Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: " Підвищення безпеки площини даних віртуального комутатора Open vSwitch "

Виконав: студент VI курсу, групи СБм-61
Спеціальності:
125 Кібербезпека та захист інформації
(шифр і назва напряму підготовки, спеціальності)
Горішний Остап Юрійович
підпис (прізвище та ініціали)

Керівник Карпінський М. П.
підпис (прізвище та ініціали)

Нормоконтроль Стадник М. А.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Горішному Остапу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Підвищення безпеки площини даних віртуального комутатора Open vSwitch

Керівник роботи Карпінський Микола Петрович,
доктор технічних наук, професор кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «24» 11 2025 року № 4/7-1024

2. Термін подання студентом завершеної роботи 12.12.2025

3. Вихідні дані до роботи Архітектура програмного комутатора Open vSwitch із підтримкою DPDK, умови багатокористувацького середовища з конкурентним доступом до CPU-ресурсів та необхідність забезпечення стабільної продуктивності й безпеки площини даних.

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз архітектури Open vSwitch і механізмів оброблення трафіку в OVS-DPDK.
Дослідження впливу конкурентного споживання CPU на продуктивність площини даних.
Побудову моделі витрат процесорних циклів. Розроблення методу ізоляції ресурсів на основі токен-бакета та ієрархічного батч-планування. Інтеграцію запропонованого рішення у PMD-потік та систему керування ovs-vsctl. Провести експериментальну оцінку ефективності.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Актуальність дослідження. Мета, об'єкт, предмет дослідження.
Наукова новизна та практичне значення.
Завдання дослідження. Архітектура хмарних платформ та віртуальних комутаторів.
Вплив ізоляції CPU на безпеку та QoS. Існуючі методи ізоляції ресурсів.
Архітектура середовища тестування.
Механізм справедливого розподілу CPU-циклів на основі токенів.
Ієрархічне планування на основі груп процесів.
Результати тестування.
Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 19.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	20.09 – 22.09	Виконано
2.	Підбір джерел для аналізу в галузі дослідження	25.09 – 10.10	Виконано
3.	Опрацювання джерел в галузі дослідження	11.10 – 15.10	Виконано
4.	Налаштування симуляційного середовища	16.10 – 25.10	Виконано
5.	Оформлення розділу «Аналіз методів забезпечення продуктивності та безпеки віртуальних мережевих сервісів»	25.10 – 05.11	Виконано
6.	Оформлення розділу «Розроблення моделі ізоляції CPU-ресурсів у vSwitch»	06.11 – 10.11	Виконано
7.	Оформлення розділу «Оцінка ефективності ізоляції CPU-ресурсів»	11.11 – 25.11	Виконано
8.	Виконання завдання до підрозділу «Охорона праці та безпека в надзвичайних ситуаціях»	26.11-01.12	Виконано
9.	Оформлення кваліфікаційної роботи	02.12 – 10.12	Виконано
10.	Нормоконтроль	08.12 – 10.12	Виконано
11.	Перевірка на плагіат	12.12 – 14.12	Виконано
12.	Попередній захист кваліфікаційної роботи	15.12 – 20.12	Виконано
13.	Захист кваліфікаційної роботи	22.12.2025	

Студент _____ Горішний О. Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінський М. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Підвищення безпеки площини даних віртуального комутатора Open vSwitch // ОР «Магістр» // Горішний Остап Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2025 // С. 87, рис. – 29, табл. – - , кресл. – 13, додат. – 1.

Ключові слова: Open vSwitch, DPDK, площина даних, CPU, планування оброблення пакетів, кіберстійкість, віртуалізація.

У магістерській кваліфікаційній роботі розглянуто проблему безпеки площини даних у віртуальному комутаторі Open vSwitch, який широко використовується у віртуалізованих і хмарних інфраструктурах. Встановлено, що перенесення оброблення пакетів у користувацький простір із застосуванням DPDK забезпечує високу пропускну здатність, проте створює ризики спільного використання процесорних ресурсів між віртуальними машинами. Це призводить до неконтрольованої конкуренції за CPU-цикли, коливань затримки пакетів і появи часових побічних каналів, що може бути використано для атак типу resource starvation або side-channel. Метою роботи є підвищення безпеки та передбачуваності роботи площини даних Open vSwitch шляхом розроблення та впровадження методів ізоляції процесорних ресурсів. Запропоновано ієрархічний механізм планування оброблення пакетів, який забезпечує стабільну затримку та контроль пріоритетів при одночасному обслуговуванні кількох орендарів.У процесі експериментальних досліджень, проведених у середовищі OVS-DPDK, продемонстровано, що реалізація запропонованих методів забезпечує стабільність пропускної здатності, зниження коливань затримки та ізоляцію навантажень між віртуальними машинами. Отримані результати підтверджують підвищення рівня кіберстійкості площини даних і зменшення впливу побічних каналів.

ABSTRACT

Enhancing data-plane security of the Open vSwitch virtual switch// Thesis of educational level "Master"// Ostap Horishnyi // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБм-61 // Ternopil, 2025 // p. 87, figs. 29, tbls. - , drws. 13, apps. 1.

Keywords: Open vSwitch, DPDK, data plane, CPU, packet processing scheduling, cyber resilience, virtualization.

The master's thesis addresses the problem of data plane security in the Open vSwitch virtual switch, which is widely used in virtualized and cloud infrastructures. It has been established that transferring packet processing to user space using DPDK ensures high throughput but introduces risks of shared CPU resource usage among virtual machines. This leads to uncontrolled competition for CPU cycles, packet delay variations, and the emergence of timing side channels that can be exploited for resource starvation or side-channel attacks.

The aim of the work is to enhance the security and predictability of the Open vSwitch data plane operation by developing and implementing methods of CPU resource isolation. A hierarchical packet processing scheduling mechanism is proposed to provide stable delay and priority control during concurrent tenant servicing. Experimental studies conducted in the OVS-DPDK environment demonstrated that the proposed methods ensure throughput stability, reduce delay variations, and achieve effective workload isolation between virtual machines. The obtained results confirm an increase in the cyber resilience of the data plane and a reduction in the impact of side channels.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SDN	—	Software-Defined Networking
NFV	—	Network Functions Virtualization
QoS	—	Quality of Service
DPDK	—	Data Plane Development Kit
VM	—	Virtual Machine
PMD	—	Poll Mode Drivers
NIC	—	Network Interface Card
NUMA	—	Non-Uniform Memory Access
DoS	—	Denial of Service
IDS	—	Intrusion Detection System
eBPF	—	extended Berkeley Packet Filter
SR-IOV	—	Single Root I/O Virtualization
VF	—	Virtual Function
VT-d	—	Intel Virtualization Technology for Directed I/O
CAT	—	Cache Allocation Technology
OVS	—	Open vSwitch
SLA	—	Service Level Agreement
KVM	—	Kernel-based Virtual Machine
QEMU	—	Quick Emulator
WFQ	—	Weighted Fair Queuing
EMC	—	Exact Match Cache
TSS	—	Tuple-Search-Space
CSP	—	Cloud Service Provider
FIFO	—	First In First Out
CDF	—	Cumulative Distribution Function

ВСТУП

Актуальність теми. Широке впровадження технологій віртуалізації та хмарних обчислень зумовило перехід мережевих функцій на програмно-визначені платформи (SDN/NFV), у яких ключову роль відіграє віртуальний комутатор (vSwitch). Саме він забезпечує обробку та маршрутизацію трафіку між віртуальними машинами, контейнерами та фізичними інтерфейсами. У таких середовищах одночасно функціонують десятки користувацьких сервісів, що конкурують за спільні апаратні ресурси центрального процесора. Неврегульований доступ до CPU-циклів призводить до деградації продуктивності, зниження якості обслуговування (QoS) та створює умови для кіберзагроз, пов’язаних із побічними каналами (side-channel attacks) і атаками виснаження ресурсів (resource starvation). Втрата продуктивності мережевих сервісів у середовищі з віртуалізованими функціями може мати критичні наслідки: порушення доступності, втрату цілісності даних, а у випадку затримки або зупинки обробки трафіку - відмову безпеки на рівні сервісів. Тому забезпечення стабільної продуктивності через ізоляцію CPU-ресурсів стає не лише завданням оптимізації, а й компонентом системного захисту інформації.

Саме тому дослідження методів ізоляції CPU у vSwitch як засобу підвищення безпеки площини даних є актуальним завданням сучасної кібербезпеки.

Мета і задачі дослідження. Метою роботи є вдосконалення підходу до ізоляції CPU-ресурсів у vSwitch, який забезпечує стабільну продуктивність і підвищує рівень кіберзахисту від атак, пов’язаних із конкурентним використанням процесорних ресурсів.

Для досягнення мети необхідно розв’язати такі задачі:

- проаналізувати архітектуру vSwitch та механізми розподілу CPU-циклів між потоками обробки трафіку;
- ідентифікувати загрози безпеці, пов’язані з побічним впливом на процесорні ресурси;

- вдосконалити методику ізоляції CPU у vSwitch, яка поєднує вимоги продуктивності та інформаційної безпеки;
- реалізувати експериментальне середовище для перевірки методу;
- провести оцінку ефективності ізоляції за показниками QoS, стабільності та стійкості до атак.

Об'єкт дослідження. Процес забезпечення безпеки та продуктивності мережевих сервісів у віртуалізованому середовищі передачі даних.

Предмет дослідження. Методи ізоляції процесорних ресурсів у vSwitch, спрямовані на запобігання побічному впливу атак і деградації QoS у багатокористувацьких інфраструктурах.

Наукова новизна одержаних результатів кваліфікаційної роботи. У кваліфікаційній роботі вдосконалено підхід до ізоляції CPU-циклів у vSwitch, орієнтований не лише на підвищення продуктивності, але й на підвищення кіберстійкості до атак побічного впливу на процесорні ресурси. Удосконалено модель управління CPU-потоками шляхом динамічного закріплення ядер за критичними потоками vSwitch. Дістало подальший розвиток аналітичне оцінювання впливу CPU contention на QoS і безпеку мережевих сервісів у середовищі з віртуалізованими функціями.

Практичне значення одержаних результатів. Розроблений метод може бути використаний при побудові захищених хмарних середовищ і NFV-інфраструктур, де забезпечення гарантованої продуктивності є складовою кіберзахисту.

Апробація результатів магістерської роботи. Основні результати дослідження були представлені на XIV Міжнародній науково-практичній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (ТНТУ, Тернопіль, Україна, 11-12 грудня 2025 р).

Публікації. Основні результати кваліфікаційної роботи опубліковано у працях конференції (див. Додаток А).

РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ ПРОДУКТИВНОСТІ ТА БЕЗПЕКИ ВІРТУАЛЬНИХ МЕРЕЖЕВИХ СЕРВІСІВ

1.1 Архітектура віртуального комутатора

1.1.1 Площини управління, контролю та даних

Архітектура сучасних програмно-визначених мереж (SDN) передбачає чітке розмежування функцій між площиною управління (control plane), площиною контролю (management plane) та площиною даних (data plane) [1]. Таке розділення є основою для масштабованості, безпеки й ефективного використання обчислювальних ресурсів у хмарних та NFV-інфраструктурах [2].

Площина управління відповідає за прийняття рішень щодо маршрутизації, комутації й політик безпеки. Вона реалізує логіку визначення, куди має бути переданий пакет, на основі таблиць потоків (flow tables), правил безпеки або динамічних оновлень від SDN-контролера. У контексті vSwitch площина управління часто взаємодіє з центральним контролером (наприклад, OpenDaylight або ONOS) через північний інтерфейс (Northbound API) [3]. Безпекові механізми цієї площини включають перевірку автентичності запитів, контроль цілісності оновлень таблиць потоків і захист від несанкціонованого перезапису політик. Площина контролю забезпечує централізоване адміністрування системи, моніторинг стану ресурсів і координацію між окремими вузлами. Саме тут розташовані функції керування життєвим циклом віртуальних комутаторів, розподілом CPU-ядер, пам’яті, черг DPDK [4] та взаємодія з оркестраторами (OpenStack, Kubernetes, NFV MANO). У контексті кібербезпеки площина контролю є критичною точкою - компрометація її компонентів може дати зловмиснику змогу змінювати ресурси, порушувати QoS або створювати умови для атак типу resource starvation. Тому вона потребує захищених каналів управління, а також розмежування доступу за ролями. Площина даних реалізує безпосередню передачу й обробку трафіку між віртуальними машинами або контейнерами. У vSwitch вона побудована на основі

механізмів швидкої обробки пакетів - наприклад, DPDK або eBPF і використовує CPU-ресурси для виконання потоків обробки (poll mode threads) [5]. Саме площина даних є найбільш навантаженою та найбільш уразливою до атак на продуктивність. Конкуренція між потоками або між орендарями за спільні CPU-цикли може спричинити небезпечні побічні ефекти: затримку трафіку, втрату пакетів, порушення QoS і навіть можливість побічного витоку інформації через вимірювання часових характеристик (side-channel timing attacks).

У межах дослідження площина даних розглядається як об’єкт ізоляції процесорних ресурсів, оскільки саме на цьому рівні відбувається найбільша взаємодія між потоками обробки. Ізоляція CPU-циклів між потоками різних сервісів або орендарів дозволяє:

- гарантувати визначену пропускну здатність і стабільну затримку;
- мінімізувати вплив атак;
- запобігти несанкціонованому спостереженню за часовими характеристиками обробки пакетів.

Таким чином, поділ функціональності на площини управління, контролю та даних створює базу для побудови багаторівневої системи безпеки у vSwitch, де кожен рівень має власні механізми захисту, а ефективна ізоляція CPU-ресурсів у площині даних виступає ключовим компонентом забезпечення конфіденційності, цілісності та доступності мережевих сервісів.

1.1.2 Обробка трафіку

У сучасних хмарних платформах фізичні ресурси такі, як процесор, пам’ять, накопичувачі та мережа абстрагуються шаром віртуалізації, що забезпечує ізольовані середовища для віртуальних машин (VM) (див. рисунок 1.1).

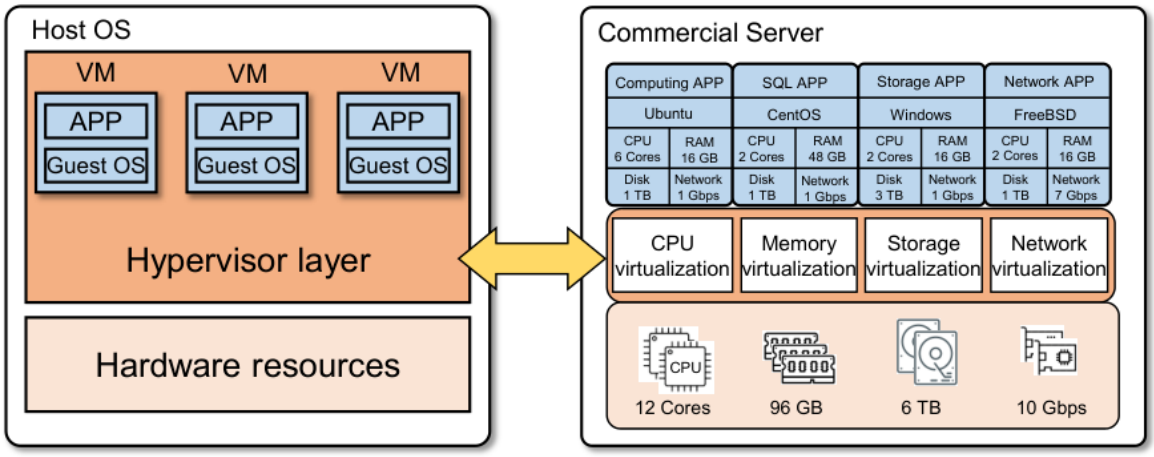

Рисунок 1.1 – Архітектура хмарних платформ

Мережева віртуалізація відрізняється від решти тим, що не спирається на фізичні пристрої напряму, а реалізується за допомогою програмного компонента vSwitch, який відповідає за маршрутизацію, класифікацію та передачу трафіку між віртуальними і фізичними портами. Саме ефективність роботи vSwitch визначає пропускну здатність, затримку і стабільність мережевих сервісів у віртуалізованому середовищі. Ідея використання vSwitch як засобу мережевої віртуалізації походить від класичного Linux Bridge, який діє на канальному рівні (Layer 2) і об’єднує декілька пристроїв у єдиний віртуальний міст [6]. Як показано на рисунку 1.2, Linux Bridge забезпечує базове пересилання кадрів, але потребує багаторазового копіювання пакетів у просторі ядра. Це створює суттєві накладні витрати й обмежує продуктивність, особливо у високонавантажених середовищах (див. рисунок 1.2).

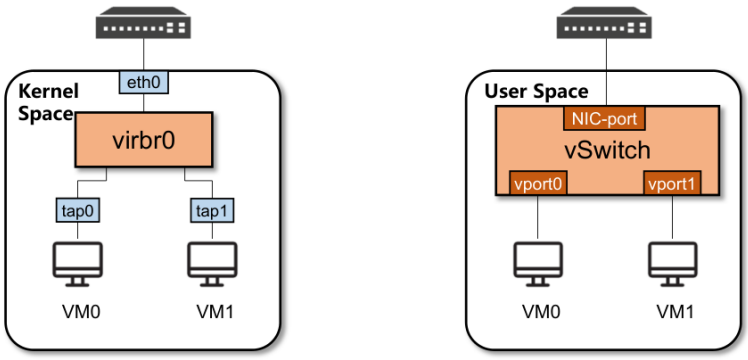

Рисунок 1.2 – Архітектура Linux Bridge та vSwitch віртуальних комутаторів

На відміну від цього, vSwitch перенесено з простору ядра в користувацький простір (user space). Це рішення дозволяє скоротити кількість копіювань пам'яті, зменшити час обробки пакетів та інтегрувати високопродуктивні бібліотеки обробки трафіку, зокрема Intel DPDK. Завдяки цьому всі етапи від прийому пакета до його пересилання виконуються у vSwitch-програмі без контекстних перемикань між просторами користувача і ядра, що забезпечує суттєве зниження затримок.

Кожна віртуальна машина, під'єднана до хмарної платформи, взаємодіє з vSwitch через віртуалізований мережевий інтерфейс. Коли VM надсилає пакет, vSwitch копіює його з пам'яті VM у власний буфер пакета через віртуальний порт (vport). Далі виконує класифікацію за атрибутами (source/destination IP, ports, protocol) та визначає цільовий порт згідно з таблицею потоків (flow table). Передає пакет далі або іншій VM, або до зовнішньої мережі через зворотний виклик (callback) функції передачі. Цей процес відбувається у режимі пакетної обробки (batch mode), що мінімізує накладні витрати на системні виклики та перемикання контексту. Саме завдяки цьому підходу vSwitch досягає високої пропускної здатності, яка може наближатися до швидкодії фізичних комутаторів.

Для підвищення продуктивності vSwitch розподіляє потоки обробки між декількома PMD-потоками, кожен з яких прив'язується до окремого ядра процесора (CPU affinity) [7]. Так реалізується модель per-core packet processing, де кожен потік обслуговує одну або кілька черг прийому-передачі (RX/TX queues). Однак така архітектура, хоч і підвищує пропускну здатність, створює ризик порушення ізоляції CPU-ресурсів. У середовищі з багатьма орендарями або сервісами кілька потоків vSwitch можуть конкурувати за одні й ті самі ядра, що призводить до виснаження процесорних ресурсів (CPU starvation) [8] або часових побічних каналів (timing side-channels) [9]. Зловмисник може штучно створити надмірне навантаження на CPU, знижуючи продуктивність критичних сервісів і потенційно отримуючи непряму інформацію про їхню активність.

У традиційних рішеннях на базі Linux Bridge кожне пересилання пакета передбачає кілька контекстних перемикань між простором ядра та користувача.

Це спричиняє додаткові затримки й споживає CPU-цикли. Перехід vSwitch у користувацький простір (DPDK-OVS) практично усуває ці втрати, проте водночас ускладнює керування доступом до CPU. Усі PMD-потоки змагаються за спільні ядра, що підвищує ймовірність деградації QoS. Таким чином, CPU-ізоляція стає ключовим засобом гарантування стабільності. Її реалізація через cgroups, cpusets або динамічне прив'язування потоків до ядер дозволяє зменшити взаємний вплив між потоками різних VM та забезпечити детерміновану обробку трафіку.

У процесі еволюції vSwitch розробники зосереджувалися переважно на продуктивності й гнучкості, що призвело до розширення його функцій та спільного використання ресурсів між орендарями. Це підвищує ризики:

- взаємного впливу між VM, коли перевантаження одного орендаря знижує продуктивність інших;
- порушення ізоляції мережевих потоків, через спільні буфери і кеш-пам'ять;
- експлуатації побічних каналів, що можуть використовувати часові характеристики обробки для отримання конфіденційних даних.

Отже, ізоляція CPU у vSwitch стає не лише засобом оптимізації, а й елементом кіберзахисту, який запобігає впливу атак типу side-channel, забезпечуючи доступність і цілісність сервісів у багатокористувацьких хмарних середовищах.

1.2 Розподіл CPU-ресурсів у віртуальному комутаторі

У віртуалізованих хмарних середовищах, де одночасно функціонує велика кількість віртуальних машин і мережевих сервісів, розподіл процесорних ресурсів між компонентами vSwitch набуває особливого значення. Ефективність обробки пакетів, пропускна здатність і затримка залежать не лише від програмних алгоритмів маршрутизації, а й від того, як саме організовані потоки обробки та які ядра CPU залучені до виконання цих операцій. Невдале планування потоків або спільне використання ядер різними орендарями може

призвести до перевантаження, затримок і навіть до побічних каналів витоку інформації. Тому питання організації обчислень у vSwitch має подвійний характер - це водночас завдання оптимізації продуктивності й складова кіберзахисту площини даних.

