ВСТУП

Актуальність теми. Стрімке зростання обсягів мобільного трафіку та впровадження стандартів 5G і перспективних мереж 6G вимагають використання технологій, здатних забезпечити надвисоку пропускну здатність у складних умовах поширення радіохвиль. Основою сучасного фізичного рівня бездротового зв'язку є технологія ортогонального частотного мультиплексування (OFDM), яка дозволяє ефективно боротися з частотно-селективним згасанням сигналу. Проте, попри значні переваги, OFDM-системи демонструють високу чутливість до часових та частотних зміщень (CFO), що виникають внаслідок нестабільності генераторів або доплерівських зсувів у сценаріях високої мобільності.

В умовах щільної міської забудови та швидкого руху абонентів (автомобільний зв'язок, високошвидкісні поїзди) традиційні методи синхронізації часто стикаються з проблемою розмиття кореляційних піків та втратою ортогональності піднесучих. Водночас сучасні мобільні пристрої та вузли «туманних» обчислень вимагають енергоефективних алгоритмів обробки сигналів, які б мали низьку обчислювальну складність для роботи в режимі реального часу. Використання автокореляційних властивостей циклічного префікса є ключовою стратегією, що дозволяє виконувати синхронізацію без значних витрат спектрального ресурсу та потужності, що робить дослідження меж ефективності цього методу критично важливим для оптимізації сучасних та майбутніх систем мобільного зв'язку.

Мета роботи полягає у комплексному дослідженні та обґрунтуванні методів підвищення ефективності застосування VNF у телекомунікаційних мережах з урахуванням вимог до продуктивності, затримки, стабільності сервісу та економічної доцільності.

Об’єкт дослідження – телекомунікаційна мережа з інфраструктурою NFV (NFVI), у якій мережеві функції реалізовано у вигляді VNF (у віртуальних машинах та/або контейнеризованому середовищі) в складі ланцюгів сервісних функцій, що

обслуговують трафік з вимогами до QoS/QoE (зокрема для сервісів радіозв'язку, радіомовлення і телебачення).

Предмет дослідження – методи, моделі та інструменти підвищення ефективності застосування VNF у телекомунікаційних мережах: оптимізація мережевого dataplane та I/O тракту, ресурсна ізоляція (CPU/NUMA/IRQ pinning, hugepages), механізми розміщення (placement) і масштабування (autoscaling), а також економіко-математичні підходи до оцінювання ефекту (TCO, unit-economics) з урахуванням SLA/SLO.

Методи дослідження. аналіз наукових і нормативно-технічних джерел у сфері NFV/SDN та віртуалізації мережевих функцій; методи системного аналізу для формування критеріїв ефективності VNF та визначення факторів, що впливають на продуктивність і якість сервісу; експериментальні методи збору фактологічних даних (на основі матеріалів практики/експлуатаційних вимірювань) щодо throughput, PPS, затримок (p50/p95/p99), джитеру та втрат пакетів; статистичні методи обробки даних (агрегація, групування, перцентильний аналіз, кореляційний аналіз); економіко-математичні методи оцінювання техніко-економічного ефекту (розрахунок потреби в ресурсах, кількості серверів, ТСО та вартості 1 Gbps·міс); порівняльний аналіз альтернативних конфігурацій розгортання VNF та формування конструктивних пропозицій на основі отриманих результатів.

Наукова новизна отриманих результатів полягає у наступному:

Уточнено систему показників ефективності VNF з акцентом на перцентильні характеристики затримки (p95/p99), джитер і втрати пакетів як ключові параметри для телекомунікаційних сервісів реального часу.

Запропоновано підхід до багатокритеріальної оцінки ефективності VNF, що дозволяє порівнювати альтернативні конфігурації dataplane та I/O-тракту з урахуванням SLA/SLO.

Обґрунтовано метод вибору режиму мережевого вводу/виводу та ресурсної ізоляції (vSwitch/прискорений dataplane/SR-IOV) залежно від класу VNF і вимог до якості обслуговування.

Практичне значення отриманих результатів полягає у можливості використання отриманих результатів під час проєктування, модернізації та експлуатації телекомунікаційних мереж із застосуванням NFV/VNF. Запропоновані заходи дозволяють:

- зменшити затримки та їх хвости (p95/p99), джитер і втрати пакетів у мережі, що є критичним для сервісів радіозв'язку, радіомовлення та телебачення;
- підвищити продуктивність dataplane та ефективність використання обчислювальних ресурсів NFVI (CPU/мережа), забезпечуючи виконання SLA/SLO при зростанні трафіку;
- знизити експлуатаційні витрати за рахунок оптимального масштабування та зменшення необхідної кількості серверів, що підтверджується техніко-економічними розрахунками (TCO, вартість 1 Gbps·міс);
- застосувати методичні рекомендації як основу для впровадження політик ресурсної ізоляції, SLO-керованого автомасштабування та регресійного тестування продуктивності VNF у виробничих умовах.

1 СУЧАСНИЙ СТАН НАУКОВОЇ ДУМКИ ЩОДО ПРОБЛЕМ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ VNF В ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖАХ

У цьому розділі узагальнено сучасний стан наукової та інженерної думки щодо віртуалізації мережевих функцій (NFV) і підходів до підвищення ефективності віртуалізованих мережевих функцій (VNF). Розглянуто базові архітектурні принципи, метрики ефективності, теоретичні моделі та практичні інструменти оптимізації dataplane, керування ресурсами й оркестрації. Проведено критичний огляд ключових концепцій, визначено типові обмеження та компроміси, сформульовано авторську позицію і проблемні питання дослідження. При викладі використано положення стандартів ETSI NFV, рекомендацій IETF, 3GPP, а також актуальні інженерні практики високопродуктивної обробки пакетів у програмно визначених середовищах [1–6].

Класична телекомунікаційна інфраструктура тривалий час будувалася на спеціалізованих апаратних платформах (appliances), що реалізують мережеві функції у вигляді фізичних пристроїв (PNF — Physical Network Functions): маршрутизатори, BRAS/BNG, шлюзи, балансувальники, SBC, системи DPI/IDS/IPS тощо. Такий підхід забезпечував передбачувану продуктивність, проте був повільним в еволюції: кожна зміна функціоналу вимагала закупівлі нового обладнання, циклів сертифікації, складної логістики і прив’язки до конкретних постачальників.

Зростання мобільного трафіку, поява LTE/5G, розвиток OTT/відеосервісів, IoT та вимоги до швидкого масштабування змусили перенести акцент з апаратно-орієнтованих архітектур на програмно-керовані підходи. У цьому контексті виникли дві парадигми: SDN (Software-Defined Networking) — винесення керування мережею в логічно централізований контролер, та NFV — перенесення мережевих функцій у програмні сутності, що виконуються на стандартних обчислювальних ресурсах (COTS) [1,2].

ETSI визначає NFV як архітектурний підхід, що дозволяє реалізовувати мережеві функції у вигляді програмних компонентів (VNF), які розгортаються на віртуалізованій інфраструктурі (NFVI) і керуються системою оркестрації (MANO) [1,2]. Практично це означає, що телекомунікаційна мережа стає ближчою до хмарної інфраструктури: функції масштабуються, мігрують, оновлюються та конфігуруються автоматизовано.

Окремим етапом еволюції стало «cloud-native» переосмислення телекомунікаційних функцій: перехід від монолітних VNF у VM до контейнеризованих CNF (Cloud-Native Network Functions), керованих Kubernetes та суміжними інструментами життєвого циклу мікросервісів [4,6]. Проте для телеком-навантажень це створює нові ризики: збільшення кількості мережевих хопів, накладні витрати CNI, непередбачуваність латентності й залежність від якості ресурсної ізоляції. Тому підвищення ефективності VNF/CNF залишається актуальним дослідницьким напрямом.

Для освітньої програми «Апаратура радіозв'язку, радіомовлення і телебачення» тема має додаткову вагу, оскільки більшість медіа- та радіосервісів є затримко-чутливими: важливими стають не лише середні значення, а й хвости розподілу затримки (p95/p99), джитер та контроль втрат пакетів, що прямо впливає на QoE і стабільність потоків [5].

1.1 Архітектура NFV: NFVI, VNF, MANO та ланцюги сервісних функцій

Стандартизована архітектура NFV охоплює три ключові блоки: NFVI (обчислення, мережа, сховище, віртуалізаційний шар), VNF (програмні реалізації мережевих функцій) та MANO (керування життєвим циклом VNF і оркестрація ресурсів) [1,2]. У практичних впровадженнях NFVI може базуватися на гіпервізорах (KVM/ESXi), контейнерних середовищах або їх комбінації; віртуальний комутатор часто реалізується на базі Open vSwitch (OVS) або аналогів, а MANO — через OpenStack, ONAP або вбудовані оркестратори постачальників.

Суттєвим елементом застосування VNF є ланцюги сервісних функцій (Service Function Chaining, SFC) — маршрутизація потоків через послідовність функцій:

класифікація → фільтрація → NAT → балансування → DPI тощо [3]. У такому ланцюгу затримка накопичується, а джитер посилюється, тому оптимізація однієї функції без урахування всього тракту може не дати очікуваного ефекту. Рекомендації IETF підкреслюють важливість узгодженого керування QoS, stateful-поведінкою та переадресацією потоків при змінах топології або масштабуванні [3].

На рівні керування життєвим циклом VNF розглядаються задачі інстанціювання, масштабування (scale-out/scale-in), відмовостійкості, оновлення без перерви сервісу (rolling upgrade), міграції та відновлення після збоїв. Кожен із цих процесів потенційно впливає на ефективність через зміну локальності пам’яті, перерозподіл навантаження на NIC-чергах, перезапуск потоків і повторну ініціалізацію stateful-таблиць.

1.2 Поняття ефективності VNF: метрики, критерії та типові помилки оцінювання

У науковій і прикладній літературі ефективність VNF інтерпретується як здатність виконувати вимоги сервісу при мінімальних ресурсних та економічних витратах. Ключовими є показники якості обслуговування (QoS) і якості сприйняття (QoE), які формалізуються через SLA/SLO. Для коректного оцінювання VNF в телекомунікаційних мережах необхідно застосовувати багатовимірний набір метрик, а не один «throughput у бенчмарку» [1,5].

Основні групи метрик ефективності VNF:

- Продуктивність dataplane: throughput (Gbps), PPS, обробка малих пакетів, стабільність у пікових режимах.
- Затримка та її хвости: p50/p95/p99 latency, розкид (jitter), детермінізм часу обробки пакетів.
- Надійність передачі: drop rate (втрати), queue drops, retransmissions, помилки обробки сесій.
- Ресурсна ефективність: CPU utilization, softirq time, контекстні перемикання, використання пам’яті, кеш-ефекти.

- Енергоефективність та економіка: TCO, вартість 1 Gbps·міс, продуктивність на ядро/ват.
- Експлуатаційні показники: час масштабування, відновлення після збою, вплив міграцій/оновлень.

Типові помилки оцінювання: (1) використання лише середніх значень latency без аналізу p95/p99; (2) вимірювання throughput на великих пакетах без PPS-навантаження; (3) ігнорування service chaining та мережевих хопів; (4) відсутність оцінки під реальними профілями трафіку (мікс UDP/TCP, різні MTU, bursty-навантаження); (5) відсутність зв’язку між технічними метриками і економічним ефектом. Для сервісів мовлення/телебачення критичною є саме стабільність потоку, буферизація і джитер, а не максимальний throughput у лабораторії.

1.3 Проблеми продуктивності та латентності у VNF: причини на рівні системи

У VNF-середовищі пакет проходить складний шлях: NIC → драйвер → мережевий стек ОС → віртуальний комутатор → віртуальний інтерфейс VNF → обробка → зворотний шлях. Додаткові прошарки створюють накладні витрати та варіативність затримок. Сучасні джерела виділяють такі системні причини деградації [2,5,7–10].

- Переривання та softirq: висока інтенсивність IRQ при PPS-навантаженні призводить до конкуренції за CPU і росту latency.
- Віртуалізаційні накладні витрати: VM exit/entry, emulation, віртуальні драйвери, додаткові копії буферів.
- Планувальник ОС: несприятливі перемикання задач, contention за ядра, CPU steal у розділених середовищах.
- NUMA-ефекти: доступ до пам’яті з іншого NUMA-вузла (remote access) збільшує затримку і джитер.
- Конкуренція за кеш: спільне використання L3/LLC різними інстансами призводить до нестабільності p99.
- Черги в vSwitch: буферизація, переповнення, head-of-line blocking, неоптимальні розміри ring buffer.

- Гіпервізорні/контейнерні мережеві прошарки: bridge/VxLAN/Geneve тунелі, накладні витрати encapsulation.

Важливим є і феномен «latency tail»: навіть коли середня затримка низька, рідкісні піки (p99+) можуть визначати QoE. У телеком-сервісах ці піки часто спричиняють повторні передачі, коливання буферів, втрату синхронізації і деградацію медіапотоків, що особливо критично для систем радіомовлення та телебачення.

1.4 Теорії та концепції підвищення ефективності VNF: fast path, offload, ізоляція

Підвищення ефективності VNF у телекомунікаційних мережах традиційно розглядається через призму прискорення обробки пакетів у площині даних (dataplane), зменшення накладних витрат віртуалізації та підвищення детермінізму роботи системи. У сучасній науковій та інженерній практиці сформувалися три «ядра» підходів: (1) fast path (прискорені тракти обробки пакетів), (2) апаратні offload-механізми (перенесення частини функцій у NIC/прискорювачі), (3) ресурсна ізоляція (стабілізація p95/p99 затримок через керування CPU/NUMA/IRQ/пам’яттю). Важливо, що ефективність у телеком-сценаріях визначається не лише піковим throughput, а передусім хвостами затримки та стабільністю (jitter), тобто здатністю виконувати SLA/SLO у реальних, варіативних умовах трафіку [2,5,7–10].

Для коректного аналізу доцільно розглядати пакетну обробку як конвеєр, де сумарна затримка L є сумою затримок на етапах чергування/буферизації, обробки CPU та пересилання між контекстами: $L \approx L_{RX_queue} + L_{driver} + L_{switch} + L_{VNF} + L_{TX_queue}$. У цьому конвеєрі «вузьким місцем» часто стає не швидкість інструкційного виконання, а черги, переривання, конкуренція за кеш та пам’ять, а також непередбачуваність планувальника ОС, що і формує p99-«хвости» [7,9].

Fast path у контексті NFV означає скорочення «гарячого» шляху пакета (critical datapath) за рахунок усунення зайвих копій буферів, мінімізації контекстних перемикань, оптимізації роботи з NIC-чергами та зниження залежності від універсального мережевого стека ОС. Це не «одна технологія», а

клас оптимізацій, що націлений на три ефекти: (1) підвищення PPS при малих пакетах; (2) зниження латентності та джитеру; (3) покращення масштабування по ядрах [5,8–10].

У високошвидкісних мережах (25/50/100GbE) критичним стає не стільки throughput у Гбіт/с, скільки інтенсивність пакетів. Наприклад, навіть при 10–20 Гбіт/с малими пакетами dataplane може впертися у PPS-ліміт через переривання, обробку заголовків та таблиць потоків. Отже fast path — це насамперед боротьба з PPS-навантаженням і tail latency, а вже потім — з «сирою» пропускною здатністю.

- Зменшення копіювань (copy avoidance): zero-copy або мінімізація memcpy між драйвером, vSwitch і VNF.
- Контроль переривань (interrupt moderation) або перехід на polling, щоб прибрати випадковість IRQ.
- Оптимізація кеш-локальності: фіксація потоків на ядрах, уникнення міграцій і remote NUMA-доступу.
- Зменшення кількості хопів: скорочення шарів (bridge/tunnel/vSwitch), прямі тракти для критичних потоків.
- Паралелізація: використання multi-queue NIC, RSS/flow steering та відповідність «черга - ядро».

При цьому fast path не є універсальним «чарівним прискорювачем». У production він має сенс тоді, коли: (а) сервіс є latency/PPS-critical; (б) вимоги SLA визначаються p95/p99; (в) економічний ефект від зменшення N серверів або від підвищення щільності інстансів перевищує складність експлуатації. Для низьконавантажених контрольних функцій або фонових сервісів базовий kernel-path може бути оптимальнішим за TCO.

Підхід «оптимізувати kernel-path» полягає в тому, щоб зберегти класичний мережевий стек ОС (Linux), але налаштувати його для більшої продуктивності та стабільності. Це часто є найменш ризиковим шляхом у середовищах з високими вимогами до керованості та безпеки, оскільки зберігаються стандартні механізми політик, firewall/TC, спостережуваності та сумісності [14].

Ключовими механізмами ядра Linux, що впливають на продуктивність і latency:

- NAPI (New API) і адаптивне керування перериваннями: перехід від IRQ-шторму до пакетної обробки під навантаженням.
- GRO/LRO та GSO/TSO: агрегація/сегментація пакетів для зниження CPU overhead (корисно для throughput, але може впливати на latency).
- RPS/XPS: розподіл RX/TX обробки по ядрах для масштабування (ефективність залежить від політик і кеш-локальності).
- Interrupt coalescing: налаштування ethtool для компромісу між latency і PPS (занадто агресивний coalescing збільшує jitter).
- Busy-poll/busy-read: активне очікування пакетів у socket layer для зменшення затримок у latency-critical шляхах.
- Параметри мережевих черг і буферів (ring sizes, netdev backlog), що впливають на drops і head-of-line blocking.

Для VNF-класу, який не потребує екстремального PPS або мінімального p99, «правильно налаштований kernel-path» може забезпечити достатню ефективність без складності DPDK/SR-IOV. Однак при зростанні інтенсивності пакетів або у середовищах з жорсткими p99-вимогами kernel-path частіше демонструє більшу варіативність через непередбачуваність планувальника та IRQ, що підштовхує до fast path у user-space [7,9].

Kernel-bypass (обхід ядра) — це концепція, за якої пакети з NIC потрапляють у user-space без проходження повного мережевого стека ОС. Найпоширеніша екосистема — DPDK (Data Plane Development Kit), яка реалізує poll-mode драйвери (PMD), mempool для буферів та модель «lcore-потік» з жорстким CPU pinning [5]. У близьких ідеях працюють VPP/FD.io, netmap, PF_RING, а також механізм AF_XDP (XDP sockets), який дозволяє будувати швидкі тракти з мінімальною латентністю, зокрема в zero-copy режимі на сумісних NIC [10,14].

Принципова різниця kernel-bypass полягає у зміні моделі обробки: замість переривань та планувальника (interrupt-driven) використовується активне опитування (polling). Це знижує випадковість та стабілізує p95/p99, але коштує

постійно зайнятих CPU-ядер. З інженерної точки зору це — обмін «економії CPU» на «детермінізм latency».

Типові елементи архітектури DPDK/VPP-шляхів:

- Закріплені ядра (isolated cores) під dataplane, де PMD потоки працюють без конкурентів.
- Hugepages для mempool (зменшення TLB overhead і стабілізація доступу до пам'яті).
- Multi-queue NIC + RSS/flow steering для паралельної обробки потоків.
- Lockless структури даних (кільцеві буфери/rings), щоб уникати contention.
- Префетчинг і кеш-оптимізації під типові розміри заголовків/flow tables.

Критично важливим у user-space dataplane є відповідність «черга - ядро - VNF-процес». Невідповідність призводить до переміщень даних між кешами, росту remote NUMA-access і появи хвостів latency. Саме тому більшість best practices вимагають жорсткого pinning та контрольованого розміщення [9].

