ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ДОМАШНІЙ СЕРВЕР З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ВІРТУАЛІЗАЦІЇ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми «Комп'ютерна інженерія»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Владислав ПОТАПЕНКО*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КІД-42
Владислав ПОТАПЕНКО
Ім'я, ПРІЗВИЩЕ

Керівник: Віра БАБЕНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерна інженерія

Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Потапенко Владислав Сергійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Домашній сервер з використанням технологій віртуалізації»
керівник кваліфікаційної роботи: Віра БАБЕНКО, доктор технічних наук, професор,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. №56
2. Строк подання кваліфікаційної роботи«05» червня 2025р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, стандарти та протоколи віртуалізації, архітектура та компоненти домашніх серверів, продуктивність і навантаження систем, методи вимірювання та аналізу ефективності, тестування та порівняльний аналіз.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Теоретичні основи віртуалізації в контексті домашніх серверів
    2. Побудова тестового середовища та методика оцінювання продуктивності
    3. Оптимізація, апаратна альтернатива та перспективи розвитку домашніх серверів
5. Перелік ілюстративного матеріалу: *презентація*
    1. Мета, об'єкт, предмет дослідження
    2. Актуальність роботи
    3. Труднощі при побудові традиційного та віртуалізованого сервера
    4. Архітектура домашнього сервера з використанням технологій віртуалізації
    5. Вибір платформи віртуалізації: переваги Proxmox VE
    6. Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: сервер розумного дому
    7. Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: віртуальна машина для тестування
    8. Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: кастомізований маршрутизатор
    9. Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: DNS- та медіа- серверів

10. Спостереження за ресурсоспоживанням всіх вузлів
11. Методи оптимізації: автоматичне очищення кешу у Linux та переналаштування портів
12. Кластеризація з кількох бюджетних пристроїв: переваги над одним потужним сервером
13. Висновки
14. Публікації та апробація роботи

6. Дата видачі завдання «24» лютого 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.02-19.03.2025	
2	Обробка матеріалу	20.03-02.04.2025	
3	Розробка теоретичної частини	03.04-07.04.2025	
5	Побудова тестового середовища та методика оцінювання продуктивності	08.04-27.04.2025	
6	Оптимізація, апаратна альтернатива та перспективи розвитку домашніх серверів	28.04-11.05.2025	
7	Висновки за результатами аналізу	12.05-15.05.2025	
8	Розробка презентації	16.05-26.05.2025	
9	Передзахист	27.05-29.05.2025	
10	Здача в деканат	30.05-05.06.2025	

Здобувач(ка) вищої освіти _____ Владислав ПОТАПЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Віра БАБЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 87 стор., 88 рис., 14 табл., 29 джерел.

Мета роботи – оптимізація продуктивності та ефективності використання ресурсів домашнього сервера за рахунок впровадження та налаштування технологій віртуалізації.

Об'єкт дослідження – процес віртуалізації в домашньому серверному середовищі.

Предмет дослідження – методи та технології віртуалізації, що впливають на продуктивність, стабільність і енергоефективність домашнього сервера.

Короткий зміст роботи: У роботі зосереджено увагу на оптимізації продуктивності домашніх серверів за допомогою віртуалізації. Аналізується вплив різних підходів до розподілу ресурсів, використання кешування, балансування навантаження та кластеризації. Особлива увага приділяється оцінці продуктивності при роботі з потоковим медіа, файловими сховищами, DNS-сервісами, а також впливу вибору обладнання на загальну стабільність системи.

Основна мета – оцінити ефективність віртуалізації в контексті домашнього серверного використання, визначити потенційні вузькі місця та запропонувати методи їх оптимізації, що допоможе користувачам розробити надійні та енергоефективні серверні рішення, що відповідатимуть потребам у домашньому сегменті.

КЛЮЧОВІ СЛОВА: ВІРТУАЛІЗАЦІЯ, ДОМАШНІЙ СЕРВЕР, ПРОДУКТИВНІСТЬ, КОНТЕЙНЕРИЗАЦІЯ, ГІПЕРВІЗОР, ОПТИМІЗАЦІЯ, РЕСУРСИ, ОБЧИСЛЕННЯ, ВІРТУАЛЬНА МАШИНА, ХМАРНІ ТЕХНОЛОГІЇ, НАВАНТАЖЕННЯ, ЕНЕРГОСПОЖИВАННЯ, ЕФЕКТИВНІСТЬ, МАСШТАБОВАНІСТЬ, МЕРЕЖЕВІ СЕРВІСИ, БЕЗПЕКА, ІЗОЛЯЦІЯ, ХОСТИНГ, ТЕСТУВАННЯ, КОНФІГУРАЦІЯ.

ABSTRACT

Text part of the qualification work for the bachelor's degree: 87 pages, 88 figures, 14 tables, 29 sources.

The purpose of the work is to optimize the productivity and efficiency of home server resources by implementing and configuring virtualization technologies.

The object of the study is the virtualization process in a home server environment.

The subject of the study is virtualization methods and technologies that affect the productivity, stability and energy efficiency of a home server.

Summary of the work: The work focuses on optimizing the productivity of home servers using virtualization. The impact of different approaches to resource allocation, the use of caching, load balancing and clustering is analyzed. Special attention is paid to assessing performance when working with streaming media, file storage, DNS services, as well as the impact of equipment selection on the overall stability of the system.

The main goal is to evaluate the effectiveness of virtualization in the context of home server use, identify potential bottlenecks, and propose methods for optimizing them, which will help users develop reliable and energy-efficient server solutions that meet needs in the home segment.

KEYWORDS: VIRTUALIZATION, HOME SERVER, PRODUCTIVITY, CONTAINERIZATION, HYPERVISOR, OPTIMIZATION, RESOURCES, COMPUTING, VIRTUAL MACHINE, CLOUD TECHNOLOGIES, LOAD, ENERGY CONSUMPTION, EFFICIENCY, SCALABILITY, NETWORK SERVICES, SECURITY, ISOLATION, HOSTING, TESTING, CONFIGURATION.

ЗМІСТ

ВСТУП..... 9
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ВІРТУАЛІЗАЦІЇ В КОНТЕКСТІ ДОМАШНІХ СЕРВЕРІВ 10
1.1 Принципи віртуалізації..... 10
1.2 Можливості та перспективи домашніх серверів..... 21
1.3 Технології віртуалізації для домашніх серверів 26
1.4 Вплив віртуалізації на продуктивність домашніх серверів 33
1.5 Виклики впровадження віртуалізації на домашніх серверах. 37
РОЗДІЛ 2 ПОБУДОВА ТЕСТОВОГО СЕРЕДОВИЩА ТА МЕТОДИКА ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ 43
2.1 Апаратна та програмна конфігурація домашнього сервера 43
2.2 Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі 54
2.3 Засоби моніторингу та методи аналізу продуктивності..... 70
2.4 Аналіз продуктивності основних ресурсів сервера..... 76
РОЗДІЛ 3 ОПТИМІЗАЦІЯ, АПАРАТНА АЛЬТЕРНАТИВА ТА ПЕРСПЕКТИВИ РОЗВИТКУ ДОМАШНІХ СЕРВЕРІВ 83
3.1 Оптимізація середовиша віртуалізації на основі тестування. 83
3.2 Альтернативні бюджетні рішення для домашньої віртуалізації..... 88
3.3 Перспективи розвитку віртуалізації домашніх серверів..... 92
ВИСНОВКИ..... 95
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ……..……..96
Додаток А. Навантаження на процесор, ОЗП, диск, мережу та сервер……...98
Додаток Б. Використання оперативної пам’яті.…….……..…....101
Додаток В. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……....102

ВСТУП

Сучасні домашні сервери дедалі частіше потребують ефективного керування ресурсами для підтримки хмарних сховищ, потокового передавання медіа, веб-хостингу та мережевих служб. Віртуалізація дозволяє запускати кілька операційних систем і сервісів на одному пристрої, оптимізуючи використання обладнання та підвищуючи гнучкість управління. Раніше ця технологія використовувалася переважно в центрах обробки даних, але сьогодні вона стала доступною і для домашніх рішень, допомагаючи максимально ефективно використовувати апаратні можливості.

Попри численні переваги, віртуалізація створює і певні виклики. Високе навантаження на процесор, обмеження оперативної пам’яті, затримки у мережевій взаємодії та загальні витрати енергії можуть впливати на стабільність і продуктивність системи. Вибір між гіпервізорами, такими як Proxmox, та контейнеризацією (Docker, LXC) визначає баланс між ізоляцією сервісів і ефективністю використання ресурсів. Для домашнього сервера важливо знайти оптимальне рішення, яке поєднує продуктивність, безпеку та енергоефективність.

Дослідження ставить за мету розгляд підходів до оптимізації віртуалізованих середовищ, зокрема балансування продуктивності, ефективного розподілу ресурсів та енергоспоживання. Аналізуються різні методи налаштування серверної інфраструктури, що дозволяють підвищити стабільність і швидкодію системи.

Отримані результати продемонструють можливості покращення роботи домашніх серверів за рахунок гнучкого управління ресурсами та вибору відповідної конфігурації. Висновки можуть бути корисними для подальших досліджень та практичного впровадження оптимізованих рішень у віртуалізації.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ВІРТУАЛІЗАЦІЇ В КОНТЕКСТІ ДОМАШНІХ СЕРВЕРІВ

1.1 Принципи віртуалізації

Віртуалізація стала ключовим рішенням для оптимізації обчислювальних ресурсів у середовищах, де використовується велика кількість комп’ютерів, таких як офіси, комп’ютерні класи та серверні системи. Коли йдеться про домашнє використання одного або кількох комп’ютерів, проблеми адміністрування та налаштування не є настільки критичними. Однак зі збільшенням кількості пристроїв зростають витрати часу на їх обслуговування та налаштування, що ускладнює управління всією інфраструктурою.

Щоб уникнути цих труднощів, застосовуються технології віртуалізації, які дають змогу створювати віртуальні версії обчислювальних ресурсів, включаючи операційні системи, апаратні платформи, пристрої зберігання даних і мережеві ресурси. Завдяки цьому фізичні компоненти абстрагуються, що забезпечує ефективне використання апаратного забезпечення та підвищує гнучкість системи.

Замість того щоб обслуговувати велику кількість окремих пристроїв, достатньо розгорнути одну фізичну платформу, на якій працюватимуть віртуальні машини або контейнери. Вони можуть виконувати різні операційні системи та програмне забезпечення, до яких користувачі підключаються віддалено, що значно спрощує управління інфраструктурою, знижує витрати на обладнання та мінімізує час, необхідний для підтримки системи.

Основна ідея віртуалізації полягає у відокремленні обчислювальних функцій від фізичних обмежень. Завдяки цьому досягається ефективне управління ресурсами, підвищується безпека та спрощується адміністрування, що особливо корисно навіть у домашньому середовищі, коли виникає потреба запускати кілька програм або служб без використання окремих фізичних машин. Віртуалізація дає

змогу створювати клоновані середовища, які працюють у форматі пісочниці, що дозволяє запускати кілька версій одного додатка під різними обліковими записами (див. рис. 1.1) [1].



Рисунок 1.1 – Відокремлення обчислювальних ресурсів
у різні віртуалізовані середовища

Віртуалізація дозволяє швидко розгортати та згортати операційні системи, забезпечуючи віддалений доступ користувачам, таким як співробітники або студенти. Для підключення можуть використовуватися малопотужні пристрої на зразок тонких клієнтів, які не мають власних накопичувачів і використовують обчислювальні ресурси сервера. Завдяки цьому досягається значна економія на обладнанні та підвищується ефективність роботи всієї системи [1, 11].

Ключові поняття віртуалізації. Щоб повністю зрозуміти віртуалізацію, необхідно розглянути її основні принципи та компоненти, з яких вона складається. До ключових понять віртуалізації належить гіпервізор — спеціалізоване програмне забезпечення, що створює віртуальні машини й керує ними. Гіпервізор розподіляє доступні ресурси, такі як процесор, оперативна пам’ять та сховища, між віртуальними машинами, забезпечуючи при цьому їхню ізоляцію. Існує два основних типи гіпервізорів (див. рис. 1.2) [1]:

— гіпервізор 1-го типу;
— гіпервізор 2-го типу.

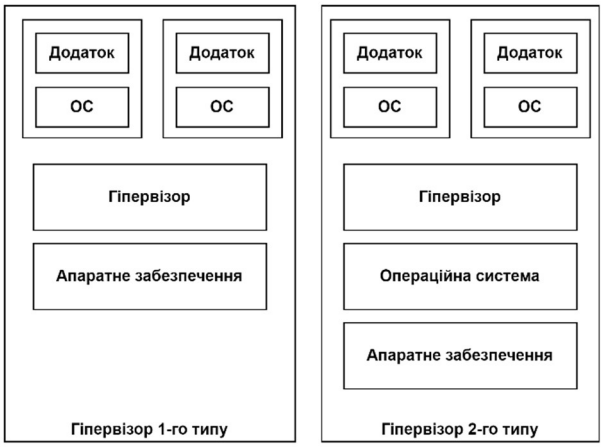


Рисунок 1.2 – Типи гіпервізорів у віртуалізованому середовищі

Перший тип, або «гіпервізор на голому залізі», працює безпосередньо на фізичному обладнанні, що забезпечує високу продуктивність і безпеку. До цього типу належать рішення (див. рис. 1.3) [1, 10]:

— VMware ESXi;
— Proxmox Virtual Environment;
— Microsoft Hyper-V.



Рисунок 1.3 – Популярні середовища, побудовані на гіпервізорі 1-го типу: Vmware ESXI, Proxmox, Microsoft Hyper-V

Використання такого гіпервізора особливо доцільне для серверних рішень, оскільки воно забезпечує максимально ефективний розподіл ресурсів і продуктивність роботи системи. Другий тип гіпервізора працює поверх операційної системи, роблячи процес віртуалізації більш доступним, але менш ефективним. Прикладами такого типу (див. рис. 1.4) [1, 10]:

— VirtualBox;

— VMware Workstation.



Рисунок 1.4 – Популярні середовища, побудовані на гіпервізорі 2-го типу: VirtualBox, VMware Workstation

Такі рішення чудово підходять для тестування програм, але в реальних умовах їхня продуктивність поступається першому типу гіпервізорів [1, 10].

У разі використання гіпервізорів другого типу є також поняття Host OS (див. рис. 1.2) — операційної системи, що керує фізичним обладнанням і взаємодіє з гіпервізором. У гіпервізорах першого типу Host OS відсутня, оскільки сам гіпервізор напряму керує апаратними ресурсами, що суттєво покращує продуктивність [1].

Контейнери, на відміну від віртуальних машин, можуть використовувати спільне ядро операційної системи, що суттєво знижує споживання ресурсів і підвищує продуктивність. Найпопулярнішими технологіями контейнеризації (див. рис. 1.5) [1, 3]:

— Docker;

— Linux Containers (LXC).

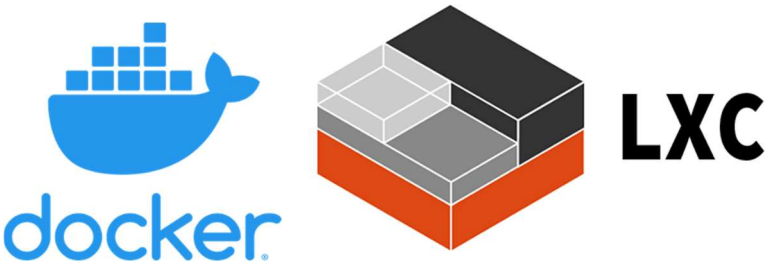


Рисунок 1.5 – Популярні технології контейнеризації: Docker, Linux Containers (LXC)

Основні переваги контейнерів полягають у швидкості запуску, ефективнішому використанні ресурсів та зменшеній накладній витраті в порівнянні з віртуальними машинами. Однак контейнеризація має й свої недоліки, зокрема меншу ізоляцію та потенційні ризики безпеки в разі неправильного налаштування [3, 12].

Класифікація технологій віртуалізації. Технології віртуалізації класифікуються залежно від способу управління ресурсами та взаємодії з обладнанням. Основні види включають (див. рис. 1.6) [1-3]:

— повну віртуалізацію;

— паравіртуалізацію;

— контейнеризацію.

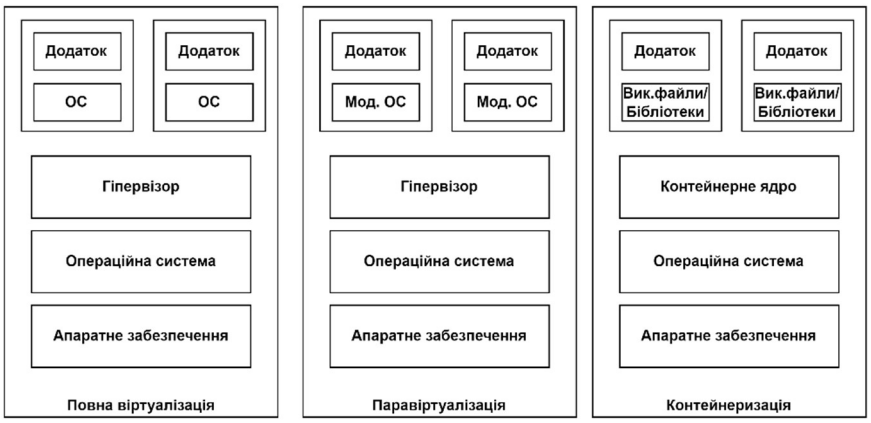


Рисунок 1.6 – Класифікація технологій віртуалізації:
повна віртуалізація, паравіртуалізація, контейнерезація

У повній віртуалізації відбувається повна емуляція апаратного забезпечення, що дозволяє запускати немодифіковані гостьові операційні системи. Гіпервізор у такому випадку перехоплює та транслює всі інструкції між віртуальною машиною та фізичним обладнанням, що забезпечує високу сумісність і сильну ізоляцію між віртуальними машинами, проте потребує значних обчислювальних ресурсів [1-2].

Паравіртуалізація, на відміну від повної віртуалізації, змінює гостьову операційну систему для кращої взаємодії з гіпервізором, що знижує витрати на

обробку та покращує продуктивність. Серед прикладів такої технології (див. рис. 1.7) [1-2, 10]:

— Xen;

— VMware ESXi з паравіртуалізованими драйверами.



Рисунок 1.7 – Популярні рішення паравіртуалізації:
Xen, VMware ESXi з паравіртуалізованими драйверами

Основною перевагою паравіртуалізації є підвищена ефективність, проте для роботи необхідна модифікація гостьової операційної системи, що обмежує сумісність. Окрім класичної віртуалізації, існують (див. рис. 1.8) [1, 3]:

— віртуалізація сховищ;

— віртуалізація мережі.

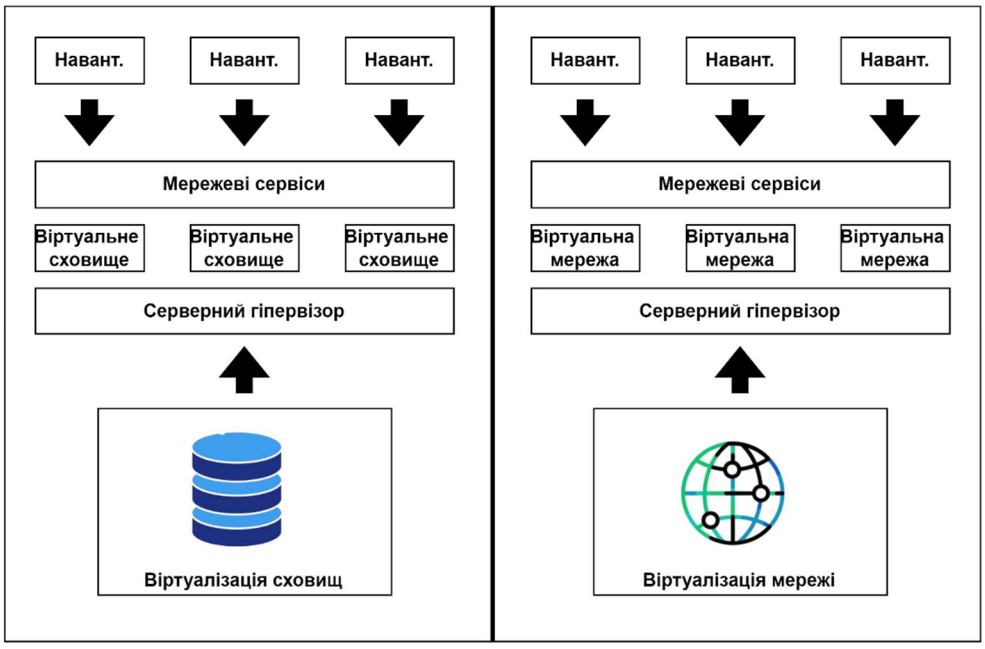


Рисунок 1.8 – Архітектура віртуалізації сховищ та мережі

Віртуалізація сховищ дозволяє створювати уніфіковані системи зберігання, що забезпечує кращу продуктивність, резервування та зручність керування даними. Прикладами таких рішень (див. рис. 1.9) [3, 10]:

— VMware vSAN;

— Ceph;

— TrueNAS.



Рисунок 1.9 – Популярні рішення віртуалізації сховищ:
VMware vSAN, Ceph, TrueNAS

Основні переваги віртуалізації сховищ включають спрощення управління сховищем і кращу масштабованість, проте налаштування таких систем може бути складним і вимагати додаткових знань [3].

Віртуалізація мережі дозволяє створювати віртуальні версії мережевих пристроїв, таких як комутатори, маршрутизатори та брандмауери, що дає змогу гнучкіше налаштовувати мережеву інфраструктуру у віртуалізованих середовищах, підвищуючи безпеку та ефективність.

Серед прикладів таких рішень (див. рис. 1.10) [3, 11, 15-16]:

— Open vSwitch;

— VMware NSX;

— віртуальні мережі Proxmox (VN).



Рисунок 1.10 – Популярні рішення віртуалізації мереж:
Open vSwitch, VMware NSX, Proxmox VN

Об’єднання різних технологій віртуалізації дозволяє створювати ефективні й масштабовані рішення як для комерційного, так і для домашнього використання. Домашні сервери, хоча й менш поширені, ніж корпоративні, можуть значно розширити можливості мережевого середовища та забезпечити ефективне використання ресурсів. Сучасні технології дають змогу розгорнути сервер навіть на компактних пристроях, таких як тонкі клієнти чи Mini PC, що дозволяє створити гнучку інфраструктуру без значних капіталовкладень [1-3, 8-9, 10-11].

Розробка підходів до віртуалізації в IT. Розробка підходів, які призвели до появи віртуалізації в інформаційних технологіях, суттєво еволюціонувала протягом багатьох років: від ранніх мейнфреймів до сучасних хмарних рішень. Розуміння цієї еволюції дає змогу простежити, як вона була побудована, звідки почалася і до чого привела. Віртуалізація набула особливої важливості як у корпоративному середовищі, так і серед домашніх користувачів, що стало поштовхом до зростання популярності домашніх серверів [4, 1].

Ще в 60-х роках минулого століття компанія IBM заклала основи віртуалізації мейнфреймів, що дозволяло запускати ізольовані робочі навантаження на одній системі, який отримав назву «розподіл часу», що дало змогу ефективніше використовувати обчислювальні ресурси, які на той момент коштували дуже дорого. Пізніше, у 90-х роках, відбувся новий етап розвитку віртуалізації, коли почали активно розповсюджуватися персональні комп’ютери на архітектурі x86. Спочатку ця 32-бітна система не була розроблена для ефективної віртуалізації, проте програмні рішення на кшталт VMware Workstation і Microsoft

Virtual PC відкрили можливість запускати кілька операційних систем на одному ПК (див. рис. 1.11) [4, 1].

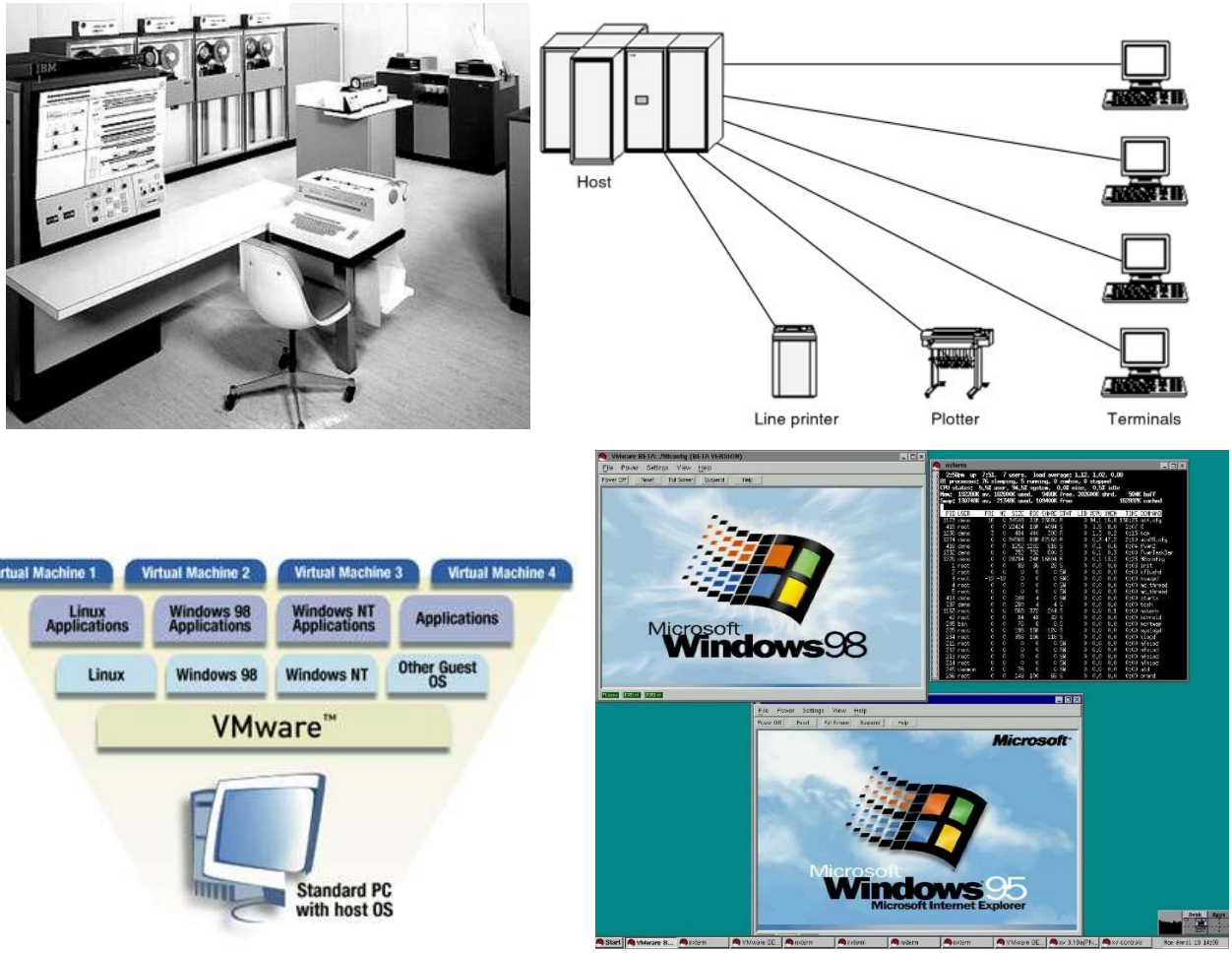


Рисунок 1.11 – Від віртуалізації мейнфреймів (розміром у кімнату) до операційних систем на одному ПК

Згодом виникла потреба у підвищенні продуктивності віртуальних машин, оскільки вони працювали повільніше за звичайні системи. Виробники процесорів почали впроваджувати підтримку апаратної віртуалізації—Intel VT-x і AMD-V, що суттєво покращило ефективність віртуалізованих середовищ. На початку 2000-х апаратна підтримка стала стандартом, що відкрило шлях до створення потужніших гіпервізорів, зокрема KVM (Kernel-based Virtual Machine) (див. рис. 1.12) [4, 1, 10].

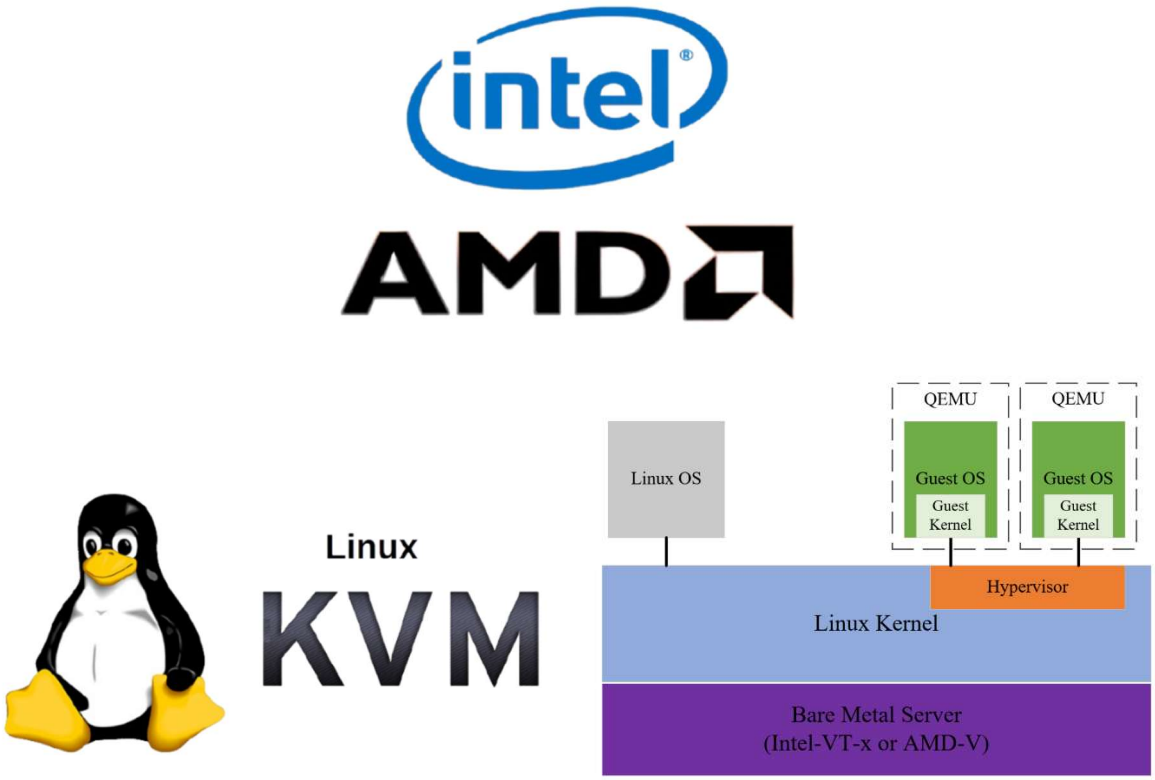


Рисунок 1.12 – Підтримка віртуалізації на апаратному рівні та створення гіпервізора KVM: Intel VT-x та AMD-V

Попри активний розвиток технології, певний час не відбувалося суттєвих проривів, поки у 2010-х роках не розпочався бум хмарних обчислень. Компанії на кшталт Amazon (AWS), Microsoft і Google Cloud почали масово впроваджувати віртуалізацію для надання масштабованих обчислювальних ресурсів на вимогу, що призвело до зростання популярності платформ з відкритим кодом, таких як Proxmox Virtual Environment, які стали економічно вигідним рішенням для підприємств і домашніх користувачів (див. рис. 1.13) [4, 1, 10, 15].

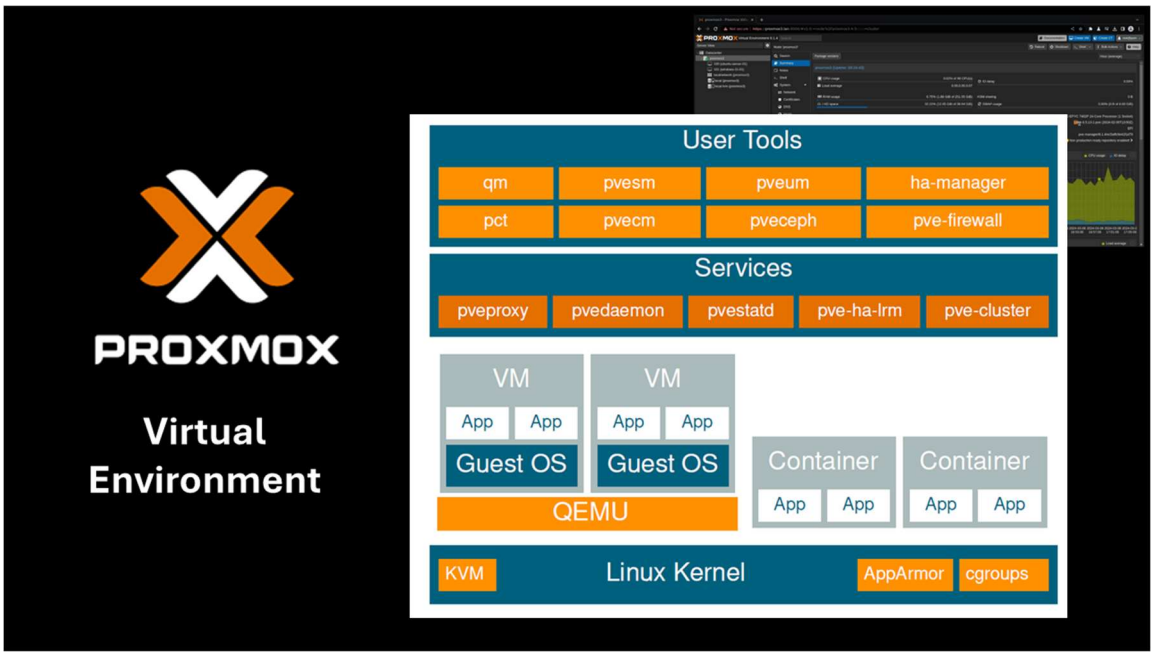


Рисунок 1.13 – Потужне рішення з відкритим програмним кодом: Proxmox Virtual Environments (VE)

Популяризація хмарних обчислень значною мірою сприяла розвитку віртуалізації на домашніх серверах. Якщо спочатку це залишалося нішевим рішенням, то з появою технологій контейнеризації, мініатюризації пристроїв та загального прогресу у сфері споживчого обладнання ситуація змінилася. Вже не виникає питання про те, що старі комп’ютери необхідно просто списувати – тепер вони можуть виконувати роль повноцінних серверів. Proxmox, Docker, Linux-контейнери на базі дистрибутивів Linux дають змогу розгортати різні служби, зокрема для зберігання даних, запуску медіа-сервісів, веб-хостингу чи середовищ розробки [4, 1, 8-9, 10, 15].

1.2 Можливості та перспективи домашніх серверів

Розвиток віртуалізації не просто зменшив кількість фізичних машин, а відкрив нові можливості, особливо в контексті домашніх серверів. Такі системи створюють централізоване середовище для виконання різноманітних завдань, працюючи у фоновому режимі безперервно, що робить їх відмінними від звичайних персональних комп'ютерів.

