ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка та аналіз алгоритмів для ефективного балансування навантаження у хмарних сервісах»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми Інженерія програмного забезпечення
(назва)

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

_____ Олексій ЛУППА
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-64
Олексій ЛУППА

Керівник: Владислав ЯСКЕВИЧ
к.т.н.

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

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Луппі Олексію Андрійовичу

1. Тема кваліфікаційної роботи: Розробка та аналіз алгоритмів для ефективного балансування навантаження у хмарних сервісах

керівник кваліфікаційної роботи Владислав ЯСКЕВИЧ к.т.н.,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, вимоги до хмарних сервісів.

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

1. Аналіз методів побудування інфраструктури і моделей забезпечення якості обслуговування у хмарних сервісах.
2. Методи поліпшення параметрів якості надання послуг у хмарній інфраструктурі.
3. Моделювання та дослідження розподілу інформаційних потоків у хмарних

сервісах.

5. Перелік графічного матеріалу: *презентація*
    1. Архітектура розподілених систем «клієнт-сервер» .
    2. Типова логічна топологія мережі центру обробки даних.
    3. Блок-схема алгоритму роботи методу пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних.
    4. Концептуальна модель системи управління хмарною мережею.
    5. Модель надання сервісу.
    6. Тривалість обслуговування запитів на надання сервісу з урахуванням запропонованих рішень.

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-05.11.23	
2	Аналіз методів побудування інфраструктури і моделей забезпечення якості обслуговування у хмарних сервісах	06.11-19.11.23	
3	Методи поліпшення параметрів якості надання послуг у хмарній інфраструктурі	20.11-03.12.23	
4	Моделювання та дослідження розподілу інформаційних потоків у хмарних сервісах	04.12-10.12.23	
5	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
6	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Олексій ЛУППА
(підпис)

Керівник
кваліфікаційної роботи _____ Владислав ЯСКЕВИЧ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 82 сторінки, 8 таблиць, 32 рисунки, 27 джерел, 2 додатки.

Мета роботи – підвищення ефективності хмарних сервісів за рахунок алгоритмів балансування навантаження.

Об'єкт дослідження – процес балансування навантаження у хмарних сервісах.

Предмет дослідження – алгоритми для ефективного балансування навантаження у хмарних сервісах.

Короткий зміст роботи: У даній роботі було вирішено завдання поліпшення характеристик надання хмарних сервісів, зокрема збільшення стійкості віртуальних топологій центру обробки даних хмарної інфраструктури з метою удосконалення алгоритмів балансування навантаження для підвищення ефективності хмарних сервісів, сформованих за допомогою дистанційно-векторних методів. Це стало актуальним в умовах зростання різноманітності потоків у сучасних гетерогенних мережах для відповіді на вимоги комунікаційних додатків.

КЛЮЧОВІ СЛОВА: ХМАРНІ СЕРВІСИ, СТІЙКІСТЬ ВІРТУАЛЬНИХ ТОПОЛОГІЙ, ЦЕНТР ОБРОБКИ ДАНИХ, АЛГОРИТМИ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ, ЕФЕКТИВНІСТЬ ХМАРНИХ СЕРВІСІВ, ДИСТАНЦІЙНО-ВЕКТОРНІ МЕТОДИ, ЯКІСТЬ ОБСЛУГОВУВАННЯ (QOS).

ABSTRACT

Text part of the master's qualification work: 82 pages, 8 tables, 32 figures, 27 sources, 2 appendices.

The purpose of the work is to increase the efficiency of cloud services through load balancing algorithms.

The object of research is the process of load balancing in cloud services.

The subject of research is algorithms for effective load balancing in cloud services.

Summary of the work: In this work, the task of improving the characteristics of the provision of cloud services was solved, in particular, increasing the stability of the virtual topologies of the data center of the cloud infrastructure with the aim of improving load balancing algorithms to increase the efficiency of cloud services formed using distance-vector methods. This has become relevant in the conditions of increasing diversity of flows in modern heterogeneous networks to meet the requirements of communication applications.

KEYWORDS: CLOUD SERVICES, SUSTAINABILITY OF VIRTUAL TOPOLOGIES, DATA PROCESSING CENTER, LOAD BALANCING ALGORITHMS, EFFICIENCY OF CLOUD SERVICES, DISTANCE VECTOR METHODS, QUALITY OF SERVICE (QOS).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - інтерфейси програмування програм

CoS - клас сервісу

EDA - підсистема розведення обладнання

ETSI – Європейський інститут стандартизації телекомунікацій

FTP - протокол передачі файлів

HDA- горизонтальна розподільча підсистема

HTTP - протокол передачі гіпертекстових документів

IaaS - інфраструктура як послуга

IETF - відкрите міжнародне співтовариство проектувальників

IoT - інтернет речей

IP - інтернет протокол

ITU-T - Міжнародна спілка телекомунікацій

LBVM - локальне балансування навантаження між віртуальними машинами

MDA - головна розподільча підсистема

MPLS - Багатопротокольна комутація на основі міток

NVF - віртуальні мережеві функції

OSI - модель взаємодії відкритих систем

PaaS - платформа як послуга

QoS - якість обслуговування, якість послуг, якість сервісу

RSVP - протокол резервування ресурсів

SaaS - програмне забезпечення як послуга

SLA – угода про рівень якості надання послуг

SMTP - простий протокол пересилання пошти

SOAP - протокол обміну структурованими повідомленнями у розподілених обчислювальних системах

ToS - рівень пріоритету IP, вид послуги

VLAN - віртуальна локальна мережа

VoIP - телефонія на основі протоколу IP

VM - віртуальна машина

ZDA- зонові сегменти з вузлами консолідації

РМ - фізична машина

ПЗ - програмне забезпечення

ТКС - телекомунікаційна система

ЦОД - центр обробки даних

ВСТУП

Актуальність теми дослідження полягає в зростанні популярності та значущості хмарних сервісів у сучасному інформаційному суспільстві. Хмарні сервіси надають різноманітні послуги, включаючи обчислення, зберігання даних та інші, що призводить до збільшення обсягу обробки даних та навантаження на інфраструктуру. Однак ефективне управління цим навантаженням та забезпечення високої якості обслуговування (QoS) залишається викликом.

За останні роки спостерігається стрімке зростання використання хмарних сервісів у бізнесі, науці та повсякденному житті. Однак цей ріст супроводжується збільшенням обсягів обробки даних та підвищенням навантаження на інфраструктуру хмарних платформ. Ефективне управління цими завданнями та забезпечення високої якості обслуговування стає актуальним завданням. Розробка алгоритмів для ефективного балансування навантаження у хмарних сервісах стає ключовою проблемою для забезпечення стабільності та продуктивності хмарних інфраструктур.

Об'єктом дослідження є процес балансування навантаження у хмарних сервісах.

Предметом дослідження є алгоритми для ефективного балансування навантаження у хмарних сервісах.

Метою дослідження є підвищення ефективності хмарних сервісів за рахунок алгоритмів балансування навантаження.

Завдання дослідження:

1. Аналіз існуючих моделей інфраструктури та методів забезпечення якості обслуговування в хмарних сервісах.

2. Розробка методів пошуку шляху та моделей надання сервісів для оптимізації центрів обробки даних.

3. Вивчення та імплементація механізмів балансування навантаження для підвищення якості надання хмарних сервісів.

4. Формування інтегрованої архітектури системи управління ресурсами на основі методів балансування навантаження та системної віртуалізації.

5. Створення імітаційних моделей для аналізу розподілу інформаційних потоків та їх впливу на параметри QoS.

Дослідження буде виконано за допомогою таких методів, як літературний аналіз, моделювання, експериментальні дослідження та аналіз результатів.

Результати дослідження дозволять розробити практичні рекомендації для підвищення ефективності та якості обслуговування хмарних сервісів, що може бути використано для оптимізації роботи бізнес-процесів та підвищення конкурентоспроможності компаній.

Робота вноситиме новизну шляхом розробки та апробації алгоритмів балансування навантаження в контексті хмарних сервісів, що є актуальною та важливою задачею в галузі інформаційних технологій.

1 АНАЛІЗ МЕТОДІВ ПОБУДУВАННЯ ІНФРАСТРУКТУРИ І МОДЕЛЕЙ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ОБСЛУГОВУВАННЯ У ХМАРНИХ СЕРВІСАХ

У першому розділі проведено аналіз та показано важливість завдань забезпечення якості обслуговування для розвитку сучасних хмарних телекомунікаційних мереж. Доведено, що важливим аспектом при покращенні основних показників QoS є зниження затримки наскрізної передачі інформації з паралельним збільшенням стійкості віртуальних топологій ЦОД, що утворюються дистанційно-векторними способами в умовах стрімкого підвищення динаміки потоків у сучасних гетерогенних мережах. Вирішення цієї суперечності можливе шляхом підвищення ефективності балансування навантаження в мережах на основі хмарних архітектур.

1.1 Моделі надання сервісів у хмарній інфраструктурі

З появою обчислювальних пристроїв та комп'ютерів розпочалася нова епоха інформатизації суспільства. Перші програми та операційні системи були монолітними і мали обмежену функціональність. Програми були в повній залежності від апаратної архітектури, для якої ці програми були розроблені, внаслідок чого вони не могли працювати на жодному ПК, який мав відмінну архітектуру.

Компанією IBM в 1980-х р. був створений перший суперкомп'ютер, що має високу продуктивність і велику потужність. Оскільки цей комп'ютер був набагато потужнішим за стаціонарні комп'ютери і дозволяв значно швидше реалізовувати складні обчислення, то його обчислювальні ресурси стали здавати в оренду користувачам за допомогою способів віддаленого доступу. У такому разі термінали клієнтів відповідали за встановлення та розрив з'єднання з суперкомп'ютером, обмін інформацією, подання результатів обчислення у зрозумілій та зручній для користувача формі. Усі обчислення у своїй здійснювалися суперкомп'ютері. Подібну архітектуру стали назвати клієнт-сервером (рисунок 1.1), яка є основою

численних розподілених систем, що є на сьогодні.

Розподілена система є колекцією незалежних елементів, які з'являються для користувача як єдина цілісна система [28]. До розподіленої системи входять компоненти, які вважаються повністю автономними. Користувачі, що взаємодіють з подібною системою, розцінюють її як єдине ціле, що означає, що елементи розподіленої системи зумовленим шляхом повинні між собою спілкуватися і взаємодіяти. Отже, одне з ключових завдань побудови розподіленої системи полягає у забезпеченні взаємодії незалежних елементів.

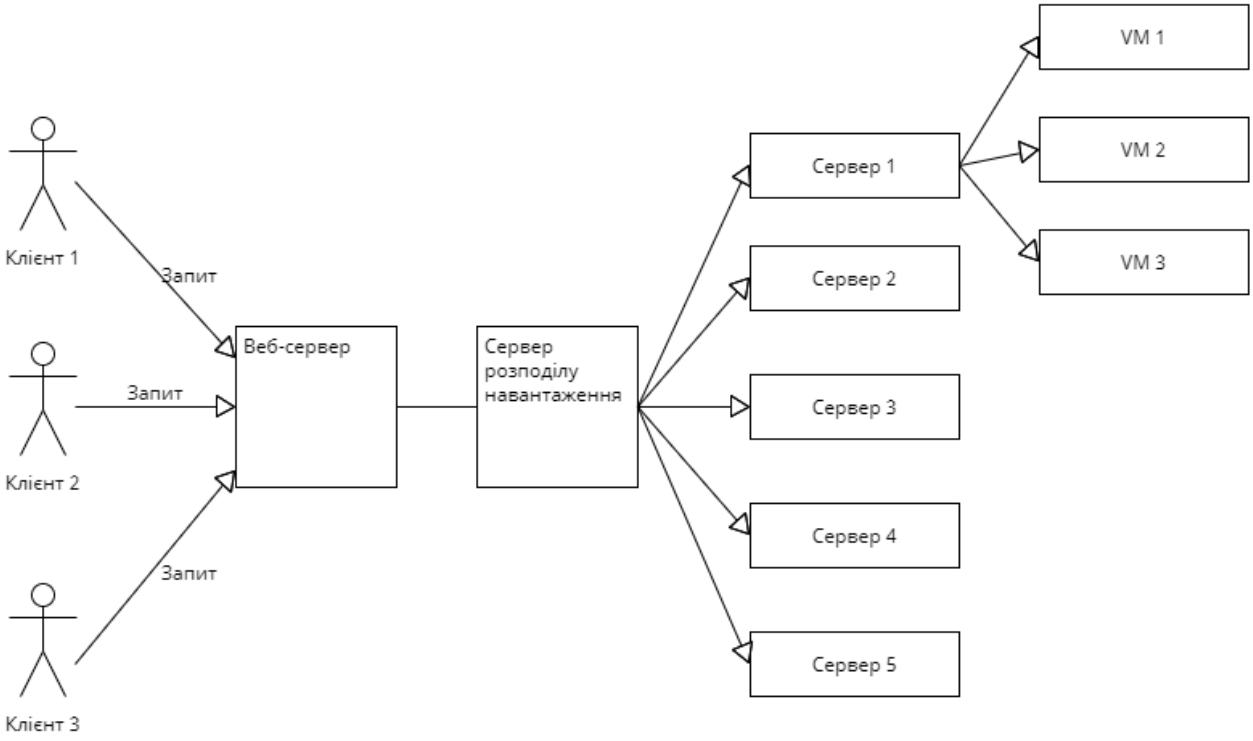


Рис. 1.1. Архітектура розподілених систем – «клієнт-сервер»

Однією з ключових характеристик розподіленої системи вважається відмінність характеристик і параметрів елементів розподіленої системи, і принцип, яким вони спілкуються ховається від користувача. Все це стосується і внутрішньої побудови розподіленої системи. Ще однією важливою характеристикою вважається те, що додатки та користувачі можуть взаємодіяти з розподіленою системою, згідно з узгодженим та уніфікованим способом, незалежно від того, коли і де ця взаємодія має місце.