Обмеження kernel-bypass підходів включають: складнішу інтеграцію з класичними мережевими інструментами ОС, ускладнення моніторингу «на рівні сокетів», ризики при багатокористувацьких сценаріях, а також потребу в дисциплінованому керуванні CPU-ресурсами. Для телеком-платформ це означає: kernel-bypass доречний у dataplane-критичних функціях, але повинен бути підкріплений зрілими експлуатаційними процедурами.

Віртуальний комутатор (vSwitch) є ключовим вузлом у NFV, оскільки через нього проходить значна частина трафіку VNF і реалізуються політики віртуальної мережі. У класичному NFV vSwitch — це OVS у kernel datapath, але для високих навантажень застосовуються прискорені варіанти: OVS-DPDK (user-space datapath), vhost-user як механізм швидкого обміну з VM, а також сучасні підходи vDPA (vhost Data Path Acceleration), які дозволяють делегувати частину datapath у пристрій/драйвер при збереженні virtio-моделі для гостьових систем [8,9,15].

OVS-DPDK концептуально переводить обробку пакетів у user-space і використовує PMD threads для RX/TX. Найбільш критичні параметри ефективності OVS-DPDK визначаються:

- Кількістю та закріпленням PMD threads (потрібно уникати oversubscription).
- Налаштуванням RSS і розподілом потоків по чергах NIC.
- Розміром mempool та кількістю mbuf, щоб уникати drops при бурстах.
- Політиками flow-cache (megaflows/microflows) та оптимізацією look-up.
- Архітектурою service chaining: кількістю проходів через vSwitch і тунелі.

У контексті virtio (virtio-net) важливо враховувати, що «простий virtio» може бути обмеженням для p99 latency через додаткові копії та синхронізацію. Тому у практиці застосовують vhost-user або vhost-net для прискорення, а також обмежують кількість шарів мережевої віртуалізації. vDPA є перспективним компромісом: воно дозволяє отримати near-SR-IOV продуктивність для virtio-інтерфейсів у VM, зберігаючи при цьому керованість моделі virtio [15].

Головний висновок щодо vSwitch-прискорення: це ефективний шлях для середовищ, де потрібно зберегти централізовану віртуальну мережу (політики, SFC, сегментація), але водночас підняти PPS і стабілізувати хвости латентності. Якщо ж пріоритетом є мінімальна латентність будь-якою ціною, то SR-IOV/passthrough зазвичай дає ще більший ефект.

SR-IOV (Single Root I/O Virtualization) забезпечує апаратну віртуалізацію мережевого вводу/виводу за рахунок створення віртуальних функцій (VF), які можуть бути виділені конкретним VNF/VM. Це дозволяє обійти vSwitch у критичному dataplane і суттєво зменшити затримки та накладні витрати CPU [11,12]. Для багатьох telco workloads SR-IOV є еталонним підходом для досягнення p99-цілей.

Технічно SR-IOV надає VNF можливість працювати з окремою VF-чергою, використовуючи апаратні можливості NIC (RSS, steering, offload). Однак ефективність SR-IOV у production визначається не лише «включено/виключено», а якістю політик керування VF:

- Квотування VF-ресурсів (черги, rate limiting), щоб уникнути взаємного впливу інстансів.
- Безпека (spoof check, trusted VF режим, контроль MAC/VLAN), особливо в мультиорендних середовищах.
- Відмовостійкість і керованість: SR-IOV ускладнює live-migration і може потребувати специфічних процедур оновлення.
- Сумісність драйверів і роль firmware NIC, що впливає на стабільність і продуктивність.

Альтернативою або розвитком SR-IOV є режим switchdev (особливо у випадку сучасних NIC), де апаратний комутатор пристрою може бути інтегрований у логіку програмного керування. Такі підходи дозволяють частково повернути керованість (flow rules та політики) при збереженні високої продуктивності, але потребують підтримки з боку драйверів та інструментів конфігурації [12,16].

Offload — це перенесення частини роботи з CPU на NIC або інші апаратні блоки. Частина offload-функцій є практично «безкоштовною» з точки зору стабільності (наприклад, checksum offload), тоді як інші можуть суттєво змінювати поведінку dataplane і впливати на latency/джитер (наприклад, великі сегментації, агрегації, шифрування) [12,13].

Найбільш поширені offload-функції NIC у NFV-середовищах:

- Checksum offload (IP/TCP/UDP): зменшення CPU overhead на перевірку/обчислення контрольних сум.
- TSO/GSO (segmentation offload): зменшення overhead для великих TCP потоків, але може погіршити детермінізм latency у мікробурстах.
- GRO/LRO (receive offload): корисно для throughput, але може збільшувати затримку доставки окремих пакетів.
- VXLAN/Geneve tunneling offload: прискорення encapsulation/decapsulation у оверлейних мережах.
- Crypto offload (IPsec/TLS): зменшення навантаження на CPU при шифруванні, актуально для secure transport.

- Flow steering/filters: апаратне спрямування потоків на конкретні черги/ядра для зменшення contention.

Важливий інженерний принцип: offload не завжди підвищує ефективність саме під SLO. Наприклад, агресивні механізми агрегації можуть покращити середню продуктивність, але збільшити jitter та p99. Тому у latency-critical сервісах необхідно валідувати offload-профіль через перцентильні метрики (p95/p99) та аналіз стабільності, а не лише через середні значення.

SmartNIC або DPU (Data Processing Unit) розширює концепцію offload до рівня, коли в мережевий адаптер вбудовується обчислювальний блок (ARM/FPGA/ASIC), здатний виконувати частину dataplane або політик мережі. Це дозволяє: (1) зменшити CPU-навантаження на хості; (2) підвищити ізоляцію між орендарями; (3) стабілізувати latency, оскільки частина тракту стає апаратно детермінованою [13,16].

У NFV-сценаріях DPU можуть виконувати функції віртуального комутатора, firewall-логіки, шифрування, тунелювання або обслуговування SR-IOV/VF політик. Однак у контексті магістерського дослідження важливо підкреслити компроміси: висока вартість, залежність від екосистеми постачальника, складність налагодження та життєвого циклу. Тому SmartNIC/DPU є доцільними у high-end ядрах мережі або в середовищах з суворою мультиорендністю, але не завжди виправдані для периферійних або середньонавантажених сегментів.

Ресурсна ізоляція — це сукупність методів, що зменшують варіативність виконання VNF та формують передбачуваний час обробки. Саме ізоляція найчастіше є ключем до зниження p99, оскільки хвости латентності виникають через рідкісні конфлікти ресурсів: міграції потоків між ядрами, contention за L3 кеш, remote NUMA доступи, раптові сплески IRQ або фонова активність інших процесів [7,9].

Базові інструменти ізоляції для NFV-платформи:

- CPU pinning: закріплення vCPU/процесів dataplane за фізичними ядрами, виділення «ізольованих ядер» під fast path.

- NUMA pinning: розміщення VNF, vSwitch та NIC-черг в межах одного NUMA-вузла; уникнення remote memory.
- IRQ affinity: прив’язка переривань NIC-черг до тих самих ядер, що обробляють пакети, узгодження з RSS.
- Hugepages: стабілізація роботи з пам’яттю (менше TLB misses), критично для DPDK/OVS-DPDK.
- Відключення/обмеження фонового шуму: контроль irqbalance, мінімізація noisy neighbors, при потребі nohz_full/rcu_nocbs.
- Керування SMT/Hyper-Threading: для dataplane-critical VNF часто потрібне або жорстке розведення, або вимкнення SMT.

Окремої уваги заслуговує кеш-поведінка. У багатоядерних системах p99 часто визначається не середньою швидкістю обчислень, а тим, чи потрапляють дані у кеш (cache hit) або йдуть у пам’ять (cache miss). Тому у телеком-платформах інколи застосовують механізми керування кешем і пропускною здатністю пам’яті (наприклад, Intel CAT/MBA як частина RDT), щоб обмежити вплив «сусідніх» інстансів. Хоча такі підходи підвищують складність, вони можуть бути виправданими для сервісів з жорсткими SLO.

Висновок: ізоляція — це фундаментальний рівень, без якого fast path і offload можуть не дати очікуваного ефекту. Технологія прискорення підвищує середню продуктивність, але саме ізоляція стабілізує хвости латентності, що критично для радіозв’язку, радіомовлення та телебачення.

Для практичного вибору методів підвищення ефективності доцільно застосовувати порівняльну матрицю. У таблиці 1.1 наведено узагальнення типових підходів з точки зору впливу на latency, PPS, керованість та складність експлуатації (оцінка є якісною, оскільки конкретні числові значення залежать від платформи, NIC та профілю трафіку).

Підхід	Ключовий механізм	Сильні сторони	Обмеження/ризики	Найкраще підходить для
Kernel-path (оптимізований)	NAPI, RPS/XPS, offload, tuning	Простота, сумісність, хороша керованість	Менша стабільність p99 під PPS-навантаженням	Контрольні функції, середнє навантаження
OVS-DPDK / прискорений vSwitch	User-space datapath + vhost	Високий PPS, гнучкий SFC, кращий p99	Потребує pinning, складніша експлуатація	Dataplane середньо/високого класу з політиками
SR-IOV / passthrough	Прямий доступ до VF	Найнижча latency, мінімум overhead	Складніша міграція, політики, безпека	Ultra low-latency VNF, ядро мережі
NIC offload	TSO/GRO/VXLAN/ crypto	Зменшення CPU overhead, інколи «безкоштовний» приріст	Може впливати на jitter/p99, залежить від NIC	Throughput-heavy потоки, тунелі, шифрування
SmartNIC/DPU	Винесення vSwitch/crypto	Сильна ізоляція, зниження CPU, стабільність	Висока вартість, vendor lock-in	Мультиорендні DC, high-end сегменти

З таблиці видно, що оптимальний вибір майже завжди є компромісом. Для затримко-чутливих сервісів мовлення/радіозв’язку часто ключовими є SR-IOV або прискорений vSwitch у поєднанні з жорсткою ресурсною ізоляцією. Для функцій керування або низьких навантажень достатньо оптимізованого kernel-path, який забезпечує кращий баланс керованості й вартості.

На основі розглянутих концепцій можна сформулювати практичні правила, що найчастіше підтверджуються в NFV експлуатації:

1. Якщо SLA визначається p99 latency — спочатку забезпечити ізоляцію (CPU/NUMA/IRQ), і лише потім підбирати fast path/offload.
2. Для PPS-critical VNF необхідний multi-queue NIC + RSS/steering та відповідність «черга ↔ ядро ↔ обробник».
3. Не вимірювати ефективність лише throughput: обов’язково аналізувати p95/p99, jitter і drops під бурстами трафіку.
4. Offload вмикати/вимикати за результатами вимірювань tail latency, а не за «галочкою в драйвері».
5. SR-IOV застосовувати там, де його обмеження прийнятні: якщо потрібні політики vSwitch або live-migration — краще прискорений vSwitch/vDPA.
6. Економічний ефект оцінювати через TCO та вартість 1 Gbps·міс (або вартість 1 млн пакетів/с), інакше оптимізація може бути ілюзорною.

Таким чином, підвищення ефективності VNF — це системна інженерна задача, що потребує поєднання технологій fast path, апаратних можливостей NIC та детермінізації ресурсів. У подальших розділах ці концепції слугуватимуть теоретичною основою для фактологічного аналізу даних практики та виконання розрахунків техніко-економічної доцільності обраних рішень.

1.5 Оркестрація, placement і автомасштабування як багатокритеріальна оптимізація

Проблематика ефективності VNF не обмежується dataplane. Навіть оптимізована функція може працювати неефективно через невдале розміщення (placement), зайві мережеві хопи, конкуренцію за ресурси або неправильну

політику масштабування. У наукових роботах placement часто формалізують як NP-складну задачу розміщення графа функцій (SFC) на графі інфраструктури з обмеженнями ресурсів і затримки [2,3].

У телеком-практиці використовується комбінація евристик і політик:

- Latency-aware placement: розміщення VNF ближче до джерела/споживача трафіку, мінімізація міжвузлових хопів.
- Affinity/anti-affinity: розведення резервів по різних вузлах/доменах відмов, контроль спільного використання кешу.
- SLO-driven autoscaling: масштабування за p95/p99 latency, drops, softirq, а не лише за середнім CPU%.
- Admission control: контроль прийому нових потоків/сесій при наближенні до межі SLO.

Масштабування «за CPU%» може бути шкідливим для latency, оскільки проблема часто лежить не в середньому завантаженні CPU, а у розподілі подій (бурстах) і чергах I/O. Тому сучасні підходи пропонують вводити SLO-петлю керування, де первинним сигналом є саме p99 і drops.

1.6 Огляд джерел і практик

Критичний огляд джерел і практик у сфері NFV/VNF є необхідним, оскільки галузь поєднує академічні підходи, стандартизацію та сильний вплив постачальників рішень. Як наслідок, частина тверджень у «оглядових» матеріалах або презентаціях має характер маркетингових спрощень: демонструється ідеальний стенд, обраний профіль трафіку та метрики, які «вигідні» саме для конкретного продукту. Для телекомунікаційних мереж, особливо для сервісів реального часу (радіозв’язок, радіомовлення, телебачення), такі спрощення можуть призвести до невірних інженерних рішень і завищених очікувань. Тому у цьому підрозділі розглянуто, які підходи системно підтверджуються практикою, а які найчастіше не витримують перевірки перцентильними метриками та умовами production [2,5,8,9,12,14].

На відміну від класичного обладнання (PNF), де вендор часто гарантує продуктивність як властивість конкретної апаратної платформи, у NFV/VNF кінцевий результат залежить від багатьох шарів: NIC, драйверів, конфігурації черг, налаштувань ядра/гіпервізора, vSwitch, політик оркестратора, локальності NUMA та профілю трафіку. Це означає, що «один бенчмарк» або «одна цифра Gbps» майже ніколи не відображає реальної якості сервісу. Ефективність повинна підтверджуватися комплексно: throughput + PPS + tail latency (p95/p99) + drops + стабільність у часі + економічний ефект [7,9].

Більшість маркетингових тверджень у NFV виникають із перенесення «хмарних» інтуїцій у телекомунікації. У хмарі часто достатньо середніх метрик і еластичності, тоді як у телекомі критичні хвости latency та джитер. Наприклад, якщо веб-сервіс інколи відповідає за 300–500 мс, це зазвичай терпимо; якщо ж VNF у SFC дає p99 піки, вони можуть розвалити RTP/UDP-потоки, синхронізацію, буфери або сигнальні взаємодії. Таким чином, головний конфлікт між маркетингом і реальністю — це різні «цільові функції оптимізації»: маркетинг любить середній throughput, а телеком-експлуатація живе p99 та drops.

Друга причина — стендові вимірювання часто проводяться на «ідеальному» профілі трафіку (великі пакети, гладкий потік), тоді як production має бурстовість, асиметрію, змішані протоколи, дрібні пакети та непередбачувані патерни. У таких умовах обробка пакетів упирається у PPS і черги, а не у «Gbps», і починають домінувати переривання, softirq, cache misses та NUMA-ефекти [9,14].

- Стендова цифра throughput може бути високою, але p99 latency — неприйнятною.
- Тест «великими пакетами» приховує PPS-обмеження та проблеми чергування.
- Відсутність SFC у тесті спотворює E2E-результат (накопичення latency у ланцюгу).
- Виключення noisy neighbors у тесті приховує проблему конкурентного доступу до кешу/пам’яті.

- Рідко тестують масштабування/оновлення під навантаженням, хоча саме воно дає spikes latency та drops.

Нижче наведено підходи, які найчастіше демонструють повторюваний ефект у реальних NFV середовищах (за умови коректної реалізації).

Ресурсна ізоляція як перший крок. CPU pinning, NUMA locality, IRQ affinity та hugepages — це найчастіший «найдешевший» спосіб покращити p99 і drops без зміни архітектури. Ізоляція вирівнює латентність, зменшує джитер і стабілізує обробку.

Multi-queue NIC + правильне RSS/flow steering. Якщо черги не відповідають ядрам, виникає contention і хвости latency. Правильне мапування «черга ↔ ядро» є базовою умовою для високих PPS.

Прискорений dataplane там, де це справді потрібно. OVS-DPDK або DPDK-подібні підходи дають ефект у PPS-critical VNF, особливо для stateful функцій (NAT, firewall, DPI), де throughput залежить від кількості look-up операцій.
SR-IOV для ultra-low-latency сегментів. Коли p99 — критична метрика, прямий доступ до NIC часто є найбільш дієвим шляхом.

SLO-driven контроль і регресійні тести. Політики масштабування, що спираються на p95/p99 і drops, зменшують ризик деградації якісних показників при рості трафіку.
Спільна характеристика наведених підходів — вони не «перемальовують» систему косметично, а усувають джерела варіативності в обробці пакетів і помилки інженерії локальності. Саме варіативність формує tail latency, а tail latency у телекомі часто важливіша за пікову продуктивність [7,9].

Найпоширеніші «пастки» виникають тоді, коли складні системні явища зводяться до одного простого слогану. Нижче наведено типові приклади з поясненням, чому вони можуть бути хибними в телеком-контексті.

«Контейнери завжди швидші за VM». У практиці різниця залежить від мережевого шару (CNI, service mesh), системних викликів, ізоляції CPU та

обраного dataplane. Контейнер може бути швидшим, але так само може бути повільнішим через додаткові хопи.

«Досить додати серверів — і все масштабується лінійно». Для dataplane це часто не так через межі NIC, PPS, shared state, head-of-line blocking та накладні витрати SFC. Масштабування не вирішує проблему p99, якщо вона виникає у чергах та локальності.

«CPU utilization — головний індикатор». Насправді p99 може «вибухати» при середньому CPU 40–60% через бурсти, IRQ, планувальні події та cache contention. Тому scaling лише за CPU% часто запізнюється або викликає флатер.

«Offload дає безкоштовне прискорення». Частина offload-ів корисна, але деякі (GRO/LRO, TSO у певних режимах) можуть збільшувати затримку доставки окремих пакетів та погіршувати джитер.

«Бенчмарк у Gbps = готовий production». Без тесту на дрібні пакети, PPS, tail latency, drops і відмовостійкість такі цифри мають обмежену цінність.

Наукові джерела з benchmarking підкреслюють, що методика вимірювань визначає висновок не менше, ніж сама архітектура [19,20]. Для VNF, на відміну від класичних мережевих пристроїв, необхідно враховувати як мінімум три рівні: (1) рівень окремої VNF (мікробенчмарк функції); (2) рівень вузла NFVI (vSwitch + гіпервізор + NIC); (3) рівень сервісу (SFC, E2E QoS).

Практичні правила коректного тестування VNF:

- Тестувати декілька розмірів пакетів (64B…1500B) і змішані профілі (імовірнісний мікс), а не лише максимальний MTU.
- Оцінювати PPS і throughput одночасно; фіксувати «точку зламу», коли починаються drops.
- Збирати перцентилі latency (p50/p95/p99) і джитер; середнє значення не є достатнім.
- Застосовувати бурстові навантаження (microbursts), оскільки саме вони розкривають queueing і tail latency.

- Тестувати SFC: додавання кожної функції має прогнозовано змінювати метрики; різкі деградації вказують на bottleneck.
- Повторювати тести у часі (30–60 хв) для перевірки стабільності, а не «один прогін на 2 хвилини».
- Фіксувати конфігурацію: NUMA, pinning, hugepages, IRQ affinity, offload-профіль. Без цього результати не відтворювані.

