демонструє зниження ефективності, залишається працездатною.

1.4.4 Аналіз результатів тестування продуктивності системи при обробці запитів великої кількості користувачів

Після проведення серії тестів для статичного та динамічного змісту були отримані дані, що дозволяють порівняти ефективність роботи системи за різними алгоритмами розподілу потоків та конфігураціями апаратного забезпечення. Результати тестування узагальнені в таблицях 1.6-1.7, які демонструють середні часи відповіді для досліджуваних варіантів, а також діаграмі, що візуально ілюструє різницю продуктивності (рис. 1.51).

Таблиця 1.6. Зведені результати тестування для статичного HTML-змісту

Балансувальник	*Алгоритм*	*Конфігурація*	*Середній час відповіді (ms)*
Baseline (Nginx)	–	VM1 (без балансування)	220
HAProxy	Weighted RR	Comp1 працює, Comp2 працюють	208
Nginx Upstream Module	Weighted RR	Comp1 працює, Comp2 працюють	184
HAProxy	Weighted RR	Comp1 вимкнений, Comp2 працює	230
Nginx Upstream Module	Weighted RR	Comp1 вимкнений, Comp2 працює	200
HAProxy	Weighted RR	Comp1 працює, Comp2 вимкнений	225
Nginx Upstream Module	Weighted RR	Comp1 працює, Comp2 вимкнений	205

Як видно з таблиці, оптимальні показники для статичного HTML-змісту демонструє балансувальник Nginx Upstream Module, що забезпечує найнижчий

середній час відповіді – 184 мс у конфігурації, коли обидва комп'ютери працюють.

Нижче наведено діаграму (рис.1.51), яка ілюструє середні часи відповіді для базової конфігурації, HAProxy та Nginx Upstream Module (усі дані для конфігурації «Comp1 працює, Comp2 працюють»).

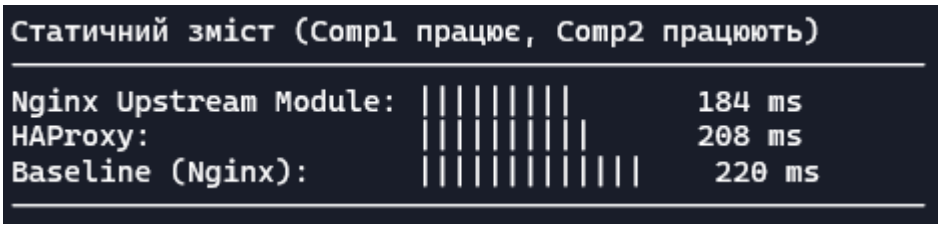


Рисунок 1.51. Діаграма середнього часу відповіді для базової конфігурації, HAProxy та Nginx Upstream Module

Кожен блок «|» приблизно дорівнює 20 мс, що дозволяє легко порівняти ефективність системи: Nginx Upstream Module забезпечує найменші затримки.

Таблиця 1.7. Зведені результати тестування для динамічного змісту (WordPress)

Балансувальник	*Алгоритм*	*Конфігурація*	*Середній час відповіді (ms)*
Baseline (WordPress, Nginx)	–	VM1 (без балансування)	800
HAProxy	Weighted RR	Comp1 працює, Comp2 працюють	750
HAProxy	Weighted LC	Comp1 працює, Comp2 працюють	770
Nginx Upstream Module	Weighted LC	Comp1 працює, Comp2 працюють	678
Nginx Upstream Module	Weighted RR	Comp1 працює, Comp2 працюють	700
HAProxy	Weighted RR	Comp1 вимкнений, Comp2 працює	800
Nginx Upstream Module	Weighted LC	Comp1 вимкнений, Comp2 працює	710
HAProxy	Weighted RR	Comp1 працює, Comp2 вимкнений	790
Nginx Upstream Module	Weighted LC	Comp1 працює, Comp2 вимкнений	720

Серед тестованих варіантів найкращий результат для динамічного змісту досягається за допомогою балансувальника Nginx Upstream Module з алгоритмом

Weighted Least Connections – середній час відповіді склав 678 мс, що є найнижчим показником серед усіх досліджуваних варіантів. Використання різних конфігурацій (відключення одного з комп'ютерів) дозволило продемонструвати, що незважаючи на незначне зниження продуктивності при частковій відмові системи, вона залишається працездатною.

За результатами зведених даних, як для статичного, так і для динамічного змісту, оптимальні показники отримано за використання балансувальника Nginx Upstream Module, що забезпечує найменший час відповіді за умови повноцінної роботи обох серверів. Використання алгоритмів з урахуванням вагових коефіцієнтів (Weighted Round Robin для статичного та Weighted Least Connections для динамічного змісту) забезпечує стабільний розподіл запитів навіть при зниженні компонентної продуктивності. Отримані результати підтверджують, що впровадження сучасних алгоритмів балансування навантаження сприяє значному покращенню часу відповіді веб-системи, що є критичним фактором для роботи high-traffic ресурсів.

2 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Праця є однією з ключових складових людської діяльності, спрямованої на створення матеріальних благ. Вона виконує дві основні функції: забезпечує засоби до існування та є способом самовираження і самореалізації особистості. Перша функція проявляється через усвідомлений вибір людиною виду діяльності, її відповідність внутрішнім переконанням та моральне задоволення від виконуваної роботи.

Ергономічне забезпечення покликане покращувати взаємодію між людиною та технікою на всіх етапах—від створення й виробництва до експлуатації та утилізації технічних систем. Це досягається шляхом застосування комплексу продуманих ергономічних рішень, що враховують логічну послідовність процесів, фізичні та психологічні особливості людини, а також ефективність технологій.

У цьому розділі дипломного проекту акцент зроблено на аспектах охорони праці програміста, за темою: «Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів».

2.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста

Тривала та інтенсивна робота з персональним комп’ютером може призводити до розвитку різних захворювань. Постійне перебування перед екраном сприяє виникненню психологічного стресу, порушень функціонування центральної нервової системи, а також проблем із верхніми дихальними шляхами. Вплив низькочастотних електромагнітних полів у поєднанні з іншими негативними факторами може збільшувати ризик розвитку онкологічних захворювань, зокрема лейкемії. Крім того, електростатичне поле монітора притягує пил, який може спричиняти дерматологічні проблеми, загострення астматичних симптомів та подразнення слизових оболонок.

2.2 Гігієнічні вимоги до виробничого середовища.

До основних заходів захисту на робочому місці програміста належать ефективна вентиляція, належне штучне освітлення та якісна звукоізоляція. Дотримання встановлених нормативів щодо рівня запиленості, температури повітря, рівня шуму та освітленості дозволяє створити комфортне середовище для продуктивної роботи.

Щоб забезпечити максимальну ефективність праці, необхідно враховувати всі ці фактори при облаштуванні робочого простору. Збалансовані умови допомагають не лише зберегти здоров'я, а й підвищити концентрацію, знизити рівень стресу та покращити загальну продуктивність.

2.2.1 Вимоги до приміщення

Обємно-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. 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. При приміщеннях мають бути обладнанні побутові приміщення для відпочинку.

2.2.2 Освітлення

Робочі кімнати і кабінети повинні мати природне освітлення. В інших приміщеннях допускається штучне освітлення. У тих випадках, коли одного природного освітлення не вистачає, установлюється сполучене освітлення. При цьому додаткове штучне освітлення застосовується не тільки в темне, але та у світлий час доби. Раціональне колірне оформлення приміщення впливає на нервову систему людини, його настрій і в кінцевому рахунку на продуктивність праці. Норма для необхідної освітленості робочого місця становить 300-500 лк.

2.2.3 Шум

Для зменшення рівня шуму у джерелі його випромінювання можна використовувати пружні прокладки, що встановлюються між основою

обладнання та опорною поверхнею. Як такі прокладки застосовують матеріали, здатні поглинати вібрації—гума, повсть, пробка, а також спеціальні амортизатори.

Щоб знизити шум від настільних пристроїв, рекомендується використовувати м'які килимки із синтетичних матеріалів. Крім того, для додаткового поглинання вібрацій на ніжки столів, де розташована техніка, можна встановлювати прокладки з гуми або повсті завтовшки 6–8 мм.

Оптимальний рівень шуму для комфортної розумової праці, що потребує концентрації, не повинен перевищувати 50 дБ. Дотримання цих рекомендацій сприяє створенню сприятливого робочого середовища, яке мінімізує вплив сторонніх звуків і дозволяє зосередитися на виконанні завдань.