У той же час архітектура клієнт-сервера проблему монолітності та повної прихильності програм до архітектури апаратного забезпечення не вирішує. Програми, які були створені різними мовами програмування та для різного роду платформ, не мали уніфікованих та стандартизованих засобів, що дозволяють їм на належному рівні спілкуватися та взаємодіяти між собою. Для цього було розроблено По та бібліотеки, що відносяться до логічного рівня Middleware, який дозволив відокремити ПЗ від апаратного завдяки застосуванню стандартизованих протоколів спілкування (CORBA, RMI, RPC, Sockets). ПЗ проміжного рівня надало програмам вищого рівня стандартний інтерфейс для спілкування з іншими програмами по мережі, при цьому архітектура операційної системи та апаратного забезпечення ховалася від цих програм.

Розвиток інформаційних технологій спричинив зміну підходів до створення програм. Програми створювалися за принципом компонентно-орієнтованого програмування. Елементи характеризувалися за принципом модульності, що дало можливість їх модифікації, заміни чи видалення системи без істотних фінансових витрат і зусиль. Втім, одним із ключових недоліків елементів вважається те, що вони надмірно великі і досі мають багато функцій. Це означає, що для модифікації або заміни однієї функції потрібно замінити весь елемент, що спричинило появу програмних елементів таких, як веб-сервіси, які є базою сучасної хмарної архітектури.

Під веб-сервісом мається на увазі функціональний елемент, можливості якого доступні для використання через Інтернет. Перевагою веб-сервісів у порівнянні з іншими технологіями вважається те, що вони не прив'язані до жодної ОС, мови програмування чи апаратної платформи [19]. Тоді, як класичні інформаційні ресурси призначені для прямої взаємодії з людиною, веб-сервіси переважно спілкуються з людиною за допомогою спеціалізованих клієнтських додатків.

Веб-сервіси для спілкування з додатками клієнтів та між собою використовують текстові повідомлення, що базуються на технології XML. Веб-сервіси надають можливість створення складних апаратно-розподілених

програмних комплексів для вирішення різноманітних завдань, вимагаючи від розробника мінімум зусиль та часу.

Веб-сервіс є компонентом хмарної архітектури. Хмарною архітектурою вважають підхід до побудови програм, який ґрунтується на застосуванні розподілених, слабо пов'язаних між собою елементів, за допомогою стандартизованих протоколів та інтерфейсів, що взаємодіють між собою. Як правило, такі програмні комплекси представлені набором веб-сервісів, що спілкуються між собою за допомогою протоколу SOAP, який застосовується, щоб передавати повідомлення у форматі XML і здатний функціонувати поверх будь-яких протоколів прикладного рівня, наприклад, HTTPS, HTTP, FTP, SMTP та ін. .

Будь-який з веб-сервісів призначений для реалізації однієї простої функції, завдяки чому забезпечується принцип модульності, що вважається вкрай важливим у процесі побудови розподілених систем, оскільки модифікація даної функції або її заміна не вимагатимуть великих витрат коштів і зусиль, і не позначаться на роботі всієї системи. Програмування низки веб-сервісів, кожен із яких реалізує конкретну функцію, надає можливість створення розподіленої програми, що має необхідну функціональність. До того ж, подібну розподілену програму можна охарактеризувати масштабованістю та гнучкістю завдяки можливості динамічного додавання функцій (веб-сервісів) до її логіки та можливості встановлення всіляких критеріїв вибору тієї чи іншої функції за певних умов виконання. Подібні розподілені програми отримали назву композитних програм.

З метою розробки та надання різного типу композитного додатку вважається за доцільне використання розподілених обчислювальних систем, заснованих на cloud-технології. Подібні системи мають потужні обчислювальні ресурси та реалізовані у формі ЦОД (рисунок 1.2).

Центр обробки даних є цілою сукупністю інженерних та IT-систем. Така сукупність вважається невід'ємною частиною численних телекомунікаційних структур, вона має забезпечити загальний інформаційний ресурс із гарантованими рівнями доступності, достовірності та безпеки даних. У хмарних мережах ЦОД містять як сервери зберігання даних, а й фізичні сервери, реалізовують обробку

запитів і надання сервісів. Також ЦОД забезпечують побудову необхідної інфраструктури, основні технології віртуалізації, спільне застосування ресурсів (multi-tenancy). Віртуалізація надає можливість надання доступу до мережевих ресурсів як до віртуальних секторів. Це означає, що пристрої або їх компоненти (наприклад, системи зберігання) надаються згідно з запитом, незалежно від свого фізичного розташування та методу фізичного підключення до мережі.

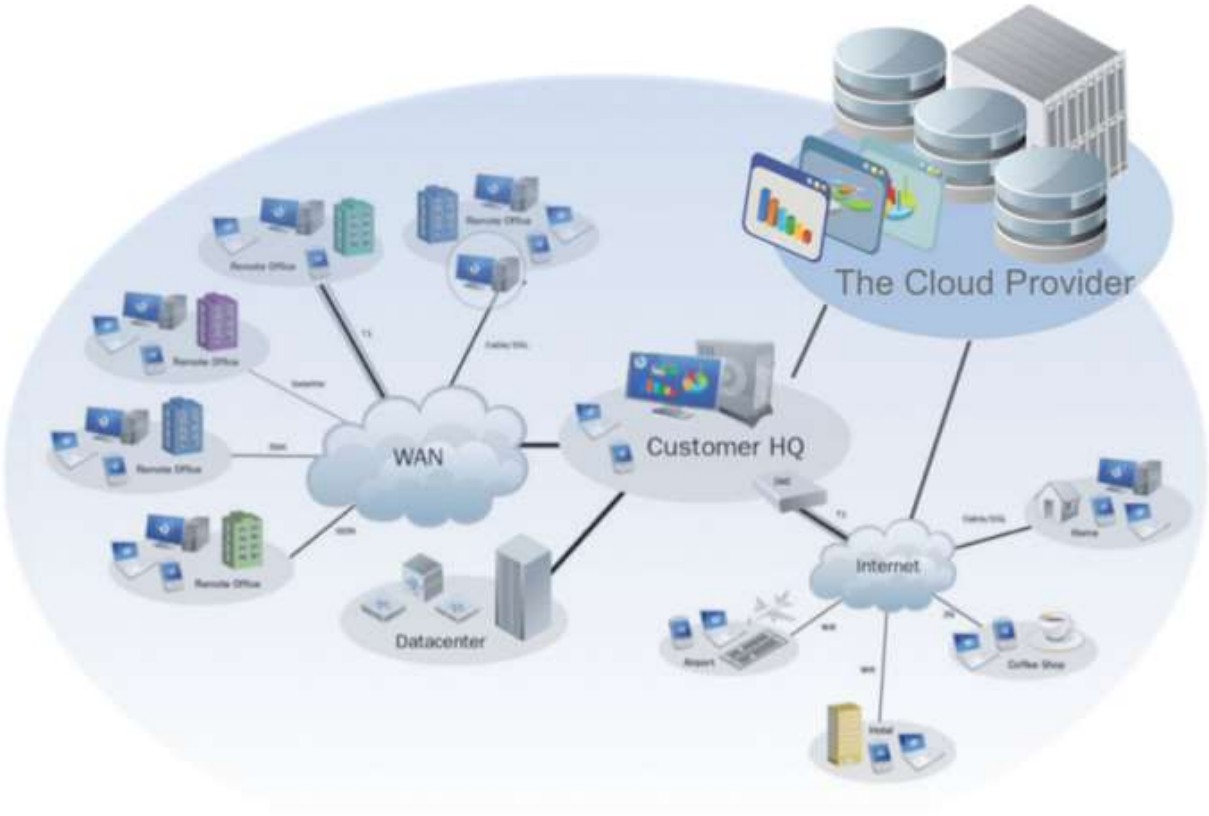


Рис. 1.2. Концепція сервісів cloud-системи

На будь-якому з таких серверів може встановлюватися від однієї до кількох десятків VM, здатних обробляти та задовольняти за допомогою відповідних компонентів або додатків запити на надання сервісу. Втім, логічна топологічна структура подібних ЦОД не завжди вважається стійкою і може динамічно змінюватися, особливо в процес іміграції VM з одного сервера в інший і навіть в інший ЦОД. Як міграції мають на увазі можливість «вимкнути» віртуальну машину в одному фізичному сервері, потім реалізувати, якщо це не було заздалегідь

зроблено, підготовчі операції, які пов'язані з перенесенням набору даних, що відповідає цій віртуальній машині, в інший фізичний сервер, та «включити» віртуальну машину на іншому фізичному сервері, тобто продати її ініціалізацію з присвоєнням, зазвичай, іншого IP - адреси [10]. Перенаправлення запитів на інші логічні, а в деяких випадках і фізичні канали впливатиме на загальний час надання сервісу. Основна перевага таких центрів полягає у високій продуктивності, масштабованості, надійності, доступності, безпеці, прозорості реалізації. На основі подібних ЦОД здійснюється створення різних моделей doud систем, що надають можливість реалізації розподіленого сервісу різної складності.

Перерахуємо деякі поширені моделі хмарних систем:

- ПЗ як послуга (SaaS)- це надання програмних послуг (наприклад, послуги Gmail), що функціонують на основі обчислювальної хмари в оренду. Користувачі застосовують виключно ті функції, які їм необхідні (і відповідно оплачують їх застосування).

- Платформа як послуга (PaaS)- це надання інтегрованої платформи для розробки, тестування, розгортання та підтримки веб-додатків як послуги, організована на основі концепції хмарних обчислень. Ця послуга доступна через Інтернет. Наприклад, Google Apps надає програми для бізнесу в режимі онлайн, доступ до яких реалізується за допомогою Інтернет-браузера, в той час ПЗ і дані зберігаються на серверах Google.

- Інфраструктура як послуга (IaaS)- це надання комп'ютерної інфраструктури (як правило у формі віртуалізації) як послуги на основі концепції хмарних обчислень. На ринку інфраструктури як послуги найбільшими гравцями вважаються Microsoft, Amazon, VMWare, Red Hat і Rackspace. Незважаючи на те, що деякі з них пропонують набагато більше, ніж просто інфраструктуру, вони об'єднані метою продажу базових обчислювальних ресурсів. IaaS складається з 3-х ключових компонентів:

• Апаратні засоби (мережеве обладнання, сервери, клієнтські системи, системи зберігання даних)

• Операційні системи та системне ПЗ (основні засоби управління

ресурсами, засоби автоматизації, віртуалізації);

- Проміжне ПЗ (наприклад, керувати системами).

IaaS має такі ключові характеристики:

- Технології віртуалізації: дозволяють взяти обладнання та його обчислювальні потужності розділити на частини, що відповідають поточним потребам, чим можна збільшити утилізацію наявних потужностей. В результаті з'являється можливість переходу від придбання, керування та амортизації апаратних активів до придбання процесорного часу, дискового простору, мережної пропускної спроможності, необхідної для виконання завдань;

-Інтегровані системи управління:у минулому для управління обладнання різного роду потрібно різне ПЗ управління. Віртуалізація надає можливість реалізації всього набору функцій управління однієї інтегрованої платформі;

-Можливість застосування найкращих архітектур та фреймворків: щоб реалізувати необхідну інфраструктуру, застосовують готові інфраструктури, реалізовані з урахуванням необхідного набору функцій.

IaaS (Infrastructure as a Service) дає можливість відмовитися від підтримки складних інфраструктур ЦОД, мережевих та клієнтських інфраструктур, а також надає можливість зменшення пов'язаних із цим капітальних витрат та поточних витрат. Робота подібної моделі надання сервісів має деякі переваги. Функції віртуалізації та реплікації сервісів впливає на погіршення якості надання сервісів. Важливим аспектом у наданні cloud послуг на базі інфраструктури як сервісу вважається швидкість надання даних сервісів, наявність вільних каналів для їх надання та необхідної смуги пропускання для задоволення потреб користувача. Як швидкість надання сервісів мають на увазі забезпечення мінімального можливого часу надання сервісу, тобто зниження тривалості обробки (обслуговування) запитів, що надходять на обслуговування до ЦОДу. Бурхливий розвиток таких інфокомунікаційних мереж та зміна мережі на рівні IaaS надає все більші запити щодо якості надання послуг. Внаслідок цього питання прискорення надання послуг вважаються актуальними.

1.2 Аналіз моделей забезпечення якості обслуговування інформаційних потоків у хмарній інфраструктурі

Основною метою всіх телекомунікаційних мереж, зрештою, є забезпечення заданих показників якості обслуговування (Quality of service, QoS). Підтримка якості обслуговування в сучасних мережах є досить трудомістким завданням і вимагає узгодженого вирішення цілого комплексу завдань управління та ефективного розподілу мережевих ресурсів. У процесі забезпечення необхідних показників якості обслуговування та підвищення продуктивності телекомунікаційних систем (ТКС) необхідно:

- ведення та постійне оновлення єдиної бази даних про стан ТКС - її топології, завантаженості вузлів, трактів передачі та ін;
- забезпечення високого рівня відмовостійкості мережі;
- реорганізації доступу до використання та збалансованого завантаження доступних мережевих ресурсів;
- автоматизованого контролю параметрів трафіку користувачів відповідно до укладеної угоди про якість обслуговування (Service Level Agreement, SLA);
- раціональної організації та адаптивної зміни стратегій маршрутизації трафіку;
- реконфігурації режимів роботи мережного обладнання, у тому числі налаштування механізмів пріоритетної обробки пакетів на всіх або частині мережних вузлів.

Відповідно до рекомендацій ITU-T E.800, якість обслуговування (QoS) - це певна інтегральна оцінка, яка визначає рівень задоволеності користувача наданої йому постачальником послуг [1]. Це визначення уточнено у рекомендації E.860: "Якість обслуговування - ступінь відповідності обслуговування, що надається користувачеві постачальником і прописаний в угоді між ними" [12].

Фахівці компанії Gsco дали своє визначення терміну "якість обслуговування" - "Здатність мережі забезпечувати необхідний сервіс заданому трафіку у конкретних технологічних рамках (Frame Relay, ATM, Ethernet та 802.1 мережі,

SONET та IP мережі)". Відповідно до змісту RFC 2475 під сервісом слід розуміти набір характеристик передачі пакетів в одному напрямку одним або декількома мережними маршрутами. [3]

Будь-який сервіс описується параметрами якості обслуговування, серед яких:

1. Смуга пропускання (bandwidth) – дає опис номінальної пропускної спроможності середовища передачі і визначає ширину каналу.