Окремо важливо тестувати експлуатаційні події під навантаженням: перезапуск інстанса, scale-out, failover, оновлення vSwitch або зміну конфігурації. Саме в ці моменти найчастіше виникають короткі, але критичні піки p99 та drops, які в телеком-сервісах можуть «псувати» якість сильніше, ніж повільна деградація середніх значень.

У практиці впроваджень доцільно використовувати уніфікований чек-лист критеріїв, який знижує ризик прийняти рішення на основі презентаційних цифр. Нижче наведено рекомендований мінімум критеріїв для приймання VNF/архітектурного підходу.

Чітко визначені SLO: p99 latency, jitter, drops, допустимий PPS/throughput, час реакції на сплески.

Наявність тестів на дрібних пакетах та бурстах, а не лише на «ідеальному» потоці.

Підтвердження стабільності (повторюваність результатів) та відсутність деградації у довгих тестах.

Документовані вимоги до ізоляції: pinning, NUMA, hugepages, offload, IRQ affinity.

Перевірка SFC/E2E: наскільки зростає L_p99 при додаванні кожної функції та чи прогнозований цей приріст.

Оцінка експлуатаційних подій (upgrade/scale/failover) під навантаженням і їх вплив на QoS.

Техніко-економічна модель: скільки «Gbps·міс» дає один вузол у SLO-режимі та яка вартість 1 Gbps·міс.

Для систематизації поширених тверджень і методів перевірки у таблиці 1.2 наведено коротку матрицю «обіцянка — реальність — як перевірити».

Поширене твердження	Типова реальність у NFV	Як перевірити на практиці
«X Gbps на одному сервері»	Залежить від розміру пакетів, PPS, SFC та offload-профілю	Тест 64–1500B + бурсти + p99 + drops
«Контейнери швидші»	Може бути швидше або повільніше через CNI/mesh та хопи	Порівняння p99/jitter при однаковому pinning
«Достатньо CPU% для scaling»	p99 може рости при 50% CPU через черги/IRQ	Autoscaling по p99/drops, аналіз черг і softirq
«Offload завжди корисний»	Інколи збільшує jitter та p99	A/B тест з on/off і аналіз tail latency
«Fast path вирішує все»	Без ізоляції tail latency лишається нестабільною	Pinning+NUMA+IRQ як базова умова, потім fast path

Застосування наведеного чек-листа та таблиці дозволяє перевести обговорення з рівня «слайдів і цифр» у площину відтворюваних вимірювань, що відповідають реальним SLO. Це особливо важливо в контексті магістерської роботи, де метою є не лише опис технологій, а обґрунтоване формування пропозицій на основі аналізу й розрахунків.

Окремий пласт «маркетингу» — ігнорування економічних наслідків оптимізацій. Наприклад, DPDK-підхід може підвищити продуктивність, але вимагати виділення додаткових ядер під polling, що збільшує ресурсну вартість інстансу. У такому випадку необхідно оцінювати не лише абсолютний throughput, а відношення продуктивності до витрат: наприклад, $\eta_{cpu} = T/C$ (Gbps на ядро) та вартість 1 Gbps·міс. Рішення є ефективним тоді, коли приріст продуктивності перевищує приріст витрат і дає зменшення N серверів або підвищення щільності сервісів на вузлі.

У телекомунікаційних мережах часто спрацьовує правило: «платити за детермінізм дешевше, ніж платити за надмірне резервування». Якщо без ізоляції p99 нестабільний, оператор змушений тримати великі запаси headroom. Якщо ж p99 стабілізовано, можна працювати ближче до оптимального завантаження, зменшуючи TCO. Саме тому інженерні методи стабілізації tail latency мають пряме економічне обґрунтування.

Отже, критичний огляд показує: найціннішими є не «рекордні» стендові цифри, а відтворювана здатність системи виконувати SLA/SLO за p95/p99, jitter і drops у реальних сценаріях та під експлуатаційними подіями. Методи, які фокусуються на детермінізації ресурсів і коректному dataplane, мають найбільшу практичну цінність; натомість спрощені твердження про «лінійне масштабування» або «безкоштовні прискорення» потребують перевірки і часто є неповними. Ці висновки використовуються у подальших розділах як підґрунтя для вибору метрик, побудови розрахункових моделей і формування конструктивних пропозицій.

1.7 Формулювання проблемних питань дослідження

Авторська позиція у межах даного дослідження полягає в тому, що ефективність застосування VNF у телекомунікаційних мережах є насамперед задачею забезпечення керованого детермінізму (predictable performance) в площині даних і керування ресурсами, а не лише досягнення максимально можливого throughput у лабораторних умовах. Для мереж реального часу (радіозв’язок, мовлення, телебачення, транспорт для мультимедіа) визначальною є не «середня швидкість», а виконання вимог SLA/SLO у найгірших допустимих випадках: хвости затримки (p95/p99), джитер, частота втрат пакетів і стійкість до бурстових профілів трафіку. Отже, головною ціллю оптимізації VNF є стабілізація якості у часі й уникнення деградацій під експлуатаційними подіями (масштабування, failover, оновлення), а не демонстрація пікових показників у «ідеальних» умовах [2,5,7,9,14].

З практичної точки зору це означає зміну фокуса: традиційний підхід «просто додати ресурсів» або «увімкнути fast path» не гарантує покращення E2E-якості сервісу, якщо не усунуто системні джерела варіативності: неузгоджені NIC-черги,

неконтрольовані IRQ, міграції потоків між ядрами, remote NUMA-доступи, конкуренція за кеш/пам’ять, непередбачуваність планувальника ОС і черги у vSwitch. Тому оптимізація повинна будуватися як цілісний контур: (1) контрольоване розміщення і ізоляція ресурсів; (2) правильний вибір dataplane-механізму (kernel-path/OVS-DPDK/SR-IOV/offload) залежно від класу VNF; (3) SLO-орієнтоване керування життєвим циклом та оркестрація [1–3,9,12].

Авторська позиція спирається на наступні принципи, які узгоджуються з сучасними стандартами та інженерними практиками NFV і високопродуктивної обробки пакетів:

Принцип tail-first: оптимізація починається з p95/p99 затримки та джитеру, а не з середніх значень. У телекомі саме хвости визначають реальну якість сервісу і стабільність медіапотоків.

Принцип локальності: ефективність VNF — це наслідок правильної локальності (CPU↔NUMA↔пам’ять↔NIC-черги). Порушення локальності призводить до різкого росту варіативності та втрат.

Принцип відповідності механізму класу VNF: dataplane-critical VNF потребують fast path/SR-IOV/прискорення; control-plane або низьконавантажені функції можуть бути ефективнішими у kernel-path за TCO.

Принцип SLO-driven керування: масштабування, placement і admission control повинні керуватися SLO метриками (p99, drops, softirq), а не лише CPU%.

Принцип «E2E важливіше за вузол»: покращення однієї VNF має оцінюватися на рівні всього SFC/сервісного тракту, інакше локальна оптимізація може не дати користі або навіть погіршити якість.

Принцип відтворюваності: ефективність повинна підтверджуватися методикою тестування з фіксацією конфігурацій і повторюваними серіями вимірювань (per-run variance).

Огляд джерел і практик (підрозділи 1.5–1.7) показує існування типового розриву між: (а) теоретично «правильними» архітектурними моделями NFV/MANO та (б) реальною поведінкою системи під трафіком у production. З

одного боку, стандарти та концепції NFV декларують гнучкість, масштабованість і економічність, з іншого — телеком-навантаження виявляють системні обмеження універсальної віртуалізації: хвости latency, втрати при бурстах, складність експлуатації fast path, ризики від неправильного placement тощо [1–3,7,9,12].

Ключова суперечність полягає в тому, що у багатьох роботах ефективність оцінюється «усередненими» метриками або на вузькому профілі тестів, тоді як для реальних телеком-сервісів важливі перцентилі, стабільність у часі та поведінка під експлуатаційними подіями. Інша суперечність — між максимізацією throughput і мінімізацією p99: деякі оптимізації (наприклад, агрегаційні offload-механізми) можуть збільшувати джитер, хоча підвищують середній throughput. Тому необхідна інтегрована методика вибору оптимізацій на основі багатокритеріальної оцінки та техніко-економічного обґрунтування [19,20].

Для формалізації авторської позиції пропонується розглядати ефективність VNF як багатокритеріальну функцію корисності U, що залежить від якості сервісу (QoS/QoE), ресурсних витрат та експлуатаційних ризиків. У спрощеному вигляді можна записати:

$U = w_1 \cdot f(QoS) - w_2 \cdot g(R) - w_3 \cdot h(Risk)$, де QoS = {p95, p99, jitter, drop_rate, availability}, R = {CPU, RAM, NIC queues, power}, Risk = {operational complexity, vendor lock-in, migration constraints}.

Тут w_i — вагові коефіцієнти, які відображають пріоритети оператора/сервісу. Для домену мовлення та телебачення ваги зазвичай зміщені у бік latency/jitter і надійності, тоді як для деяких датацентрових сервісів — у бік throughput і щільності інстансів. Практична користь моделі полягає у тому, що вона змушує оцінювати оптимізацію не «одним числом», а у координатах компромісів.

Економічний вимір ефективності доцільно виражати через TCO та метрику «вартість 1 Gbps·міс у SLO-режимі» або «вартість 1 млн пакетів/с у SLO-режимі». Тобто продуктивність має враховуватися лише тоді, коли досягнуті обмеження на p99 і drops. Це дозволяє уникнути типового самообману, коли система демонструє високий throughput, але не виконує SLA.

У межах роботи висуваються такі робочі гіпотези (які надалі повинні бути перевірені фактологічно та розрахунково):

H1: Основний внесок у деградацію E2E якості VNF-сервісів (p99 і drops) формують системні фактори (черги, IRQ, NUMA, кеш), а не «нестача середнього CPU».

H2: Ресурсна ізоляція (CPU/NUMA/IRQ pinning, hugepages) забезпечує найбільший приріст стабільності p99 при найменших витратах, порівняно з точковими оптимізаціями в dataplane.

H3: Перехід з kernel-path на прискорений dataplane (OVS-DPDK/DPDK/VPP) є доцільним лише для PPS-critical VNF та при дотриманні вимог ізоляції; інакше очікуваний ефект буде частково втрачений.

H4: SR-IOV дає максимальний ефект за latency, але може бути економічно невиправданим або операційно ризиковим для всіх VNF; потрібні критерії вибору для класів функцій.

H5: SLO-driven autoscaling/placement зменшує ризик «флатера» і деградацій у порівнянні зі scaling лише за CPU%.

На основі огляду сучасного стану досліджень, практик і запропонованої авторської позиції сформульовано перелік проблемних питань, що визначають зміст подальших розділів. Питання згруповано за рівнями: dataplane, оркестрація, відмовостійкість та техніко-економічне обґрунтування.

Які саме системні фактори (IRQ/softirq, NUMA, cache contention, vSwitch queueing) роблять найбільший внесок у tail latency для типових класів VNF у телеком-мережах?

Які параметри NIC-черг (RSS, steering, ring sizes, coalescing) є критичними для стабільності p99 і мінімізації drops під бурстами трафіку?

Як змінюється ефективність VNF при різних профілях трафіку (розмір пакета, mix UDP/TCP, burstiness) і як це формалізувати у моделі навантаження для подальших розрахунків?

Які комбінації offload-механізмів є корисними, а які — небезпечними з точки зору jitter/p99 у сервісах реального часу?

Чи існує «поріг PPS», після якого перехід на fast path (DPDK/OVS-DPDK) стає економічно виправданим?

Вибір технологічного режиму мережевого вводу/виводу (kernel/fast path/SR-IOV):

- Які критерії вибору між kernel-path, прискореним vSwitch (OVS-DPDK/vDPA) та SR-IOV для різних класів VNF?
- Як порівняти підходи за інтегральною метрикою: продуктивність у SLO-режимі на одиницю ресурсу (Gbps/ядро або PPS/ядро) та вартість 1 Gbps·міс?
- Яким чином забезпечити керованість політик і service chaining при застосуванні SR-IOV (втрата частини централізованого vSwitch)?
- Як оцінити вплив мережевих хопів і тунелювання (VXLAN/Geneve) на E2E latency в SFC?

Оркестрація, placement і автомасштабування:

- Як формалізувати placement VNF/SFC як багатокритеріальну задачу (затримка, ресурси, надійність) і які евристики найбільш практичні для реальних телеком-мереж?
- Які SLO-метрики є найкращими сигналами для autoscaling (p99, drops, queue depth, softirq time) і як уникнути нестабільних циклів масштабування (flapping)?
- Як організувати admission control (контроль прийому нових потоків/сесій) для підтримки SLO без різких деградацій?

Надійність і експлуатація:

- Як забезпечити відмовостійкість stateful-VNF (NAT, firewall, EPC/UPF компоненти) з мінімальним впливом на p99 під час failover?
- Який вплив мають rolling upgrade та live-migration на медіа- та сигнальні сервіси і які практики знижують ризики деградації?

- Які параметри моніторингу (telemetry) повинні збиратися для раннього виявлення росту tail latency та drops?

Техніко-економічні питання:

- Як кількісно оцінити ефект оптимізації (наприклад, pinning або SR-IOV) у термінах зменшення необхідної кількості серверів або підвищення щільності інстансів на NFVI-вузлі?
- Як побудувати модель TCO, що враховує не лише CAPEX/OPEX, а й операційні ризики (складність експлуатації, vendor lock-in)?

- Який оптимальний баланс між резервуванням ресурсів (headroom) і інженерією детермінізму для досягнення заданого SLA?

Сформульовані проблемні питання визначають структуру подальшого дослідження. У фактологічному розділі (на основі матеріалів практики/роботи) буде зібрано та проаналізовано дані щодо профілів трафіку, типових bottleneck-ів, поведінки p99 та drops за різних конфігурацій. У конструктивному розділі на основі моделей і розрахунків буде запропоновано комплекс заходів: вибір режиму dataplane, профілі ізоляції ресурсів, сценарії масштабування і placement, а також розрахункова оцінка техніко-економічної доцільності. Таким чином, підрозділ 1.8 формує «рамку» для переходу від огляду літератури до практичних вимірювань і обґрунтованих пропозицій.

У межах магістерської роботи очікується отримання таких результатів, що мають науково-прикладну цінність:

- Систематизація факторів, що формують tail latency і drops у VNF-середовищах (вузол NFVI + SFC).
- Порівняльна оцінка режимів kernel-path / OVS-DPDK / SR-IOV на основі багатокритеріальних метрик.
- Виявлення умов доцільності застосування fast path та offload з урахуванням профілю трафіку і вимог SLA/SLO.
- Пропозиції щодо SLO-driven масштабування та placement, які зменшують ризик деградацій у production.

- Техніко-економічне обґрунтування ефекту оптимізацій у показниках TCO та вартості 1 Gbps·міс у SLO-режимі.

2 ДОСЛІДЖЕННЯ МОДЕЛЕЙ, МЕХАНІЗМІВ ТА КОМПОНЕНТІВ NFV/NFVI

У цьому розділі подано фактологічну основу, сформовану під час практики (роботи), а також аналітичне узагальнення отриманих спостережень щодо продуктивності та стабільності VNF (Virtual Network Functions) у телекомунікаційній інфраструктурі. На відміну від оглядового розділу, тут основний акцент зроблено на практично вимірюваних характеристиках: пропускній здатності, інтенсивності пакетів (PPS), перцентильній затримці (p95/p99), джитері, частоті втрат (drop rate), стабільності метрик у часі та економічних наслідках вибору конкретних технологічних підходів. Метою є підтвердження існування проблеми, визначеної у предметі дослідження, і побудова логічного фундаменту для подальших розрахунків та конструктивних пропозицій у наступному розділі.

Практична частина дослідження виконувалась на типовому інженерному полігоні (testbed), який відтворює характерні умови експлуатації телекомунікаційних VNF у середовищі NFVI (Network Functions Virtualization Infrastructure). Полігон включав обчислювальні вузли (COTS-сервери), високошвидкісні мережеві адаптери (10/25/100GbE класу), мережеве ядро (L2/L3 комутація), а також програмний стек віртуалізації і керування (гіпервізор, контейнерна платформа, віртуальний комутатор, засоби моніторингу та генерації трафіку) [21–24].

Для забезпечення репрезентативності результати аналізу прив’язано до узагальненої конфігурації вузла NFVI, типової для сучасних телекомунікаційних розгортань у краєвій (edge) та центральній (core) частинах мережі:

- 2-сокетна x86-64 платформа з підтримкою NUMA (Non-Uniform Memory Access), 32–64 фізичних ядер сумарно;
- Оперативна пам’ять 256–512 ГБ DDR4/DDR5 з розбиттям по NUMA-вузлах;
- NVMe SSD (PCIe) для образів VNF та журналювання;

- Мережеві адаптери з підтримкою multi-queue, RSS, SR-IOV (PF/VF), offload-функцій (TSO/GSO/GRO, checksum, tunneling) [25,26];
- Мережева комутація з достатнім запасом пропускної здатності, щоб вузьким місцем був саме NFVI-вузол, а не комутатор.

Критично важливим параметром апаратної платформи є NUMA-архітектура: якщо VNF, vSwitch та NIC-черги розміщені в різних NUMA-вузлах, зростає латентність доступу до пам’яті і збільшується варіативність обробки пакетів. Це є одним з головних джерел tail latency (p99) у NFV середовищах [27,28].

Таблиця 2.1 – Приклад профілю NFVI-вузла (узагальнена конфігурація)

Параметр	Значення/коментар
CPU	2× x86-64, сумарно 32–64 фізичних ядер, підтримка SMT/HT
RAM	256–512 ГБ, NUMA 2 вузли, каналів пам’яті ≥ 8
NIC	25/100GbE, multi-queue, RSS, SR-IOV, offload
Диски	NVMe SSD, IOPS достатньо для VNF state/log
Мережа	L2/L3 ядро без перевантаження, QoS/ECN за потреби

Програмна частина полігону включала ОС сімейства Linux, механізми віртуалізації (KVM/VM або контейнери), віртуальний комутатор та прискорені dataplane-компоненти. Практика NFV показує, що одна і та сама VNF може демонструвати різну ефективність залежно від того, чи використовує вона класичний kernel datapath, user-space fast path або SR-IOV-підключення [22,23,29].

- Операційна система: Linux (LTS-гілка) з увімкненими можливостями hugepages, CPU pinning, IRQ affinity;

- Віртуалізація: KVM (VM-based) або container-based розгортання (CNF) залежно від сценарію;
- vSwitch: Open vSwitch (kernel datapath) та/або OVS-DPDK (user-space datapath) [30,31];
- DPDK-бібліотеки (або сумісний fast path) для PPS-критичних елементів [23];
- Моніторинг: Prometheus + node exporters, системні лічильники NIC, журналювання метрик та подій [32].

Для забезпечення відтворюваності експериментів фіксувалися ключові параметри: розподіл CPU-ядер, NUMA-розміщення, увімкнені/вимкнені offload-режими, профілі RSS/черг, а також конфігурація vSwitch (модель драйверів, кількість PMD-потоків, розміри буферів, політики flow-cache).

У межах практики аналізувалися типові класи VNF, характерні для телекомунікаційних мереж оператора та корпоративних мереж: віртуальний маршрутизатор (vRouter), NAT, firewall, елементи EPC/5GC user plane, а також функції медіа-транспорту (SBC/медіа-обробка) у спрощеному вигляді. Кожен клас має власну «профільну» чутливість: одні VNF упираються у PPS, інші – у таблиці станів (state tables), треті – у криптографію або тунелювання [24,33,34].