1.2.1 Планування потоків та багатоядерна обробка

Віртуальний комутатор виконує обробку трафіку у вигляді багатопотокової програми, в якій кожен потік відповідає за окрему функцію: прийом, класифікацію, маршрутизацію або передавання пакетів. Основу площини даних складають PMD-потоки - спеціалізовані цикли опитування черг прийому-передачі (RX/TX queues), які постійно читають пакети з буфера мережевого адаптера та передають їх на подальшу обробку. Такий підхід усуває затримки, пов’язані з перериваннями, але натомість потребує постійної участі CPU. Кожен PMD-потік безперервно використовує виділене ядро, що забезпечує передбачувану латентність, проте робить vSwitch високочутливим до розподілу обчислювальних ресурсів.

У багатоядерних процесорах vSwitch зазвичай застосовує модель per-core packet processing, коли кожен потік обслуговує певну підмножину портів або черг. Планувальник потоків формує набір «ядро - черга», мінімізуючи перемикання контексту між ядрами. Це підвищує ефективність кеш-пам’яті та зменшує накладні витрати на синхронізацію. Однак у багатокористувацькому середовищі, де на одному фізичному сервері можуть співіснувати декілька vSwitch-екземплярів або орендарів, виникає проблема конкуренції за CPU-цикли. У такому випадку некоректне планування може призвести до явища resource starvation, коли один потік інтенсивно використовує процесор, блокуючи виконання інших. Щоб уникнути цього, сучасні платформи використовують ізоляцію CPU-пулів. Потоки vSwitch поділяються на групи - робочі (data plane) та службові (control/management plane). Для кожної групи виділяється власний набір ядер, який не використовується сторонніми процесами. Така стратегія, реалізована через механізми Linux cpusets або

контейнерні обмеження cgroups, гарантує стабільну пропускну здатність і передбачуваний час обробки пакетів навіть за високих навантажень.

З точки зору кібербезпеки, ізольоване планування потоків виконує ще одну важливу функцію - воно запобігає часовим побічним каналам (timing side-channels), через які зловмисник може спостерігати коливання часу обробки пакетів, отримуючи інформацію про активність інших орендарів або стан системи. Таким чином, правильна організація багатоядерної обробки є не лише засобом оптимізації, а й механізмом підвищення кіберстійкості віртуалізованого середовища.

1.2.2 Використання DPDK, CPU affinity та NUMA-топології

Для досягнення максимальної продуктивності віртуальні комутатори широко використовують бібліотеку DPDK - це набір користувацьких драйверів і буферів для високошвидкісної обробки пакетів у режимі опитування. DPDK дозволяє програмам отримувати прямий доступ до мережевих інтерфейсів (NIC) без посередництва ядра Linux, обходячи стек TCP/IP і мінімізуючи копіювання даних. Це забезпечує значне зростання швидкодії (до десятків мільйонів пакетів за секунду на ядро), але вимагає ретельного керування CPU-ресурсами. Кожен PMD-потік DPDK прив’язується до конкретного ядра процесора - ця операція відома як CPU affinity або CPU pinning. Вона дозволяє уникнути контекстних перемикань і гарантує, що дані, з якими працює потік, залишаються в кеші обраного ядра. Крім того, CPU pinning усуває міжпотокову конкуренцію і забезпечує визначену затримку обробки, що є критично важливим для сервісів реального часу. У середовищах з багатьма орендарями цей підхід сприяє підвищенню ізоляції між потоками, оскільки ядро, призначене одному клієнту, не використовується іншими. Завдяки цьому зменшується площа потенційних побічних каналів і підвищується рівень кіберзахисту площини даних.

Водночас у багатопроцесорних системах із NUMA-архітектурою (Non-Uniform Memory Access) важливо враховувати топологію взаємодії процесорів і пам’яті [10]. Якщо потік vSwitch виконується на ядрі одного процесора, але

звертається до пам’яті, що належить іншому, це призводить до додаткових затримок через міжвузлові звернення. Для уникнення таких втрат застосовується принцип NUMA-aware scheduling - прив’язка потоків vSwitch і їхніх буферів пам’яті до одного NUMA-вузла. Це не лише зменшує латентність, а й стабілізує показники QoS, що є критично важливим для сервісів безпеки, які потребують гарантованого часу реакції.

Використання DPDK у поєднанні з CPU affinity та NUMA-оптимізацією створює основу для побудови високопродуктивної площини даних, здатної забезпечувати швидкість обробки, порівнянну з апаратними комутаторами. Проте така архітектура вимагає додаткових механізмів контролю доступу до CPU-пулів і моніторингу їхнього стану, адже будь-яке втручання або некоректна конфігурація можуть призвести до деградації продуктивності та порушення ізоляції. Для цього застосовуються інструменти ovs-dpctl, dpdk-procdump, perf, які дозволяють спостерігати використання ядер у реальному часі, а також динамічно регулювати розподіл потоків.

З точки зору кібербезпеки, така організація обчислень мінімізує ризики resource contention-атак. Виділення окремих ядер для критичних сервісів, наприклад систем моніторингу чи фільтрації трафіку, гарантує, що навіть при перевантаженні звичайних потоків функції безпеки залишатимуться працездатними. Це відповідає принципу security by design, згідно з яким продуктивність і безпека мають розглядатися як взаємопов’язані властивості системи.

Таким чином, ефективний розподіл ресурсів процесора у віртуальному комутаторі базується на трьох ключових принципах: плануванні потоків у багатоядерному середовищі, використанні DPDK із закріпленням потоків за ядрами (CPU affinity) та врахуванні NUMA-топології. Сукупне застосування цих методів дозволяє досягти високої пропускної здатності, зберегти ізоляцію між орендарями та забезпечити стійкість площини даних vSwitch до атак, пов’язаних із виснаженням ресурсів або побічними каналами. У подальшому ці принципи стануть основою для розроблення методу ізоляції CPU-циклів, спрямованого на підвищення продуктивності й безпеки віртуальних мережевих сервісів.

1.3 Вплив ізоляції CPU на безпеку та QoS

Ізоляція процесорних ресурсів у vSwitch має важливе значення не лише для стабільної продуктивності, але й для забезпечення кібербезпеки площини даних у хмарних і багатокористувацьких середовищах. Віртуалізація обчислювальних ресурсів створює спільне апаратне середовище, де кілька VM або орендарів можуть використовувати одні й ті самі ядра, кеш-пам’ять, шини та контролери вводу-виводу. У таких умовах відсутність належної ізоляції CPU породжує потенційні канали побічного впливу, атак на виснаження ресурсів і неконтрольоване зниження якості обслуговування (QoS).

Саме тому механізми ізоляції CPU розглядаються як частина системного захисту, що поєднує аспекти продуктивності та інформаційної безпеки.

1.3.1 Побічні канали у багатокористувацьких середовищах

Побічні канали (side channels) - це непрямі шляхи отримання інформації через спостереження за фізичними або часовими характеристиками системи. У віртуалізованих середовищах вони виникають через спільне використання процесорних ресурсів між ізольованими віртуальними машинами. Найбільш поширеними є часові побічні канали (timing side-channels), які дозволяють зловмиснику робити висновки про активність інших процесів на основі вимірювання часу виконання інструкцій, затримок доступу до кешу або черг.

Відомими прикладами таких атак є Prime+Probe, Flush+Reload та Spectre/Meltdown [11], які експлуатують механізми кешування та спекулятивного виконання в сучасних процесорах. В умовах багатокористувацьких хмарних платформ ці вектори атак стають особливо небезпечними, оскільки зловмисник може розмістити свою VM на тому ж фізичному хості, що й цільова, і спостерігати за коливаннями часу доступу до спільних кешів. Це дозволяє йому отримувати непряму інформацію про структуру трафіку, криптографічні ключі або стан мережевих з’єднань.

Перенесення vSwitch у користувацький простір створює високопривілейований процес, який одночасно обслуговує потоки декількох орендарів і активно використовує кеш, пам’ять і CPU. У такій архітектурі без ізоляції процесорних циклів один орендар може впливати на затримки обробки іншого, формуючи побічний канал через вимірювання часу відповіді або завантаження ядра.

Використання CPU affinity та ізольованих пулів ядер (CPU pools) є ефективним контрзаходом проти таких атак. Закріплення окремих потоків vSwitch за конкретними ядрами усуває спільне використання кешів L1/L2 між орендарями, знижує варіативність часу виконання та мінімізує кореляцію між активністю різних VM. Таким чином, ізоляція CPU не лише підвищує стабільність продуктивності, але й зменшує ймовірність реалізації побічних каналів, що базуються на часових або ресурсних спостереженнях.

1.3.2 Атаки типу resource starvation у площині даних

Іншим типом загроз, тісно пов’язаних із продуктивністю та безпекою vSwitch, є атаки виснаження ресурсів (resource starvation attacks) [12]. Їх метою є спровокувати деградацію продуктивності системи шляхом зайняття всіх доступних CPU-циклів або переповнення черг обробки пакетів. Такі атаки належать до категорії DoS [13-16] у площині даних (data plane), оскільки їх наслідком є порушення доступності критичних сервісів або падіння пропускної здатності.

У середовищі vSwitch атака може бути реалізована через генерацію великої кількості малих пакетів (packet flooding), створення надмірної кількості потоків або ініціювання запитів із високою частотою, що змушує PMD-потоки безперервно обробляти нові пакети без можливості вивільнення CPU. Через відсутність апаратного пріоритету або системи справедливого розподілу ресурсів (fair scheduling) звичайні потоки можуть повністю витіснити критичні, такі як контрольні повідомлення SDN або сигнали безпеки. Результатом такої атаки є порушення QoS, підвищення затримок та зниження загальної пропускної

здатності. У найгіршому випадку - це відмова мережевих функцій (IDS, VPN, DHCP, DNS), що загрожує доступності всієї платформи.

Механізми ізоляції CPU здатні значно знизити ризик подібних атак. Якщо для кожного типу трафіку або орендаря створено власний пул ядер, перевантаження одного не вплине на роботу інших. Наприклад, виділення окремих ядер для службових потоків (management plane) гарантує, що навіть за умов перевантаження площини даних система зможе приймати команди від контролера або адміністратора. Додатково можуть застосовуватись механізми cgroups, які обмежують споживання процесорного часу для певних процесів, а також token bucket або rate limiting, які регулюють швидкість обробки пакетів на рівні черг.

З точки зору кібербезпеки, атаки типу resource starvation не лише знижують доступність, а й можуть бути використані як інструмент відволікання або розвідки. Наприклад, спостерігаючи, як змінюється затримка або пропускна здатність при різних навантаженнях, зловмисник може робити висновки про конфігурацію CPU, кількість активних потоків або пріоритети планування у vSwitch. Ізоляція CPU-ресурсів руйнує такі залежності, роблячи поведінку системи більш детермінованою та непрозорою для спостереження.

1.3.3 Взаємозв’язок QoS та інформаційної безпеки

Якість обслуговування (QoS) традиційно асоціюється з технічними параметрами мережі такими, як пропускна здатність, затримка і втрата пакетів. Проте в контексті сучасних віртуалізованих платформ QoS має безпосередній зв’язок з інформаційною безпекою. Зниження продуктивності сервісу через перевантаження CPU або неконтрольовану конкуренцію потоків може призвести до відмови в обслуговуванні, що є порушенням принципу доступності. Більш того, нестабільність часу обробки пакетів створює умови для побічних каналів, а некоректне планування ресурсів може викликати витік або маніпуляцію даними в реальному часі. Ізоляція процесорних ресурсів у vSwitch забезпечує детерміновану поведінку системи, коли кожен потік або орендар має гарантовану

кількість CPU-циклів, а час обробки не залежить від зовнішніх факторів. Це підвищує передбачуваність QoS і одночасно зміцнює три основні властивості інформаційної безпеки. Конфіденційність, завдяки усуненню побічних каналів, які можуть розкривати часові або структурні характеристики трафіку. Цілісність, оскільки стабільне планування ресурсів мінімізує ризик втрати або пошкодження пакетів через перевантаження. Доступність, через гарантування стабільного виділення CPU-ресурсів навіть під час атак або аномального навантаження.

Таким чином, забезпечення QoS і захист інформації не є окремими напрямами, а формують єдину стратегію управління ресурсами у віртуальному середовищі. Ізоляція CPU-циклів виступає ключовим технічним механізмом, який поєднує ці аспекти, створюючи безпечну та передбачувану площину даних.

1.4 Аналіз існуючих методів ізоляції ресурсів

У багатокористувацьких хмарних інфраструктурах ефективна ізоляція ресурсів визначає як рівень продуктивності, так і ступінь безпеки мережевих сервісів. Віртуальний комутатор, виконуючи функції пересилання та обробки трафіку в користувацькому просторі, безпосередньо взаємодіє з CPU, пам’яттю, мережевими інтерфейсами та кешами процесора. Унаслідок цього відсутність контролю за розподілом ресурсів може спричинити побічні канали, атаки виснаження та деградацію QoS.

Сучасні системи ізоляції охоплюють як програмні (CPU pinning, cgroups, cpusets, eBPF), так і апаратні (SR-IOV, VT-d, NUMA-зонування) механізми, які діють на різних рівнях від операційної системи до мікроархітектури процесора.

1.4.1 Класичні підходи до ізоляції процесорних ресурсів

Найпоширенішим підходом до ізоляції процесорних ресурсів є CPU pinning, що передбачає жорстке закріплення окремого процесу або потоку за конкретним ядром CPU. У випадку vSwitch це означає, що кожен PMD-потік DPDK виконується на визначеному ядрі без міграції між процесорними вузлами. Такий

метод дозволяє усунути контекстні перемикання, зменшити затримки доступу до кешу L1/L2 та забезпечити детерміновану продуктивність. Крім того, CPU pinning обмежує спільне використання кешів між орендарями, тим самим зменшуючи ймовірність часових побічних каналів. Недоліком є відсутність гнучкості, тому що закріплені ядра можуть простоювати при нерівномірному навантаженні.

Другим рівнем контролю виступає механізм control groups (cgroups) в Linux. Cgroups дають змогу обмежувати й обліковувати використання ресурсів (CPU, пам’яті, мережі) окремими процесами або групами процесів. Для vSwitch це означає можливість задати квоти процесорного часу для кожного потоку чи орендаря, що запобігає атакам типу resource starvation і гарантує мінімальний рівень QoS. Сучасні версії Linux cgroup v2 дозволяють також динамічне перерозподілення ресурсів між групами, що підвищує ефективність у змінних умовах навантаження.

Механізм cpusets є підсистемою cgroups, що дозволяє визначати набори ядер (CPU sets), доступних певним процесам. Cpusets створюють логічні CPU-пули для різних типів задач, наприклад, окремі ядра для data plane та management plane. Це дозволяє одночасно підвищити продуктивність і забезпечити ізоляцію між потоками, які виконують критичні функції безпеки.

Іншим перспективним інструментом є eBPF - механізм динамічного виконання безпечного байт-коду в ядрі Linux. Завдяки eBPF можна створювати легкі програми для моніторингу використання CPU, профілювання потоків і динамічного застосування політик ізоляції без перезавантаження системи. Наприклад, BPF-програма може виявляти нерівномірний розподіл навантаження між ядрами vSwitch і автоматично коригувати CPU affinity або накладати ліміти на агресивні потоки. Таким чином, eBPF є доповненням до класичних методів pinning та cgroups, що додає адаптивність і можливість інтеграції з системами безпеки в реальному часі.

1.4.2 SR-IOV та апаратні методи ізоляції

На апаратному рівні ізоляція реалізується за допомогою технології SR-IOV, що забезпечує поділ фізичного мережевого адаптера на незалежні віртуальні функції (VF). Кожна VF призначається окремій VM або VNF і має власний PCIe інтерфейс, що дозволяє обійти vSwitch і здійснювати прямий доступ до NIC. У такий спосіб зменшується кількість копіювань даних і затримок, а також усувається конкуренція за CPU між орендарями на рівні площини даних. Перевагою SR-IOV є майже повна ізоляція потоків I/O та апаратне розділення ресурсів між VM, що суттєво підвищує продуктивність. Однак існують і недоліки: складність керування політиками безпеки, обмежена гнучкість динамічного перепризначення портів і зниження прозорості моніторингу трафіку, оскільки пакети минають vSwitch. У системах, де безпека важливіша за швидкодію, SR-IOV зазвичай комбінується з програмним моніторингом на рівні hypervisor або SDN-контролера.

Додатковими апаратними засобами ізоляції є:

- VT-d забезпечує захист пам’яті DMA та відокремлення доступу пристроїв до фізичних адрес;
- NUMA-зонування, яке дозволяє ізолювати пам’ять і ядра в межах окремого процесорного вузла, зменшуючи міжвузлові затримки;
- cache partitioning та Intel CAT, що дозволяють резервувати частину кешу L3 для певних потоків vSwitch, мінімізуючи вплив інших процесів.

Ці технології створюють базу для реалізації hard isolation - апаратно гарантованого поділу ресурсів, що є важливим у критичних інфраструктурах і середовищах із підвищеними вимогами до безпеки 5G Core або NFV-платформах.

1.5 Висновки до розділу 1

У першому розділі проведено аналіз архітектури віртуального комутатора та досліджено вплив ізоляції процесорних ресурсів на продуктивність і безпеку

мережевих сервісів у хмарних середовищах. Показано, що перенесення vSwitch у користувацький простір забезпечує високу швидкодію, проте створює ризики конкурентного доступу до CPU і появи побічних каналів між орендарями.

Розглянуто принципи багатоядерної обробки та планування потоків. Встановлено, що використання DPDK, CPU pinning і NUMA-топології підвищує ефективність кешування, зменшує затримки й стабілізує QoS. Доведено, що ізоляція CPU є водночас засобом оптимізації та складовою кіберзахисту, оскільки зменшує ймовірність атак типу side-channel і resource starvation.

Проаналізовано програмні (cgroups, cpusets, eBPF) та апаратні (SR-IOV, VT-d) методи ізоляції. Установлено, що гібридний підхід, який поєднує гнучке програмне керування з апаратним розподілом ресурсів, є найбільш ефективним для забезпечення стабільної продуктивності та кіберстійкості vSwitch. Отримані результати стали основою для подальшого вдосконалення методу ізоляції CPU-циклів у другому розділі.

РОЗДІЛ 2 РОЗРОБЛЕННЯ МОДЕЛІ ІЗОЛЯЦІЇ CPU-РЕСУРСІВ У VSWITCH

2.1 Аналіз проблеми ізоляції CPU-ресурсів у віртуальному комутаторі

У хмарних платформах віртуальний комутатор є ключовим елементом мережевої підсистеми, що забезпечує передачу трафіку між VM та зовнішньою мережею. Його робота базується на використанні ресурсів центрального процесора, адже вся логіка пересилання пакетів реалізується програмно, без апаратної комутації. Такі компоненти, як OVS із прискоренням DPDK, виконують оброблення трафіку у користувацькому просторі, що дозволяє досягти високої пропускної здатності. Водночас саме цей підхід створює нову проблему - спільне використання CPU-ресурсів різними орендарями, що призводить до взаємного впливу їхніх потоків і втрати ізоляції продуктивності. vSwitch розташований між рівнем гіпервізора та фізичними мережевими інтерфейсами. На відміну від апаратного комутатора, який має фіксовану швидкість пересилання, vSwitch динамічно розподіляє процесорний час між потоками оброблення пакетів. У результаті різні VM конкурують за спільні ядра CPU, і поведінка системи стає непередбачуваною. Фактична пропускна здатність віртуального порту визначається не лише обсягом трафіку, а й кількістю спожитих процесорних циклів, що робить традиційні методи QoS неефективними. vSwitch здійснює пересилання у користувацькому просторі та обробляє пакети пакетами. Така оптимізація зменшує затримку, але призводить до неізольованого використання CPU. Усі черги обслуговуються послідовно, що створює часову конкуренцію між потоками. Із зростанням щільності розміщення VM на фізичних серверах проблема ізоляції стає критичною. У середовищах з десятками орендарів навіть одна «шумна» VM може істотно вплинути на інших користувачів. Наприклад, при передаванні дрібних пакетів розміром 64 байти одна VM споживає майже весь процесорний час на одиницю пропускної здатності, тоді як інші VM з більшими пакетами 1264 байтами відчувають затримки та втрату пропускної здатності. Одне ядро CPU може забезпечити до 9

Gbps для великих пакетів, але лише близько 1 Gbps для малих. Це підтверджує, що залежність пропускної здатності від типу трафіку безпосередньо зумовлена нерівномірним використанням CPU-циклів. У традиційних QoS-механізмах використовуються ліміти швидкості пакетів або кількості байтів, але не враховується обчислювальна складність їхнього оброблення. Якщо дві VM мають однакові швидкісні обмеження, але одна генерує короткі пакети, а інша - великі, то перша витратить значно більше CPU-ресурсів на обслуговування того ж обсягу трафіку. Отже, обмеження за швидкістю не гарантує ізоляції за процесорним часом. Це особливо помітно в OVS-DPDK, де кожен PMD-потік послідовно обробляє пакети кількох VM, створюючи колізії під час зростання кількості активних портів. vSwitch не має механізму, який би пов'язував реальне споживання CPU-циклів із мережею SLA. Замість фізичної ізоляції ресурсів система використовує спільний пул ядер, де всі PMD-потоки обслуговують різні черги VM. Коли одна VM збільшує навантаження, решта отримують менше процесорного часу, що призводить до коливань латентності, перевищення затримок та зриву SLA. Таким чином, існуючі QoS-стратегії не здатні забезпечити стале ізольоване виконання потоків у межах vSwitch. Проблема ускладнюється також архітектурними особливостями DPDK-потоків. Вони працюють у режимі активного опитування (busy polling), виконуючи циклічне оброблення черг приймання та передавання. Кожен цикл включає множину VM, тож виникає змагальня за черговість виконання. На практиці, коли кількість портів зростає, збільшується й час очікування обслуговування черги, що виражається у збільшенні середньої затримки пакетів.