Основні функції, які можна реалізувати [4, 6-7]:

— зберігання файлів;

— хмарну синхронізацію;

— потокову передачу медіа;

— управління мережею;

— автоматизацію розумного будинку.

Типові сценарії використання. Одним із найбільш поширених сценаріїв є зберігання даних, де рішення на базі NAS забезпечують централізоване управління файлами та їхній обмін між різними пристроями. Таке рішення дозволяє автоматизувати резервне копіювання важливих даних, а також підвищити надійність сховища шляхом використання конфігурацій RAID (див. рис. 1.14) [6-7].



Рисунок 1.14 – Сценарій використання домашнього сервера:
мережеве сховище TrueNAS зі вбудованою інтеграцією різних додатків

Використовуючи домашній сервер, можна створити власний медіацентр із персональною бібліотекою фільмів, музики та фотографій, що дає можливість потокового передавання контенту всередині домашньої мережі або віддаленого доступу до мультимедійних файлів із будь-якого пристрою. Додатково, сервер здатний перекодовувати медіаконтент у реальному часі, забезпечуючи його сумісність із різними пристроями (див. рис. 1.15) [3, 6-7, 21].

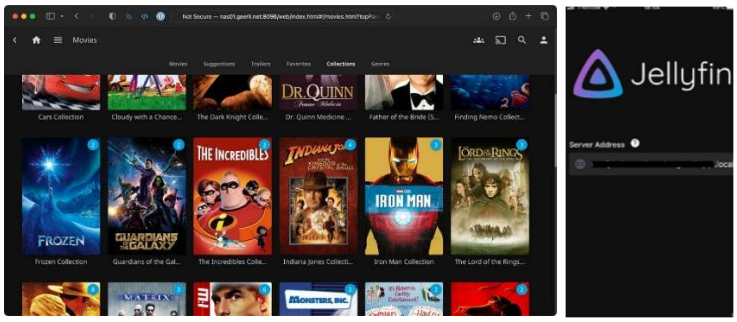


Рисунок 1.15 – Сценарій використання домашнього сервера:
мережеве сховище TrueNAS зі вбудованою інтеграцією різних додатків

Розгортання власних хмарних сервісів також є поширеною практикою, оскільки багато користувачів прагнуть більшої конфіденційності та контролю над власними даними. Використовуючи такі рішення, як Nextcloud, можна налаштувати безпечну синхронізацію файлів між пристроями, організувати спільну роботу над документами, а також користуватися додатковими функціями, такими як відеозв’язок і календарі (див. рис. 1.16) [3, 6-7].



Рисунок 1.16 – Сценарій використання домашнього сервера:
хмарне сховище Nextcloud зі функціями для співпраці

Домашній сервер може також виконувати роль віртуальної лабораторії, що особливо корисно для IT-фахівців, студентів і розробників. У цьому середовищі можна тестувати різні операційні системи, створювати віртуальні машини, керувати мережами та серверами для навчання або експериментування. Віртуалізація дає змогу проводити практичні дослідження в контрольованому середовищі без ризику пошкодити основну систему (див. рис. 1.17) [1-3, 6-7, 11].

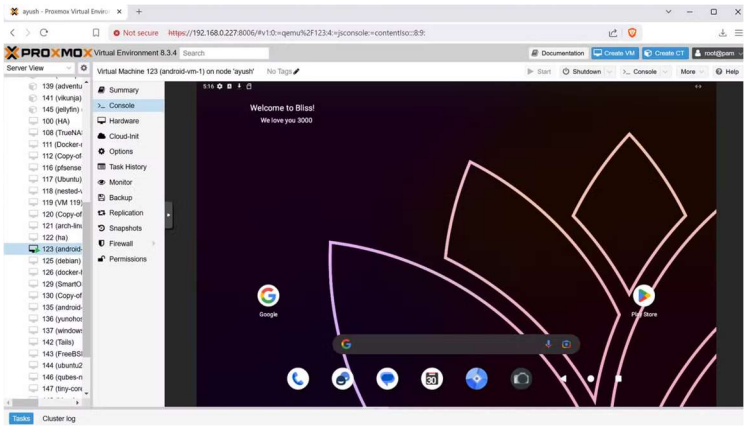


Рисунок 1.17 – Сценарій використання домашнього сервера:
домашня лабораторія Proxmox для тестування та набуття практичних навичок

Домашній сервер може працювати як мережевий екран із встановленим фаєрволом, VPN-сервером і засобами моніторингу трафіку, що дозволяє контролювати потоки даних, виявляти потенційні загрози, блокувати фішингові сайти та підвищувати загальний рівень безпеки в домашній мережі (див. рис. 1.18) [1-2, 6-7].

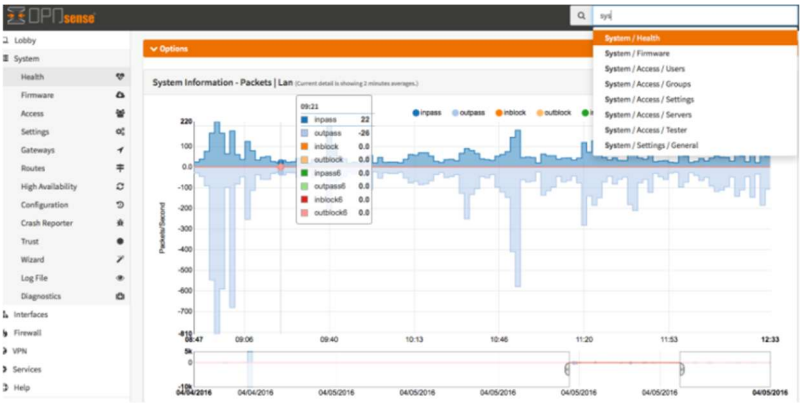


Рисунок 1.18 – Сценарій використання домашнього сервера: фаєрвол OPNsense

Застосування домашнього сервера не обмежується лише наведеними сценаріями. Кожен користувач може адаптувати систему під власні потреби, поєднуючи різні технології та програмні рішення для досягнення конкретних цілей.

Вимоги до обладнання та програмного забезпечення. Налаштування домашнього сервера потребує відповідного обладнання та програмного забезпечення, які повинні відповідати конкретним сценаріям використання.

Таблиця 1.1 – Вимоги до обладнання для віртуалізації домашнього сервера

Компонент	**Рекомендації / Вимоги**
Процесор (CPU)	Багатоядерний, з підтримкою віртуалізації (Intel VT-x, AMD-V).
Оперативна пам’ять (RAM)	Мінімум 8 ГБ для базових задач, бажано 16–32 ГБ для стабільної роботи з кількома сервісами
Накопичувачі	SSD (120–240 ГБ) для ОС та програм; HDD для зберігання даних і резервного копіювання
Мережева підсистема	Гігабітний Ethernet (RJ-45), бажано 2 і більше порти (в т.ч. 2,5 Гбіт/с) для розширених конфігурацій
Альтернативні пристрої	Raspberry Pi — для простих задач; NVIDIA Jetson Nano — для AI/ML;
Масштабування	Підтримка кластеризації для розподілу навантаження і підвищення відмовостійкості

Вибір компонентів (див. табл. 1.1) залежить від поставлених завдань, оскільки для одних користувачів буде достатньо одноплатного комп’ютера на зразок Raspberry Pi, тоді як іншим знадобляться потужніші рішення, наприклад, NVIDIA Jetson Nano, оптимізований для роботи з нейронними мережами та штучним інтелектом. Одним із основних факторів є вибір процесора, який має бути багатоядерним і підтримувати технологію віртуалізації, наприклад, Intel VT-x або AMD-V. Сучасні процесори, такі як Intel Core i5, i7 8-10 покоління або AMD Ryzen 4-6 серії, забезпечують достатню продуктивність для більшості завдань [8-9].

З точки зору пам’яті мінімально рекомендованим обсягом для простих серверних завдань є 8 ГБ оперативної пам’яті, проте для комфортної роботи з декількома сервісами варто розглянути варіанти з 16 або навіть 32 ГБ. Велика кількість запущених служб та сценаріїв збільшує потребу в пам’яті, тому її

розширення дозволяє ефективніше розподіляти ресурси та забезпечує стабільність роботи серверу, який функціонує цілодобово.

Щодо накопичувачів оптимальним варіантом є комбінація SSD та HDD. Твердотільні накопичувачі використовуються для операційної системи та часто використовуваних програм, тоді як жорсткі диски – для зберігання великих обсягів даних. Мінімальний рекомендований обсяг SSD – 120-240 ГБ, що дозволяє швидко завантажувати систему та додатки, а HDD використовується для створення резервних копій та архівного зберігання файлів.

Бажано використовувати гігабітний Ethernet (RJ-45), а за можливості – пристрої з двома і більше портами, що відкриває можливості для розширених мережевих конфігурацій, таких як створення маршрутизатора або міжмережевого екрану за допомогою pfSense чи OPNsense. Деякі Mini PC вже оснащені кількома портами RJ-45 із пропускною здатністю 2,5 Гбіт/с, що дозволяє створювати ефективні та надійні мережеві рішення.

Таблиця 1.2 – Вимоги до ПЗ для віртуалізації домашнього сервера

Категорія	**Рекомендації / Варіанти**
Гіпервізори / Віртуалізація	Proxmox VE (підтримує VM та контейнери); KVM/QEMU (висока продуктивність); VirtualBox, VMware Workstation — менш ефективні
Операційні системи	Debian, Ubuntu Server (сумісні з Docker)
Контейнеризація	Docker — для спрощеного розгортання сервісів
Файлові сховища	TrueNAS — гнучке керування даними
Особливості	Можливість роботи з кластером; Підтримка безперервності сервісів у разі збою вузла

Програмне забезпечення для серверів менш вимогливе, проте вибір залежить від конкретних потреб (див. табл. 1.2). Гіпервізори відіграють ключову роль у віртуалізації, серед популярних варіантів – Proxmox Virtual Environment, який підтримує як віртуальні машини, так і контейнери Linux. VirtualBox і VMware Workstation менш ефективні для серверного використання, оскільки працюють поверх основної операційної системи, а KVM/QEMU, інтегрований у Linux,

забезпечує високу продуктивність і підходить для розгортання серверних середовищ [1-3, 8-9, 10].

Серед операційних систем найбільш популярними є Debian або Ubuntu Server, які дозволяють встановлювати Docker для контейнеризації додатків. Контейнери спрощують розгортання служб без необхідності запуску повноцінних віртуальних машин. Для створення файлового сховища доцільно використовувати TrueNAS, який забезпечує гнучке керування збереженими даними, або Unraid, який має платну модель, проте надає розширені можливості управління [1-3, 6-7, 8-9].

При виборі апаратного забезпечення та програмних рішень важливо враховувати бюджет. Для зниження витрат можна використовувати старі або відновлені комп’ютери, які вже не виконують основних завдань. Альтернативним варіантом є Mini PC або тонкі клієнти, які можна об’єднати в кластер для підвищення відмовостійкості. Використання кількох пристроїв у кластерній конфігурації дозволяє рівномірно розподіляти навантаження та забезпечує безперервність роботи сервісів у разі виходу з ладу одного з вузлів [8-9].

Налаштування домашнього сервера – це процес, який може бути адаптований під конкретні потреби, починаючи з мінімальних конфігурацій та поступово розширюючись до більш складних і масштабованих рішень.

1.3 Технології віртуалізації для домашніх серверів

Популярні платформи віртуалізації дозволяють запускати кілька віртуальних машин або контейнерів на одному фізичному сервері, що забезпечує гнучкість у розподілі завдань. Контейнеризація у багатьох випадках надає більшу ефективність порівняно з класичною віртуалізацією, проте використання віртуальних машин залишається актуальним. Віртуалізація дозволяє створювати повноцінні ізольовані середовища, всередині яких можуть запускатися контейнери, що забезпечує додатковий рівень безпеки та гнучкості [1-3, 10, 11].

Запуск віртуальних машин та контейнерів у межах одного сервера надає можливість ефективного розподілу ресурсів, мультизадачності та адаптації до різних сценаріїв використання. Вибір між віртуальними машинами та контейнерами залежить від поставлених завдань, доступного обладнання та необхідного рівня ізоляції для кожного конкретного випадку [3, 10, 11].

Популярні платформи віртуалізації. Серед поширених платформ віртуалізації, придатних для домашніх серверних середовищ, варто відзначити VMware ESXi — гіпервізор корпоративного рівня, що забезпечує потужні функції віртуалізації, включаючи розширене керування ресурсами, високу доступність та можливості створення знімків. Надійність та оптимізація для корпоративних середовищ дозволяють ефективно використовувати ресурси, мінімізуючи накладні витрати (див. рис. 1.19) [1-3, 10, 11].

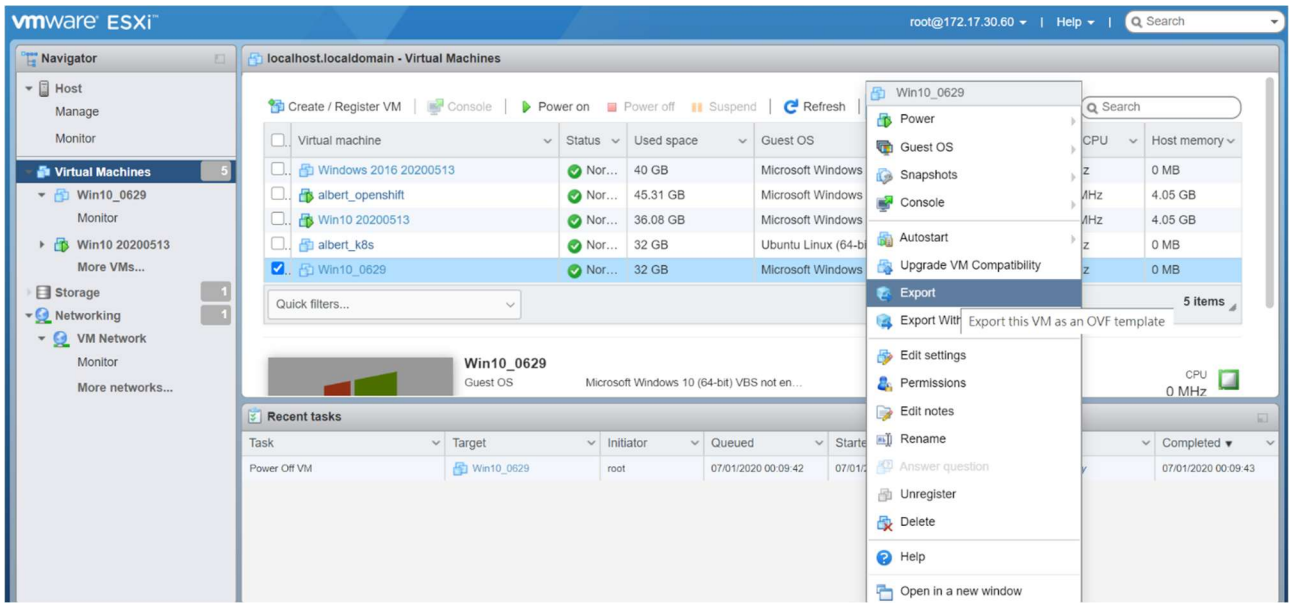


Рисунок 1.19 – Веб-інтерфейс платформи VMware ESXi

Разом із цими перевагами платформа підтримує такі функції, як оперативна міграція vMotion, знімки та кластеризація, що є популярним рішенням для об’єднання великої кількості серверів у єдину систему. Висока сумісність із широким спектром апаратного та програмного забезпечення забезпечує стабільну роботу в різних середовищах (див. рис. 1.20) [10, 11].

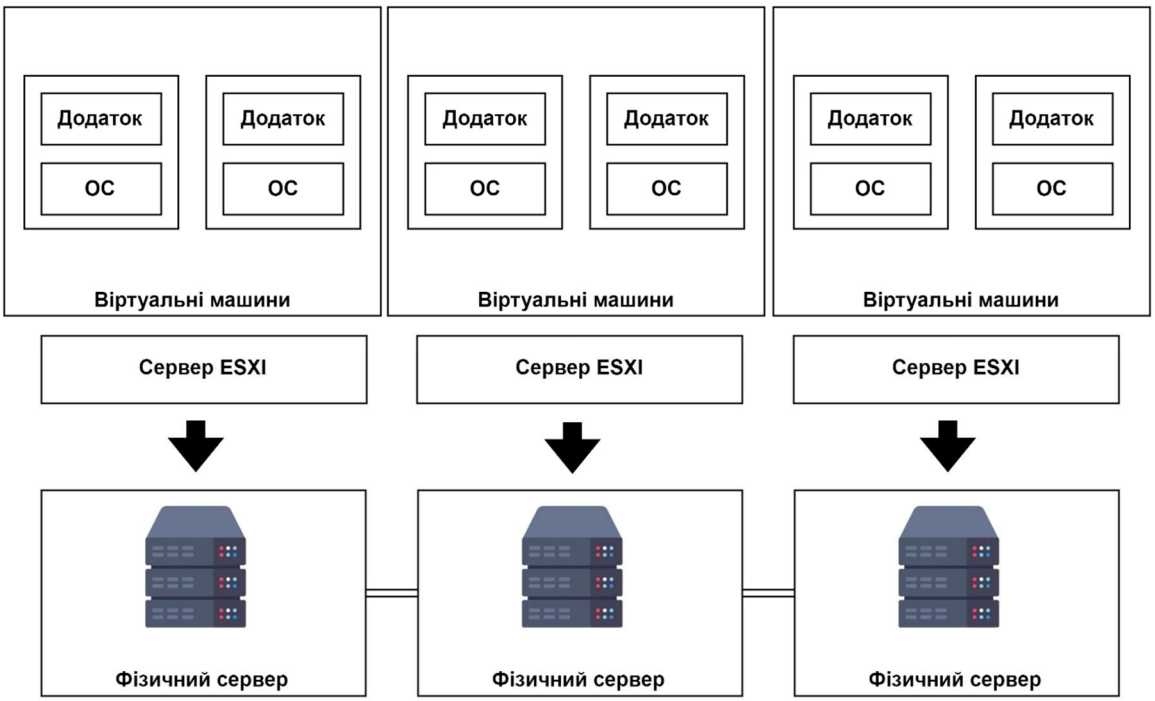


Рисунок 1.20 – Архітектура платформи VMware ESXi

Серед недоліків варто виділити значну ресурсомісткість, адже для ефективної роботи потрібен потужний процесор та значний обсяг оперативної пам’яті. Безкоштовна версія ESXi має певні обмеження, що можуть суттєво впливати на можливості використання, адже багато додаткових функцій доступні лише у платній версії. Крім цього, складність налаштування та управління може стати бар’єром для початківців, оскільки платформа вимагає глибокого розуміння екосистеми VMware. Попри всі переваги, висока вартість ліцензування та системні вимоги роблять це рішення менш привабливим для домашнього використання, тому для серверів у таких умовах доцільніше розглядати альтернативні варіанти [10, 11, 12].

Одним із таких рішень є Proxmox — платформа віртуалізації з відкритим вихідним кодом, яка поєднує можливості KVM для повної віртуалізації та LXC для контейнеризації. Повністю безкоштовна система, що має активну підтримку спільноти, пропонує широкий набір функцій без необхідності платного ліцензування, хоча комерційна версія з підтримкою Enterprise доступна за потреби (див. рис. 1.21) [1-3, 10, 15-16].

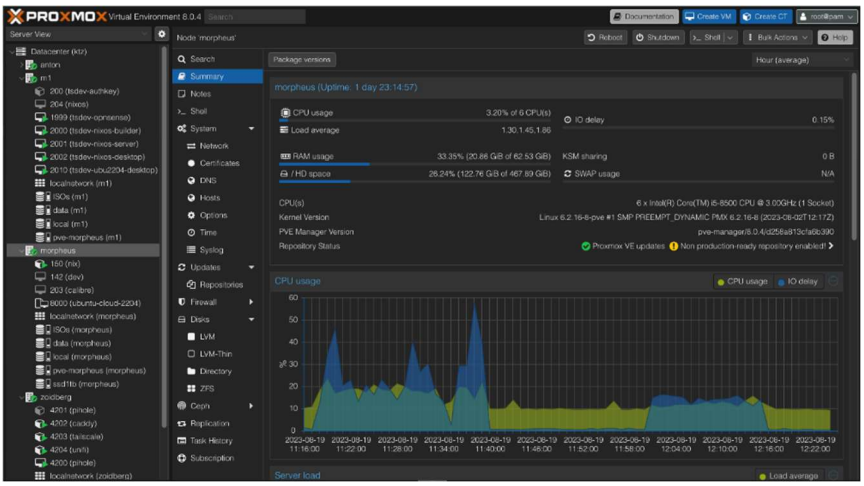


Рисунок 1.21 – Інтерфейс платформи Proxmox VE

Гібридний підхід до віртуалізації дозволяє використовувати як віртуальні машини, так і контейнери, що дає змогу ефективно адаптувати сервер до різних навантажень. Управління здійснюється через веб-інтерфейс, що спрощує адміністрування, а динамічний розподіл ресурсів дозволяє оптимізувати роботу системи (див. рис. 1.22) [3, 10, 15-16].

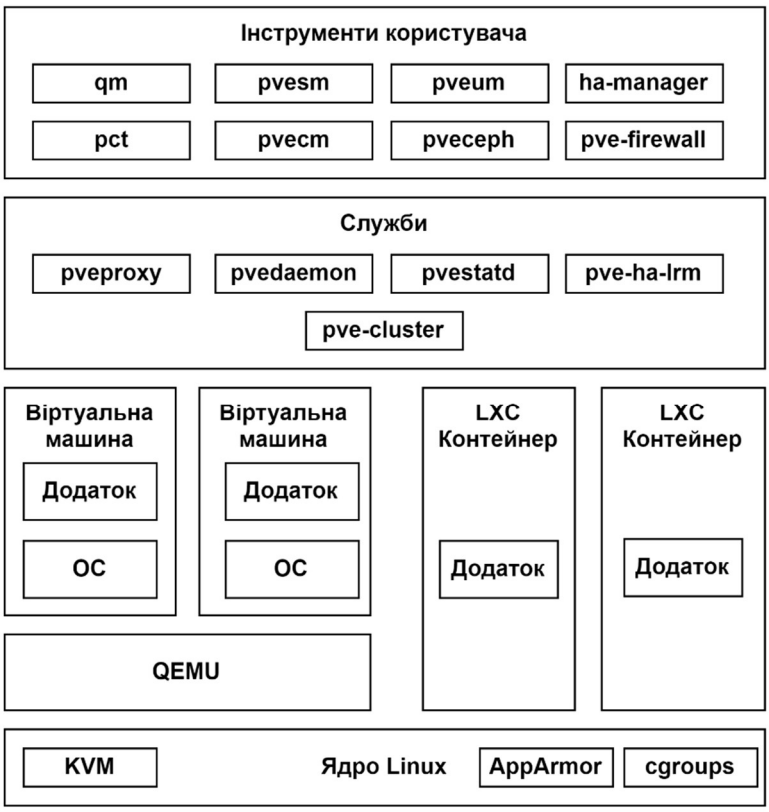


Рисунок 1.22 – Архітектура платформи Proxmox VE

Серед недоліків можна відзначити складність у порівнянні з VirtualBox, адже для налаштування та обслуговування потрібні базові знання Linux та розуміння мережевих технологій. Додатково можуть виникати труднощі із сумісністю деяких апаратних компонентів, хоча більшість сучасного обладнання підтримується без проблем [10, 11, 15-16].

Proxmox є одним із найкращих варіантів для домашніх серверів, адже забезпечує баланс між продуктивністю, гнучкістю та простотою управління, що робить його підходящим як для новачків, так і для досвідчених користувачів [10, 11, 15-16].

Серед інших платформ віртуалізації, які заслуговують на увагу, варто згадати VirtualBox від Oracle – це більше рішення для віртуалізації на основі настільного комп'ютера, а не для налаштування виділеного домашнього сервера (див. рис. 1.23) [1-2, 10].

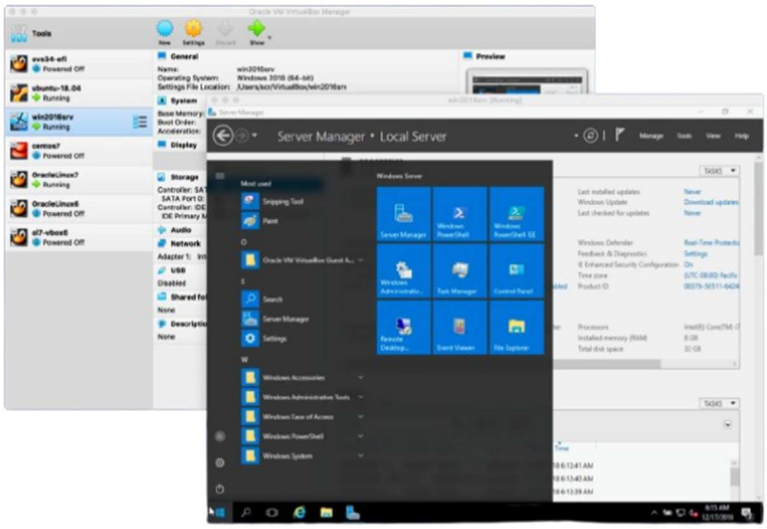


Рисунок 1.23 – Інтерфейс платформи Oracle VirtualBox

Простота у використанні робить його зручним для початківців, які хочуть експериментувати з віртуалізацією. Кросплатформеність дозволяє працювати на Windows, macOS і Linux, а відкритий вихідний код та безкоштовна ліцензія для особистого та освітнього використання роблять його привабливим варіантом для навчання та тестування (див. рис. 1.24) [1, 10].

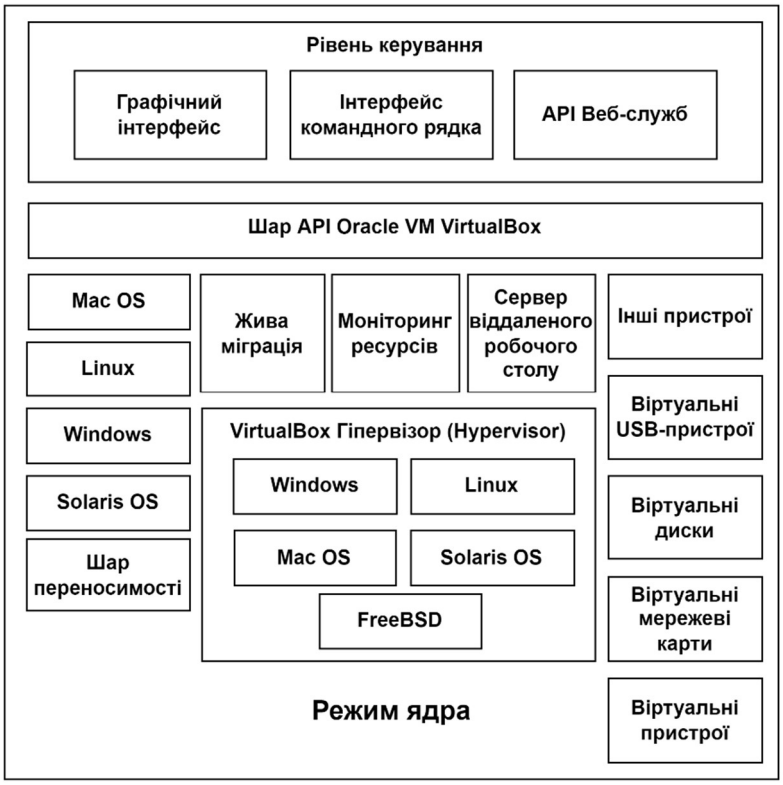


Рисунок 1.24 – Архітектура платформи Oracle VirtualBox

Недоліки включають нижчу продуктивність порівняно з KVM і Proxmox, вищі накладні витрати та залежність від операційної системи хоста. Відсутність розширених функцій, таких як оперативна міграція або кластеризація, обмежує його можливості, а масштабованість є суттєвою проблемою, оскільки VirtualBox найкраще підходить для однокористувацьких середовищ [1-3, 10, 15].

Попри це, залишається корисним інструментом для тестування та навчання, хоча через низьку продуктивність та обмежені функції не є оптимальним вибором для запуску виділеного домашнього сервера. Проте для перших кроків у віртуалізації його можна використовувати [10, 11].

Ще одним популярним рішенням є KVM (Kernel-based Virtual Machine), що працює на базі ядра Linux. Дане рішення перетворює ядро Linux на гіпервізор, що дозволяє безпосередньо створювати та керувати віртуальними машинами (див. рис. 1.25) [10, 2-3].

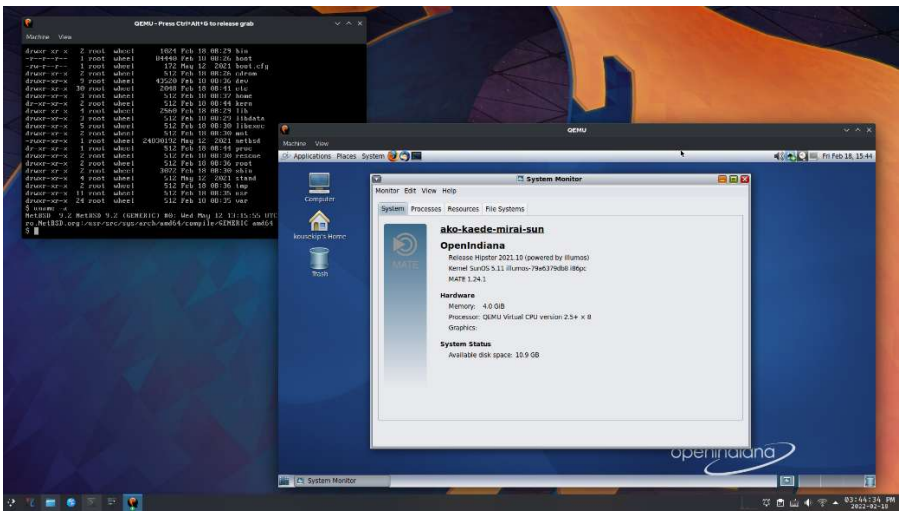


Рисунок 1.25 – Інтерфейс QEMU на базі гіпервізора KVM (Kernel-based Virtual Machine)

Використання разом із QEMU надає розширені можливості управління, а інтеграція з Linux забезпечує високу продуктивність завдяки апаратному прискоренню. Широкі можливості налаштування дозволяють гнучко контролювати параметри віртуальних машин, а продуктивність наближена до роботи на фізичному обладнанні (див. рис. 1.26) [10].

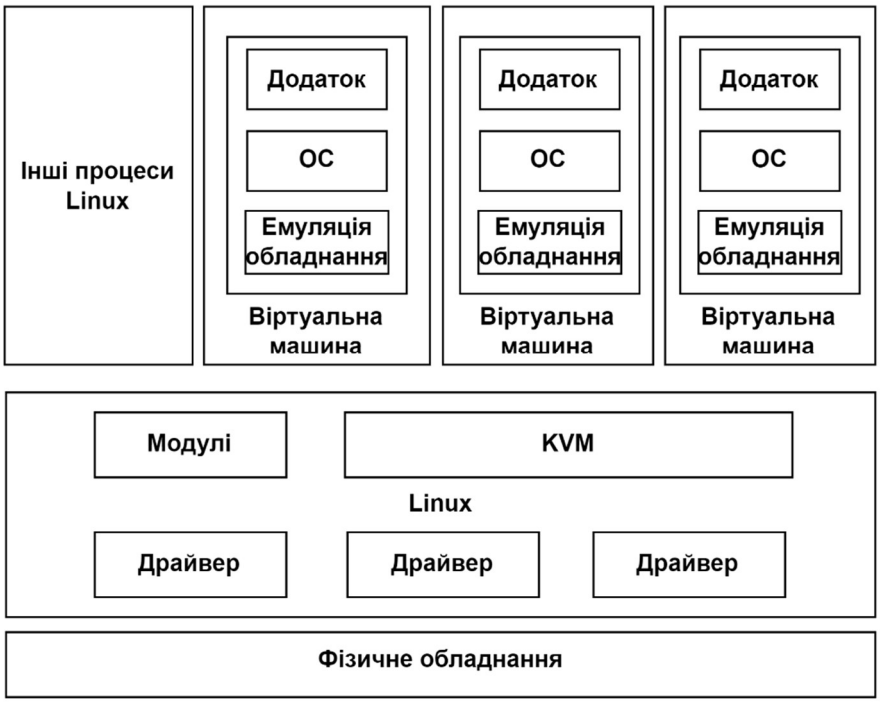


Рисунок 1.26 – Архітектура гіпервізора KVM (Kernel-based Virtual Machine)

Недоліком є складність керування, оскільки для роботи часто використовується командний рядок або додаткові інструменти, такі як Virt-Manager. Високий поріг входу робить його менш зручним для початківців у порівнянні з Proxmox або VirtualBox [10, 11, 15].

Попри це, KVM залишається одним із найефективніших рішень для користувачів Linux, які шукають потужну платформу віртуалізації. Саме тому Proxmox використовує його як основу для створення віртуальних машин, що дозволяє поєднувати його переваги з більш простим інтерфейсом керування [2-3, 10, 11, 15-16].

1.4 Вплив віртуалізації на продуктивність домашніх серверів

Віртуалізація забезпечує значні переваги з точки зору гнучкості, масштабованості та ефективного використання ресурсів, проте також впливає на продуктивність системи. Щоб досягти оптимальної роботи процесора, пам’яті, диска та мережі, необхідно враховувати низку аспектів, серед яких:

— належне налаштування середовища домашнього сервера;

— розгляд ключових показників ефективності.

Ключові показники ефективності. Продуктивність віртуалізованого домашнього сервера залежить від декількох факторів, кожен з яких впливає на швидкість реагування віртуальних машин та контейнерів. Завантаження центрального процесора є одним з головних аспектів, оскільки віртуалізація створює додаткові накладні витрати через потребу гіпервізора керувати декількома екземплярами гостьових операційних систем [6-7, 8, 11].

Факторами, що впливають на продуктивність процесора [8-9, 11]:

— навантаження гіпервізора;

— розподіл віртуальних процесорів;

— закріплення ресурсів;

— підтримка інструкцій віртуалізації.

Легкі гіпервізори, такі як KVM та Proxmox, дозволяють мінімізувати вплив накладних витрат, а правильний розподіл віртуальних процесорів допомагає уникнути затримок у плануванні та зменшити втрати продуктивності. Закріплення віртуальних ядер за фізичними може підвищити ефективність за рахунок зменшення перемикання контексту та втрат кешу. Додаткову роль відіграє підтримка процесорами інструкцій віртуалізації, таких як Intel VT-x та AMD-V, які зменшують накладні витрати на переклад команд (див. рис. 1.27) [1, 8-9, 10-12].