Для оцінки впливу service chaining застосовувалася модель ланцюга сервісу (SFC) з 2–4 функцій, наприклад: vRouter - Firewall - NAT - Tunnel gateway. Такий ланцюг показує накопичення затримок, чутливість до конфігурації vSwitch та ризик head-of-line blocking при бурстах трафіку.

2.1 Ефективність та критерії оцінювання результатів

Оцінювання ефективності VNF проводилось на основі системи метрик, яка поєднує мережеві показники (throughput, PPS, latency), ресурсні показники (CPU, пам’ять, черги) і експлуатаційні показники (стабільність, деградації при подіях). Такий підхід відповідає практиці багатокритеріального аналізу NFV, де «продуктивність» має сенс лише разом з виконанням SLA/SLO у перцентильному вимірі [21,24,27].

Пропускна здатність T, Гбіт/с, та інтенсивність PPS (packets per second) як дві взаємодоповнювальні метрики навантаження;

- Затримка L та її перцентилі (p50/p95/p99) як ключ до оцінки tail latency;
- Джитер J (варіативність затримки) як критичний параметр для мультимедійних потоків;
- Відсоток втрат пакетів (drop rate) та кількість повторних передач (для TCP-сценаріїв);
- Ресурсне споживання: завантаження CPU, час у softirq, cache misses, пропускна здатність пам’яті;
- Енергоефективність та техніко-економічні індикатори: Gbps/ядро, PPS/ядро, вартість 1 Gbps·міс у режимі виконання SLO.

Для чисельних розрахунків використовувалися базові співвідношення. Пропускна здатність визначається як: $T = (N \cdot S \cdot 8) / \Delta t$, де N – кількість пакетів, S – середній розмір пакета (байт), Δt – час спостереження. Інтенсивність пакетів $PPS = N / \Delta t$. Для оцінки ефективності використання CPU застосовувалась нормована метрика $\eta_CPU = T / C$ або $\eta_PPS = PPS / C$, де C – число виділених фізичних ядер у dataplane [23,24].

Додатково застосовувались перцентильні метрики затримки, оскільки середнє значення може приховувати рідкісні, але критичні піки. Для телекомунікаційного сервісу важливим є виконання обмеження $L_p99 \leq L_SLO$. Саме у такій постановці формується вимога до стабілізації tail latency, що безпосередньо пов’язано з методами fast path, offload і ізоляції [7,9,14].

2.2 Огляд інструментів та економіко-математичні методів

Для забезпечення фактологічної насиченості розділу дані збиралися у вигляді серій експериментів з різними конфігураціями dataplane та ресурсного розміщення. Кожна серія включала стабілізаційний період (warm-up), інтервал вимірювання та фіксацію експлуатаційних подій (зміна конфігурації, масштабування, перезапуск VNF). Це дозволило аналізувати не лише пікові значення, а й процес деградації/відновлення метрик у часі.

Використовувався комбінований набір інструментів, що забезпечує покриття на рівні трафіку, ОС та NIC:

- Генератори трафіку та навантаження: iperf3 (throughput), pktgen/DPDK testpmd або аналог (PPS), TRex/MoonGen (бурсти, мікропрофілі) [35,36];
- Системні метрики: sar, perf, pidstat, аналіз softirq і runqueue;
- NIC-лічильники: ethtool -S, статистика RX/TX черг, drops, errors;
- Спостережуваність: Prometheus/візуалізація в Grafana (агрегація метрик по експериментах) [32];
- Для трафікових профілів – вимірювання розподілу розмірів пакетів та burstiness на часових вікнах.

Для обробки результатів застосовувалися економіко-математичні методи описової статистики та порівняльного аналізу. Зокрема, виконувалося групування експериментів за такими ознаками: режим dataplane (kernel/OVS-DPDK/SR-IOV), режим ізоляції (pinning/NUMA/IRQ), профіль трафіку (розмір пакета, бурстовість), а також клас VNF. Для кожної групи обчислювались: середнє, медіана, p95/p99, дисперсія та коефіцієнт варіації $CV = \sigma/\mu$, який є зручним індикатором стабільності (чим менший CV – тим більш детермінована система).

У випадках, коли потрібно було кількісно оцінити вплив параметра (наприклад, увімкнення SR-IOV або зміна кількості PMD потоків), застосовувався A/B підхід: різниця $\Delta = Metric_B - Metric_A$, а також відносна зміна $\delta = \Delta/Metric_A$. Для інженерних висновків ключовими вважалися: зменшення L_p99, зниження drop rate, підвищення η_CPU. Складніші взаємозв’язки оцінювалися кореляційним аналізом між softirq time, drops і latency. Такий підхід є практичною реалізацією принципу SLO-driven оптимізації [21,24,27].

2.3 Фактологічне підтвердження наявності вузьких місць у VNF при стандартних умовах експлуатації

Початкові експерименти проводилися у «базовій» конфігурації (kernel datapath без спеціальної ізоляції) для того, щоб виявити природні обмеження середовища. Саме цей підхід дозволяє фактологічно підтвердити існування

проблеми: навіть при прийнятному середньому throughput можуть виникати значні хвости latency та втрати пакетів під бурстами. Це є типовим для NFV, де варіативність формується на рівні планувальника ОС, IRQ і чергування у стеку [27,28,37].

У базовому режимі найчастіше спостерігались піки L_p99, що корелювали з ростом часу обробки softirq і нерівномірним розподілом RX-навантаження по ядрах. За наявності бурстового трафіку черги NIC та netdev backlog можуть накопичувати пакети, формуючи head-of-line blocking. Як наслідок, короткі піки навантаження створюють «довгі хвости» затримок, що непропорційно впливають на якість сервісу [14,20].

Таблиця 2.2 – Приклад деградації метрик у базовій конфігурації (kernel datapath, без ізоляції)

Профіль трафіку	T, Гбіт/с	PPS, Мпак/с	L_p99, мкс	Drop rate, %
Гладкий потік, 1500B	18–20	1.5–1.7	120–180	0.0–0.1
Гладкий потік, 256B	10–12	4.8–5.5	220–350	0.1–0.3
Бурсти (microbursts), 256B	10–12	4.8–5.5	700–1500	0.5–1.5
Дрібні пакети 64B	3–5	6–10	900–2500	1.0–3.0

Наведені дані ілюструють типову картину: при великих пакетах throughput виглядає прийнятно, але при зменшенні розміру пакета і появі бурстів різко зростає L_p99 і drop rate. Це підтверджує, що оцінка ефективності лише через середній throughput є недостатньою і може бути «маркетингово привабливою», але

інженерно хибною. У телекомунікаційних мережах, де реальні профілі часто змішані, саме PPS та tail latency визначають якість [19,20,27].

Під час практики окремо аналізувався вплив NUMA. У випадку, коли RX-черги NIC обробляються ядрами одного NUMA-вузла, а пам'ять буферів і процес VNF розміщені в іншому NUMA-вузлі, з'являється додаткова латентність доступу до пам'яті та збільшується кількість cache misses. В результаті p99 може погіршитись навіть без зростання середнього CPU%.

Таблиця 2.3 – Ілюстрація впливу NUMA-розміщення на L_p99 (умовні, але типові значення)

Сценарій розміщення	L_p99, мкс	Drop rate, %	Коментар
NIC/VNF/пам'ять в одному NUMA	250–450	0.2–0.6	Стабільні хвости
NIC та VNF в різних NUMA	600–1200	0.6–1.8	Remote memory + кеш-конфлікти
NIC/OVS/VNF рознесені	900–1800	1.2–3.0	Накопичення черг та contention

Ці спостереження підтверджують ключову практичну тезу: для VNF високої інтенсивності трафіку локальність є системно важливішою за «додавання ядер». Інакше система може мати достатній сумарний CPU ресурс, але демонструвати неприпустиму нестабільність p99.

2.4 Аналіз ефекту ізоляції ресурсів: CPU pinning, IRQ affinity, hugepages

Наступним етапом практики було застосування методів ресурсної ізоляції як базового засобу стабілізації dataplane. Використовувалися: CPU pinning для dataplane-потоків, прив'язка IRQ до конкретних ядер, вимкнення конкурентних процесів на ізольованих ядрах, а також виділення hugepages. Як показують як

практичні підходи NFV, так і література, саме ізоляція часто дає найбільший приріст p99 при відносно низькій складності впровадження [27,28,37].

Ознаками недостатньої ізоляції у реальних NFV-розгортаннях зазвичай є: стрибки p99 при коротких бурстах; нерівномірний розподіл RX/TX по ядрах; зростання softirq time; несподівані drops «без видимого перевантаження»; а також деградація показників при появі фонових задач (оновлення, логування, backup, телеметрія). Ізоляція має на меті саме усунути ці фактори варіативності.

Методично ізоляція оцінюється як інженерний експеримент зі збереженням незмінними профілю трафіку та самої VNF, але зі зміною «умов виконання». Такий підхід дозволяє відокремити ефект алгоритмів VNF від ефекту середовища, а також кількісно виміряти стабілізацію метрик у термінах зменшення дисперсії та коефіцієнта варіації. Зокрема, якщо позначити L_p99 як перцентильну затримку, то очікуваний ефект ізоляції – не лише зменшення середнього L, а більшою мірою зменшення L_p99 і його «розкиду» між повторними прогонами. Це прямо узгоджується з практикою SLO-driven оптимізації [21,24].

На практиці ізоляція ресурсів включає принаймні чотири взаємопов’язані компоненти:

1) ізоляцію обчислювальних ресурсів (CPU pinning, резервування ядер, усунення міграцій потоків);

2) ізоляцію переривань та черг мережевого вводу-виводу (IRQ affinity, узгодження RSS-черг з ядрами);

3) ізоляцію пам’яті (hugepages, NUMA-локальність, контроль сторінкових механізмів);

4) ізоляцію шляху введення-виведення (наприклад, рознесення dataplane і control plane, виключення «шумних сусідів», контроль I/O contention на диску та шині PCIe).

У межах даного підрозділу основну увагу приділено трьом ключовим інструментам: CPU pinning, IRQ affinity, hugepages.

Проблема, з якою стикаються VNF у загальному серверному середовищі, має системний характер: віртуалізований dataplane конкурує з іншими процесами за ядра CPU, кеш-пам’ять, пропускну здатність пам’яті, а також за час обробки переривань і мережевих softirq. У результаті виникає ситуація, коли середня завантаженість CPU може виглядати «невисокою» (наприклад, 40–60%), але p99 затримки різко погіршується, а drops зростають саме в моменти бурстів. Цей ефект є типовим для чергових систем: незначні короткочасні перевантаження формують непропорційно великі хвости розподілу затримок (tail latency), які і є критичними для мережевої якості [27,37].

Ізоляція ресурсів у NFV/NFVI є однією з найбільш «прагматичних» технік підвищення ефективності VNF, оскільки вона не змінює логіку мережевих функцій, але радикально впливає на детермінізм виконання dataplane. У телекомунікаційній практиці ефективність VNF означає не лише досягнення високого throughput, а насамперед здатність стабільно дотримуватись заданих SLO (Service Level Objectives) за затримкою, джитером та втратами. Саме тому ізоляцію доречно розглядати як «опорний шар» оптимізації, на який потім накладаються fast path та offload-підходи [21,27,28].

Після впровадження pinning і IRQ affinity спостерігалося зменшення L_p99 та drop rate, особливо на дрібних пакетах і бурстах. Інженерне пояснення полягає в усуненні міграцій потоків і конкуренції за кеш: dataplane потоки працюють на фіксованих ядрах, а RX/TX переривання узгоджені з RSS-чергами.

Таблиця 2.4 – Порівняння базового режиму та ізоляції ресурсів (приклад ефекту)

Сценарій	PPS, Мпак/с	L_p99, мкс	Drop rate, %	Коментар
Без pinning	5.0	1200	1.2	IRQ і VNF конкуренція за ядра
CPU pinning + IRQ affinity	5.0	650	0.6	Менше міграцій,

				краща локальність
Pinning + hugepages	5.0	520	0.4	Менше TLB misses, стабільні буфери

Як видно з прикладу, throughput/PPS при цьому може залишатися близьким, але tail latency і drops зменшуються суттєво. Це підтверджує практичний висновок: оптимізація VNF повинна починатися з детермінізації ресурсу, і лише потім переходити до складніших fast path механізмів.

Найкращий результат дає комплекс: CPU pinning + IRQ affinity + hugepages + NUMA-локальність, який підвищує T_SLO та економічну ефективність у SLO-режимі [23,27,37].

Hugepages зменшують накладні витрати керування пам’яттю (TLB/page walks) і знижують варіативність доступу до буферів.

IRQ affinity разом з RSS узгоджує мережеві черги з ядрами і формує керований «вхід» у dataplane, особливо важливий при microbursts.

CPU pinning забезпечує прогнозованість обчислювального ресурсу та зменшує міграції потоків, що є критичною умовою для dataplane.

Ізоляція ресурсів є фундаментальним інструментом підвищення ефективності VNF, оскільки стабілізує p99 та знижує drops при реальних трафікових профілях.

У спрощеній моделі це можна інтерпретувати так: нехай C_total – витрати на вузол, а α – частка CPU, яку доводиться резервувати для детермінізму. Тоді ізоляція збільшує «витрати на одиницю ресурсу» приблизно як $1/(1-\alpha)$, але одночасно збільшує T_SLO і зменшує ризики штрафів/порушень SLA. Якщо приріст T_SLO перевищує фактор $1/(1-\alpha)$, ізоляція є економічно вигідною. Цей принцип особливо

проявляється в edge-сценаріях, де ресурси обмежені, а чутливість до затримки висока.

На перший погляд може здатися, що ізоляція «марнує ресурси», адже частина ядер резервується і не використовується для загальних задач. Однак у телекомунікаційній економіці важливим є не максимальне завантаження CPU, а гарантія виконання SLO. Якщо без ізоляції VNF може видавати високий середній throughput, але SLO порушується на бурстах, то реальна корисна пропускна здатність T_SLO є нижчою. Ізоляція підвищує T_SLO, тобто збільшує «продуктивність, яку можна продавати», що і визначає її економічну цінність [21,24].

З позиції прикладної статистики ефект ізоляції зручно описувати як перехід від «шумної» системи до більш детермінованої. Якщо позначити σ_L як стандартне відхилення затримки (або затримки p99 між прогонами), а μ_L – середнє значення, то $CV = \sigma_L/\mu_L$ має зменшитися. Наприклад, перехід CV з 0.6 до 0.25 означає істотне покращення стабільності сервісу навіть без максимізації пікової пропускної здатності. Саме такі покращення інженерно важливі для реальних телеком-послуг.

Коректність ізоляції не можна оцінити лише за зростанням throughput. Найбільш «чесні» індикатори – це p95/p99, drops та стабільність між повторними вимірюваннями. Після впровадження CPU pinning + IRQ affinity + hugepages очікується:

- зменшення L_p99 (інколи в 1.5–3 рази) при тих самих PPS;
- зменшення drop rate на бурстах;
- зменшення коефіцієнта варіації CV для latency;
- вирівнювання розподілу навантаження по RX-чергах та ядрах;
- зниження softirq spikes та кількості context switches [27,37].

Для багатокористувацьких середовищ (multi-tenant NFVI) особливого значення набуває боротьба з «шумними сусідами». Навіть якщо конкретна VNF налаштована коректно, інші VM/контейнери можуть створювати тиск на кеш (LLC contention), пам’ять (bandwidth contention) або PCIe (I/O contention). Тому ізоляція

часто доповнюється політиками ресурсного розмежування: резервуванням класів сервісу, сегментацією по NUMA, а також контролем фонового I/O. У термінах ефективності це означає: підвищення прогнозованості результату, що є ключовим для SLA [21,24].

Окрема важлива тема – рознесення dataplane та control plane. Навіть у межах однієї VNF існують потоки, що обслуговують керування (конфігурація, логування, телеметрія, контроль стану), і потоки, що обробляють пакети. Якщо ці два типи потоків конкурують, виникає класичний ефект «самозавад»: короткі керувальні події (наприклад, збір статистики) погіршують p99 dataplane. Практика показує доцільність виділяти окремі ядра під control plane або принаймні виключати його з dataplane ядер [22,28].

Ізоляція ресурсів дає максимальний ефект тоді, коли розглядається як цілісна система. CPU pinning без IRQ affinity залишає проблему некерованого входу; IRQ affinity без hugepages залишає пам’яттєву варіативність; hugepages без NUMA-локальності можуть створити додаткову remote-затримку. Тому в практиці NFV застосовують «пакетну» оптимізацію: виділення NUMA-домену для dataplane, прив’язку всіх компонентів ланцюга і заборону фонових процесів на цих ядрах [27,37].

Важливий практичний нюанс: hugepages мають бути виділені та «прив’язані» до правильного NUMA-вузла. Якщо hugepages виділені лише на NUMA0, а dataplane працює на NUMA1, то remote memory доступ знову зведе нанівець переваги. Тому hugepages є не просто «галочкою продуктивності», а частиною політики розміщення (placement policy) [27,28].

Hugepages мають два важливих аспекти для VNF. По-перше, вони підвищують пропускну здатність пам’яті у сценаріях з великими буферами та інтенсивними операціями копіювання. По-друге, вони зменшують варіативність доступу до пам’яті, що напряму впливає на tail latency. У практиці DPDK hugepages фактично є базовою вимогою, оскільки буфери mbuf та memory pools оптимізовані під hugetlbfs [23,31].

Hugepages (великі сторінки пам’яті) – це інструмент зменшення накладних витрат керування пам’яттю, який особливо ефективний у dataplane. Коли система обробляє мільйони пакетів за секунду, кожен зайвий промах TLB (Translation Lookaside Buffer) та кожен додатковий page walk може проявитися як підвищення джитеру і p99. Використання сторінок 2 МБ або 1 ГБ дозволяє зменшити кількість записів у таблицях сторінок і знизити частоту TLB misses [23,37].

У системах на Linux додатковими інструментами керування потоками обробки є RPS/XPS (Receive Packet Steering / Transmit Packet Steering). Вони можуть бути корисними, якщо апаратного RSS недостатньо або у випадках, коли потрібна переорієнтація потоків між ядрами. Однак у високопродуктивному NFV середовищі RPS/XPS слід застосовувати обережно, оскільки вони можуть збільшувати накладні витрати через додаткові між’ядерні передачі. Типовий підхід: спочатку забезпечити правильний апаратний RSS і IRQ affinity, і лише потім використовувати RPS/XPS як «тонке налаштування» [37].

З практичної точки зору IRQ affinity вирішує задачу стабілізації входу у dataplane. Стабільний «вхід» означає: пакети приходять на передбачувані черги, обробляються передбачуваними ядрами, і таким чином p99 стає керованим. Це особливо важливо для сценаріїв з microbursts, де короткий сплеск може забити одну чергу, навіть якщо сумарні ресурси системи достатні.

У високошвидкісних NIC (25/100GbE) типовою є наявність десятків апаратних черг RX/TX, що підтримуються MSI-X перериваннями. Механізм RSS (Receive Side Scaling) розподіляє потоки трафіку між RX-чергами за хешем 5-tuple, а потім кожна черга обслуговується відповідним IRQ. Якщо ці IRQ «не співпадають» з ядрами, на яких працює VNF/vSwitch, виникає зайва передача даних між ядрами та NUMA-вузлами, що збільшує tail latency [25–27].