2. Затримка в процесі передачі пакета (delay) - є сумарною величиною, що поєднує в собі різновиди затримок: затримка вхідного інтерфейсу, затримка розповсюдження, час очікування в черзі, затримка комутації (час передачі та обробки пакета), затримка формування трафіку, затримка мережі. Деталі виникнення кожної з перерахованих типів затримки наведено у таблиці 1.1.

Таблиця 1.1

Типи затримок

Тип затримки	Причина виникнення затримки
Затримка вхідного інтерфейсу (serialization delay).	Час, необхідне передачі пакета у фізичне середовище. Виникає при виході будь-якого фізичного інтерфейсу.
Затримка розповсюдження (propagation delay).	Час, необхідне передачі інформації на інший кінець каналу. Залежить від середовища поширення та відстані.
Затримка у черзі (queuing delay). величина непостійна	Час, витрачений пакетом на перебування в черзі в очікуванні подальшої передачі (вихідна черга) або в очікуванні можливості перетнути комутаційне поле (вхідна черга) (в очікуванні комутації)
Час пересилання чи обробки (forwarding or processing delay)	Час, необхідний для прийняття вхідного пакета та його обробки, доки пакет не буде поставлений у чергу для подальшої передачі
Затримка, пов'язана із формуванням трафіку (shaping delay).	За умови здійснення формування трафіку, це час, на який пакети, що підлягають передачі, затримуються, щоб уникнути втрат пакетів у середовищі
Мережева затримка (Network Delay)	Затримка компонентів мережі провайдера послуг

3. Варіація затримки під час передачі пакетів (jitter) - різниця у величина сумарної затримки під час передачі різних пакетів тієї самої потоку. Параметр

визначає максимальну затримку при наскрізній передачі пакетів з кінця до кінця.

4. Втрати пакетів (jacket loss) - визначає кількість пакетів, що відкидаються мережею під час передачі. Основні причини втрат пакетів полягають у перевантаженні мережі та ушкодженні пакетів у процесі передачі за допомогою лінії зв'язку. Відкидання пакетів найчастіше відбувається з першої причини - у місцях навантаження, де кількість пакетів, що надходять, значно перевищує верхню межу розміру вихідної черги. Крім цього, відкидання пакетів може викликатись недостатнім розміром вхідного буфера.

Вимірювання цих параметрів здійснюється протягом певного часового інтервалу, причому чим цей інтервал менше, тим більш жорсткі вимоги пред'являються до всіх елементів мережі. Забезпечення QoS при передачі "з кінця в кінець" вимагає взаємодія всіх вузлів на шляху пакетів трафіку і визначається продуктивністю, функціональністю та надійністю "слабкої ланки". Для мереж із хмарною архітектурою, особливо таких як cloud, ці параметри дуже важливі, оскільки надання віддаленого сервісу тісно пов'язане із взаємодією віртуалізованих частин системи під час передачі всіх компонентів сервісу. Гранично допустимі значення параметрів QoS згідно з рекомендаціями ITU-T для різних типів хмарних сервісів наведені в таблиці 1.2.

Таблиця 1.2.

Гранично допустимі значення параметрів QoS

Тип сервісу	Параметри QoS				
	Час встановлення з'єднання, з	Смуга пропускання каналу, Мбіт/с	Ймовірність розриву з'єднання	Затримка, мс	Джіттер, мс
IP-телефонія	0,5..1	до 085	10-3	<400	<150
Відео дзвінки	0,5..1	0,512	10-3	30..100	<30
Мережеве "радіо"	0,5..1	0,256	10-3	<1000	-
Відео на запит	0,5..1	2..20	10-3	30..100	<30
Передача даних	0,5..1	0,128..100	10-6	50..1000	-
IP телебачення	0,5..1	0,512..5	10-6	<1000	-

QoS можна розглядати як міру якості передачі та доступності сервісу в мережі. Виділяють три основні моделі забезпечення якості обслуговування [11]:

1. Модель з негарантованою доставкою (Best Effort Service) – полягає у забезпеченні зв'язності вузлів мережі без гарантії доставки пакета адресату; при цьому відкидання пакета може відбутися при переповненні буфера вхідних або вихідних черг будь-якого комунікаційного вузла;

2. Модель інтегрованого обслуговування (Integrated Service) – реалізує метод резервування ресурсів та забезпечує наскрізну (End-to-End) якість обслуговування. В основі архітектури Int-Serv лежить протокол резервування ресурсів – RSVP (Resource Reservation Protocol);

3. Модель диференційованого обслуговування (Differentiated Service) – базується на методі пріоритезації навантаження. Клієнт може вибрати потрібний рівень якості надання послуги шляхом встановлення відповідного значення поля коду диференційованої послуги (DSCP - Differentiated Services Code Point). Ідея архітектури з диференціацією сервісів полягає у мінімізації службового навантаження з метою виключення затримок.

Перевагою моделі IntServ є забезпечення чітко визначеної та гарантованої пропускної спроможності. Однак збільшення часу встановлення з'єднання, неефективне резервування смуги пропускання заважає широкому використанню RSVP в пакетних мережах роблять таку модель неефективною. Однак, найістотніший недолік IntServ пов'язаний з масштабованістю RSVP, особливо у високошвидкісних магістральних мережах, в яких обсяг ресурсів, які необхідні маршрутизатору для обробки та збереження інформації RSVP, пропорційно підвищується кількості потоків QoS.

Перевагами моделі DiffServ є простота пріоритезації трафіку, можливість масштабування, підвищена надійність. Все це визначає гнучкість та універсальність технології. Однак технологія має певні значні недоліки. Зокрема, при передачі однорідного трафіку його пріоритезація стає не ефективною, адже мережа починає працювати в режимі Best Effort, а через вибіркове відкидання пакетів у періоди сплесків існує велика ймовірність відмови в обслуговуванні

з'єднань з низьким пріоритетом.

Очевидно, що взаємна робота DiffServ та IntServ вважається найкращим варіантом для надання необхідної якості QoS з кінця до кінця. У таких мережах, як cloud, використовується гарантована доставка сервісів користувачу з підтримкою необхідного рівня QoS, тому основним показником якості надання послуги кінцевого користувача є затримка. Вибір параметра ґрунтується на рекомендаціях ITU-T Y.1540. У цьому він виділяється як як основний критерій передачі трафіку реального часу, бо як параметр, найповніше відбиває функціонування мережі. При цьому слабкі місця однієї моделі компенсуватимуться відповідними рішеннями іншої [9].

Час затримки передачі кожної моделі QoS визначається по-різному. Зокрема, під час роботи механізму IntServ враховується його поетапність, де процес передачі включає час передачі сигнального повідомлення Path-t_{path}; час передачі сигналу Resv –t_{RESV}; час передачі блоку абонентських даних-t_{data}:

$$T_{IntServ} = t_{path} + t_{RESV} + t_{data} \tag{1.1}$$

Кожна складова часу може бути представлена як сумарний час затримки на вузлах мережі.$t_{обр\_path}, t_{обр\_RESV}, t_{обр\_data}$ та час затримки передачі по лінії зв'язку повідомлень Path, Resv та даних відповідно -$t_{line\_path}, t_{line\_RESV}, t_{line\_data}$:

$$t_{path} = t_{line\_path} + t_{обр\_path} \tag{1.2}$$

$$t_{RESV} = t_{line\_RESV} + t_{обр\_RESV} \tag{1.3}$$

$$t_{data} = t_{line\_data} + t_{обр\_data} \tag{1.4}$$

При використанні моделі DiffServ враховується, що процес передачі даних мережею$T_{DiffServ}$ включає проведення попередньої (класифікація та маркування) трафіку на граничних вузлах мережі -t_{class_diff} затримки в буферах вузлівt_{delay_diff} та затримки в лінії –t_{line_diff}:

$$T_{DiffServ} = t_{line\_diff} + t_{delay\_diff} + t_{class\_diff} \quad (1.5)$$

Час обробки даних у вузлах мережі визначається затримками буферів. Для розрахунку часу затримки на вузлах можна використовувати теорію дифузійної апроксимації [4]:

$$t_{delay\_diff} = P \cdot \frac{t_s \cdot C_a^2 \cdot C_s^2}{2m \cdot (1-\rho)} \quad (1.6)$$

де m - розмір буфера вузла мережі;

ρ - навантаження системи трафіком;

P - можливість відмови в обслуговуванні через зайнятість приладів; t_s - Середній час обслуговування пакету мережевим пристроєм;

C_a^2, C_s^2 - квадратичні коефіцієнти варіації розподілу вхідного потоку та часу обслуговування відповідно.

Відповідно до теорії телетрафіку, навантаження системи визначається:

$$\rho = \frac{\lambda}{\mu} \quad (1.7)$$

де λ - інтенсивність надходження абонентського трафіку;

μ - інтенсивність обслуговування трафіку пристроєм.

Імовірність відмови в обслуговуванні залежить від завантаження вузлів мережі та може бути обчислена за другою формулою Ерланга. Квадратичні коефіцієнти варіації, відповідно до розподілу Парето, будуть визначатися [16]:

$$C_s^2 = \frac{(1-\alpha)^2 (L^\alpha - k^\alpha)}{\alpha (L \cdot k^\alpha - L^\alpha \cdot k)^2} \cdot \left(\frac{L^2 \cdot k^\alpha - L^\alpha \cdot k^2}{(2-\alpha)} - \frac{\alpha (L \cdot k^\alpha - L^\alpha \cdot k)^2}{(1-\alpha)^2 (L^\alpha - k^\alpha)} \right) \quad (1.8)$$

де α - коефіцієнт ваги розподілу;

L - максимальний розмір блоку даних;

k - мінімальний розмір блоку даних.

При цьому під розміром блоку даних розуміють розмір пакета, що генерується додатком або розміри груп пакетів, що виникають в результаті роботи програми або проходження пакетів мережі.

1.3 Забезпечення параметрів QoS у хмарній інфраструктурі

Зростання попиту на телекомунікаційні мережі висуває нові вимоги до мережевих технологій, які мають забезпечити захищений, надійний і, головне, якісний доступ до інфокомунікаційних послуг, що надаються користувачеві. Відповідно до стрімкого зростання чисельності користувачів та значного розширення спектру послуг оператору зв'язку необхідно модернізувати власну мережу хмарної інфраструктури [2].

Звичайний центр обробки даних умовно має три основні рівні (відповідно до TIA/EIA-942) [18]:

1. MDA / Main Distribution Area є головною розподільчою підсистемою, що забезпечує інтерфейс доступу до ЦОД і розподіляє трафік головної магістралі внутрішніми магістралями. Ця система включає маршрутизатори, кінцеве обладнання операторів зв'язку, магістральні комутатори тощо;
2. HDA / Horizontal Distribution Area являє собою горизонтальну підсистему, що спрямовує трафіку внутрішніх магістралей локальними лініями (довжиною не більше 100 м), виходить в апаратні зони (стійки);
3. EDA / Equipment Distribution Area є підсистемою розведення обладнання, доставляє трафік у робочі області до серверів. З метою обслуговування областей, де необхідні часті переконфігурації, можуть застосовуватися зонові сегменти з вузлами консолідації (ZDA).

Центри обробки даних таких мереж, як cloud, дещо відрізняються. У тому структурі ці граничні рівні нечіткі і взаємно поглащающие. Характерними ознаками хмарного ЦОД є консолідація та віртуалізація серверів, наявність функціонального гіпервізора (оркестратор) та високий рівень автоматизації управління обчислювальною інфраструктурою [21].

Архітектура центру обробки даних, як правило, будується за загальною тришаровою моделлю топології мережі доступу, агрегації та ядра мереж з можливими елементами мережі (комутатори та маршрутизатори). Розглянемо топологію, що показано на рисунку 1.3. Сервери можуть бути підключені через 1

Гбіт лінії до початку стійки (Top of Rack TOR) комутатора, який, у свою чергу, підключається через одну або кілька 10 Гбіт ліній агрегації кінця рядка (End of Row EOR) комутатора. Комутатор EOR використовується підключення між серверів через стійки. Агрегаційні комутатори підключені до комутатора ядра для підключення зовнішніх центрів обробки даних.

Хмарний ЦОД логічно складається із п'яти основних рівнів (рисунок 1.4):

- рівня агрегації;
- рівня доступу;
- рівня додатків;
- рівня зберігання даних;
- рівня оптичних каналів.

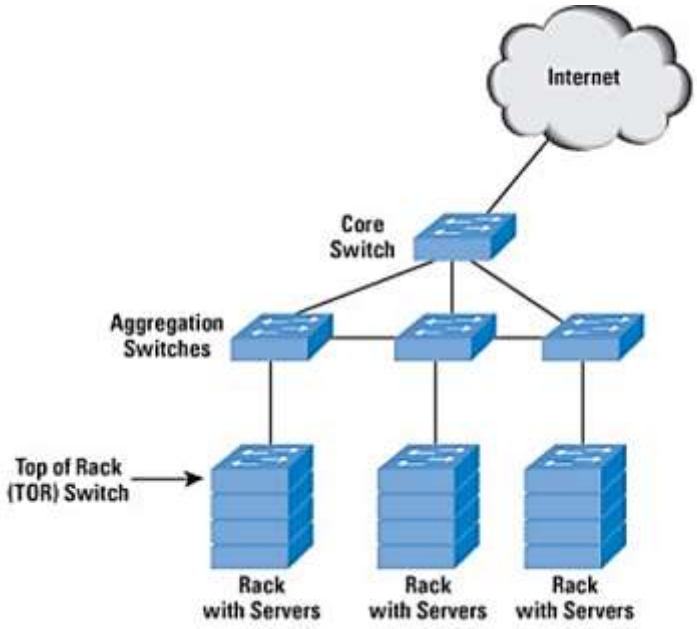


Рис. 1.3. Приклад архітектури мережного комутатора центру обробки даних

Рис. 1.4. Логічна топологія мережі ЦОД

З погляду логічної топології, «зовнішні» сервери головної розподільної підсистеми логічно відокремлені від серверів додатків підсистеми HDA, які, своєю чергою, розділені, від серверів підсистеми EDA [25]. Передача трафіку здійснюється спочатку від клієнта до "зовнішнього" сервера, потім від "зовнішнього" сервера до сервера додатків і далі від сервера додатків до сервера БД. Логічним поділом мається на увазі, кожен рівень вважається особливої функціональної зоною і має свої логічні канали.