2.2 Середовище тестування

Експериментальні дослідження ефективності методу ізоляції CPU-ресурсів у віртуальному комутаторі проводились у віртуалізованому середовищі, що відтворює типову архітектуру хмарної інфраструктури з використанням сучасних засобів віртуалізації та прискорення оброблення мережевого трафіку. Тестовий стенд побудовано за принципом фізичного сервера-гіпервізора з

розгорнутим комутатором OVS, оптимізованим через DPDK, що забезпечує мінімальні накладні витрати при передачі пакетів між віртуальними машинами.

Фізичний сервер, який виконує роль гіпервізора, побудовано на базі процесора Intel Xeon Silver 4314 із тактовою частотою 2.4 GHz та 32 логічними ядрами. Для забезпечення паралельної роботи віртуальних машин використано 64 GB оперативної пам’яті типу DDR4-3200. Використано накопичувачі NVMe SSD об’ємом 1 TB, що дозволяє знизити затримки при доступі до образів віртуальних машин та журналів OVS-DPDK. Мережеве підключення реалізовано за допомогою двопортового адаптера Intel X710-DA2 (10 GbE), який підтримує апаратне прискорення та сумісний із драйверами DPDK igb_uio і vfio-pci. Архітектура сервера побудована з урахуванням NUMA-розділення (дві області пам’яті), що дозволило оцінити вплив розподілу потоків vSwitch між різними процесорними вузлами на латентність та використання CPU-циклів. Схематично архітектура тестового середовища показана на рисунку 1.1.

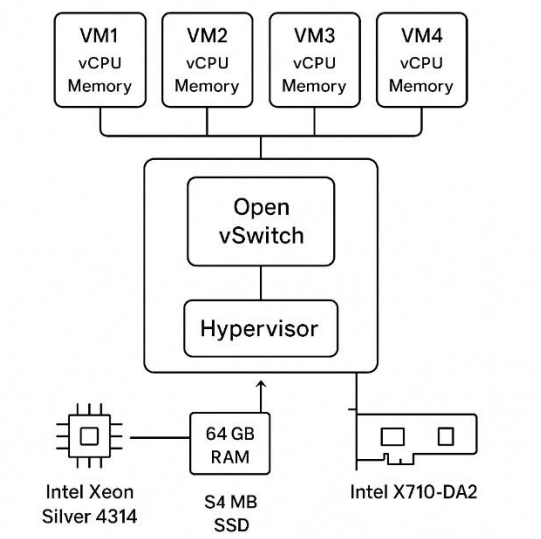

Рисунок 2.1 – Архітектура середовища тестування

На сервері встановлено операційну систему Ubuntu Server 22.04.4 LTS [22] із ядром Linux 6.5.0, що забезпечує стабільну роботу з сучасними версіями DPDK. Для реалізації віртуалізації застосовано гіпервізор KVM [18-20] у комбінації з QEMU 8.2.1 [21], який забезпечує прямий доступ віртуальних машин до апаратних ресурсів CPU через VT-x-інструкції.

Віртуальний комутатор реалізовано за допомогою Open vSwitch 3.3.0 із інтегрованою підтримкою DPDK 24.03 (LTS). Ця комбінація дає можливість створювати високопродуктивні мережеві шляхи між віртуальними портами без контекстних перемикань ядра. DPDK було скомпільовано з підтримкою бібліотеки rte_timer та API rte_eth_rx_burst() для пакетного опитування.

Для відтворення багатокористувацького середовища на гіпервізорі було розгорнуто чотири віртуальні машини (VM1–VM4). Кожна з них отримала 1 віртуальне ядро (vCPU), 2 GB оперативної пам’яті та один віртуальний мережевий інтерфейс, підключений до vSwitch через драйвер vhost-user. Дві віртуальні машини виконували роль добропорядних орендарів (передавання трафіку з фіксованим розміром пакетів 1024 B), а дві інші — роль «шумних сусідів», що генерували короткі пакети розміром 64 B з високою частотою, створюючи конкурентне навантаження на CPU-ядра.

На всіх VM встановлено Ubuntu Server 22.04.4 із утилітами iperf3 [23], qperf [24], pktgen-DPDK та [25] сервісами FTP і Nginx, які використовувались для вимірювання пропускної здатності, затримки, точності лімітування і впливу ізоляції CPU-циклів на прикладні сервіси. Кожна VM підключалася до власного віртуального порту vSwitch.

2.3 Проблема забезпечення QoS у віртуальному комутаторі

Питання забезпечення якості обслуговування (QoS) у мережах є одним із найважливіших завдань як для апаратних, так і для програмно визначених комутаторів. В апаратних рішеннях, зокрема в мережевих процесорах і ASIC-комутаторах, реалізація QoS-механізмів має суттєві переваги: фіксовану тривалість оброблення кожного пакета, апаратну підтримку черг і токен-бакетів, а також високу точність апаратних таймерів. Саме тому механізми керування пропускною здатністю, такі як token bucket або WFQ, працюють у них стабільно та передбачувано, практично не впливаючи на продуктивність.

Однак жодна з перелічених переваг не притаманна програмному комутатору, який функціонує в користувацькому просторі гіпервізора. Його

обчислювальні ресурси обмежені кількома ядрами CPU, і пропускна здатність суттєво залежить від характеристик трафіку. Так, у дослідженнях компанії Google встановлено, що передавання потоку зі швидкістю 512 Мбіт/с і пакетами 64 байти потребує більше процесорних циклів, ніж потік зі швидкістю 2.4 Гбіт/с і пакетами 1518 байт. Ця варіативність продуктивності означає, що стратегії QoS, успадковані з апаратних комутаторів, не можуть бути напряму перенесені у vSwitch, оскільки вони не враховують конкуренцію за CPU-ресурси та змінну пропускну здатність процесорних ядер.

2.3.1 Проблема розподілу пропускної здатності

Існуючі методи лімітування швидкості у vSwitch зазвичай базуються на токен-бакетах, що оперують одиницями bps або pps (біти чи пакети на секунду). Проте такі алгоритми контролюють лише обсяг трафіку, не враховуючи реальне споживання CPU-циклів для оброблення пакетів різного розміру. Для кращого розуміння проблеми в тестовому середовищі було проведено експеримент. На сервері з OVS-DPDK було запущено дві віртуальні машини VM1 і VM2 що спільно використовували одне ядро CPU для форвардингу. Їхня пропускна здатність за допомогою токен-бакету обмежувалася до 2 Гбіт/с і 8 Гбіт/с відповідно. Упродовж перших 10 с VM1 передавала пакети розміром 512 B, і її bps-обмеження дотримувалося точно. Починаючи з 10-ї секунди, VM1 змінила розмір пакета на 64 байти, щоб зберегти ту саму швидкість 2 Гбіт/с (див. рисунок 2.2).

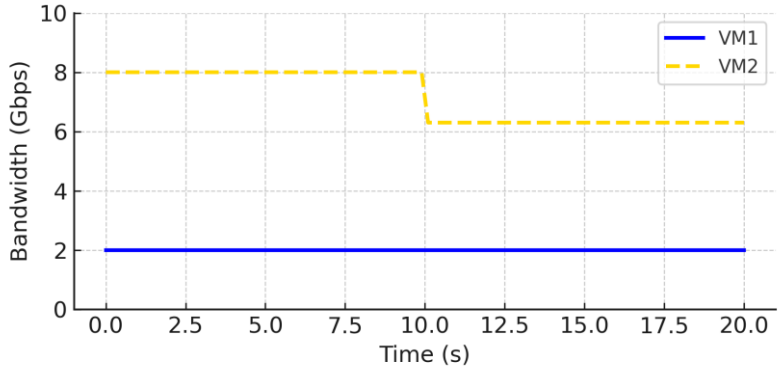

Рисунок 2.2 – Пропускна здатність (Gbps) VM1 та VM2 при bps-обмеженні

Для цього потрібно було виконати значно більше операцій оброблення, що призвело до збільшення споживання CPU на 20 % (див. рисунок 2.3).

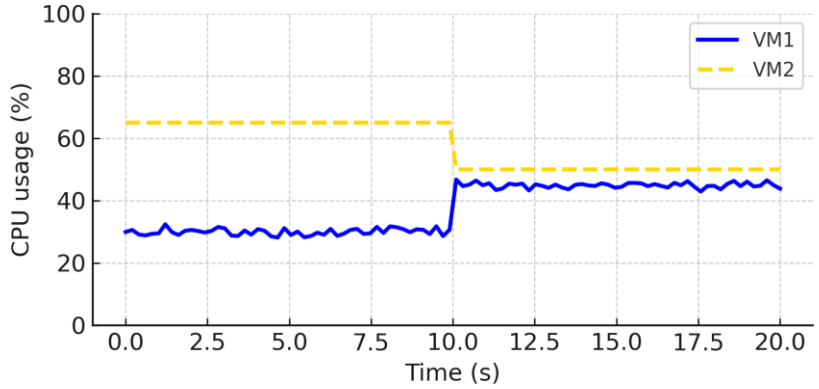

Рисунок 2.3 – Використання CPU (%) обома VM при bps-обмеженні

Через це VM2 отримала менше процесорних циклів і її пропускна здатність зменшилася приблизно на 20 %. Аналогічний ефект спостерігався у pps-орієнтованому токен-бакеті. При збільшенні кількості потоків у VM1 її класифікаційні операції в OVS сповільнювались, що зумовило 16 % втрату пропускної здатності у VM2 (див. рисунок 2.4 та 2.5).

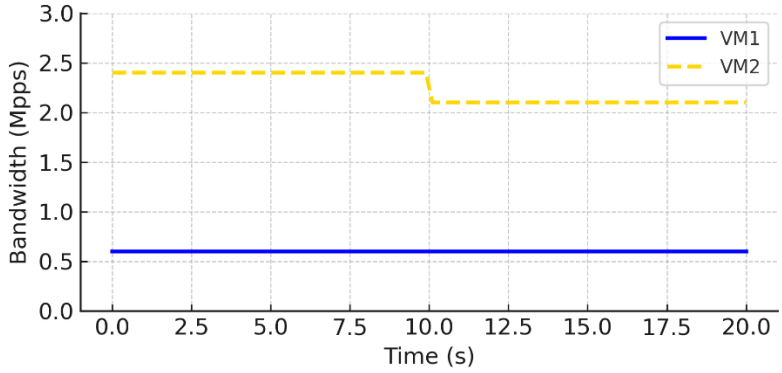

Рисунок 2.4 – Пропускна здатність (Mpps) при pps-обмеженні

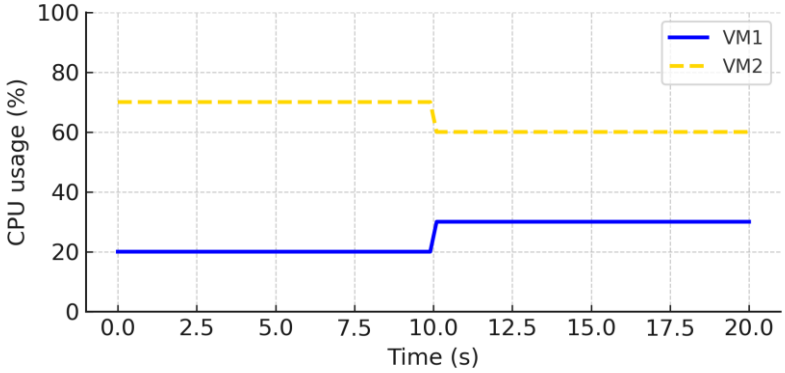

Рисунок 2.5 – Використання CPU (%) при pps-обмеженні

Ці результати свідчать, що поведінка VM може впливати на інші VM, оскільки всі вони конкурують за одні й ті самі ядра CPU, відведені під vSwitch. Таким чином, будь-який агресивний або шкідливий орендар здатний «вичерпати» CPU-ресурси та погіршити продуктивність добропорядних користувачів. У результаті постачальник хмарних послуг не може гарантувати стабільний рівень QoS.

2.3.2 Проблема затримки у віртуальному комутаторі

Питання затримки під час передавання трафіку у віртуалізованих середовищах є одним із найскладніших аспектів забезпечення якості обслуговування. Навіть за умов, коли пропускна здатність між VM формально відповідає встановленим лімітам, практична продуктивність мережі часто виявляється нестабільною. Це проявляється у вигляді коливань затримки (latency jitter), чергування пакетів та нерівномірності часу оброблення пакетів або групи пакетів. У середовищах, де десятки орендарів одночасно використовують один фізичний сервер, такі явища накопичуються і стають причиною істотного погіршення SLA-гарантій. У традиційних апаратних комутаторах причини виникнення затримок добре вивчені, і для їхнього усунення розроблено широкий спектр механізмів планування трафіку - weighted fair queuing (WFQ), deficit round robin (DRR), generalized processor sharing (GPS) тощо. Вони забезпечують пріоритетне або рівноправне обслуговування потоків, виходячи з пропускної

здатності портів і політик SLA. Ключовою перевагою апаратних систем є наявність спеціалізованих мікросхем, які здатні обробляти пакети на лінійній швидкості (line rate) і виконувати планування черг без втрати продуктивності. Таким чином, більшість алгоритмів QoS в апаратних комутаторах реалізуються на рівні egress, тобто на виході порту, де відбувається агрегування кількох потоків на один фізичний інтерфейс.

У рисунку 3.6 показано спрощену логічну схему апаратного комутатора.

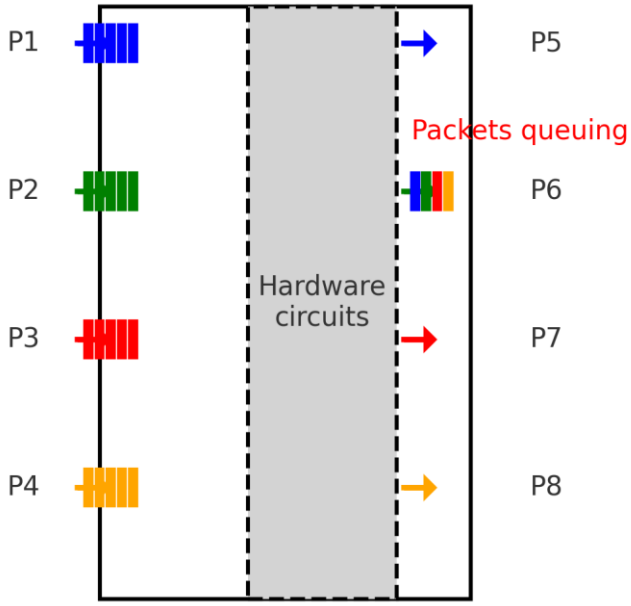

Рисунок 2.6 – Черги пакетів при перевантаженні egress-порту в апаратному комутаторі

Кілька вхідних портів (P1–P4) передають пакети до загального вихідного порту (P6). Черги формуються саме на стадії egress, де реалізуються механізми WFQ, DRR або GPS. Оскільки затримка на оброблення одного пакета у таких пристроях практично стала, планувальник може забезпечити суворе дотримання SLA-затримки для різних потоків.

На відміну від цього, у програмному віртуальному комутаторі, зокрема Open vSwitch з DPDK вся логіка пересилання реалізується у користувацькому просторі та виконується звичайними ядрами CPU. Тут не існує апаратних черг, а планування потоків відбувається на рівні програмних циклів. Це означає, що

будь-яка конкуренція між потоками безпосередньо впливає на використання процесорного часу, а отже і на затримку.

Як показано на рисунку 3.7, у vSwitch основні ресурси спільно використовуються під час етапу ingress, тобто на вході в комутатор, коли віртуальні машини передають свої пакети до vSwitch.

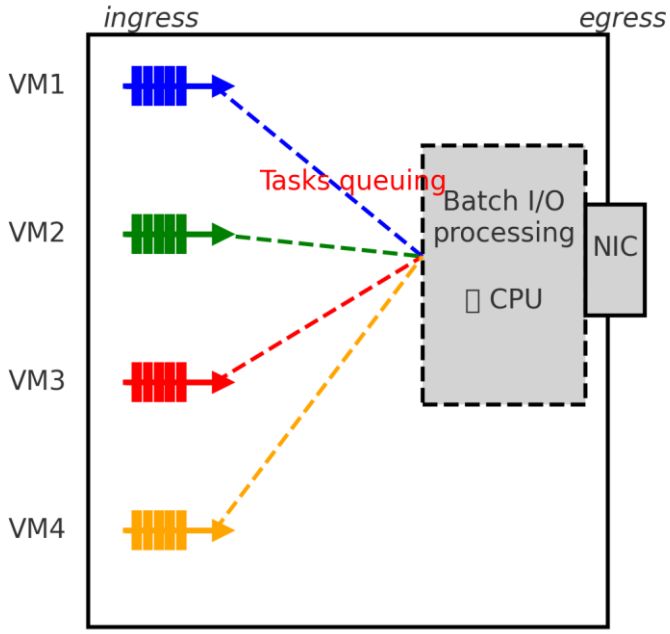

Рисунок 2.7 – Черги задач перед CPU у спільному блоці Batch I/O для VM1–VM4

Кожен PMD-потік опитує черги приймання та передавання, виконуючи пакетне оброблення (функції rte_eth_rx_burst() та rte_eth_tx_burst() в DPDK). При цьому кілька VM ділять між собою один і той самий PMD-потік або ядро CPU. Унаслідок цього групи пакетів від різних VM виконуються послідовно, без справжнього паралелізму, що створює часову конкуренцію (temporal contention). Коли таких VM багато, черги пакетів накопичуються в пам’яті користувацького простору, а кожна наступна партія чекає завершення попередньої. Виникає ефект «ланцюгової затримки», коли навіть короткі пакети змушені очікувати закінчення великої серії оброблення від іншого орендаря. На практиці це означає, що кожна VM вимушена чекати n – 1 разів часу оброблення пакетів інших VM, якщо на сервері працює n віртуальних машин. Така поведінка є

прямим наслідком відсутності механізму планування на ingress-етапі, який би розподіляв CPU-ресурси між VM відповідно до їхніх пріоритетів чи політик SLA.

Для моделювання багатокористувацького середовища на одному фізичному сервері було розгорнуто 1, 4, 8 та 16 віртуальних машин, які з'єднувалися через OVS-DPDK. Для вимірювання затримки TCP використовувалася утиліта qperf, що запускалася на всіх VM у ролі клієнтів, тоді як серверна частина програми розміщувалася на іншому фізичному вузлі, підключеному через 10-гігабітний інтерфейс. Кожен експеримент повторювався двадцять разів, аби усереднити результати та виключити випадкові коливання. Результати подано у рисунку 3.8, де наведено діаграми розмаху (box-plot) для кожного варіанта конфігурації.

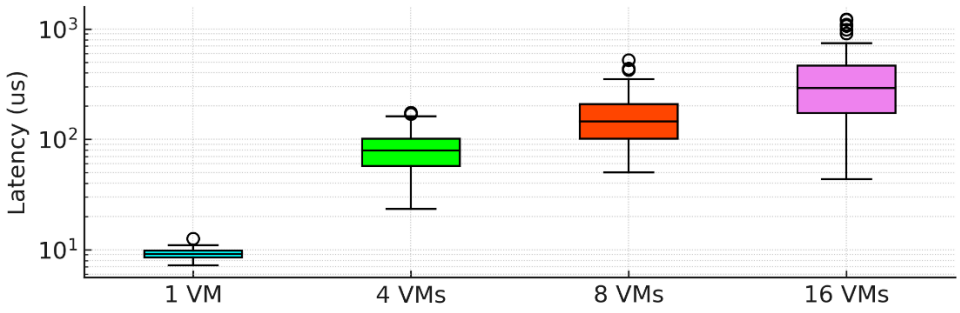

Рисунок 2.8 – Затримка TCP при різній кількості розгорнутих віртуальних машин

При роботі лише однієї VM затримка TCP була стабільною - близько 26–27 мкс, що відповідає очікуваному часу оброблення пакета на рівні одного PMD-потоку. Проте зі збільшенням кількості VM спостерігається експоненційне зростання як середнього, так і максимально можливого значення затримки. Для чотирьох VM затримка подвоїлася, для восьми - зросла приблизно в десять разів, а при шістнадцяти VM досягла сотень мікросекунд. При цьому всі VM зазнавали однакової деградації, незалежно від їхніх пріоритетів або інтенсивності трафіку.

Причина полягає у тому, що єдине ядро CPU, виділене для vSwitch, послідовно опрацьовувало I/O-запити всіх VM. Оскільки процесор не розрізняє походження пакетів, черга задач формується за принципом «першим прийшов - першим обслужили». Це створює ефект недиференційованої високої затримки,

коли навіть критичні служби не мають переваги перед низькопріоритетними потоками. У середовищах із великою кількістю VM це призводить до порушення SLA та потенційних відмов у часі реакції, особливо для додатків реального часу (VoIP, IoT-моніторинг, онлайн-ігри).