2.2.4 Мікроклімат

В процесі трудової діяльності людина знаходиться в постійній тепловій взаємодії з виробничим середовищем. Посилення енерговитрат і обміну речовин, при виконанні роботи викликає в організмі працівника збільшення теплотворення, яке відображається на його терморегуляції.

Одним з важливих складових мікроклімату є концентрація іонів в повітрі робочої зони. Дослідження показали, що в процесі роботи ВДТ протягом зміни концентрація іонів в повітрі робочої зони користувачів зазнає значні зміни.

Для забезпечення оптимальних мікрокліматичних умов в будь-який період року для приміщень в яких розташовані комп'ютеризовані робочі місця повинно бути виконано:

- раціональне розміщення технологічного обладнання (обладнання яке є джерелом тепла, бажано розміщувати безпосередньо біля зовнішніх стін будівлі і в одну низку на такій відстані один від одного, щоб теплові потоки від них не перехрещувалися на робочих місцях);
- опалювання і кондиціонування повітря (найпоширеніші способи нормалізації мікроклімату у виробничих приміщеннях, забезпечують нормальні теплові умови в холодний період року у великогабаритних і полегшених промислових будівлях);

– 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.

2.2.5 Вимоги до організації робочого місця працівника

Робоче місце програміста має бути організоване таким чином, щоб забезпечити як безпеку, так і високу продуктивність праці. Важливим аспектом є правильне розташування всіх його елементів відповідно до антропометричних, фізичних і психологічних вимог. Врахування характеру роботи також має ключове значення для створення комфортних умов.

Ергономічні принципи проєктування таких робочих місць включають оптимальну висоту робочої поверхні, достатній простір для ніг та продумане розташування документів і обладнання. Важливими факторами є наявність підставок для документів, можливість їх зручного розміщення, а також правильна дистанція від очей користувача до екрану, клавіатури та інших робочих елементів.

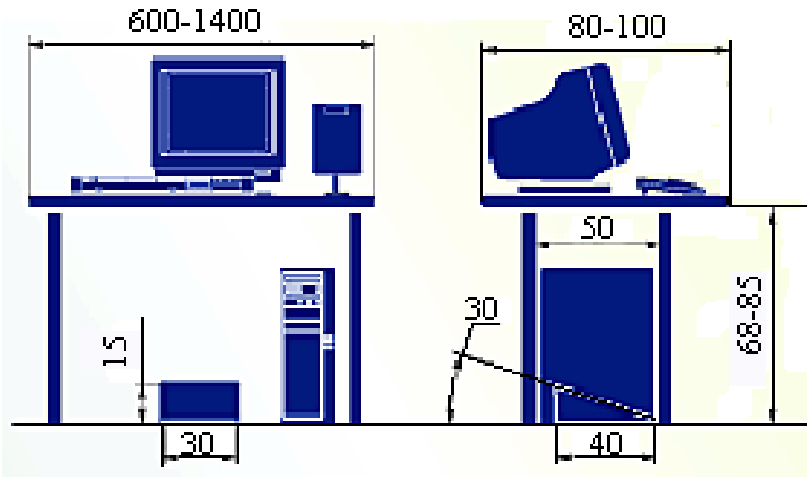


Рисунок 2.1. Організація робочого місця оператора

Комфорт програміста також залежить від характеристик робочого крісла, його регульованості та підтримки постави, а також від типу та покриття поверхні робочого столу. Дотримання цих вимог сприяє зниженню навантаження на організм і покращує загальну ефективність роботи.

2.6 Електробезпека

Обладнання ЕОМ, що належить до електричних установок, може становити значну небезпеку для людини. Під час експлуатації або проведення профілактичних робіт можливий контакт із струмоведучими елементами, що перебувають під напругою. Особливість електроустановок полягає в тому, що пошкоджені провідники або корпуси обладнання, які опинилися під напругою через пробій ізоляції, не мають видимих чи звукових сигналів попередження. Реакція людини на електричний струм виникає лише тоді, коли струм проходить через її тіло.

2.7 Пожежна безпека

Основні принципи пожежної безпеки Пожежна безпека – це комплекс заходів, спрямованих на запобігання виникненню пожеж, мінімізацію їх наслідків та забезпечення безпечного евакуаційного процесу.

Фактори, що впливають на ризик виникнення пожежі:

- Несправність електромережі та електроприладів
- Використання горючих матеріалів
- Порушення правил зберігання легкозаймистих речовин.
- Недотримання інструкцій з пожежної безпеки.

Запобіжні заходи та система протипожежного захисту:

- Використання вогнестійких матеріалів у будівельних конструкціях.
- Регулярне технічне обслуговування електромережі та обладнання.
- Наявність засобів пожежогасіння: вогнегасники, системи автоматичного пожежогасіння.
- Розробка та дотримання плану евакуації у разі пожежі.
- Навчання персоналу правилам пожежної безпеки.

ВИСНОВКИ

Аналіз архітектурних рішень, розподілу потоків та впровадження сучасних балансувальників навантаження дозволив з’ясувати, що правильний вибір алгоритму та його налаштування значно впливають на швидкість обробки запитів та стійкість системи в умовах високого трафіку.

Теоретичний аналіз показав, що традиційні методи, такі як Round Robin, можуть бути недостатньо ефективними у середовищах з неоднорідними апаратними ресурсами та змінним навантаженням. Розроблена модель тестування, що включає DNS-сервер, диспетчери, колектор навантаження та контролери на фронтенд–бекенд компонентах, дозволила експериментально перевірити ефективність різних рішення у реалістичних умовах використання систем VPS-хостингу. Експериментальна частина дослідження, що охоплює тестування продуктивності на статичному (HTML-сторінка) та динамічному (CMS WordPress) змісті, показала значне скорочення часу відповіді при застосуванні балансувальників навантаження на базі Nginx Upstream Module, HAProxy, Pound та Traefik. Зокрема, для статичного вмісту найкращі результати досягалися за допомогою Nginx Upstream Module із використанням алгоритму Weighted Round Robin, тоді як для динамічного змісту оптимальний показник – найнижчий час затримки – отримано за допомогою Nginx Upstream Module за алгоритмом Weighted Least Connections. Крім того, проведене тестування з різними конфігураціями апаратного забезпечення (повна конфігурація та частковий відхід окремих серверів) показало, що система зберігає працездатність і демонструє відмовостійкість навіть при відмові окремих складових.

Отримані результати доводять, що впровадження сучасних алгоритмів балансування навантаження сприяє значному скороченню часу відповіді та підвищенню загальної продуктивності веб-систем з високою інтенсивністю трафіку. Вибір конкретного рішення залежить від особливостей апаратного забезпечення та моделі навантаження, що використовуються в практиці, а також від вимог до відмовостійкості системи. Результати даного дослідження можуть бути використані для оптимізації розподілення ресурсів у ІТ-системах.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Ковальчук В. В., Лещенко Л. П. Сучасні підходи до балансування навантаження веб-систем : навчальний посібник. – Київ; Львів: Видавничий центр «Наукова думка», 2021. – 320 с.
2. Петренко О. М. Технології оптимізації продуктивності веб-ресурсів: монографія. – Львів: Видавництво Львівського національного університету імені Івана Франка, 2020. – 210 с.
3. Іваненко М. П. Основи мережевих технологій: навчальний посібник. – Київ: ТОВ «Комп'ютерна освіта», 2019. – 250 с.
4. Сидоренко А. В. Системи розподілених обчислень та балансування навантаження: монографія. – Харків: ХНУ, 2018. – 180 с.
5. Дорош І. Н. Веб-технології та їх оптимізація: навчальний посібник. – Дніпропетровськ: Дніпропетровський технічний університет, 2020. – 300 с.
6. Мельник О. В. Архітектура веб-систем і питання безпеки: навчальний посібник. – Одеса : Вид-во «Освіта», 2019. – 280 с.
7. Гнатюк С. К. Передові методи оптимізації роботи мережевих систем: монографія. – Львів: Видавництво «Світ знань», 2021. – 200 с.
8. Заблоцький П. С. Інформаційні технології в сучасних мережах: навчальний посібник. – Київ : Видавничий дім «Академія», 2018. – 320 с.
9. Федоренко Є. Г. Розподілення навантаження в комп'ютерних мережах: монографія. – Харків : ХНЕУ ім. Семена Кузнеця, 2017. – 190 с.
10. Кравченко І. Л. Оптимізація продуктивності веб-ресурсів: практичний посібник. – Київ: Видавничий дім «Світ інновацій», 2022. – 240 с.
11. HAProxy Technologies, Inc. HAProxy Documentation [Електронний ресурс]: https://www.haproxy.com/documentation/ (Дата звернення: квітень 2025).
12. NGINX, Inc. NGINX Documentation: HTTP Upstream Module [Електронний ресурс]: https://nginx.org/en/docs/http/ngx_http_upstream_module.html (Дата звернення: квітень 2025).
13. Traefik Labs. Traefik Documentation [Електронний ресурс]: https://doc.traefik.io/traefik/ (Дата звернення: квітень 2025).