Запит на надання сервісу фізичними каналами передається на рівень агрегації, де система управління здійснює пошук та виділення необхідних ресурсів для його обслуговування. Під ресурсами розуміється наявність вільних фізичних серверів із необхідним користувачеві ПЗ. На рівні доступу на основі даних про

вільні фізичні та логічні канали система виділення ресурсів знаходить необхідні ресурси та здійснюється обробка запитів на доступ до необхідного сервісу. Між рівнем доступу та рівнем додатків за допомогою алгоритму маршрутизації здійснюється виділення та передача фізичними каналами, а також запускається алгоритм пошуку логічних каналів для доступу до віртуальних машин. Слід зазначити, що на одній фізичній машині може утримуватися кілька десятків віртуальних машин, які мають здатність виконувати тільки один тип програмного комплексу. Доступ і пошук оптимального як фізичного, і логічного шляху передачі програмних комплексів на віртуальних машинах надання сервісу здійснюється з допомогою алгоритму мінімального зв'язного дерева. Однак такий алгоритм здійснює автоматичне блокування надлишкових у цей час зв'язків для повної зв'язності портів, і, як наслідок, не може забезпечити належної якості сервісу. Крім того, таке блокування зв'язків може призвести до великої кількості втрачених запитів, а велика надмірність службової інформації призводить не лише до завантаженості каналів, а й до збільшення часу надання сервісу. Вибір маршрутів повинен враховувати виділений ресурс у фізичній структурі та зіставляти його з вимогами потоків, тобто контролювати завантаженість. Якщо цієї вимоги не дотримуватися, виникає неефективне використання доступного ресурсу, або погіршення якості сервісу для потоків. Метрика даного алгоритму враховує завантаженість лише до «центральних» (до яких здійснюється найбільше запитів) фізичних машин і аналізує структуру інших сполук, тобто немає можливості щоразу аналізувати логічну топологію між віртуальними машинами. Такий аналіз стає особливо необхідним під час міграції віртуалізованих частин центру обробки даних з одного сервера на інший. Перенаправлення запитів на інші логічні, а й час та фізичні канали, які впливатимуть на загальний термін надання сервісу. Тому така «нестійка» структура ЦОД вносить затримки у процесі обслуговування запитів на надання сервісу і, як наслідок, призведе до погіршення якості обслуговування. Отже, виникає завдання зменшення часу обробки запитів, що надходять на обслуговування в ЦОД, з урахуванням топологічних впливів на загальний термін надання сервісу. Однак обліку структури ЦОД недостатньо: важливим фактором

при цьому є живучість такої структури, адже чим стійкіша структура, тим система швидше виконує та перенаправляє користувачеві необхідний сервіс.

Дослідженням цієї проблеми займалося багато вчених. Так у роботі [27] проведено оцінку та аналіз живучості «типових» структур (зірка, кільце), в яких процес появи нових вузлів не здійснює значного впливу на процес надання сервісу кінцевому користувачеві. У роботі [5] запропоновано модель для оцінки ефективності високо віртуалізованого хмарного центру за пуассонівським розподілом надходження завдань та звичайним розподілом розміру завдань. Модель базується на двоступінчастій техніці апроксимації, де основний не марківський процес спочатку моделюється, як вбудований підлозі марківський процес, який, у свою чергу, потім моделюється, як апроксимований процес Маркова, але тільки в моменти надходження надзавдань. Однак, дана модель не передбачає зміни положення віртуальних машин і, як наслідок, їх вплив на час надання сервісу кінцевому користувачеві. Навіть при великому різноманітті моделей надання сервісів доступність до компонентів сервісу в залежності від стійкості топологічної структури мережі все одно залишається пріоритетним завданням. Зокрема, дослідження доступності фізичних серверів в умовах динамічного розгортання віртуальних машин та зміни їхнього розташування проводиться вченими в [7]. Ще одна робота, присвячена дослідженню доступності складних програм на основі хмарної архітектури. У цій роботі проведено дослідження структури складних програм та способу її оптимізації для підвищення доступності [22]. Однак, навіть за такої кількості досліджень в одній із робіт не проведено паралелей та взаємозв'язку між стійкістю динамічно змінних топологій та якістю надання сервісів користувачам.

Головною характеристикою мереж з хмарною архітектурою є їхня здатність до розподілу навантаження за великою кількістю пулів ресурсів. Всі види програм мають можливість отримати достатню кількість обчислювальної потужності, дискового простору та інформаційних послуг [18]. Однак, спільне використання ресурсів призводить до цілого ряду проблем: велика кількість одночасних запитів на обслуговування сервера може призвести до його зупинки, в той час як інші

сервери досі простоюють. Внаслідок такого розбалансування системи збільшується час надання сервісів кінцевим користувачам, і, як наслідок, погіршується рівень якості обслуговування.

Балансування навантаження сприяє підвищенню продуктивності розподіленої системи в аспекті розподілу інформаційних потоків між безліччю хостів, що взаємодіють [23]. Така система або прагне рівномірно розподілити навантаження на кожен хост і мати дуже малі відхилення від робочого навантаження на інших фізичних хостах, або забезпечує уникнення перевантажень і блокувань на окремих серверах.

Проблема оцінки доступних параметрів у центрах обробки даних досліджується як вітчизняними, а й іноземними дослідниками. Однак, навіть за великого різноманіття моделей надання сервісів, доступність до компонентів сервісу все одно залишається пріоритетним завданням. Над проблемою доступності компонентів на різних рівнях cloud системи працюють багато вчених.

Так, у роботі [13] запропонований алгоритм LBVM дозволяє розділяти навантаження на віртуальні машини між фізичними вузлами по заздалегідь визначеним кластерам. Навантаження на кожному сервері періодично реєструється, причому з різними мітками, і кожні кілька хвилин проводиться запуск централізованого алгоритму, який відстежує, чи відбулася міграція VM чи ні. Коли кількість фізичних хостів збільшується, цей алгоритм буде вузьким місцем системи.

Ще одна робота, присвячена дослідженню доступності складних програм на основі хмарної архітектури. У цій роботі розроблено модель балансування навантаження з розподілом віртуальних машин у центрі обробки даних із використанням методу TOPSIS. Результати показують, що система може досягти більшого балансування навантаження у великомасштабному середовищі хмарних обчислень із меншою кількістю міграцій віртуальних машин. Автори роботи [16] запропонували свій підхід до розподілу навантаження у великомасштабних розподілених файлових системах (DFS). У цій роботі алгоритм балансування працює так, щоб найбільш завантажений вузол перевіряв наявність репліки в

найменш завантаженому вузлі і здійснював перенаправлення частини запитів до такого вузла. [14]. Автори роботи [6] запропонували генетичний алгоритм (GA) щодо стратегії балансування навантаження. Цей спосіб оптимізації балансує навантаження cloud середовища, мінімізуючи робочий цикл сервісу (сумарну тривалість обслуговування запиту всіма компонентами сервісу). Таким чином, GA забезпечує необхідний рівень QoS і ефективно використовує ресурси кожного фізичного сервера. У роботі [17] запропонований новий алгоритм балансування навантаження різнотипних запитів. Запропонована методика досягає справедливого балансу навантаження між віртуальними машинами таким чином, що практично досягається максимальна пропускна здатність. Причому, обслуговування запитів здійснюється відповідно до пріоритетів так, що час очікування обробленого запиту зведений до мінімуму. Автор використовує різні алгоритми планування надходження завдань, щоб отримати найкращу продуктивність. Такий алгоритм балансування навантаження з урахуванням планування надходження завдань та його пріоритезації дозволяє зменшити час відгуку системи обробку запитів. Однак, запропоновані методи базуються на основі дистанційно-векторної маршрутизації та не враховують динамічності зміни структур дата центрів.

Збалансований розподіл навантаження є основним завданням у сфері хмарних обчислень. Він дозволяє забезпечити оптимальне використання ресурсів системи та здійснювати аналіз функціональної доступності кожного окремого елемента. Дослідження доступності фізичних серверів за умов динамічного розгортання VM проводиться вченими Bruno Parreira, Jorge Carapinha, Miguel Dias, Joao Soares, Susana Sargento Carapinha [15]. Їхня робота присвячена розробці платформи Cloud4NFV з використанням функцій NVF. Ця платформа включає моделювання NVF, як інфраструктурних ресурсів. Однак дана платформа та алгоритм на основі якої вона працює, не враховує доступних вільних апаратних чи програмних ресурсів кожної фізичної машини. Робота [20] пропонує своєрідну реалізацію адаптивної динамічної міграції віртуальних машин. Пропонується алгоритм розподіленого балансування навантаження на вибірковій основі, що

дозволяє відстежувати міграцію між VLAN для спрощення площини керування та зменшення часу реконфігурації мережі ЦОД. Реалізується проста модель, яка зменшує час міграції віртуальних машин між центрами обробки даних та виконує переміщення віртуальних машин шляхом перетворення їх на Red Hat Cluster послугу. Проте реалізація такого алгоритму балансування навантаження потребує значних витрат апаратних ресурсів і оцінює стану інших фізичних машин.

Саме тому потрібна розробка підходу, який дозволить оцінити показники доступних ресурсів та дозволить підвищити ефективність балансування навантаження, що призведе до зменшення затримки при наданні сервісів та забезпечить необхідний рівень QoS.

Висновки з 1-го розділу

1. Проведено аналіз моделей надання сервісів у мережах з хмарною архітектурою. Встановлено, що одним із ключових способів побудови інфраструктури та взаємодії в таких мережах є «клієнт-серверна» модель, що дозволяє не здійснювати прив'язки сервісних програм, що надаються у вигляді композитного веб-сервісу, до архітектури апаратного забезпечення. Кожен із веб-сервісів призначений для реалізації однієї простої функції, завдяки чому забезпечується принцип модульності, що вважається дуже важливим у процесі побудови розподілених систем, так як модифікація подібної функції або її заміна не вимагатимуть великих витрат коштів і зусиль, і не позначається на роботі всієї системи. Логічно, що для розробки та надання будь-якого типу композитного додатка вважається за доцільне використання розподілених обчислювальних систем, заснованих на cloud-технології. В результаті аналізу обґрунтовано, що найважливішим компонентом у cloud мережах є центр обробки даних, який забезпечує обробку та побудову необхідної інфраструктури, основних технологій віртуалізації та спільне застосування ресурсів (multi-tenancy). Від його роботи безпосередньо залежить підтримка та забезпечення відповідного рівня якості обслуговування. Функції віртуалізації та реплікації сервісів впливають на погіршення якості надання сервісів.

2. У роботі проаналізовано моделі забезпечення якості обслуговування інформаційних потоків у телекомунікаційній мережі. Досліджено основні параметри, що характеризують QoS у мережах із хмарною архітектурою, які є базовими при наданні сервісів користувачам cloud мережі та технологій за допомогою яких у сучасних пакетних мережах реалізуються методи забезпечення гарантованої якості обслуговування. Встановлено, що взаємна робота IntServ і DiffServ моделей є оптимальним варіантом для надання необхідної якості QoS під час передачі запитів з кінця до кінця. Така «гібридна» модель обслуговування дозволить забезпечити гарантовану доставку сервісів із підтримкою необхідного рівня QoS. За основний критерій якості обслуговування вибрано значення часу затримки передачі, що базується на рекомендаціях ITU-T Y.1540. При цьому, він виділяється не тільки як основний критерій передачі трафіку реального часу, а як параметр, найбільш повно відображає функціонування мережі.

3. Розглянуто основні недоліки та переваги при застосуванні існуючих методів забезпечення параметрів QoS у cloud мережах. Сформульовано основні невирішені завдання щодо забезпечення якості обслуговування в таких телекомунікаційних мережах. Зростання потреби в інформаційно-комунікаційних додатках реального часу, у тому числі в рамках реалізації концепції IoT, формує вимоги щодо зниження затримки наскрізної передачі інформації з паралельним збільшенням стійкості віртуальних топологій ЦОД, що утворюються за допомогою дистанційно-векторних методів в умовах стрімкого підвищення динаміки потоків у сучасних гетерогенних мереж. Вирішення цієї суперечності можливе шляхом підвищення ефективності балансування навантаження в мережах на основі хмарних архітектур.

2 МЕТОДИ ПОКРАЩЕННЯ ПАРАМЕТРІВ ЯКОСТІ НАДАННЯ ПОСЛУГ У ХМАРНІЙ ІНФРАСТРУКТУРІ

У розділі запропоновані моделі, методи та алгоритми підвищення параметрів якості надання послуг у мережах із хмарною архітектурою. Розвинено спосіб пошуку шляху передачі за критерієм мінімальної затримки центрів обробки даних. Запропоновано модель надання сервісу на основі методу пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних. Розроблено інтегровану архітектуру системи управління ресурсами з використанням функцій NVF. Розроблено метод балансування навантаження на основі формалізації ресурсів ЦОД.

2.1 Топологічно-динамічний пошук шляху за критерієм мінімальної затримки для центрів обробки даних хмарної інфраструктури

Нехай сервіс є об'єктом для надання послуги клієнтам хмарної мережі, ключовою характеристикою якого вважається тривалість обробки запиту tобр. Докладніше розглянемо атомарний сервіс, тобто сервіс, реалізований однією програмою, встановленої однією VM. Припустимо, користувач надсилає запит на надання сервісу до хмарного центру обробки даних [24].