Варто підкреслити, що така поведінка не є наслідком браку пропускної здатності мережевого адаптера. Навіть за наявності запасу смуги пропускання 10 Gbps, vSwitch не здатен одночасно обробляти декілька потоків через обмежену кількість CPU-циклів і відсутність внутрішнього планувальника. Тому вузьким місцем у програмних комутаторах є не мережевий інтерфейс, а саме процесорне оброблення. Додатково, механізм batch I/O processing, хоча і підвищує загальну пропускну здатність, негативно впливає на QoS з точки зору затримки. У DPDK пакети обробляються групами (наприклад, по 32 або 64 пакети за раз), що мінімізує виклики до NIC. Проте якщо одна VM генерує інтенсивний трафік, вона може постійно наповнювати чергу пакетів, не залишаючи вікна часу для інших VM. У результаті короткі запити, такі як DNS чи HTTP control-пакети, змушені чекати завершення великого пакету, що складається з даних іншого орендаря. Це призводить до ефекту черги домінування (queue domination), коли один трафік блокує інші. Крім того, програмні таймери, що використовуються у vSwitch для вимірювання часу оброблення, не мають апаратної точності. Асиметрія між ядрами, гіперпотоковість (SMT) та між NUMA-зв’язки збільшують варіацію затримок. Коли PMD-потік перемикається між NUMA-вузлами, доступ до L3-кешу та LLC затримується, що також накопичується у сумарну затримку. Навіть за правильно налаштованих лімітів пропускної здатності, відсутність керування CPU-ресурсами на етапі ingress є головною причиною непередбачуваних затримок у vSwitch. Цю проблему неможливо усунути простим розширенням пропускної здатності мережі або додаванням нових черг, оскільки вона має обчислювальну, а не мережеву природу. Лише ізоляція та планування процесорних циклів здатні забезпечити контрольовану затримку.

Таким чином, аналіз результатів тестування показує, що класичні апаратні методи планування не можуть бути безпосередньо застосовані у vSwitch через

принципово іншу природу конкуренції ресурсів. У той час як у hardware switch ресурсом є пропускна здатність лінії, у software vSwitch основним обмежувальним чинником стають CPU-цикли. Ігнорування цього фактора призводить до неконтрольованого зростання затримки, особливо в умовах високої щільності віртуалізації. Отже, проблема затримки у vSwitch має системний характер. Вона виникає через відсутність управління процесорним часом на рівні оброблення пакетів. Для її вирішення необхідно інтегрувати моделі розподілу CPU-ресурсів у саму архітектуру QoS, забезпечивши пріоритетне планування задач на основі процесорних циклів. Саме такий підхід реалізовано в технології ізоляції CPU-циклів, яка буде детально розглянута в наступних розділах цієї роботи.

2.4 Модель взаємозв'язку пропускної здатності та використання CPU

Щоб коректно спроєктувати стратегію QoS у віртуальному комутаторі, необхідно формально пов'язати мережеву спроможність пересилання з реальними витратами процесорного часу. У цій роботі ми спираємося на архітектуру OVS-DPDK, яка відображає типову логіку оброблення пакетів у більшості сучасних vSwitch і є промисловим стандартом. Відтак метод моделювання, показаний нижче, легко переноситься й на інші платформи віртуальних комутаторів.

2.4.1 Процедура форвардингу та розклад циклів CPU

Архітектура віртуального комутатора Open vSwitch з підтримкою DPDK (OVS-DPDK) ґрунтується на ідеї перенесення критичних операцій пересилання трафіку у користувацький простір із використанням високопродуктивних бібліотек доступу до мережевих інтерфейсів. Основу механізму оброблення пакетів становлять так звані PMD-потоки. Кожен із них закріплюється за певним логічним ядром процесора, утворюючи набір так званих I/O-dedicated cores, тобто ядер, призначених виключно для мережевого введення-виведення.

На відміну від традиційних мережевих драйверів, які реагують на переривання від мережевого адаптера, PMD-потоки працюють у режимі активного опитування (polling). Вони постійно перевіряють черги приймання та передавання пакетів, мінімізуючи затримки, пов’язані з перемиканням контексту ядра. Для підвищення ефективності оброблення використовується пакетна обробка (batch I/O processing) - оброблення одразу кількох дескрипторів пакетів за один цикл. Це суттєво зменшує виклики до апаратури, покращує кеш-локальність і дозволяє досягати продуктивності, наближеної до лінійної швидкості 10 Gb/s.

Згідно зі схемою, поданою на рисунку 2.9, процедура форвардингу у vSwitch OVS-DPDK складається з трьох основних етапів: ingress, classification та egress.

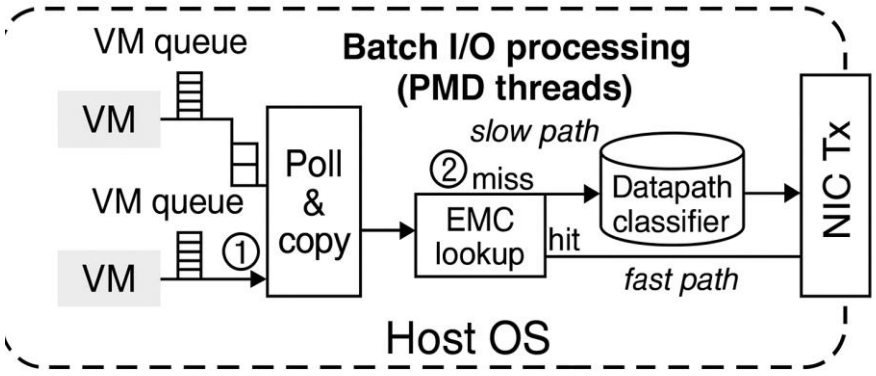

Рисунок 2.9 – Процедура форвардингу у vSwitch OVS-DPDK

Кожен етап відповідає певному набору операцій і споживає різну частку процесорних циклів. Загальні витрати обчислювальних ресурсів на пересилання пакетів можна подати у вигляді суми трьох компонентів:

$$B = B_{ingress} + B_{classification} + B_{egress} \quad (2.1)$$

де: $B_{ingress}$ - кількість процесорних циклів, витрачених на етапі приймання пакетів від віртуальної машини (копіювання даних із пам’яті VM у буфер vSwitch); $B_{classification}$ - витрати на класифікацію пакетів і пошук відповідного правила маршрутизації; B_{egress} - витрати на підготовку та передавання дескрипторів пакетів до черги NIC для відправлення у зовнішню мережу.

На першій стадії PMD-потік зчитує чергу пакетів, створену VM, і копіює групу дескрипторів у власний буфер користувацького простору vSwitch. На цьому етапі відбувається безпосередній обмін із пам'яттю гостя, тому $B_{ingress}$ суттєво залежить від розміру пакетів і місця розташування оперативної пам'яті VM у NUMA-архітектурі. Чим більший обсяг даних потрібно передати та чим віддаленіший NUMA-вузол, тим більше циклів CPU споживається для копіювання. У типових вимірюваннях збільшення розміру пакета з 64 до 1500 байт при незмінній частоті пакетів приводить до зростання $B_{ingress}$ більш ніж удвічі.

На другій стадії PMD-потік виконує класифікацію пакетів, тобто визначення вихідного порту або черги, куди має бути направлений пакет. Для цього використовується система багаторівневого пошуку за п'ятьма ключами (five-tuple: IP-адреси джерела та призначення, порти, протокол). Спочатку здійснюється перевірка у високошвидкісному кеші (EMC), який містить нещодавно використані п'ятірки полів. Ємність EMC у OVS-DPDK обмежена (приблизно 8192 записи), тому він не може зберігати повну таблицю потоків. Якщо збіг знайдено (cache hit), витрати $B_{classification}$ мінімальні, оскільки PMD-потік просто зчитує попередньо відому інформацію. Якщо ж відбувається промах (cache miss), запускається повільніший алгоритм пошуку у повному datapath classifier - структурі на зразок tuple-search-space (TSS), яка містить усі правила комутатора. Після вдалого пошуку новий запис додається назад у EMC, витісняючи найменш використовуваний. Отже, у разі промахів процесорні витрати зростають кратно. До витрат на пошук додається час оновлення кешу. На практиці різниця між випадками hit і miss може сягати порядку величини. Цей ефект особливо відчутний при великій кількості коротких або динамічних потоків, коли таблиця EMC часто переписується.

Третій етап полягає у підготовці оброблених дескрипторів до передавання на мережевий адаптер. PMD-потік записує їх у чергу NIC (Tx Queue) і ініціює передачу. Тут затрати циклів порівняно невеликі, однак із зростанням кількості VM чи паралельних потоків можливе збільшення часу очікування через необхідність синхронізації доступу до черг Tx та Rx. У сучасних реалізаціях

OVS-DPDK (версії 3.3 і вище) частину цих витрат компенсують за рахунок оптимізації блокувань і використання lock-free черг, але залежність між кількістю PMD-потоків і ефективністю все одно залишається майже лінійною.

Сумарне споживання CPU-циклів у vSwitch визначається не лише швидкістю потоку (pps), а й його структурою, кількістю потоків, частотою промахів у EMC та розташуванням VM у NUMA-середовищі. Таким чином, модель дає змогу розкласти загальні витрати на елементарні компоненти та оцінити, які чинники мають найбільший внесок. Для однотипних великих пакетів переважає частка $B_{ingress}$, тоді як для багатопотокового трафіку критичним стає $B_{classification}$. Саме останній компонент є головним джерелом непередбачуваності продуктивності, оскільки кількість промахів у EMC не фіксована й залежить від профілю трафіку.

Отже, процедура форвардингу у vSwitch OVS-DPDK є складною багатоступеневою системою, де кожен етап має власну «вартість» у CPU-циклах. Її аналіз дозволяє побудувати кількісну модель зв’язку між пропускною здатністю та використанням процесора, що є фундаментом для подальшого розроблення технології ізоляції ресурсів vSwitch на основі CPU-циклів, орієнтованої на управління ресурсами не в bps чи pps, а в реальних процесорних циклах.

2.4.2 Чинники споживання CPU на трьох етапах

Ефективність пересилання пакетів у віртуальному комутаторі залежить від кількох взаємопов’язаних чинників, які по-різному впливають на три основні складові процесорних витрат: $B_{ingress}$, $B_{classification}$, B_{egress}.

Згідно з експериментальними результатами, усі фактори можна умовно поділити на дві групи. Перша група - характеристики трафіку, які визначаються самим орендарем (tenant) усередині віртуальної машини. Друга - параметри розгортання, що контролюються постачальником хмарних послуг (CSP) і пов’язані з апаратною організацією сервера. Обидві групи мають бути враховані при побудові моделі взаємозв’язку «пропускна здатність – CPU».

Вплив характеристик трафіку. Найочевидніший чинник - це частота відправлення пакетів, що зазвичай вимірюється у пакетах за секунду (pps). У контрольованому експерименті, де використовувалася одна VM і одне ядро CPU, а розмір пакета залишався сталим (1500 байт), виявлено майже ідеально лінійну залежність між інтенсивністю трафіку та споживанням процесорних циклів у всіх трьох етапах оброблення (див. рисунок 2.10).

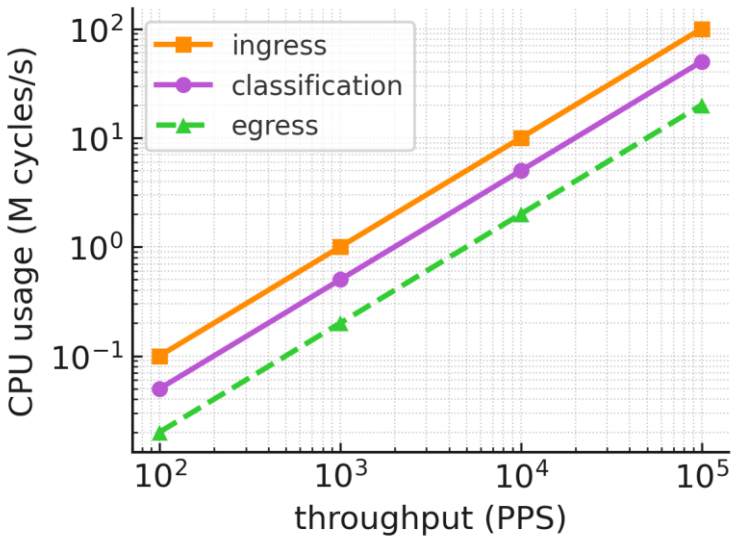

Рисунок 2.10 – Залежність споживання процесорних циклів від інтенсивності трафіку (pps)

Це пояснюється тим, що кожен пакет потребує фіксованої кількості базових інструкцій для копіювання, класифікації та передачі, тому збільшення кількості пакетів пропорційно збільшує загальні витрати CPU. Цей результат підтверджує емпіричну основу, на якій ґрунтуються класичні методи лімітування bps/pps: доки не змінюються інші характеристики трафіку, конкуренція за процесорні ресурси між VM не виникає.

Друга характеристика, що впливає насамперед на $B_{ingress}$, - це середній розмір пакета. Копіювання великих блоків даних з пам’яті VM у буфер користувацького простору комутатора збільшує тривалість виконання інструкцій memcpy() у PMD-потоці. Під час експерименту при сталому значенні pps = 10^5 і фіксованій кількості потоків (один) було зафіксовано, що збільшення розміру пакета з 64 до 1500 байт призводить до зростання $B_{ingress}$ більш ніж удвічі (див. рисунок 2.11).

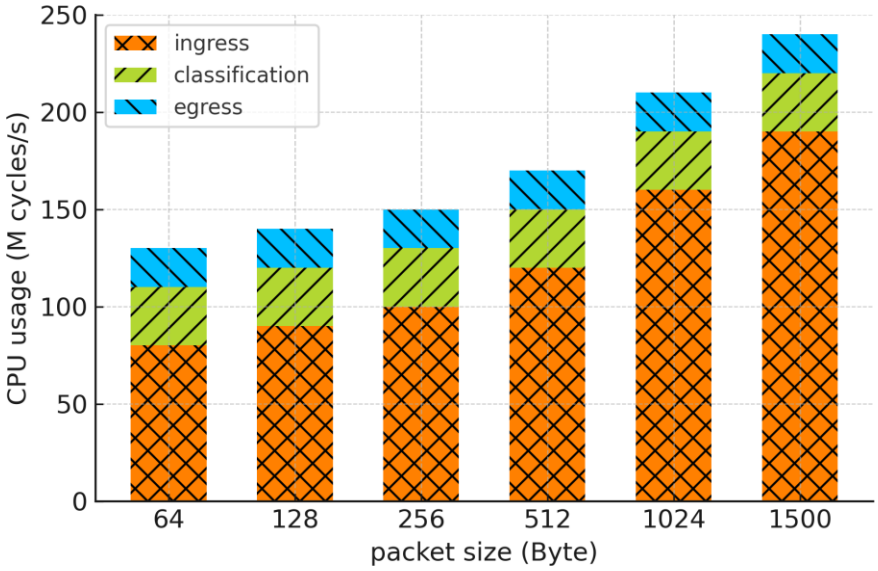

Рисунок 2.11 – Вплив розміру пакета на споживання CPU

При цьому витрати на класифікацію $B_{classification}$ та передачу B_{egress} залишаються майже незмінними, оскільки ці етапи не залежать від розміру даних, а визначаються кількістю об’єктів метаданих (дескрипторів) і складністю пошуку у таблиці потоків.

Третя характеристика трафіку - множинність активних потоків. У випадку, коли віртуальна машина генерує лише один потік (single flow), більшість пошуків у таблиці відповідностей завершується в кеші exact match cache (EMC), що забезпечує мінімальну вартість класифікації. Однак, коли кількість потоків зростає до сотень або тисяч, кеш швидко заповнюється, і частота промахів зростає. Кожен промах змушує PMD-потік виконувати пошук у повнішому класифікаторі (datapath classifier), який має вищу обчислювальну складність. Згідно з вимірюваннями (див. рисунок 2.12), при переході від одиночного до багатопотокового режиму споживання циклів на класифікацію може зрости приблизно у 1.6 раза.

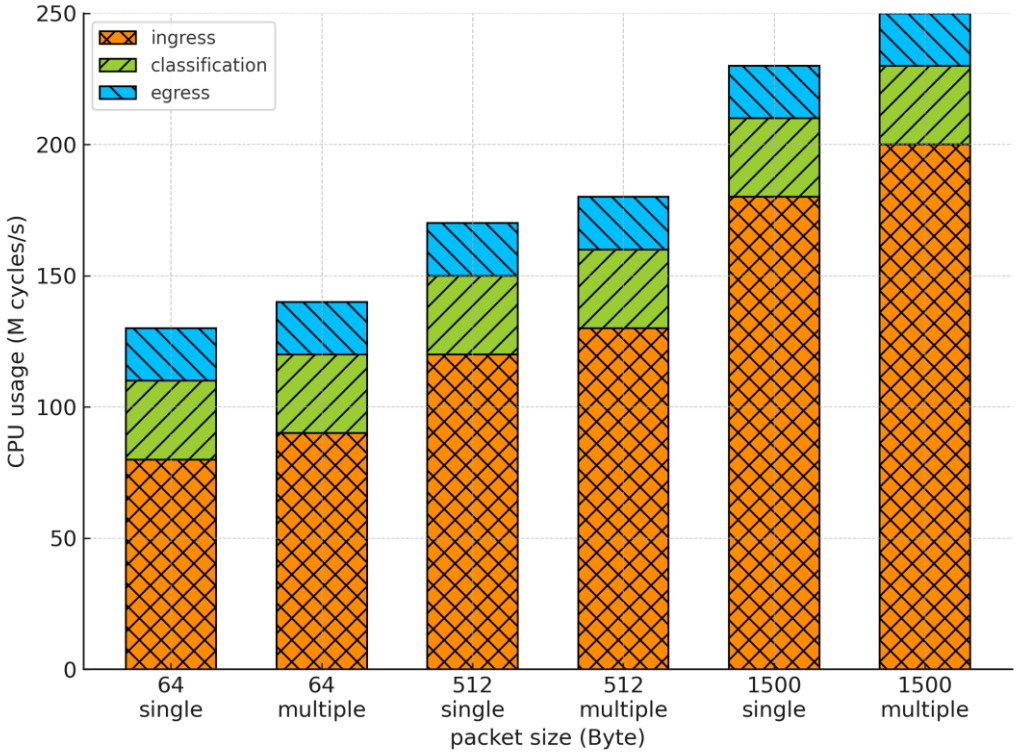

Рисунок 2.12 – Вплив кількості потоків на споживання CPU

Таким чином, $B_{classification}$ виявляється найчутливішим компонентом до зміни характеру трафіку.

Загалом ці результати свідчать, що для ізольованої VM залежність між параметрами трафіку та витратами CPU є детермінованою й може бути формалізована через функцію:

$$B_{single} = f(pps, pkt\_size, flows) \quad (2.2)$$

Яка є базовим елементом подальшої моделі «пропускна здатність – CPU».

Вплив параметрів розгортання у середовищах із багатьма орендарями додаткові фактори апаратного та конфігураційного характеру здатні істотно змінити споживання CPU навіть при однакових характеристиках трафіку. Вони охоплюють розташування VM відносно NUMA-вузлів, кількість VM, запущених на одному вузлі та кількість ядер, виділених для обслуговування PMD-потоків у vSwitch.

У сучасних серверах пам’ять поділена між кількома вузлами NUMA (Non-Uniform Memory Access). Якщо віртуальна машина працює на вузлі, відмінному від вузла, де розташовані ядра PMD-потоків, кожен доступ до її пам’яті здійснюється через міжвузловий інтерфейс QPI/UPI, що збільшує латентність

копіювання. Для системи з чотирма вузлами було зафіксовано, що VM, розміщені на вузлах 1–3, споживають у середньому на 45 % більше процесорних циклів, ніж VM, розташовані на вузлі 0 (де працює vSwitch) (див. рисунок 2.13).

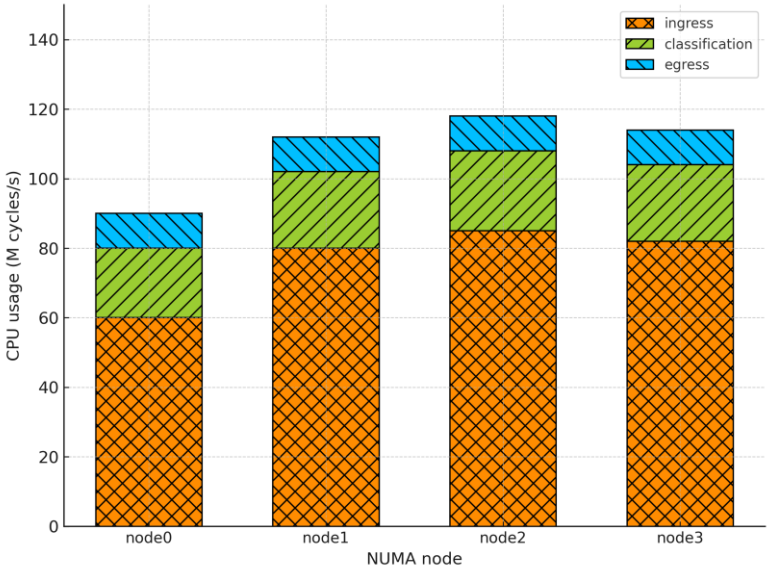

Рисунок 2.13 – Вплив розташування віртуальної машини (NUMA-вузол) на споживання процесорних циклів VM

Отже, коригувальний коефіцієнт для цього чинника становить приблизно $K_{NUMA} \approx 1.45$

Зі зростанням кількості VM на одному фізичному вузлі зростає конкуренція за ресурси пам’яті, кеш-підсистеми та міжпроцесорну шину. Особливо чутливою є взаємодія через L3-кеш, який є спільним для кількох ядер. На графіку (див. рисунок 2.14) показано, що коли кількість VM збільшується до 8, додаткове споживання CPU швидко зростає, після чого стабілізується.