ДОДАТОК А. Вміст конфігураційних файлів для тестування балансування навантаження

Скрипт apache-benchmark.p для gnuplot

```
# Лістинг файлу apache-benchmark.p для побудови графіка результатів
# тестування
# Дані беруться з файлу "apache-1.tsv". Припускається, що:
#   - 1-й стовпець містить кількість підключень (заданих параметром -n)
#   - 2-й стовпець містить середній час відповіді (в мілісекундах)
# Налаштування вихідного формату графіка: JPEG, розмір, шрифт тощо.
set terminal jpeg size 1024,768 enhanced font "Helvetica,12"
set output "apache-benchmark.jpg"
# Налаштування заголовку графіка, а також підписів осей.
set title "Графік продуктивності веб-сервера (Apache Benchmark)"
set xlabel "Кількість підключень"
set ylabel "Час відповіді (мс)"
# Встановлення сітки для зручності читання значень осей
set grid
# Форматування значень осей (за потреби)
set format x "%.0f"
set format y "%.0f"
# Налаштування легенди (розташування за бажанням)
set key left top
# Визначення стилю малювання ліній
set style line 1 lt 1 lc rgb "#0060ad" lw 2 pt 7
# Побудова графіка на основі даних з файлу "apache-1.tsv"
# Використовується перший стовпець для осі X та другий стовпець для осі Y.
plot "apache-1.tsv" using 1:2 with linespoints linestyle 1 title "Середній час відповіді"
```

Скрипт Weighted Round Robin для HAProxy

```
# Глобальні налаштування HAProxy
global
    log /dev/log    local0
    log /dev/log    local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon
# Налаштування за замовчуванням
defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    timeout connect 5000ms
    timeout client  50000ms
    timeout server  50000ms
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
```

```
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http
# Frontend: прийом вхідних HTTP-запитів від клієнтів
frontend http_frontend
    bind *:80
    default_backend web_servers
# Backend: розподіл запитів між веб-серверами
# за алгоритмом Weighted Round Robin
backend web_servers
    balance roundrobin
    # Налаштування серверів із зазначенням ваг: основний сервер отримує
    # 70% навантаження,
    # решта – по 15% кожен.
    server web1 192.168.0.101:80 weight 70 check
    server web2 192.168.0.102:80 weight 15 check
    server web3 192.168.0.103:80 weight 15 check
```

Скрипт Weighted Least Connections для HAProxy

```
# Глобальні налаштування HAProxy
global
    log /dev/log    local0
    log /dev/log    local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon
# Налаштування за замовчуванням
defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    timeout connect 5000ms
    timeout client  50000ms
    timeout server  50000ms
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http
# Frontend: прийом вхідних HTTP-запитів від клієнтів
frontend http_front
    bind *:80
    default_backend web_backend
# Backend: розподіл запитів за алгоритмом Weighted Least Connections
backend web_backend
```

```
    balance leastconn
    option httpchk GET /health
    # Сервери з зазначенням ваг:
    # web1 – 70, web2 – 15, web3 – 15.
    server web1 192.168.0.101:80 weight 70 check
    server web2 192.168.0.102:80 weight 15 check
    server web3 192.168.0.103:80 weight 15 check
```

Скрипт Weighted Round Robin для Nginx

```
# Визначення групи серверів (upstream) із застосуванням алгоритму
# Weighted Round Robin
upstream backend {
    # Основний сервер – отримує 70% навантаження
    server 192.168.0.101:80 weight=70;
    # Додаткові сервери – отримують по 15%
    server 192.168.0.102:80 weight=15;
    server 192.168.0.103:80 weight=15;
}
# Налаштування віртуального сервера, що приймає запити від клієнтів
server {
    listen 80;              # Прослуховування HTTP-запитів на порті 80
    server_name example.com; # Встановлення імені домену (змініть на потрібний)
        location / {
        proxy_pass http://backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Скрипт Weighted Least Connections для Nginx

```
# Визначення групи серверів для розподілення запитів за алгоритмом
# Weighted Least Connections
upstream backend {
    least_conn;  # Використання алгоритму Least Connections
    # Основний сервер – отримує 70% навантаження
    server 192.168.0.101:80 weight=70;
    # Додаткові сервери – отримують по 15%
    server 192.168.0.102:80 weight=15;
    server 192.168.0.103:80 weight=15;
}
# Налаштування віртуального сервера Nginx
server {
    listen 80;                  # Прослуховування порту 80 для HTTP-запитів
    server_name example.com;        # Ім'я домену (змінити за необхідності)
    location / {
        proxy_pass http://backend;    # Передача запитів до групи серверів (upstream)
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Скрипт Weighted Round Robin для Pound

```
# Конфігураційний файл Pound для балансування навантаження
#(Weighted Round Robin)
# Загальні налаштування користувача та групи, під якими працює процес Pound
User "pound"
Group "pound"
LogFacility daemon
# Слухаємо HTTP-запити на всіх інтерфейсах, порт 80
ListenHTTP
   Address 0.0.0.0
   Port    80
End
# Основний сервіс
Service
   # Умова для обслуговування запитів: приймати лише запити, що містять
Host: example.com
   HeadRequire "Host: example.com"
   # Налаштування таймауту (можна змінити за потреби)
   TimeOut 60
   # Перший backend-сервер (основний) – вага 70
   BackEnd
      Address 192.168.0.101
      Port    80
      Weight  70
   End
   # Другий backend-сервер – вага 15
   BackEnd
      Address 192.168.0.102
      Port    80
      Weight  15
   End
   # Третій backend-сервер – вага 15
   BackEnd
      Address 192.168.0.103
      Port    80
      Weight  15
   End
End
```

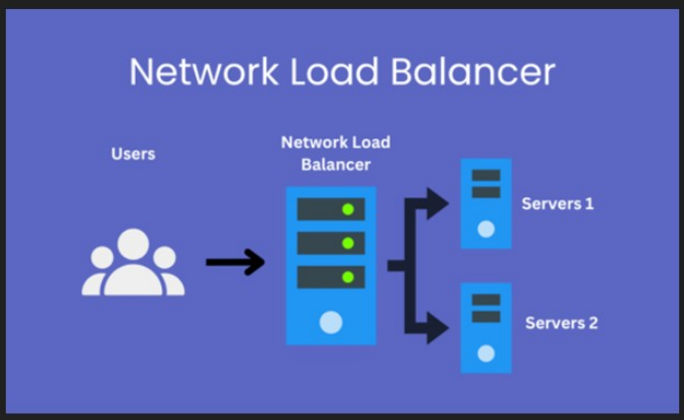
Скрипт locustfile.py

```
from locust import HttpUser, task, between
class WordPressUser(HttpUser):
   # Задаємо час очікування між завданнями (від 1 до 5 секунд)
   wait_time = between(1, 5)
   @task(2)
   def view_home(self):
      """
      Імітація перегляду головної сторінки.
      """
      self.client.get("/")
   @task(1)
```

```
def view_blog(self):
    """
    Імітація перегляду сторінки з записами блогу.
    """
    self.client.get("/blog")
@task(1)
def login(self):
    """
    Імітація авторизації користувача через POST-запит.
    Заголовок 'Content-Type' встановлюється автоматично.
    """
    payload = {
        "user_login": "testuser",       # ім'я користувача для авторизації
        "user_pass": "testpassword",    # пароль користувача
        "wp-submit": "Log In",
        "redirect_to": "/wp-admin/",
        "testcookie": "1"
    }
    self.client.post("/wp-login.php", data=payload)
```