ЦОД ініціалізує виділення фізичного сервера (PM), що містить необхідний тип сервісу чи додатку. При надходженні декількох запитів на ту саму PM продуктивність подібних сервісів знижується, оскільки доступ сервісів до ресурсів фізичної машини здійснюється за способом тимчасового поділу. Як наслідок, при збільшенні кількості сервісів на одній PM та при збільшенні інтенсивності надходження запитів на цю PM, а також при підвищенні інтенсивності надходження запитів на будь-який із сервісів підвищується час обробки запитів кожним сервісом. При зниженні продуктивності сервісу, система управління цей переносить в іншу VM чи PM. Логічна топологія не змінюється, проте дані

проходять іншими фізичними каналами. У цьому випадку загальний час передачі сервісу від користувача центру обробки даних і назад розраховуватиметься:

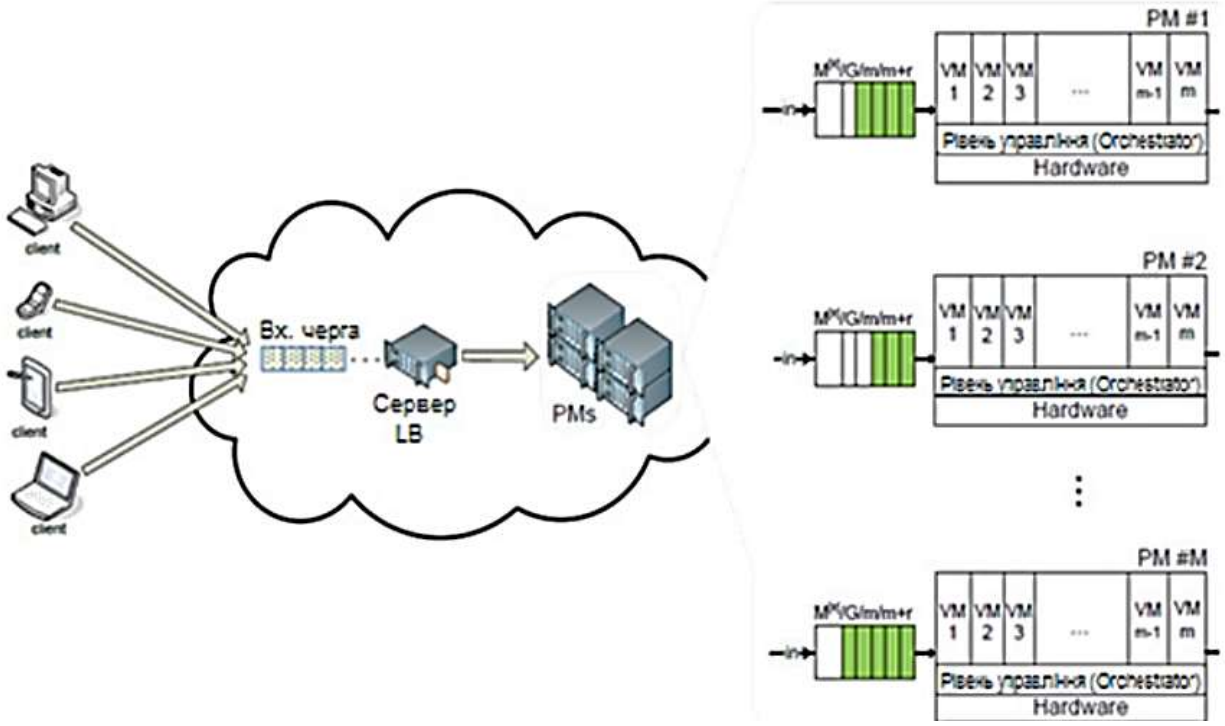


Рис. 2.1. Принцип надходження запитів до хмарного центру обробки даних

$$t_{пер} = \sum_{1}^{n} t_{комут.} + \sum_{1}^{n-1} t_{п.к.з.} + t_{обр.} \tag{2.1}$$

де n-вважається кількістю запитів; а $t_{комут.}$- часом проходження запиту через систему комутації; $t_{п.к.з.}$- часом пошуку каналів, якими реалізовуватиметься передача; $t_{обр.}$- часом обробки запиту, який вважається сумою часу обробки запиту сервісом, що складається з k атомарних сервісів

$$t_{обр.} = \sum_{1}^{k} t_{а.с.} \tag{2.2}$$

Оскільки змінюється оптимальний шлях передачі, це тягне у себе збільшення часу пошуку каналів, з яких реалізовуватиметься передача -$t_{п.к.з.}$, що у свою чергу призведе до підвищення сумарного часу передачі та виникнення затримки (рисунки 2.1-2.2).

Для вирішення цієї проблеми пропонуємо здійснювати пошук шляху згідно з критерієм найменшого часу проходження [26]. Основу даного методу становить метод розрахунку оптимального шляху передачі на базі даних про поширення інформації та зміни у топології мережі. Такі дані становлять єдину метрику маршруту, що виявляє компроміс між властивостями трафіку і вибором оптимального маршруту, оскільки ймовірність одночасного існування потоків з максимальним сервісом на маршрутах із загальною лінією мала. Оптимальність шляху забезпечується кращим показником загальної метрики, при цьому потенційний обсяг доступних ресурсів є набагато більшим за необхідний [16].

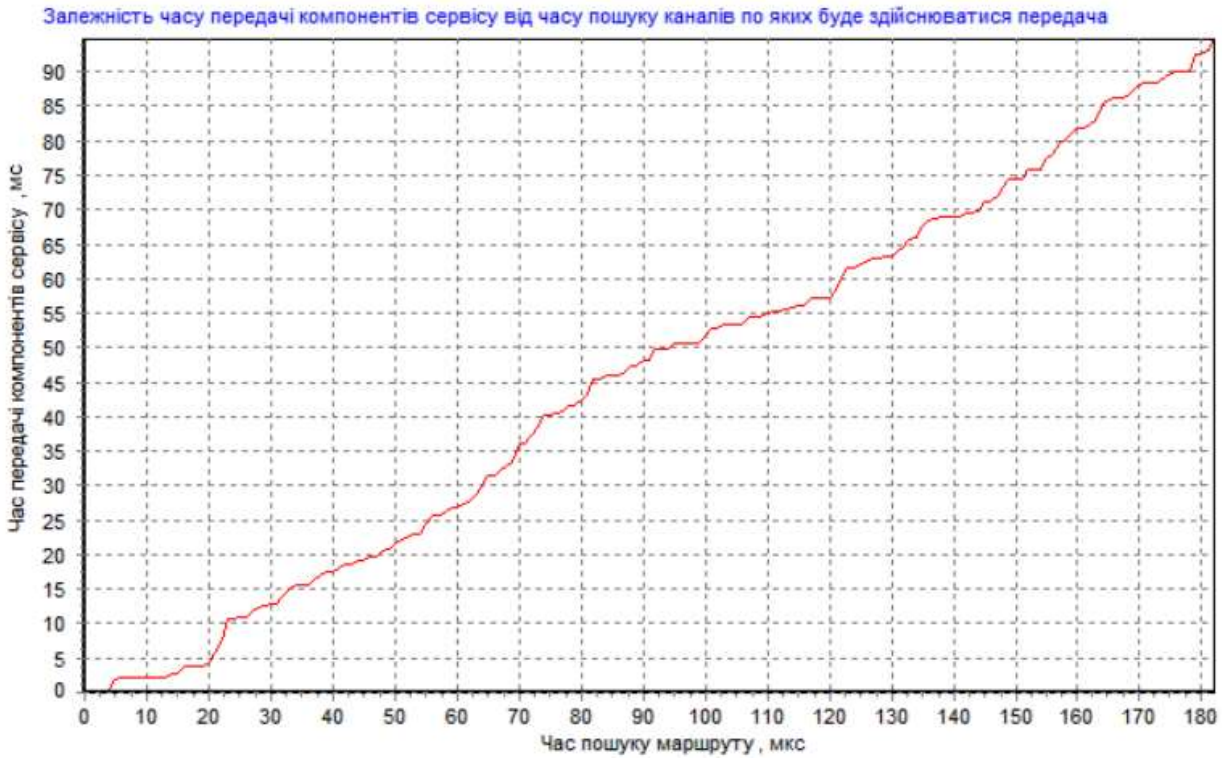


Рис. 2.2. Залежність часу надання сервісу від часу пошуку каналу, яким будездійснюватись передача

Метрика представляється як модифікованої метрики протоколу EIGRP, яка враховує поточну затримку на передачу кожного компонента сервісу на інтерфейсах мережевих вузлів:

$$M = \left(K_1 * \min C + \left(\frac{K_2 * \min C}{256 - L} \right) + K_3 * D \right) * \left(\frac{K_5}{R + K_4} \right) * 256 \quad (2.3)$$

де K_i - Коефіцієнти, які задає адміністратор для коригування композитної метрики; $\min C$ - мінімальне значення пропускної спроможності на шляху проходження оптимального маршруту, яким направлятимуться дані; L - завантаженість кожної ланки у мережі; D – сумарна затримка на інтерфейсах; R – надійність шляху.

Ця метрика між маршрутизаторами не передається. Вона розраховується локально на маршрутизаторі і може існувати виключно на ньому. Далі маршрутизатор здійснює передачу лише змінених параметрів та зберігає дані про структуру зв'язків з усіма сусідами у таблиці, надає можливість швидкого пошуку альтернативних маршрутів при відключенні основних. Коли відповідний маршрут не знайдено, маршрутизатор опитує сусідів про наявність альтернативних маршрутів.

Припустимо, що інтерфейс на виході, у процесі передачі від фізичного сервера A до VM, має значення затримки t, при пороговому значенні затримки T, тобто t <T [17]. VM з цілим програмним комплексом внаслідок впливу різноманітних факторів мігрує на фізичний сервер (рисунок 2.3).

При передачі запиту на надання сервісу в динамічно-змінній структурі аналізується наявність логічного маршруту, для якого значення затримки на передачу кожного компонента сервісу на інтерфейсах мережевих вузлів t не перевищує встановлене граничне значення T. Якщо внаслідок порушення стійкості структури такий маршрут відсутній - відбувається перерахунок комбінованої метрики, відповідно до значення затримки і встановлюється вже новий маршрут для передачі запитів. Це дозволяє суворо контролювати тривалість пошуку каналів передачі і, відповідно, підтримувати належний рівень QoS. Даний метод дозволяє зменшити час на пошуки маршруту, не втрачаючи при цьому запитів і не збільшуючи загальний час їх передачі, оскільки адаптивно враховується зміна логічної структури топологічної мережевої платформи.

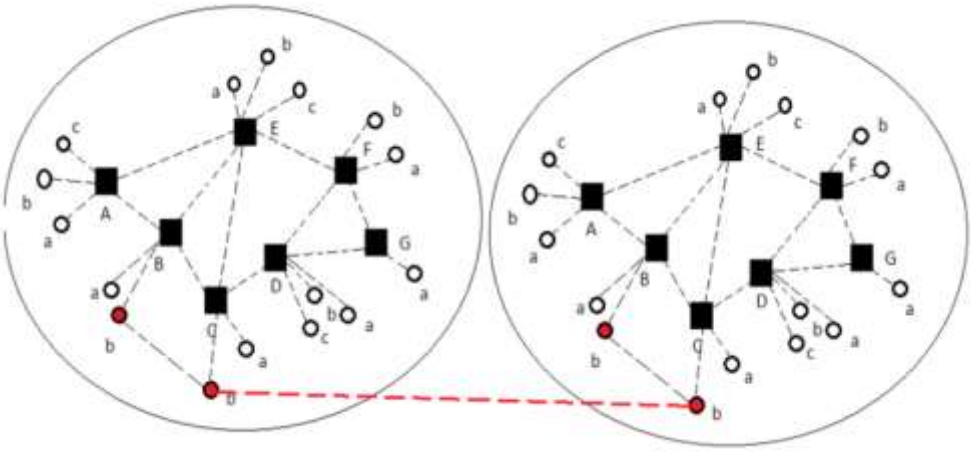


Рис. 2.3. Принцип міграції компонентів сервісу

2.2 Модель надання сервісу на основі методу пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних

Для повноцінної роботи та забезпечення необхідного рівня якості обслуговування інформаційна система повинна мати цілком певний запас стійкості до зовнішніх впливів, що дестабілізують, із зовнішнього середовища [19]. Як на всю інформаційну систему, так і на її окремі елементи можуть впливати різні фактори, що дестабілізують такі як атаки, видалення окремих матеріалів з веб-сайтів мережі Інтернет, знищення або відключення інформаційних серверів, публікація об'єктів (сервісів, додатків, компонентів), які в деякому чином конфігурують вихідну інформаційну систему, або народження нової інформаційної системи, яка може знизити актуальність або знищити вихідну систему.

Кількісні показники живучості перебувають у суттєвій залежності від параметрів, що визначають умови працездатності інформаційної системи. Поточним рівнем працездатності визначаються зміст, якість та кількість функцій, що узагальнюються поняттям «мета функціонування системи». Щоб забезпечити мету функціонування системи, можливе застосування однієї зі стратегій [16]:

F -стратегії - стратегії забезпечення відмовостійкості (Fault-tolerance)

S -стратегії - стратегії забезпечення живучості (survivability).

При формуванні F -стратегії необхідне визначення безлічі станів системи

$S^{(f)} = \{S_v^{(f)}\}$, при яких буде потрібна протидія загрозам порушення працездатності, завдання варіантів розподілу функцій між працездатними елементами інформаційної системи серед безлічі станів $S(f)$.

Стратегія забезпечення відмовостійкості націлена на повну компенсацію передбачених функціональних відмов та забезпечення показників ефективності роботи систем.

У процесі формування S-стратегії для кожного стану з безлічі станів $S(f)$ необхідно додатково розробити рішення, що стосуються функцій системи: звужувати чи ні безліч функцій, що разом складають мету функціонування; як це зробити; спрощувати чи ні алгоритм реалізації функцій тощо.

У процесі аналізу та оцінки функціональної живучості інформаційної системи допускається, що можливе забезпечення необхідних зв'язків між окремими функціональними елементами. Що стосується оцінки стійкості в таких інформаційних мережах, як мережі центрів обробки даних, необхідно забезпечити якомога більшу стійкість зв'язків між кожним з компонентів сервісу. Під стійкістю зв'язків розуміється стійкість топологічних структур під час передачі чи міграції компонентів, тобто щоб існував як фізичний, і логічний маршрут при реплікації компонентів [13].