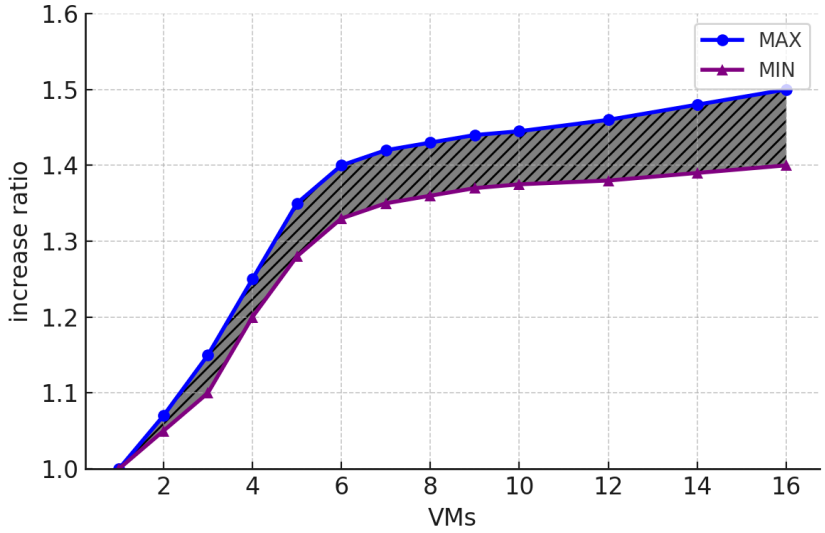

Рисунок 2.14 – Вплив кількості віртуальних машин на споживання CPU

При запуску 4 VM на одному NUMA-вузлі CPU-витрати ($K_{VM\_density}$) кожної з них збільшуються у 1.15 раза порівняно з одиночним випадком.

Ще одним важливим параметром є кількість логічних ядер, виділених для обслуговування PMD-потоків. На перший погляд, збільшення їх кількості має покращити продуктивність, проте в реальності це супроводжується накладними витратами на синхронізацію. У багатопотоковій конфігурації OVS-DPDK кілька PMD-потоків конкурують за доступ до спільних структур даних (черги, статистика, таблиці потоків), тому необхідні блокування та атомарні операції. За результатами вимірювань, перехід від одного до двох ядер збільшує загальне споживання CPU (K_{cores}) приблизно у 1.5 раза. Цей приріст пов’язаний не з подвоєнням реальної роботи, а з додатковими витратами на координацію потоків.

2.4.3 Формування моделі споживання CPU

Сукупний аналіз експериментальних результатів показав, що реальні витрати процесорних циклів у віртуальному комутаторі є функцією не лише параметрів трафіку, а й конфігураційних характеристик середовища розгортання. Тобто швидкість оброблення пакетів у vSwitch визначається комплексом факторів, де перетинаються інтереси двох сторін - орендаря, який

генерує трафік певної інтенсивності та структури, і постачальника хмарних послуг (CSP), який визначає фізичне розміщення віртуальних машин і виділяє їм ресурси CPU. Таке багатофакторне середовище потребує універсальної аналітичної моделі, що дозволить оцінювати вплив кожного з чинників на загальні витрати CPU та забезпечувати справедливий розподіл процесорних циклів між клієнтами. У результаті дослідження було запропонувано модель споживання процесорних циклів, яка описує реальну потребу кожної віртуальної машини у CPU при заданому профілі трафіку та конкретній конфігурації апаратного середовища.

Узагальнене рівняння моделі має вигляд:

$$B_{VM} = B_{single} \times \prod_{i=1}^{n} K_i \qquad (2.3)$$

де: B_{VM} - реальна кількість процесорних циклів, необхідних для забезпечення заданої пропускної здатності віртуальної машини у багатокористувацькому середовищі; B_{single} - базове значення споживання CPU, отримане в експерименті для ізольованої VM при відомих характеристиках трафіку (pps, розмір пакета, кількість потоків); K_i - множники зростання, що враховують вплив кожного фактора розгортання (наприклад, розміщення у NUMA-вузлі, кількість VM на вузлі, кількість ядер). Множники K_i визначаються експериментально, і для кожного з них може бути створено довідкову таблицю або емпіричну залежність, яку зберігає система керування ресурсами CSP.

Модель демонструє, що поведінка vSwitch не є лінійною системою, у якій пропускна здатність напряму пропорційна кількості виділених ядер або частоті процесора. Навпаки, на кожному рівні існують приховані нелінійні залежності - наприклад, через NUMA-ефекти. Використання моделі дозволяє інтегрувати всі ці ефекти в єдину формулу без складних аналітичних перетворень. Таким чином, модель має адаптивний характер: для нової апаратної конфігурації CSP може повторно провести вимірювання, оновити значення множників K_i і автоматично скоригувати систему планування CPU-циклів. Це забезпечує високу масштабованість і практичну придатність для реальних центрів оброблення даних. Модель є базовим елементом у технології ізоляції ресурсів vSwitch на

основі CPU-циклів. Завдяки їй віртуальний комутатор може оцінити, скільки процесорних циклів потрібно виділити кожному трафіку, щоб гарантувати його SLA-рівень пропускної здатності та затримки. На відміну від класичних підходів, які оперують лише обсягом даних (bps або pps), модель напряму пов’язує мережеві показники із реальними обчислювальними витратами. Це відкриває можливість для створення CPU-орієнтованих політик планування, коли орендарі отримують не просто ліміт смуги пропускання, а частку процесорних циклів комутатора, що відповідає їхньому трафіку. Такий підхід забезпечує ізоляцію продуктивності між віртуальними машинами, зменшує вплив «шумних сусідів» і підвищує передбачуваність мережевої поведінки системи.

2.5 Висновки до розділу 2

В другому розділі було обґрунтовано потребу ізоляції процесорних ресурсів у vSwitch і показано, що саме конкуренція за CPU-цикли є ключовою причиною нестабільної пропускної здатності та затримок у багатокористувацьких середовищах. На відміну від апаратних комутаторів, де QoS підтримується апаратно, vSwitch у користувацькому просторі залежить від часу PMD-потоків, тому bps/pps-ліміти не гарантують ізоляції продуктивності. Спроєктоване тестове середовище дозволило виміряти вплив профілю трафіку та конфігурації розгортання на пропускну здатність, затримки і використання CPU. Експерименти показали, що «шумні сусіди» здатні зменшувати пропускну здатність добропорядних VM і підвищувати затримки навіть за формально дотриманих лімітів. Проаналізовано два вузькі місця QoS у vSwitch.

Запропоновано модель, що розкладає витрати CPU на три компоненти (ingress, classification, egress) та показує їхню залежність від pps, розміру пакета, кількості потоків і NUMA, щільності розміщення VM та кількості ядер. Сформульовано модель де реальна потреба VM у CPU дорівнює базовому значенню, помноженому на добуток емпіричних коефіцієнтів розгортання. Таким чином, розділ створив теоретико-експериментальну основу для переходу до CPU-орієнтованого QoS і подальшого впровадження технології ізоляції

ресурсів vSwitch на основі CPU-циклів для гарантованої пропускної здатності та керованої затримки кожної VM.

РОЗДІЛ 3 ОЦІНКА ЕФЕКТИВНОСТІ ІЗОЛЯЦІЇ CPU-РЕСУРСІВ

3.1 Умови експеременту

3.1.1 Передумови та політика розміщення VM

Віртуалізовані комутаційні середовища сучасних хмарних дата-центрів ґрунтуються на програмному комутаторі OVS, який реалізує усі операції пересилання пакетів у користувацькому просторі, використовуючи DPDK. Така архітектура забезпечує високу продуктивність – понад 10 Гбіт/с на одне ядро, але створює нову проблему – спільне використання CPU-ресурсів кількома орендарями. Якщо кілька віртуальних машин одночасно генерують інтенсивний трафік, потоки DPDK можуть неконтрольовано змагатися за цикли процесора. Це спричинює непередбачувані коливання затримки (latency jitter), втрати пакетів і деградацію QoS. Відсутність механізму ізоляції CPU призводить до класичного ефекту “noisy neighbor”, коли одна VM впливає на продуктивність іншої.

Для усунення цієї проблеми розроблено метод ізоляції ресурсів vSwitch на основі CPU-циклів, який виконує ізоляцію CPU-циклів між VM на основі математичного зв’язку між смугою пропускання й обчислювальними витратами. Ідея полягає в тому, що для заданої швидкості V (Гбіт/с) можна наперед визначити кількість процесорних циклів B (циклів/с), потрібних для оброблення цього обсягу трафіку, і обмежити доступ VM до CPU згідно з цим бюджетом.

Залежність між необхідною кількістю CPU-циклів і пропускною здатністю лінійна в робочому діапазоні навантажень DPDK:

$$B = k \times V + B_0 \tag{3.1}$$

де k – коефіцієнт витрат циклів на 1 Гбіт/с трафіку; B_0 – постійна частина (сервісні операції ingress/egress, класифікація, облік).

Для процесорів сімейства Intel Xeon Silver 4314 із тактовою частотою 2.4 ГГц у ході попередніх експериментів було встановлено емпіричну залежність

між пропускною здатністю мережевого каналу та кількістю процесорних циклів, необхідних для її забезпечення у середовищі OVS-DPDK. Для кожного трафіку в системі OVS-DPDK усі пакети обробляються у користувацькому просторі через послідовність етапів приймання (ingress), класифікації, пересилання (forwarding) та передавання (egress). Кожен із цих етапів споживає певну кількість обчислювальних інструкцій, що безпосередньо перетворюється на витрати CPU-циклів. Таким чином, кількість циклів на секунду, яку необхідно виділити процесору для оброблення певного обсягу мережевого трафіку, визначається добутком частоти оброблення пакетів на середню кількість інструкцій, що виконуються для одного пакета.

У результаті численних вимірювань, проведених за допомогою інструментів perf stat та внутрішніх лічильників OVS-DPDK, виявлено майже лінійний характер залежності між обсягом трафіку та витратами процесорних циклів. Для Xeon Silver 4314 визначено коефіцієнт перетворення $k \approx 0.2 \times 10^9$ циклів на 1 Гбіт/с. Це означає, що для підтримки сталої швидкості передавання даних 1 Гбіт/с комутатору необхідно приблизно 2×10^8 циклів за секунду активного часу одного ядра, а для 2 Гбіт/с - 4×10^8 циклів за секунду. Решта частоти ядра залишається доступною для службових операцій операційної системи та керуючих потоків DPDK. Виявлена пропорційність пояснюється тим, що при зростанні швидкості обміну даними кількість пакетів, які потрапляють до черг OVS, збільшується пропорційно, а час виконання кожного циклу обробки пакета (polling, lookup, enqueue/dequeue, TX) залишається майже сталим. На відміну від апаратних комутаторів, де продуктивність лінійно обмежується пропускною здатністю ASIC, у програмному OVS-DPDK основним вузьким місцем є саме кількість доступних процесорних інструкцій за одиницю часу. Отже, для кожного ядра існує граничне значення пропускної здатності, за якої використовується 100 % його обчислювальних циклів. Для Xeon 4314 це значення становить близько 10–11 Гбіт/с на одне ядро при 2.4 ГГц, що узгоджується з отриманим коефіцієнтом k. Практична корисність цієї залежності полягає в тому, що вона дозволяє попередньо розрахувати потребу у CPU-ресурсах для кожної VM залежно від обсягу придбаної нею смуги пропускання.

Знаючи k і пропускну здатність інтерфейсу $V_{purchased}$, адміністратор може точно визначити обсяг циклів $B_{alloc}=k\times V_{purchased}$, який має бути виділений конкретній VM із пулу ядер. Завдяки цьому забезпечується строгий баланс між загальною кількістю доступних циклів $C_{IO\text{-}cores}$ та сумою обчислювальних вимог усіх активних VM .

Наприклад, якщо на сервері доступні 8 ядер по 2.4 ГГц (загалом 19.2×10^9 циклів/с), то обслуговування чотирьох VM із купленою смугою 1, 1, 2 та 2 Гбіт/с потребує відповідно 0.2, 0.2, 0.4 та 0.4 мільярда циклів за секунду, тобто лише 1.2×10^9 циклів/с менше ніж 7 % від наявного ресурсу. Такий запас дозволяє реалізувати політику MIN–MAX, коли надлишкові 18×10^9 циклів/с можуть динамічно розподілятися між VM , що активно передають дані.

Таким чином, емпірично встановлене співвідношення $k\approx 0.2\times 10^9$ циклів/Гбіт/с стало основним аналітичним параметром планування CPU-ресурсів, оскільки воно перетворює вимоги з боку мережевого QoS у конкретні значення обчислювальних ресурсів, які можна ізолювати й контролювати в межах віртуального комутатора.

3.1.2 Правила розміщення VM на фізичному сервері

У системах віртуалізації, де десятки або навіть сотні віртуальних машин розгортаються на одному фізичному сервері, баланс між доступними обчислювальними й мережевими ресурсами є критичним фактором стабільності. Коли кількість віртуальних машин перевищує реальні можливості мережевого адаптера або процесора, виникає стан ресурсного дефіциту - ядра перевантажуються, затримки ростуть, а пропускна здатність стає непередбачуваною. Саме для уникнення таких ситуацій метод ізоляції ресурсів vSwitch на основі CPU-циклів запроваджує два аналітичні правила, які формалізують умову стійкої роботи віртуального комутатора Open vSwitch із прискоренням DPDK.

Перша умова стосується мережевої підсистеми і вимагає, щоб сума придбаних смуг пропускання всіх віртуальних машин не перевищувала фізичної

пропускної здатності мережевого інтерфейсу. Це означає, що сумарний запит на трафік, який генерується усіма орендарями, повинен залишатися в межах того, що здатен обробити сам мережевий адаптер:

$$\sum_{i=1}^{N} V^{i}_{\text{purchased}} \leq V_{NIC} \tag{3.2}$$

Тут $V^{i}_{\text{purchased}}$ - гарантована смуга пропускання, замовлена і-ою віртуальною машиною, а V_{NIC} - максимальна апаратна швидкість передавання даних через фізичний порт.

Порушення цієї умови означає, що комутатор прийматиме більше трафіку, ніж може фізично передати, унаслідок чого почнуть переповнюватися черги приймання DPDK, зростатиме затримка, і з'являться втрати пакетів. Особливо небезпечним це стає в середовищах із сервісами реального часу, де навіть короткочасне перевищення ліміту NIC може спричинити деградацію всієї системи. Ця вимога також визначає граничну щільність розміщення VM : якщо, наприклад, у системі використовується двопортовий мережевий адаптер 2 × 1 Гбіт/с, то сума гарантованих смуг усіх віртуальних машин не повинна перевищувати 2 Гбіт/с. VM , що мають змінні навантаження, можуть працювати вище цього рівня лише тимчасово й лише за наявності невикористаної смуги інших орендарів, але на рівні базової конфігурації система завжди повинна відповідати цьому обмеженню.

Друга умова пов'язана з обчислювальними ресурсами і визначає межу навантаження на ядра процесора, виділені для IO-операцій. Кожна віртуальна машина споживає певну кількість CPU-циклів, необхідних для оброблення її мережевого трафіку. На підставі емпіричної моделі «bandwidth → CPU-cycles» для кожної VM визначається параметр - кількість циклів, що мають бути зарезервовані для забезпечення замовленої пропускної здатності. Щоб система працювала стабільно, сума всіх цих значень не може перевищувати загальний ресурс процесора, виділений для оброблення IO:

$$\sum_{i=1}^{N} B^{i}_{\text{alloc}} \leq B_{\text{IO}-\text{cores}} \tag{3.3}$$

Тут $B_{\mathrm{IO-cores}}$ - загальна кількість циклів за секунду, яку можуть надати ядра, закріплені під PMD-потоки DPDK. Для процесора з частотою 2.4 ГГц одне ядро здатне виконати 2.4×10^9 інструкцій за секунду, а якщо для IO виділено, скажімо, 8 ядер, то граничне значення $B_{\mathrm{IO-cores}}$ =19.2×10^9 циклів/с. Якщо сума B^i_{alloc} перевищує це значення, навіть оптимальні механізми пріоритезації не врятують ситуацію - процесор фізично не зможе забезпечити необхідну швидкість обробки пакетів. У цьому випадку виникають затримки, а токени в механізмі токен-бакета на основі CPU-циклів починають накопичуватись із відставанням, що призводить до лавиноподібного зростання латентності.

Поєднання цих двох нерівностей утворює область стабільності системи, у межах ізоляції ресурсів vSwitch на основі CPU-циклів гарантує детерміновану роботу без перевантажень. Якщо обидві умови виконуються, тоді мережевий комутатор функціонує у збалансованому стані: кількість пакетів, які надходять від усіх VM , дорівнює кількості пакетів, які система встигає обробити й передати за доступний час одного такту. У цьому стані можливе точне планування QoS. Система може призначати пріоритети, проводити ізоляцію циклів і навіть динамічно змінювати розподіл ресурсів між орендарями, не порушуючи загальної рівноваги. Якщо хоча б одна з умов не виконується, виникає режим перевантаження (overload). У цьому випадку CPU працює з повним навантаженням, а планувальники DPDK починають відкладати обробку нових пакетів. Унаслідок цього затримки між ingress і egress етапами стрімко зростають, а черги на портах перевищують допустиму глибину. У гіршому випадку це призводить до збоїв у класифікації потоків і неконтрольованого скидання пакетів.

Звідси випливає принцип, який лежить в основі ізоляції ресурсів vSwitch на основі CPU-циклів. Система не допускає перепродавання (overprovision) ресурсів. Кожна віртуальна машина отримує стільки процесорного часу, скільки фізично доступно на ядрах, а кількість активних орендарів завжди корелює з реальною кількістю обчислювальних блоків. Такий підхід усуває випадкові коливання продуктивності, дозволяє точно передбачити затримку для кожної VM і робить політику QoS стійкою навіть під піковими навантаженнями.

3.1.3 Механізм справедливого розподілу CPU-циклів на основі токенів

Механізм справедливого розподілу CPU-циклів на основі токенів (токен-бакет) є центральним елементом методу, що забезпечує динамічний контроль і ізоляцію CPU-ресурсів на рівні кожної віртуальної машини. Його основна ідея полягає у тому, щоб відмовитися від традиційного представлення токенів у вигляді обсягів переданих даних бітів чи пакетів і перейти до більш фундаментального показника кількості обчислювальних циклів процесора, необхідних для їх оброблення. Такий підхід робить можливим безпосереднє регулювання пропускної здатності через контроль використання CPU, що є єдиним дійсно обмежувальним ресурсом у програмному комутаторі OVS-DPDK.

Для кожної віртуальної машини створюється власний токен-бакет, у якому токени відповідають кількості доступних CPU-циклів, що можуть бути витрачені на пересилання її трафіку. Кожна VM отримує швидкість генерації токенів, пропорційну до розрахованого для неї бюджету CPU - B^i_{alloc}, який визначається на основі моделі взаємозв'язку між пропускною здатністю та кількістю процесорних циклів. Це означає, що якщо VM отримала смугу пропускання 2 Гбіт/с, то система на кожну секунду генерує для неї обсяг токенів, еквівалентний приблизно 0.4×10^9 CPU-циклів. У момент, коли токени вичерпуються, віртуальна машина втрачає право виконувати операції оброблення пакетів, доки у бакеті не накопичиться нова порція токенів, що забезпечує жорстку ізоляцію між орендарями. Якщо реальне навантаження на певну VM є нижчим за передбачене, її токен-бакет починає накопичувати невикористані токени. У традиційних QoS-механізмах така ситуація просто ігнорується, але в методі ізоляції ресурсів vSwitch на основі CPU-циклів вона використовується як можливість підвищення ефективності системи. Невикористані цикли не зникають - вони враховуються як надлишковий резерв і згодом перерозподіляються між активними віртуальними машинами, які наразі потребують додаткових ресурсів. Для цього у модулі керування системи ізоляції ресурсів vSwitch на основі CPU-циклів реалізовано періодичний виклик функції

оновлення токенів, яка підраховує суму переповнених циклів у всіх бакетах та пропорційно розподіляє їх відповідно до вагових коефіцієнтів орендарів. Така процедура дозволяє динамічно реагувати на зміну навантаження, забезпечуючи високу ефективність використання CPU при збереженні гарантій ізоляції. Для гнучкого керування цим процесом використовується політика MIN–MAX bandwidth guarantee, що поєднує два рівні обслуговування. Перший рівень гарантує кожній VM мінімальну кількість процесорних циклів B^i_{alloc}, достатню для забезпечення придбаної смуги V_{MIN}. Другий рівень дозволяє віртуальній машині тимчасово отримати додаткові ресурси за рахунок вільних CPU-циклів B_{idle}, якщо вони є доступними, і таким чином збільшити пропускну здатність до рівня V_{MAX}. Взаємозв'язок між цими параметрами описується рівнянням

$$B^i_{max} = B^i_{\text{alloc}} + w_i \cdot B_{idle}, \qquad \sum_{i=1}^{N} w_i = 1 \tag{3.3}$$

де w_i - ваговий коефіцієнт, що визначає частку участі i-ої VM у розподілі вільних ресурсів. Значення ваги встановлюється адміністратором або системою автоматично, залежно від класу обслуговування чи пріоритету орендаря.

Таким чином, токен-бакет кожної VM не є статичним. Він відображає поточний баланс між гарантованими і доступними додатковими ресурсами. Якщо, наприклад, на CPU частотою 2.2 ГГц після забезпечення мінімальних смуг для чотирьох VM залишається 1×10^9 циклів/с невикористаного часу, то цей резерв розподіляється між машинами пропорційно їхнім вагам (див. рисунок 3.1).

	VM1	VM2	VM3	VM4
weight	1	1	2	2
purchased bandwidth	1 Gbps	1 Gbps	2 Gbps	2 Gbps
MIN(cycles/s)	0.2G	0.2G	0.4G	0.4G
MAX(cycles/s)	0.2G + 0.16G	0.2G + 0.16G	0.4G + 0.32G	0.4G + 0.32G

Рисунок 3.1 – Приклад механізму токен-бакета на основі CPU-циклів

Дві VM можуть отримати по 0.36 G cycles/s, а ще дві - по 0.72 G cycles/s. Це дозволяє досягти максимальної сумарної пропускної здатності без порушення балансу і без зменшення гарантій для інших користувачів.