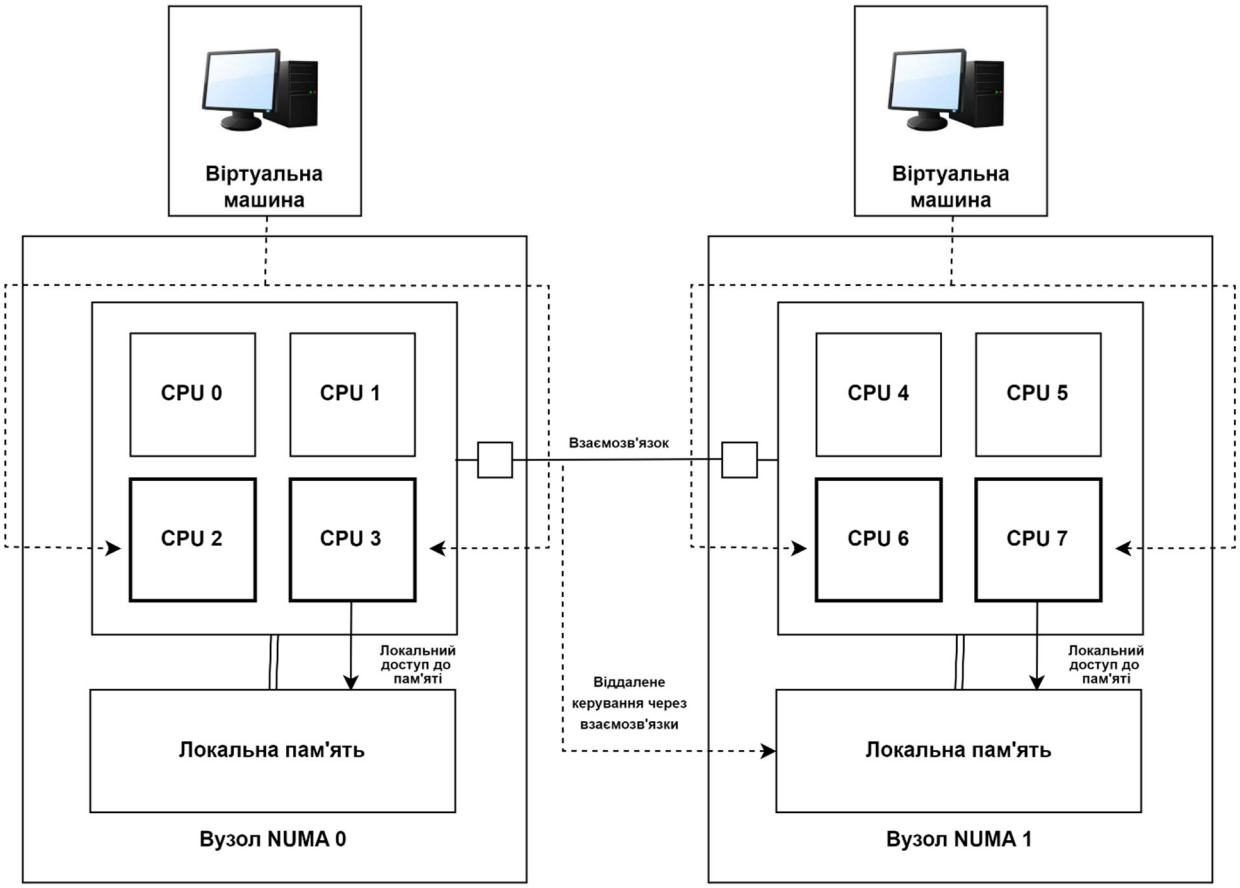


Рисунок 1.27 – Принцип закріплення ядер (CPU Pinning)
за віртуальними машинами

Надмірне перевищення доступного обсягу оперативної пам’яті може призвести до використання swap-файлу (файла підкачки для ОЗП), що значно сповільнить роботу системи. Надмірне використання підкачки диска може

створити додаткове навантаження на систему, тому рекомендується або обмежувати її, або повністю вимикати у критичних середовищах. Для забезпечення стабільної роботи кількох віртуальних машин оптимальним є використання щонайменше 16 ГБ оперативної пам’яті (див. рис. 1.28) [2, 8-9, 10-12].

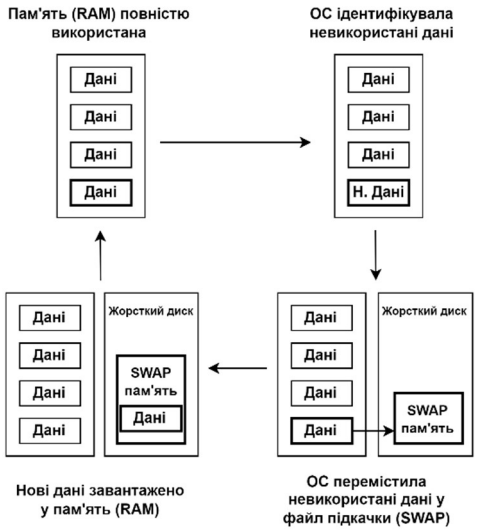


Рисунок 1.28 – Принцип функціонування файлу підкачки (swap-file) у ОС

Швидкість роботи сховища безпосередньо впливає на продуктивність віртуальних машин, оскільки операції введення-виведення визначають час відгуку системи. Використання твердотільних накопичувачів (SSD) замість традиційних жорстких дисків (HDD) значно покращує продуктивність завдяки вищій швидкості читання та запису. Проблеми можуть виникати у випадку, коли декілька віртуальних машин одночасно звертаються до одного сховища, створюючи вузьке місце у підсистемі введення-виведення. Оптимізувати ситуацію можна розподілом навантаження між окремими фізичними дисками та використанням RAID-масивів [8-9, 11-12].

Конфігурація RAID 1 забезпечує резервування даних, тоді як RAID 10 комбінує переваги резервування та високої продуктивності запису. Використання NVMe-накопичувачів є ще одним рішенням для підвищення продуктивності, оскільки вони мають вищий IOPS та нижчі затримки порівняно з традиційними SATA SSD (див. рис. 1.29) [8-9, 11-12].

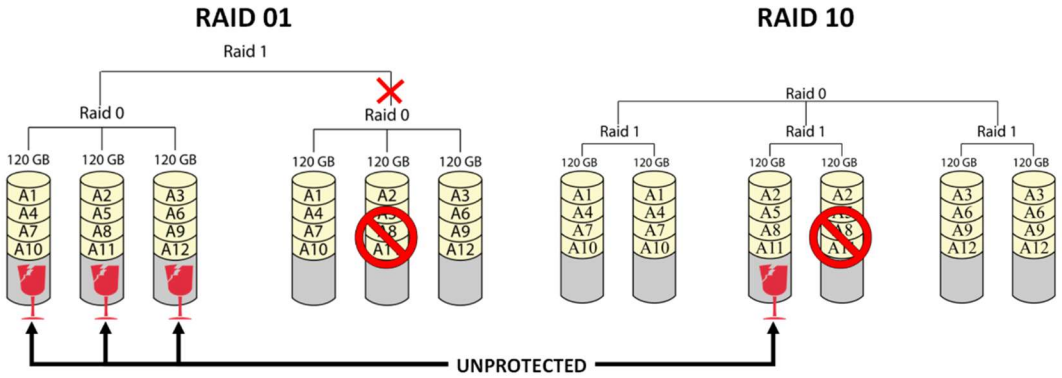


Рисунок 1.29 – Порівняння RAID 01 та RAID 10
як комбінація резервування та високої продуктивності запису [12]

Пропускна здатність мережі також впливає на загальну продуктивність віртуалізованого середовища, оскільки додаткові накладні витрати можуть виникати через віртуалізовані мережеві адаптери. Мережеві режими, такі як Bridge та NAT, відрізняються за продуктивністю: режим Bridge забезпечує кращу інтеграцію з домашньою мережею, тоді як NAT створює додаткові обмеження через необхідність трансляції трафіку (див. рис. 1.30) [2-3, 8-9, 11-12].

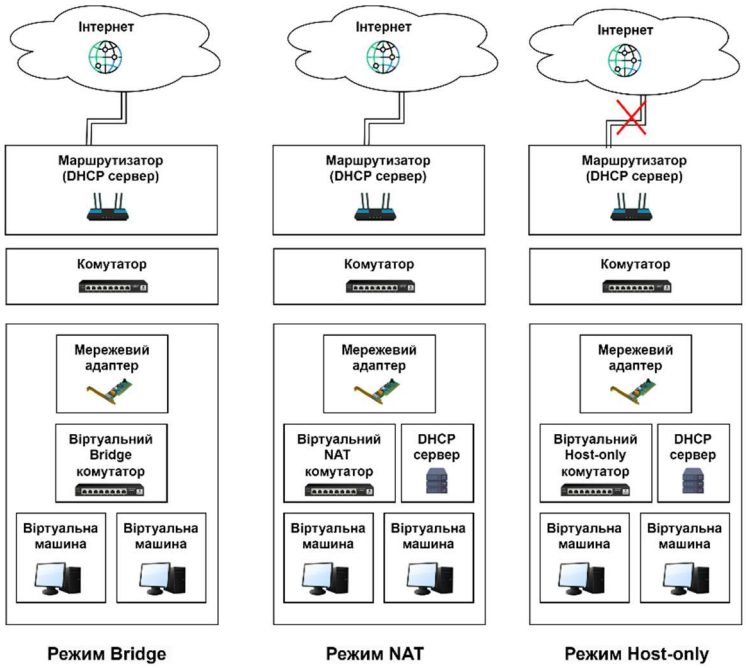


Рисунок 1.30 – Відмінність у роботі мережевих режимів:
режим Bridge забезпечує кращу інтеграцію з домашньою мережею

Використання гігабітних мережевих інтерфейсів є мінімальною вимогою для забезпечення стабільної роботи віртуалізованого середовища, тоді як 2,5 Гбіт/с або 10 Гбіт/с підключення значно покращують продуктивність при передачі великих обсягів даних або роботі з потоковим відео. Всі ці аспекти є критично важливими для досягнення високої продуктивності віртуалізованого домашнього сервера та потребують ретельного налаштування та оптимізації ресурсів (див. рис. 1.31) [1-3, 8-9, 11-12].

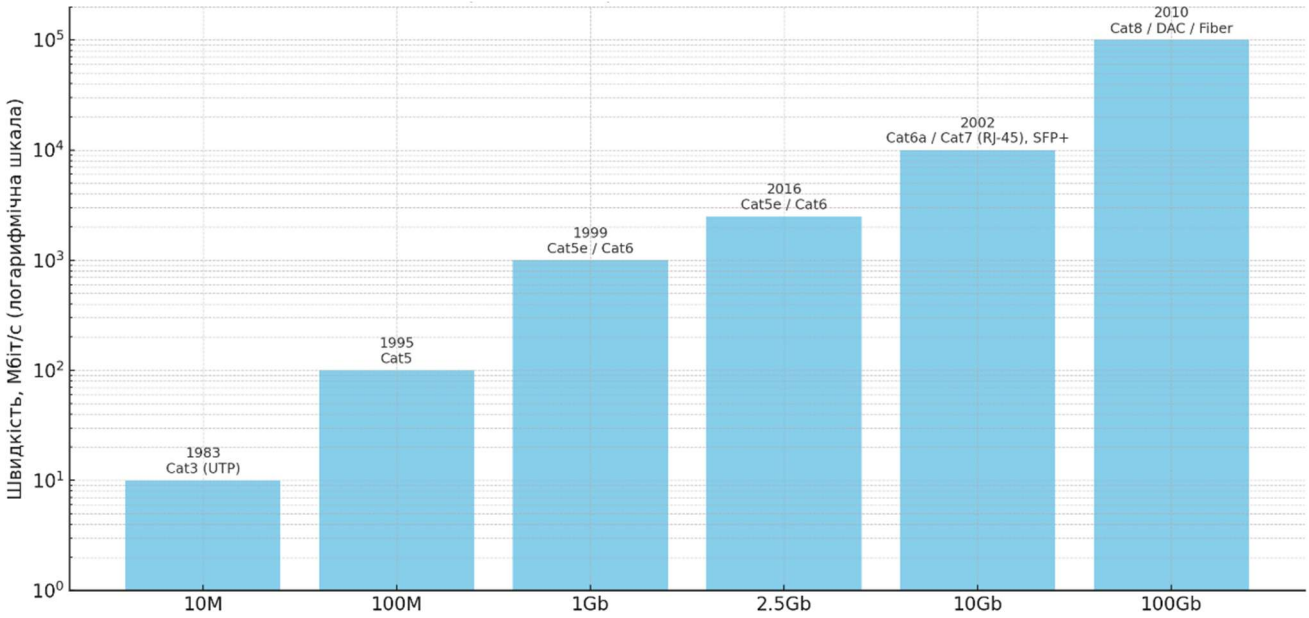


Рисунок 1.31 – Порівняння мережевих технологій Ethernet

1.5 Виклики впровадження віртуалізації на домашніх серверах

Віртуалізація на домашніх серверах забезпечує значну гнучкість, економію коштів і ефективне використання ресурсів, проте вона супроводжується низкою проблем. На відміну від корпоративних середовищ, де апаратне забезпечення й інфраструктура оптимізовані для віртуалізації, домашні сервери покладаються на компоненти споживчого ринку, що може обмежувати продуктивність, надійність і сумісність.

Окрім цього, важливими викликами стають:

— оптимальне використання ресурсів;

— забезпечення безпеки;

— керування сумісністю ПЗ;

— контроль споживання енергії.

Апаратні обмеження у впровадженні віртуалізації на домашньому сервері. Більшість обладнання споживчого класу не розраховане на безперервні навантаження, характерні для віртуалізації, що може спричиняти вузькі місця продуктивності та потенційні проблеми з надійністю. Одним із критичних аспектів є відсутність підтримки пам’яті ECC, яка використовується в корпоративних серверах для виявлення та виправлення помилок, запобігаючи пошкодженню даних. Більшість споживчих материнських плат і процесорів не підтримують ECC, що підвищує ризик помилок у пам’яті, що особливо критично для баз даних і файлових серверів. Використання звичайної пам’яті без ECC може спричинити нестабільну роботу системи під високим навантаженням, а для роботи з ECC необхідні сумісні серверні платформи на базі Intel Xeon або AMD EPYC. Деякі процесори Intel Core і AMD Ryzen підтримують ECC, але це скоріше виняток (див. рис. 1.32) [2-3, 8-9, 10-12].

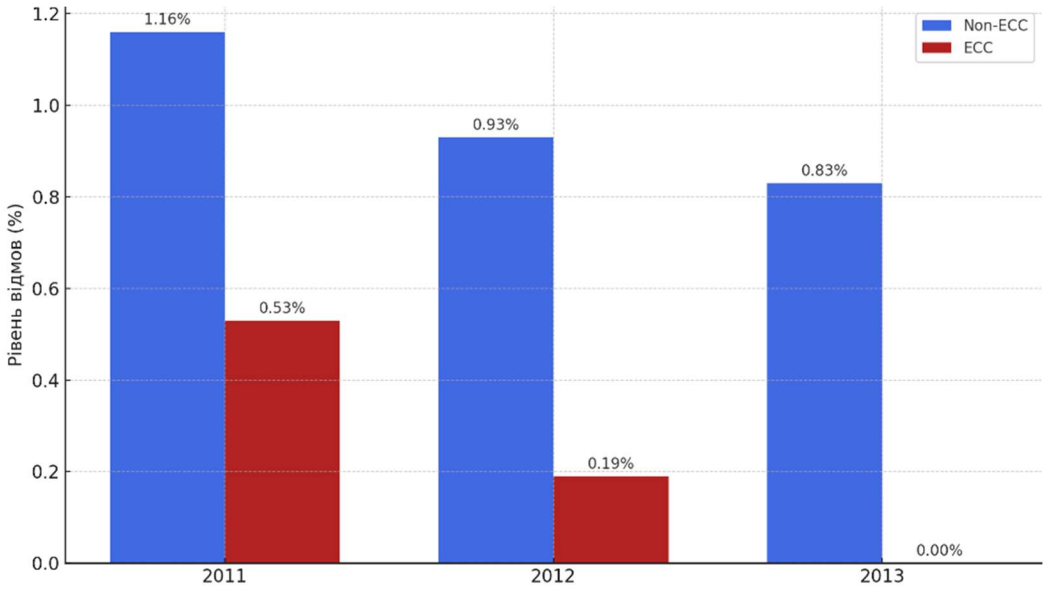


Рисунок 1.32 – Рівень відмов пам’яті Kingston ECC та Non-ECC [27]

Для віртуалізації потрібні високопродуктивні багатоядерні процесори з підтримкою апаратного прискорення, як-от Intel VT-x і AMD-V, що дозволяють ефективніше розподіляти навантаження. Наявність функцій VT-d або AMD-V покращує прямий доступ до апаратного забезпечення, що важливо для проходження GPU та мережевої віртуалізації. Процесори бюджетного рівня, як-от Intel Core i3 чи AMD Ryzen 3, можуть мати труднощі при роботі з кількома віртуальними машинами, тому рекомендується використовувати щонайменше чотири- або шестиядерні процесори з підтримкою SMT для багатопотокової обробки [2-3, 8-9, 11-12].

Жорсткі диски мають низькі IOPS (кількість операцій введення-виведення за секунду), що призводить до повільного завантаження віртуальних машин і уповільнення дисково-інтенсивних завдань. Крім того, споживчі твердотільні накопичувачі мають обмежений ресурс запису, а постійне навантаження з боку віртуальних машин може скоротити їхній термін служби. Відсутність апаратної підтримки RAID-контролерів на споживчих материнських платах ускладнює побудову надійних масивів збереження даних (див. рис. 1.33) [2-3, 8-9, 11-12].

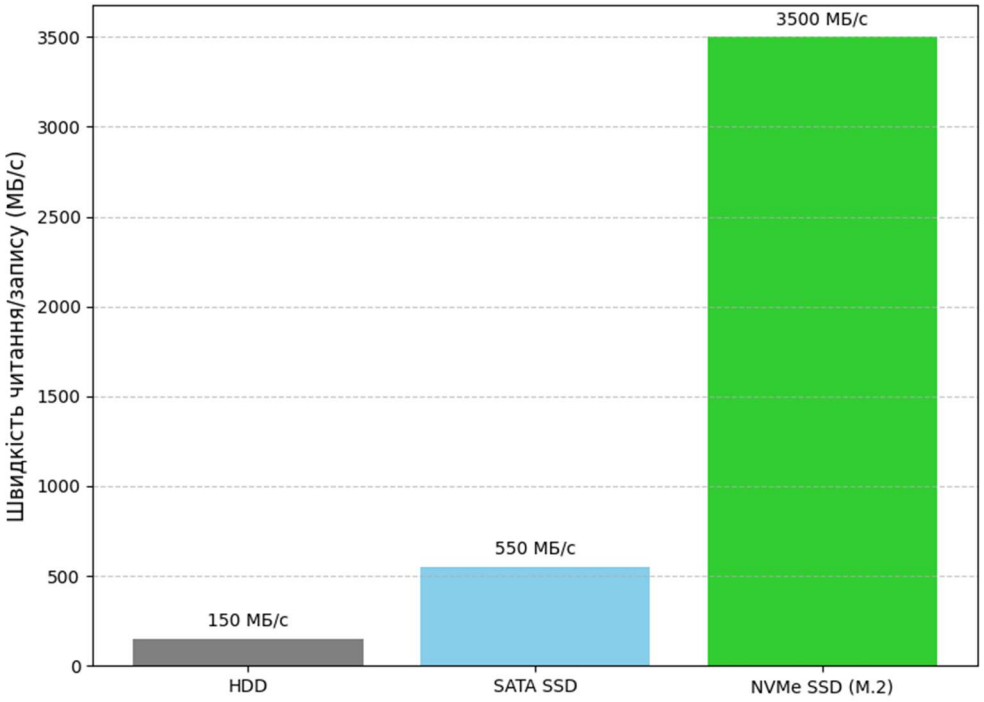


Рисунок 1.33 – Різниця у швидкості накопичувачів різних інтерфейсів: HDD, SATA SSD, NVMe SSD (M.2)

Оптимальним рішенням для дискової підсистеми є використання NVMe SSD для операційних систем і критичних віртуальних машин, тоді як жорсткі диски залишаються найкращим варіантом для масового зберігання, що дозволить збалансувати продуктивність і довговічність, забезпечуючи надійну роботу домашнього сервера у довгостроковій перспективі.

Безпека та продуктивність мережі. Віртуальні мережі забезпечують взаємодію між віртуальними машинами та зовнішніми пристроями, проте їхня неправильна конфігурація може стати вразливим місцем для атак. Доступ до віртуальних мереж через Інтернет, а також неправильно налаштовані правила брандмауера можуть як заблокувати важливий трафік, так і дозволити небажані підключення. Впровадження VLAN вимагає відповідного апаратного забезпечення та точного налаштування для ефективної ізоляції трафіку (див. рис. 1.34) [2-3, 8-9, 11-12].

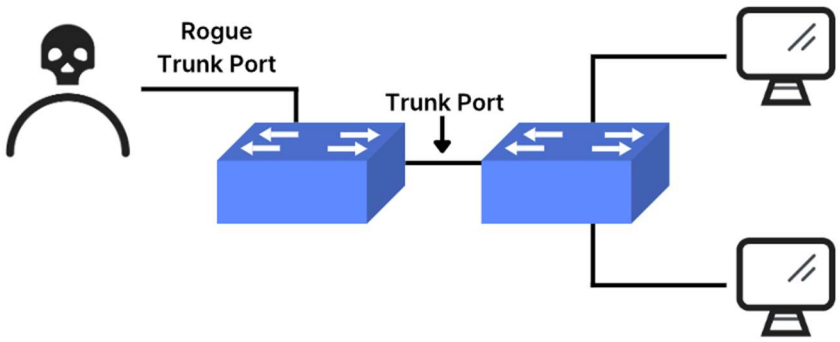


Рисунок 1.34 – Невірна конфігурація VLAN:
вразливість до атак через «перестирубування» VLAN (VLAN Hopping) [12]

Використання мережевої трансляції (NAT) або мостових підключень (Bridge Mode) створює додаткове навантаження, що знижує пропускну здатність. Високе навантаження на обробку пакетів може сповільнити роботу віртуалізованих брандмауерів і маршрутизаторів [2-3, 11-12].

Проблеми сумісності програмного забезпечення. Проблеми сумісності програмного забезпечення можуть виникати через обмежену підтримку віртуалізації, коли певні програми не оптимізовані для роботи у віртуалізованому

середовищі, що стосується, зокрема, медіапрограм із захистом DRM, які через апаратні обмеження не працюють у віртуальних машинах. Програми, що використовують прискорення GPU та потребують прямого доступу до PCI-пристроїв, також можуть зіткнутися з труднощами в налаштуванні. Аналогічно, передача USB-пристроїв, зокрема апаратних ключів або спеціалізованих периферійних пристроїв, може бути нестабільною, що ускладнює їх використання [2-3, 11-12].

Хоча споживання енергії в домашніх умовах менше, ніж у корпоративних середовищах, воно все ж суттєве, особливо при використанні процесорів із великою кількістю ядер, таких як Intel Xeon або AMD AS. Високопродуктивні графічні процесори з підтримкою PCI passthrough вимагають додаткової потужності, а жорсткі диски з обертовими пластинами споживають більше енергії, ніж SSD, та додатково нагрівають систему. Чим більше таких накопичувачів у сервері, тим вище загальне енергоспоживання (див. рис. 1.35) [8-9, 11-12].

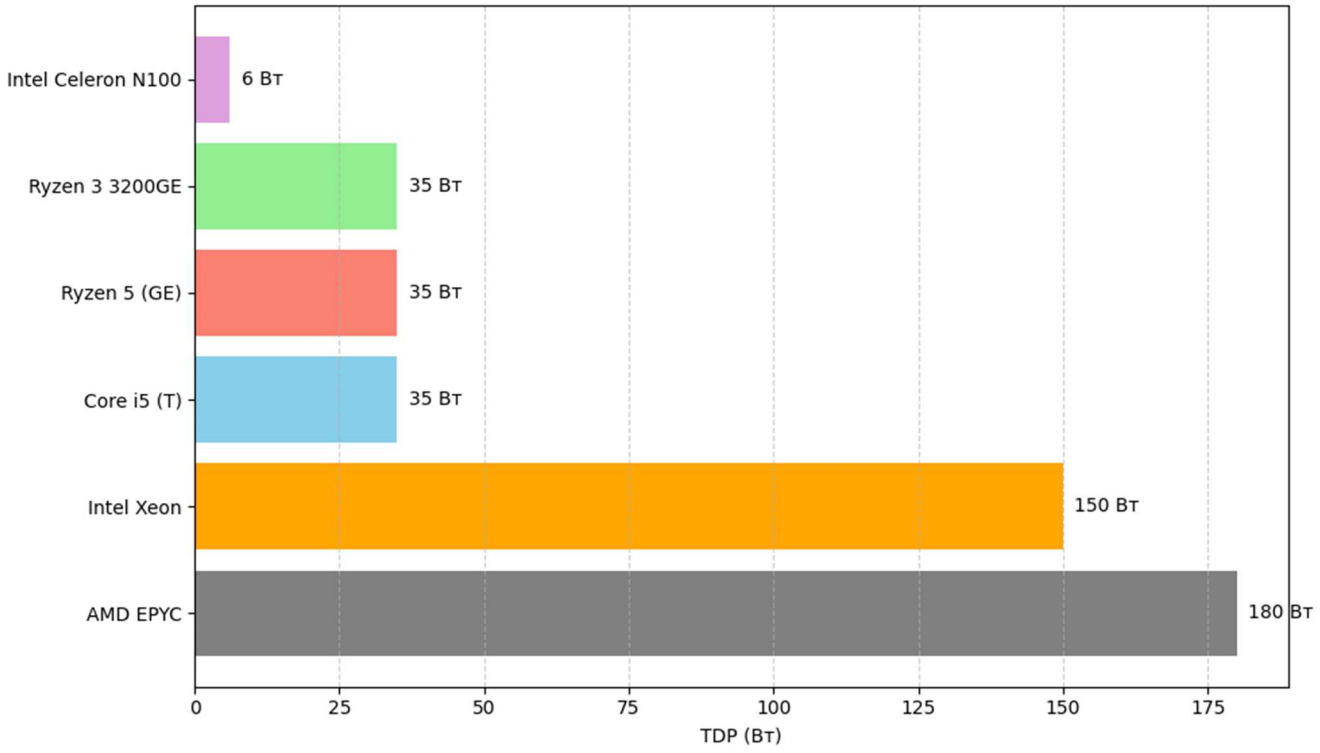


Рисунок 1.35 – Корпоративні та споживчі лінійки процесорів:
вплив потужності на енергоспоживання у TDP (Вт)

Зменшити витрати на електроенергію можна за рахунок використання енергоефективних процесорів, таких як Intel Xeon із низьким енергоспоживанням або AMD Ryzen 5-7, а також налаштування параметрів енергозбереження в BIOS.

Корпуси малого форм-фактора часто мають поганий повітряний потік, що спричиняє перегрів компонентів у разі неправильної конструкції. Високопродуктивні системи можуть потребувати активного охолодження, що підвищує рівень шуму та збільшує споживання енергії. Додатковою проблемою стає накопичення пилу в постійно увімкнених системах, яке знижує ефективність охолодження (див. рис. 1.36).



Рисунок 1.36 – Вплив розміру на повітряний потік:
E-ATX, ATX, micro ATX та ITX

Для оптимального тепловідведення можна використовувати великі радіатори, що знижують необхідність активного охолодження, а також регулярно проводити технічне обслуговування, очищуючи систему від пилу, що дозволяє підтримувати стабільну температуру компонентів і зберігати продуктивність сервера на належному рівні.

РОЗДІЛ 2 ПОБУДОВА ТЕСТОВОГО СЕРЕДОВИЩА ТА МЕТОДИКА ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ

2.1 Апаратна та програмна конфігурація домашнього сервера

Конфігурація домашнього сервера у віртуалізованому середовищі потребує структурного та практичного підходу, що включає:

— вибір відповідного апаратного забезпечення;

— визначення платформи віртуалізації;

— визначення стека програмного забезпечення.

Вибрані компоненти безпосередньо впливають на продуктивність:

— використання центрального процесора;

— ефективність оперативної пам’яті;

— пропускну здатність дискової системи;

— пропускну здатність мережі.

Вибір обладнання у якості сервера. Для демонстрації конфігурації домашнього сервера було обрано Mini PC, що тимчасово використовувався для тестування віртуалізації. Вибір такого пристрою пояснюється його компактністю, енергоефективністю та достатньою потужністю для виконання поставлених завдань. Хоча для домашнього сервера немає потреби у високопродуктивному обладнанні, було вирішено протестувати GMKtec K8 Plus, придбаний на AliExpress у найновішій модифікації (див. рис. 2.1).



Рисунок 2.1 – Зовнішній вигляд Mini PC GMKtec K8 Plus

У сервері встановлено процесор AMD Ryzen 7 845HS із вісьмома ядрами та шістнадцятьма потоками, що забезпечує достатню продуктивність для багатозадачності та керування віртуальними машинами. Процесор має інтегровану графіку Radeon 780M, яка за продуктивністю наближається до дискретної GTX 1650, проте споживає значно менше енергії—лише 15 Вт, тоді як сам процесор у пікових навантаженнях не перевищує 70 Вт, що робить систему енергоефективною, що є важливою перевагою для домашнього сервера.

Пристрій компактний, що дозволяє легко розмістити його поряд із маршрутизатором, забезпечуючи мінімальні затримки в мережевому підключенні. Для стабільної роботи важливим є достатній обсяг оперативної пам’яті. У даній конфігурації використовується 32 ГБ DDR5 від Crucial із частотою 5600 МГц, що забезпечує швидку взаємодію із системою (див. рис. 2.2). Хоча для деяких сценаріїв могло б вистачити 16 ГБ, більший обсяг пам’яті гарантує стабільну роботу віртуальних машин та дає змогу запускати більше процесів одночасно. Використання пам’яті з підтримкою ECC було б перевагою, однак у даній конфігурації це неможливо через мобільну архітектуру пристрою.



Рисунок 2.2 – Оперативна пам’ять Crucial SODIMM DDR5 32GB 5600Mhz

Для зберігання даних було обрано SSD NVMe Lexar на 2 ТБ (див. рис. 2.3), що забезпечує швидкість передачі до 5 ГБ/с через інтерфейс PCIe 4.0, що значно покращує продуктивність порівняно з жорсткими дисками, особливо для роботи з віртуальними машинами, де важливі швидкість доступу до даних та низькі затримки. Використання HDD могло б бути виправданим для масового зберігання файлів, проте в даній конфігурації додати внутрішній жорсткий диск неможливо, тому такий варіант поки не розглядається.



Рисунок 2.3 – Накопичувач SSD NVMe M.2 Lexar NM710 2 TB

SSD забезпечує більшу надійність, кращий показник IOPS та швидший час завантаження, що важливо для серверних задач, таких як веб-хостинг чи бази даних. Хоча деякі сервери можуть використовувати SSD меншого обсягу, наприклад 256 ГБ, поточна конфігурація гарантує запас пам’яті для майбутніх розширень. Водночас додавання HDD для масового зберігання може вимагати

використання зовнішніх рішень, що не завжди є зручним варіантом для компактного Mini PC.

Використання цього Mini PC як великого хмарного сховища розглядалося як можливий варіант, однак через певні обмеження було прийнято рішення відмовитися від такої ідеї. Основна проблема полягає в тому, що єдиним способом підключення додаткових накопичувачів є використання зовнішніх корпусів за типом DAS, що по суті є простою коробкою з можливістю підключення через USB-інтерфейс, що значно обмежує швидкість передачі даних, що робить такий підхід недоцільним для повноцінного NAS-сервера (див. рис. 2.4). При використанні більш гнучкої платформи з можливістю встановлення накопичувачів безпосередньо всередині корпусу можна було б отримати більш ефективне рішення.



Рисунок 2.4 – Недоцільність використання DAS сховищ у зв’язку з Mini PC із-за малої пропускної спроможності інтерфейса USB

Цей Mini PC краще підходить для продуктивних завдань, таких як рендеринг відео, обробка фотографій або навіть певні ігрові сценарії, однак основною метою було перевірити його можливості у якості сервера. Важливим фактором стало тестування продуктивності у реальних умовах, а також оцінка можливих обмежень. Після розгляду всіх аспектів конфігурації обладнання логічним було узагальнити критерії вибору компонентів та визначити їхню відповідність поставленим задачам.

У разі можливості встановлення кількох жорстких дисків, наприклад, двох або чотирьох, можна було б використовувати RAID-масиви для підвищення надійності зберігання даних. Програмний RAID з SSD був би надто дорогим рішенням, а оптимальним варіантом зазвичай є апаратний RAID-контролер, який у даному випадку відсутній. Можливий варіант RAID 1 або ZFS для дзеркального зберігання даних міг би забезпечити додатковий рівень захисту, особливо з огляду на використання Proxmox, який має розширені функції управління сховищами, включаючи знімки, стиснення та інші можливості. Проте через відсутність можливості встановлення додаткових накопичувачів у корпус Mini PC цей варіант також довелося відкинути.

Щодо підключення до мережі, дана конфігурація навіть перевершує базові очікування, оскільки GMKtec K8 Plus оснащений двома портами RJ-45 із пропускною здатністю 2,5 Гбіт/с кожен, що дає можливість призначити їх для різних задач у віртуалізованому середовищі, наприклад, виділивши один порт для зовнішнього трафіку, а інший для внутрішніх з'єднань. Таке рішення дозволяє створити кастомний маршрутизатор із гнучким управлінням мережею (див. рис. 2.5).



Рисунок 2.5 – Наявність у Mini PC двох портів RJ-45 на 2.5 Гбіт/с: створення кастомізованого маршрутизатора зі захистом

Використання цього пристрою у якості маршрутизатора дає змогу під'єднати його до невеликого комутатора та отримати потужну комбінацію для розширення мережевих можливостей. Додавання точки доступу дозволило б отримати функціональність, що значно перевершує стандартні маршрутизатори за

продуктивністю. Базового гігабітного Ethernet-порту могло б вистачити для багатьох завдань, однак наявність двох портів 2,5 Гбіт/с дає запас пропускної здатності для майбутніх розширень. Ідеальним рішенням був би 10-гігабітний порт, однак у домашніх мережах його використання наразі не є масовим, тому 2,5 Гбіт/с виглядає більш реалістичним варіантом.

У подальшому дослідженні буде показано, наскільки обрана конфігурація Mini PC відповідає заявленим вимогам та чи варто віддавати перевагу більш традиційним серверним рішенням замість одного потк Mini PC. Можливий варіант використання менш продуктивного Mini PC, якщо завдання не потребують значних ресурсів, але це буде проаналізовано окремо.

Вибір платформи віртуалізації – Proxmox VE. Серед різноманітних доступних рішень перевага надається Proxmox VE завдяки його відкритому коду, гнучкості та функціям корпоративного рівня, які можна використовувати без необхідності платної ліцензії. Він забезпечує повну віртуалізацію на основі KVM, а також підтримує Linux-контейнери LXC, що надає додаткові можливості для гнучкого керування ресурсами [2-3, 10-11, 15].

Веб-інтерфейс Proxmox VE зручний для моніторингу та управління, і після певного часу адаптації він стає інтуїтивно зрозумілим навіть для тих, хто раніше використовував VirtualBox або VMware Workstation.