Позначимо безліч компонентів сервісу (інформаційних функцій), що надаються системою ЦОД через $F = \bigcup_{k \in 1} F_k = \{f_1, f_2, ..., f_n\}$, причому кожна компонента F_k потенційно може виконати безліч функцій $\varphi_n : \{1, 2, ..., p\} \to P(F)$, де $P(F)$ - безліч всіх підмножин F. Якщо $\varphi_n(k) = \{f_{k_1}, f_{k_2}, ..., f_{k_j}\}, 1 \le k_m \le n, 1 \le m \le j$, то функціональна компонента F_k може виконати функції $f_{k_1}, f_{k_2}, ..., f_{k_j}$.

У кожний момент часу F_k виконує певну безліч функцій, що визначаються $\varphi_n : \{1, 2, ..., p\} \to P(F)$. Якщо $\varphi_n(k) = \{f_{k_1}, f_{k_2}, ..., f_{k_j}\}, 1 \le k_m \le n, 1 \le m \le j$, то компонента F_k може виконувати функції $f_{k_1}, f_{k_2}, ..., f_{k_j}$. Коли $\varphi_t(k) = \varnothing$ компонента F_k недоступна.

Кожна $f_{k_m} \in F, m = \overline{1, j}$ є характеризується ефективністю виконання або

кількістю оброблених запитів на компоненти c_{k_m}. Ефективність виконання визначається як $\varphi_{ef} : F \times \{1, 2...., p\} \times P(F) \to C$, де C - деяке числове безліч $\varphi_{ef}(f_{k_m}, k, \varphi_t(k)) = c_{k_m}$ означає, що функціональна компонента F_k виконує функції $\varphi_t(k) = \{f_{k_1}, f_{k_2}, ..., f_{k_j}\}$, тоді ефективність виконання $f_{k_m} \in \{f_{k_1}, f_{k_2}, ..., f_{k_j}\}$ дорівнює c_{k_m}.

Тоді умова доступності надання компоненти (тобто ефективної обробки запитів на компоненті) матиме вигляд:

$$\bigcup_{k=1}^{P} \varphi_n(k) \supseteq F$$

$$\varphi_t(k) \subseteq \varphi_n(k) \forall k = \overline{1, p}$$

$$\sum_{k=1}^{p} \varphi_{ef}(f_{k_m}, k, \varphi_t(k)) \geq c_{k_m}, \forall k_m = \overline{1, n} \qquad (2.4)$$

Якщо компоненти F_k функціонально однорідними, а ціль забезпечення відповідних параметрів QoS, що забезпечується завдяки наявності відповідної кількості доступних функціональних компонентів:

$$R(F_k, t) \geq R^* = const$$

де $R(F_k, t)$ - середня кількість доступних компонентів сервісу в момент часу $t \geq 0, R^*$ - мінімально допустима кількість доступних компонентів, при якій працездатність системи та рівень якості обслуговування не менше певного порогового мінімуму, то оцінка функціональної живучості системи буде функцією:

$$N(F_k, t) = \frac{\overline{\Omega}(F_k, t)}{N_\omega} \qquad (2.5)$$

де $\overline{\Omega}(F_k, t)$ - математичне очікування працездатності системи на момент часу $t \geq 0; N_\omega$ - Сумарна доступність всіх компонентів сервісу.

Структурна живучість сприймається як можливість реконфігурацій мережі, які дозволять створити структуру, що забезпечує виконання критичного рівня підмножини функцій мережі.

У процесі розгляду структурної живучості враховується топологія мережі,

міжкомпонентні зв'язки та доступність елементів. Завдання щодо аналізу структурної живучості можуть зводитися до завдань стійкості, надійності та зв'язності топологічних структур [23].

Аналіз структурної живучості вимагає визначення:

- Структури надання сервісів у певний час, коли виникають небажані на систему;

- Вимог до функціональних можливостей елементів сервісу та системи загалом;

- Вимог до деяких видів ресурсів системи та їхнього взаємозв'язку.

Оцінюватися структурна живучість системи може при окремих припущеннях, які надають можливість спрощення завдання оцінки та зведення її до завдання аналізу зв'язності графів, оцінки можливості формування стійкості структури при небажаних впливах тощо.

У процесі розгляду інформаційної системи слід враховувати кількість структур у цій системі, які можуть реалізовувати критичну кількість інформаційних функцій. Щоб підрахувати цю кількість необхідно виділяти такі структури, іншими словами, з позиції теорії графів і складних мереж - знаходити в мережевих структурах взаємозв'язок компонента, який повинен базуватися на оцінці складності побудови мереж, їх мінімальних та максимальних перерізах, потоках у даних мережах, доступних ресурсах та і т.д.

У процесі дослідження структурної живучості за допомогою графів моделей сукупність компонентів сервісів у системі $G = (V, R)$ представляють як вершину графа $v \in V$, а його ребра $r \in R$ відповідають зв'язкам між ними. Моделювану за допомогою графа систему вважають зруйнованою, коли при видаленні вершин або ребра графа виконуватиметься хоча б одна з умов:

- Граф складається щонайменше з 2-х компонентів;

- Шляхів для певних множин вершин не існує;

- Кількість вершин у найбільшому компоненті графа $G = (V, R)$ менше, ніж деяке задане заздалегідь число;

- Найкоротший шлях перевищуватиме певну задану величину.

Отже, система вважатиметься живучою, якщо ці умови не виконуються. Як правило, структурну живучість різних систем можна охарактеризувати за допомогою різних показників зв'язності. Розрахунок показників, таких як, наприклад, ймовірність зв'язності за умов випадкового існування ребер графа, практично може обмежуватися обчислювальної складності подібних завдань. Одночасно, застосовуючи зв'язки графа, які моделюють систему, можна отримати прості максимальні оцінки необхідних показників.

З урахуванням відключення чи міграції віртуальних машин такий граф буде детермінований, і математично його можна описати лише з допомогою теорії випадкових графів (моделями Балобаші-Альберта чи Ердеша- Реньє [18]).

Нехай структура центру обробки даних представлена у вигляді графа $G = (V, R)$ який має випадкову кількість вузлів (під вузлами розуміється VM) та ребер R (фізичних або логічних каналів), причому $R = \{(r,s)\}$ - множина логічних каналів зв'язку між компонентами сервісів. Будь-який канал зв'язку (r,s) характеризується своєю довжиною - l_{rs} (км), пропускною спроможністю - μ_{rs}, (біт/с, пакет/с), вартістю - $C_{rs}(l_{rs}, \mu_{rs})$ та потоками f_{rs}, причому $f_{rs} \le \mu_{rs} \forall (r,s) \in E$ Кожній вершині V характерні значення $w(V)$ і $\overline{w}(V)$, що відображають поточні та граничні завантаження елемента системи, які залежать від інтенсивності потоків запитів λ. Якщо поточне завантаження $w(V)$ компонента системи досягає максимального значення $\overline{w}(V)$, то компоненти системи виходять з ладу, або відбувається їх міграція на менш завантажену фізичну машину (PM). Потоки запитів на надання компонентів сервісу, що проходять цю вершину, перерозподіляються по «сусідним» компонентам системи. Вихід з ладу компонента системи до теоретико-графів термінології відповідає видаленню з графа системи вершини з інцидентними їй ребрами.

Матриця якості потоків, що передаються між довільними вершинами i і j буде представлено:

$$H = \left\| h_{ij} \right\|, i, j = 1, n \text{ (пакетів/с)}, \qquad (2.6)$$

де h_{ij} - Вважається інтенсивністю потоку, який потрібно передати від i до j.

В результаті передачі потоків запитів на компоненти сервісів у мережі дата-центру виникатимуть:

1) затримка між суміжною парою вузлів t_{rs} ;

2) T_{ij} - Середня затримка між заданою парою вузлів

3) D- загальна середня затримка у системі

Для визначення середньої затримки надання сервісу введемо такі умови та обмеження.

• кількість функціонально-незалежних, незамінних компонентів програми становить k;

• у разі обслуговування запиту компонент буде доступний для інших запитів, тобто компонент перейде до обслуговування наступного запиту паралельно з обробкою поточного запиту;

• на одній віртуальній машині можна встановити лише один компонент або його екземпляр незалежно від функціональності, тому для розгортання окремого компонента використовується нова віртуальна машина;

• на всіх фізичних серверах можуть одночасно виконуватися не більше ніж N віртуальних машин одночасно;

• кількість компонентів всім функцій програми є однаковою;

• вхідні потоки запитів на компоненти надходять за ступінчастим законом розподілу;

• затримкою у вузлах нехтуємо, вважаємо, що вони буфер необмежені ємності;

• часи обслуговування одного і того ж пакета у різних каналах єстатистично незалежними випадковими величинами.

Якщо потік запитів на компонент йде за кількома маршрутами від i до j то λ_{ij} поділяється так що:

$$\lambda_{ij} = \left\{ \lambda_{ij}^{1}, ..., \lambda_{ij}^{k} \right\}$$

$$\lambda_{ij} = \sum_{q=1}^{k} \lambda_{ij}^{q} \tag{2.7}$$

де λ_{ij}^{q} - Частка потоку, що протікає по q-му маршруту.

$$T_{ij} = \sum_{q=1}^{k} p_{ij}^{q} T_{ij}^{q} \tag{2.8}$$

де p_{ij}^{q} - ймовірність вибору q-го маршруту $M_{ij}^{q}; T_{ij}^{q}$ - Затримка на q-му маршруті; $T_{ij} = \frac{1}{\lambda_{ij}} \sum_{q=1}^{k} \lambda_{ij}^{q} T_{ij}^{q}$. Враховуючи що $\lambda_{rs} = \lambda_{ij}^{q}, \forall (r,s) \in \mu_{ij}^{q}$ та за умови, що маршрути не перетинаються, отримаємо:

$$T_{ij} = \frac{1}{\lambda_{ij}} \sum_{q=1}^{k} \sum_{(r,s) \in M_{ij}^{q}} \frac{\lambda_{ij}^{q}}{\mu_{rs} - \lambda_{ij}^{q}} \tag{2.9}$$

В результаті загальна середня сумарна затримка на надання компонентів сервісу в системі визначатиметься:

$$D = \frac{1}{h_{\Sigma}} \sum_{(r,s) \in E} \frac{f_{rs}}{\mu_{rs} - f_{rs}} \tag{2.10}$$

де $h_{\Sigma} = \sum_{i} \sum_{j} h_{ij}$ - Сумарна величина вхідного потоку; f_{rs} - сумарний потік, що протікає каналами (r,s).

Якщо зважати на стійкість кожної вершини, тобто працездатність кожної віртуальної машини центру обробки даних, то під матрицею якості можна розуміти ймовірність стійкості структури системи, при якій забезпечується необхідний рівень параметрів QoS. необхідний рівень параметрів якості залежить від завантаженості кожної вершини часу t, на рівень якої впливає сумарний потік, що протікає каналами (r,s), та пропускної спроможності каналів.

$$D = \frac{1}{P_{cm}} \sum_{Deg.v} \frac{w_t(v)}{C - w_t(v)} \tag{2.11}$$

де P_{cm} - можливість стійкості структури системи, $w_t(v)$ - завантаженість вершин протягом часу t, C - пропускна спроможність каналу зв'язку, $Degv$ - кількість ребер (каналів), приєднаних до вершини v.

Фактично проблема зниження часу обробки (обслуговування) запитів, що

надходять на обслуговування до ЦОДу, з урахуванням топологічної структури даного центру зводиться до зменшення середньої сумарної затримки D. Проте така затримка має задовольняти умови:

$$D = \frac{1}{P_{cm}} \sum_{Deg.v} \frac{w_T(n)}{C - w_T(n)} \le D_{зад} \qquad (2.12)$$

Для пошуку шляху за критерієм мінімального часу проходження максимальна затримка, при швидкості 100 Мбіт\с, що дорівнює 100 мс. Затримка може бути зменшена тільки в тому випадку, коли структура центру обробки даних буде стійкою протягом часу. T . Стійкість структури (живучість) пропонуємо оцінити на основі методу, що включає: оцінку структури мережі (моделі мережі), завантаженості кожної віртуальної машини в момент часу t , і навіть взаємозв'язок вузлів між собою.

У момент часу $t = 0$ слід провести перевірку по всіх вершинах $v \in V$, і сформувати безліч з $\overline{V_1}$ вершин, для яких буде справедливою нерівність $w_0(\overline{v_j}) \ge \overline{w(v_j)}$. У всі наступні моменти часу $t = 1, 2, ..., T$ буде використовуватися правило:

$$w_{t+1}(v_{ij}^j) = w_t(v_{ij}^j) + \varepsilon_j \overline{w(v_j)}, i_j = 1, 2, ..., \left|\overline{\xi(v_j^t)}\right|, j = 1, 2, ..., \left|\overline{V_t}\right| \qquad (2.13)$$

де ε_j .- вважається параметром розподілу завантаження, який може залежати від усіляких чинників, найпростішому разі він поступово розподіляє максимальне завантаження по сусіднім вершинам. Для кожної вершини $\overline{v_j}$ значення параметра ε_j визначається як $\varepsilon_j = \frac{1}{\deg v_j^t}$.

Якщо $w_{t+1}(v_{ij}^j) \ge \overline{w}(v_{ij}^j)$, то це означає, що вершина $v_{i_j}^j$ припинила свою роботу та видаляється з графа, тобто відбувається реконфігурація мережі. Відповідно, стійкість структури системи порушується.

Стійкість до руйнування графа пропонуємо оцінити, як одиниця мінус суму ймовірності взаємопов'язаності вузлів між собою та завантаженості вузлів у момент часу t , не перевищує максимально можливого завантаження системи

$$P_{cm} = 1 - \sum_{i=1}^{n} w_i(t) + P_{зв_i} \qquad (2.14)$$

де $w_i(t)$ - завантаженість вузлів у момент часу t; $P_{зв_i}$ - ймовірність зв'язності вузлів.

Імовірність зв'язків між парою вузлів можна оцінити, як:

$$P_{\text{зв}} = 1 - \prod_{i-1}^{\text{M}} \prod_{i-1}^{A,B} P_{\text{корот.шлях}} * P_{\text{ребер.з}\max k_i} \quad (2.15)$$

де $P_{крат.путь_i}$ - можливість існування і-го найкоротшого маршруту; $P_{ребер.з.\max k_i}$ - ймовірність прокладання і-го найкоротшого маршруту по ребрах з максимальним ваговим коефіцієнтом k.