Перевага механізму токен-бакета на основі CPU-циклів полягає у тому, що він здійснює керування пропускною здатністю не на основі кількості переданих байтів, а виходячи з обчислювальних можливостей процесора. Завдяки цьому система автоматично адаптується до змінних умов - наприклад, при появі більш складних правил класифікації або збільшенні розміру пакетів. Поки VM не вичерпала свій ліміт CPU-циклів, вона може передавати пакети з будь-якою швидкістю, а після його перевищення негайно обмежується. Таким чином, механізм встановлює пряму відповідність між обсягом доступних процесорних інструкцій і гарантованою мережею пропускною здатністю, що робить систему детермінованою та стійкою до перевантажень. У результаті токен-бакет на основі CPU-циклів виступає своєрідним мостом між фізичною продуктивністю сервера й рівнем мережевих гарантій. Він забезпечує точне, гнучке та адаптивне керування ресурсами. При низькому навантаженні дозволяє перерозподіляти вільні цикли між активними VM , а при високому строго дотримується меж виділених бюджетів, запобігаючи колапсу системи. Саме тому цей механізм є ключовим компонентом архітектури ізоляції ресурсів vSwitch на основі CPU-циклів, яка забезпечує не лише справедливість, але й повну ізоляцію CPU-ресурсів між орендарями у програмному комутаторі Open vSwitch із DPDK.

3.1.4 Ієрархічне планування на основі груп процесів

Механізм планування на основі груп процесів (ієрархічного батч-планування) є другим ключовим компонентом стратегії ізоляції ресурсів vSwitch на основі CPU-циклів і призначений для управління затримками виконання мережевих операцій між віртуальними машинами, що мають різні вимоги до якості обслуговування. Ієрархічне батч-планування - це коли процеси не кидають у одну загальну чергу, а спершу об’єднують у групи, і планувальник працює з групами як з єдиними одиницями, а вже потім розглядає процеси всередині груп.

Якщо механізм токен-бакета на основі CPU-циклів забезпечує справедливий розподіл процесорних циклів, то механізм ієрархічного батч-планування дозволяє досягти диференціації затримок. Тобто створює пріоритети між класами трафіку, щоб більш чутливі до затримки сервіси отримували доступ до CPU раніше за ті, які можуть працювати з більшими інтервалами часу.

Ієрархічна модель планування базується на ідеї класів пріоритетів. Усі віртуальні машини групуються у певну кількість класів, що відповідають різним рівням пріоритету від 1 (найвищий) до 8 (найнижчий). Класи визначаються постачальником хмарних послуг (CSP) під час створення VM, виходячи з вимог сервісів, які вони підтримують. Так, наприклад, служби відеоконференцій, системи онлайн-ігрових серверів або веб-додатки з низькою толерантністю до затримки призначаються до високих класів пріоритету, тоді як аналітичні системи, резервне копіювання чи фонові процеси - до нижчих.

Для кожного класу створюється черга готовності (ready queue), куди потрапляють віртуальні машини, що мають позитивний баланс токенів у механізмі токен-бакета на основі CPU-циклів, тобто ті, які мають право виконати операцію пересилання пакетів. Віртуальні машини, у яких токени вичерпано або баланс став від’ємним, переміщуються до черги очікування (waiting queue) і залишаються там, доки не отримають нову порцію токенів. Такий підхід гарантує, що до оброблення потрапляє лише той трафік, який має обчислювальні ресурси, виділені згідно з механізму токен-бакета на основі CPU-циклів, а порядок його виконання регулюється механізмом ієрархічного батч-планування.

Кожне ядро процесора, яке виконує роль PMD-потоку, обробляє пакети послідовно, опитуючи черги класів у порядку спадання пріоритетів. Всередині кожного класу використовується політика FIFO, що забезпечує справедливість для машин з однаковим пріоритетом. Таким чином, коли ядро переходить до виконання черг, воно спочатку обслуговує всі завдання з класу 1, далі клас 2, і лише коли черги з вищими пріоритетами спорожніють, приступає до класів із меншими пріоритетами. Цей підхід дозволяє створити ієрархію доступу до CPU, за якої час очікування кожної віртуальної машини прямо залежить від її класу.

Щоб оцінити затримку виконання в межах кожного класу, розглядається кількість віртуальних машин N_i у кожній черзі та час оброблення одного батча пакетів c. У найпростішому випадку, якщо всі черги мають однакову кількість машин, а пріоритети - однакову вагу, затримка для кожної черги буде пропорційна її номеру в ієрархії. Проте в реальній системі машини з вищими пріоритетами мають привілейоване право на виконання кількох послідовних батчів перед тим, як планувальник перейде до нижчих класів. Цей ефект описується коефіцієнтами $k_i \geq 1$, які показують, скільки разів задачі класу i можуть бути виконані, поки черги нижчого рівня очікують. Таким чином, найгірша (максимальна) затримка для класів визначається виразом:

$$\{N_1 c,\ (N_1 k_1 + N_2)c,\ (N_1 k_1 + N_2 k_2 + \cdots + N_7 k_7 + N_8)c\} \tag{3.4}$$

де N_i - кількість віртуальних машин у класі i, а c - час обробки одного батча.

Ця формула означає, що навіть у найгіршому випадку, коли всі черги заповнені, затримка виконання для класів залишається обмеженою і передбачуваною. Для віртуальних машин із найвищим пріоритетом (клас 1) затримка дорівнює лише часу оброблення їхніх власних батчів, тоді як для нижчих класів вона зростає поступово залежно від кількості активних машин у чергах із вищими пріоритетами. Таким чином, механізм забезпечує градуйовану гарантію затримки (hierarchical latency guarantee), що дозволяє CSP реалізовувати більш гнучкі політики SLA для різних категорій клієнтів.

Одним із найважливіших аспектів ієрархічного батч-планування є здатність уникати голодування (starvation) - ситуації, коли низькопріоритетні віртуальні машини можуть залишатися невиконаними надто довго через постійне завантаження високопріоритетних черг. Проблема вирішується за допомогою механізму динамічного коригування пріоритету. Якщо віртуальна машина не генерує трафіку впродовж кількох циклів опитування, її пріоритет поступово знижується. Це дозволяє системі скоротити час холостого опитування та зменшити втрати ресурсів на неактивні потоки. Коли ж така VM знову починає передавання пакетів, її початковий пріоритет миттєво відновлюється, і вона повертається у свій клас. Така адаптивна поведінка гарантує, що навіть у

системах із великою кількістю високопріоритетних орендарів нижчі рівні все одно отримують свій мінімальний обсяг CPU-часу.

Крім цього, механізм ієрархічного батч-планування забезпечує сталість у межах одного класу. Завдяки політиці FIFO усі VM з однаковим пріоритетом отримують приблизно однакову затримку. Це особливо важливо для сервісів, що масштабуються горизонтально (наприклад, кілька інстансів вебсервера чи бази даних), де однакова затримка між вузлами гарантує узгодженість у часі.

З технічної точки зору, реалізація ієрархічного батч-планування в OVS-DPDK не потребує радикальної зміни архітектури. Кожен PMD-потік підтримує окремі структури черг для свого набору віртуальних машин. Черги організовані у вигляді багаторівневого дерева, де кореневий рівень відповідає класам пріоритетів, а підрівні - конкретним чергам віртуальних машин. Планувальник циклічно обходить ці черги, починаючи з найвищого рівня. Якщо черги вищого класу порожні, обробка автоматично переходить до нижчих. Така архітектура дозволяє одночасно забезпечити високу ефективність і передбачувану затримку без використання блокувань і конкурентного доступу між потоками, оскільки кожне ядро обслуговує власний набір черг.

У підсумку механізм ієрархічного батч-планування відіграє роль механізму диференціації часу доступу до CPU в межах одного фізичного сервера. Він не тільки встановлює пріоритети для різних категорій трафіку, але й забезпечує стабільну, математично описану межу найгіршої затримки, що є необхідною умовою для підтримки гарантованої якості обслуговування. Разом із механізмом токен-бакета на основі CPU-циклів цей механізм формує цілісну систему керування ресурсами, у якій обчислювальні цикли процесора розподіляються не лише справедливо, але й із урахуванням реальних потреб орендарів у швидкодії та затримці.

3.2 Реалізація методу ізоляції ресурсів vSwitch

Відповідно до концепції, розробленої у попередньому підрозділі, реалізація стратегії була виконана на базі платформи OVS з підтримкою DPDK. Саме ця

комбінація забезпечує високопродуктивну обробку мережевого трафіку в користувацькому просторі, використовуючи модель PMD, коли оброблення пакетів відбувається без переривань, у циклі безпосереднього опитування. Реалізація методу вимагала втручання у внутрішню логіку OVS-DPDK - зокрема, у головний цикл PMD-потоку та політику обмеження швидкості на віртуальних портах. Архітектурні зміни показано на рисунку 3.2.

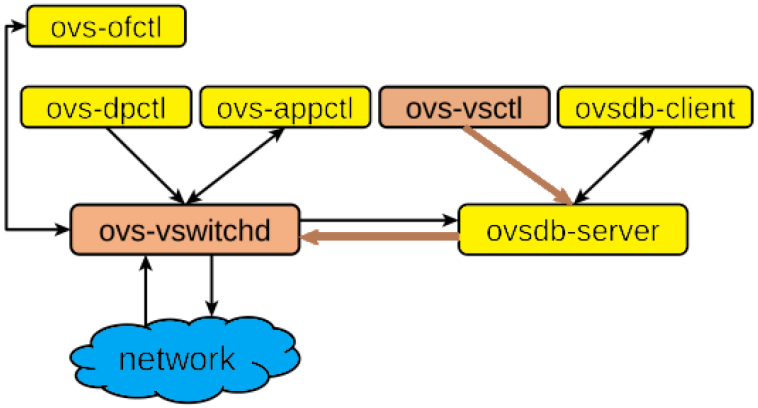

Рисунок 3.2 – Архітектурні зміни Open vSwitch

Основна модифікація стосувалася головного циклу PMD-потоку, який у звичайному OVS працює у режимі послідовного опитування всіх черг без розрізнення пріоритетів. У реалізованій версії цей режим було замінено на ієрархічне батч-планування. Відтепер кожен PMD-потік має власний модуль батч-планування, який керує кількома віртуальними машинами, закріпленими за цим потоком, і формує для них окремі черги готовності (ready queues). PMD-потік не має доступу до черг інших потоків, що повністю усуває проблему конкурентного доступу до спільних структур пам’яті й дозволяє уникнути блокувань. Під час виконання чергової ітерації головного циклу ієрархічне батч-планування вибирає віртуальну машину з найвищим пріоритетом серед тих, що мають позитивний баланс токенів, і запускає для неї процедуру batch I/O processing - тобто пакетну обробку трафіку.

Другим етапом інтеграції стало впровадження механізму токен-бакета на основі CPU-циклів, який замінив стандартну політику обмеження швидкості srTCM, реалізовану в модулі ovs-vswitchd. Тепер перед кожним викликом batch-

процесингу PMD-потік звертається до токен-бакета для перевірки, чи має поточна VM достатній обсяг CPU-циклів для виконання чергової порції обробки пакетів. Якщо баланс токенів від'ємний, обробка пакета відкладається до наступного циклу оновлення токенів. Таким чином, механізм токен-бакета на основі CPU-циклів фактично стає механізмом апаратно-незалежного контролю використання CPU, який обмежує швидкість пересилання даних не за кількістю пакетів чи байтів, а за кількістю процесорних інструкцій, доступних кожній віртуальній машині.

Для спрощення керування всі параметри методу ізоляції ресурсів vSwitch на основі CPU-циклів інтегровані до системи керування ovs-vsctl. Було додано два нових типи команд, що дозволяють адміністратору або CSP налаштовувати параметри обмеження CPU-циклів і пріоритети планування безпосередньо з командного рядка. Команда `ovs-vsctl set interface vhost-user-1 ingress_policing_cpucycles=10000` призначає для віртуального інтерфейсу vhost-user-1, пов'язаного з VM1, ліміт у 10 000 циклів/с, що визначає її гарантований бюджет CPU. Аналогічно, команда `ovs-vsctl set interface vhost-user-1 options:priority=1` встановлює для VM1 найвищий пріоритет у системі ієрархічного батч-планування.

Такі параметри зберігаються у базі конфігурацій OVSDB і активуються без необхідності перезапуску служби ovs-vswitchd. Це забезпечує динамічність конфігурації. CSP може змінювати політики QoS «на льоту» залежно від поточного навантаження чи вимог SLA.

Під час реалізації особливу увагу було приділено оптимізації вимірювання використання CPU. Найскладнішою частиною стала необхідність частих обчислень фактичної кількості процесорних циклів, витрачених на пересилання пакетів, без суттєвого впливу на пропускну здатність. Використання традиційних інструментів (perf, getrusage) є занадто повільним для високочастотних потоків DPDK. Для вирішення цієї проблеми застосовано апаратну інструкцію rdtsc (Read Time-Stamp Counter), яка зчитує кількість тактів процесора з моменту завантаження системи. Ця інструкція виконується напряму на рівні регістрів і має практично нульову затримку, тому її можна безпечно

використовувати в критичному циклі обробки даних. За допомогою rdtsc система точно визначає, скільки циклів було витрачено на оброблення певної порції пакетів, і оновлює баланс токенів кожної VM у реальному часі.

Додаткове навантаження створює й підтримка структур черг у модулі ієрархічного батч-планування. Для уникнення споживання зайвих обчислювальних ресурсів, що могло б знизити продуктивність, у методі ізоляції ресурсів vSwitch на основі CPU-циклів передбачено менеджер-потік, який відповідає за оновлення стану черг та перерозподіл токенів між віртуальними машинами. Замість постійного активного опитування (busy polling), цей потік працює у подієвому режимі, періодично прокидаючись із певним інтервалом часу. Цей інтервал встановлено рівним 40 мікросекундам, що приблизно відповідає середньому часу виконання одного циклу пакетної обробки (batch I/O processing). Такий вибір є компромісом між точністю й ефективністю: зменшення інтервалу покращує точність оновлення токенів, але підвищує навантаження на CPU, тоді як збільшення інтервалу зменшує накладні витрати, проте може спричинити короткочасні коливання затримки. У тестових конфігураціях цей баланс забезпечив оптимальне співвідношення між точністю керування і споживанням ресурсів.

Таким чином, реалізація методу ізоляції ресурсів vSwitch на основі CPU-циклів у OVS-DPDK демонструє приклад легкої, модульної інтеграції механізмів контролю CPU у вже існуючу архітектуру комутатора. Завдяки застосуванню інструкції rdtsc для обліку циклів, автономному механізму ієрархічного батч-планування для планування черг і додатковим CLI-командам для конфігурації, система отримала можливість ізолювати обчислювальні ресурси між віртуальними машинами з мінімальними витратами. Отримане рішення повністю сумісне з іншими компонентами OVS, не змінює логіку передачі пакетів і може бути застосоване до будь-яких типів віртуальних портів. Практична перевірка ефективності та продуктивності реалізованого механізму проводиться в наступному підрозділі, де буде проаналізовано вплив методу ізоляції ресурсів vSwitch на основі CPU-циклів на пропускну здатність, затримку та споживання CPU у різних сценаріях навантаження.

3.3 Оцінювання ефективності методу ізоляції ресурсів vSwitch

Метою експериментального оцінювання є перевірка того, наскільки метод ізоляції ресурсів vSwitch на основі CPU-циклів забезпечує гарантії пропускної здатності та затримки для VM у порівнянні з наявною в OVS-DPDK політикою ovs-ingress-policy, а також наскільки точно працюють складові лімітування за CPU-циклами ієрархічне батч-планування. Стенд, апаратні характеристики та налаштування платформи наведено раніше. Ключовий сценарій - чотири VM із придбаною смугою 4, 4, 1 та 1 Гбіт/с, закріплені за одним виділеним IO-ядром. Для калібрування використано емпіричну модель зв'язку «смуга ↔ CPU-цикли» описану в розділі 2 згідно з якою для сценарію 1 (4 Гбіт/с, 1024-байтові пакети, один потік) потрібно близько 0.792 G cycles/s (≈36% ядра), а для сценарію 2 (1 Гбіт/с, 1024 байти, один потік) - 0.198 G cycles/s (≈9%). Далі трафік змінюється по етапах. Спочатку працюють лише VM1 і VM3 з 1024-байтовими пакетами. Потім до них приєднуються VM2 і VM4 із 64-байтовими пакетами. На фінальному відрізку всі, окрім VM1, переходять на 64 байти. Це дозволяє явно проявити різницю між поведінкою ovs-ingress-policy, механізм токен-бакета на основі CPU-циклів-strict (лише B_{alloc}) та механізм токен-бакета на основі CPU-циклів-MINMAX (повне використання B_{idle}).

Під ovs-ingress-policy VM1 і VM3 утримують свою смугу лише до моменту, коли VM2 і VM4 починають генерувати 64-байтові пакети. Невеликі кадри збільшують частоту пакетів, і послідовне опитування у PMD спричиняє перерозподіл CPU на користь «шумних» сусідів. Як наслідок, смуга VM1 просідає приблизно на 12%, тоді як VM3 формально зберігає замовлену смугу, але її CPU-бюджет стає нестабільним. Ефект детально видно на рисунку 3.3 та 3.4, де зафіксовано як пропускну здатність, так і завантаження CPU для кожної VM .

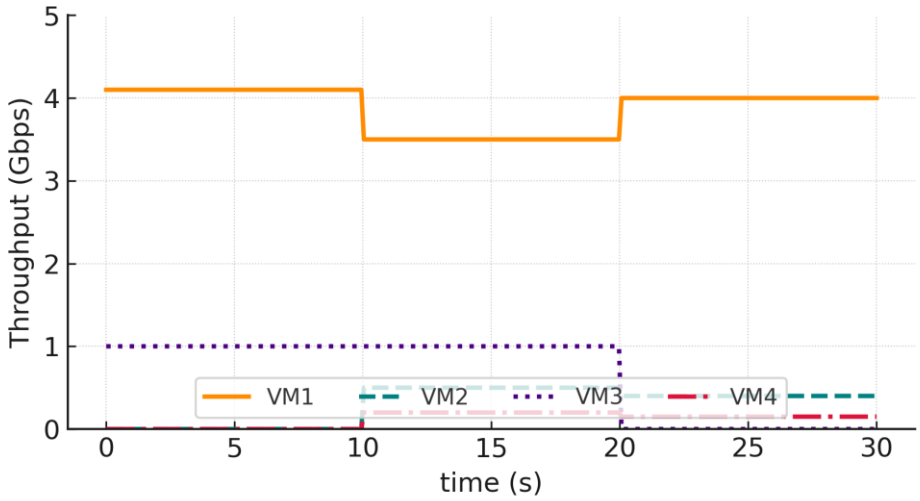

Рисунок 3.3 – Пропускна здатність мережі віртуальної машини при ovs-ingress-policy

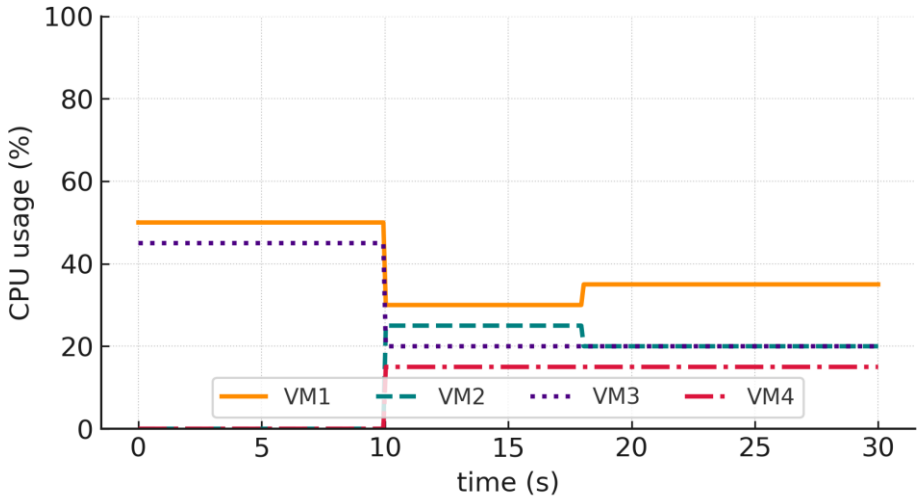

Рисунок 3.4 – Використання процесора при ovs-ingress-policy

Цей самий механізм пояснює «парадоксальне» збільшення смуги VM1 наприкінці експерименту. Коли VM3 переходить на 64-байтові кадри, кількість пакет-ітерацій/с зростає для всіх VM , і VM1 встигає відправити більше пакетів до того, як спрацює BPS-ліміт OVS; водночас це робить розподіл CPU непередбачуваним і не дозволяє гарантувати SLA.

У режимі strict кожна VM отримує рівно той бюджет CPU-циклів, що відповідає її замовленій смузі (36%, 36%, 9%, 9%). На рисунку 3.5 та 3.6 видно, що смуга VM1 стабільно тримається на рівні 4 Гбіт/с за сталого споживання ≈36% CPU; зміна розміру пакета в VM3 на 64 байти наприкінці знижує її власну

смугу, але не впливає на інших, оскільки «продавити» додаткові цикли поза B_{alloc} неможливо.