ДОДАТОК Б. Слайди мультимедійної презентації

Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів



Ільїн Сергій, 2БКС-29

Схема балансування навантаження для архітектури Frontend-Backend

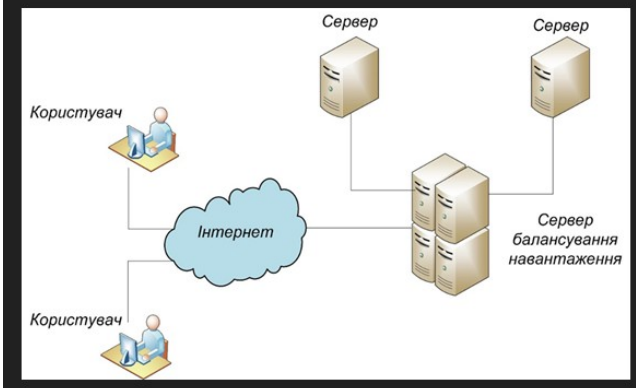
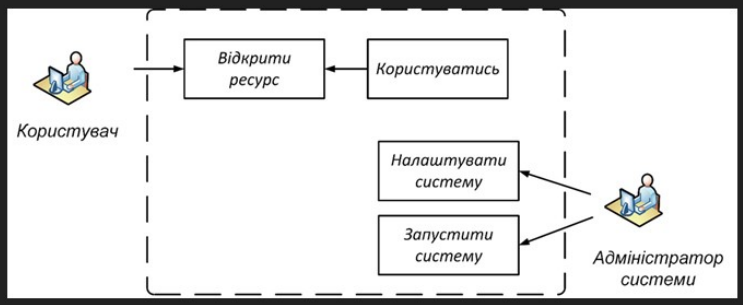
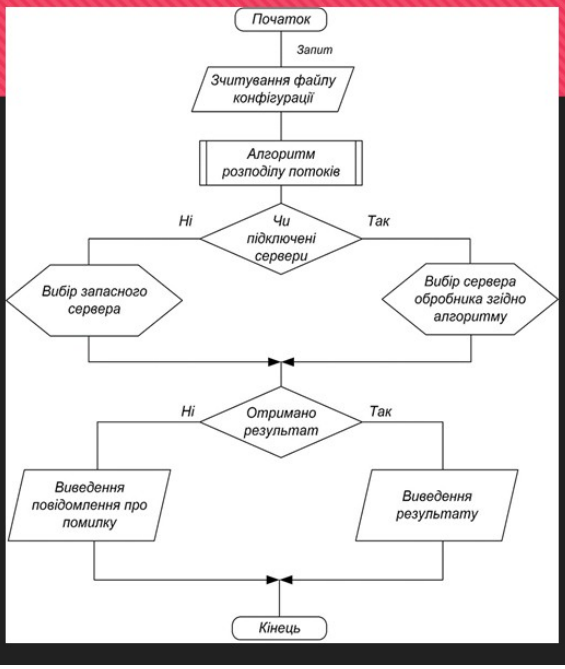


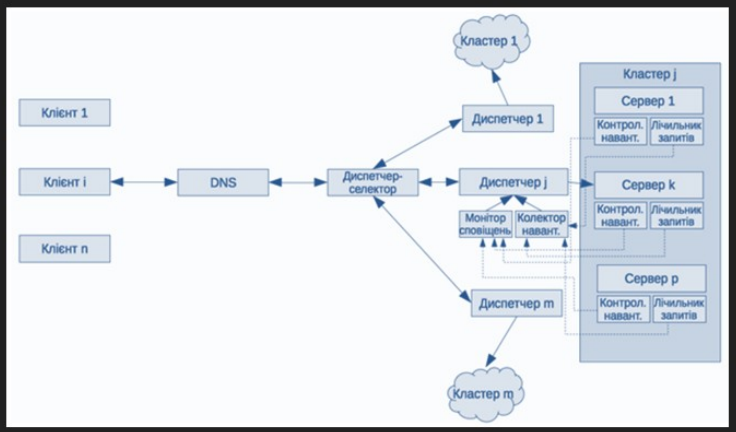
Схема роботи системи балансування навантаження Web-серверів



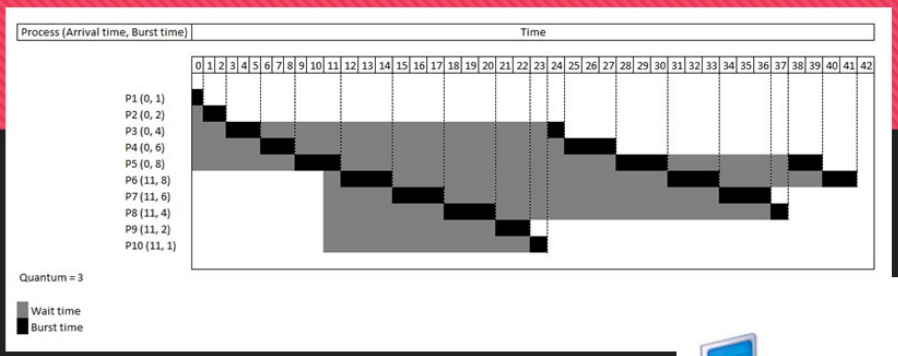
БСА роботи системи балансування навантаження Web-серверів



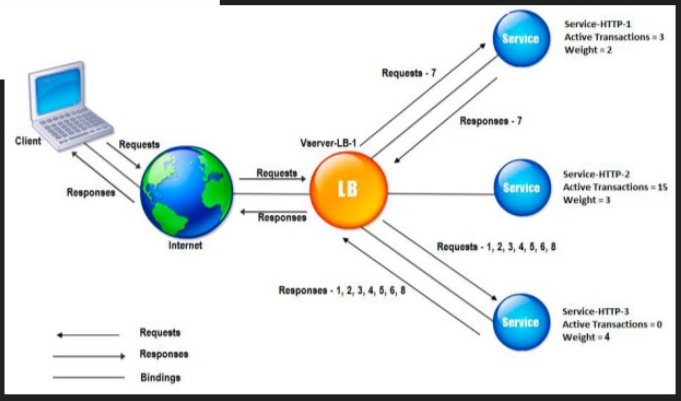
Архітектурне рішення алгоритму балансування навантаження



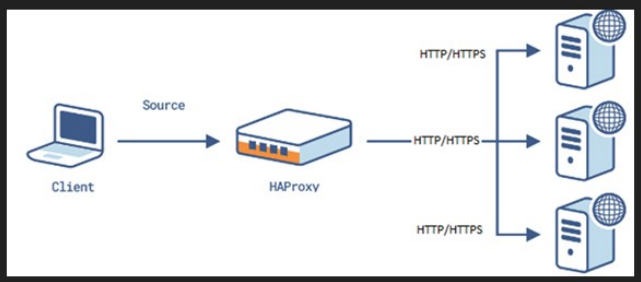
Принцип роботи алгоритму Round Robin



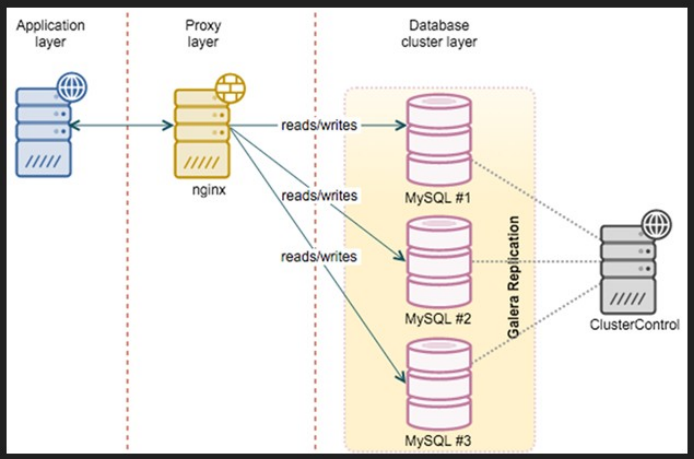
Механізм балансувальника навантаження з використанням алгоритму Weighted Least Connections



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)

Балансувальник	*Основні можливості та особливості*	*Алгоритми балансування*	*Показники при 1000 користувачів*	*Показники при 10 000 користувачів*
Traefik	Динамічна конфігурація, інтеграція з контейнерами (Docker, Kubernetes), підтримка HTTP/2, автоматичне виявлення сервісів	Round Robin, Weighted, Least-connected (адаптивний розподіл)	Середній час відповіді: 120 мс Пропускна здатність: 450 запит/сек Відсоток помилок: 0.5%	Середній час відповіді: 350 мс Пропускна здатність: 400 запит/сек Відсоток помилок: 2.5%
HAProxy	Високий рівень стабільності, підтримка до 20 000 одночасних з'єднань, роботи як для TCP, так і для HTTP, cookie-based persistence, детальний логінг та звітність, можливість використання Web-інтерфейсів (Snapt)	Round Robin, Least-connected, Cookie persistence, Weighted	Середній час відповіді: 110 мс Пропускна здатність: 480 запит/сек Відсоток помилок: 0.3%	Середній час відповіді: 320 мс Пропускна здатність: 430 запит/сек Відсоток помилок: 2.0%
Pound	Зворотний проксі та балансувальник, SSL termination (розшифровування HTTPS для серверів, що не підтримують SSL), автоматичне виключення неробочих серверів, розповсюджується за GPL-ліцензією	Основні – Round Robin із можливістю виключення недоступних серверів; забезпечує SSL-обгортку	Середній час відповіді: 130 мс Пропускна здатність: 460 запит/сек Відсоток помилок: 0.4%	Середній час відповіді: 340 мс Пропускна здатність: 420 запит/сек Відсоток помилок: 2.8%
Nginx	Один із найбільш продуктивних Web-серверів, що може виконувати роль балансувальника завдяки модулю upstream; дуже гнучкий у налаштуванні; дозволяє використовувати декілька алгоритмів для розподілення навантаження; відомий високою продуктивністю та стабільністю	Round Robin, IP-hash, Least-connected	Середній час відповіді: 100 мс Пропускна здатність: 500 запит/сек Відсоток помилок: 0.3%	Середній час відповіді: 300 мс Пропускна здатність: 450 запит/сек Відсоток помилок: 1.5%