Якщо завантаженість вузлів у момент часу t менше або дорівнює максимально можливому завантаженню системи, це призведе до збільшення стійкості структури системи, що в свою чергу призведе до зменшення затримки. Використання запропонованого методу дозволить зменшити тривалість пошуку каналів, якими буде здійснюватися передача, оскільки зменшиться затримка, яка закладена в метрику алгоритму пошуку оптимального шляху, що враховує стійкість структури. Це забезпечить швидше передачу компонентів сервісу і, відповідно, прискорить його надання.

Для оцінки ефективності методу необхідно випадковим чином сформувати зв'язки (згідно з умовами моделі мережі) між вузлами та оцінити ймовірність взаємозв'язку спочатку довільних пар вузлів між собою, а потім загальний взаємозв'язок мережі, який визначатиметься, як сума зв'язаності всіх пар вузлів. Оцінка завантаженості кожної віртуальної машини, протягом часу T, буде проводитися з використанням методу Нороса, з урахуванням, що кожна VM, згідно з теорією систем масового обслуговування, буде системою типу G\G\1. Для спрощення вважається, що коефіцієнт варіації інтервалів між запитами та тривалості їх обслуговування постійними величинами. Максимальна завантаженість має визначатись за максимальною кількістю запитів на обслуговування.

Для оцінки стійкості структури ЦОД необхідно врахувати: якщо в моменти часу $T = 1...t$ ймовірність стійкості структури була меншою, то структура не є стійкою і йде перерахунок оптимального маршруту, що веде до збільшення часу

надання сервісу кінцевому користувачеві. Оцінка буде проводитися до того часу, поки граф існуватиме.

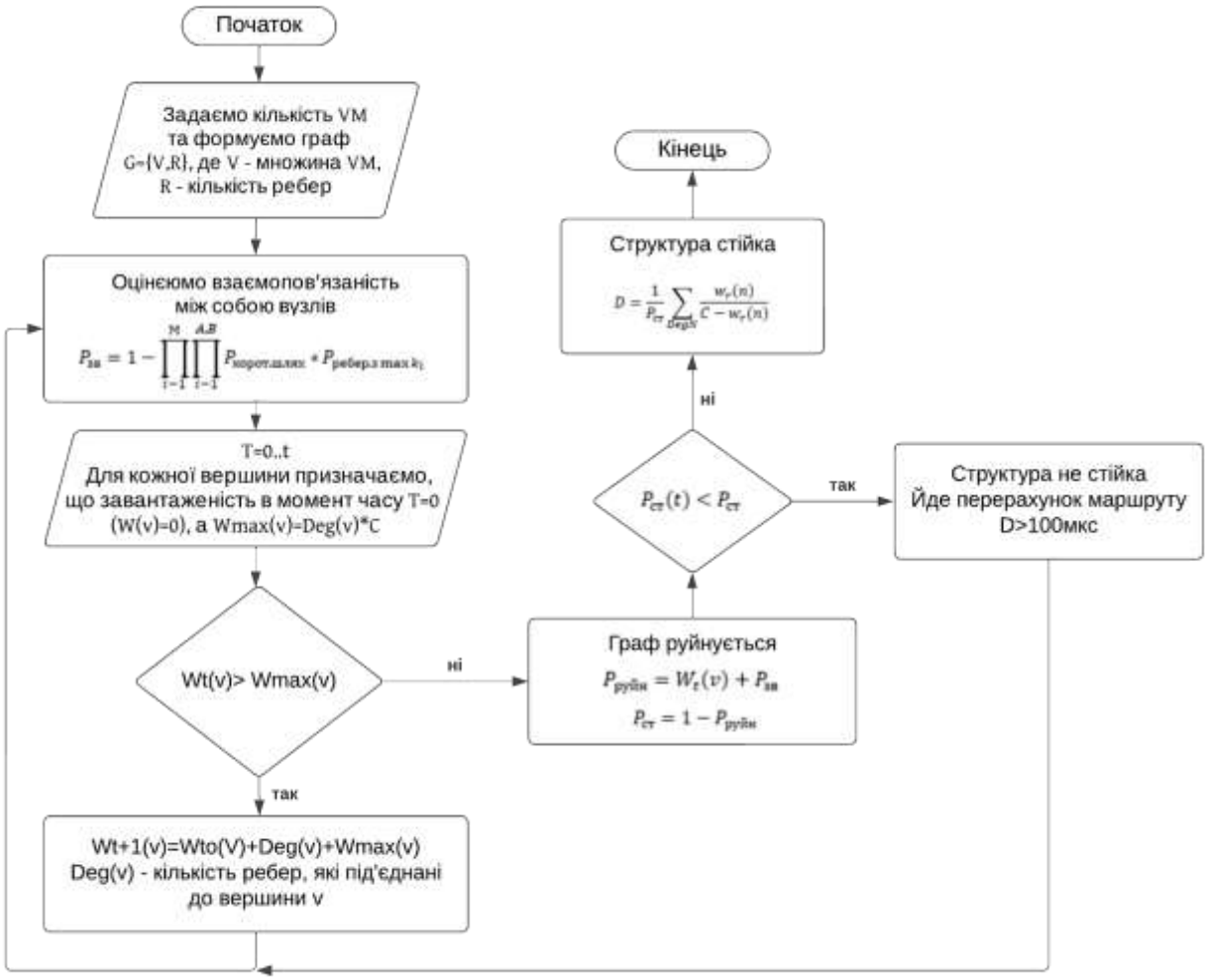


Рис. 2.5. Блок-схема алгоритму роботи методу пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних

2.3 Підвищення якості надання хмарних сервісів із використанням механізмів балансування навантаження

Одна з найважливіших ідей, що лежать в основі хмарних обчислень – це масштабованість, а ключовою технологією, яка забезпечує цю властивість, є віртуалізація [17]. Віртуалізація дозволяє ефективніше використовувати сервери, оскільки потребує нарощування апаратної частини. Віртуалізація також забезпечує міграцію в реальному часі, особливо коли сервер перевантажений, і екземпляр

операційної системи з її програмами може бути перенесений на новий менш завантажений сервер.

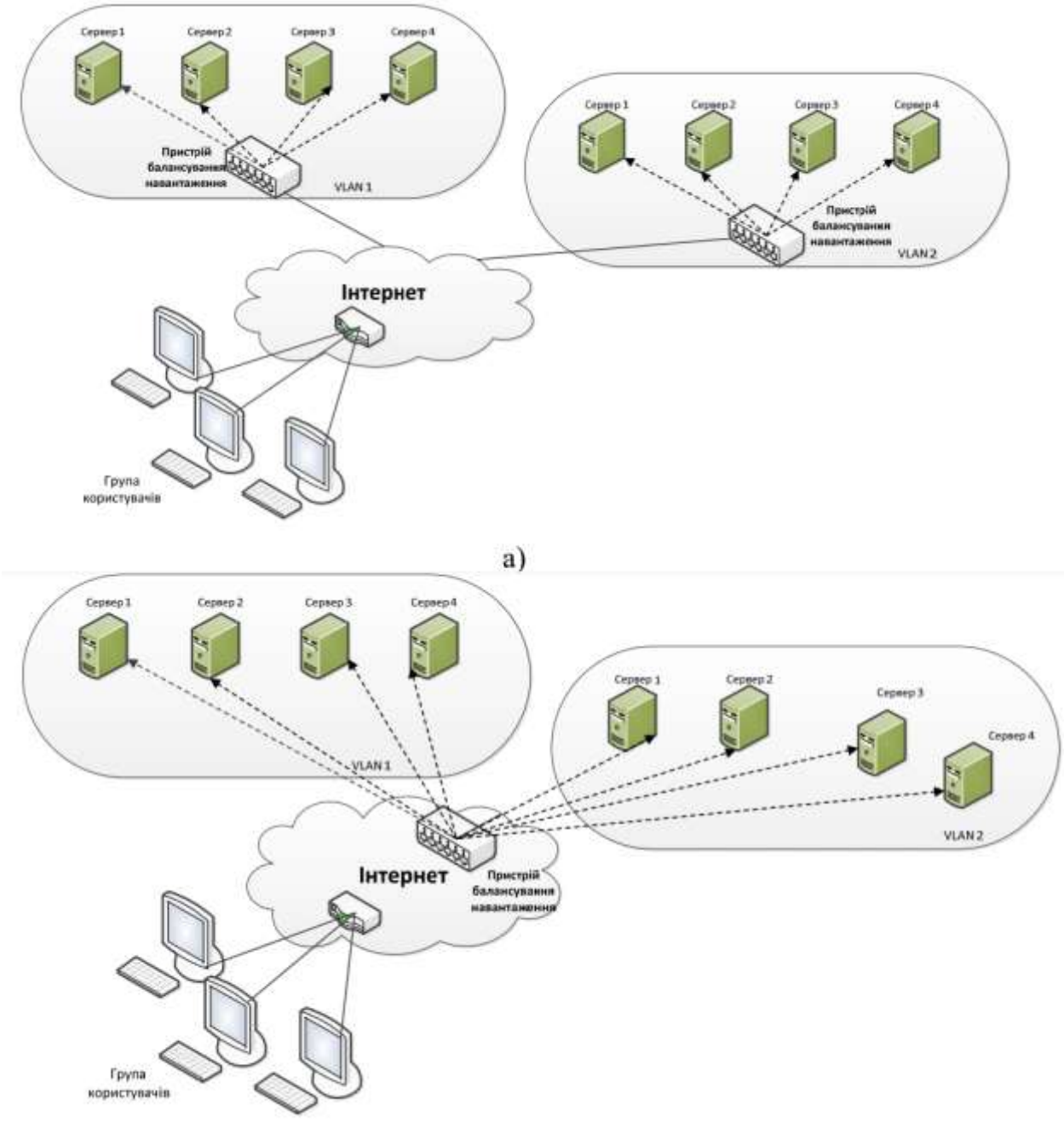


Рис. 2.6. Приклад балансування навантаження між серверами за допомогою пристрою балансування навантаження: а) на локальному рівні; б) на глобальному рівні

Балансування навантаження сприяє підвищенню продуктивності розподіленої системи в аспекті розподілу інформаційних потоків між безліччю

хостів, що взаємодіють [11]. Така система або прагне рівномірно розподілити навантаження на кожен хост і мати дуже малі відхилення від робочого навантаження на інших фізичних хостах, або забезпечує уникнення перевантажень і блокувань на окремих серверах. Балансування навантаження є стратегією розподілу навантаження між двома або більше системами для забезпечення надмірності та високої доступності.

Розрізняють балансування навантаження на двох різних рівнях:

• балансування навантаження локального рівня – серверні балансування навантаження (SLB) по локальній мережі (LAN);

• балансування навантаження глобального рівня - балансування навантаження на сервері (GSLB) через Wide Area Network (WAN) мережі.

Балансування навантаження локального рівня відноситься до стратегій розподілу навантаження у локальних мережах центрів обробки даних. Його можуть здійснювати маршрутизатори або комутатори рівня додатків, які можуть розподілити навантаження між двома або більше серверами, віртуальними машинами або системами. Розподіл здійснюється або за адресою IP або за адресою канального рівня (зазвичай Ethernet MAC).

Ідея балансування навантаження на глобальній мережі полягає в тому, щоб зменшити навантаження на одному центрі обробки даних шляхом глобального розподілу навантаження на кілька центрів обробки даних, які розташовані в інших країнах світу. GSLB може базуватися на двох незалежних системах – система доменних імен (DNS) та Border Gateway Protocol (BGP). Балансування навантаження DNS є, мабуть, найстарішим методом балансування навантаження та існувало задовго до звичайного SLB. DNS є одним із способів розподілу навантаження, або відмовостійкості завдяки надмірності кількості серверів, що забезпечується шляхом управління відповідями DNS-сервера, відповідно до якоїсь статистичної моделі. У найпростішому випадку DNS функціонує, відповідаючи на запити не лише однією IP-адресою, а й переліком з кількох адрес серверів, які надають ідентичний сервіс, що повторюється циклічно. Зазвичай, прості клієнти намагаються підключитися до першої адреси з переліку, отже, різним клієнтам

видаватимуться адреси різних серверів, що дозволить розподілити загальне навантаження між серверами.

Відсутня стандартна процедура для визначення, які адреси застосовуватимуться додатком - окремі сервери намагаються змінити порядок адрес у переліку. Окремі настільні клієнти намагаються отримати альтернативні адреси післятого, як не вдалося встановити з'єднання за 30-45 секунд.

Найчастіше система DNS застосовується для розподілу навантаження розподілених територіальних веб-серверів. Наприклад, у компанії є один домен і три схожі сайти, які розташовані на серверах з трьома різними адресами. Якщо один користувач отримав доступ до головної сторінки, то він буде спрямований на першу IP-адресу. Другий користувач, який звертається до головної сторінки, буде направлений на наступну IP-адресу, а третій - буде направлений на третю адресу IP. У будь-якому випадку, коли IP-адреса представляється, то вона направляється в кінець переліку. Четвертий користувач, відповідно, буде відправлений назад на першу адресу IP, і так далі.

Балансування навантаження на основі BGP системи є ефективним тоді, якщо існують декілька маршрутів з однаковими адміністративними відстанями та цінностями. Маршрутизатор рівня додатків повинен вирішити, який маршрут використовувати, враховуючи, що його кожен наступний вносить свої зміни до таблиці маршрутизації і зможе вибирати різні маршрути. Така стратегія балансування навантаження використовується у Netscape. Для підключення до доступних веб-сайтів прописується в браузері Netscape список можливих та доступних IP-адрес на які можна перерозподіляти потоки трафіку [19]. Однак цей підхід ефективний тільки для вихідного трафіку на сайт Netscape і не підходить для того величезної кількості доступних браузерів, веб-сайтів та хмарних сервісів, які існують сьогодні.