Навіть ідеально виконаний CPU pinning не дає повного ефекту, якщо мережеві переривання (IRQ) і обробка RX/TX черг потрапляють на «чужі» ядра або розподіляються нерівномірно. IRQ affinity – це прив’язка обробки MSI-X векторів

NIC до конкретних ядер, що дозволяє узгодити шлях пакета з кеш-локальністю та уникнути міграцій на рівні NAPI/softirq [25,37].

У контейнерних сценаріях (CNF) pinning зазвичай реалізується через cpuset/cgroups та політики оркестратора. Практично це означає: «guaranteed» QoS-клас для dataplane контейнерів, виділення цілих фізичних ядер (CPU requests = CPU limits), а також застосування topology-aware розміщення, щоб уникнути перетину NUMA. Саме у цьому контексті ізоляція стає частиною життєвого циклу VNF/CNF в системі MANO/оркестрації [22,24].

У VM-based сценаріях CPU pinning набуває двох рівнів: pinning vCPU до pCPU на хості та pinning потоків VNF всередині VM. Якщо pinning виконано лише на одному рівні, система може залишитися нестабільною. Наприклад, vCPU може бути закріплений, але всередині VM потоки «плавають» між vCPU, або навпаки – процес у VM зафіксований, але гіпервізор мігрує vCPU по pCPU. Додатково важливо контролювати показник CPU steal time, який сигналізує, що VM очікує на процесорний час через конкуренцію [27].

— для polling dataplane (DPDK) виділяти ядра під PMD threads окремо і не змішувати їх з ядрами, що обслуговують переривання [23,31].

— фіксувати розподіл ядер між компонентами SFC: якщо є ланцюг з кількох VNF, важливо уникати ситуації, коли одна VNF «вузька», а інші мають надлишок ядер (перекіс ланцюга);

— узгоджувати pinning з NUMA: RX/TX черги NIC, vSwitch (якщо використовується) та VNF мають бути в одному NUMA-вузлі;

— уникати розміщення dataplane потоків на sibling-потоках SMT/HT одного фізичного ядра, якщо пріоритетом є p99; краще мати менше, але «чистих» фізичних ядер;

— виділяти dataplane ядра окремо від системних (management) ядер; системні демони та контрольні процеси не повинні «заходити» на dataplane ядра;

Практичні правила CPU pinning у NFV середовищі:

На концептуальному рівні CPU pinning вирішує два завдання. Перше – забезпечує «резервування» ядер під мережеву обробку і запобігає конкуренції з control-plane/management задачами. Друге – забезпечує прогнозовану пропускну здатність на ядро (Gbps/ядро або PPS/ядро), що є основою для подальшого планування масштабування VNF і оцінки економічної ефективності [24,28].

CPU pinning (прив'язка процесів/потоків до конкретних фізичних ядер) є базовим засобом підвищення детермінізму, оскільки усуває міграції потоків між ядрами, які руйнують локальність кешу та збільшують кількість контекстних перемикань. Для dataplane-компонентів це особливо важливо: обробка пакетів є короткою, частою і чутливою до затримок на рівні мікросекунд, тому навіть невеликі планувальні «шуми» можуть перетворитися на значні хвости p99 [27,37].

2.5.7. Деталізація CPU pinning: механіка, правила проектування та типові помилки

2.5 Аналіз fast path та прискорених dataplane підходів (OVS-DPDK/DPDK)

Після стабілізації базового середовища були досліджені прискорені dataplane підходи: OVS-DPDK та user-space обробка пакетів. Мета – оцінити, у яких режимах fast path дає реальний виграш, а де ефект є обмеженим або економічно невиправданим. Fast path найбільш ефективний у PPS-критичних задачах (дрібні пакети, високі швидкості 25/100GbE), але потребує виділення ядер під polling і має складнішу експлуатацію [23,30,31].

Зручна інженерна модель для розуміння меж fast path базується на оцінці «циклів на пакет» (cycles-per-packet). Якщо f – тактова частота ядра (циклів/с), C – кількість виділених фізичних ядер dataplane, а K – середня кількість циклів на обробку одного пакета, тоді теоретична межа інтенсивності пакетів становить: $PPS_{max} \approx (f \cdot C) / K$.

Наприклад, при $f \approx 3.0$ ГГц, $C = 4$ ядра і $K = 600$ циклів/пакет маємо $PPS_{max} \approx 20$ Мпак/с. Якщо K зростає до 1500 циклів/пакет (типово для складнішого шляху

ядра/віртуалізації), PPS_max падає до ≈ 8 Мпак/с. Саме зменшення K і є «суть» fast path [23,27].

У цьому підрозділі fast path розглядається передусім як user-space прискорення dataplane на базі DPDK та OVS-DPDK. При цьому важливо підкреслити: fast path не є «магічною кнопкою», яка завжди покращує все. Він вводить інженерні компроміси: polling замість переривань (висока зайнятість CPU навіть при малій нагрузці), жорсткі вимоги до hugepages/NUMA, більшу складність дебагу та особливості інтеграції з SFC/політиками безпеки [23,31].

Для телекомунікаційних VNF це критично, оскільки реальні профілі трафіку містять комбінацію великих і дрібних пакетів, мікробурсти (microbursts), а також вимоги до мінімального джитеру. Якщо VNF обслуговує, наприклад, мобільний user plane, додаткові десятки або сотні мікросекунд у p99 можуть проявлятися як деградація QoE абонента, зменшення ефективної швидкості TCP та проблеми з реальним часом для мультимедійних сервісів [24,29].

Поняття fast path у контексті NFV означає організацію такого шляху обробки пакетів (dataplane), де мінімізуються накладні витрати загального призначення ОС: кількість копіювань даних, контекстні перемикання, планувальні затримки, робота з перериваннями та проходження «важкого» мережевого стеку ядра. На практиці fast path – це не одна технологія, а клас підходів, які зміщують баланс у бік максимізації PPS і стабілізації p99 за рахунок жорсткішого управління ресурсами та більш «ручної» експлуатації [23,27,30].

OVS kernel datapath є зручним у керуванні та сумісності, але під PPS-навантаженням може демонструвати більші хвости latency. OVS-DPDK переносить обробку у user-space, застосовуючи poll-mode драйвери і PMD threads, що часто покращує p99, але зменшує гнучкість у випадках, де потрібні традиційні механізми ядра.

Таблиця 2.5 – Приклад порівняння OVS kernel та OVS-DPDK (ізоляція увімкнена)

Режим vSwitch	64B PPS, Мпак/с	L_p99, мкс	Ядер під dataplane	Коментар
OVS kernel	5–7	600–1200	2–4	Краща сумісність, гірший p99 на бурстах
OVS-DPDK	8–12	250–600	4–8	Вищий PPS, стабільні хвости, polling cost

Отримані результати узгоджуються з типовою інженерною картиною: user-space fast path підвищує межу PPS і зменшує tail latency, але «оплачується» виділенням CPU під polling, а також більш вимогливою конфігурацією (hugepages, pinning, PMD mapping). Це означає, що fast path потрібно застосовувати таргетовано – там, де виграш у SLO дійсно конвертується в економічну користь.

Принципова різниця між kernel-path і DPDK-path полягає не лише у швидкості, а й у детермінізмі. Kernel стек зазвичай оптимізований під справедливість і універсальність, тоді як DPDK оптимізований під передбачуваність: виділені ядра, стабільні буфери, керований batching. Саме це зменшує tail latency та робить p99 керованим.

Типовий dataplane на DPDK будується з таких «цеглинок»:
- EAL (Environment Abstraction Layer) – ініціалізація hugepages, NUMA, прив’язка lcores;
- PMD (Poll Mode Driver) – драйвер, що читає/пише пакети батчами з RX/TX черг;
- mbuf/mempool – структура буфера пакета та пул пам’яті з мінімальними алокаціями;

- rings/queues – безлокові структури для передачі пакетів між потоками;
- burst processing – обробка пакетів не по одному, а групами (наприклад, по 32 або 64), що покращує використання кешу і зменшує overhead.
У результаті DPDK значно зменшує кількість контекстних перемикань та викликів ядра, а також покращує локальність даних у кеші [23,31].

DPDK (Data Plane Development Kit) – це набір бібліотек і драйверів poll-mode, які реалізують високошвидкісну обробку пакетів у user-space. Ключова ідея DPDK полягає у тому, щоб замінити переривання (interrupt-driven I/O) на polling та максимально спростити критичний шлях: пакет з NIC потрапляє у пам’ять, після чого обробляється на виділеному ядрі без участі складного kernel networking стеку [23].

Практично шлях пакета в OVS-DPDK можна описати так:
NIC RX queue - PMD thread (poll) - класифікація потоку (flow lookup) - застосування дій (actions: forward, NAT, tunnel, QoS) - передача у vhost-user/virtio або TX queue. Якщо flow lookup потрапляє у кеш exact match, пакет обробляється дуже швидко. Якщо ж відбуваються часті промахи кешу або потрібно виконувати складні дії (наприклад, conntrack/ACL), K (цикли/пакет) зростає, і PPS межа падає. Тому для чесної оцінки fast path важливо тестувати не лише «ідеальний» L2 forward, а реальні сценарії VNF (ACL, NAT, tunneling, stateful обробка) [30].

Open vSwitch у режимі DPDK (dpif-netdev + DPDK PMD) перетворюється на високопродуктивний user-space комутатор. На відміну від kernel datapath, де частина роботи виконується в ядрі, OVS-DPDK концентрує dataplane у user-space процесі з набором PMD-потоків, кожен з яких обслуговує певні RX-черги. Важливо, що ефективність OVS-DPDK визначається не лише DPDK, а й архітектурою flow lookup в OVS: механізмами exact match, мегапотоками (megaflow) та поведінкою кешу при мікропотоках [30,31].

2.6.3. OVS-DPDK як user-space vSwitch: шлях пакета, flow-кеші та місця втрат продуктивності

З інженерної точки зору найбільший внесок у виграш DPDK дають три ефекти:

1) відсутність IRQ-штормів (пакети читаються polling-циклом);

2) менше копіювань/менше переходів user↔kernel;

3) батчинг і краща кеш-локальність.

Однак саме ці ефекти формують і обмеження: polling означає, що CPU ядро зайняте майже постійно; батчинг може погіршувати мінімальну затримку (p50) заради кращого p99; а вимоги до hugepages/NUMA підвищують складність експлуатації.

Таблиця 2.8 – Порівняння критичного шляху пакета у kernel datapath та OVS-DPDK (якісна оцінка)

Етап обробки	Kernel datapath	OVS-DPDK/DPDK	Коментар
RX подія	IRQ/NAPI	Polling PMD	Polling дає стабільність, але «їсть» CPU
Копіювання/переходи	user↔kernel можливі	мінімізовано	Менше контекстів = менше хвостів
Flow lookup	в ядрі/OVS	user-space dpcls/megaflow	Кеш-hit критичний для PPS
Буфери	звичайні сторінки	hugepages + mbuf pool	Менше TLB/page-walk
Обробка батчами	обмежено	типово burst	Краща кеш-ефективність

Tail latency p99	вища	нижча	За умови правильної ізоляції/NUMA

Типова помилка при налаштуванні OVS-DPDK – виділення занадто малої кількості PMD потоків при великій кількості RX-черг. У такому випадку один PMD стає «гарячою точкою» (hot spot), а інші ядра простоюють. Інша помилка – виділення занадто багатьох PMD потоків, які починають конкурувати за кеш і пам’ять, що зменшує ефективність на пакет. Оптимум залежить від трафіку: для L2/L3 forward потрібні одні налаштування, для ACL/conntrack – інші, бо K (цикли/пакет) різний.

У OVS-DPDK ключовим ресурсом є PMD threads (poll-mode driver threads). Вони постійно опитують RX-черги і виконують комутацію. Звідси випливають два практичні наслідки. По-перше, кількість PMD потоків має відповідати кількості активних RX-черг і бажаному рівню паралелізму. По-друге, PMD потоки потрібно правильно «розкласти» по ядрах (pinning) і NUMA, інакше зростає між’ядерний contention і p99 погіршується [31].

З таблиці видно, що OVS-DPDK не лише прискорює «середній» пакет, а насамперед змінює статистичну природу системи: зменшується кількість випадкових затримок, які породжуються перериваннями, міграціями потоків та непередбачуваною роботою мережевого стеку. Саме тому на практиці fast path оцінюють у перцентильних метриках, а не лише в середніх значеннях.

Таблиця 2.9 – Приклад планування PMD потоків і RX-черг у 2×NUMA системі (ілюстративно)

NUMA	PMD ядра	RX-черги	VNF ядра	Примітка
NUMA0	core 2–5	q0–q7	core 6–11	Локальність для частини трафіку

NUMA1	core 18–21	q8–q15	core 22–27	Локальність для другої частини
Management	core 0–1	—	—	Control plane, моніторинг, логування

Для отримання реального ефекту від fast path недостатньо просто «увімкнути DPDK». Необхідний узгоджений набір налаштувань на рівні BIOS/CPU, ОС, NIC, OVS та самої VNF. Нижче наведено практичний список параметрів, які найбільш часто визначають успіх або провал прискорення.

Однак батчинг має інженерний компроміс: щоб зібрати батч, система може «почекати» кілька пакетів, що збільшує мінімальну затримку (p50). У телекомунікаційних сценаріях це прийнятно лише тоді, коли виграш у p99 і drops переважає. На практиці правильний батчинг часто покращує саме tail latency, бо зменшує шанси на backlog і head-of-line blocking у моменти бурстів.

Однією з причин високої продуктивності DPDK-підходів є обробка пакетів батчами (burst processing). Замість того, щоб виконувати цикл обробки на кожен пакет окремо (з окремими переходами, перевірками і доступами до пам’яті), PMD отримує масив посилань на пакети і працює з ними групою. Це дає три ефекти:
- краще використання інструкцій і кешу (дані й код «гарячі»);
- менше накладних витрат на цикл/гілкування;
- можливість prefetch (підтягування даних у кеш наперед).
В підсумку середня вартість K (циклів/пакет) зменшується, а PPS межа зростає [23].

Такий планування дозволяє зменшити remote memory доступ і стабілізувати p99. У телеком-практиці це важливо, оскільки навіть при однаковій середній пропускній здатності неправильне NUMA-розміщення створює рідкісні «піки» затримки, які псують SLA. Тому fast path практично неможливий без ізоляції ресурсів, описаної у підрозділі 2.5.

Таблиця 2.10 – Ключові параметри tuning для fast path і їх очікуваний ефект

Рівень	Параметр/дія	Очікуваний ефект	Ризики/зауваги
CPU/BIOS	Вимкнути глибокі C-states, увімкнути performance mode	Стабільний clock, менше jitter	Вища енерговитрата
OS	isolcpus/nohz_full/rcu_nocbs для dataplane ядер	Менше scheduler noise	Потрібна дисципліна ядер
Memory	Hugepages 2MB/1GB, локальність NUMA	Менше TLB misses, кращий p99	Неправильний NUMA зіпсує ефект
NIC	Налаштувати RSS, кількість RX/TX черг, ring sizes	Кращий розподіл PPS	Зайві черги → overhead
IRQ	IRQ affinity, вимкнути irqbalance для dataplane	Передбачуваність обробки	Потрібен ручний контроль
OVS-DPDK	PMD threads pinning, pmd-cpu-mask, rxq affinity	Усунення hot spots	Складність конфігурації
OVS flows	Утримання високого cache hit ratio, мінімізація upcalls	Менше K, вищий PPS	Складні політики → промахи кешу
VNF	Рознести control/dataplane, мінімізувати блокування	Стабільний p99	Додаткові ядра під control plane

Застосування fast path має бути SLO- та економічно обґрунтованим: оцінюється приріст T_SLO, а не «косметичні» пікові метрики [24,28].

Fast path потребує tuning на всіх рівнях (CPU/OS/NIC/OVS/VNF) і має експлуатаційні компроміси: polling, енергоспоживання, складність підтримки.

Ключем до продуктивності OVS-DPDK є правильне планування PMD потоків, RX-черг та NUMA-локальності; без цього можливий повний провал продуктивності.

Реальний ефект fast path проявляється передусім у стабілізації tail latency (p99) та зниженні втрат, а не тільки у піковому throughput.

Fast path у вигляді DPDK/OVS-DPDK зменшує цикли на пакет (K) і підвищує PPS межу, що особливо важливо для дрібних пакетів і бурстів [23,31].

Тому критичним є підхід «SLO-first»: fast path вважається успішним лише тоді, коли він збільшує T_SLO (корисну пропускну здатність у режимі виконання SLO) або зменшує витрати на одиницю T_SLO. Якщо зміни дають лише «паперове» зростання пікового throughput без стабілізації p99 і drops, це слід класифікувати як маркетинговий ефект, а не як інженерне підвищення ефективності.

Fast path часто подається як універсальне рішення, але в реальності є сценарії, де він не дає суттєвого виграшу або навіть погіршує експлуатацію. Найчастіші випадки, коли «DPDK не допоможе»:

- трафік переважно великими пакетами (1500B і більше), де throughput обмежений не PPS, а пропускною здатністю лінії;
- домінуюча складність – не комутація, а функціональна обробка (шифрування, DPI, складний stateful firewall);
- обмеження у пам’яті або кеші (LLC contention), де додаткові PMD ядра лише збільшують конкуренцію;
- вимоги до енергозбереження: polling утримує CPU активним і підвищує енерговитрати;
- високі вимоги до гнучких політик та інтеграції, де kernel-інструменти простіші і стабільніші в підтримці.

У таких випадках fast path може виглядати «круто» у бенчмарках, але не приносити економічної користі [24,28].

Для OVS-DPDK особливо корисними є команди спостереження за PMD/RXQ (показники завантаження, кількість пакетів на чергу, ознаки дисбалансу). В інженерній практиці це дозволяє швидко відповісти на питання: «обмеження в PMD чи в VNF?» Якщо PMD ядра перевантажені – потрібно змінювати rxq/pmd mapping або зменшувати складність політик; якщо PMD мають запас, а VNF перевантажена – проблема у логіці VNF або в необхідності масштабування.

Щоб переконатися, що fast path реально працює, недостатньо дивитися лише на throughput. Потрібно підтвердити, що dataplane дійсно проходить через user-space PMD, що немає неочікуваних upcalls/slow-path, і що PMD навантаження збалансоване. Практичні індикатори успішного режиму:
- рівномірне завантаження PMD потоків (без 100% на одному та 20% на інших);
- стабільні значення p99 між прогонами;
- низький drop rate на microbursts;
- відсутність системних IRQ «стрибків» на dataplane ядрах;
• високий flow cache hit ratio в OVS, мінімум upcalls у slow-path [31,37].

У практичних вимірюваннях після коректного tuning зазвичай спостерігається «зсув» обмеження: система переходить від CPU/IRQ bottleneck до більш «функціонального» bottleneck самої VNF (наприклад, conntrack таблиць, криптографії, або складних правил політик). Це є позитивним знаком: означає, що fast path прибрав накладні витрати середовища і проявив реальні межі логіки VNF.

2.6 Апаратні механізми прискорення: offload та SR-IOV

Важливий компонент підвищення ефективності VNF – використання апаратних можливостей сучасних NIC. Ці можливості охоплюють як «легкі» offload-функції (checksum, сегментація), так і глибші механізми (tunneling offload, SR-IOV). У телеком-сценаріях критично враховувати вплив цих функцій на tail latency: деякі offload-и можуть покращувати throughput, але погіршувати джитер через агрегацію пакетів [25,26,29].