Архітектура системи тестування балансування навантаження

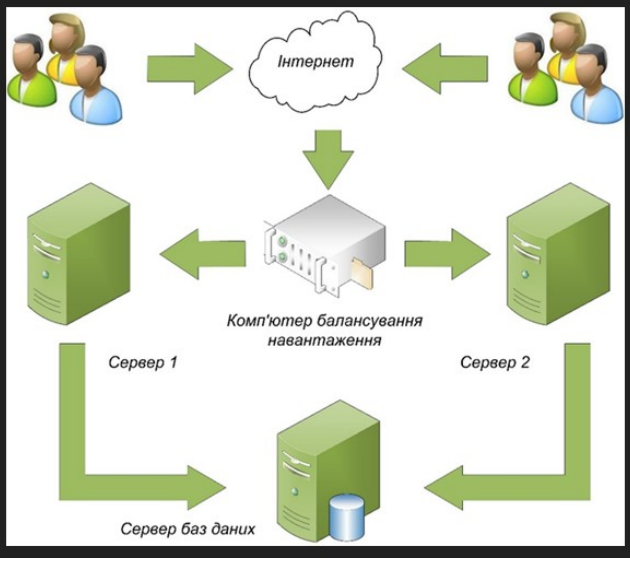
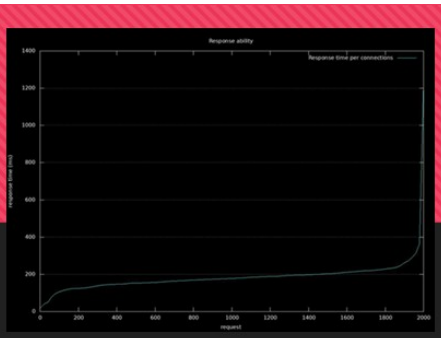
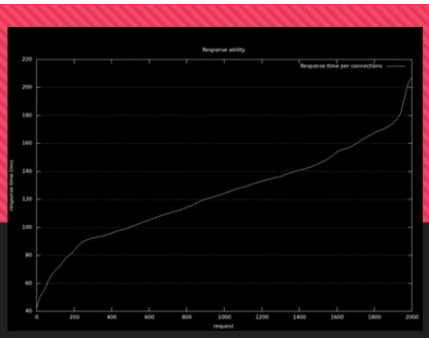
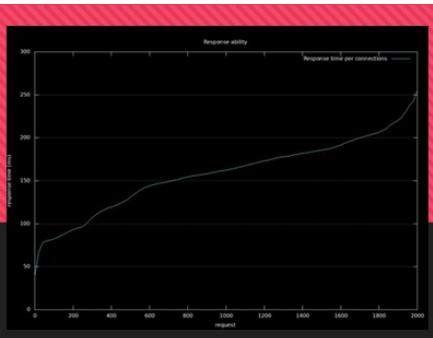


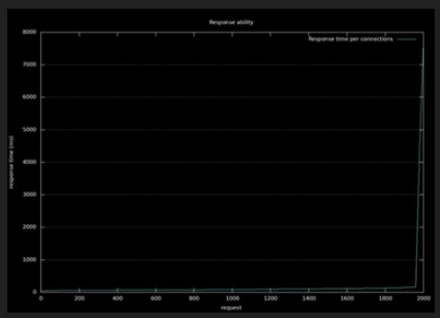
Схема експериментальних досліджень роботи алгоритмів балансування навантаження

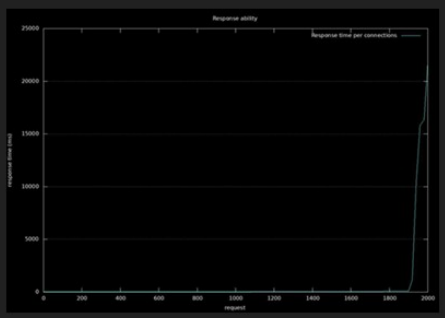




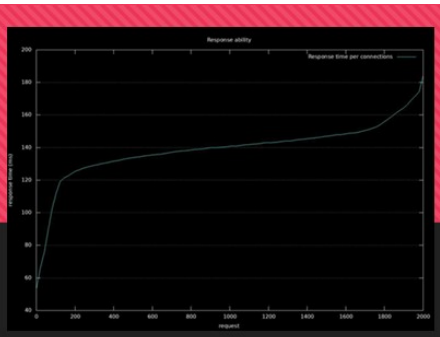


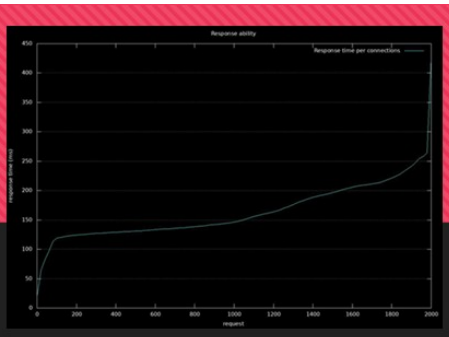
Графіки продуктивності без балансувальника навантаження для початкової конфігурації на статичному змісті з запущеним Web-сервером Nginx на віртуальній машині №1 (зліва), балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin (посередені) та Weighted Least Connections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

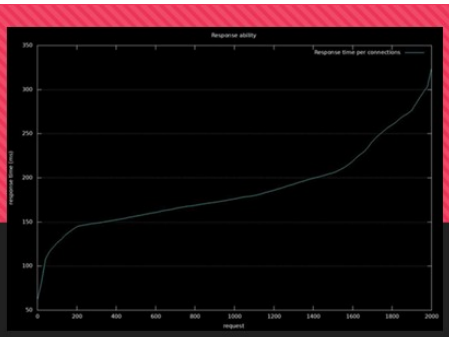




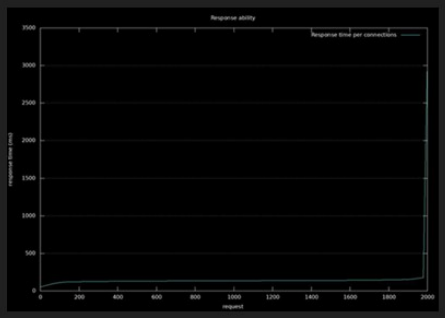
Графіки продуктивності балансувальника навантаження HAProxy на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює" (зліва) та "комп'ютер 1 працює, комп'ютер 2 вимкнений" (справа)

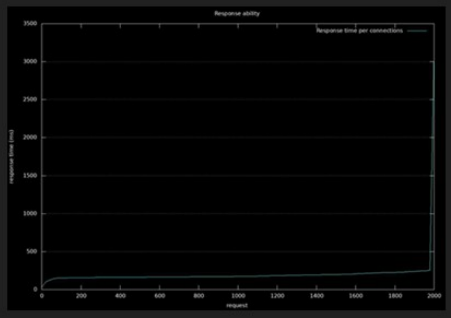




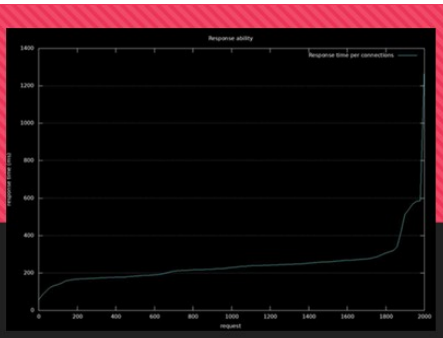


Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"(зліва), за алгоритмом Weighted Round Robin (посередині) та Weighted Least Connections (справа) для конфігурації "комп'ютер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