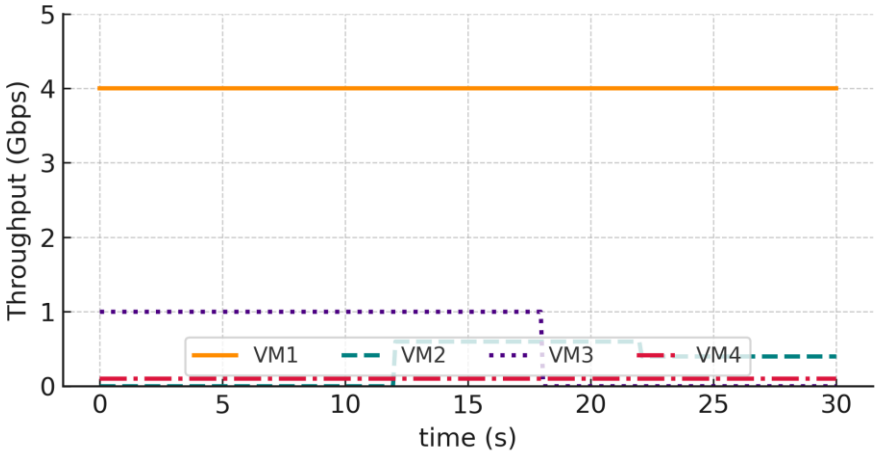

Рисунок 3.5 – Пропускна здатність мережі віртуальної машини при strict режимі

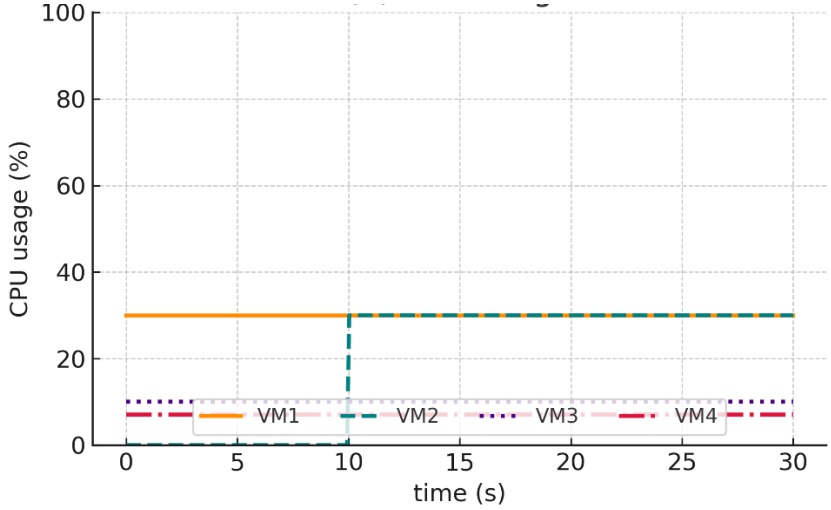

Рисунок 3.6 – Використання процесора при strict режимі

Поведінка VM2/VM4 із короткими кадрами не порушує SLA інших VM . Шум локалізовано за рахунок ізоляції CPU-циклів. Видима «майже нульова» смуга для VM3/VM4 на фінальному етапі не є starvation у класичному розумінні. Це очікуваний наслідок того, що синтетична «агресія» 64-байтовими кадрами не підкріплена виділеними циклами. Метод ізоляції ресурсів vSwitch на основі CPU-циклів, по суті, змушує такі VM впиратися у власний ліміт, а не у спільний ресурс.

Недолік strict - потенційне простоювання CPU, коли активних запитів менше за доступні цикли. На рисунку 3.6 у перші 10 секунд видно до ~75% невикористаного часу ядра. Активувавши механізм токен-бакета на основі CPU-циклів-MINMAX, система перерозподіляє B_{idle} і «переливи» токенів до активних VM за вагами, що дозволяє VM1 і VM3 піднятися до ~6.3 та ~2.1 Гбіт/с відповідно в «тихий» період, не порушуючи гарантій (див. рисунок 3.7 та 3.8). Таким чином, MIN–MAX-політика знімає проблему марнування CPU, зберігаючи ізоляцію і передбачуваність.

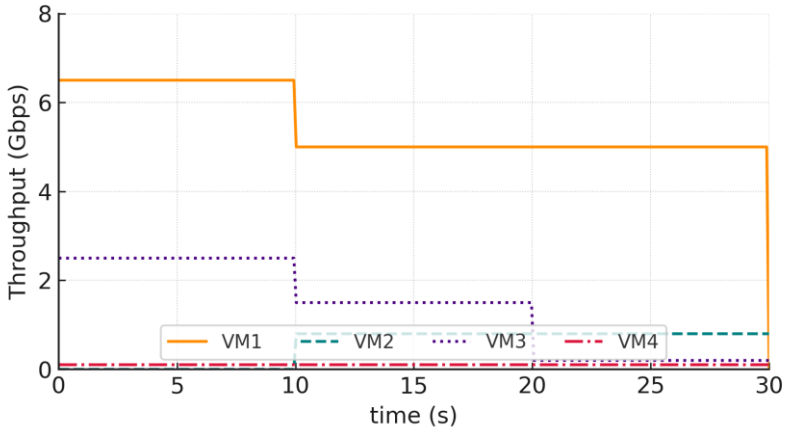

Рисунок 3.7 – Пропускна здатність мережі віртуальної машини при MIN–MAX режимі

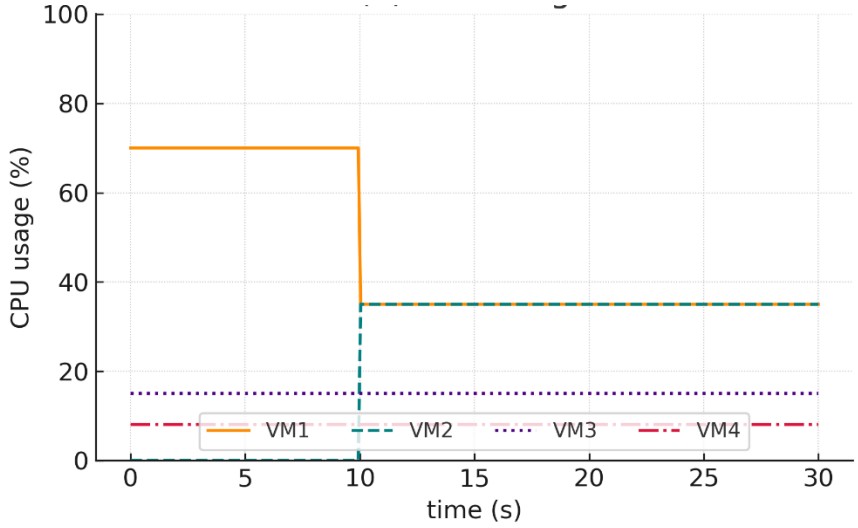

Рисунок 3.8 – Використання процесора при MIN–MAX режимі

Точність механізму токен-бакета на основі CPU-циклів оцінювалась у двох площинах: стосовно коливань розміру пакета та масштабування по кількості VM /ядер. Для змішаних розмірів кадрів із фіксованим середнім, за умови, що середній розмір параметризовано згідно з моделлю, відхилення здебільшого позитивні або близькі до нуля - переважно у межах (−2%, +3%), що означає недопоставки майже не трапляється (див. рисунок 3.9).

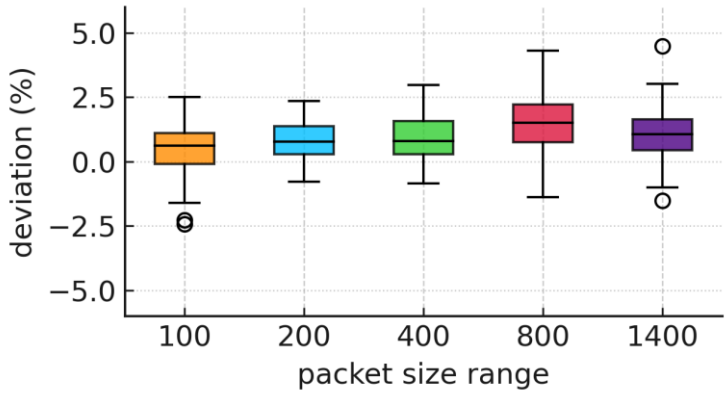

Рисунок 3.9 – Оцінка точності: Пакети різного розміру

За збільшення кількості VM і рознесення по NUMA-вузлах похибка зростає, але залишається прийнятною для софт-форвардингу. При одному IO-ядрі - (−2%, +4%), при двох - приблизно вдвічі ширше, (−3%, +6%), що корелює з додатковою невизначеністю конкуренції між потоками (див. рисунок 3.10).

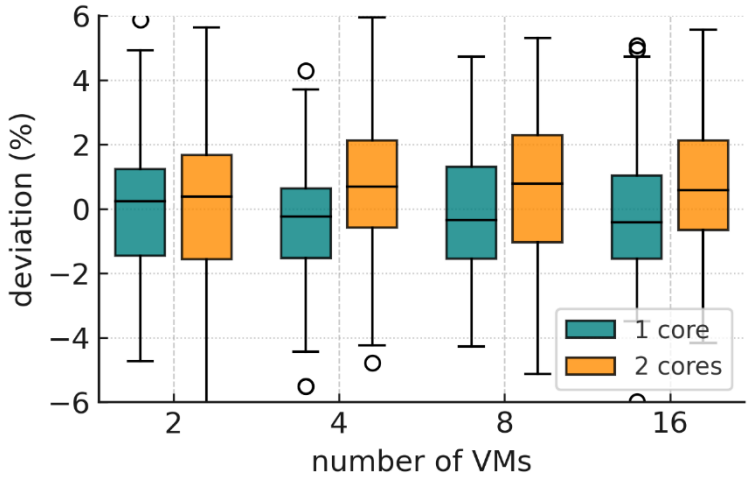

Рисунок 3.10 – Оцінка точності: Розгортання

Для механізму ієрархічного батч-планування задіяно 16 VM у 4 пріоритетах. Розподіли затримки чітко розшаровуються відносно «середнього» рівня токен-бакета на основі CPU-циклів-only. Пріоритети 1–2 дають затримку нижчу за середню і з меншою дисперсією, 3–4 - вищу і більш «розкидану» (див. рисунок 3.11).

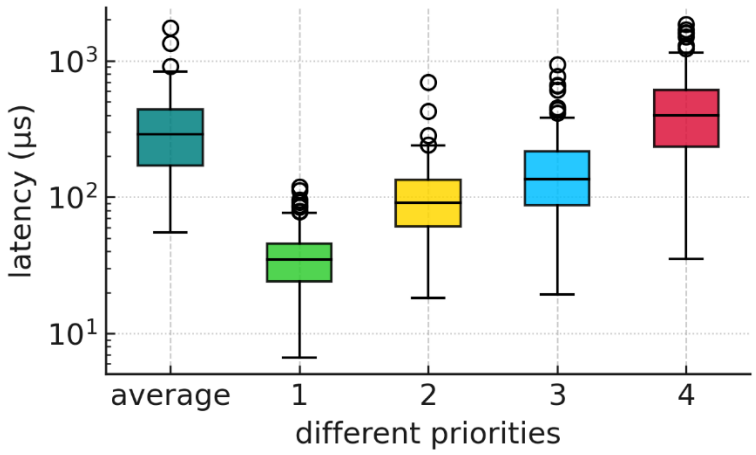

Рисунок 3.11 – Оцінка точності: Ієрархічна затримка

Це підтверджує практичну придатність механізму ієрархічного батч-планування для побудови SLA з диференційованими класами затримки.

Приклад прикладних навантажень поєднує «чутливі до затримки» (Nginx) і «чутливі до смуги» (FTP) сервіси на спільному сервері. Для FTP під ovs-ingress-policy одночасний «вибух» дрібних запитів Nginx у вікні 30–70 с знижує сумарну смугу приблизно на 11%, тоді як механізм токен-бакета на основі CPU-циклів утримує її на рівні замовленої (див. рисунок 3.12).

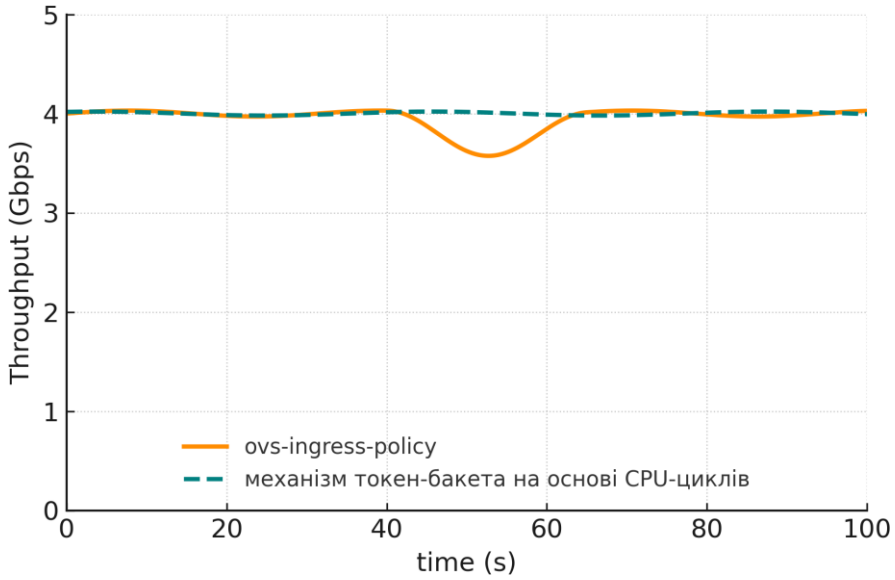

Рисунок 3.12 – Результати продуктивності програм: пропускна здатність FTP

Для Nginx розподіли часу відповіді показують: порівняно з «нативним» виконанням ovs-ingress-policy подвоює затримки, механізм токен-бакета на основі CPU-циклів зменшує додаткову затримку приблизно на 50% за рахунок пропуску «нульових» портів, а механізм токен-бакета на основі CPU-циклів в парі з механізмом ієрархічного батч-планування більш ніж на 70%, майже відтворюючи базову продуктивність (див. рисунок 3.13).

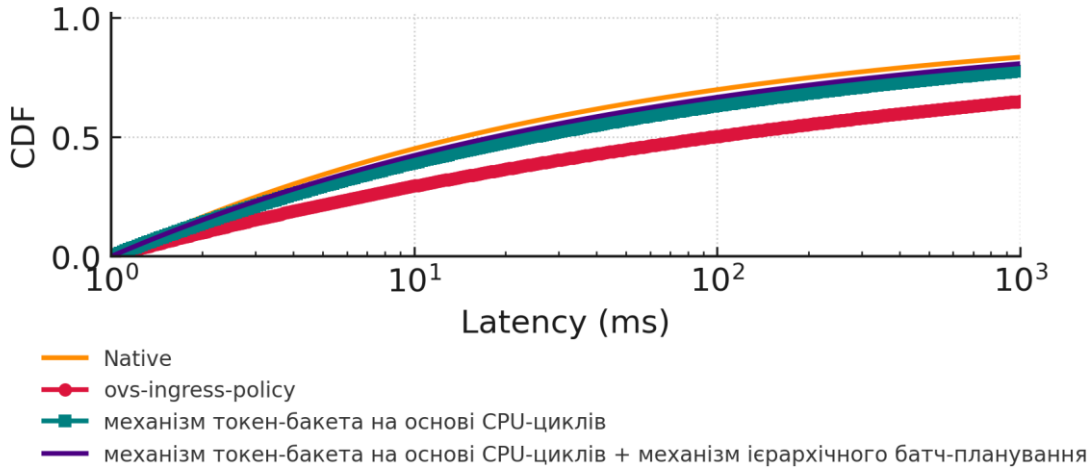

Рисунок 3.13 – Результати продуктивності програм: час відгуку Nginx

Функція накопиченого розподілу (CDF) відображає частку виміряних значень, що не перевищують певного порогу [26]. У контексті мережевих

досліджень CDF показує, яка частина затримок є меншою або дорівнює заданому значенню latency. Наприклад, значення CDF = 0.9 для певної точки означає, що 90% запитів були оброблені із затримкою, що не перевищує відповідний поріг. Така форма представлення дозволяє не лише оцінити середню продуктивність системи, а й аналізувати поведінку її «хвостів» (90th/95th/99th percentile), що є критично важливим для дослідження стабільності QoS під навантаженням та порівняння різних механізмів розподілу ресурсів.

Разом це свідчить, що метод ізоляції ресурсів vSwitch на основі CPU-циклів коректно працює і для різних профілів сервісів, забезпечуючи одночасно незмінність смуги для «bandwidth-sensitive» і керовану латентність для «latency-sensitive».

Накладні витрати методу ізоляції ресурсів vSwitch на основі CPU-циклів оцінено окремо. В одно- та багато VM сценаріях просідання продуктивності OVS-DPDK не спостерігалось. datapath не змінювався суттєво, а додаткова логіка у PMD - це, по суті, підрахунок тактів rdtsc. На IO-ядрах облік циклів додає близько 0.018% використання CPU і не масштабується з числом VM , оскільки інтегрований у батч-процесинг портів. Окремо враховано менеджер-потік, що оновлює токени й керує чергами. За розгортання 18 VM додаткове споживання становило близько 3.09% CPU, і навіть у «щільних» конфігураціях не перевищувало 5%; інтервал «пробудження» 70 μs підібрано як компроміс між точністю та витратами. Сукупно це підтверджує, що програмна ціна методу ізоляції ресурсів vSwitch на основі CPU-циклів низька і прийнятна для продакшн-хмар.

Метод ізоляції ресурсів vSwitch на основі CPU-циклів демонструє саме ті властивості, заради яких його спроєктовано: жорстку ізоляцію за CPU-циклами, що захищає надану смугу VM навіть у присутності «шумних» сусідів; ієрархічну, передбачувану затримку завдяки механізму ієрархічного батч-планування; ефективне використання ресурсів у режимі MIN–MAX без простоїв CPU; мінімальні накладні витрати на облік і планування. Графічні докази підтверджують стабільність смуги під механізм токен-бакета на основі CPU-циклів, значне зниження та детермінізацію затримки під механізм токен-бакета

на основі CPU-циклів та механізм ієрархічного батч-планування і прийнятність додаткових витрат на CPU. У сукупності це дає змогу формувати реалістичні SLA для орендарів зі змішаними профілями навантажень, спираючись на вимірювані, керовані й ізольовані CPU-ресурси в межах програмного vSwitch.

У контексті кібербезпеки результати експериментів із методом ізоляції ресурсів vSwitch на основі CPU-циклів мають особливо важливе значення, адже вони показують, що контроль і ізоляція CPU-циклів можуть виконувати не лише роль механізму підвищення якості обслуговування, а й засобу підвищення безпеки віртуалізованої інфраструктури. Відомо, що спільне використання процесорних ядер різними орендарями створює умови для появи побічних каналів (side-channel attacks), зокрема атак на основі вимірювання часу виконання або споживання ресурсів (наприклад, Prime+Probe чи Flush+Reload). Ізоляція CPU-ресурсів у даному методі безпосередньо зменшує ризик таких атак, оскільки кожна VM отримує суворо визначений бюджет циклів і не має можливості впливати на стан кешу або черг інших орендарів через конкурентне планування. Крім того, механізми токен-бакета на основі CPU-циклів і ієрархічного батч-планування обмежують можливість навмисного створення resource starvation, коли зловмисна VM намагається зайняти PMD-потік інтенсивним дрібнопакетним трафіком, щоб спричинити деградацію QoS у сусідів. За рахунок жорсткого контролю споживання CPU та пріоритетного планування такі атаки стають неефективними, а поведінка комутатора - детермінованою. Таким чином, метод ізоляції ресурсів vSwitch на основі CPU-циклів не лише покращує стабільність і передбачуваність мережевої продуктивності, але й підсилює рівень ізоляції між орендарями, підвищуючи загальну резистентність віртуального комутатора до побічних каналів і атак типу DoS у площині даних.

3.4 Висновки до розділу 3

В третьому розділі було експериментально обґрунтовано метод ізоляції ресурсів vSwitch на основі CPU-циклів для OVS-DPDK і показано, що мережеві

гарантії напряму пов’язані з обчислювальним бюджетом ядер. Підтверджено лінійність зв’язку «bandwidth → CPU-cycles» для Intel Xeon Silver 4314 і визначено практичні межі продуктивності на ядро.

Сформульовано дві умови стійкого розміщення VM : сума куплених смуг не перевищує можливості NIC, а сума виділених CPU-циклів - ресурс IO-cores. Це визначає область детермінованої роботи без перевантажень і унеможливлює overprovision. Запропоновано й реалізовано механізм токен-бакета на основі CPU-циклів (strict і MIN–MAX). Strict гарантує жорстку ізоляцію між VM і стабільне утримання SLA навіть за «noisy neighbor». MIN–MAX безпечно залучає B_{idle} за вагами, знімаючи простої CPU без порушення гарантій. Ієрархічне батч-планування додає керовану, градуйовану затримку між класами пріоритетів, запобігає starvation і зберігає FIFO-справедливість.

Інтеграція виконана у PMD-циклі OVS із обліком через rdtsc і керуванням через ovs-vsctl/OVSDB. Накладні витрати залишаються малими навіть за «щільних» конфігурацій. Порівняльні випробування показали, що ovs-ingress-policy вразлива до дрібних кадрів (джиттер, просідання смуги), тоді як механізм токен-бакета на основі CPU-циклів в режимі strict утримує профілі VM , а в режимі MIN-MAX підвищує фактичну смугу у «тихі» періоди. Точність лімітування зберігається для змішаних розмірів пакетів і при масштабуванні за кількістю VM /NUMA. У прикладних сценаріях FTP/Nginx метод забезпечує стабільну смугу для «bandwidth-sensitive» і помітно знижує час відповіді для «latency-sensitive», особливо у зв’язці з ієрархічним плануванням.

З позицій кібербезпеки ізоляція CPU-циклів і пріоритетне планування підсилюють міжорендарську ізоляцію, знижують ризики side-channel і роблять неефективними спроби resource starvation у площині даних. Сукупно метод демонструє стабільну смугу, керовану затримку і низькі накладні витрати, надаючи CSP практичний інструмент для реалістичних SLA у багатокористувацьких хмарах із програмним vSwitch.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Метою кваліфікаційної роботи є дослідження методу ізоляції CPU-ресурсів у vSwitch, який забезпечує стабільну продуктивність і підвищує рівень кіберзахисту від атак, пов’язаних із конкурентним використанням процесорних ресурсів.. Оскільки, проведення робіт з розробки та використання системи передбачає використання комп’ютерної техніки, зокрема ПК та периферійних пристроїв, то обов’язковим є дотримання вимог з охорони праці і техніки безпеки.