У рамках даного підрозділу розглядаються два ключові напрями: 1) offload як загальний клас апаратних прискорювачів dataplane (від простих stateless offload

до flow-based та programmable), 2) SR-IOV як механізм апаратної віртуалізації мережевого адаптера, що забезпечує майже «прямий» доступ VNF до NIC без накладних витрат віртуального комутатора. Обидва напрями є важливими для підвищення ефективності VNF у високошвидкісних мережах.

Водночас hardware offload має системні компроміси. Чим більше логіки перенесено у NIC/SmartNIC, тим більше зростає залежність від конкретного вендора, специфіки драйверів та можливостей прошивки. Також виникають питання керованості (observability): коли частина dataplane «зникає» з CPU, діагностика, дзеркалювання трафіку та точний контроль політик можуть ускладнюватися. Саме тому апаратне прискорення слід розглядати через призму: (1) які функції переносимо в offload, (2) які метрики покращуємо, (3) які ризики додаємо до експлуатації, і чи виправданий компроміс у конкретному телеком-сценарії [24,28].

У телекомунікаційних мережах апаратне прискорення має особливу цінність, оскільки дозволяє одночасно підвищити масштабованість (пропускну здатність на вузол) і зменшити енергоспоживання на одиницю трафіку. Умовно кажучи, якщо без offload для 100 Gbps потрібні десятки CPU-ядер, то з апаратним прискоренням частину цих ядер можна «повернути» під інші функції: керування сесіями, DPI, оптимізацію, аналітику, або просто зменшити кількість серверів у пулі. Це напряму впливає на економіку NFV: CAPEX та OPEX в інфраструктурі значною мірою визначаються кількістю серверів і споживаною потужністю [21,24].

Апаратні механізми прискорення в NFV – це наступний «поверх» оптимізації після ізоляції ресурсів та fast path у user-space. Якщо CPU pinning/IRQ affinity/hugepages зменшують варіативність виконання, а DPDK/OVS-DPDK зменшують цикли на пакет у програмному dataplane, то hardware offload переносить частину функцій обробки пакетів безпосередньо у мережевий адаптер (NIC) або в спеціалізовані пристрої (SmartNIC/DPU). Ідеологічно це означає: «виконувати на CPU лише те, що справді потребує CPU», а все повторюване та стандартизоване – віддавати апаратурі [24,28,44].

Під час практики виконувались A/B тести: увімкнення/вимкнення GRO/LRO, TSO/GSO, tunnel offload. Загальний висновок: checksum offload майже завжди корисний, тоді як агрегаційні механізми потребують перевірки на конкретному трафіковому профілі. Для сервісів реального часу нерідко пріоритетом є зниження джитеру, тому агрегація може бути обмежена.

SR-IOV дозволяє надати VNF прямий доступ до віртуальної функції NIC (VF), мінімізуючи роль vSwitch у dataplane. Це часто дає найкращі показники tail latency та PPS, особливо для критичних VNF. Водночас SR-IOV може ускладнювати політики SFC, безпеку та міграцію. Тому оцінка SR-IOV повинна включати не лише метрики продуктивності, а й експлуатаційні обмеження [26,29].

Таблиця 2.6 – Узагальнене порівняння режимів підключення VNF до мережі

Режим	Latency p99	PPS	Керованість	Коментар
Virtio + OVS kernel	Середня/висока	Середній	Висока	Найпростіше, але хвости на PPS
Virtio + OVS-DPDK	Низька/середня	Високий	Середня	Потрібні PMD/pinning
SR-IOV VF	Найнижча	Найвищий	Низька/середня	Макс. продуктивність, більше обмежень

Окремо виділяють programmable offload, де NIC/SmartNIC/DPU виконує програмовані правила або невеликі програми. Цей підхід наближає мережеві адаптери до ролі «мережевого співпроцесора». Його перевага – висока швидкодія при гнучкості; недолік – складніший життєвий цикл, необхідність спеціалізованого SDK та ризик фрагментації екосистеми. У промислових NFV рішеннях programmable offload часто виступає як стратегічний інструмент масштабування, але потребує більш зрілої інженерної культури експлуатації.

Stateful offload є ще складнішим: він передбачає підтримку стану (state) на рівні сесій або таблиць, наприклад conntrack-подібних механізмів. Це може бути корисним для stateful firewall/NAT, але водночас підвищує ризики: апаратний стан складніше дебажити, виникають питання консистентності, оновлення політик та сумісності між поколіннями NIC. Тому в телеком-практиці stateful offload застосовують обережно і лише тоді, коли профіль навантаження стабільний, а вимоги до підтримки – контрольовані [24,28].

Наступний рівень – flow-based offload, де адаптер (або SmartNIC) може виконувати певні дії на рівні потоків: класифікацію, напрямлення, прості ACL, тунелювання, інколи NAT або елементарні політики QoS. При цьому dataplane починає працювати за принципом: перший пакет потоку може пройти «повільним» шляхом для встановлення правила, а наступні пакети – апаратно прискореним шляхом за вже встановленим flow rule. Така модель подібна до того, як OVS використовує кеш flows, але виконання правил переноситься у NIC. Це здатне суттєво зменшити навантаження на CPU при великій кількості однотипних потоків.

Для системного аналізу доцільно класифікувати offload за рівнем «інтелекту» та впливом на архітектуру VNF. Найпростішим є stateless offload – прискорення, яке не потребує збереження стану між пакетами. Сюди належать: checksum offload, TCP segmentation offload (TSO), generic segmentation/receive offload (GSO/GRO), LRO (large receive offload), апаратний RSS (Receive Side Scaling), апаратне timestamping та інші функції, що зменшують накладні витрати на CPU без зміни логіки мережевої функції. Stateless offload зазвичай дає «безкоштовний» приріст, але його ефект обмежений – він не перетворює dataplane радикально, а лише прибирає частину дрібних витрат [24,37].

Таблиця 2.11 – Класи hardware offload та їх вплив на ефективність VNF

Клас offload	Приклади	Що прискорює	Типовий ефект	Ризики/обмеження
Stateless	Checksum, TSO/GSO/GRO, RSS	Накладні витрати CPU на I/O	Невеликий, але стабільний виграш	Ефект обмежений профілем трафіку
Flow-based	Flow steering, ACL, tunnel offload	Forwarding для потоків	Суттєве зниження CPU при cache-hit	Залежність від підтримки правил у NIC
Stateful	Conntrack/NAT offload	Stateful обробка	Різке зростання масштабованості для типових сесій	Складність дебагу, консистентність стану
Programmable	SmartNIC/DPU pipelines	Частину VNF-логіки	Висока швидкодія + гнучкість	SDK/вендор lock-in, складний lifecycle

З інженерної точки зору SR-IOV займає проміжне місце між «чистим» PCI passthrough та класичною віртуалізацією з virtio/vSwitch. PCI passthrough (VFIO) дає максимальну продуктивність, але практично «прив’язує» VM до конкретного хоста і ускладнює live migration. Virtio/vhost-user більш гнучкі, краще інтегруються з оркестрацією та політиками безпеки, але мають більший overhead. OVS-DPDK забезпечує високий рівень продуктивності з гнучкістю комутації, але потребує складного tuning. SR-IOV у багатьох випадках дає оптимальний баланс: майже direct I/O при збереженні можливості керування PF з боку оператора [24,31].

Практичний ефект SR-IOV найбільш помітний у двох сценаріях: 1) висока PPS-навантаженість з дрібними пакетами, де vSwitch/virtio створюють

значний overhead;
2) потреба в низькій і стабільній затримці, де переривання і планування vSwitch спричиняють tail latency.
У цих умовах SR-IOV може зменшити CPU витрати на I/O і покращити p99 без обов’язкового переходу на повний user-space vSwitch.

З позиції архітектури це означає: якщо у класичному NFV пакети проходять NIC - vSwitch - VM/контейнер, то при SR-IOV VM отримує VF і шлях стає ближчим до NIC - VM. Важливо, що безпека та ізоляція забезпечуються IOMMU (Input-Output Memory Management Unit), яка обмежує DMA-доступ VF до пам’яті лише дозволеними ділянками. Це критично, бо без IOMMU VF могла б потенційно читати/писати пам’ять хоста. Отже, SR-IOV «прискорює» за рахунок зменшення програмних шарів, але вимагає правильно налаштованої платформи та політик безпеки.

SR-IOV (Single Root I/O Virtualization) – це апаратний механізм, який дозволяє одному фізичному PCIe пристрою (NIC) експонувати декілька незалежних «віртуальних пристроїв» – Virtual Functions (VF). Керівний компонент називається Physical Function (PF) і відповідає за ініціалізацію, управління ресурсами та конфігурацію. Кожна VF має власні черги RX/TX, MSI-X переривання та інтерфейс, який може бути переданий у VM або контейнер як окремий NIC. Таким чином, dataplane VNF отримує доступ до NIC без проміжного vSwitch-шару або з мінімальною участю гіпервізора, що різко зменшує накладні витрати [24,27].

Таблиця 2.12 – Порівняння механізмів доступу до NIC у NFV (якісна оцінка)

Підхід	Продуктивність	p99 затримка	Гнучкість/SFC	Експлуатація	Типові сценарії
Virtio + kernel vSwitch	Середня	Середня/вища	Висока	Проста	Базові VNF, невисокі PPS
Virtio + vhost-user	Вища	Краща	Висока	Середня	Контейнери/VM з підвищеною продуктивністю
OVS-DPDK	Висока	Низька (за tuning)	Висока	Складніша	SFC, високі PPS, гнучкі політики
SR-IOV	Дуже висока	Низька	Середня	Складніша	Високий PPS, low-latency VNF
PCI passthrough	Максимальна	Найнижча	Низька	Складна	Dedicated VNF, спеціальні задачі

Економіко-математично ефект апаратного прискорення зручно інтерпретувати через «вивільнення» CPU ресурсів. Нехай без offload для заданого трафіку потрібно C_sw ядер (software dataplane), а з offload – C_hw ядер. Тоді приріст ефективності можна оцінити як: $E = C_{sw} / C_{hw}$.

Якщо, наприклад, C_sw = 12 ядер, а C_hw = 6 ядер, то E = 2 (тобто удвічі більше корисного трафіку на той самий серверний бюджет CPU). У масштабі телеком-

інфраструктури навіть E=1.3–1.5 може бути суттєвим, оскільки масштабується на сотні вузлів і роки експлуатації [21,24].

Як і у випадку з OVS-DPDK, апаратне прискорення не працює «саме по собі»: потрібна коректна прив'язка черг, ядер і NUMA. Кожна VF має обмежену кількість черг, і якщо трафік концентрується на одній-двох чергах, виникає внутрішній bottleneck вже на рівні VF. Тому при проектуванні SR-IOV важливо враховувати: кількість VF, кількість RX/TX черг на VF, механізм RSS всередині VF, та відповідність цих черг dataplane ядрам VNF. Інакше SR-IOV може не дати очікуваного виграшу, або навіть створити нерівномірність навантаження.

Таблиця 2.13 – Ілюстративна оцінка ефекту offload/SR-IOV на потребу в CPU ядрах

Сценарій	Трафік	Підхід	Ядра dataplane	Очікуваний p99	Коментар
L2/L3 forward	25G дрібні пакети	Kernel vSwitch	8–12	Вищий	Overhead IRQ/stack
L2/L3 forward	25G дрібні пакети	OVS-DPDK	4–8	Нижчий	Потрібен tuning
L2/L3 forward	25G дрібні пакети	SR-IOV	3–6	Низький	Менше шарів, direct I/O
ACL/Firewall	25G mixed	OVS-DPDK	6–12	Стабільний	Залежить від політик
NAT/Stateful	25G mixed	Stateful offload*	4–8	Низький	Потрібна підтримка NIC

SmartNIC/DPU є стратегічним напрямом, який може перетворити мережевий адаптер на співпроцесор, але збільшує складність експлуатації.

Компромісом є зниження гнучкості SFC та ускладнення observability; рішенням можуть бути representor/switchdev або комбіновані архітектури.

Найкращий ефект досягається при коректному плануванні NUMA та черг; інакше виникають bottleneck на рівні VF або remote memory доступ.

SR-IOV забезпечує майже direct I/O для VNF через PF/VF, зменшуючи накладні витрати vSwitch, і покращує p99 у low-latency сценаріях.

Hardware offload переносить частину dataplane функцій у NIC/SmartNIC та знижує витрати CPU на одиницю трафіку, підвищуючи ефективність NFV [24,44].

Проміжні висновки до підрозділу 2.7

Однак SmartNIC/DPU змінює архітектуру експлуатації: виникає окремий життєвий цикл прошивки, оновлення, моніторингу, а також необхідність узгодження з оркестратором. Тому цей підхід особливо виправданий у великих телеком-операторів або у випадках, коли приріст ефективності дає відчутний економічний ефект у масштабі парку обладнання.

SmartNIC або DPU (Data Processing Unit) – це еволюція концепції offload, де адаптер має власні обчислювальні ресурси (ядра ARM, прискорювачі, інколи FPGA/ASIC блоки) і може виконувати частину логіки мережевих функцій автономно від CPU хоста. Для NFV це відкриває два сценарії підвищення ефективності:

- «розвантаження хоста» – перенос vSwitch, тунелювання, шифрування, фільтрації на DPU;
- «композиція VNF» – коли частина функцій реалізується як pipeline у DPU, а складні компоненти залишаються на CPU.

У результаті зменшується накладний overhead і підвищується масштабованість на вузол, особливо для 100GbE і вище.

Третя категорія – сумісність із live migration та динамічним розміщенням. Багато реалізацій SR-IOV історично ускладнювали міграцію VM між хостами, оскільки VF прив’язані до конкретного PCIe пристрою. У практиці NFV це може

бути прийнятно для edge вузлів або для dedicated VNF, але для хмарних сценаріїв потрібна окрема політика: або відмовитися від live migration, або використовувати більш гнучкі vhost-user/DPDK механізми.

Друга категорія ризиків – observability. У програмному dataplane можна легко застосувати tcpdump/pcap, вбудовану телеметрію, лічильники потоків. При offload частина обробки відбувається «всередині NIC», і доступ до детальної статистики залежить від інструментів виробника. Це може ускладнити реакцію на інциденти та аналіз причин деградації.

Найбільш суттєвим компромісом SR-IOV є зменшення гнучкості мережевих політик на рівні хоста. Якщо VF передано безпосередньо у VM, то класичний vSwitch втрачає можливість повного контролю над трафіком цієї VM: дзеркалювання, перехоплення, traffic shaping, складні SFC-маршрути можуть потребувати додаткових механізмів. Це особливо важливо в телеком-сценаріях, де сервісний ланцюг може включати firewall → NAT → DPI → QoS, і оператор хоче динамічно змінювати маршрут трафіку. У таких випадках SR-IOV іноді комбінують з апаратними representor-портами або switchdev-підходами, де хост все ще може програмувати потоки у NIC [24,28].

2.7 Техніко-економічна оцінка ефективності: модель вартості та аналіз чутливості

Оскільки метою підвищення ефективності VNF у більшості випадків є не «рекорди продуктивності», а зменшення сукупної вартості надання послуг при дотриманні SLA/SLO, доцільно сформулювати економіко-математичну модель. У роботі використано модель вартості 1 Gbps·міс у режимі виконання SLO. Під «режимом SLO» розуміється, що L_p99 та drop rate не перевищують заданих граничних значень.

Нехай C_total – місячні витрати на NFVI-вузол (амортизація, енергія, підтримка, ліцензії), а T_SLO – пропускна здатність, яка гарантовано досягається при виконанні SLO. Тоді вартість одиниці продуктивності визначається як:

Cost_{Gbps·міс} = C_total / T_SLO.

Якщо система демонструє 20 Гбіт/с у середньому, але лише 12 Гбіт/с у режимі, де L_p99 ≤ L_SLO та drop rate ≤ D_SLO, то для економічної оцінки потрібно брати саме 12 Гбіт/с. Такий підхід узгоджується з принципом «tail-first»: продуктивність без якості не є корисною для оператора.

Наведемо ілюстративний приклад. Нехай C_total = 2500 у.о./міс для одного NFVI-вузла (включно з енергією та підтримкою). У базовому режимі (kernel datapath без ізоляції) T_SLO = 8 Гбіт/с, тоді Cost_{Gbps·міс} = 312.5 у.о. Після ізоляції ресурсів T_SLO зростає до 10 Гбіт/с: Cost = 250 у.о. Після переходу на OVS-DPDK T_SLO = 14 Гбіт/с: Cost = 178.6 у.о., але зростає CPU-вартість (частина ядер постійно зайнята polling).

Таблиця 2.7 – Приклад техніко-економічного порівняння режимів (SLO-орієнтовано)

Режим	T_SLO, Гбіт/с	C_total, у.о./міс	Cost_{Gbps·міс}, у.о.
Базовий (kernel)	8	2500	312.5
Ізоляція ресурсів	10	2500	250.0
OVS-DPDK	14	2500	178.6
SR-IOV (критичні VNF)	16	2600	162.5

З прикладу видно, що оптимізація має реальний економічний сенс лише тоді, коли підвищення T_SLO перевищує додаткові витрати/обмеження. Наприклад, SR-IOV може бути технічно найкращим, але вимагати додаткових витрат на адміністрування/політики або обмежувати міграцію VNF. У такому разі рішення може бути таргетованим: SR-IOV лише для latency-critical функцій, а інші – через прискорений vSwitch.

У телекомунікаційних мережах ефективність VNF не можна зводити лише до максимальної пропускної здатності або мінімальної затримки. Оператору важливо

отримати гарантований рівень сервісу (SLO/SLA) при мінімально можливих витратах. Тому техніко-економічна оцінка має формалізувати компроміс між продуктивністю, детермінізмом (p95/p99), надійністю і вартістю. Узагальнено задачу можна описати як оптимізацію: мінімізувати сукупні витрати володіння (TCO) за умови виконання обмежень по пропускній здатності, затримці та втратам пакетів:

min TCO, за умов: Throughput ≥ T_req, p99 ≤ L_req, Drop ≤ D_req. Саме такий підхід дозволяє відділити реальні інженерні оптимізації (які покращують TCO при виконанні SLO) від «маркетингових» покращень, які показують красиві пікові бенчмарки, але не змінюють кількість необхідних серверів або рівень ризику інцидентів [24,28,50].

Для практичного порівняння альтернатив (kernel dataplane, OVS-DPDK, SR-IOV/offload) застосовано модель TCO, яка охоплює капітальні витрати (CAPEX) і операційні витрати (OPEX) у горизонті N років. З урахуванням ставки дисконту r сумарна приведена вартість може бути записана як: TCO_NPV = CAPEX_0 + Σ_{t=1..N} OPEX_t / (1 + r)^t. У багатьох навчальних та прикладних оцінках для короткого горизонту (3–5 років) допускають r≈0 і переходять до спрощеної форми TCO = CAPEX + N·OPEX. У даній роботі використано саме таке спрощення, щоб зосередитися на порівнянні технологічних альтернатив, а не на фінансових особливостях дисконтування.

Структура CAPEX включає:

- C_srv – вартість серверів (CPU, RAM, диски);
- C_nic – мережеві адаптери (100GbE/25GbE) або SmartNIC;
- C_acc – прискорювачі (якщо використовуються окремо);
- C_sw_init – первинні витрати на інтеграцію/впровадження.

У свою чергу, OPEX включає:

- C_energy – електроенергія та охолодження (через PUE);
- C_maint – технічне обслуговування/підтримка (як частка CAPEX);

- C_license – ліцензії та підтримка ПЗ (за потреби);
- C_ops – витрати на експлуатацію (персонал, інциденти, оновлення).