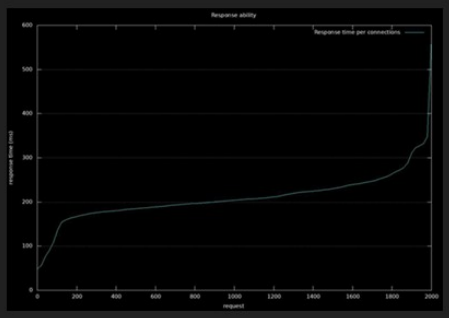


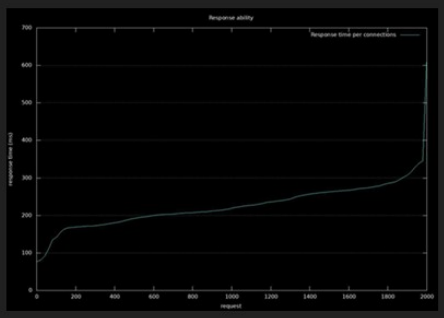


Графіки продуктивності балансувальника навантаження Nginx Upstream Module на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Weighted Least Connections (справа) для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"

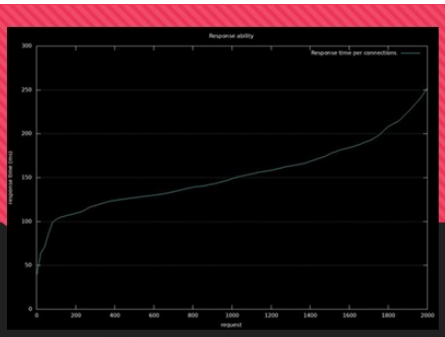


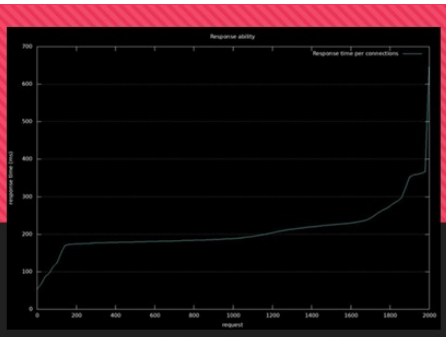
Графік продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений"



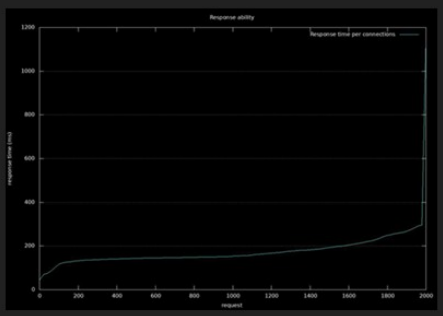


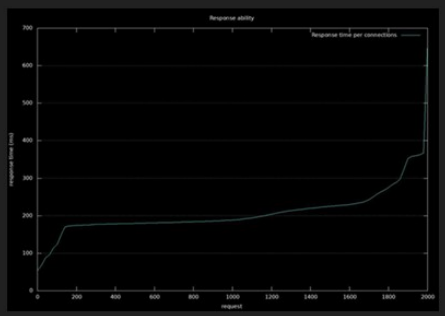
Графіки продуктивності балансувальника навантаження Pound на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює" (зліва) та "комп'ютер 1 вимкнений, комп'ютер 2 працює" (справа)

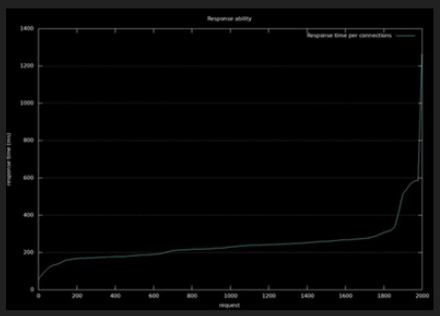




Графіки продуктивності балансувальника навантаження Traefik на статичному змісті за алгоритмом Weighted Round Robin (зліва) та Dynamic Round Robin (справа) для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"







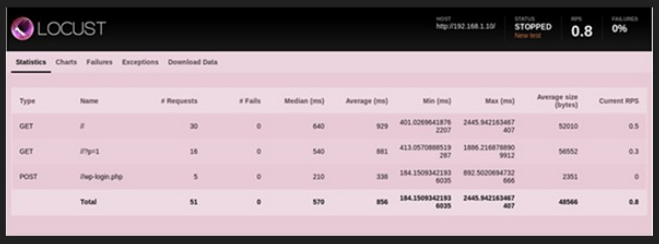
Графіки продуктивності балансувальника навантаження Traefik Load Balancer на статичному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 вимкнений, комп'ютер 2 працює" (зліва), для конфігурації "комп'ютер 1 працює, комп'ютер 2 вимкнений" (посередині) та за алгоритмом Dynamic Round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

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	34	0	2000	4689	626.2259483337402	20907.03105926 5137	51996	0.2
GET	/?p=1	11	0	1600	1559	794.3689823150635	2591.985225677 4902	56520	0.2
POST	/wp-login.php	5	0	14000	13480	12625.99396705 6274	14144.11783218 3838	2347	0
	Total	**50**	**0**	**2000**	**4880**	**626.2259483337 402**	**20907.03105926 5137**	**48026**	**0.4**

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

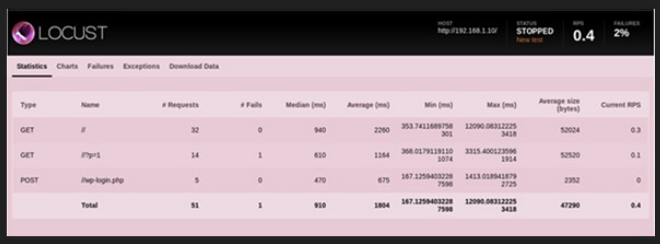
Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	28	1	830	1005	380.0649914550781	2343.137025833 13	50162	0.4
GET	/?p=1	20	1	890	1144	373.2089996337 8906	3127.830028533 9355	53722	0.1
POST	/wp-login.php	5	0	460	500	184.6339702606 2012	947.2730159759 521	2351	0
	Total	**53**	**2**	**840**	**1010**	**184.6339702606 2012**	**3127.830028533 9355**	**46994**	**0.5**

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	413.0570885519 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**

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**

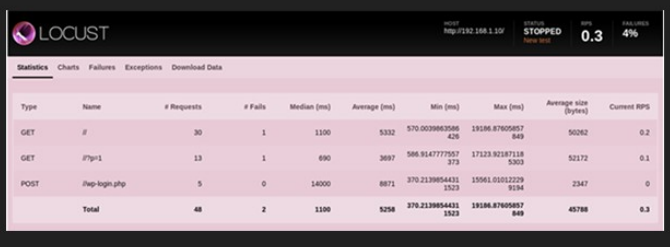
Результати тестування продуктивності балансувальника навантаження HAProxy на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 вимкнений, комп’ютер 2 працює”

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**

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

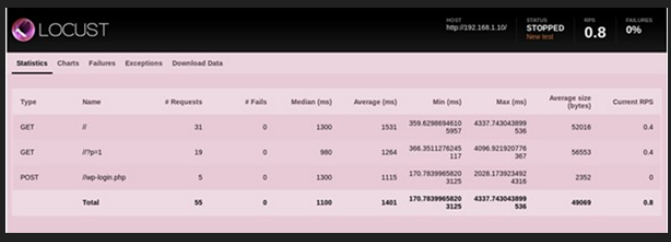
Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	27	4	7000	7738	835.6750011444 092	16521.92902565 0024	44292	0
GET	/?p=1	13	1	6000	7550	633.5608959197 998	15644.39702033 9966	52172	0.1
POST	/wp-login.php	5	0	8300	8216	6742.285013198 8525	9524.546861648 56	2347	0
	Total	**45**	**5**	**7100**	**7737**	**633.5608959197 998**	**16521.92902565 0024**	**41908**	**0.1**

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



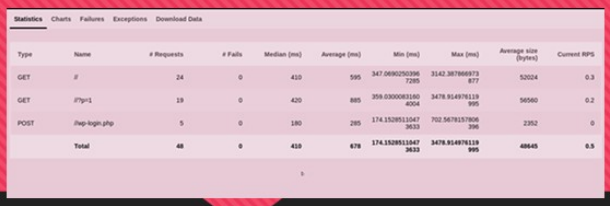
Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	30	1	1100	5332	570.0039863586 426	19186.87605857 849	50262	0.2
GET	/?p=1	13	1	690	3697	586.9147777557 373	17123.92187118 5303	52172	0.1
POST	/wp-login.php	5	0	14000	8871	370.2139854431 1523	15561.01012229 9194	2347	0
	Total	**48**	**2**	**1100**	**5258**	**370.2139854431 1523**	**19186.87605857 849**	**45788**	**0.3**