Для ефективної і безпечної роботи колективу працівників з розробки системи комп’ютерних систем, в тому числі і фахівців з підвищення ефективності контролю доступу в приміщення, необхідно організувати безпечні умови праці. Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Згідно Вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до цих вимог:

– переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 22.08.2005 N 161, зареєстрованого в Міністерстві юстиції України 05.09.2005 за N 990/11270 (НАПБ Б.06.004-2005);

– Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом Держбуду

України від 28.10.98 N 247 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками.

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014. Приміщення, в яких розміщуються робочі місця операторів сервера загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп’ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97.

У приміщенні, де одночасно експлуатуються понад п’ять комп’ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач,

який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників.

Порядок роз’єднання при відключенні має бути зворотним. Не допускається підключати комп’ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення комп’ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі.

Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18.
Організація робочого місця фахівця із дослідження методів та програмно-апаратних засобів оптимізаційних процесів повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп’ютера, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля.

Отже, у результаті аналізу вимог щодо охорони праці користувачів комп'ютерів, визначено особливості організації робочих місць, вимог з електробезпеки, природного та штучного освітлення для ефективної і безпечної роботи інженерів з розробки та впровадження системи резервування та керування трафіком.

4.2 Державна система моніторингу довкілля, як складова частина національної інформаційної інфраструктури, сумісної з аналогічними системами інших країн.

Законодавство України, а саме Статті 20 та 22 Закону України «Про охорону навколишнього природного середовища» [27], передбачають створення державної системи моніторингу довкілля та проведення спостережень за станом навколишнього природного середовища і рівнем його забруднення.

Основні принципи функціонування державної системи моніторингу довкілля визначені у постанові Кабінету Міністрів України від 30.03.1998 №391 «Про затвердження

Положення про державну систему моніторингу довкілля» [18]. Згідно з цим положенням, «Державна система моніторингу довкілля (ДСМД) – це система спостережень, збирання, оброблення, передавання, збереження та аналізу інформації про стан довкілля, прогнозування його змін і розроблення науково-обґрунтованих рекомендацій для прийняття рішень про запобігання негативним змінам стану довкілля та дотримання вимог екологічної безпеки».

Також визначається, що система моніторингу є відкритою інформаційною системою, складовою частиною національної інфраструктури, сумісної з аналогічними системами інших країн.

Пріоритетами такої системи є збереження екосистем, відвернення кризових змін в екологічному стані довкілля та запобігання відповідним надзвичайним ситуаціям. На даний час, у ДСМД функції і задачі спостережень та інформаційного забезпечення виконують такі суб`єкти системи моніторингу [28]:

- Міністерство захисту довкілля та природних ресурсів України;
- Міністерство аграрної політики та продовольства України;
- Міністерство розвитку громад і територій України;
- Державна служба з надзвичайних ситуацій;
- Державна служба геології та надр України;
- Державне агентство України з управління зоною відчуження;
- Державне агентство лісових ресурсів України;
- Державне агентство водних ресурсів України;
- Державна служба України з питань геодезії, картографії та кадастру;
- Державне космічне агентство України.

Окрім цього, виконання таких функцій покладено на інші центральні органи виконавчої влади, які є суб'єктами державної системи моніторингу довкілля, а також на підприємства, установи та організації, діяльність яких призводить або може призвести до погіршення стану довкілля. В загальному, система моніторингу спрямована на декілька основних цілей, серед яких підвищення рівня вивчення і знань про екологічний стан довкілля, зростання якості інформаційного обслуговування користувачів на всіх рівнях, покращення якості обґрунтування природоохоронних заходів та ефективності їх здійснення, а також сприяння розвитку міжнародного співробітництва у галузі охорони довкілля, раціонального використання природних ресурсів та екологічної безпеки [28].

Для досягнення поставлених цілей, положення про ДСМД визначає такі завдання: систематичні спостереження за станом довкілля, аналіз стану навколишнього середовища, прогнозування змін довкілля, інформаційна підтримка прийняття рішень, забезпечення органів державної та місцевої влади, населення та партнерів інформацією про актуальний стан довкілля [28]. Також згідно з положенням складовими частинами державного моніторингу навколишнього середовища України є моніторинг атмосферного повітря, води, земель, біологічного різноманіття, лісів, відходів, геологічного середовища, фізичних факторів впливу.

Нормативними актами, що регламентують моніторинг таких об’єктів є відповідні постанови Кабінету Міністрів України щодо порядків здійснення

моніторингу повітря, вод, земель та ґрунтів. Система моніторингу ґрунтується на використанні існуючих організаційних структур суб'єктів моніторингу і функціонує на основі єдиного нормативного, організаційного, методологічного і метрологічного забезпечення, об'єднання складових частин та уніфікованих компонентів цієї системи [29]. Зазначимо, що функціонування ДСМД здійснюється на трьох рівнях, які розподіляються за територіальним принципом, а саме: загальнодержавний, регіональний та локальний рівні.

Відповідальні суб’єкти здійснюють моніторинг різного роду об’єктів, серед яких [28]:

- ґрунти на природоохоронних територіях, а також ґрунти сільськогосподарського та лісового фондів;

- види рослинного і тваринного світу, що перебувають під загрозою зникнення чи під особливою охороною;

- VM іст радіонуклідів в атмосферному повітрі, водах та ґрунтах;

- наявність та серйозність повеней, паводків, снігових лавин, селів;

- об’єкти зберігання та захоронення радіоактивних відходів;

- сільськогосподарські рослини, тварини і продуктів з них, мисливська фауна та лісова рослинність;

- якість вод водогосподарських систем міжгалузевого та сільськогосподарського водопостачання;

- зрошувані та осушувані землі у сенсі глибини залягання та мінералізації ґрунтових вод, ступені засоленості та солонцюватості ґрунтів;

- ґрунти і ландшафти щодо проявів ерозійних та інших екзогенних процесів та просторового забруднення земель об'єктами промислового і сільськогосподарського виробництва;

- берегові лінії річок, морів, озер, водосховищ, лиманів, заток, гідротехнічних споруд;

- стічні води міської каналізаційної мережі та очисні споруди, джерела скидання таких вод;

- зелені насадження у містах і селищах міського типу.

Якщо розглянути моніторинг повітря, то Державною гідрометеорологічною службою здійснюється спостереження за забрудненням атмосферного повітря у містах України. Державна екологічна інспекція здійснює відбір проб на джерелах викидів, а санітарно-епідеміологічна служба координує моніторинг якості атмосферного повітря у житлових зонах. Контроль якості повітря також включає аналіз опадів та снігового покрову. Програма обов`язкового моніторингу якості атмосферного повітря охоплює сім забруднюючих речовин: пил, двоокис азоту, двоокис сірки, оксид вуглецю, формальдегід, свинець та бензапірен. Спостереження за водами суші на 151 об’єкті проводить Державна гідрометеорологічна служба, що включає в себе оцінку хімічного складу вод, біогенних параметрів, наявність зважених часток та органічних речовин, основних забруднюючих речовин, важких металів та пестицидів. Контроль за водами суші також здійснюють Державна екологічна інспекція, Державний комітет по водному господарству, Санітарно-епідеміологічна служба та Державна геологічна служба. Дослідження включають моніторинг річок, водосховищ, каналів тощо, контроль хімічних, радіаційних та фізичних показників, а також придатності води до споживання. За схожими параметрами відбувається й моніторинг прибережних вод. До моніторингу ґрунтів входить вимірювання забруднення ґрунтів пестицидами, агровідходами, токсинами та важкими металами на сільськогосподарських землях та промислових майданчиках. Також досліджується забруднення ґрунту у місцях захоронення відходів. Контроль здійснюється державною гідрометеорологічною службою, Міністерством захисту довкілля та Міністерством аграрної політики. Моніторинг радіаційного випромінювання включає в себе спостереження за радіоактивним забрудненням атмосфери, поверхневих вод та ґрунтів поблизу атомних електростанцій та у зоні відчуження [29].

Згідно з положенням [28] суб’єкти системи моніторингу інформаційно підтримують рішення в галузі охорони довкілля, безкоштовно обмінюються результатами спостережень на об’єктах та колективно використовують інформаційні ресурси, надаючи всім зацікавленим сторонам відповідні дані.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра було сформовано, обґрунтовано та експериментально перевірено підхід до ізоляції процесорних ресурсів у віртуальному комутаторі (Open vSwitch з DPDK), який одночасно підвищує продуктивність і кіберстійкість площини даних у багатокористувацьких хмарних середовищах.

В першому розділі було проаналізовано архітектуру vSwitch у контексті SDN із чітким розмежуванням площин управління, керування та даних. Показано, що перенесення обробки в користувацький простір забезпечує високу швидкодію, але створює спільний пул CPU-циклів для різних орендарів, що веде до варіації затримки, втрат і часових побічних каналів. Розглянуто програмні та апаратні методи ізоляції (CPU pinning, cgroups/cpusets, eBPF, SR-IOV, VT-d, NUMA, CAT) і обґрунтовано ефективність їхнього комбінованого застосування.

В другому розділі було формалізовано зв’язок між пропускною здатністю та витратами процесорного часу у vSwitch, розкладеними на етапи приймання, класифікації та передавання. Побудовано модель оцінювання потреби VM у CPU з урахуванням характеристик трафіку та параметрів розгортання (NUMA-топологія, щільність VM , кількість IO-ядер). Сформульовано умови стабільної роботи системи та запропоновано два ключові механізми: механізм токенів на основі CPU-циклів (із політикою мінімум–максимум) і ієрархічне пакетне планування для керованої затримки.

В третьому розділі було реалізовано запропонований підхід у OVS-DPDK через модифікацію циклу PMD-потоку, додано високоточний облік процесорних циклів та інтегровано керування параметрами через інструменти конфігурації. Експериментально показано, що CPU-орієнтований механізм токенів забезпечує жорстку ізоляцію продуктивності навіть за наявності «шумних сусідів», а ієрархічне пакетне планування зменшує варіацію затримки і гарантує передбачувану затримку для пріоритетних сервісів без голодування низькопріоритетних. Дотримання ресурсних обмежень мережевого інтерфейсу

та IO-ядер підтримує виконання SLA, а політика мінімум–максимум ефективно використовує невикористані ресурси без порушення ізоляції.

Підсумково, у роботі запропоновано CPU-орієнтовану модель QoS для vSwitch, розроблено й інтегровано механізми ізоляції CPU-циклів, а також емпірично підтверджено підвищення передбачуваності пропускної здатності та затримки і зменшення впливу атак на виснаження ресурсів і часових побічних каналів. Практична цінність підходу полягає в прямому відображенні мережевих SLA у виділення процесорного часу комутатора, що робить роботу vSwitch більш детермінованою, масштабованою та безпечною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. GeeksforGeeks. (2019, July 2). What is Software Defined Networking (SDN)? - GeeksforGeeks. https://www.geeksforgeeks.org/computer-networks/software-defined-networking/
2. Goodwin, M. (n.d.). What Is Network Functions Virtualization (NFV)? | IBM. IBM. https://www.ibm.com/think/topics/network-functions-virtualization
3. IBM PowerVC for Private Cloud. (n.d.). IBM. https://www.ibm.com/docs/en/powervc-cloud/2.3.0?topic=apis-northbound-rest
4. DPDK. (n.d.). The open source data plane development kit accelerating network performance. https://www.dpdk.org/
5. Open vSwitch. (n.d.). Open vSwitch. https://www.openvswitch.org/
6. Linux Bridge vs OVS Bridge – NodeSpace Blog. (n.d.). NodeSpace Blog. https://blog.nodespace.com/linux-bridge-vs-ovs-bridge/
7. AIX. (n.d.). IBM. https://www.ibm.com/docs/en/aix/7.1.0?topic=architecture-processor-affinity-binding
8. Your CPU is NOT Starving! (n.d.). IBM. https://www.ibm.com/support/pages/your-cpu-not-starving
9. Mazaheri, M. E., Sarmadi, S. B., & Ardakani, F. T. (2022, January 1). A Study of Timing Side-Channel Attacks and Countermeasures on JavaScript and WebAssembly. The ISC International Journal of Information Security.
10. Awati, R. (2022, September 22). What is non-uniform memory access (NUMA)? WhatIs. https://www.techtarget.com/whatis/definition/NUMA-non-uniform-memory-access
11. Raj, A., & Dharanipragada, J. (2017). Keep the PokerFace on! Thwarting cache side channel attacks by memory bus monitoring and cache obfuscation. Journal of Cloud Computing, 6(1). https://doi.org/10.1186/s13677-017-0101-4
12. Resource Starvation - Application Security Tactics & Techniques Matrix. (n.d.). Index - Application Security Tactics & Techniques Matrix. https://app-attack-

matrix.com/techniques/Impact/Service%20Disruption/subtechniques/Resource%20Starvation/

13. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.
14. Klots, Y., Titova, V., Petliak, N., Tymoshchuk, D., Zagorodna, N. Intelligent data monitoring anomaly detection system based on statistical and machine learning approaches. CEUR Workshop Proceedings, (2025), 4042, pp. 80 – 89
15. Petliak, N., Klots, Y., Karpinski, M., Titova, V., Tymoshchuk, D. Hybrid system for detecting abnormal traffic in IoT. CEUR Workshop Proceedings, (2025), 4057, pp. 21 – 36
16. B. Lypa, I. Horyn, N. Zagorodna, D. Tymoshchuk, T. Lechachenko, Comparison of feature extraction tools for network traffic data, CEUR Workshop Proceedings, 3896, 2024, pp. 1-11.
17. Open vSwitch (OVS): What Is It and How Does Open Virtual Switch Work? (n.d.). Cloudification - Private, Hybrid, Edge Cloud Solutions for Your Business. https://cloudification.io/cloud-blog/open-vswitch-ovs-what-is-it-and-how-does-open-virtual-switch-work/
18. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56. https://doi.org/10.31891/2219-9365-2024-79-7
19. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220. https://doi.org/10.31891/2219-9365-2024-80-26
20. Тимощук, В., Долінський, А., & Тимощук, Д. (2024). ЗАСТОСУВАННЯ ГІПЕРВІЗОРІВ ПЕРШОГО ТИПУ ДЛЯ СТВОРЕННЯ ЗАХИЩЕНОЇ ІТ-ІНФРАСТРУКТУРИ. Матеріали конференцій МЦНД, (24.05. 2024; Запоріжжя, Україна), 145-146.

21. Welcome to QEMU’s documentation! — QEMU documentation. (n.d.). QEMU. https://www.qemu.org/docs/master/
22. Ubuntu Server documentation. (n.d.). Ubuntu Server. https://documentation.ubuntu.com/server/
23. iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. (n.d.). iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. https://iperf.fr/
24. GitHub - rbruenig/qperf: qperf is a performance measurement tool for QUIC similar to iperf. (n.d.). GitHub. https://github.com/rbruenig/qperf
25. GitHub - pktgen/Pktgen-DPDK: DPDK based packet generator. (n.d.). GitHub. https://github.com/pktgen/Pktgen-DPDK
26. Using the cumulative distribution function (CDF) - Minitab. (n.d.). Support | Minitab. https://support.minitab.com/en-us/minitab/help-and-how-to/probability-distributions-random-data-and-resampling-analyses/supporting-topics/basics/using-the-cumulative-distribution-function-cdf/
27. Микитишин А. Г., Митник М. М., Стухляк П. Д. Телекомунікаційні системи та мережі. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2017. 384 с.
28. Nedzelskyi, D., Derkach, M., Tatarchenko, Y., Safonova, S., Shumova, L., & Kardashuk, V. (2019, August). Research of efficiency of multi-core computers with shared memory. In 2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW) (pp. 111-114). IEEE.
29. Закон України «Про охорону навколишнього природного середовища» №1264-XII. URL: https://zakon.rada.gov.ua/laws/ show/1264-12 (дата звернення: 14.12.2023).
30. Постанова Кабінету Міністрів України «Про затвердження Положення про державну систему моніторингу довкілля» №391-98-п. URL: https://zakon.rada.gov.ua/laws/show/391-98 (дата звернення: 14.12.2023).
31. Стручок В.С. Техноекологія та цивільна безпека. Частина «Цивільна безпека». Навчальний посібник. Тернопіль: ТНТУ. 2022. 150 с.

Додаток А Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Краківський економічний університет (Польща)
Вроцлавський економічний університет (Польща)
Університет «Опольська Політехніка» (Польща)
Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Вінницький національний аграрний університет
Львівський національний університет ім. І. Франка
Головне управління Пенсійного фонду в Тернопільській області
Наукове товариство ім. Шевченка
Тернопільський обласний комунальний інститут післядипломної педагогічної освіти
Сумський державний педагогічний університет
Запорізький національний університет

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів

11-12 грудня 2025 року

УКРАЇНА
ТЕРНОПІЛЬ – 2025

УДК 004.056.5
О. Ю. Горішний, В.Д. Тимощук
(Тернопільський національний технічний університет імені Івана Пулюя)

ПІДВИЩЕННЯ БЕЗПЕКИ ПЛОЩИНИ ДАНИХ ВІРТУАЛЬНОГО КОМУТАТОРА OPEN VSWITCH

O. Horishnyi, V. Tymoshchuk

ENHANCING THE DATA PLANE SECURITY OF OPEN VSWITCH

У роботі розглянуто проблему забезпечення стабільної продуктивності та кібербезпеки віртуальних мережевих сервісів у хмарних інфраструктурах, що базуються на програмному комутаторі Open vSwitch [1] з підтримкою DPDK (Data Plane Development Kit) [2]. Перенесення площини даних у користувацький простір дозволяє досягати пропускної здатності, близької до лінійної швидкості мережевого інтерфейсу, однак водночас створює новий клас загроз, пов'язаних із конкурентним використанням CPU-ресурсів кількома орендарями. Ефект "noisy neighbor" призводить до деградації QoS, зростання затримок, втрати пакетів та побічного витоку інформації через вимірювання часових характеристик (side-channel timing attacks), що суперечить вимогам до доступності й стійкості критичних сервісів у хмарних дата-центрах. Метою дослідження є розроблення та експериментальна перевірка методу ізоляції CPU-ресурсів у vSwitch, який забезпечує гарантовану пропускну здатність і керовану затримку для кожної віртуальної машини, розглядаючи процесорний час як базовий ресурс площини даних. На основі детального аналізу процедури форвардингу в OVS-DPDK побудовано модель споживання процесорних циклів, що розкладає загальні витрати на три компоненти (ingress, classification та egress) і враховує вплив інтенсивності трафіку, розміру пакетів, кількості потоків, NUMA-топології, щільності розміщення ВМ і конфігурації PMD-потоків (Poll Mode Driver Thread). Показано, що залежність між пропускною здатністю та кількістю необхідних CPU-циклів у робочому діапазоні навантажень є практично лінійною, що дозволяє перейти від bps/pps-орієнтованих політик до CPU-орієнтованого QoS.

Запропонований метод ізоляції ресурсів vSwitch на основі CPU-циклів поєднує два ключові механізми. Перший - token bucket, у якому токени еквівалентні кількості доступних процесорних циклів для кожної ВМ [3]. Швидкість генерації токенів визначається з урахуванням придбаної смуги пропускання та емпіричних коефіцієнтів середовища розгортання, що забезпечує жорстку ізоляцію CPU-бюджетів і можливість реалізації політики MIN–MAX для динамічного перерозподілу надлишкових циклів між активними орендарями. Другий - hierarchical batch scheduling у PMD-потоці, який впорядковує оброблення трафіку за класами пріоритетів і забезпечує обмежену найгіршу затримку для кожного класу без використання блокувань і конкурентного доступу до спільних структур даних [4].

Реалізація методу виконана шляхом модифікації головного циклу PMD-потоку OVS-DPDK та інтеграції нових параметрів у систему керування ovs-vsctl, що дозволяє конфігурувати бюджети CPU та пріоритети без перезапуску сервісів. Для вимірювання фактичних витрат процесорних циклів використано інструкцію rdtsc, що мінімізує накладні витрати моніторингу. Експериментальні дослідження в середовищі з декількома ВМ [5,6] продемонстрували, що запропонований метод усуває вплив "noisy neighbor", забезпечує близьку до заданої пропускну здатність для добропорядних орендарів і істотно знижує варіативність затримки порівняно зі стандартним OVS-DPDK. Отримані результати підтверджують доцільність переходу до CPU-

орієнтованого QoS як інструмента підвищення продуктивності й кіберстійкості програмно-визначених віртуалізованих мереж.

Література

1. Open vSwitch. URL: https://www.openvswitch.org/ (date of access: 30.11.2025).

2. DPDK Support – Open vSwitch 3.6.90 documentation. Project. URL: https://docs.openvswitch.org/en/latest/topics/dpdk/ (date of access: 30.11.2025).

3. Hossain T. Token Bucket Algorithm Explained. DEV Community. URL: https://dev.to/0xtanzim/token-bucket-algorithm-explained-4ceo (date of access: 30.11.2025).

4. Multilevel Queue (MLQ) CPU Scheduling - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/operating-systems/multilevel-queue-mlq-cpu-scheduling/ (date of access: 30.11.2025).

5. Interactive cybersecurity training system based on simulation environments / D. Tymoshchuk et al. Measuring and computing devices in technological processes. 2024. No. 4. P. 215–220. URL: https://doi.org/10.31891/2219-9365-2024-80-26 (date of access: 30.11.2025).

6. Tymoshchuk D., Yatskiv V. Using hypervisors to create a cyber polygon. Measuring and computing devices in technological processes. 2024. No. 3. P. 52–56. URL: https://doi.org/10.31891/2219-9365-2024-79-7 (date of access: 30.11.2025).