Окрім цього, однією з основних причин вибору цієї платформи стала можливість кластеризації, що особливо корисно в умовах використання Mini PC. Замість одного потужного сервера можна придбати кілька менш продуктивних пристроїв, таких як Intel NUC, і об’єднати їх у кластер (див. рис. 2.6) [10-11, 13, 15].

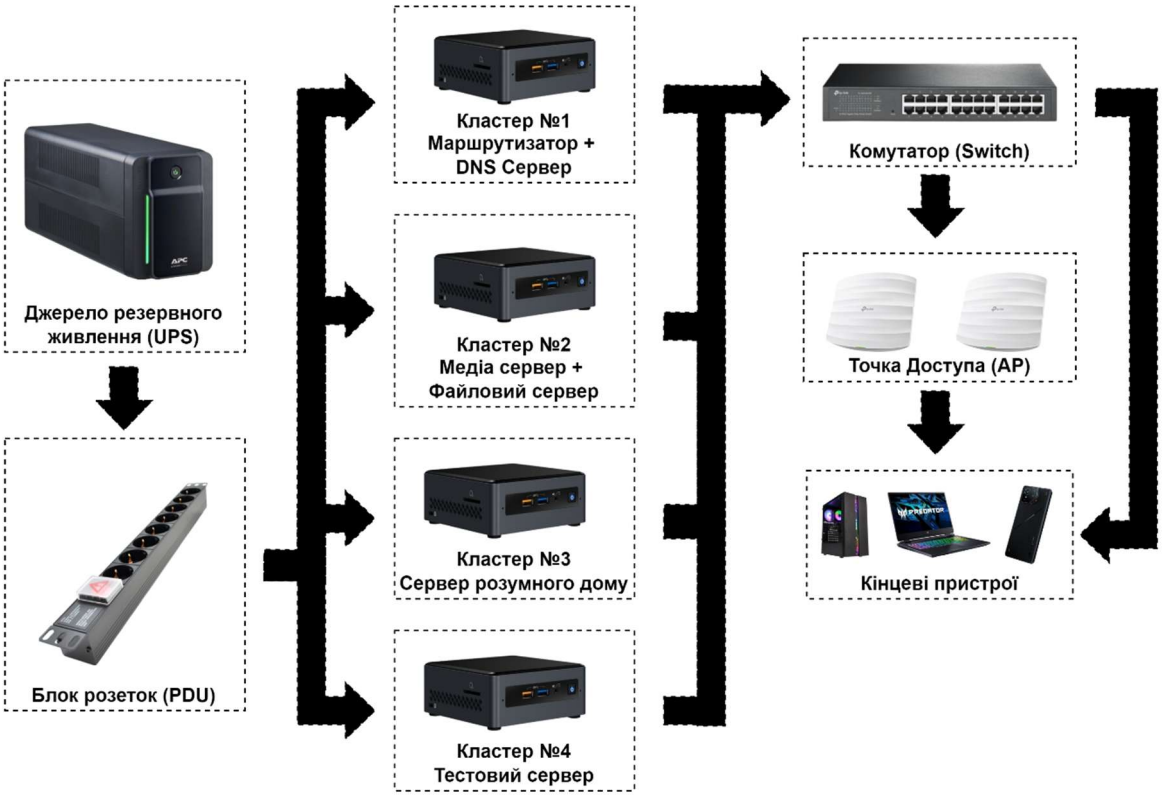


Рисунок 2.6 – Архітектура кластерної відмовостійкої системи

Така архітектура дозволяє забезпечити вищу відмовостійкість: у разі виходу з ладу одного з вузлів інші продовжуватимуть роботу, що принципово важливо для серверних систем і схоже на те, як в енергетиці окремі блоки атомних електростанцій можуть відключатися для ремонту, тоді як інші продовжують працювати на повну потужність. Аналогічно кластери серверів дозволяють безперервно підтримувати роботу системи навіть під час технічного обслуговування або несподіваних збоїв [10-11, 13, 15].

Ще однією перевагою Proxmox VE є функція резервних знімків, що дає змогу зберігати стан віртуальної машини у будь-який момент і повертатися до нього при необхідності. Крім того, система інтегрується з ZFS, що забезпечує розширені можливості керування сховищем даних. Хоча в даному випадку ця функція не є критичною, її наявність додає гнучкості (див. рис. 2.7) [10-11, 13, 15].

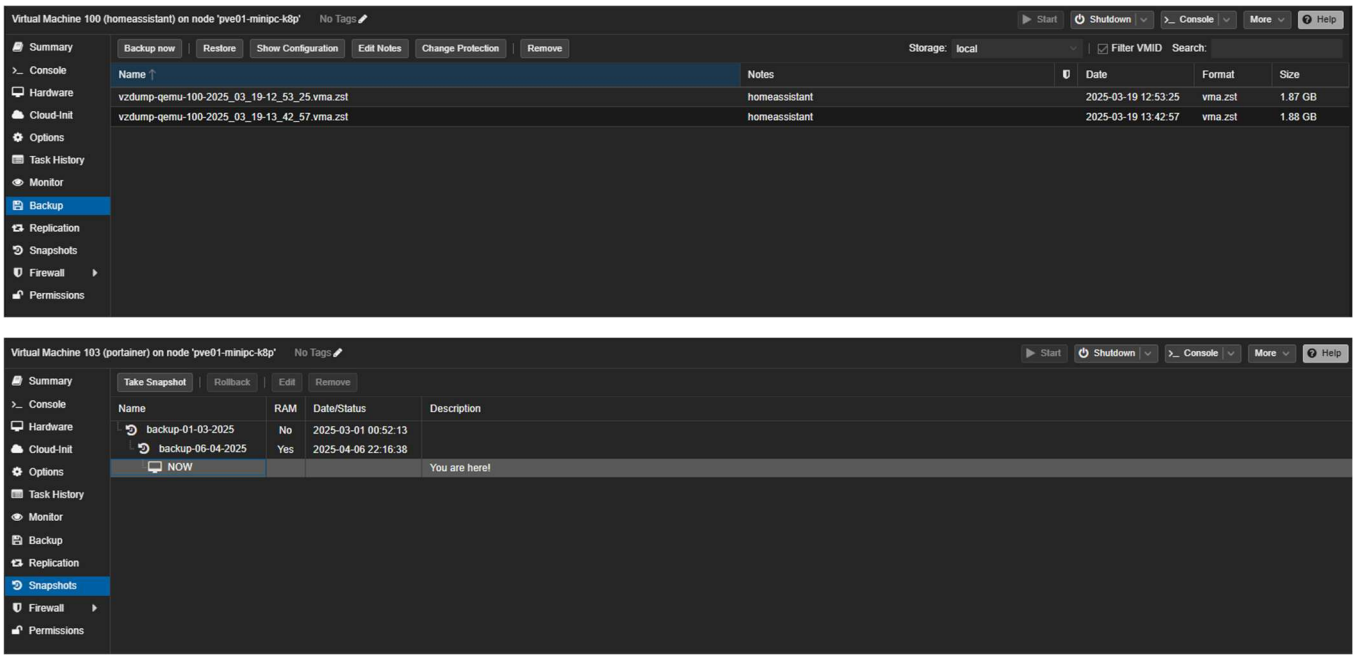


Рисунок 2.7 – Варіанти резервного копіювання у Proxmox VE: "Backup" та "Snapshots"

На відміну від комерційних рішень, таких як VMware ESXi, що потребують ліцензії та не підходять для бюджетних систем, Proxmox VE є безкоштовним, стабільним та повноцінним рішенням для домашніх серверів. VirtualBox і VMware Workstation хоч і прості у використанні, проте не оптимізовані для великої кількості віртуальних машин, оскільки працюють через гіпервізор другого типу [1-3, 10-11, 15].

Що стосується KVM, то він є ефективним завдяки нативній інтеграції з Linux, проте потребує значно більшої кількості ручних налаштувань. Додатково можна було б використовувати Docker або LXC-контейнери, які відзначаються швидким розгортанням, проте вони не забезпечують повної віртуалізації операційної системи [3, 10-11, 13, 15].

Proxmox VE поєднує в собі найкраще з обох світів: повну віртуалізацію та контейнеризацію, що дозволяє оптимально розподіляти ресурси. На відміну від встановлення стандартного серверного дистрибутива, де ресурси можуть витрачатися неефективно, Proxmox VE дає змогу створювати віртуальні машини з повною віртуалізацією або використовувати легкі Linux-контейнери, якщо не потрібна окрема операційна система, що забезпечує ефективне використання

обчислювальних ресурсів, високу гнучкість і можливість масштабування залежно від потреб (див. рис. 2.8).

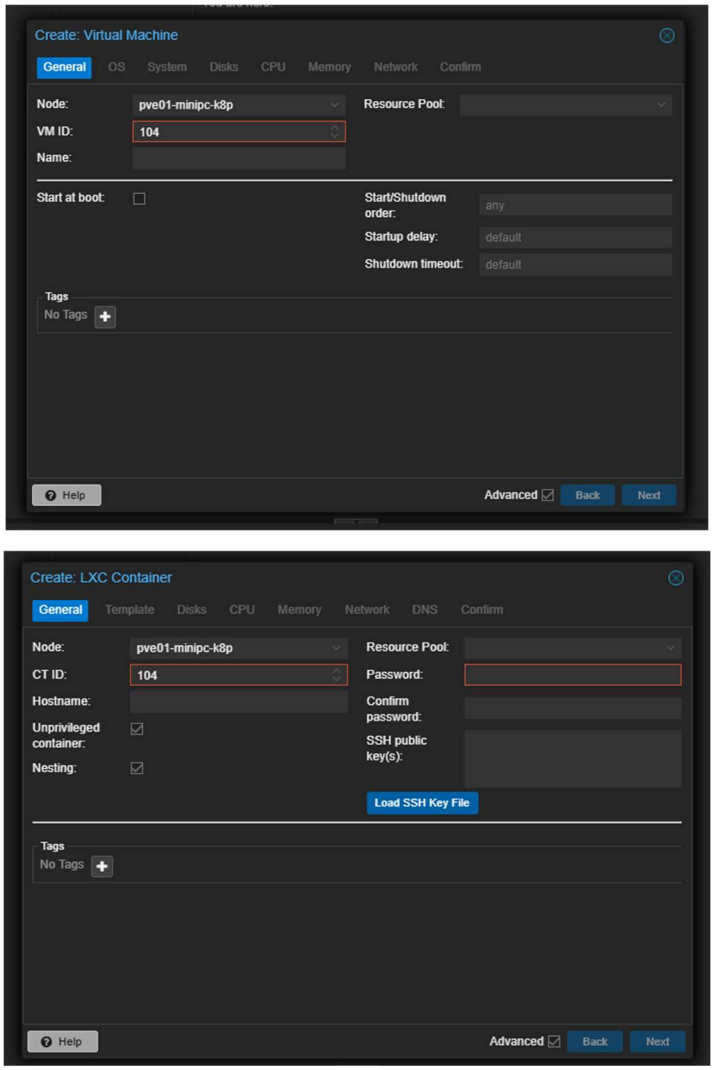


Рисунок 2.8 – Можливість створювати повноцінну віртуальну машину (VM) та Linux-контейнер (LXC)

Вбудований веб-інтерфейс спрощує моніторинг і управління (див. рис. 2.9), а підтримка ZFS додає додаткових переваг, що забезпечує найкращий баланс між функціональністю, стабільністю та доступністю, що робить його оптимальним вибором для домашнього використання [3, 10-11, 13, 15].



Рисунок 2.9 – Вбудований системний моніторинг над ресурсами: динамічні графіки навантаження CPU, RAM, мережі та сервера

Стек програмного забезпечення та засоби керування. Серед програмного забезпечення, необхідного для моніторингу та оптимізації продуктивності віртуального домашнього сервера, важливу роль відіграє відповідний стек. Його склад включає операційну систему хоста, проте у цьому випадку використовується Proxmox VE, який надає широкий набір вбудованих інструментів для аналізу роботи системи [1-3, 10-11, 15].

Proxmox VE побудований на основі Debian, що забезпечує стабільність, підтримку великої кількості програмних інструментів та глибоку інтеграцію з екосистемою Linux. Він містить розширені можливості для моніторингу, які дозволяють відстежувати навантаження процесора, використання пам’яті, мережевий трафік та загальний стан системи. Можливий аналіз як загальних показників сервера, так і деталізованих даних щодо окремих віртуальних машин або контейнерів [13, 15].

Використання Proxmox VE як платформи для віртуалізації дозволяє отримати розширені можливості керування сервером, включаючи доступ до журналів подій, контроль оновлень та діагностику можливих збоїв. Система дозволяє легко

аналізувати причини відмов у роботі, переглядати логи та швидко реагувати на можливі проблеми (див. рис. 2.10) [10, 15].

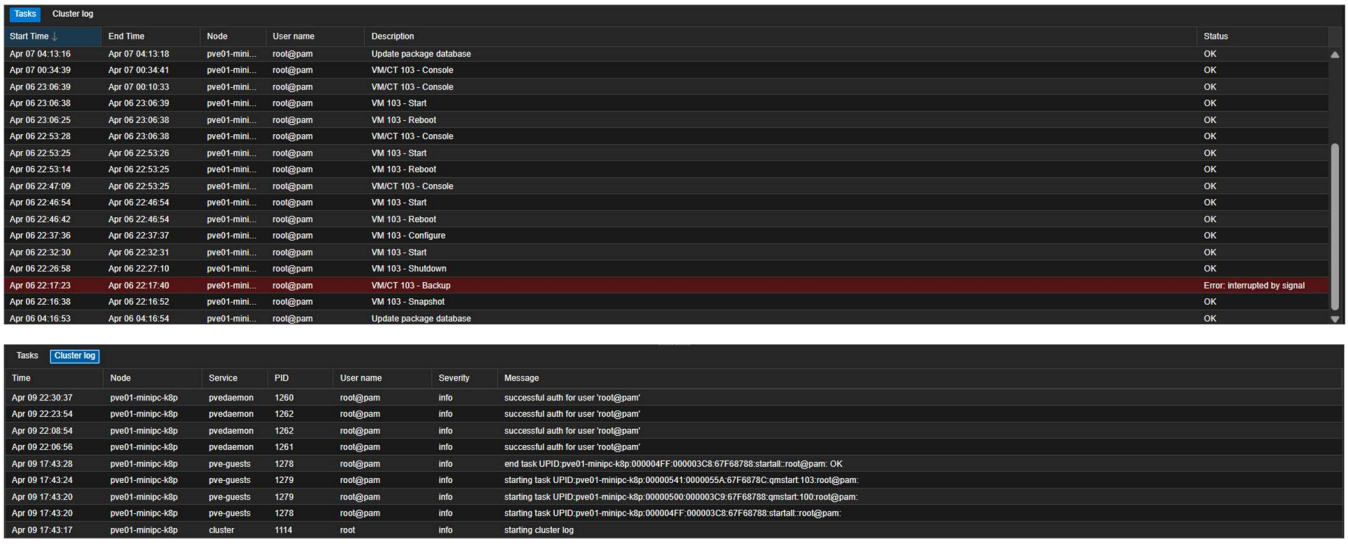


Рисунок 2.10 – Перегляд системного журналу:
"Tasks" та "Cluster log"

Загальна архітектура включає Proxmox VE як основу для віртуалізації, Home Assistant як автоматизація домашніх пристроїв, Ubuntu Server для роботи з контейнерами та Docker для керування програмними компонентами. Серед встановлених сервісів є системи для блокування реклами в мережі, медіасервери, такі як Jellyfin, а також інструменти для потокової обробки контенту (див. рис. 2.11) [2-3, 10-11, 15, 17, 20, 21].

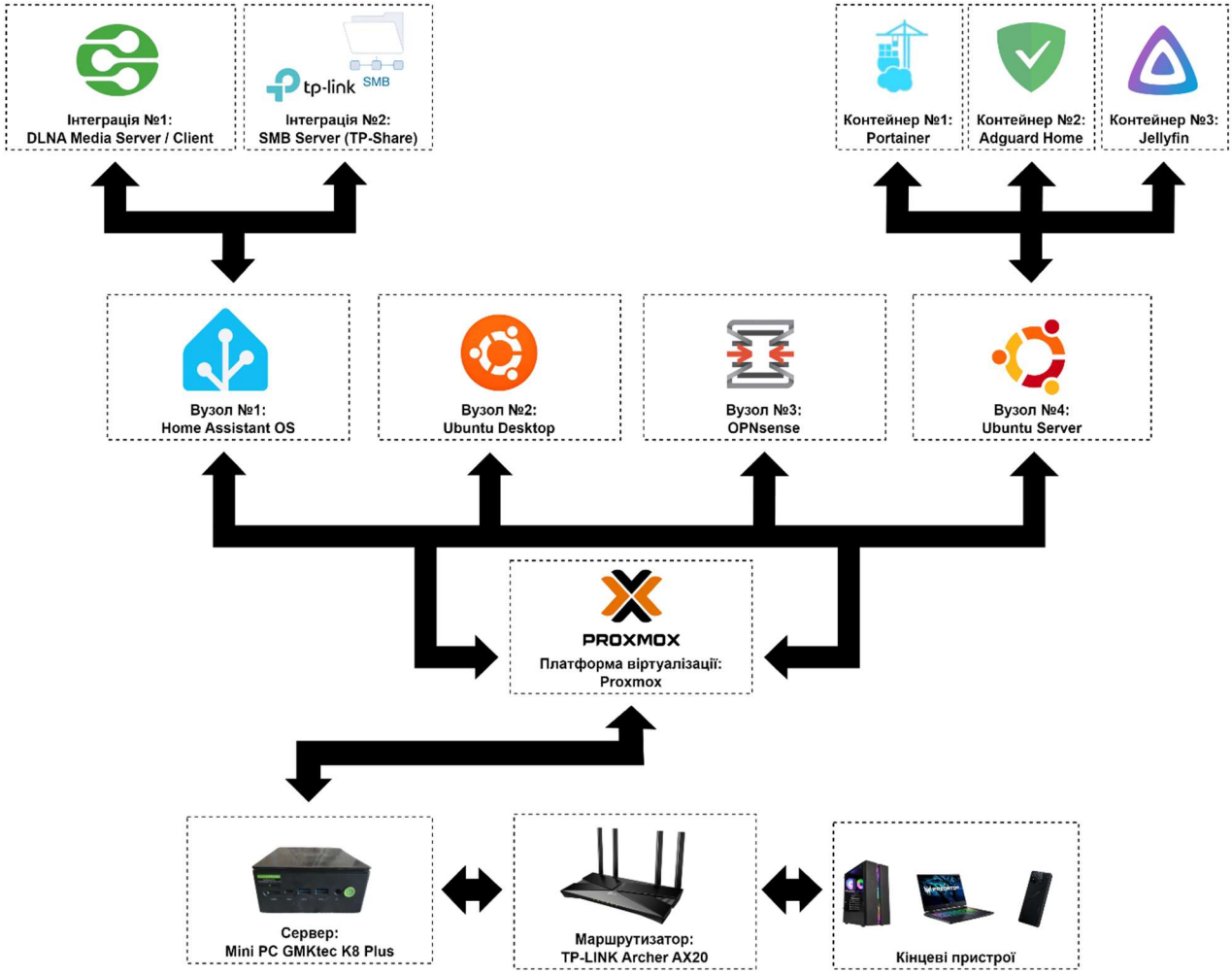


Рисунок 2.11 – Архітектура домашнього сервера на основі віртуалізації

Така конфігурація забезпечує гнучке та стабільне середовище для домашнього сервера, поєднуючи високу функціональність, зручність керування та мінімальні вимоги до технічного обслуговування.

2.2 Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі

Конфігурація віртуальних машин для кожного робочого навантаження потребує детального підходу, оскільки розподіл ресурсів безпосередньо впливає на продуктивність. У процесі тестування було визначено оптимальні параметри для

кожного сценарію використання, що дозволило забезпечити стабільну роботу всіх сервісів.

Конфігурації віртуальної машини для кожного робочого навантаження. Одним із перших протестованих сценаріїв став Home Assistant. Для його роботи було виділено 4 віртуальні ядра, хоча потенційно можна було обмежитися двома. Процесор із вісьмома фізичними ядрами та шістнадцятьма потоками дозволяв гнучко керувати ресурсами. Для оперативної пам'яті було встановлено 6 ГБ, що є достатнім для стабільної роботи. Загальний обсяг доступної постійної пам'яті становив 32 ГБ, тому її розподіл здійснювався з урахуванням потреб інших сервісів. Для зберігання даних Home Assistant отримав 32 ГБ дискового простору, а мережеве з'єднання було забезпечене через гігабітний канал (див. рис. 2.12).

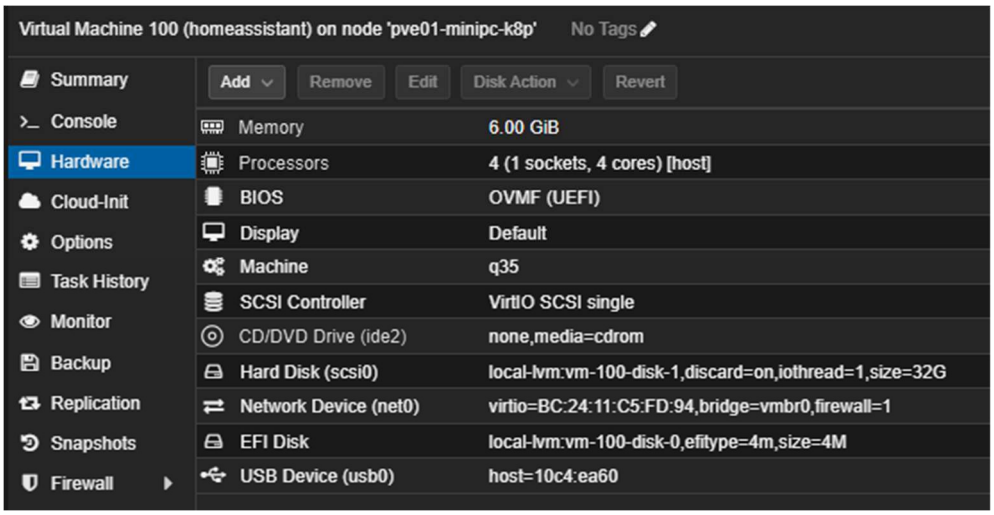


Рисунок 2.12 – Налаштована конфігурація вузла:
сервер розумного дому

Наступний сценарій включав роботу контейнеризованих сервісів, таких як DNS-сервер AdGuard Home та медіасервер Jellyfin, що працювали у межах однієї віртуальної машини на базі Ubuntu Server. Для цієї конфігурації було виділено 16 ГБ оперативної пам'яті через значне навантаження з боку Jellyfin. Віртуальний процесор складався з 6 ядер, що забезпечувало достатню продуктивність для всіх контейнерів. Спочатку для сховища було виділено 50 ГБ, але згодом додано ще 256 ГБ для медіаконтенту та 100 ГБ під Docker контейнери. Ідеальним варіантом стало

б використання SSD-накопичувача об’ємом 500 ГБ як кешу разом із кількома терабайтами жорсткого диска, проте наявна конфігурація вимагала балансування ресурсів. Мережеве підключення реалізовано через гігабітний інтерфейс, що забезпечувало достатню швидкість для передачі даних між контейнерами (див. рис. 2.13).

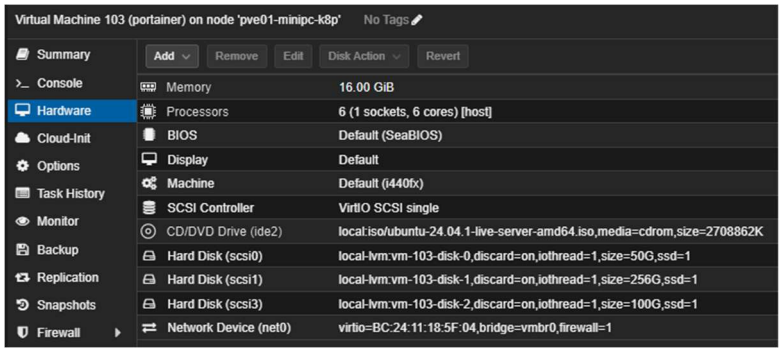


Рисунок 2.13 – Налаштована конфігурація вузла:
DNS- та медіа- сервери

Окремим сценарієм тестування було створення власного маршрутизатора на базі OPNsense. Для цієї задачі віртуальній машині виділено 2 ГБ оперативної пам’яті, 2 віртуальні ядра та 32 ГБ SSD-дискового простору. Цей обсяг пам’яті був достатнім, оскільки маршрутизатор не вимагав значного сховища, окрім випадків резервного копіювання. Мережеве підключення використовувало гігабітний канал із можливістю маршрутизації трафіку між зовнішнім та внутрішнім сегментами мережі (див. рис. 2.14).

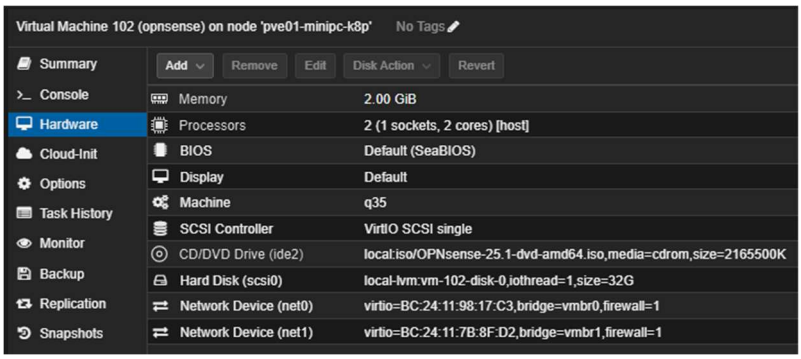


Рисунок 2.14 – Налаштована конфігурація вузла:
кастомізований маршрутизатор зі фаєрволом

Останній сценарій включав запуск тимчасових віртуальних машин для тестування різних дистрибутивів Linux, а саме Debian. Для таких машин виділялися всі доступні ресурси, що залишалися в системі, з урахуванням необхідного резерву для основних сервісів, що дозволяє проводити експерименти без ризику для стабільності інших віртуальних середовищ (див. рис. 2.15).

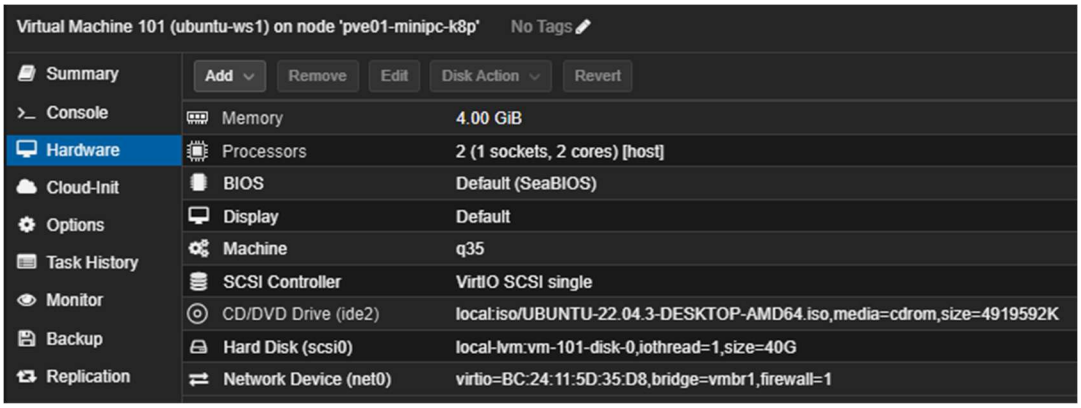


Рисунок 2.15 – Налаштована конфігурація вузла:
віртуальна машина для тестування

У результаті конфігурація віртуальних машин була оптимізована таким чином, щоб забезпечити стабільну роботу всіх сервісів, балансуючи між продуктивністю та доступними апаратними ресурсами.

Варіанти використання віртуального домашнього сервера. Домашній сервер часто виконує декілька завдань одночасно, а не лише підтримку одного єдиного сервісу. Під час дослідження активно працювали три платформи, які виконували свої сценарії, хоча загалом було передбачено чотири можливих варіантів використання. Додаткові сценарії навантаження застосовувалися переважно для тестування.

Три сценарії мали практичне застосування, тоді як четвертий передбачав тестування операційних систем, що не працювали у режимі 24/7, а періодично вмикалися та вимикалися. Перелічені сценарії не є вичерпним списком, адже можливе розширення функціоналу за рахунок додаткових навантажень. Навіть за поточного рівня використання сервер працює з мінімальним навантаженням на ресурси та споживає відносно небагато енергії.

Перший сценарій використання передбачав розгортання Home Assistant для перевірки роботи сервера в умовах автоматизації будинку. Було протестовано два основних моменти: базове використання Home Assistant у простої та робота в режимі медіасервера через вбудований DLNA (див. рис. 2.16).

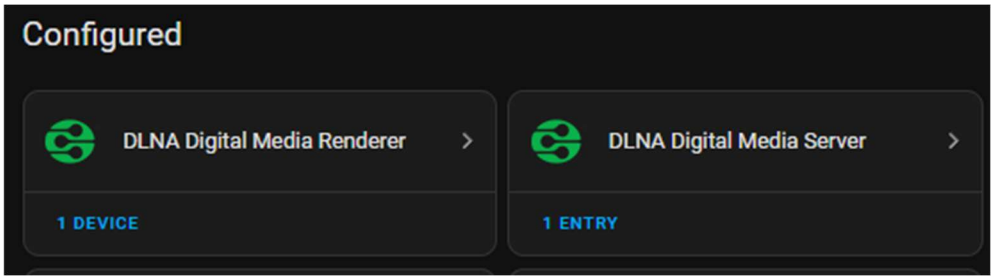


Рисунок 2.16 – Зконфігурований медіасервери DLNA

Home Assistant автоматично сканував мережу на наявність пристроїв та сервісів, зокрема TP-Link Archer AX20, який мав USB-порт та підтримував Samba-сервер. Для тесту було підключено USB-накопичувач, після чого маршрутизатор з’явився у Home Assistant як доступне медіа-джерело (див. рис. 2.17).

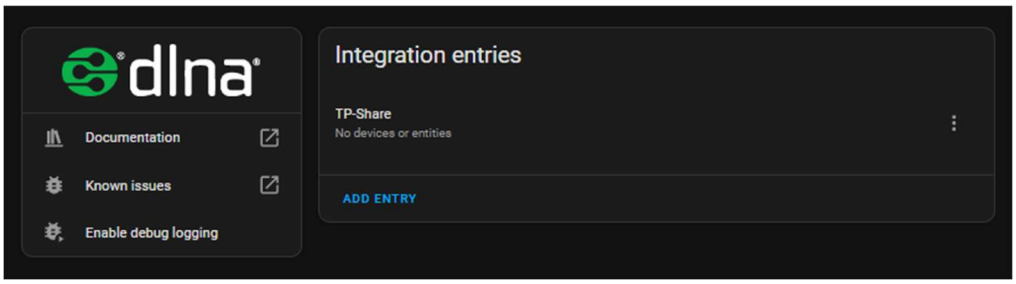


Рисунок 2.17 – Підв’язана інтеграція TP-Share від маршрутизатора TP-Link Archer AX20 для роботи зі протоколом SMB

Додатково до системи було підключено телевізор, який також підтримував DLNA. Після сканування телевізор було визначено як пристрій Digital Media Renderer, що дозволило транслювати контент без встановлення додаткових застосунків (див. рис. 2.18).

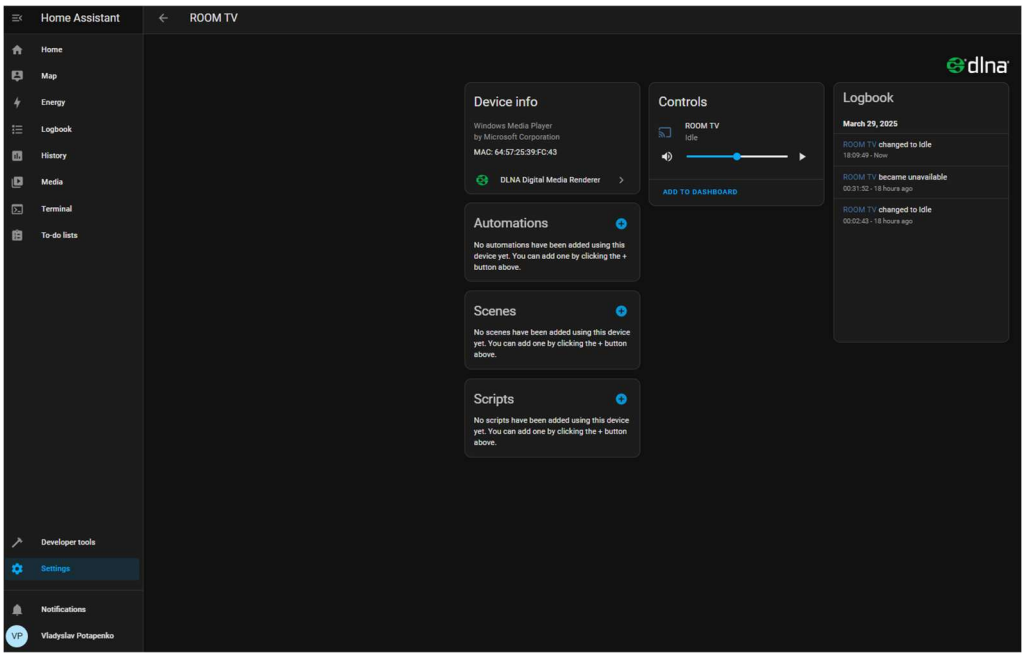


Рисунок 2.18 – Підв’язаний телевізор через інтеграцію з DLNA Digital Media Recorder

Через вкладку "Media" у Home Assistant було обрано DLNA-сервер, після чого відеоконтент із USB-накопичувача було запущено на телевізорі (див. рис. 2.19).

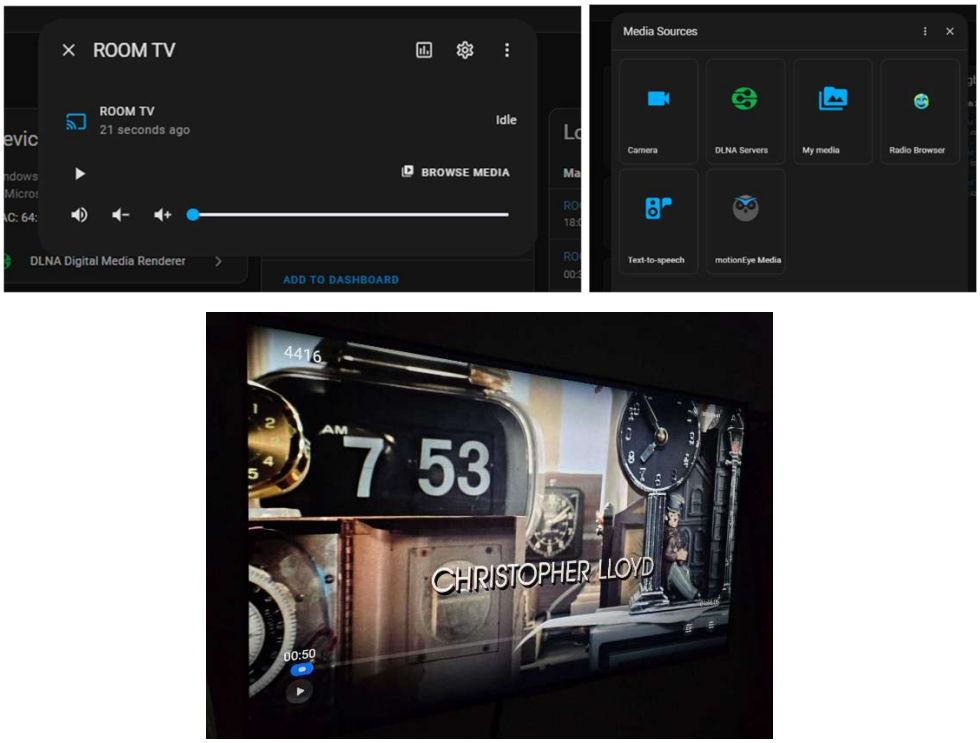


Рисунок 2.19 – Передача медіаконтенту на інший пристрій через медіасервер DLNA

Навантаження на сервер залишалося мінімальним, оскільки не відбувалося транскодування, а лише передача файлів через мережу (див. рис. 2.20).