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



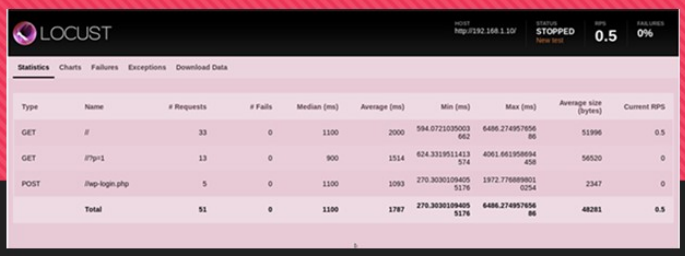
Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	31	0	1300	1531	359.6298694610 5957	4337.743043899 536	52016	0.4
GET	/?p=1	19	0	980	1264	366.3511276245 117	4096.921920776 367	56553	0.4
POST	/wp-login.php	5	0	1300	1115	170.7839965820 3125	2028.173923492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

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	174.1529511047 3633	702.5678157806 396	2352	0
	Total	**48**	**0**	**410**	**678**	**174.1529511047 3633**	**3478.914976119 995**	**48645**	**0.5**

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”

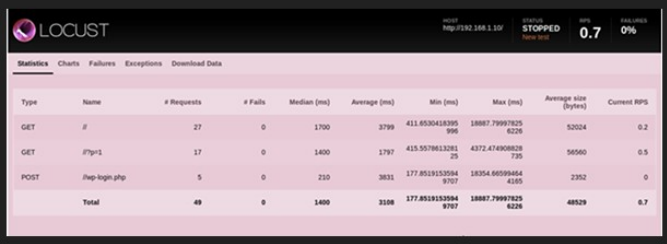


Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	33	0	1100	2000	594.0721035003 662	6486.274957656 86	51996	0.5
GET	/?p=1	13	0	900	1514	624.3319511413 574	4061.661958694 458	56520	0
POST	/wp-login.php	5	0	1100	1093	270.3030109405 5176	1972.776889801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

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	355.7698726654 053	2996.472835540 7715	52016	0.3
GET	/?p=1	21	0	800	1054	357.0899963378 906	2476.346969604 492	56552	0.3
POST	/wp-login.php	5	0	190	196	178.0800819396 9727	226.9659042358 3984	2352	0
	Total	**52**	**0**	**550**	**845**	**178.0800819396 9727**	**2996.472835540 7715**	**49072**	**0.6**

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	411.6530418395 996	18887.79997825 6226	52024	0.2
GET	/?p=1	17	0	1400	1797	415.5578613281 25	4372.474908828 735	56560	0.5
POST	/wp-login.php	5	0	210	3831	177.8519153594 9707	18354.66599464 4165	2352	0
	Total	**49**	**0**	**1400**	**3108**	**177.8519153594 9707**	**18887.79997825 6226**	**48529**	**0.7**

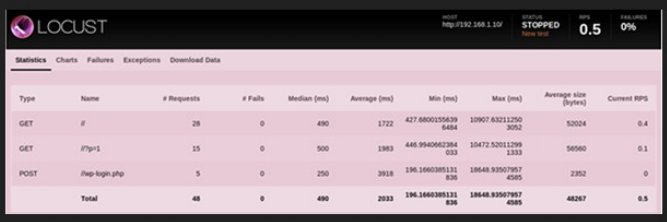
Результати тестування продуктивності балансувальника навантаження Nginx Upstream Module на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації “комп’ютер 1 працює, комп’ютер 2 вимкнений”

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	33	0	2500	2683	587.5980854034 424	6962.255001068 115	51996	0.5
GET	/?p=1	13	0	2700	2994	671.6239452362 06	4912.293910980 225	56520	0
POST	/wp-login.php	5	0	9700	9350	490.8871650695 801	18338.78898620 6055	2347	0
	Total	**51**	**0**	**2700**	**3416**	**490.8871650695 801**	**18338.78898620 6055**	**48281**	**0.5**

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	476.7949581146 24	1923.223972320 5566	52013	0.6
GET	/?p=1	15	0	680	820	485.4149818420 41	2048.344135284 424	56549	0.1
POST	/wp-login.php	5	0	2800	2976	2044.450998306 2744	4691.068172454 834	2351	0
	Total	**62**	**0**	**700**	**940**	**476.7949581146 24**	**4691.068172454 834**	**49105**	**0.7**

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

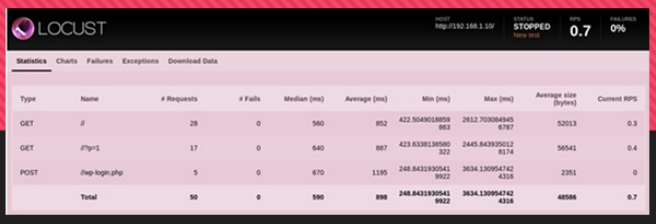


Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	28	0	490	1722	427.6800155639 6484	10907.63211250 3052	52024	0.4
GET	/?p=1	15	0	500	1983	446.9940662384 033	10472.52011299 1333	56560	0.1
POST	/wp-login.php	5	0	250	3918	196.1660385131 836	18648.93507957 4585	2352	0
	Total	**48**	**0**	**490**	**2033**	**196.1660385131 836**	**18648.93507957 4585**	**48267**	**0.5**

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	693.7940120697 021	34198.22192192 078	48529	0.2
GET	/?p=1	14	2	3000	6918	687.3719692230 225	21504.54711914 0625	48445	0
POST	/wp-login.php	5	0	17000	11429	524.3470668792 725	18637.55702972 412	2347	0
	Total	**49**	**4**	**2400**	**6949**	**524.3470668792 725**	**34198.22192192 078**	**43793**	**0.2**

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

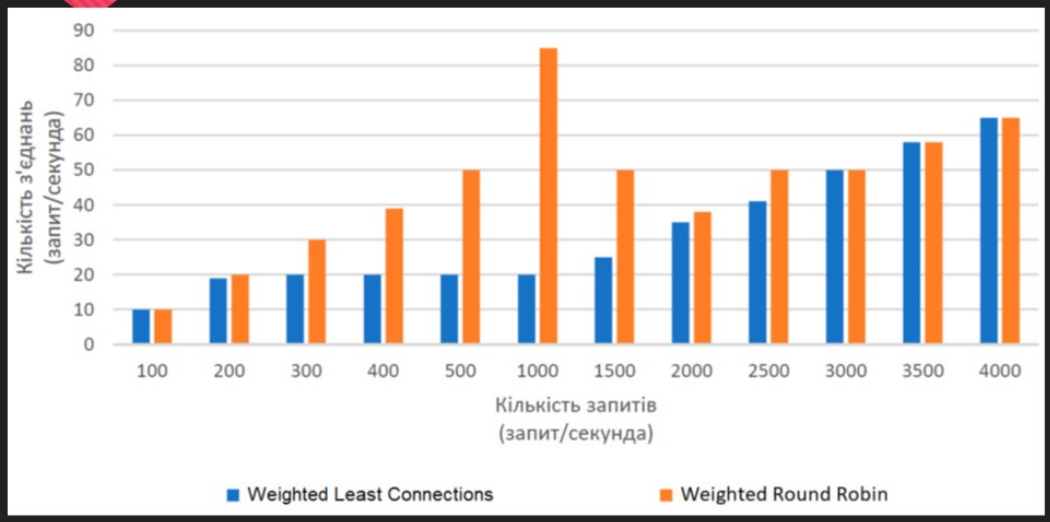


Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює"

Результати тестування продуктивності балансувальника навантаження Traefik на динамічному змісті за алгоритмом Weighted Round Robin для конфігурації "комп'ютер 1 працює, комп'ютер 2 працює, комп'ютер 3 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



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Ільїна Сергія Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Кривченко Анастасія Анатоліївна*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів*

Обсяг розрахунково-пояснювальної записки *71* сторінок

Обсяг графічної (презентаційної) частини *18* аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію кваліфікаційна робота бакалавра повністю відповідає меті випускної роботи та технічному завданню. Тематика кваліфікаційної роботи є актуальною для своєї галузі та присвячена аналізу алгоритмів балансування навантаження при обробці запитів великої кількості користувачів

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається зі вступу, двох розділів, висновків, переліку використаних джерел. У основному розділі виконано аналітичний огляд методів та засобів балансування навантаження; вибір балансувальників навантаження при обробці запитів великої кількості користувачів; підготовка апаратних засобів для дослідження роботи балансувальників навантаження; тестування роботи алгоритмів балансування навантаження при обробці запитів великої кількості користувачів. Розглянуто питання охорони праці та техніки безпеки

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

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана охайно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату ідей у роботі не виявлено

г) перелік позитивних якостей кваліфікаційної роботи

Оцінено чотири популярні рішення (Traefik, HAProxy, Pound, Nginx Upstream), що дозволяє порівняти їхню продуктивність в різних сценаріях. Застосовано стандартизовані інструменти (ApacheBench, Locust, Gnuplot), наведено числові показники середнього часу відповіді, пропускної здатності та відсотка помилок під навантаженням.