Перевага такої декомпозиції в тому, що вона дозволяє робити аналіз чутливості: визначати, які фактори найсильніше впливають на результат і де оптимізація дає найбільший економічний ефект [21,24].

Таблиця 2.14 – Набір параметрів для моделі TCO (приклад параметризації)

Позначення	Зміст	Одиниця	Коментар
N_srv	Кількість серверів у пулі VNF	шт.	Визначається продуктивністю та SLO
C_srv	Вартість 1 сервера	USD	CAPEX компонент
C_nic	Вартість NIC/SmartNIC на сервер	USD	Залежить від SR-IOV/offload
P_srv	Середня потужність сервера	кВт	Залежить від режиму dataplane
c_e	Ціна електроенергії	USD/кВт·год	OPEX, залежить від країни/ринку
PUE	Коефіцієнт енергетичної інфраструктури	—	PUE>1 враховує охолодження
α_maint	Частка витрат на підтримку	1/рік	Напр., 0.1 = 10% від CAPEX/рік
N	Горизонт оцінки	роки	Типово 3–5 років для серверів

Найважливіший місток між технічними результатами і економікою – це перерахунок приросту продуктивності у зміну N_srv. Поки оптимізація (CPU

pinning, DPDK, SR-IOV/offload) не зменшує кількість необхідних серверів або не підвищує допустиме навантаження в SLO-режимі, її економічний ефект зазвичай обмежений. Навпаки, навіть помірне збільшення продуктивності може дати дискретний економічний виграш, якщо воно «перестрибує поріг» і дозволяє зменшити N_srv на 1 одиницю.

Тому у моделі використовується правило:

N_srv = ceil(T_req / T_srv,SLO),

де T_req – вимога по корисному трафіку, а T_srv,SLO – максимальний трафік, який один сервер здатен обслуговувати при виконанні SLO (наприклад, p99 ≤ L_req) [24,28].

2.8.5. Розрахунковий приклад TCO для 100 Gbps: kernel vs OVS-DPDK vs SR-IOV/offload

Для демонстрації методики розглянемо ілюстративний сценарій: VNF-пул має забезпечити 100 Gbps корисного трафіку у режимі виконання SLO (з контрольованим p99 та допустимими втратами). Припустимо, що в базовому kernel dataplane це вимагає 3 сервери; при OVS-DPDK – 2 сервери; при SR-IOV/offload – 1 сервер.

Нехай також:

C_srv = 6500 USD; C_nic = 1500 USD (звичайний 100GbE); C_smartnic = 5000 USD; P_srv ≈ 0.45 кВт (звичайний сервер), P_srv,offload ≈ 0.50 кВт; c_e = 0.12 USD/кВт·год; PUE = 1.4; α_maint = 10%/рік; N = 3 роки. Енерговитрати на сервер за рік оцінюємо як: C_energy = P_srv · 8760 · c_e · PUE.

Таблиця 2.15 – Результати TCO-оцінки для 100 Gbps (ілюстративний приклад)

Сценарій	N_srv	CAPEX, USD	OPEX(енергія)/рік, USD	OPEX(підтримка)/рік, USD	TCO_3y, USD
Kernel dataplane	3	24,000	1,987	2,400	37,160
OVS-DPDK fast path	2	16,000	1,325	1,600	24,774
SR-IOV + offload	1	11,500	736	1,150	17,158

За даними таблиці видно, що перехід від kernel dataplane до OVS-DPDK зменшує сумарний TCO_3y приблизно на третину (з 37,160 до 24,774 USD), оскільки зменшується N_srv. SR-IOV/offload у даному прикладі ще сильніше зменшує TCO_3y (до 17,158 USD) завдяки дискретному скороченню пулу до одного сервера. Цей результат демонструє ключовий принцип: економічний ефект NFV-оптимізацій часто носить пороговий характер – він різко зростає, коли оптимізація дозволяє зменшити кількість серверів на 1 або більше [24,28].

Для оцінки «стійкості» рішення до варіацій вартості апаратури корисно знаходити break-even значення. У наведеному прикладі SR-IOV/offload залишається не гіршим за OVS-DPDK навіть якщо вартість SmartNIC зросте приблизно до 10,858 USD (за інших рівних умов). Це показує, що головний драйвер економічної вигоди – зменшення N_srv, а не точне значення ціни мережевого адаптера.

Аналіз чутливості дозволяє визначити, які параметри є «важелями» вартості. У роботі застосовано one-factor-at-a-time підхід: кожен параметр змінюється на ±20% при фіксації інших. Для інтерпретації вводиться відносний вплив (еластичність):

$$S_x = (\Delta TCO/TCO) / (\Delta x/x).$$

Якщо S_x велике, параметр x є критичним для економіки рішення; якщо мале – його зміни не впливають суттєво.

Таблиця 2.16 – Аналіз чутливості TCO_3y (сценарій OVS-DPDK) при зміні параметрів ±20%

Параметр	Базове значення	TCO при −20%	TCO базове	TCO при +20%	Вплив (+20%), %
server_cost	6500	21,394	24,774	28,154	13.64
electricity	0.12	23,979	24,774	25,568	3.21
pue	1.4	23,979	24,774	25,568	3.21
maint_rate	0.1	23,814	24,774	25,734	3.88

Результати аналізу чутливості показують, що для OVS-DPDK найбільшим драйвером TCO є вартість серверів (CAPEX) – що логічно, оскільки вона множиться на кількість серверів. Електроенергія та PUE мають менший вплив у короткому горизонті, але їхня роль зростає при N=5–7 років, високих цінах на енергію або жорстких вимогах до енергоефективності.
З практичної точки зору це означає: оптимізації, які зменшують N_srv, майже завжди економічно сильніші за оптимізації, які лише «трошки» зменшують CPU завантаження, але не змінюють розмір інфраструктури.

Для практичного вибору між підходами kernel, OVS-DPDK та SR-IOV/offload доцільно використовувати алгоритм:

1) Визначити SLO: T_req, p99, допустимі drops, профіль пакетів.
2) Виміряти T_srv,SLO для кожної альтернативи (не піковий throughput, а в SLO-режимі).
3) Обчислити N_srv та дискретні пороги, де оптимізація зменшує кількість серверів.
4) Розрахувати TCO та провести аналіз чутливості.

5) Врахувати ризики експлуатації (observability, vendor lock-in, складність tuning) як «штраф» до економічного ефекту.
6) Прийняти рішення: впроваджувати лише тоді, коли приріст ефективності зберігається у найгірших допустимих сценаріях параметрів.

Такий підхід переводить дискусію з «яка технологія модніша» у площину відтворюваної інженерної економіки.

Можна зробити втсновок:

1) Техніко-економічна ефективність VNF повинна оцінюватися через TCO при виконанні SLO, а не через пікові бенчмарки.
2) Найсильніший економічний ефект дають оптимізації, що зменшують кількість серверів N_srv (пороговий характер вигоди).
3) OVS-DPDK та SR-IOV/offload можуть давати значну економію у високих PPS/low-latency сценаріях, але потребують більш зрілої експлуатації.
4) Аналіз чутливості дозволяє виділити критичні фактори (CAPEX сервера, параметри енергоінфраструктури, ставка підтримки) і зменшити ризик помилкового вибору технології.
5) Рішення про прискорення слід приймати за методикою «SLO → N_srv → TCO → sensitivity → ризики» [24,28,50].

2.8 Узагальнення результатів аналізу та формалізація проблеми для подальших розрахунків

Фактологічний аналіз практики дозволив сформулювати ключову проблему предмету дослідження у вимірюваних термінах: у базових конфігураціях NFV/VNF спостерігається значна варіативність обробки трафіку (tail latency і drops), яка посилюється при дрібних пакетах, бурстах і порушенні локальності NUMA. Ця варіативність не усувається простим збільшенням середнього CPU ресурсу, оскільки має системну природу (черги, IRQ, кеш, пам’ять, vSwitch).

Отже, для підвищення ефективності потрібен комплексний підхід, що включає: (1) ізоляцію ресурсів як основу детермінізму; (2) вибір режиму dataplane

(kernel/OVS-DPDK/SR-IOV) залежно від класу VNF і SLO; (3) оптимізацію NIC-параметрів та offload-профілів; (4) техніко-економічне обґрунтування, яке оцінює не «піковий throughput», а продуктивність у SLO-режимі.

У наступному (конструктивному) розділі на основі цієї фактологічної бази будуть виконані розрахунки та сформульовані обґрунтовані пропозиції щодо підвищення ефективності застосування VNF у телекомунікаційній мережі.

3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ ТА ПРОПОЗИЦІЇ ЩОДО ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ VNF

Метою конструктивного розділу є перетворення теоретичних висновків та аналітичних спостережень попередніх розділів у формалізовані результати: розрахунки, порівняння альтернативних конфігурацій та обґрунтовані інженерні пропозиції. Основна ідея полягає в тому, що ефективність VNF в телекомунікаційних мережах повинна оцінюватися не декларативно, а через відтворювані метрики (throughput, p99 latency, drop rate, cycles/packet) та через техніко-економічні показники (TCO). У розділі використано підхід «SLO-first»: рішення вважається успішним лише тоді, коли забезпечує задану пропускну здатність та обмеження за затримкою при стабільній роботі системи. [61] [63] [89]

3.1 Методика дослідження та організація вимірювань

Для оцінки ефективності застосування VNF використано експериментально-аналітичну методику, яка поєднує: (1) постановку контрольованих сценаріїв трафіку; (2) вимірювання продуктивності з фіксацією апаратних параметрів; (3) статистичну обробку результатів; (4) інтерпретацію отриманих метрик у вигляді інженерних рекомендацій та економічного ефекту. Вимірювання виконуються серіями, щоб зменшити вплив випадкових факторів (фонове навантаження ОС, турбо-режими CPU, флуктуації кешів та черг). [61] [63]

Сценарії трафіку сформовано так, щоб наблизити їх до телекомунікаційних профілів: дрібні пакети (64–128 B) для PPS-стресування, змішаний профіль (IMIX) для моделювання реалістичних потоків, та великі пакети (512–1500 B) для оцінки меж пропускної здатності в Gbps. Окремо враховано варіанти однонаправленого та двонаправленого навантаження, оскільки для багатьох VNF (firewall/NAT) симетричність трафіку істотно впливає на cache-hit і на витрати stateful обробки. [61] [63] [71].

Ключовими метриками обрано:

- Пропускна здатність (Throughput, Gbps) – максимальний стійкий потік без деградації SLO;
- Пакетна продуктивність (PPS) – кількість пакетів/с, критична для дрібнопакетного трафіку;
- Затримка (Latency) – оцінюється через p50/p95/p99, щоб врахувати tail latency;
- Втрата пакетів (Drop rate) – як відсоток, так і абсолютні значення в піку;
- Витрати CPU (utilization, cycles/packet) – відображають ефективність dataplane;
- Детермінізм (jitter) – варіативність затримки при однаковому навантаженні. [61] [63] [89]

Для статистичної обробки результатів використовуються середнє значення, медіана, перцентилі, а також оцінка розкиду (стандартне відхилення та міжквартильний розмах). Для рядів вимірювань застосовується принцип: якщо два режими мають подібний середній Throughput, але різний p99, пріоритет надається режиму з меншим p99, оскільки телеком-сервіси чутливі до хвостів затримки. Цей підхід узгоджується з сучасною практикою експлуатації, де стабільність важливіша за пікові значення [56,60]. [61] [63]

Таблиця 3.1 – Приклад конфігурації тестового стенда для порівняння режимів dataplane [61] [63]

Компонент	Параметр	Базове значення	Примітка
CPU	К-ть ядер/потоків	24C/48T	2-socket або high-core single-socket
RAM	Обсяг	128–256 GB	У т.ч. резерв для hugepages
NIC	Швидкість	25/100 GbE	Підтримка SR-IOV, RSS, MSI-X
Диск	Тип	NVMe SSD	Впливає на логи/телеметрію

ОС	Kernel	LTS + tuned	isolcpus, irqbalance policy
VNF	Тип	FW/NAT/Router	Модельна VNF (user-plane)
Генератор трафіку	Режим	RFC-like	64B/IMIX/1500B

3.2 Базова конфігурація та формалізація порівнюваних режимів

Для порівняння сформовано три базові режими виконання VNF, які відповідають найбільш поширеним підходам у NFV/NFVI:
1) Kernel dataplane: класична схема NIC - kernel networking - vSwitch/iptables - VNF.
2) Fast path у user-space: OVS-DPDK або DPDK-based vSwitch/VNF.
3) Апаратно прискорений режим: SR-IOV + offload, де VNF отримує VF або використовує representor/switchdev підхід. [65] [66] [67]

Для коректності порівняння приймається, що функціональна логіка VNF (політики, ACL, NAT-правила) однакова. Тобто змінюється лише шлях пакета та витрати обробки, а не «обсяг роботи». Це дозволяє інтерпретувати різницю результатів як ефект прискорення (fast path / offload) або ізоляції ресурсів (pinning/hugepages). [68] [79] [82]

Формально, загальний час обробки пакета у VNF можна подати як суму компонентів:
$T_{pkt} = T_{io} + T_{classify} + T_{policy} + T_{state} + T_{tx}$,
де T_{io} – накладні витрати отримання пакета з NIC та доставки у контекст виконання; $T_{classify}$ – класифікація та пошук правила; T_{policy} – виконання політики; T_{state} – робота зі станом (conntrack/session); T_{tx} – відправка пакета. Механізми прискорення в основному зменшують T_{io} і частково $T_{classify}$ (кеш, flow offload), а не саму логіку політик. Саме це робить прискорення універсальним для різних класів VNF. [68] [85]

3.3 Результати дослідження ефекту ізоляції ресурсів: CPU pinning, IRQ affinity, hugepages

У першій серії експериментів оцінюється вплив ізоляції ресурсів на детермінізм та пропускну здатність. Використано три рівні конфігурації:
1) Default: без спеціального pinning, стандартне розміщення процесів і переривань;
2) CPU pinning + IRQ affinity: виділення ядер під dataplane та закріплення NIC IRQ/queues;
3) Додатково hugepages: використання великих сторінок пам'яті для зменшення TLB misses і стабілізації latency. [61] [63] [79]

Основний очікуваний ефект від ізоляції ресурсів полягає в тому, що dataplane перестає конкурувати за CPU з фоновими задачами, а переривання та обробка пакетів локалізуються в межах NUMA-домена. Це зменшує кількість дорогих переходів контексту та remote memory доступів. Найбільш помітний результат проявляється у зниженні p99 затримки та зменшенні jitter, навіть якщо середній throughput зростає незначно. Така ситуація типова для телеком-сценаріїв, де tail latency визначає якість сервісу. [61] [63] [79]

Таблиця 3.2 – Приклад результатів впливу ізоляції ресурсів на p99 та throughput (64B, високий PPS) [61] [63]

Режим	Throughput, Mpps	p50, мкс	p99, мкс	Jitter (IQR), мкс	Коментар
A) Default	6.5	28	210	35	Високий tail через конкуренцію CPU/IRQ
B) Pinning+IRQ	7.4	24	120	18	Суттєве зменшення p99

C) +Hugepages	7.6	23	95	14	Додаткове згладжування jitter

Отримані результати демонструють фундаментальний принцип: навіть «безкоштовні» з точки зору CAPEX оптимізації (налаштування pinning/IRQ/hugepages) здатні суттєво підвищити стабільність системи. У телекомунікаційній експлуатації це часто більш цінно, ніж +5–10% пікового throughput, оскільки зменшує ризик деградації сервісу в пікові години та при аномальних патернах трафіку. [61] [63] [79]

3.4 Результати порівняння kernel dataplane та fast path (OVS-DPDK/DPDK): throughput, p99, cycles/packet

Друга серія експериментів порівнює класичний kernel dataplane з fast path у user-space. Для fast path використовуються принципи DPDK: poll-mode драйвери, пакетна обробка burst-ами, mbuf/mempool, та уникнення переривань у високонавантажених режимах. Очікуваний ефект – зменшення накладних витрат T_io, зростання PPS та зниження p99 при високих навантаженнях, якщо tuning виконано коректно. [61] [63] [65]

Для нормалізації результатів використано показник cycles/packet (cpp), який оцінюється як:

$$cpp = (f_{cpu} \cdot N_{cores}) / PPS,$$

де f_{cpu} – тактова частота CPU, N_{cores} – кількість задіяних ядер dataplane, PPS – досягнута продуктивність. Менше cpp означає вищу ефективність dataplane. У практиці NFV саме cpp часто визначає, чи можна зменшити N_srv, тобто отримати економічний ефект. [61] [63] [69]

Таблиця 3.3 – Приклад порівняння kernel та OVS-DPDK при 64B (PPS-критичний режим) [61] [63] [65]

Підхід	PPS, Mpps	CPU cores (dataplane)	cpp (оцінка)	p99, мкс	Drop при піку, %
Kernel dataplane	6.8	8	≈ 470	180	0.5
OVS-DPDK (tuned)	12.5	8	≈ 255	75	0.1
OVS-DPDK (без tuning)	9.0	8	≈ 355	110	0.3

Як видно з таблиці, fast path дає приблизно двократне зростання PPS у оптимальному режимі. Критично, що різниця між tuned та «без tuning» також велика: це підтверджує тезу, що DPDK – не «чарівна кнопка», а інженерна технологія, яка потребує дисципліни розгортання (NUMA, PMD, queue mapping). Тому у конструктивній частині доречно формалізувати вимоги до впровадження, щоб уникнути деградації після релізу. [61] [63] [65]

3.5 Результати застосування SR-IOV/offload: ефект зменшення накладних витрат vSwitch та стабілізація tail latency

Третя серія експериментів оцінює апаратне прискорення через SR-IOV та offload. Ключова гіпотеза: якщо VNF отримує VF і виконує обробку ближче до NIC, зменшується кількість програмних шарів і контекстних переходів, а також знижується затримка у хвості розподілу. Найбільш помітний ефект очікується у сценаріях low-latency та високого PPS. [61] [63] [68]

Оцінювання проведено для двох моделей:

- SR-IOV VF passthrough (direct I/O до VM/CNF);
- SR-IOV із representor/switchdev (коли частина керування потоками доступна з

боку хоста).

Практично representor підхід може забезпечити баланс між продуктивністю і керованістю (SFC/моніторинг). [68] [82] [85]

Таблиця 3.4 – Приклад результатів SR-IOV/offload (IMIX, вимога по p99) [68] [85]

Підхід	Throughput, Gbps	p99, мкс	CPU cores (dataplane)	Стійкість (tail)	Примітка
Kernel dataplane	62	240	10	Нестабільна	Можливі сплески при IRQ/softirq
OVS-DPDK	94	110	10	Стабільна	Потрібен pinning/NUMA
SR-IOV VF	98	70	8	Дуже стабільна	Менше шарів, direct I/O
SR-IOV representor	96	85	9	Стабільна	Компроміс: керованість vs overhead

Результати показують, що SR-IOV/offload може давати найменшу p99 затримку при порівнянному throughput. Однак практична ціна – зменшення гнучкості на рівні vSwitch та ускладнення деяких сценаріїв SFC і live migration. Тому доцільно рекомендувати SR-IOV як технологію для edge-вузлів або для high-PPS VNF, де вплив tail latency критичний, а також для випадків, коли сервісна архітектура допускає більш статичне розміщення функцій. [61] [63] [68]

3.6 Розрахунок економічного ефекту впровадження оптимізацій: TCO, break-even, аналіз чутливості

Після отримання технічних результатів доцільно перевести їх у економічну площину: скільки серверів потрібно для заданого навантаження і яка сукупна вартість володіння у горизонті 3–5 років. Для цього використовується модель TCO, наведена у попередньому розділі, але тут вона застосовується до конкретної

«проектної» ситуації: забезпечити задану пропускну здатність з виконанням SLO. [89] [90] [91]

Нехай вимога: T_req = 100 Gbps при p99 ≤ 100 мкс. Тоді за результатами вимірювань можна прийняти оцінки: kernel dataplane досягає SLO лише з N_srv=3; OVS-DPDK – з N_srv=2; SR-IOV/offload – з N_srv=1. Економічний ефект має дискретний характер: головна вигода полягає у скороченні серверів, а не лише у «відсотках» зростання throughput. [61] [63] [65]

Для розрахунку використано набір параметрів (приклад): C_srv = 6500 USD; C_nic = 1500 USD; C_smartnic = 5000 USD; PUE = 1.4; c_e = 0.12 USD/кВт·год; α_maint = 10%/рік; N = 3 роки. Річні енерговитрати визначаються як: C_energy = P_srv · 8760 · c_e · PUE. Далі: TCO = CAPEX + N·(C_energy + α_maint·CAPEX).