Рисунок 2.20 – Метод оцінки використання віртуалізованого домашнього сервера у якості сервера розумного дому

Другий сценарій передбачав запуск Ubuntu Desktop як віртуальної машини, що дозволило оцінити поведінку сервера під час інтерактивної роботи користувача (див. рис. 2.21).



Рисунок 2.21 – Робота віртуальної машини Ubuntu Desktop у якості перевірки поведінки на систему

Після запуску системи було відкрито веб-браузер і запущено потокове відео, що створило навантаження на центральний процесор, оперативну пам’ять, мережевий інтерфейс та дискову підсистему (див. рис. 2.22).

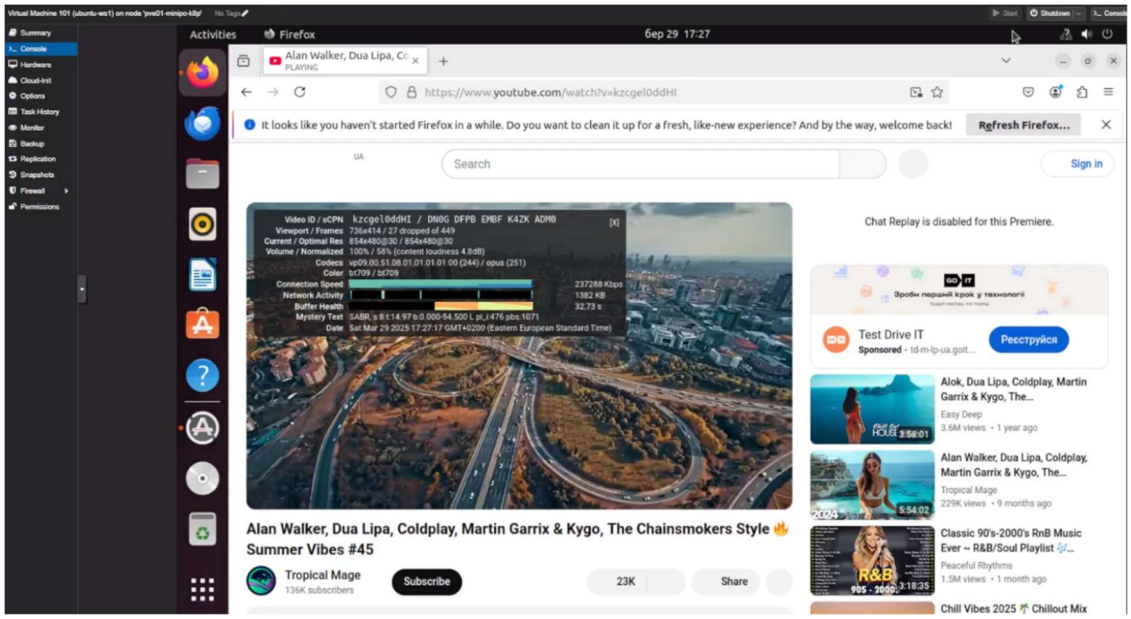


Рисунок 2.22 – Перегляд відеоролика на YouTube як навантаження на вузол

Паралельно моніторинг системи фіксував рівень споживання ресурсів, що дозволило оцінити, як розподіл навантаження впливає на продуктивність усієї інфраструктури (див. рис. 2.23).

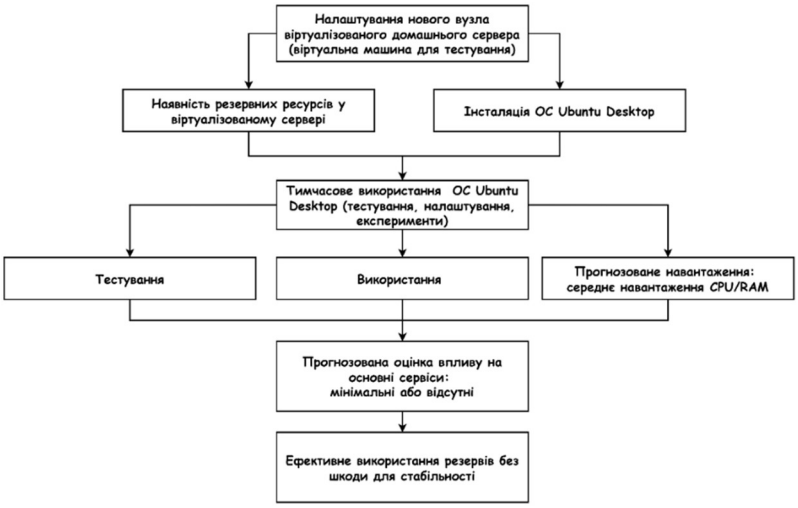


Рисунок 2.23 – Метод оцінки використання віртуалізованого домашнього сервера у якості віртуальної машини для тестування

Третій тестовий сценарій, а саме тестування OPNsense проводилося як експеримент використання віртуалізованого маршрутизатора та міжмережевого екрану (Firewall). Наявність додаткових мережевих портів на пристрої дозволяє призначати їх для окремих підмереж або VLAN, що робить OPNsense гнучким рішенням для організації мережевого середовища.

Запуск OPNsense у середовищі Proxmox на Mini PC виглядає надлишковим для типового домашнього використання, оскільки наявний сервер має значно більший запас ресурсів, ніж потрібно для роботи такого маршрутизатора. Проте тестування показало загальну картину взаємодії різних сервісів у межах єдиного середовища. Інтернет-з’єднання здійснювалося через провайдера Київстар, і вплив роботи OPNsense на загальне навантаження системи залишався мінімальним. Фільтрація трафіку, аналіз з’єднань та інші мережеві функції працювали у штатному режимі без суттєвого збільшення споживання ресурсів (див. рис. 2.24).

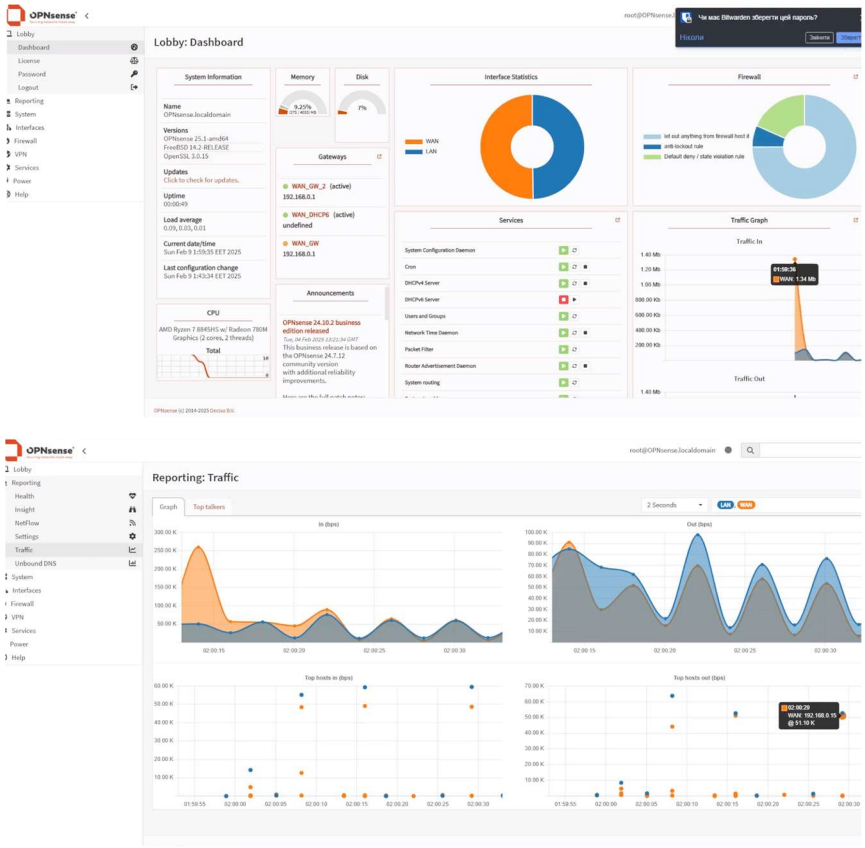


Рисунок 2.24 – Робота віртуальної машини OPNsense
у якості відслідковування та фільтрації небажаного трафіку

У стандартному режимі використання OPNsense виконує лише базові функції маршрутизації, а навантаження залежить від додаткових правил фільтрації, VPN-з’єднань або інших активних модулів (див. рис. 2.25).

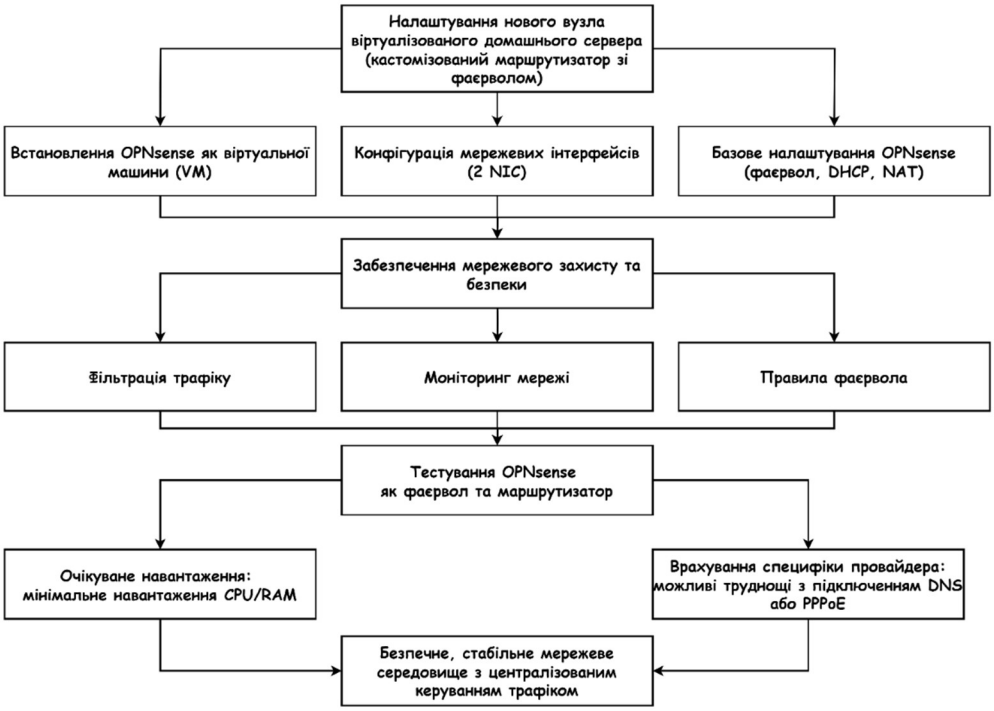


Рисунок 2.25 – Метод оцінки використання віртуалізованого домашнього сервера у якості кастомізованого маршрутизатора зі фаєрволом

Четвертий сценарій передбачав використання операційної системи Ubuntu Server, яка дозволяла керувати контейнерами, у яких запускалися різні сервіси. Основними тестовими додатками стали AdGuard Home, який відповідає за блокування реклами та аналіз DNS-запитів, і Jellyfin як медіасервер для потокової трансляції відео. AdGuard Home має низькі вимоги до ресурсів і може запускатися навіть на слабких пристроях, таких як Raspberry Pi, тому його вплив на загальне навантаження був мінімальним (див. рис. 2.26).

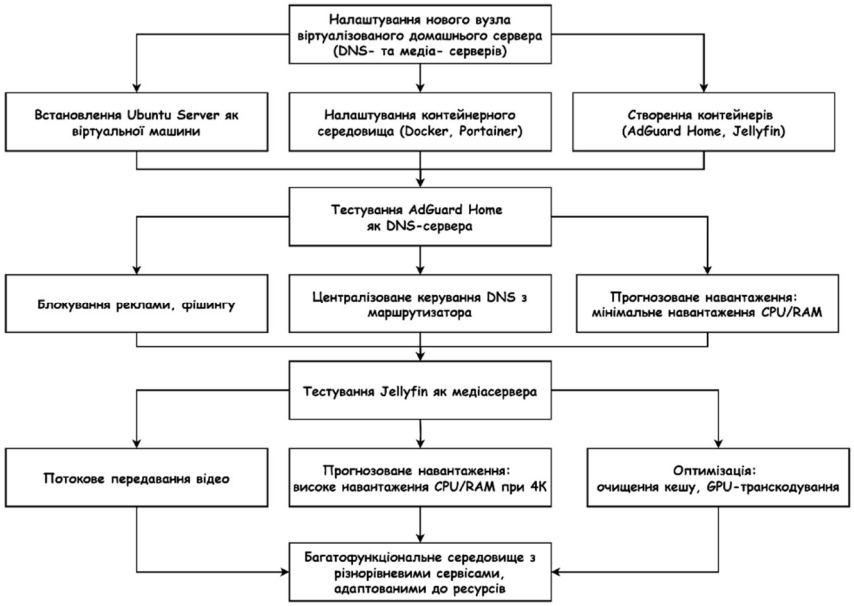


Рисунок 2.26 – Метод оцінки використання віртуалізованого домашнього сервера у якості DNS- та медіа- серверів

Jellyfin, навпаки, створює суттєве навантаження, особливо під час транскодування відео. На відміну від DLNA-сервера в Home Assistant, який просто передає медіафайли без обробки, Jellyfin виконує перекодування відео в режимі реального часу залежно від можливостей клієнтського пристрою. Для тестування було використано відеофайл у форматі 4K, який відтворювався на телевізорі без змін, а також на смартфоні, що вимагало зниження якості та додаткового перетворення, що спричинило значне зростання навантаження на процесор і оперативну пам’ять сервера (див. рис. 2.27).



Рисунок 2.27 – Робота Docker контейнера Jellyfin у якості медіа-сервера (зі транскодуванням)

Моніторинг продуктивності показав, що під час запуску Jellyfin сервер працював із максимальним навантаженням, особливо у випадку одночасного відтворення на кількох пристроях. Усі сервіси запускалися після перезавантаження системи, що дозволило відстежити поведінку ресурсів у різні моменти роботи. Деякі тести фіксувалися одразу після старту сервера, а інші – через годину або більше після його безперервної роботи.

Контейнер AdGuard Home мав низьке навантаження на процесор і оперативну пам’ять, працюючи в середовищі віртуальної машини з Ubuntu Server під керуванням Proxmox та Portainer. Швидкість обробки DNS-запитів залишалася високою, а продуктивність не викликала жодних нарікань. Додатково тестувався Pi-hole, але через надмірне блокування він перешкоджав коректній роботі деяких сервісів, таких як Megogo, служби доставки їжі, виклику таксі та моніторингу громадського транспорту. AdGuard Home не мав подібних проблем, що зробило його більш зручним у використанні (див. рис. 2.28).

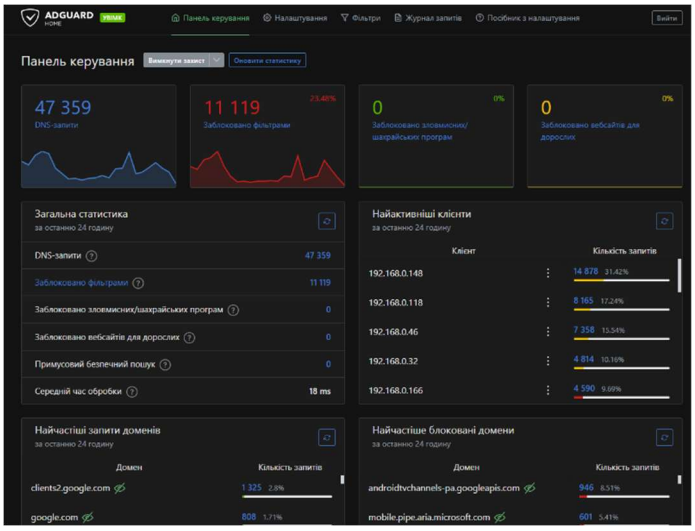


Рисунок 2.28 – Робота Docker контейнера Adguard Home
у якості DNS-сервера (блокувальника реклами та фішингових сайтів)

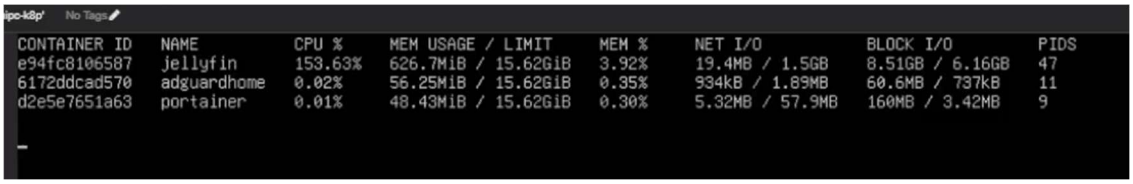
Під час налаштування DNS-сервера важливо враховувати особливості роботи провайдерів, оскільки деякі з них можуть блокувати зміну DNS на сторонні сервери. Наприклад, у деяких провайдерів є механізми, які автоматично перенаправляють DNS-запити на власні сервери або вимагають підтвердження

використання стороннього DNS. У випадку провайдера Ланет спостерігався подібний нюанс (див. рис. 2.29).



Рисунок 2.29 – Деякі провайдери ставлять обмеження на використання сторонніх DNS-серверів

Використання Ubuntu Server у вигляді віртуальної машини для керування Jellyfin та іншими сервісами дало можливість контролювати споживання ресурсів. Аналізуючи роботу контейнерів через Docker через "sudo docker stats", було зафіксовано різке збільшення навантаження під час активного транскодування, що вимагало втручання (див. рис. 2.30).



CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
e94fc8106587	jellyfin	153.63%	626.7MiB / 15.62GiB	3.92%	19.4MB / 1.5GB	8.51GB / 6.16GB	47
6172ddcad570	adguardhome	0.02%	56.25MiB / 15.62GiB	0.35%	934kB / 1.89MB	60.6MB / 737kB	11
d2e5e7651a63	portainer	0.01%	48.43MiB / 15.62GiB	0.30%	5.32MB / 57.9MB	160MB / 3.42MB	9

Рисунок 2.30 – Стрімке зростання споживання та резервування RAM Docker контейнером Jellyfin у вигляді неконтрольваного кешу

Для підвищення ефективності відтворення медіа було рекомендовано використовувати апаратне декодування через GPU, що зменшує навантаження на центральний процесор. Додатково важливу роль відіграє висока швидкість мережевого з’єднання, оскільки для плавного потокового передавання контенту

необхідний інтернет зі швидкістю 1 Гбіт/с і більше. У цьому випадку застосовувалося підключення зі швидкістю 2,5 Гбіт/с, що забезпечувало безперебійну роботу сервісу.

Збереження медіаконтенту здійснювалося на SSD-диску, що значно прискорювало доступ до файлів і забезпечувало швидке завантаження навіть об’ємних відеофайлів. Однак, у випадку масштабування сервера для зберігання великої кількості контенту доцільніше використовувати HDD-диски з великим об’ємом, що більше підходить для стаціонарних серверів. Загалом, Jellyfin став сервісом, який створював найбільше навантаження серед усіх протестованих рішень.

Розподіл ресурсів і стратегії оптимізації. Також необхідно враховувати розподіл ресурсів та стратегії оптимізації, оскільки детальніше було розглянуто конфігурації, застосовані у сервері. Аналіз розподілу центрального процесора включав визначення доступної кількості віртуальних ядер, які могли бути використані для віртуальних машин. У більшості випадків не відбувалося жорсткого закріплення ядер за конкретними віртуальними машинами, а лише вказувалася кількість доступних процесорів.

Усі сценарії використання, включно з Home Assistant, OPNsense, Ubuntu Server (Portainer) (див. рис. 2.31-2.32) та іншими сервісами, покладалися на керування ресурсами самим гіпервізором. У такій конфігурації ядра процесора призначалися віртуальним машинам динамічно, залежно від поточного завантаження, без статичного закріплення.

Розподіл оперативної пам’яті (див. рис. 2.31-2.32) вимагав особливого підходу, оскільки надмірне перевиділення ресурсів могло призвести до вичерпання пам’яті всієї системи. Обмеження використання оперативної пам’яті для кожної віртуальної машини запобігало її неконтрольованому споживанню, а можливість динамічного розширення дозволяла в будь-який момент змінювати її обсяг відповідно до навантаження.

Максимальне навантаження на пам’ять (див. рис. 2.31-2.32) спостерігалося у медіасервері, розгорнутому в контейнері. Хоча контейнеризовані додатки зазвичай

мають невеликі вимоги до ресурсів, інтенсивна обробка відео та транскодування призводили до споживання всієї виділеної оперативної пам’яті. Оптимізація включала скидання кешу програм через певні інтервали часу, щоб звільняти пам’ять і запобігати її надмірному використанню.

Щодо дискового простору (див. рис. 2.31-2.32), всі робочі навантаження працювали на NVMe SSD, який забезпечував швидкий доступ до даних. Найкраще використовувати SSD для операційної системи та активних процесів, а для зберігання великих обсягів даних доцільніше застосовувати жорсткі диски, щоб уникнути швидкого зносу SSD-накопичувачів.



Рисунок 2.31 – Навантаження на кожен вузол системи



Рисунок 2.32 – Загальне максимальне навантаження на сервер

Обмеженням Mini PC стала можливість розширення дискового простору – максимальна конфігурація передбачала використання двох M.2 SSD по 4 ТБ, що в сумі давало 8 ТБ. Підключення додаткових накопичувачів через USB не

розглядалося через низьку пропускну здатність, що значно збільшувало час обробки даних.

Оптимізація мережі включала правильний розподіл IP-адрес та портів. Кожен контейнер отримував власну підмережу, що дозволяло йому взаємодіяти з іншими сервісами у внутрішній мережі. Віртуальні машини отримували окремі локальні IP-адреси.

Ubuntu Server мав одну основну IP-адресу, яка використовувалася для доступу через веб-інтерфейс. Для забезпечення доступу до сервісів, таких як AdGuard Home чи Jellyfin, використовувалися різні порти на цій самій IP-адресі. AdGuard Home працював через стандартний порт 80, тоді як Jellyfin – через порт 8096, що дозволяло заходити на сторінку медіасервера та отримувати доступ до контенту. Home Assistant працював на іншій IP-адресі по порту 8123, також це стосувалось і OPNsense: інша IP-адреса та порт 80. До того ж Proxmox також працює з використанням унікального IP-адреса та порта 8006 (див. рис. 2.33).

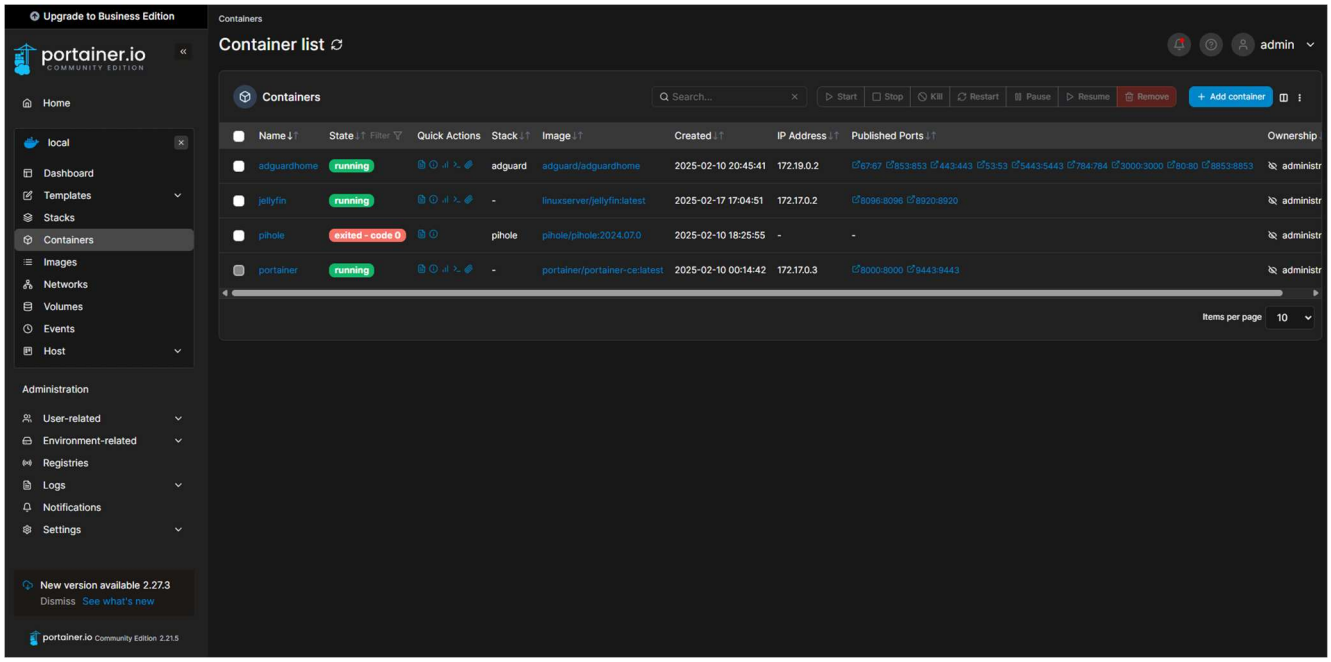


Рисунок 2.33 – Присвоєння вільних портів для Docker контейнерів через Portainer

2.3 Засоби моніторингу та методи аналізу продуктивності

Аналіз отриманих даних дозволяє зрозуміти, наскільки поточна конфігурація відповідає вимогам навантаження, а також чи є можливості для подальшої оптимізації ресурсів.

При оцінці продуктивності з’являється можливість глибше зрозуміти, наскільки виправдане використання конкретного Mini PC для розгортання віртуалізаційного середовища. Аналіз дозволяє відповісти на питання, чи можна використовувати менш потужний, але дешевший пристрій, який все ж здатний впоратися з покладеними на нього завданнями.

Ключові показники ефективності. Завантаженість центрального процесора є одним із ключових параметрів, що відображає ефективність використання ресурсів віртуальними машинами та контейнерами. Чим більше навантаження на процесор, тим більша ймовірність зниження загальної продуктивності, збільшення затримок і навіть перегріву системи. Основні фактори, що впливають на навантаження, включають кількість виділених віртуальних ядер для кожної віртуальної машини, накладні витрати гіпервізора, інтенсивні завдання, такі як перекодування відео у Jellyfin, а також методи планування процесів. Враховуючи обмеженість доступних ресурсів, важливо правильно розподіляти обчислювальні потужності, оскільки перевищення допустимого рівня завантаженості може призвести до серйозних наслідків, включаючи нестабільність роботи віртуалізованих середовищ (див. рис. 2.34).

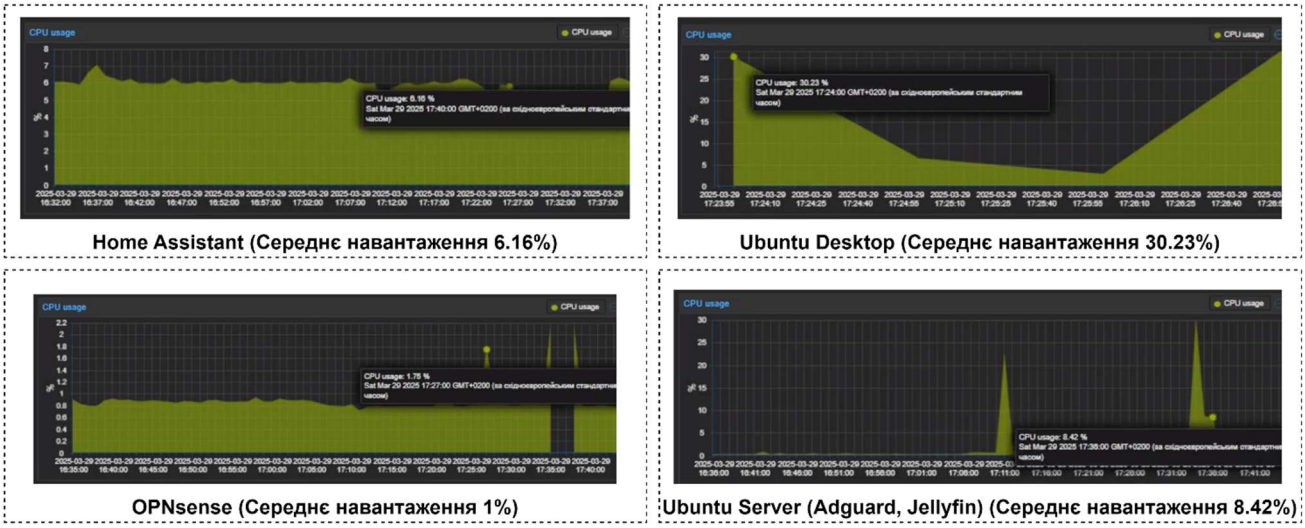


Рисунок 2.34 – Середнє навантання на CPU окремими вузлами

Окрім процесора, важливим фактором є споживання оперативної пам’яті, яка також є обмеженим ресурсом. Використання 32 ГБ DDR5 передбачає необхідність ретельного керування пам’яттю, оскільки перевантаження може спричинити затримки, зависання системи або навіть аварійні перезавантаження. Неправильний розподіл ресурсів здатен викликати активне використання підкачки, що суттєво знижує загальну продуктивність (див. рис. 2.35).

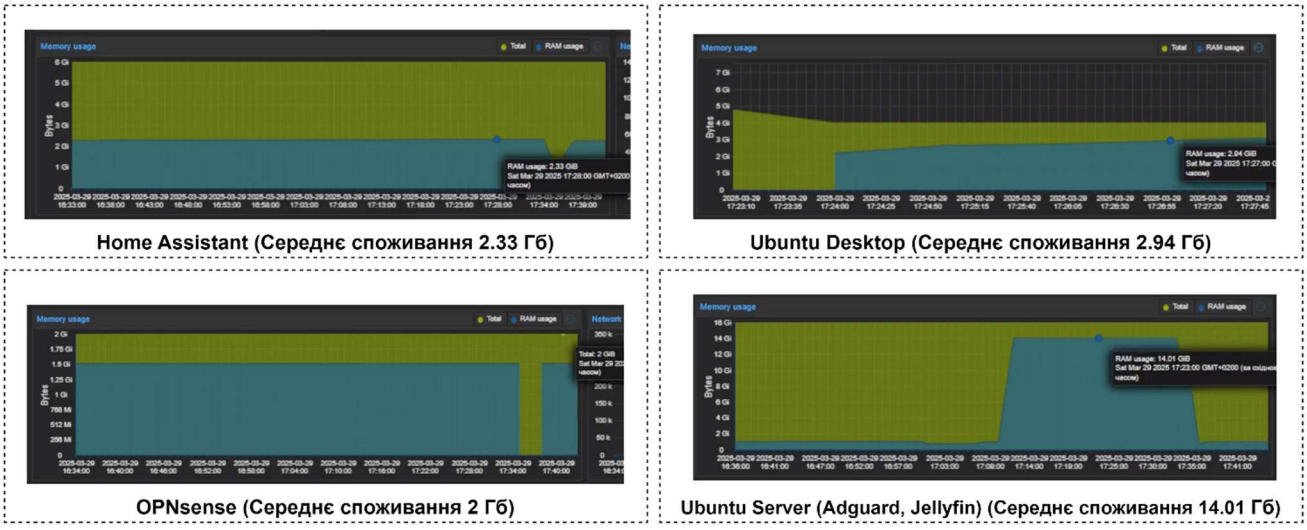


Рисунок 2.35 – Середнє навантання на RAM окремими вузлами

Для оптимального керування пам’яттю бажано використовувати статичний розподіл, визначаючи фіксований обсяг RAM для кожної віртуальної машини або

контейнера. Враховуючи, що всі інстанси запускаються одночасно, загальний обсяг виділеної пам’яті не повинен перевищувати наявний фізичний ресурс. Недотримання цього принципу може спричинити нестабільність роботи сервера та суттєве зниження швидкодії.

Найбільше оперативної пам’яті споживають сервіси, що працюють з мультимедійним контентом. Наприклад, медіасервер Jellyfin активно використовує ресурси системи, що потребує ретельного моніторингу та періодичного очищення кешу, яке дозволяє уникати перевантажень.

Окрім оперативної пам’яті, важливим параметром залишається швидкодія дискової підсистеми, оскільки продуктивність операцій введення-виведення безпосередньо впливає на швидкість роботи програм, особливо тих, що інтенсивно працюють із даними. До таких можна віднести зберігання файлів, операції з базами даних, потокове передавання та будь-яку активність, що вимагає швидкого доступу до накопичувачів. Повільна робота дискової підсистеми може призводити до затримок, довготривалого пошуку файлів і нестабільної роботи програмного забезпечення (див. рис. 2.36-2.37).