д) основні недоліки кваліфікаційної роботи

Хоча наведено графіки, бракує аналізу дисперсії вимірювань, довірчих інтервалів та порівняння з альтернативними конфігураціями (наприклад, без вагової оптимізації та з іншими кількісними метриками: CPU/memory). Деякі діаграми не супроводжені достатнім текстовим коментарем, що ускладнює розуміння логіки.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

« 23 » 08 2025 р.

Національний університет «Одеська політехніка» Інститут комп'ютерних систем

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

керівника про випускну роботу бакалавра

Ільїна Сергія Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Тема випускної роботи *Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів*

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 71 сторінку. У пояснювальній записці розглянуті алгоритми балансування навантаження для підвищення продуктивності web-ресурсів, а також програмні засоби реалізації балансувальників навантаження. Графічна частина складається з 18 слайдів, 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 Ільїн С.С. поступово та послідовно виконував всі етапи, проявив ініціативу у створенні загальної концепції та реалізації випускної роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) Теоретична підготовка здобувача освіти

Ільїн С.С. 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 Ільїн С.С. 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 *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім`я, по батькові *Кривченко Анастасія Анатоліївна*

Місце роботи і посада керівника роботи *ВСП "Одеський технічний фаховий коледж ОНТУ", викладач кафедри комп'ютерної інженерії, голова обласної методичної комісії викладачів комп'ютерної інженерії*

Підпис

«18» 06 2025 р.

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

Ми, що нижче підписалися,

Ільїн С.С.,
здобувач освіти гр. 2БКС-29, та

Кривченко А.А.,
керівник дипломного проекту,

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

«Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів» (автор роботи – Ільїн С.С., керівник роботи – Кривченко А.А.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 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 _____ / Ільїн С.С. /

Керівник _____ / Кривченко А.А. /

«16» червня 2025 р.

ДОВІДКА

кафедри комп'ютерної інженерії
про допуск до захисту кваліфікаційної роботи
здобувача (здобувачки) освіти II курсу
відділення комп'ютерних систем групи 2БКС-29

Ільїна Сергія Сергійовича

на тему *Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до кваліфікаційної роботи виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 03.06.2025 р. значення коефіцієнту подібності в роботі становить 8,89%, коефіцієнт цитування – 4,31%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) кваліфікаційної роботи

здобувача (здобувачки) освіти *Циганова В.С.* (П.І.Б.)

проведена « *20* » *червня* 2025 р.

Висновки *Пояснювальна записка до кваліфікаційної роботи виконана у повному обсязі. Випускна кваліфікаційна робота відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Зав. кафедри КІ (підпис) *Іванова Л.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/3/2025
Дата редагування 6/3/2025

Документ прийнятий

Звіт подібності

метадані

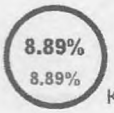
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Аналіз алгоритмів балансування навантаження при обробці запитів великої кількості користувачів

Автор
Ільїн Сергій Сергійович
Науковий керівник / Експерт
Кривченко Анастасія Анатоліївна

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КЦ

25
Довжина фрази для коефіцієнта подібності 2

12425
Кількість слів

102858
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		56

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://thelib.info/psihologiya/190888-vpliv-parametriv-mikroklimatu-na-zahvorjuvanist-koristuvachiv-personalnogo-komp-jutera/	82 0.66 %
2	https://card-file.ontu.edu.ua/server/api/core/bitstreams/d5a3d14f-d5cb-460f-9c49-cba3f9d50554/content	77 0.62 %
3	https://card-file.ontu.edu.ua/bitstreams/11562741-24e6-4201-bc41-a00c8013fca1/download	77 0.62 %
4	https://serversforhackers.com/c/load-balancing-with-haproxy	68 0.55 %

5	https://serversforhackers.com/c/load-balancing-with-haproxy	66 0.53 %
6	https://thelib.info/psihologiya/190888-vpliv-parametriv-mikroklimatu-na-zahvorjuvanist-koristuvachiv-personalnogo-komp-jutera/	66 0.53 %
7	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	50 0.40 %
8	https://lektsii.org/2-52678.html	43 0.35 %
9	https://serversforhackers.com/c/load-balancing-with-haproxy	35 0.28 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/21ac499a-a9e9-4137-810c-5f21a0318048/content	32 0.26 %

з домашньої бази даних (0.05 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.05 %

з програми обміну базами даних (1.08 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Система балансування навантаження веб-серверів [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	106 (12) 0.85 %
2	dm_2023_263_020 [redacted] 8/20/2024 O.M.Beketov National University of Urban Economy in Kharkiv (O.M.Beketov National University of Urban Economy in Kharkiv)	15 (1) 0.12 %
3	Магистр_Полторак_Шатковський_А.pdf [redacted] 12/26/2019 State University of Telecommunications (HHIT)	13 (1) 0.10 %

з Інтернету (7.77 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://thelib.info/psihologiya/190888-vpliv-parametriv-mikroklimatu-na-zahvorjuvanist-koristuvachiv-personalnogo-komp-jutera/**	**179 (3) 1.44 %**
2	**https://serversforhackers.com/c/load-balancing-with-haproxy**	**169 (3) 1.36 %**
3	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	116 (11) 0.93 %
4	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/d5a3d14f-d5cb-460f-9c49-cba3f9d50554/content**	**107 (2) 0.86 %**
5	**https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download**	**102 (5) 0.82 %**
6	**https://card-file.ontu.edu.ua/bitstreams/11562741-24e6-4201-bc41-a00c8013fca1/download**	**97 (3) 0.78 %**
7	**https://lektsii.org/2-52678.html**	**56 (2) 0.45 %**
8	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/21ac499a-a9e9-4137-810c-5f21a0318048/content**	**37 (2) 0.30 %**

9	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	19 (1) 0.15 %
10	https://m.riunet.upv.es/bitstream/handle/10251/106555/RAMOS%20-%20Servidor%20de%20Internet%20de%20alta%20disponibilidad%20con%20equilibrado%20de%20carga.pdf	18 (2) 0.14 %
11	https://card-file.ontu.edu.ua/server/api/core/bitstreams/e86ba9fc-9135-43bb-922a-10bf0bce46b5/content	17 (2) 0.14 %
12	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	13 (2) 0.10 %
13	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c5cd348b-fc64-4a25-9a5b-6cc8d62db909/content	11 (1) 0.09 %
14	https://card-file.ontu.edu.ua/bitstreams/f789da43-3034-4ad8-bf34-640a47414f93/download	10 (1) 0.08 %
15	https://card-file.ontu.edu.ua/bitstreams/d42aac6d-ab01-4a74-b9cb-ced2a9eff719/download	8 (1) 0.06 %
16	http://um.co.ua/13/13-2/13-22602.html	6 (1) 0.05 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

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

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Комп'ютерна інженерія» Група:** 2БКС- 29

КВАЛІФІКАЦІЙНА
РОБОТА

здобувача **освіти денної форми навчання БКС.** 29.10.000.КРБ

ІЛЬЇНА
СЕРГІЯ СЕРГІЙОВИЧА

м. Одеса
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Комп'ютерна інженерія»**
Група: 2 БКС-29

ПОЯСНЮВАЛЬНА **ЗАПИСКА До кваліфікаційної роботи бакалавра на тему:**_____
_____
_____ **Проектний матеріал складається з пояснювальної записки на** _____ **сторінках та графічного (презентаційного) матеріалу на** _____ **аркушах (слайдах)** Виконавець _____ **(** Ільїн С.С.)
_____ **Керівник проекту** _____ **(** Кривченко А.А.)

_____ Консультанти: **з розділу охорони праці та техніки безпеки** _____ **(** Чорновол Н.І.) з нормоконтролю _____ **(** Петрашова **В.І.) старший** консультант _____ **(** Кривченко Ю. В.) **До захисту** допущений
_____ Завідувач кафедри _____ **(Іванова Л.В.) Завідувач** відділення _____ (Краснокутська К.Г.)

_____ Захист «____»_____ **202**____ **р.** Протокол ЕК No_____ **Оцінка** ЕК _____

Секретар ЕК _____

Аналіз алгоритмів балансування
навантаження при обробці запитів великої кількості користувачів
71
18