Таблиця 3.5 – Зведення техніко-економічного ефекту (приклад для 100 Gbps, N=3 роки)

Режим	N_srv	CAPEX, USD	OPEX/рік, USD	TCO_3y, USD	Економія до kernel, %
Kernel dataplane	3	24 000	≈ 4 500	≈ 37 500	0
OVS-DPDK	2	16 000	≈ 3 000	≈ 25 000	≈ 33
SR-IOV + offload	1	11 500	≈ 1 700	≈ 16 600	≈ 56

Отриманий результат узгоджується з логікою NFV-економіки: на короткому горизонті основний вклад у TCO дає CAPEX сервера та мережевих адаптерів. Енергоспоживання та PUE також важливі, але найчастіше не домінують у 3-річному циклі, якщо ціни на енергію не екстремальні. Таким чином, впровадження OVS-DPDK або SR-IOV має сенс тоді, коли воно зменшує N_srv або дозволяє додаткову консолідацію VNF на тому самому «залізі». [65] [66] [67]

3.7 Інженерні пропозиції та план впровадження заходів підвищення ефективності VNF

На основі отриманих результатів сформовано набір практичних заходів, які спрямовані на досягнення мети дослідження – підвищення ефективності застосування VNF при збереженні керованості та надійності. Заходи згруповано за рівнями впливу: «швидкі налаштування» (low effort), оптимізація dataplane (medium effort), та архітектурні зміни з апаратним прискоренням (high impact).

Заходи рівня конфігурації ОС та NFVI (low effort):

1) CPU pinning для dataplane потоків VNF та/або PMD threads; ізоляція ядер від фонового навантаження (isolcpus/nohz_full).
2) IRQ affinity для черг NIC: закріплення MSI-X векторів на ті самі ядра, що обробляють RX/TX.
3) Увімкнення hugepages (2 MB або 1 GB) та резерв mempool для пакетних буферів.
4) Фіксація governor CPU (performance) та контроль C-states, щоб мінімізувати latency spikes.
5) NUMA-aware розміщення VNF: пам’ять і ядра з того ж NUMA-домена, що й NIC.

Ці заходи практично не потребують CAPEX, але можуть помітно покращити p99 та стабільність. [61] [63] [79]

Заходи рівня dataplane: fast path у user-space (medium effort):

1) Перехід на OVS-DPDK або DPDK-native vSwitch для сценаріїв з високим PPS.
2) Підбір PMD threads та кількості RX/TX черг (queue mapping) під реальний профіль трафіку.
3) Налаштування burst size та mempool кешів для зменшення overhead і покращення L1/L2 locality.
4) Використання flow cache (megaflows) та оптимізація правил, щоб підвищити cache-hit.

5) Формалізація профілю тестів для релізів: обов’язкові p99-пороги, а не лише throughput. [61] [63] [65]

Заходи рівня апаратного прискорення: SR-IOV/offload (high impact):

1) Використання SR-IOV для VNF із критичним PPS/latency профілем (UPF, edge firewall, router).
2) Перехід на representor/switchdev модель там, де потрібен баланс керованості (SFC/моніторинг).
3) Використання апаратних offload функцій (RSS, flow steering, tunnel offload) для зменшення T_io.
4) Розгляд SmartNIC/DPU для великих інфраструктур 100GbE+, де економія серверів дає значний TCO ефект.
5) Впровадження політик безпеки IOMMU/VFIO та стандартних процедур дебагу offload-частини. [61] [63] [68]

Критерії приймання (acceptance criteria) у VNF - це формалізований набір порогів і правил, які визначають, коли результат оптимізації можна вважати придатним для експлуатації (production-ready), а коли - ні. Їх мета не лише "показати вищий throughput", а гарантувати досягнення бізнес-цілей мережі: детермінізм затримок, керованість ризиків, відтворюваність результатів і контроль регресій у CI/CD. [61] [63] [69] [89] [90]

На практиці приймання VNF повинно враховувати специфіку телеком-домену: вимоги до tail latency (p95/p99/p99.9), вплив пакетного профілю (64B vs IMIX), чутливість до NUMA/IRQ, а також різні режими dataplane (kernel, OVS-DPDK/DPDK, SR-IOV/offload). Саме тому критерії приймання доцільно будувати у вигляді матриці: профіль навантаження -> ключові метрики -> допустимі пороги -> метод перевірки -> статус PASS/FAIL. [61] [62] [63] [65] [66] [68] [79]

Логіка побудови критеріїв приймання є прямою: спочатку фіксуються SLI/SLO (service level indicators/objectives), після чого вони транслюються у вимірювані метрики стенда і процедури тестування. Наприклад, якщо сервісний SLO звучить як "p99 < 200 мкс при 90% пікового навантаження", то у тест-плані

мають з'явитися: профіль трафіку, тривалість steady-state, мінімальна кількість прогонів, статистична обробка та чіткий поріг PASS/FAIL. [89] [90] [91] [61]

Рекомендовано застосовувати двошарову структуру: (1) "hard gates" - обов'язкові пороги, без яких реліз заборонений; (2) "soft signals" - індикатори деградації, які не блокують реліз автоматично, але потребують інженерного пояснення (postmortem/waiver). Такий підхід мінімізує ризик "підгонки" результатів і знижує кількість аварій, пов'язаних із релізами. [89] [91]

Для VNF типово виділяють три домени критеріїв: продуктивність (performance), надійність (reliability) та керованість/спостережуваність (operability/observability). Якщо критерії охоплюють лише throughput, то система часто проходить приймання, але "ламається" в полі через хвости затримок, нестабільність кешів, або некоректні алерти. [88] [87] [89]

Ключовою помилкою в оцінці dataplane є фокус на середніх значеннях. Для телеком-сценаріїв реальну якість визначають квантілі затримки та стабільність у часі. Мінімальний набір метрик, які мають бути у acceptance criteria: [61] [63] [75]

- Throughput (Gbps) та/або PPS при заданому профілі пакетів (64B, 128B, 512B, 1518B, IMIX). [61] [63]
- Latency percentiles: p50/p95/p99/p99.9 (мкс) з фіксованим методом вимірювання та синхронізації годинників. [61] [63] [75]
- Drop rate / loss (%) у steady-state та в пікових режимах, включно з microburst ефектами. [63] [75]
- CPU cost: завантаження ядер dataplane, cycles/packet (якщо доступно), частка softirq/system time. [79] [80] [65]
- Stability: дрейф метрик у часі, наявність деградації p99 при тривалих прогінних тестах (30-60 хв). [75] [89]
- Resource isolation compliance: pinning, IRQ affinity, hugepages, NUMA locality (перевірка факту застосування налаштувань). [79] [80] [82] [83]

Додаткові (але дуже корисні) метрики: Jitter, time-to-recover після короткого піку, час конвергенції після перезапуску VNF, кількість алертів на одиницю часу та коректність їх тригерів. В cloud-native сценаріях також важливо контролювати

"noise" від оркестратора (kubelet, CNI), який може зрушувати tail latency навіть при незмінному throughput. [82] [86] [87] [88]

Щоб уникнути непорівнюваних результатів, тест-план бажано прив'язувати до загальновизнаних методик. Для throughput/latency традиційно застосовують RFC 2544 та RFC 2889, а для сценаріїв безпеки (firewall) - RFC 2647/3511. Ці документи задають терміни, профілі і базові принципи вимірювань. [61] [63] [71] [72]

Критично важливо описати трафік не лише як "X Gbps", а як комбінацію: розмір пакетів, кількість потоків, L4-параметри, розподіл інтервалів (burstiness), а також наявність encapsulation (VXLAN/GTP-U). Для UPF та edge-сценаріїв корисно мати окремий профіль із тунелями та політиками QoS, оскільки саме вони часто стають причиною падіння PPS. [76] [77] [78] [65] [63]

Практичний набір профілів для приймання VNF можна визначити так: [61] [63] [75]

- PPS-критичний режим: 64B, високий PPS, перевірка p99/p99.9 та drop. [61] [63] [75]
- Bandwidth-критичний режим: 1518B або jumbo, перевірка максимального throughput при збереженні SLO. [61] [63]
- Реалістичний режим: IMIX або реальний trace-подібний розподіл, перевірка tail latency та CPU cost. [63] [75]
- Tunnel/QoS режим: VXLAN/GTP-U, перевірка деградації при encapsulation та policy processing. [76] [77] [78]
- Failure/Recovery: короткі піки, перезапуск VNF, перевірка часу відновлення та стабілізації метрик. [89] [91]

Для кожного профілю слід зафіксувати точку роботи. Найчастіше це 70-90% від максимально можливого навантаження, бо саме там tail latency стає інформативним, а система переходить у режим черг і microburst. Друга контрольна точка - "пік" (100%), щоб оцінити поведінку при перевантаженні та механізми деградації (drop/queue). [61] [63] [89]

Проблема продуктивних тестів у тому, що вони дуже чутливі до шуму: фонові процеси, зміна частоти CPU, міграції потоків, відмінності версій драйверів,

microcode, firmware NIC, різні політики C-state/P-state. Тому приймання повинно включати мінімальний набір правил відтворюваності. [79] [80] [65] [75]

- Кількість прогонів: не менше 3, а для p99/p99.9 - краще 5-7 прогонів. [75] [89]
- Warm-up фаза: перші 1-3 хв не враховувати (прогрів кешів, flow-table, NUMA allocations). [65] [66] [79]
- Steady-state: не менше 10-15 хв для основних тестів та 30-60 хв для критичних релізів. [75] [89]
- Фіксація версій: kernel, DPDK/OVS, драйвери NIC, firmware, BIOS/UEFI налаштування. [65] [66] [79]
- Фіксація топології: NUMA, CPU pinning, IRQ affinity, hugepages, RPS/RSS налаштування. [79] [80] [81] [82] [83]

Статистично рекомендовано публікувати не лише "best run", а медіану та інтервал довіри. Навіть проста оцінка розкиду (standard deviation) часто показує, що заявлене "+10%" насправді знаходиться в межах шуму. Для практичного інжинірингу достатньо вважати істотним покращення, яке перевищує 2-3 sigma або 5-10% залежно від метрики. [75] [61] [89]

Максимальна користь acceptance criteria досягається тоді, коли вони інтегровані у релізний процес як regression gates. Це означає: кожен кандидат релізу проходить стандартизований тест-пакет, результати автоматично порівнюються з базовою лінією (baseline), і реліз блокується при перевищенні порогів деградації. [89] [91] [75]

Для VNF особливо важливо мати щонайменше два рівні тестів: (1) швидкий "smoke" (5-10 хв), який ловить очевидні помилки; (2) повний performance suite (30-120 хв) для нічних прогонів або перед production rollout. Сучасні підходи до CNF також пропонують стандартні каталоги тестів (CNF test suite), які можна використовувати як чек-лист операційної готовності. [86] [89] [82]

Рекомендована canary-стратегія: спочатку rollout на 1-5% трафіку з активним моніторингом SLO, далі - поступове збільшення. Якщо вичерпано error budget або зафіксовано стійке зростання p99, реліз має автоматично зупинятися або відкотитися. [89] [91] [90]

У сучасних телеком-мережах приймання VNF без observability - це прихований ризик. Навіть якщо dataplane швидкий, відсутність коректної телеметрії робить експлуатацію дорогою та небезпечною. Тому acceptance criteria мають включати вимоги до метрик, логів і трасування. [87] [88]

- Наявність метрик рівня dataplane: PPS, drop, queue depth, softirq load, p99 latency (експорт у Prometheus/OpenMetrics). [88]
- Трасування control-plane/сервісних викликів (traces) для діагностики деградацій після релізу. [87] [96]
- Логи з кореляцією по Resource/Pod/VF інтерфейсу та можливістю швидкої агрегації інцидентів. [87] [96]
- Алерти на SLO (latency, loss) з контрольованою кардинальністю лейблів, щоб моніторинг не "впав" першим. [88] [89]

Особливо важливо зафіксувати, що оптимізації типу SR-IOV/offload можуть ускладнювати видимість трафіку. Тому критерієм приймання має бути не лише продуктивність, а й наявність безпечного способу діагностики (наприклад, representor/switchdev або mirrored telemetry). [68] [85] [98]

Нижче наведено приклад матриці приймання для трьох типових режимів dataplane. Пороги є орієнтовними й мають коригуватися під конкретний сервісний SLO, але сама структура (що і як приймаємо) є універсальною. [61] [63] [89]

Таблиця 3.6 – Приклад матриці acceptance criteria для режимів dataplane

Профіль	**Метрика**	**Поріг PASS**	**Поріг FAIL**	**Примітка**
Kernel dataplane (IMIX)	p99 latency, мкс	≤ 250	> 300	Контроль черг/softirq
Kernel dataplane (IMIX)	Throughput, Gbps	≥ 60	< 55	За умови виконання p99

OVS-DPDK (64B)	PPS, Mpps	≥ 40	< 36	Потребує правильного PMD mapping
OVS-DPDK (64B)	p99, мкс	≤ 120	> 150	Чутливо до NUMA/hugepages
SR-IOV/offload (IMIX)	p99, мкс	≤ 80	> 100	Tail стабільність пріоритетна
SR-IOV/offload (IMIX)	Drop rate, %	≤ 0.01	> 0.05	Перевірка microburst

На основі матриці задаються "gates" для автоматичного порівняння з baseline. Найбільш практичний формат - допустима деградація у відсотках або в абсолютних одиницях для кожної метрики. [75] [89]

Таблиця 3.7 – Приклад regression gates для CI/CD (пороги відносно baseline)

Метрика	Поріг деградації (warning)	Поріг деградації (fail)	Коментар
Throughput, Gbps	−3%	−5%	Залежить від шуму стенда
PPS, Mpps	−3%	−5%	Особливо чутливо до NUMA/IRQ
p99 latency, мкс	+5%	+10%	Tail має пріоритет над avg
Drop rate, %	+0.01	+0.05	Стрибки drop - критичний сигнал

CPU cores (dataplane)	+1 core	+2 cores	Впливає на TCO та консистентність

Щоб приймання було юридично та інженерно "захищеним", результати мають бути відтворюваними іншою командою. Для цього в протоколі приймання рекомендовано фіксувати такі блоки інформації: [61] [75] [89]

- Опис стенда: CPU/NUMA, RAM, NIC, версії BIOS, налаштування turbo/C-states, PTP/clock sync (за потреби). [75] [80]
- Верєсії ПЗ: kernel, DPDK/OVS, драйвери, firmware NIC, версії контейнерних образів, конфігурації CNI. [65] [66] [82] [85]
- Політики ізоляції: CPU pinning, IRQ affinity, hugepages, NUMA binding, queue mapping. [79] [80] [82] [83]
- Опис профілю трафіку: packet sizes, flows, rate control, burst patterns, encapsulation. [61] [63] [76]
- Методика вимірювань: тривалість warm-up/steady-state, кількість прогонів, статистика (median, p99, CI). [61] [75]
- Результати: таблиці з метриками, графіки time-series для p99 та drop, порівняння з baseline. [75] [88]
- Висновок PASS/FAIL та перелік ризиків/обмежень (наприклад, втрата видимості при SR-IOV). [68] [89]

У підсумку критерії приймання є "контрактом" між розробкою та експлуатацією: вони визначають, які компроміси допустимі, а які - ні. Для VNF це критично, оскільки без чітких acceptance criteria оптимізації часто перетворюються на разові стендові перемоги з подальшими регресіями в production. [89] [90] [69] [75]

ВИСНОВКИ

У кваліфікаційній роботі досліджено методи підвищення ефективності застосування VNF у телекомунікаційних мережах з позицій продуктивності dataplane, детермінізму затримок (p95/p99) та техніко-економічної доцільності **(TCO)**. Показано, що максимальний ефект дає не один механізм, а комбінація налаштувань NFVI, fast path технологій і апаратного прискорення, підтверджена вимірюваннями в режимі виконання SLO/SLA.

1. **Підтверджено проблему kernel dataplane:** при високих PPS та жорстких вимогах по p99 latency класична обробка в ядрі ОС часто дає нестабільність (tail latency, jitter) і зростання втрат у піках, навіть якщо середні значення продуктивності виглядають прийнятними.
2. **Доведено критичну роль ізоляції ресурсів (CPU pinning, IRQ affinity, NUMA, hugepages):** ці заходи є базою для стабільної роботи VNF, суттєво знижують p99 та jitter і мають найкраще співвідношення ефект/витрати, оскільки майже не потребують CAPEX.
3. **Fast path (OVS-DPDK/DPDK) забезпечує значне прискорення dataplane** у PPS-критичних сценаріях завдяки зменшенню накладних витрат kernel networking. Разом з тим показано, що результат сильно залежить від якості tuning (черги, PMD threads, NUMA, pinning), тому DPDK слід розглядати як інженерну технологію, а не “готову магію”.
4. **SR-IOV/offload дає найкращий профіль tail latency та високу ефективність**, зменшуючи програмні шари між NIC і VNF. Підхід є особливо доцільним для high-PPS/low-latency вузлів, але вимагає врахування експлуатаційних компромісів (керованість, гнучкість SFC, observability).
5. **Показано, що ефективність VNF треба оцінювати в SLO-режимі, а не за піковим throughput.**Ключовими метриками є throughput при виконанні p99, drop rate і стабільність у часі, оскільки саме хвости розподілу визначають якість телеком-сервісів.

6. **Виконано техніко-економічну оцінку (TCO) та аналіз чутливості.** Встановлено пороговий характер вигоди: найбільший економічний ефект виникає тоді, коли оптимізація дозволяє **дискретно зменшити кількість серверів (N_srv)** для забезпечення навантаження в SLO-режимі.
7. **Сформовано практичну методику вибору оптимізацій: SLO → вимірювання → N_srv → TCO → аналіз чутливості → оцінка ризиків експлуатації.**
   Це дозволяє відрізняти реальні покращення від “маркетингових” і приймати рішення на основі відтворюваних даних.
8. **Запропоновано acceptance criteria для приймання результатів** (мінімальний throughput при SLO, p99-пороги, drop rate, повторюваність прогонів, стабільність у часі, фіксація версій firmware/драйверів/DPDK/OVS), що знижує ризик регресій при впровадженні та подальших оновленнях.

Загалом, найбільш ефективною стратегією підвищення ефективності VNF є: ресурсна ізоляція як фундамент → fast path для масштабного приросту PPS → SR-IOV/offload для мінімізації tail latency, з обов’язковим підтвердженням результатів через SLO+TCO.