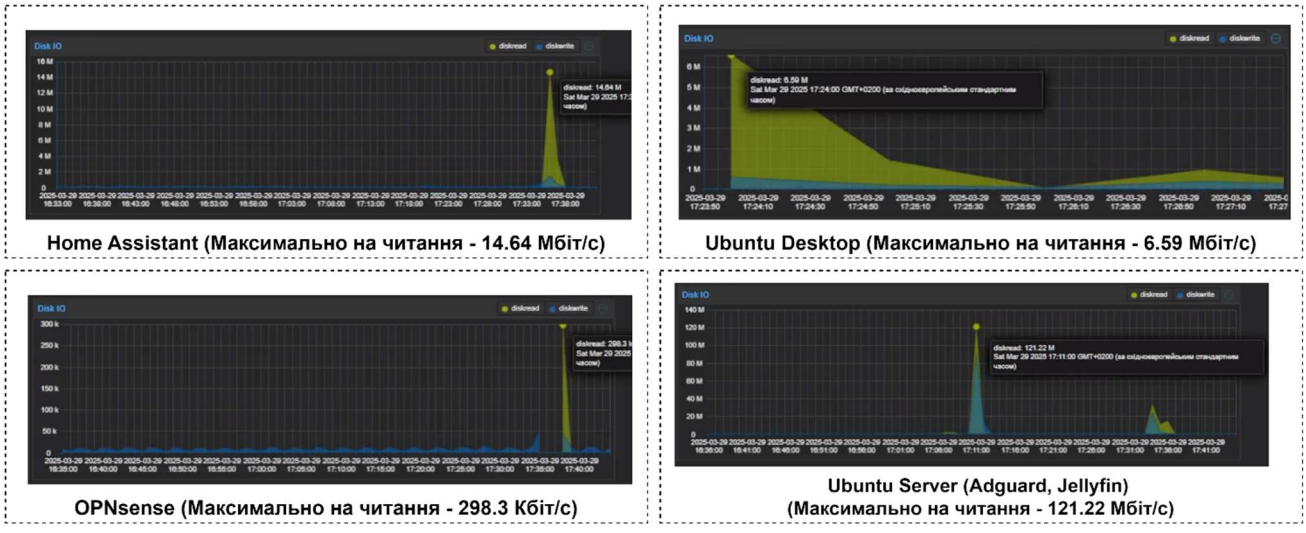


Рисунок 2.36 – Максимальне навантання (читання) на SSD окремими вузлами

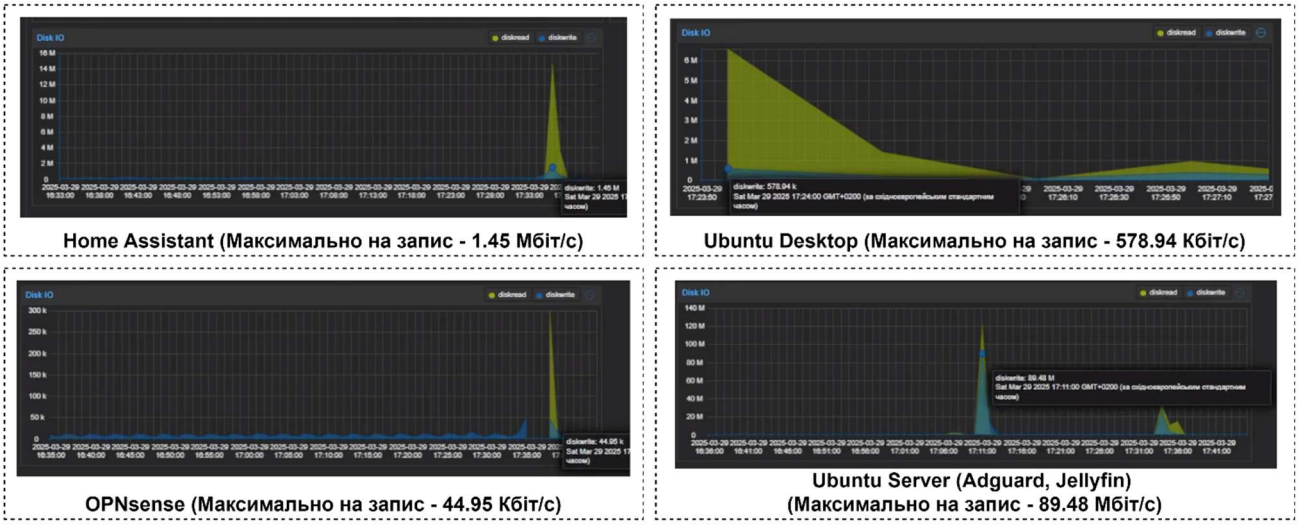


Рисунок 2.37 – Максимальне навантання (запис) на SSD окремими вузлами

Серед ключових факторів, що впливають на продуктивність дисків, можна виділити відмінності між SSD і HDD у віртуалізованому середовищі, тип інтерфейсу підключення накопичувачів, а також спосіб їхньої взаємодії з іншими компонентами системи. Використання DAS-сховища, підключеного через USB, значно обмежує швидкість передачі даних, оскільки навіть у найкращих умовах цей варіант демонструє суттєво нижчу продуктивність порівняно з прямим підключенням через SATA або NVMe.

Високі дискові навантаження можуть виникати через інтенсивні процеси введення-виведення, такі як синхронізація файлів, робота баз даних або функціонування хмарних сховищ. Планування операцій введення-виведення також впливає на продуктивність, оскільки використання різних стратегій хешування та кешування може як покращити, так і погіршити загальну швидкодію системи. Наприклад, для контейнеризованого медіасервера продуктивність файлового введення-виведення безпосередньо залежить від оптимізації кешу в Linux [3, 12].

Для вимірювання швидкодії дискової підсистеми у Proxmox передбачено вбудований моніторинг активності накопичувачів, який дозволяє відстежувати дискове навантаження для кожної віртуальної машини та контейнера, а також переглядати аналітику в поєднанні з іншими ресурсами, такими як центральний процесор і оперативна пам’ять [14-16].

Окрім дискової продуктивності, важливим фактором є пропускна здатність мережі та затримки при передачі даних, що можуть суттєво впливати на роботу віртуалізованого середовища. Для забезпечення ефективного функціонування необхідний стабільний зв’язок між віртуальними машинами, фізичними пристроями та зовнішніми службами. Повільна робота мережі може спричиняти довге завантаження веб-сайтів, проблеми з буферизацією під час потокового передавання відео, а також нестабільну роботу хмарних сервісів (див. рис. 2.38).

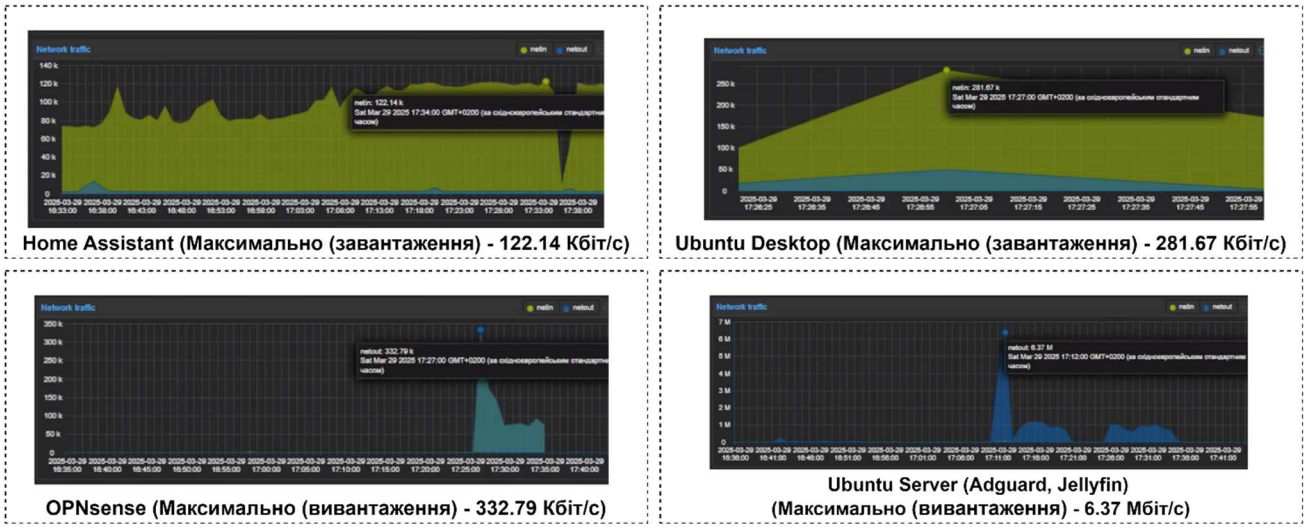


Рисунок 2.38 – Максимальне навантання на мережу (завантаження/вивантаження) окремими вузлами

Продуктивність мережевих з’єднань залежить від таких факторів, як розподіл пропускної здатності між віртуальними машинами та службами, конфігурація віртуальних мережевих інтерфейсів, а також навантаження від програм, що активно використовують мережеві ресурси. Наприклад, потокове передавання відео у Jellyfin із високим бітрейтом або синхронізація великих обсягів даних через хмарне сховище можуть створювати значне навантаження на мережу, що ускладнює роботу інших сервісів.

Методи збору даних. Після визначення ключових показників продуктивності наступним етапом стає впровадження стратегії збору даних за допомогою як вбудованих засобів моніторингу, так і сторонніх інструментів аналітики.

Використання вбудованих інструментів у Proxmox дозволяє отримувати інформацію в режимі реального часу про навантаження на центральний процесор, використання оперативної пам’яті, продуктивність дисків і мережевий трафік. Ці засоби корисні для базової діагностики системи та початкового усунення несправностей [14-16]

Proxmox містить вбудовані засоби для аналізу продуктивності не лише окремого сервера, а й конкретних віртуальних машин, що дозволяє переглядати статистику використання ресурсів у різних часових проміжках (див. рис. 2.39).

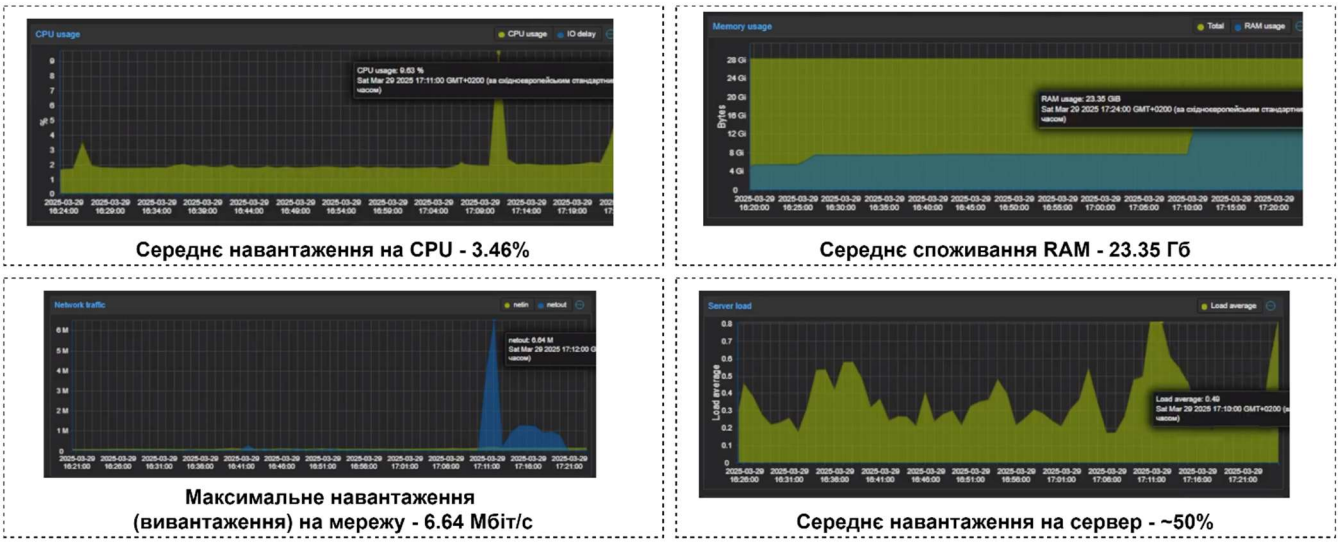


Рисунок 2.39 – Загальне навантаження на сервер
під час навантаження всіма вузлами

За допомогою вбудованих інструментів можна переглядати статистику навантаження за днями, годинами, місяцями та навіть роками (див. рис. 2.40). У разі використання кластерної архітектури забезпечується можливість моніторингу не лише окремих вузлів, а й усієї системи в цілому [14-16].



Рисунок 2.40 – Перегляд статистики серед усіх вузлів по навантаженю: година, день, тиждень, рік (мінімум, середнє, максимум)

2.4 Аналіз продуктивності основних ресурсів сервера

Проводячи попереднє тестування компактного Mini PC у якості сервера, необхідно оцінити, наскільки такі пристрої виправдовують себе у ролі серверного обладнання та чи є сенс розглядати альтернативні, більш доступні варіанти.

Для оцінки продуктивності було розгорнуто Proxmox з кількома віртуальними машинами та контейнерами. Контейнери не використовувалися в нативному середовищі Proxmox, а були розгорнуті всередині віртуальних машин через Docker, що дозволить зберігати гнучкість керування сервісами без необхідності глибокої інтеграції в систему Proxmox.

Таблиця 2.1 – Загальні показники продуктивності серверного вузла та віртуальних машин

№	Сервер/Віртуальна машина	Вузол Proxmox	Home Assistant OS	Ubuntu Desktop	OPNsense	Ubuntu Server (Portainer – Adguard Home – Jellyfin)
1	Навантаження на Процесор (максимум)	9.63 %	7.06 %	32.65 %	2.11 %	29.98 %
2	Навантаження на Процесор (середнє)	3.46 %	6.16 %	6 %	1.75 %	8.42 %
3	Навантаження на Процесор (мінімум)	1.66 %	3.66 %	2.76 %	0.8 %	0.68 %
4	Використання ОЗП (доступно)	28.22 Гб	6 Гб	4 Гб	2 Гб	16 Гб
5	Використання ОЗП (максимум)	20.79 Гб	2.33 Гб	2.94 Гб	1.51 Гб	14.01 Гб
6	Використання ОЗП (середнє)	7.66 Гб	2 Гб	2.66 Гб	1.50 Гб	1.50 Гб
7	Використання ОЗП (мінімум)	5.45 Гб	770.85 Мб	2.33 Гб	1.34 Гб	1.01 Гб
8	Навантаження на Мережу (максимум)	6,8 Мбіт/с	122.14 Кбіт/с	281.67 Кбіт/с	333.79 Кбіт/с	6.37 Мбіт/с
9	Навантаження на Мережу (середнє)	1.38 Мбіт/с	120.27 Кбіт/с	154.43 Кбіт/с	80.28 Кбіт/с	1.05 Мбіт/с
10	Навантаження на Мережу (мінімум)	12.98 Кбіт/с	10.6 Кбіт/с	812,65	759.65	818.68
11	Навантаження на Диск (читання - максимум)	-	14.64 Мбіт/с	6.59 Мбіт/с	298.3 Кбіт/с	121.22 Мбіт/с
12	Навантаження на Диск (запис - максимум)	-	1.45 Мбіт/с	578.94 Кбіт/с	44.95 Кбіт/с	89.48 Мбіт/с
13	Навантаження на Диск (читання - мінімум)	-	0	81,59 Кбіт/с	273.07 Кбіт/с	33.33 Мбіт/с
14	Навантаження на Диск (запис - мінімум)	-	138.48 Кбіт/с	81,59 Кбіт/с	12.83 Кбіт/с	24.74 Мбіт/с
15	Навантаження на Сервер (максимум)	0.82	-	-	-	-
16	Навантаження на Сервер (середнє)	0.49	-	-	-	-
17	Навантаження на Сервер (мінімум)	0.21	-	-	-	-

Аналіз результатів тестування – процесорне навантаження. Аналізуючи навантаження на процесор, найвищі пікові значення зафіксовані у віртуальних машин з Ubuntu Desktop (32,65%) та Ubuntu Server з контейнерами (29,98%), що напряму пов’язано з віртуалізацією робочого середовища у випадку з Ubuntu

Desktop та роботою потокових відеосервісів, таких як Jellyfin, який виконує транскодування відео. Саме ці два сценарії є найбільш ресурсомісткими у плані навантаження на центральний процесор.

Причина таких показників полягає у розподілі ресурсів: Ubuntu Desktop отримав лише два ядра, тоді як Ubuntu Server мав шість ядер через вищі вимоги до обчислювальних потужностей, пов’язані з роботою контейнерів, що дозволяє краще балансувати ресурси між віртуальними машинами, але водночас створює обмеження, що можуть впливати на продуктивність. Незважаючи на фіксацію максимального навантаження, воно не є постійним, а лише піковим у певні моменти роботи.

Якщо аналізувати навантаження OPNsense та Home Assistant, вони використовували мінімальні ресурси процесора. OPNsense показав максимальне навантаження у 2,11%, а Home Assistant – 7,06%. Ці значення свідчать про низьку потребу у процесорних ресурсах, що дозволяє запускати їх навіть у легких Linux-контейнерах без значного впливу на загальну продуктивність системи.

Розподіл виділених ресурсів також вплинув на результати: Home Assistant отримав чотири ядра, хоча ефективніше було б обмежитися двома, оскільки його навантаження не потребує значної обчислювальної потужності. OPNsense отримав два ядра, що є достатнім для його роботи.

Аналіз результатів тестування – використання оперативної пам'яті (RAM). Перехід до аналізу використання оперативної пам’яті показує, що найбільше споживання знову зафіксовано у Ubuntu Server через роботу контейнерів, зокрема Jellyfin. Аналіз за допомогою docker stats продемонстрував, що саме цей сервіс є найбільш ресурсомістким, оскільки займається транскодуванням відео з високої роздільної здатності у нижчу.

Максимальне споживання оперативної пам’яті досягло 14,01 ГБ і залишалося на цьому рівні впродовж тривалого часу. Причиною такого стабільно високого навантаження є кешування відео в оперативній пам’яті, що дозволяє миттєвий доступ до контенту без додаткових затримок. Проблема полягала у тому, що кеш

не вивантажувався автоматично, і єдиним способом його очищення було перезавантаження системи.

З метою оптимізації було внесено зміни у налаштування Linux, що дозволили автоматично очищати кеш при досягненні певного рівня заповнення, що дозволило уникнути зайвого споживання пам’яті та покращити стабільність системи при тривалому використанні.

Додатково було протестовано Docker Desktop, який також використовував значну кількість оперативної пам’яті. Його максимальні показники сягали 2,94 ГБ, що є очікуваним для віртуалізованого середовища на базі Ubuntu Desktop. Враховуючи, що тестування проводилося під час перегляду відео на YouTube, такі показники є виправданими.

Home Assistant та OPNsense демонстрували мінімальне споживання оперативної пам’яті. Home Assistant використовував від 770 до 850 МБ у звичайному режимі, з піковими значеннями до 2,33 ГБ, що траплялося лише один раз. OPNsense працював стабільно в межах 1,34 ГБ, із максимальним використанням до 1,51 ГБ.

Загалом використання ресурсів Home Assistant та OPNsense залишалося на низькому рівні, що дозволяє запускати їх навіть на менш потужних пристроях у середовищі Proxmox, що підтверджує можливість ефективної роботи таких сервісів на бюджетних домашніх серверах без суттєвих обмежень продуктивності.

Аналіз результатів тестування – дискове навантаження. Аналізуючи дискове навантаження, найбільше споживання ресурсів зафіксовано у Jellyfin із показниками 121,22 МБ/с на читання та 89,48 МБ/с на запис, що очікувано для потокового медіасервера, який активно працює з файлами великого розміру.

Ubuntu Desktop демонстрував мінімальне навантаження на диск – 6,59 МБ/с на читання та 578,94 КБ/с на запис, що є вкрай малими значеннями у порівнянні з іншими тестованими системами. Аналіз показує, що рівень використання дискових ресурсів у цьому випадку не є критичним і не створює суттєвого впливу на продуктивність.

Найменше навантаження серед усіх учасників тесту мав OPNsense – 298,3 КБ/с на читання та 44,95 КБ/с на запис. Оскільки цей пристрій виконує роль маршрутизатора, більшість його операцій обробляється в оперативній пам’яті, а доступ до диску відбувається рідко. Схожа ситуація спостерігалася і з Home Assistant, який, хоча й мав дещо вищі значення, також належить до категорії систем із низьким дисковим навантаженням.

Додатково було протестовано сервер, що використовував Samba для файлового доступу. Читання з диска у цьому випадку досягало 14,64 МБ/с, а запис – 1,45 МБ/с. Використання такого сервера як простого файлового сховища без транскодування відео значно знижує навантаження у порівнянні з Jellyfin, який працює в іншому режимі та вимагає значних ресурсів.

Аналіз результатів тестування – мережеве навантаження. Перехід до аналізу мережевого навантаження демонструє, що найбільший вплив на трафік створював контейнер Jellyfin, який споживав 6,37 МБ/с, що логічно, оскільки потокове передавання відео потребує значних пропускних можливостей мережі.

Інші віртуальні машини мали значно нижчі показники. Наприклад, Home Assistant використовував лише 122,14 КБ/с, Ubuntu Desktop – 281,67 КБ/с, а OPNsense – 333,79 КБ/с. Основною причиною низького навантаження є відсутність інтенсивного обміну даними між цими системами та мережею.

В Ubuntu Server, де було розгорнуто контейнер Jellyfin, загальний трафік значно вищий через активне передавання мультимедійного контенту. Контейнер стабільно споживав значну частину мережевих ресурсів, що підкреслює його роль як основного навантажувача всієї системи.

OPNsense, хоча й посів друге місце за рівнем мережевого трафіку, все ж демонстрував стабільну роботу з навантаженням у 80,28 КБ/с – це досить низький показник, що свідчить про ефективну обробку мережевого трафіку без зайвого споживання ресурсів.

Аналіз результатів тестування – навантаження на вузол (сервер) GMKtec K8 Plus. Попередньо розглядалося споживання окремими віртуальними машинами та контейнерами, однак сам сервер працює під управлінням Proxmox, і його

продуктивність безпосередньо залежить від обчислювальних можливостей процесора, оперативної пам’яті, мережевого інтерфейсу та системи зберігання даних.

Процесорне навантаження під час тестування досягало максимуму в 9,63%, що є вкрай низьким показником навіть при одночасній роботі всіх віртуальних середовищ. Середнє значення знаходилося на рівні 3,46%, а мінімальне – 1,66%. Протягом більшості часу система працювала саме на середньому рівні навантаження, що вказує на значний запас обчислювальної потужності. Використання слабшого процесора могло б бути доцільним у подібному сценарії, оскільки доступних ресурсів більше, ніж фактично споживається.

Щодо оперативної пам’яті, встановлена конфігурація з 32 ГБ DDR5 забезпечувала максимальний об’єм у 28,22 ГБ, оскільки частина пам’яті зарезервована під потреби Proxmox. Під час пікового навантаження використовувалося 20,79 ГБ, що є суттєвим показником, особливо коли Jellyfin працював у режимі медіасервера. У середньому споживання становило 7,66 ГБ, а мінімальне – 5,45 ГБ. Аналіз показує, що оперативна пам’ять є основним обмежувальним фактором, оскільки при її збільшенні можна було б запускати більше ресурсомістких сервісів.

Мережеве навантаження складало максимум 6,8 Мбіт/с, що цілком очікувано, оскільки основне навантаження створював Ubuntu Server з Jellyfin. Середнє значення було на рівні 1,38 Мбіт/с, а мінімальне – 12,98 КБ/с, що спостерігалося під час простою системи. У загальному випадку навіть гігабітного каналу вистачає з великим запасом, оскільки більшість операцій відбувається локально, без активного виходу в інтернет.

Дискове навантаження оцінити складніше, оскільки кожен сервіс мав власний пул, а Proxmox безпосередньо не керував цими операціями. Максимальне загальне навантаження на сервер у пікові моменти досягало 82%, що спостерігалося при запуску системи. Середнє значення знаходилося на рівні 49%, а мінімальне – 21%. Високе споживання ресурсів пов’язане передусім із використанням оперативної пам’яті, яка є найбільш завантаженим компонентом.

Теоретично, збільшення її обсягу вдвічі могло б знизити загальне навантаження сервера до 25%.

Загальний висновок щодо GMKtec K8 Plus у ролі сервера демонструє, що 32 ГБ DDR5 та Ryzen 7 8845HS (8 ядер, 16 потоків) повністю справляються із навантаженнями. Використання мережі є мінімальним, а одного гігабітного каналу вистачає для всіх операцій. Локальна взаємодія між сервісами та клієнтами є основним сценарієм роботи, тому навіть за відсутності інтернет-з’єднання сервер функціонує стабільно.

Jellyfin залишається найресурсоємнішим сервісом, що генерує найбільше дискового навантаження. Крім нього, розгорнуті ще два контейнери – Portainer для керування іншими контейнерами та AdGuard для блокування реклами. Процесорне навантаження залишалося стабільним, і сервер майже ніколи не перевищував 50% використання доступних ресурсів.

У загальному підсумку можливо додавати додаткові сервіси з помірним споживанням ресурсів без ризику нестабільної роботи, однак для запуску ресурсомістких завдань обсяг оперативної пам’яті є критичним фактором, що обмежує потенціал системи.

РОЗДІЛ 3 ОПТИМІЗАЦІЯ, АПАРАТНА АЛЬТЕРНАТИВА ТА ПЕРСПЕКТИВИ РОЗВИТКУ ДОМАШНІХ СЕРВЕРІВ

3.1 Оптимізація середовища віртуалізації на основі тестування

Під час розгортання домашнього серверного середовища на базі Proxmox було виявлено ряд технічних викликів, що стосуються стабільності та ефективності роботи віртуалізованих сервісів. Основна увага під час оптимізації приділялася зменшенню надмірного споживання ресурсів, запобіганню конфліктам між контейнерами, а також забезпеченню стабільного функціонування DNS-сервісів.

Автоматичне очищення кешу в Linux. Однією з проблем стала надмірна активність кешування RAM з боку мультимедійного сервісу Jellyfin. Навіть після завершення трансляції відео кешований вміст залишався в оперативній пам’яті, що призводило до нестачі ресурсів для інших віртуальних машин та контейнерів. Для вирішення цієї проблеми було впроваджено механізм автоматичного очищення кешу операційної системи, що дозволяє динамічно звільняти пам’ять у фоновому режимі, що значно покращило загальну продуктивність та ефективність розподілу ресурсів між сервісами (див. рис. 3.1).

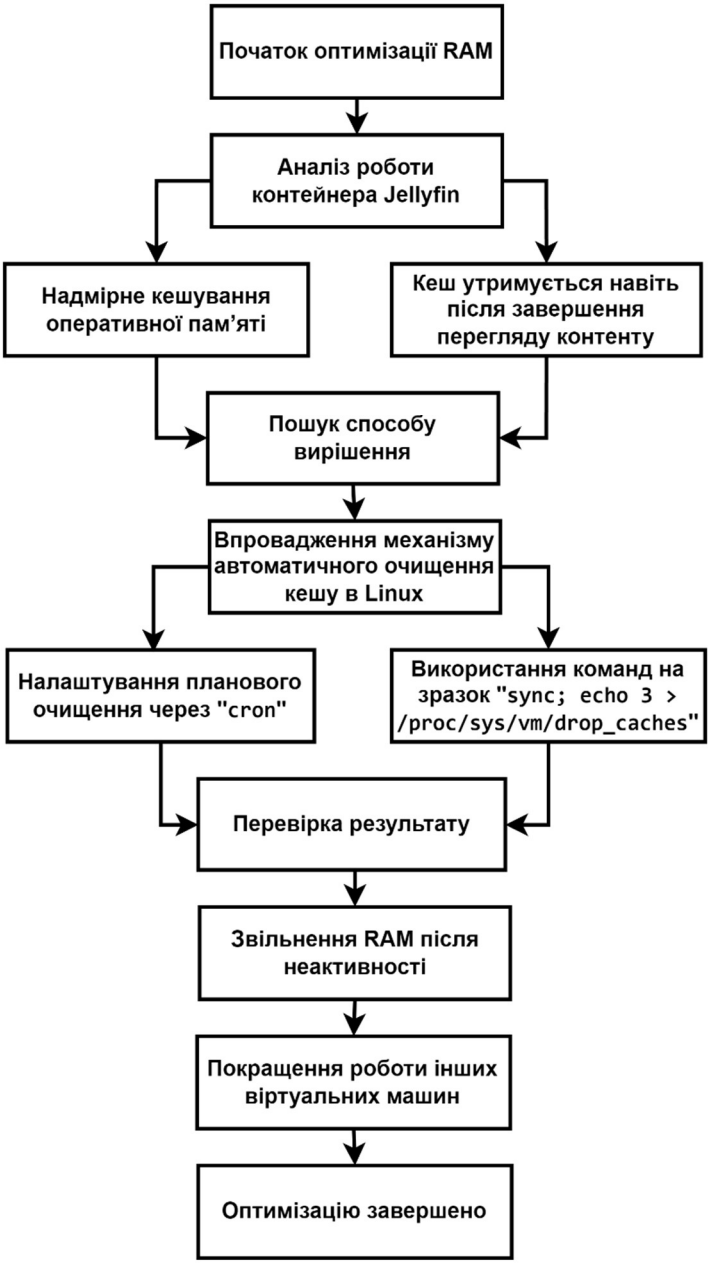


Рисунок 3.1 – Метод оптимізації навантаження на сервер шляхом впровадження автоматичного очищення кешу в Linux

Усунення конфліктів портів між контейнерами та віртуальними машинами. Іншою проблемою стали конфлікти портів між окремими контейнерами, які використовували однакові мережеві порти на спільному IP-адресі. Зокрема, віртуальна машина з Ubuntu Server мала активований вбудований DNS-сервер, що блокував роботу DNS-рішення на основі AdGuard Home. Після його повного вимкнення та переналаштування портів усіх контейнерів на унікальні

значення, було досягнуто стабільної роботи мережевих сервісів і зменшено ризик дублювання запитів чи конфліктів у системі (див. рис. 3.2).

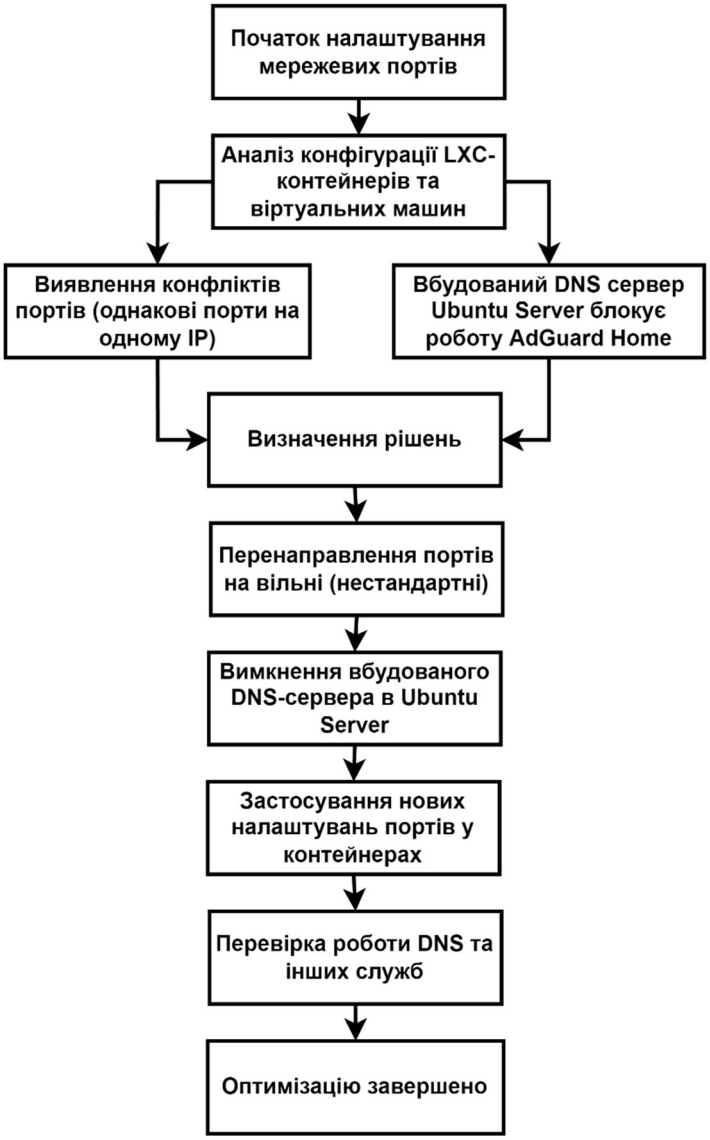


Рисунок 3.2 – Метод оптимізації навантаження на сервер шляхом усунення конфліктів портів між контейнерами та віртуальними машинами

Подальші можливості оптимізації. Крім описаних практичних заходів, існує низка методів, які можуть бути впроваджені в майбутньому для ще кращої продуктивності та стабільності домашнього серверного середовища.

1. *CPU Pinning (закріплення ядер процесора).* Закріплення фізичних ядер за конкретними віртуальними машинами дозволяє уникнути контекстного перемикання та забезпечити передбачувану продуктивність. Такий підхід

рекомендовано застосовувати до ВМ з високим навантаженням, як-от Jellyfin або Ubuntu Desktop, щоб мінімізувати затримки під час роботи з відео або при віддаленому підключенні (див. рис. 3.3).



Рисунок 3.3 – Метод оптимізації навантаження на сервер шляхом впровадження CPU Pinning (закріплення ядер процесора)

2. *RAM Ballooning (динамічний розподіл оперативної пам'яті)*. Цей метод дозволяє гіпервізору динамічно змінювати обсяг оперативної пам'яті, виділений для кожної ВМ, залежно від реальних потреб, що особливо корисно для серверів з обмеженим обсягом RAM, адже дозволяє ефективніше розподіляти ресурси між сервісами (див. рис. 3.4).

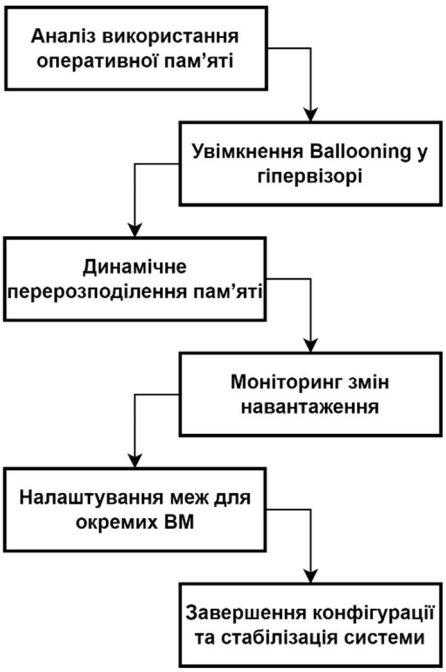


Рисунок 3.4 – Метод оптимізації навантаження на сервер шляхом впровадження RAM Ballooning (динамічний розподіл оперативної пам'яті)

3. *ZFS ARC Cache (кешування у файловій системі).* ZFS використовує ARC (Adaptive Replacement Cache) для зберігання часто запитуваних блоків даних у RAM, що зменшує кількість звернень до фізичних носіїв, що підвищить швидкість читання, особливо для медіасервісів, та знижує знос SSD-дисків (див. рис. 3.5).

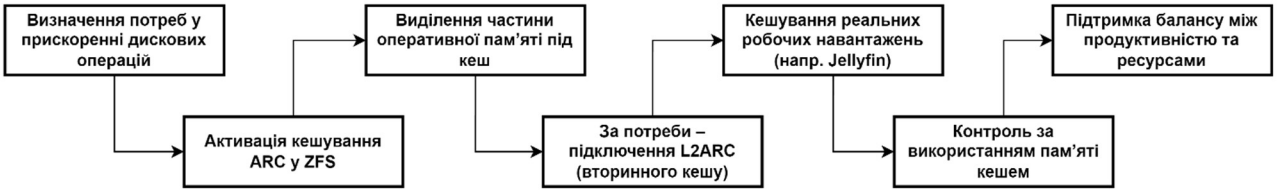


Рисунок 3.5 – Метод оптимізації навантаження на сервер шляхом впровадження ZFS ARC Cache (кешування у файловій системі)

4. *SR-IOV (зниження мережевої латентності).* SR-IOV дозволяє віртуальним машинам напряму взаємодіяти з фізичними мережевими адаптерами, минаючи шар емуляції, що значно знижить затримки та підвищує пропускну здатність, що критично важливо для сервісів типу OPNsense або VPN. Для його використання необхідно переконатися у підтримці SR-IOV як з боку мережевого адаптера, так і BIOS/UEFI. Альтернативою може слугувати PCIe Passthrough, що дає змогу повністю передати мережевий інтерфейс конкретній ВМ (див. рис. 3.6).

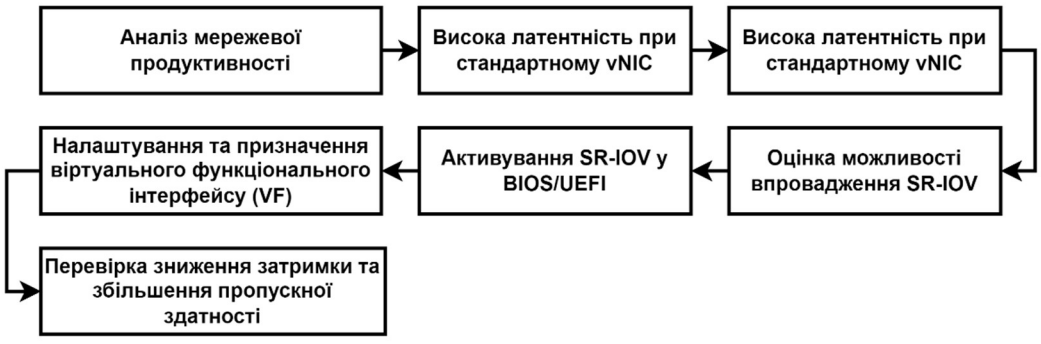


Рисунок 3.6 – Метод оптимізації навантаження на сервер шляхом впровадження SR-IOV (зниження мережевої латентності)

Оптимізація Proxmox за допомогою цих методів дозволяє досягти стабільної продуктивності та ефективного використання ресурсів. Комбінація різних підходів дає змогу орієнтуватися на конкретні потреби сервісів у домашньому серверному середовищі. Не всі методи оптимізації є необхідними в кожному випадку, тому перед їх впровадженням варто оцінювати доцільність використання залежно від сценаріїв навантаження та вимог віртуалізованих сервісів.

3.2 Альтернативні бюджетні рішення для домашньої віртуалізації

Розвиток технологій віртуалізації відкриває нові можливості для користувачів домашніх серверів, змушуючи їх не лише обирати оптимальне програмне середовище, а й знаходити відповідне апаратне рішення. Вибір сервера залежить від його продуктивності, енергоефективності, довговічності та економічної доцільності. Тестування продуктивності віртуалізації на міні-ПК GMKtec K8 Plus продемонструвало, що подібні високопродуктивні пристрої можуть виявитися надмірними для домашнього використання, особливо якщо основні завдання обмежуються Jellyfin, домашньою автоматизацією Home Assistant та базовими контейнеризованими сервісами. Замість цього варто розглядати альтернативні бюджетні конфігурації, які краще відповідатимуть потребам та фінансовим можливостям.

Альтернативи високопродуктивним Mini PC для домашнього сервера. Під час тестування стало очевидним, що GMKtec K8 Plus має значно вищу продуктивність, ніж потрібно для стандартних домашніх серверних завдань. Його восьмиядерний Ryzen 7 8845HS з інтегрованою графікою Radeon 780M та підтримкою DDR5-пам’яті роблять його надмірним для більшості користувачів, особливо з урахуванням вартості. Повна конфігурація з 64 ГБ оперативної пам’яті DDR5 та 2 ТБ NVMe SSD коштує приблизно 33 000 грн, тоді як базова версія

обійдеться у 25 000 грн, що також є досить значною сумою для домашнього сервера.

Замість такого дорогого рішення можна розглядати більш бюджетні альтернативи, які забезпечать достатню продуктивність для віртуалізації та водночас будуть економічно виправданими. Одним із варіантів є Lenovo ThinkCentre M910q – б/в рішення, що коштує близько 5000 грн. Оснащений процесорами Intel Core i5-7500T або i7-7700T (4 ядра, 8 потоків), підтримує до 32 ГБ DDR4 та має можливість встановлення SATA або NVMe SSD, а також мережевий інтерфейс на 1 Гбіт/с. Цей варіант забезпечує достатню продуктивність для Home Assistant, Nextcloud та інших легких віртуальних машин, хоча може відчувати значне навантаження під час роботи з Jellyfin через відносно слабкий процесор у порівнянні з Ryzen 7 8845HS (див. рис. 3.7).



Рисунок 3.7 – Альтренатива для побудови сервера:
вживаний Lenovo ThinkCentre M910q

Ще одним варіантом є Beelink SER5 Pro, який коштує приблизно 10 000 грн. Оснащений сучаснішим процесором AMD Ryzen 5 5600H, підтримує до 64 ГБ DDR4 та використовує виключно NVMe SSD. Мережевий інтерфейс також має пропускну здатність 1 Гбіт/с. Порівняно з M910q, цей Mini PC забезпечує вищу продуктивність завдяки новішому процесору та енергоефективнішому техпроцесу, проте може мати проблеми з охолодженням під високим навантаженням (див. рис. 3.8).



Рисунок 3.8 – Альтренатива для побудови сервера:
Beelink SER5 Pro

Більш доступним варіантом може стати Intel NUC 8–10 покоління, який коштує від 4 000 до 8 000 грн у б/в стані. Такі пристрої зазвичай оснащені процесорами Intel Core i5 або i7 (від 8-го до 10-го покоління), підтримують до 64 ГБ DDR4 та мають можливість встановлення NVMe SSD. Компактний форм-фактор, достатня продуктивність для більшості домашніх сервісів та низьке енергоспоживання роблять їх привабливим варіантом. Недоліком є те, що вони дорожчі за M910q, але слабші за Beelink SER5 Pro (див. рис. 3.9).



Рисунок 3.9 – Альтренатива для побудови сервера:
вживаний Intel NUC 8–10 покоління

Ще одним варіантом є Mini PC на базі Celeron N100, які коштують від 4 000 грн у новому стані. Процесор Intel Celeron N100 (4 ядра, 4 потоки) підтримує до 16 ГБ DDR4, має NVMe SSD та мережевий інтерфейс 1 Гбіт/с. Ці системи дуже енергоефективні, тихі та дешеві, проте підходять лише для легких сервісів на кшталт Pi-hole, AdGuard Home чи Home Assistant, оскільки не здатні працювати з важкими віртуальними машинами (див. рис. 3.10).



Рисунок 3.10 – Альтренатива для побудови сервера:
GMKtec G3 Mini (Celeron N100)

Альтернативним підходом є збірка власного сервера на базі micro-ITX материнських плат, що може забезпечити кращу гнучкість у виборі компонентів. Наприклад, використання процесора AMD Ryzen 5 PRO 4650G у поєднанні з ASRock B550 Pro4, яка підтримує ECC-пам’ять, дозволяє створити більш надійний сервер для роботи з базами даних та резервними копіями. Встановлення 32 ГБ ECC DDR4 Kingston Server Premier забезпечує високу стабільність, а кулер Thermalright AXP90-X36 гарантує ефективне охолодження у компактному корпусі. Використання корпуса Sagittarius 8-bay або альтернативного компактного варіанту дозволяє розмістити систему у мінімальному просторі, а блок живлення Cooler Master MWE 500W забезпечить необхідну потужність для стабільної роботи. У такій збірці можливе встановлення до 8 HDD разом із NVMe SSD для операційної системи, що дозволяє створити сховище великого обсягу (див. рис. 3.11).



Рисунок 3.11 – Альтренатива для побудови сервера:
власний сервера на базі micro-ITX

Загальна вартість такої збірки буде наближеною до вартості GMKtec K8 Plus, однак вона матиме кращі можливості розширення, підтримку ECC-пам’яті та більший дисковий масив. Хоча в деяких аспектах вона поступається в продуктивності, можливість масштабування робить її привабливим рішенням для зберігання великих обсягів даних та забезпечення стабільної роботи серверних служб (див. рис. 3.12).

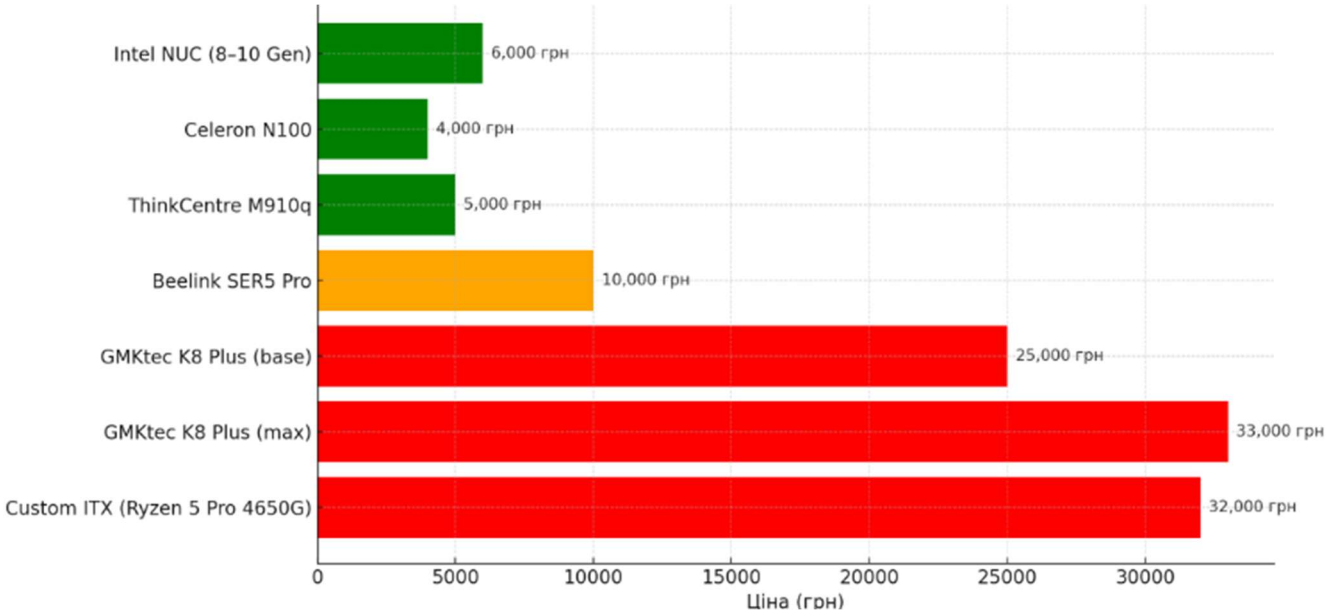


Рисунок 3.12 – Порівняння різності у вартості між бюджетними, середніми та високопродуктивними рішеннями для домашнього сервера

3.3 Перспективи розвитку віртуалізації домашніх серверів

Дослідження використання альтернативних серверів показало, що обрана конфігурація на базі Mini PC GMKtec K8 Plus є надмірною для більшості домашніх задач. Значна кількість ресурсів, зокрема обчислювальних потужностей та оперативної пам’яті, не використовувалася повною мірою, що робить таке рішення не завжди економічно виправданим. Оптимізація продуктивності та

енергоефективності домашніх серверів потребує переосмислення підходів до вибору обладнання та архітектури системи.

Оптимізація домашніх серверів через кластеризацію. Одним із перспективних напрямів є кластеризація, яка передбачає використання кількох бюджетних серверів замість одного потужного. Об'єднання декількох пристроїв у кластер дозволяє підвищити відмовостійкість, забезпечити баланс навантаження та уникнути простоїв критичних сервісів. Наприклад, вихід з ладу одного вузла не призведе до відключення всієї системи, а технічне обслуговування може проводитись без зупинки основних сервісів.

Кластеризацію можна реалізувати на базі кількох доступних пристроїв, таких як Lenovo ThinkCentre M910Q, Beelink SER5 Pro або Intel NUC. Вони коштують дешевше, ніж використовуваний Mini PC, і дозволяють більш гнучко масштабувати ресурси. Такий підхід дозволяє уникнути ситуації, коли потужності одного пристрою залишаються невикористаними.

Мінімізація витрат на електроенергію. Енергоефективність є ще одним важливим аспектом експлуатації серверів, що працюють у режимі 24/7. Зменшення витрат на електроенергію досягається шляхом:

— використання процесорів із низьким TDP (наприклад, Intel i5 або Ryzen 5);
— застосування енергозберігаючих технологій (Intel SpeedStep, AMD Cool'n'Quiet, C/P-States);
— переходу на SSD або NVMe-накопичувачі замість HDD;
— вимкнення непотрібних портів і пристроїв у BIOS/UEFI;
— впровадження контейнеризації (LXC, Docker) замість повноцінних віртуальних машин, де це можливо.

Накопичувачі також впливають на енергоспоживання. SSD є більш енергоефективними, ніж HDD, тому використання NVMe-дисків для гіпервізора та системи, а HDD — для зберігання резервних копій у RAID-масиві, дозволяє досягти балансу між продуктивністю та відмовостійкістю.

Розширення сервера без зайвих витрат. Масштабування ресурсів домашнього сервера має відбуватися поступово, у відповідь на реальні потреби. Збільшення обсягу оперативної пам’яті або підключення віддалених NAS-сховищ (на базі TrueNAS, Synology) дозволяє уникнути перевантаження основного вузла. Крім того, контейнеризація замість повноцінної віртуалізації в окремих задачах дозволяє раціонально використовувати ресурси та знижувати навантаження.

Одним із найефективніших рішень для масштабування та забезпечення гнучкості є кластер на базі Proxmox VE. Додавання навіть одного додаткового вузла дозволяє розподілити навантаження, підвищити надійність та спростити адміністрування. У довгостроковій перспективі це ефективніше, ніж експлуатація потужного, але одного Mini PC, який споживає більше енергії та має обмеження в масштабуванні.

Подальші дослідження у цій сфері можуть бути спрямовані на:

— тестування кластерних конфігурацій на базі бюджетних Mini PC;
— аналіз енергоспоживання різних платформ у реальних сценаріях;
— порівняння ефективності контейнеризації проти повноцінної віртуалізації;
— автоматизацію розподілу навантаження в кластерному середовищі.

Така ініціатива дозволить не лише підвищити ефективність і надійність домашніх серверів, а й зробити їх більш доступними для широкого кола користувачів.

ВИСНОВКИ

Використання гіпервізора 1-го типу Proxmox VE дало змогу суттєво підвищити ефективність роботи домашнього сервера завдяки впровадженню цілеспрямованих методів оптимізації. У ході практичного дослідження було реалізовано автоматичне очищення кешу в Linux для контейнера Jellyfin, що вирішило проблему надмірного використання оперативної пам’яті та підвищило стабільність системи. Також було проведено переналаштування мережевих портів і вимкнення конфліктного DNS-сервера в Ubuntu Server, що дозволило забезпечити коректну роботу DNS-сервісу AdGuard Home.

Крім цього, проаналізовано перспективу впровадження SR-IOV для зниження мережевої латентності в OPNsense, що особливо актуально для VPN-серверів та інших чутливих до затримок сервісів. Задіяні рішення забезпечили не лише підвищення швидкодії, а й кращу керованість ресурсами віртуального середовища.

Порівняльне тестування показало, що один потужний Mini PC поступається за ефективністю кластеру з кількох бюджетних пристроїв. Кластерне рішення забезпечує рівномірний розподіл навантаження між вузлами, підвищує відмовостійкість і дозволяє обслуговувати критичні сервіси без простоїв, навіть під час оновлень або технічного обслуговування.

Завдяки використанню контейнеризації, профілів живлення процесора та розподіленої архітектури вдалося оптимізувати енергоспоживання. Легкоресурсні сервіси, зокрема Jellyfin, ефективно використовували ресурси кластера, демонструючи помірне навантаження на vCPU, мережу, RAM і SSD, що підтверджує доцільність використання бюджетної кластерної архітектури для стабільної та енергоефективної роботи домашніх серверів.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IBM. What is virtualization? | IBM. IBM - United States. URL: https://www.ibm.com/think/topics/virtualization (дата звернення: 25.03.2025).
2. Bigelow S. J. Full virtualization vs. paravirtualization: Key differences | TechTarget. Search IT Operations. URL: https://www.techtarget.com/searchitoperations/tip/Full-virtualization-vs-paravirtualization-Key-differences (дата звернення: 27.03.2025).
3. Containerization vs. virtualization: key differences and use cases. Aqua. URL: https://www.aquasec.com/cloud-native-academy/docker-container/containerization-vs-virtualization/ (дата звернення: 07.04.2025).
4. History of virtualization - inventive HQ. Inventive HQ. URL: https://inventivehq.com/history-of-virtualization/ (дата звернення: 27.03.2025).
5. Dillenburg S. A brief history of container virtualization. Medium. URL: https://medium.com/an-idea/a-brief-history-of-container-virtualization-57fc96c02924 (дата звернення: 28.03.2025).
6. Saafan A. Why a home server is a must-have for every tech enthusiast. DEV Community. URL: https://dev.to/nilebits/why-a-home-server-is-a-must-have-for-every-tech-enthusiast-49nf (дата звернення: 01.04.2025).
7. 7 reasons why you need a home server in 2025 - Surfshark. Surfshark. URL: https://surfshark.com/blog/home-server (дата звернення: 03.04.2025).
8. Maximize efficiency with server hardware virtualization. Scale Computing. URL: https://www.scalecomputing.com/resources/server-hardware-virtualization-maximizing-resource-utilization-and-flexibility (дата звернення: 03.04.2025).
9. Virtual server hardware requirements. Server Mall. URL: https://servermall.com/blog/how-to-choose-a-server-for-virtualization/?srsltid=AfmBOoqzO_5Ey5kLrgGsdVgZQruodvs00tQlxO81RZoB_2IzrAen_Rlq (дата звернення: 05.04.2025).
10. Most popular server virtualization platforms | hostomize. hostomize. URL: https://hostomize.com/blog/server-virtualization-popular-platforms/ (дата звернення: 07.04.2025).
11. Home lab for beginners: how to start with virtualization and networking - hostbor - tech reviews, home labs & AI computing guide. Hostbor - Tech Reviews, Home Labs & AI Computing Guide. URL: https://hostbor.com/home-lab-for-beginners/ (дата звернення: 08.04.2025).
12. Castagna R. Top 10 disadvantages of server virtualization | TechTarget. Search IT Operations. URL: https://www.techtarget.com/searchitoperations/feature/Top-10-disadvantages-of-server-virtualization (дата звернення: 10.04.2025).
13. How server virtualization works - calsoft blog. Calsoft Blog. URL: https://www.calsoftinc.com/blogs/how-server-virtualization-works.html (дата звернення: 12.04.2025).

14. Network Monitoring Software by ManageEngine OpManager. ManageEngine OpManager. URL: https://www.manageengine.com/network-monitoring/virtualization-challenges.html (дата звернення: 15.04.2025).

15. Features. Proxmox. URL: https://www.proxmox.com/en/products/proxmox-virtual-environment/features (дата звернення: 17.04.2025).

16. 10 Proxmox tools that can transform your entire experience. XDA-Developers. URL: https://www.xda-developers.com/proxmox-tools-that-can-transform-your-entire-experience/ (дата звернення: 17.04.2025).

17. Documentation. Home Assistant. URL: https://www.home-assistant.io/docs/ (дата звернення: 19.04.2025).

18. Integrations. Home Assistant. URL: https://www.home-assistant.io/integrations/?brands=featured (дата звернення: 21.04.2025).

19. OPNSense vs pfsense: which one is better in 2025?. StationX. URL: https://www.stationx.net/opnsense-vs pfsense/#:~:text=If%20you%20want%20high%20customizability,out-of-box%20experience. (дата звернення: 23.04.2025).

20. OPNsense documentation. OPNsense. URL: https://docs.opnsense.org/index.html (дата звернення: 07.04.2025).

21. Introduction | jellyfin. The Free Software Media System | Jellyfin. URL: https://jellyfin.org/docs/ (дата звернення: 24.04.2025).

22. Logvinov S. Proxmox virtual machine optimization. DEV Community. URL: https://dev.to/sergelogvinov/proxmox-virtual-machine-optimization-7mn (дата звернення: 27.04.2025).

23. TechTarget. Limitations of server virtualization | TechTarget. Search IT Channel. URL: https://www.techtarget.com/searchitchannel/feature/Limitations-of-server-virtualization (дата звернення: 30.04.2025).

24. Network virtualization in future home environments. ResearchGate. URL: https://www.researchgate.net/publication/220926794_Network_Virtualization_in_Future_Home_Environments (дата звернення: 01.05.2025).

25. Overview of pi-hole - pi-hole documentation. Pi-hole. URL: https://docs.pi-hole.net/ (дата звернення: 03.05.2025).

26. Documentation for admins | AdGuard Knowledge Base. GitHub. URL: https://adguard.com/kb/adguard-for-windows/admins-documentation/ (дата звернення: 05.05.2025).

27. Advantages of ECC memory. Puget Systems. URL: https://www.pugetsystems.com/labs/articles/advantages-of-ecc-memory-520/ (дата звернення: 05.05.2025).

28. Потапенко В.С. Дослідження продуктивності домашнього сервера при використанні віртуалізації. «Проблеми комп'ютерної інженерії»: Науково-практ. конф., м.Київ, 3 грудня 2024 р. С.59-62.

29. Потапенко В.С., Бабенко В.Г. Розгортання домашнього сервера з використанням віртуалізації. «Сучасний стан та перспективи розвитку IoT» : VI Науково-техн. конф., м.Київ, 15 квітня 2025 р. (подано до публікації).

Додаток А

Навантаження на процесор, ОЗП, диск, мережу та сервер

Таблиця А.1 – Максимальне навантаження на процесор (CPU) (у % співвідношенні)

№	Сервер/Віртуальна машина	CPU (%)	Відносно всіх ядер (100%)
1	Вузол Proxmox	9.63%	9.63%
2	Home Assistant OS	7.06%	7.06%
3	Ubuntu Desktop	32.65%	32.65%
4	OPNsense	2.11%	2.11%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	29.98%	29.98%

Таблиця А.2 – Середнє навантаження на процесор (CPU) (у % співвідношенні)

№	Сервер/Віртуальна машина	CPU (%)	Відносно всіх ядер (100%)
1	Вузол Proxmox	3.46%	3.46%
2	Home Assistant OS	6.16%	6.16%
3	Ubuntu Desktop	6.00%	6.00%
4	OPNsense	1.75%	1.75%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	8.42%	8.42%

Таблиця А.3 – Мінімальне навантаження на процесор (CPU) (у % співвідношенні)

№	Сервер/Віртуальна машина	CPU (%)	Відносно всіх ядер (100%)
1	Вузол Proxmox	1.66%	1.66%
2	Home Assistant OS	3.66%	3.66%
3	Ubuntu Desktop	2.76%	2.76%
4	OPNsense	0.8%	0.8%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	0.68%	0.68%

Продовження додатка А

Навантаження на процесор, ОЗП, диск, мережу та сервер

Таблиця А.4 – Максимальне навантаження (читання) на диск (у % співвідношенні)

№	Віртуальна машина	Читання (МБ/с)	Відносно SSD (100%)
1	Home Assistant OS	14.64 МБ/с	14.64%
2	Ubuntu Desktop	6.59 МБ/с	6.59%
3	OPNsense	298.3 КБ/с	0.29%
4	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	121.22 МБ/с	121.22%

Таблиця А.5 – Максимальне навантаження (запис) на диск (у % співвідношенні)

№	Віртуальна машина	Читання (МБ/с)	Відносно SSD (100%)
1	Home Assistant OS	1.45 МБ/с	1.45%
2	Ubuntu Desktop	578.94 КБ/с	0.58%
3	OPNsense	44.95 КБ/с	0.04%
4	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	89.48 МБ/с	89.48%

Таблиця А.6 – Максимальне навантаження на мережу (у % співвідношенні)

№	Сервер/Віртуальна машина	Швидкість (Мбіт/с)	Відносно 1 Gbps (100%)
1	Вузол Proxmox	6.8 Мбіт/с	0.68%
2	Home Assistant OS	122.14 Кбіт/с	0.012%
3	Ubuntu Desktop	281.67 Кбіт/с	0.028%
4	OPNsense	333.79 Кбіт/с	0.033%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	6.37 Мбіт/с	0.637%

Продовження додатка А

Навантаження на процесор, ОЗП, диск, мережу та сервер

Таблиця А.7 – Середнє навантаження на мережу (у % співвідношенні)

№	Сервер/Віртуальна машина	Швидкість (Мбіт/с)	Відносно 1 Gbps (100%)
1	Вузол Proxmox	1.38 Мбіт/с	0.138%
2	Home Assistant OS	120.27 Кбіт/с	0.012%
3	Ubuntu Desktop	154.43 Кбіт/с	0.015%
4	OPNsense	80.28 Кбіт/с	0.008%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	1.05 Мбіт/с	0.105%

Таблиця А.8 – Навантаження на сервер (системні ресурси, у % співвідношенні)

№	Параметр	Значення	Відносно 100% ресурсу CPU
1	Навантаження (макс.)	0.82	82% одного ядра
2	Навантаження (середнє)	0.49	49% одного ядра
3	Навантаження (мін.)	0.21	21% одного ядра

Додаток Б

Використання оперативної пам’яті

Таблиця Б.1 – Максимальне використання оперативної пам’яті (RAM) (у % співвідношенні)

№	Сервер/Віртуальна машина	Використання ОЗП (ГБ)	Відносно 32 ГБ (%)
1	Вузол Proxmox	20.79 ГБ	65.0%
2	Home Assistant OS	2.33 ГБ	7.28%
3	Ubuntu Desktop	2.94 ГБ	9.19%
4	OPNsense	1.51 ГБ	4.72%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	14.01 ГБ	43.78%

Таблиця Б.2 – Середнє використання оперативної пам’яті (RAM) (у % співвідношенні)

№	Сервер/Віртуальна машина	Використання ОЗП (ГБ)	Відносно 32 ГБ (%)
1	Вузол Proxmox	7.66 ГБ	23.94%
2	Home Assistant OS	2.00 ГБ	6.25%
3	Ubuntu Desktop	2.66 ГБ	8.31%
4	OPNsense	1.50 ГБ	4.69%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	1.50 ГБ	4.69%

Таблиця Б.3 – Мінімальне використання оперативної пам’яті (RAM) (у % співвідношенні)

№	Сервер/Віртуальна машина	Використання ОЗП (ГБ)	Відносно 32 ГБ (%)
1	Вузол Proxmox	5.45 ГБ	17.03%
2	Home Assistant OS	770.85 МБ	2.41%
3	Ubuntu Desktop	2.33 ГБ	7.28%
4	OPNsense	1.34 ГБ	4.19%
5	Ubuntu Server (Portainer – Adguard Home – Jellyfin)	1.01 ГБ	3.16%

Додаток В

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерної інженерії



КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

НА ТЕМУ: «ДОМАШНІЙ СЕРВЕР З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ВІРТУАЛІЗАЦІЇ»

Виконав студент групи КІД-42
Потапенко Владислав Сергійович
Керівник дипломної роботи:
Бабенко Віра Григорівна,
д-р техн. наук, проф.

Київ – 2025

Мета, об’єкт, предмет дослідження

Мета роботи – оптимізація продуктивності та ефективності використання ресурсів домашнього сервера за рахунок впровадження та налаштування технологій віртуалізації

Об’єкт дослідження – процес віртуалізації в домашньому серверному середовищі

Предмет дослідження – методи та технології віртуалізації, що впливають на продуктивність, стабільність і енергоефективність домашнього сервера

Актуальність роботи

У роботі зосереджено увагу на оптимізації продуктивності домашніх серверів за допомогою віртуалізації. Аналізується вплив різних підходів до розподілу ресурсів, використання кешування, балансування навантаження та кластеризації. Особлива увага приділяється оцінці продуктивності при роботі з потоковим медіа, файловими сховищами, DNS-сервісами, а також впливу вибору обладнання на загальну стабільність системи.

Основна мета – оцінити ефективність віртуалізації в контексті домашнього серверного використання, визначити потенційні вузькі місця та запропонувати методи їх оптимізації, що допоможе користувачам розробити надійні та енергоефективні серверні рішення, що відповідатимуть потребам у домашньому сегменті.

Труднощі при побудові традиційного та віртуалізованого сервера

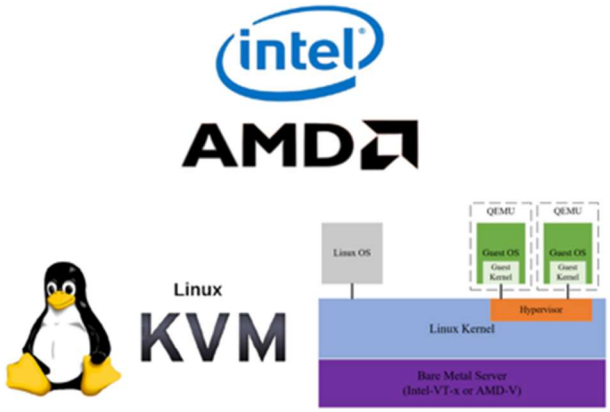


Рис. 1.12 – Підтримка віртуалізації на апаратному рівні та створення гіпервізора KVM: Intel VT-x та AMD-V

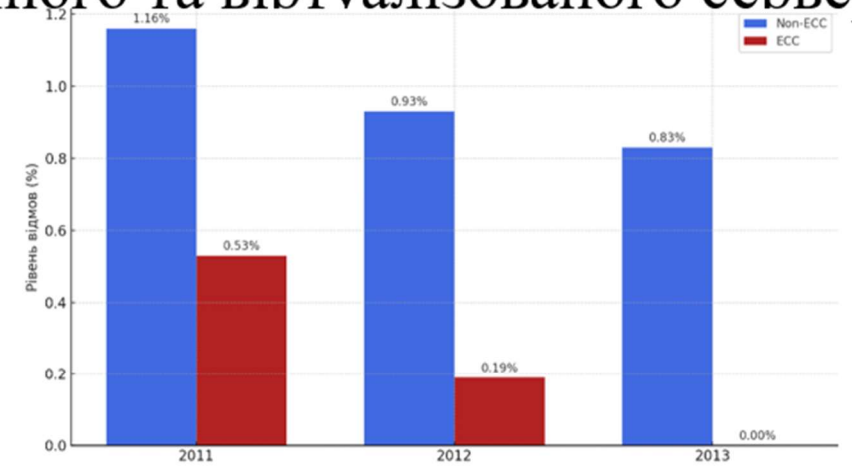


Рис. 1.32 – Рівень відмов пам'яті Kingston ECC та Non-ECC [27]

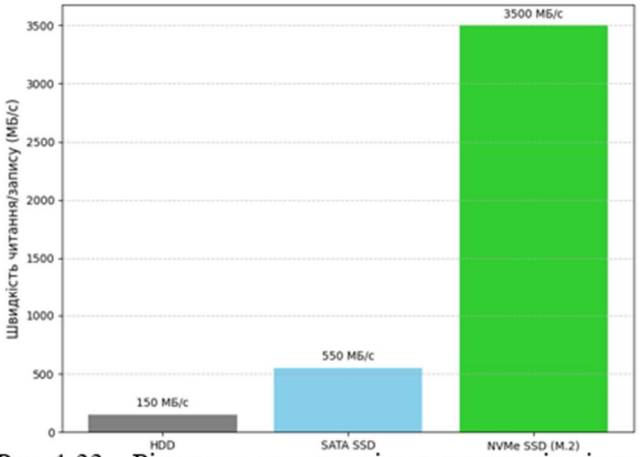


Рис. 1.33 – Різниця у швидкості накопичувачів різних інтерфейсів: HDD, SATA SSD, NVMe SSD (M.2)

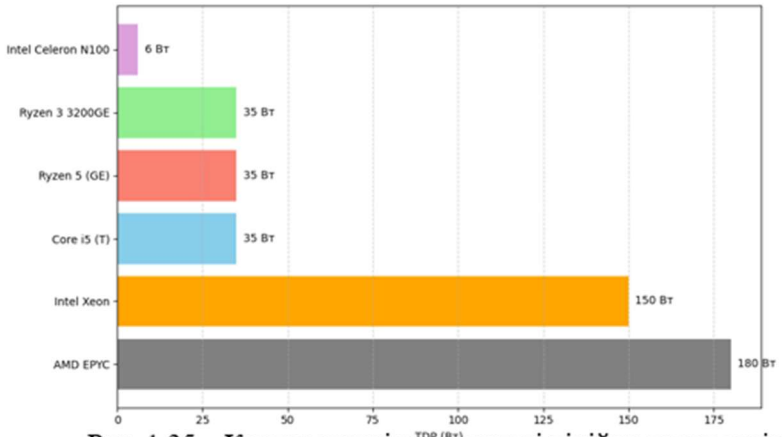


Рис. 1.35 – Корпоративні та споживчі лінійки процесорів: вплив потужності на енергоспоживання у TDP (Вт)

Архітектура домашнього сервера з використанням технологій віртуалізації

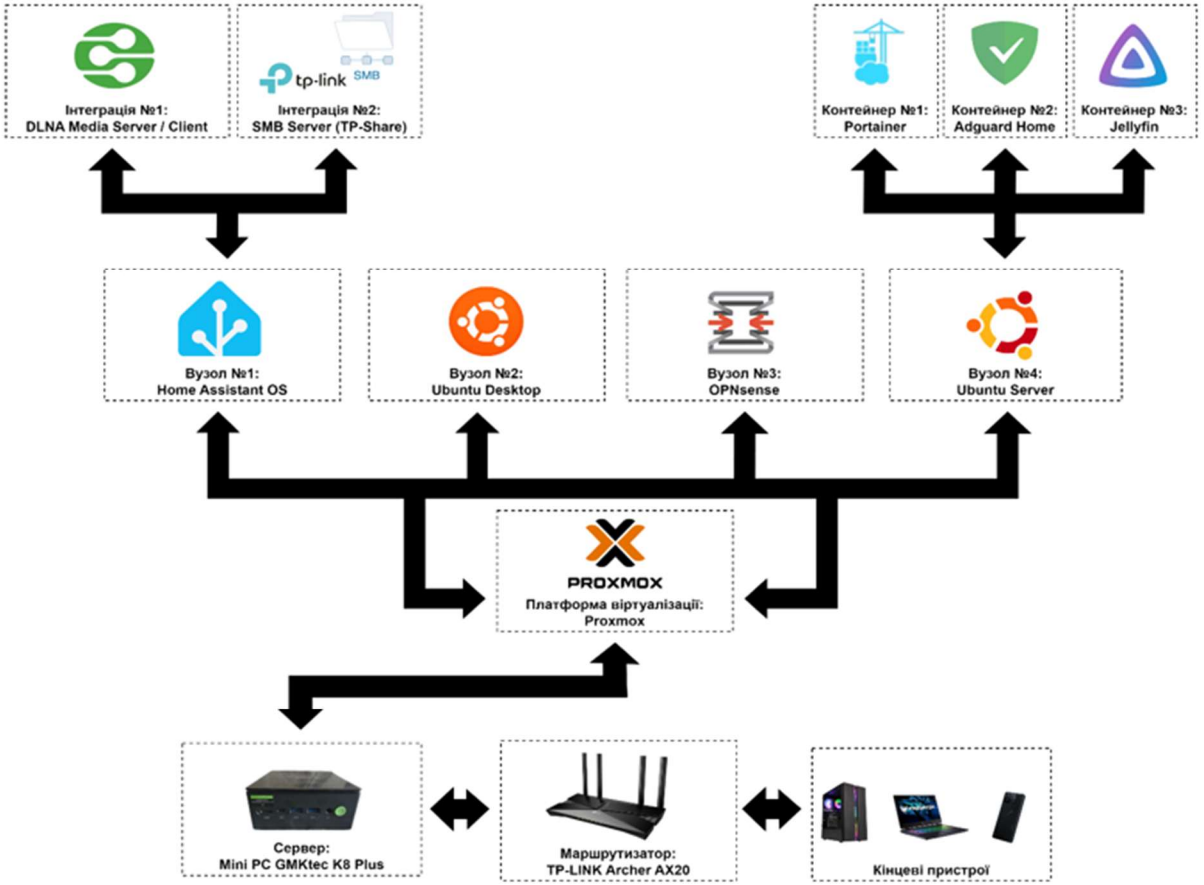
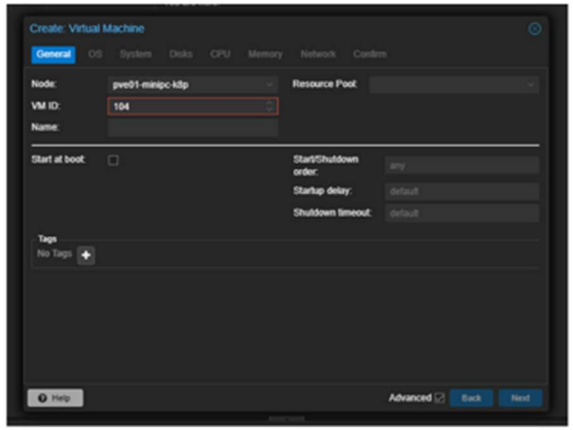


Рис. 2.11 – Архітектура домашнього сервера на основі віртуалізації

Вибір платформи віртуалізації: переваги Proxmox VE



Рис. 2.9 – Вбудований системний моніторинг над ресурсами: динамічні графіки навантаження CPU, RAM, мережі та сервера



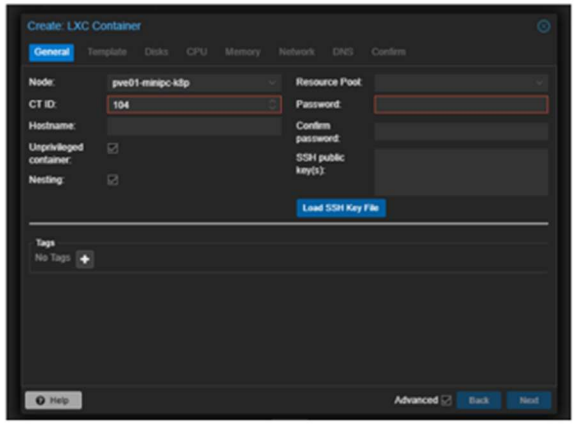


Рис. 2.8 – Можливість створювати повноцінну віртуальну машину (VM) та Linux-контейнер (LXC)

Вибір платформи віртуалізації: переваги Proxmox VE

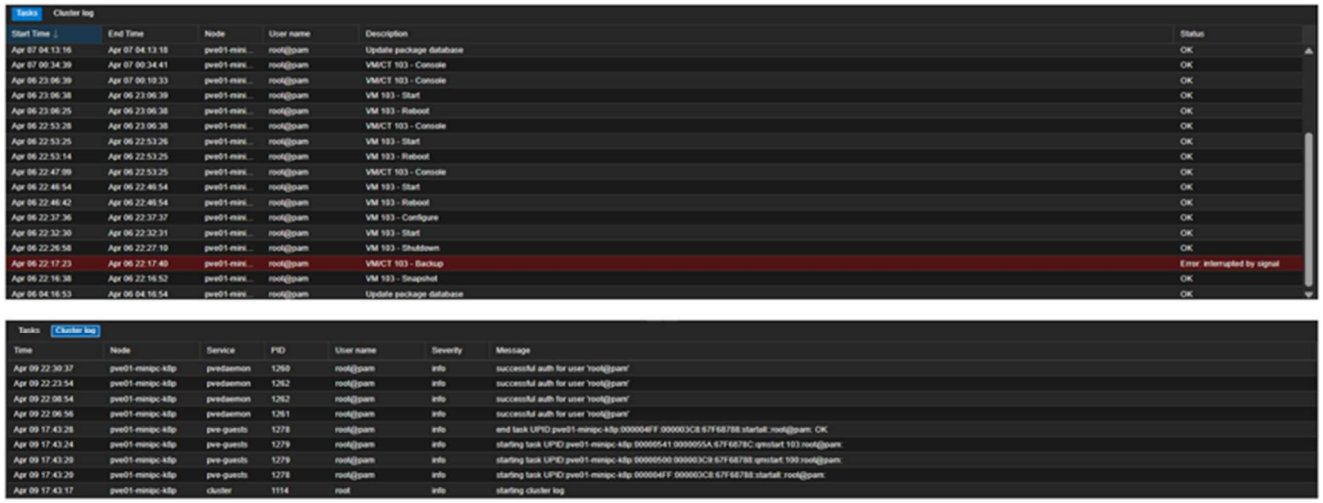


Рис. 2.10 – Перегляд системного журналу: "Tasks" та "Cluster log"

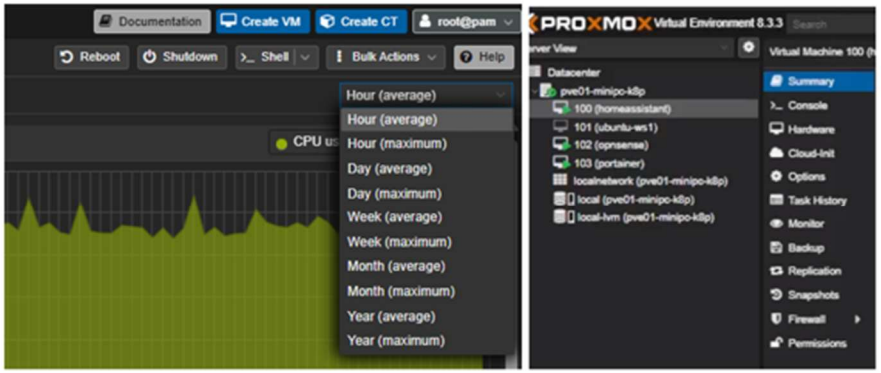


Рис. 2.40 – Перегляд статистики серед усіх вузлів по навантаженю: година, день, тиждень, рік (мінімум, середнє, максимум)

Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: сервер розумного дому

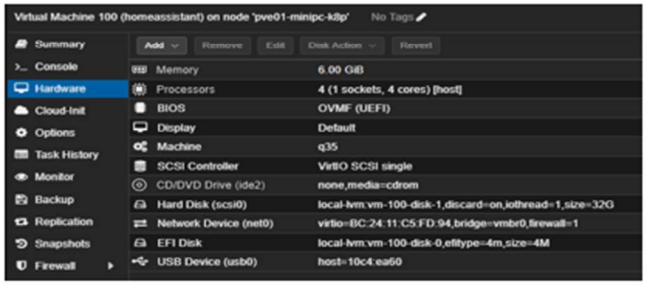
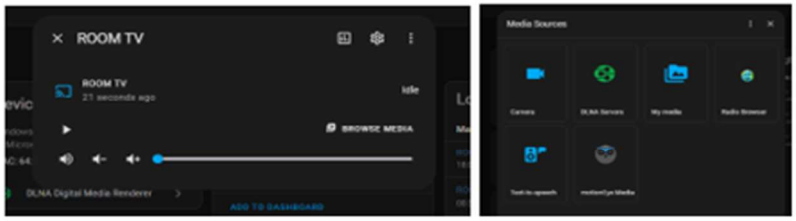


Рис. 2.12 – Налаштована конфігурація вузла: сервер розумного дому



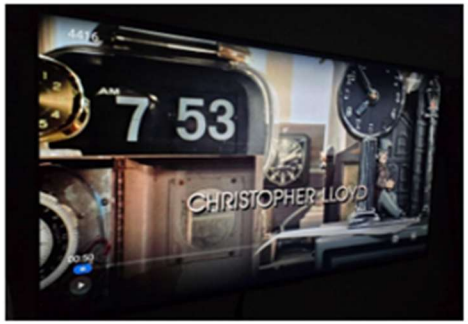


Рис. 2.19 – Передача медіаконтенту на інший пристрій через медіасервер DLNA



Рис. 2.20 – Метод оцінки використання віртуалізованого домашнього сервера у якості сервера розумного дому

Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: віртуальна машина для тестування

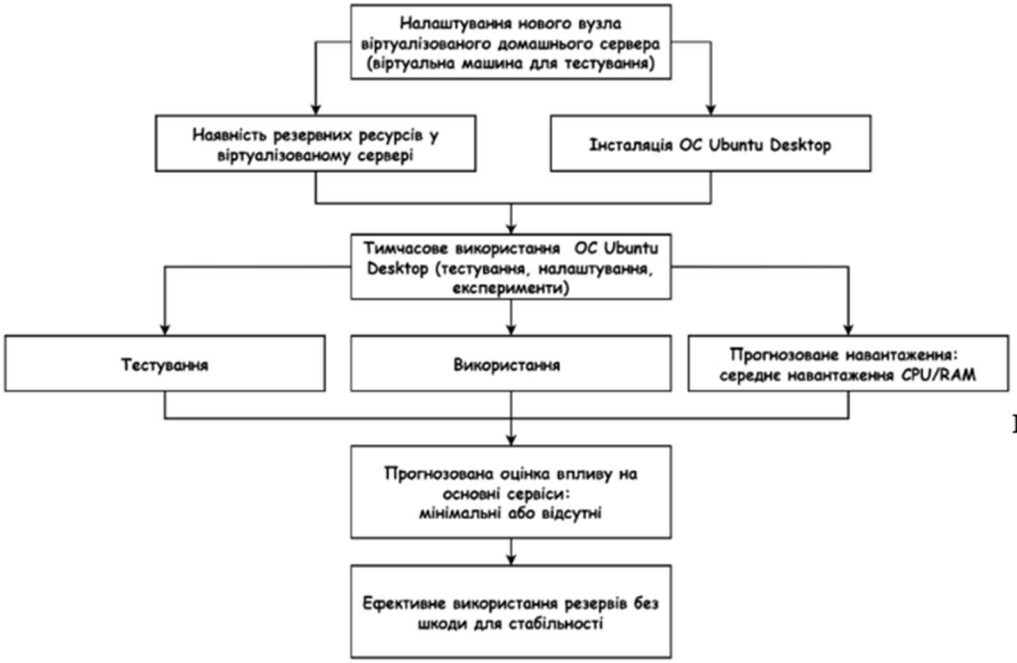


Рис. 2.23 – Метод оцінки використання віртуалізованого домашнього сервера у якості віртуальної машини для тестування

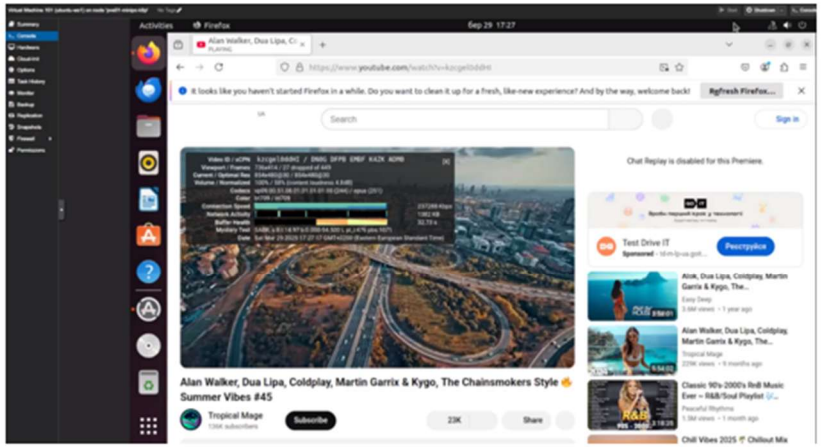


Рис. 2.22 – Перегляд відеоролика на YouTube як навантаження на вузол

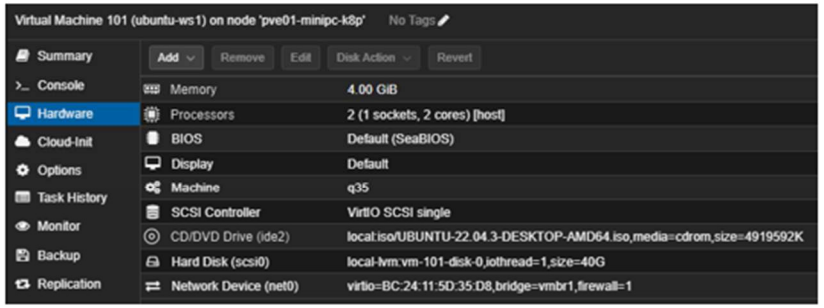


Рис. 2.15 – Налаштована конфігурація вузла: віртуальна машина для тестування

Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: кастомізований маршрутизатор

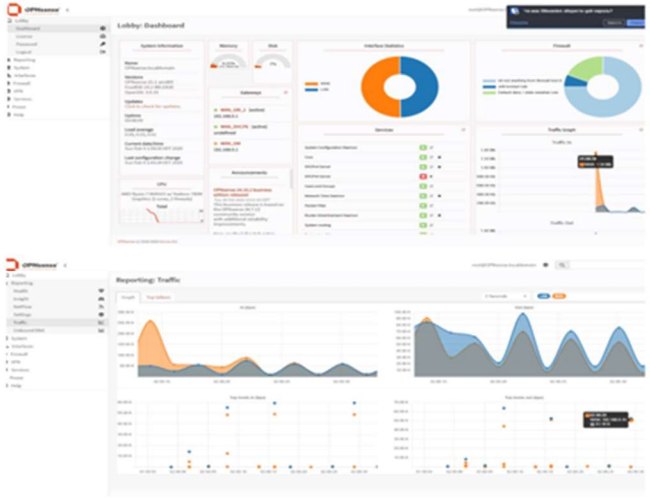


Рис. 2.24 – Робота віртуальної машини OPNsense у якості відслідковування та фільтрації небажаного трафіку

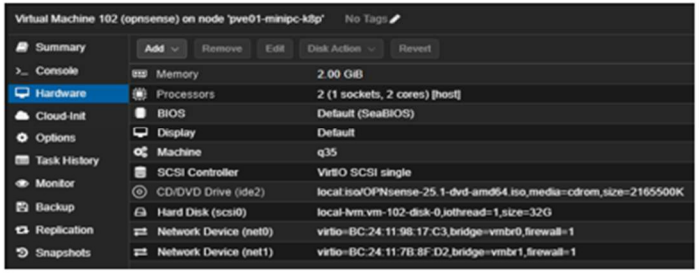


Рис. 2.14 – Налаштована конфігурація вузла: кастомізований маршрутизатор зі фаєрволом

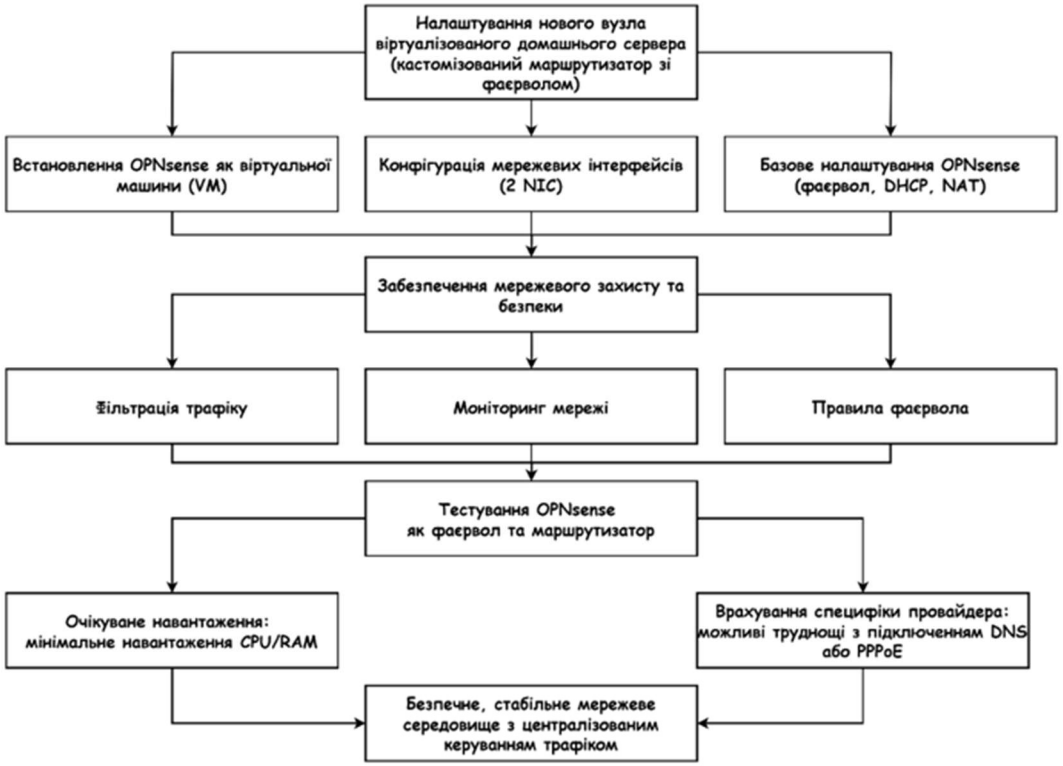


Рис. 2.25 – Метод оцінки використання віртуалізованого домашнього сервера у якості кастомізованого маршрутизатора зі фаєрволом

Конфігурація робочих вузлів та сценарії навантаження у віртуалізованому середовищі: DNS- та медіа- серверів

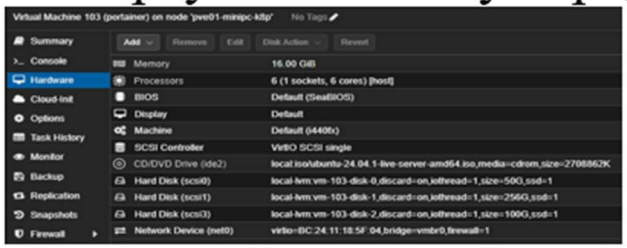


Рис. 2.13 – Налаштована конфігурація вузла: DNS- та медіа- сервери



Рис. 2.26 – Метод оцінки використання віртуалізованого домашнього сервера у якості DNS-та медіа- серверів



Рис. 2.27 – Робота Docker контейнера Jellyfin у якості медіа-сервера (зі транскодуванням)

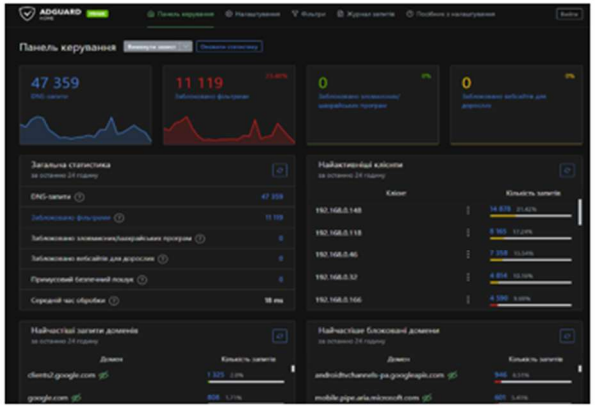


Рис. 2.28 – Робота Docker контейнера Adguard Home у якості DNS-сервера (блокувальника реклами та фішингових сайтів)

Спостереження за ресурсоспоживанням всіх вузлів

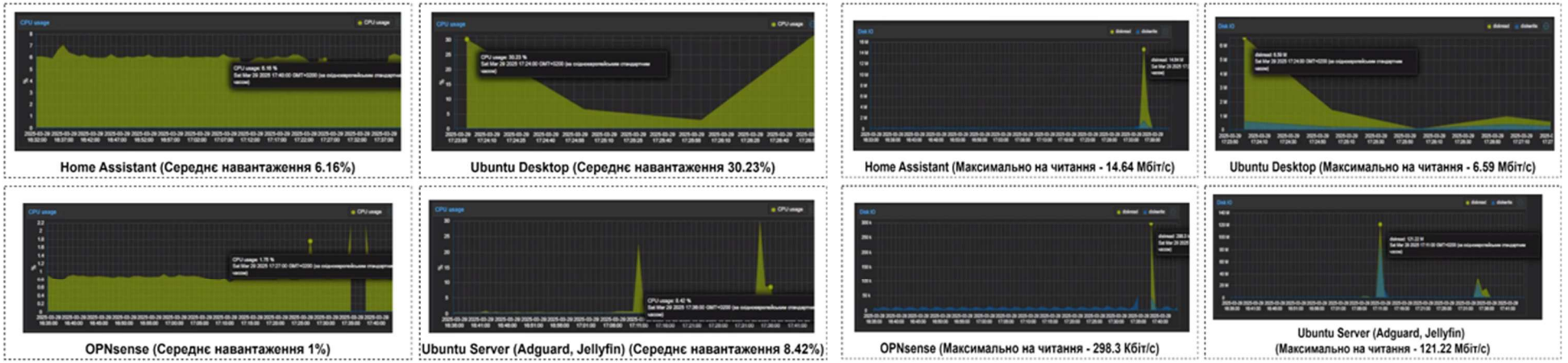


Рис. 2.34 – Середнє навантання на CPU окремими вузлами

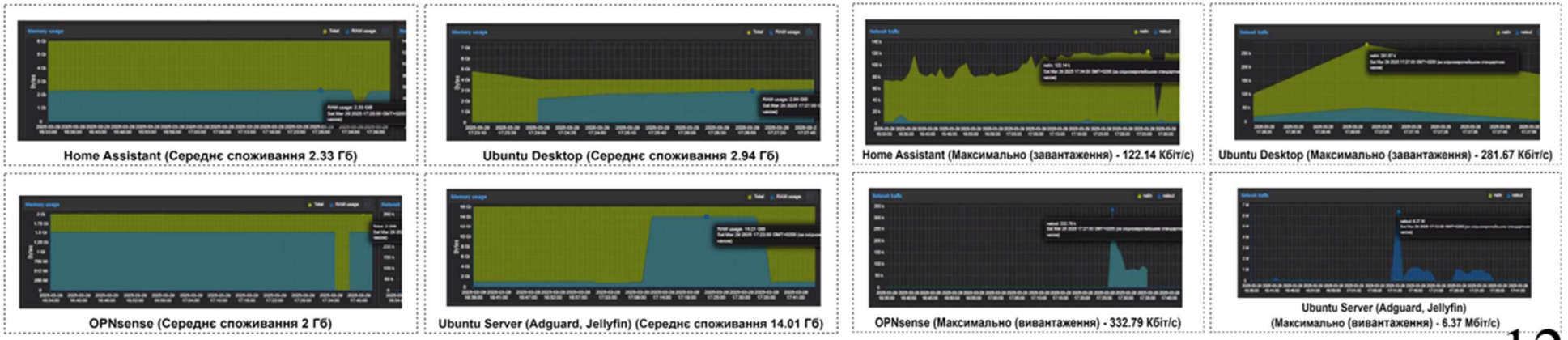


Рис. 2.35 – Середнє навантання на RAM окремими вузлами

Рис. 2.36 – Максимальне навантання (читання) на SSD окремими вузлами

Рис. 2.38 – Максимальне навантання на мережу (завантаження/вивантаження) окремими вузлами

Спостереження за ресурсоспоживанням всіх вузлів

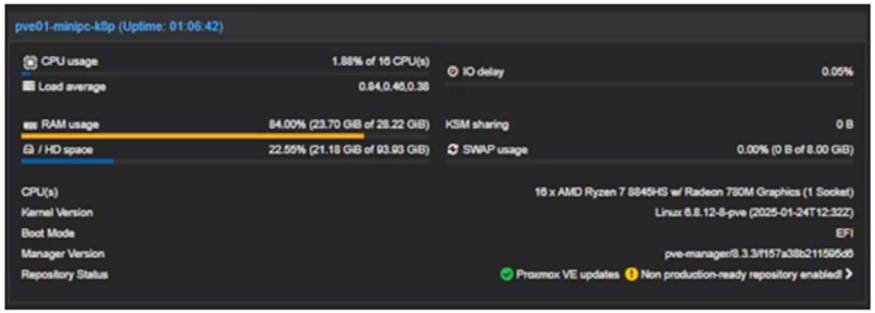


Рис. 2.32 – Загальне максимальне навантаження на сервер



Рис. 2.31 – Навантаження на кожен вузол системи

Табл. 2.1 – Загальні показники продуктивності серверного вузла та віртуальних маши

№	Сервер/Віртуальна машина	Вузол Proxmox	Home Assistant OS	Ubuntu Desktop	OPNsense	Ubuntu Server (Portainer – Adguard Home – Jellyfin)
1	Навантаження на Процесор (максимум)	**9.63 %**	7.06 %	32.65 %	2.11 %	29.98 %
2	Навантаження на Процесор (середнє)	**3.46 %**	6.16 %	6 %	1.75 %	8.42 %
3	Навантаження на Процесор (мінімум)	**1.66 %**	3.66 %	2.76 %	0.8 %	0.68 %
4	Використання ОЗП (доступно)	**28.22 Гб**	6 Гб	4 Гб	2 Гб	16 Гб
5	Використання ОЗП (максимум)	**20.79 Гб**	2.33 Гб	2.94 Гб	1.51 Гб	14.01 Гб
6	Використання ОЗП (середнє)	**7.66 Гб**	2 Гб	2.66 Гб	1.50 Гб	1.50 Гб
7	Використання ОЗП (мінімум)	**5.45 Гб**	770.85 Мб	2.33 Гб	1.34 Гб	1.01 Гб
8	Навантаження на Мережу (максимум)	**6,8 Мбіт/с**	122.14 Кбіт/с	281.67 Кбіт/с	333.79 Кбіт/с	6.37 Мбіт/с
9	Навантаження на Мережу (середнє)	**1.38 Мбіт/с**	120.27 Кбіт/с	154.43 Кбіт/с	80.28 Кбіт/с	1.05 Мбіт/с
10	Навантаження на Мережу (мінімум)	**12.98 Кбіт/с**	10.6 Кбіт/с	812,65	759.65	818.68
11	Навантаження на Диск (читання - максимум)	-	14.64 Мбіт/с	6.59 Мбіт/с	298.3 Кбіт/с	121.22 Мбіт/с
12	Навантаження на Диск (запис - максимум)	-	1.45 Мбіт/с	578.94 Кбіт/с	44.95 Кбіт/с	89.48 Мбіт/с
13	Навантаження на Диск (читання - мінімум)	-	0	81,59 Кбіт/с	273.07 Кбіт/с	33.33 Мбіт/с
14	Навантаження на Диск (запис - мінімум)	-	138.48 Кбіт/с	81,59 Кбіт/с	12.83 Кбіт/с	24.74 Мбіт/с
15	Навантаження на Сервер (максимум)	**0.82**	-	-	-	-
16	Навантаження на Сервер (середнє)	**0.49**	-	-	-	-
17	Навантаження на Сервер (мінімум)	**0.21**	-	-	-	-

Методи оптимізації: автоматичне очищення кешу у Linux та переналаштування портів

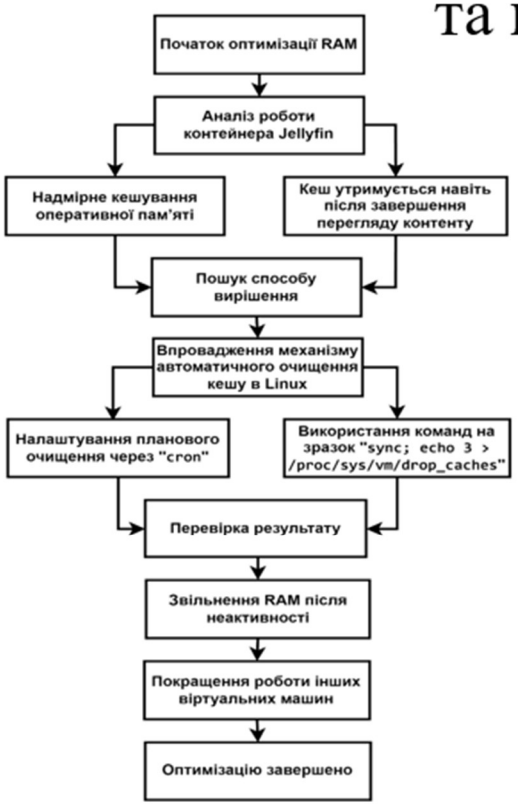


Рис. 3.1 – Метод оптимізації навантаження на сервер шляхом впровадження автоматичного очищення кешу в Linux

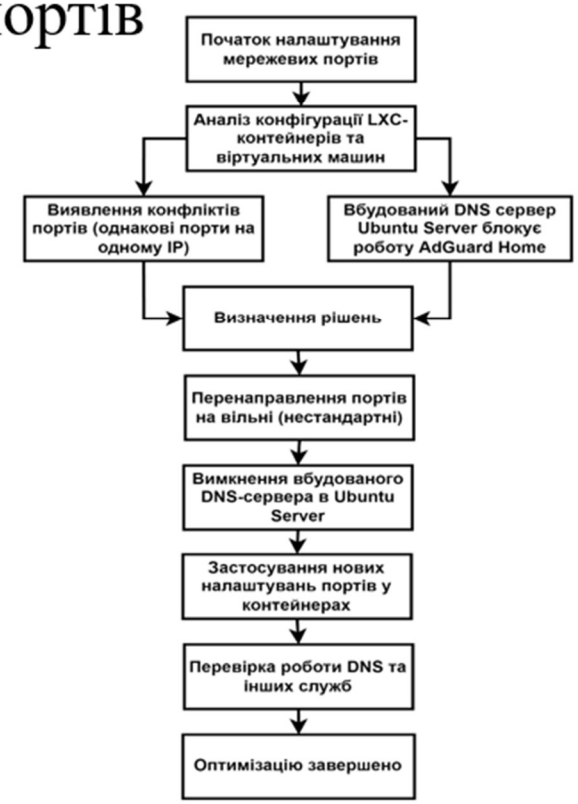


Рис. 3.2 – Метод оптимізації навантаження на сервер шляхом усунення конфліктів портів між контейнерами та віртуальними машинами

Кластеризація з кількох бюджетних пристроїв: переваги над одним потужним сервером



Рис. 3.7 – Альтренатива для побудови сервера: вживаний Lenovo ThinkCentre M910q



Рис. 3.8 – Альтренатива для побудови сервера: Beelink SER5 Pro



Рис. 3.9 – Альтренатива для побудови сервера: вживаний Intel NUC 8–10 покоління



Рис. 3.10 – Альтренатива для побудови сервера: GMKtec G3 Mini (Celeron N100)



Рис. 3.11 – Альтренатива для побудови сервера: власний сервера на базі micro-ITX

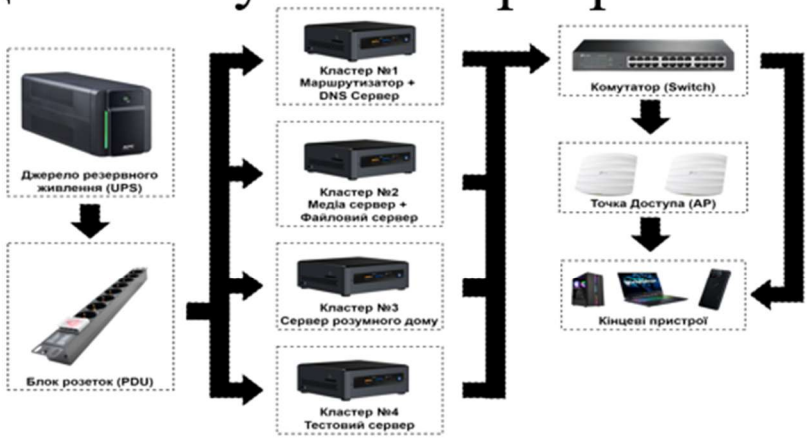


Рис. 2.6 – Архітектура кластерної відмовостійкої системи

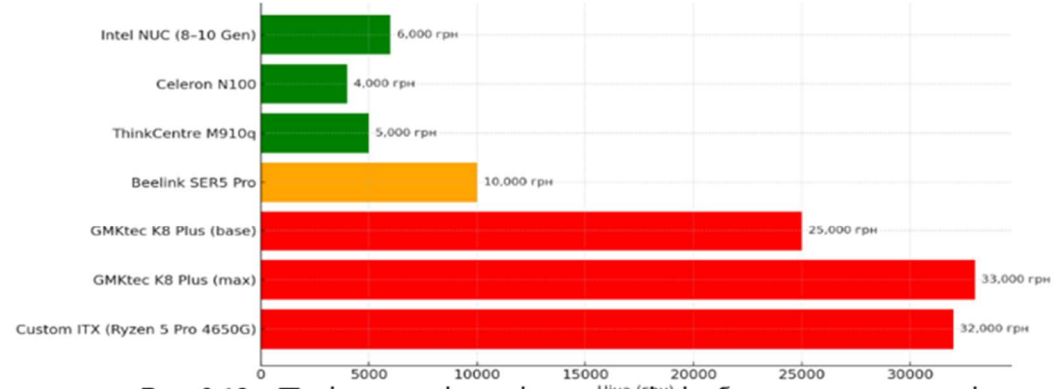


Рис. 3.12 – Порівняння різності у вартості між бюджетними, середніми та високопродуктивними рішеннями для домашнього сервера

Висновки

Використання Proxmox VE як гіпервізора 1-го типу дозволило значно підвищити ефективність роботи домашнього сервера завдяки впровадженню автоматичного очищення кешу в Linux для Jellyfin та усуненню конфліктів портів і DNS-сервісів між контейнерами, що забезпечило стабільну роботу мультимедійних і мережевих сервісів, зменшило споживання оперативної пам’яті та покращило керованість системою.

Порівняльне тестування показало переваги кластеру з кількох бюджетних пристроїв над одним потужним Mini PC: кращий розподіл навантаження, підвищена відмовостійкість та енергоефективність. Завдяки контейнеризації, профілям живлення процесора й розподіленій архітектурі, сервери забезпечують стабільну роботу навіть для ресурсомістких сервісів без перевантаження основних компонентів.

Публікації та апробація роботи

1. Потапенко В.С. Дослідження продуктивності домашнього сервера при використанні віртуалізації. «Проблеми комп'ютерної інженерії»: Науково-практ. конф., м.Київ, 3 грудня 2024 р. С.59-62.
2. Потапенко В.С., Бабенко В.Г. Розгортання домашнього сервера з використанням віртуалізації. «Сучасний стан та перспективи розвитку IoT» : VI Науково-техн. конф., м.Київ, 15 квітня 2025 р. (подано до публікації).