Характеристики обслуговування запитів веб-сервісами (компонентами) розподіленого сервісу повинні відповідати певним вимогам, які б дозволили забезпечити задовільний рівень обслуговування запитів. Зазвичай у разі складних процесів кількість веб-сервісів, які будуть задіяні в процесі обслуговування запиту,

їхня різноманітність, цільове призначення, доступність та характеристики можуть по-різному впливати на якість обслуговування користувача. Кожен веб-сервіс у процесі обслуговування по-різному впливає на якість обслуговування конкретного запиту в залежності від свого поточного завантаження, стану завантаження апаратних ресурсів сервера, продуктивності, ефективності та доступності необхідних ресурсів для виконання операцій. Таким чином, сумарний вплив всіх компонентів на якість обслуговування запиту можна виразити формулою:

$$QoS_s = \frac{\sum_{i=1}^{n} QoS_i}{n} \tag{2.16}$$

QoS_s - середнє значення якості обслуговування на певному проміжку часу для n-ого веб-сервісу.

Це значення можна знайти за допомогою наступної формули:

$$QoS = \sum_{i=1}^{n} QoS_i = \frac{\sum_{j=1}^{m} QoS_j(t)}{m}, t \in (t_1; t_2), \tag{2.17}$$

де $QoS_j(t)$ - якість обслуговування конкретного запиту, який надійшов на обслуговування i-ий веб-сервіс у момент часу t, що належить проміжку часу від t_1 до t_2.

При аналізі якості обслуговування запиту кожному за можливого маршруту можна спостерігати те що, що якість обслуговування кожному з них різна. Тому правильна організація композитного додатка, виконує функції конкретного сервісу та рівномірний перерозподіл запитів між його компонентами, має критичне значення надання послуг абоненту з високою якістю обслуговування.

Однією з головних причин збільшення затримок при передачі є великі обсяги, що передаються між веб-сервісами. Це спричинено технологією XML, яка передбачає використання тегів конструкції опису елементів повідомлення. Опис елементів здійснюється за допомогою тегів-слів, реалізується за допомогою вилучення з композитних програм веб-сервісів, які негативно впливають на процес обслуговування та заміни їх на більш продуктивні, з високими показниками якості

обслуговування. Це дозволяє динамічно налаштовувати продуктивність композитної програми за різних умов завантаження як мережі, так і серверного обладнання, а відповідно покращувати якість обслуговування запитів. Управління цим процесом може здійснюватися за допомогою методів оркестрування або хореографії в залежності від того, який метод використовували для створення сервісу.

У разі застосування методу хореографії веб-сервіси за допомогою спеціалізованих протоколів спілкування визначать можливу заміну для веб-сервісу, продуктивність якого дуже низька.

Однак в умовах динамічного середовища, в якому знаходяться композитні програми та сервіси, фізичні та логічні адреси веб-сервісу, який використовується для доступу до композитної програми, може змінюватися. Це здійснюється з метою підвищення ефективності системи за рахунок міграції різних компонентів системи з одного фізичного пристрою на інший.

Якщо говорити про хмарні послуги та їх компоненти, то при балансуванні навантаження між серверами локального або глобального рівня необхідно врахувати і можливість їх міграції. Зазвичай, рішення про міграцію віртуальної машини приймається на основі даних про завантаженість серверів: наявність вільної пропускної спроможності логічного та фізичного каналу, наявність ресурсів віртуальних машин (оперативної пам'яті, процесора тощо) [15]. У цій роботі пропонується новий підхід для оцінки цих показників, що дозволить підвищити ефективність балансування навантаження через реалізацію інтегрованої архітектури управління з використанням технології NVF (Network Function Virtualization).

2.4 Формування інтегрованої архітектури системи управління ресурсами з використанням методу балансування навантаження та функцій системної віртуалізації

Принцип побудови архітектури управління з використанням функцій NVF

наведено на рисунок 2.7 [15]. Запропонована архітектура управління включає:

- Віртуальний менеджер інфраструктури (VIM), який відповідає за управління ресурсами NFV;
- Менеджер віртуалізації мережевих функцій (NFV) - відповідає за управління життєвим циклом екземплярів NFV (примірника, конфігурації, оновлення, масштабування вгору/вниз). Менеджер NFV несе відповідальність за весь життєвий цикл NFV і через Оркестратора може вимагати ресурсів інфраструктури та взаємодіяти з ними, наприклад встановити програмний компонент або налаштувати NVF.
- Менеджер з аналізу ресурсів (RD) - відповідає за аналіз ресурсів системи. До його обов'язків входить оцінка наявних фізичних, віртуальних, апаратних та телекомунікаційних ресурсів системи. На основі даних про достатню вільну пропускну здатність каналів та доступність компонентів того чи іншого сервісу, приймається рішення або про надання сервісу, або про міграцію його компонентів на менш завантажений сервер. Він надсилає запити до оркестратора, який приймає рішення щодо необхідності міграції віртуальних машин з метою балансування навантаження та оптимального використання ресурсів;
- Менеджер управління міграціями - відповідає за процес міграції віртуальних машин та направляє команду VDE (Virtual Distributed Ethernet) менеджеру про зміну місця знаходження VM (номера порту, за яким вона була доступна, VLAN, до якого вона переноситься тощо);
- VDE менеджер - підтримує та керує роботою VDE комутаторів, які дають можливість об'єднання віртуальних машин у VLAN'n, що полегшує їхнє адміністрування.
- Комутатор VDE є віртуальним аналогом Ethernet-комутаторів. Всі віртуальні пристрої, підключені до VDE, мають можливість бачити один одного, начебто вони є в реальній мережі Ethernet. Вони мають здатність швидко здійснювати конфігурацію мережі без будь-яких впливів на вхідну чергу запитів і незалежно кількості портів. Ця характеристика має значення для мінімізації затримки при передачі трафіку з кінця в кінець. Такий комутатор дозволяє керувати

потоками, регулюючи кількість пакетів між електронними буферами вхідних та вихідних портів.

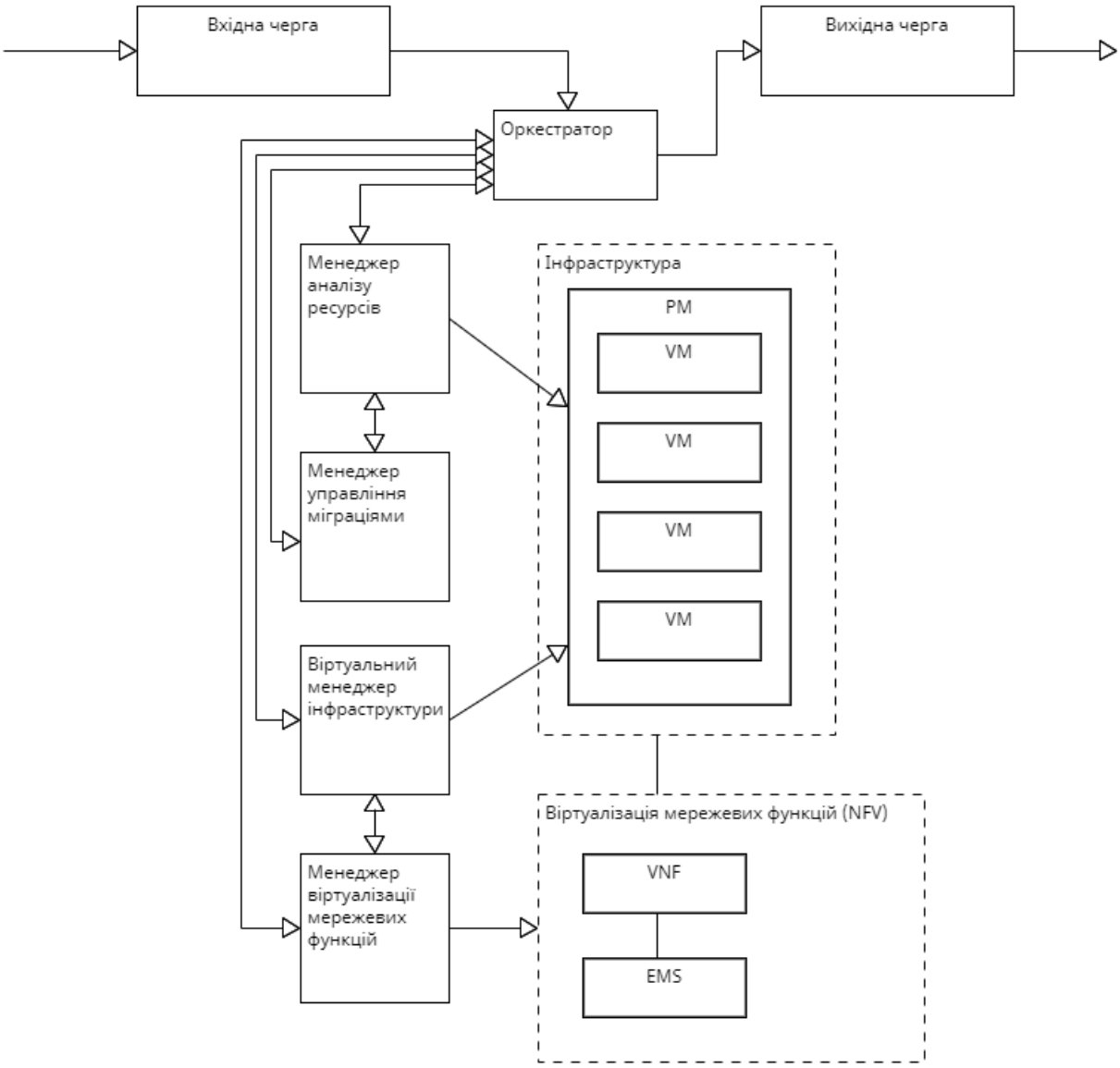


Рис. 2.7. Концептуальна модель системи управління хмарною мережею

• Оркестратор - відповідає за оркестрування та управління ресурсами NFV та надає мережеві послуги на NFVI (Network Function Virtualization Infrastructure). Оркестратор містить спільний інтерфейс, який стежить за розвитком та наданням послуг; обробляє інформацію служб моніторингу та конфігурації; на основі даних від RD приймає рішення про міграцію та надсилає вказівку Менеджеру управління міграціями про перенесення віртуальних машин та VDE менеджеру - про зміну місця їх перебування. З іншого боку, він має інтерфейс до

різних менеджерів NFV, а також до VIM4b. Крім того, він має доступ до сервісу та інфраструктури модуля NFV, який містить інформацію про пріоритетність послуг.

Нехай в хмарну мережу надходить запит на надання сервісу. Інтенсивність надходження таких запитів підпорядковується експоненційному закону розподілу дискретної випадкової величини. Запит потрапляє у вхідну чергу. Оркестратор дає вказівку RD перевірити, чи всі компоненти необхідного сервісу доступні, тобто вистачить обчислювальних потужностей у фізичних, а як наслідок, і у віртуальних машинах, на обробку та надання всіх компонентів. Поки PM (Phisical mashine) RD перевіряє доступність вільних ресурсів, стан фізичних і логічних каналів, і визначає оптимальний шлях передачі обміну компонентами найменш завантажених каналах. У випадку, якщо RD повідомляє про недостатню обчислювальну потужність або відсутність необхідної пропускної спроможності, оркестратор і надсилає інформацію про вільні та доступні ресурси до Менеджера управління міграціями, який здійснює вибір PM та переносить недоступний компонент сервісу (здійснює міграцію VM), залучаючи до цього VDE менеджера . VDE менеджер визначає, в якій VLAN була перевантажена запитами VM і в якій VLAN знаходиться недовантажена VM, і за допомогою VDE комутатора здійснює перемикання логічної частини компонента сервісу та проводить реконфігурацію системи відповідно до проведених змін (тобто здійснює оновлення таблиць маршрутизації). Весь процес міграції та робота системи ведеться під наглядом оркестратор та VIM. Якщо кількість запитів на той чи інший компонент буде невпинно зростати, то VIM може запустити виконання функцій NFV. Технологія віртуалізації мережевих функцій дозволяє створити програмний аналог будь-якого фізичного мережного пристрою та запустити на ньому обробку запитів на надання сервісу. Весь процес міграції та робота системи ведеться під наглядом оркестратор та VIM. Якщо кількість запитів на той чи інший компонент буде невпинно зростати, то VIM може запустити виконання функцій NFV. Технологія віртуалізації мережевих функцій дозволяє створити програмний аналог будь-якого фізичного мережного пристрою та запустити на ньому обробку запитів на надання сервісу. Весь процес міграції та робота системи ведеться під наглядом оркестратора та VIM. Якщо кількість запитів

на той чи інший компонент буде невпинно зростати, то VIM може запустити виконання функцій NFV. Технологія віртуалізації мережевих функцій дозволяє створити програмний аналог будь-якого фізичного мережного пристрою та запустити на ньому обробку запитів на надання сервісу.

Як було описано у розділі, RD відповідає за аналіз ресурсів системи. До його обов'язків входить оцінка наявних фізичних, віртуальних, апаратних та телекомунікаційних ресурсів системи. Такий аналіз здійснюватиметься на основі максимального інтегрального показника доступних ресурсів кожної фізичної машини. Для його визначення необхідно контролювати наявність незайнятих апаратних ресурсів та їх доступність. Під доступністю маємо на увазі наявність вільних ресурсів віртуальних машин та каналів зв'язку. Параметри незадіяних потужностей кожної віртуальної машини, де розташований M_i компонент, розраховуються за співвідношенням:

$$CPU_{pr} = \frac{\sum_{i=1}^{k} M_i * CPU}{\sum_{i=1}^{k} M_i} \tag{2.18}$$

$$RAM_{pr} = \frac{\sum_{i=1}^{k} M_i * RAM}{\sum_{i=1}^{k} M_i} \tag{2.19}$$

де M_i - кількість додатків (компонентів) у i-тій VM (Virtual mashine) CPU_i - Тактова частота процесора VM, яку використовує i-й компонент; RAM_i - Об'єм оперативної пам'яті VM, яку використовує i-й компонент; k - кількість VM однієї PM.

Для оцінки стану вільних ресурсів пропонуємо наведено на рисунку 2.8 Алгоритм. Він забезпечує прискорення обробки запитів та аналізує ступінь завантаженості кожної машини на базі інформації про обчислювальні потужності кожного вузла (у нашому випадку вузлом є VM). Нехай $z = \overline{1, Z}$ - Віртуальна машина, на який знаходиться $i = \overline{1, K}$ компонентів сервісу $j = \overline{1, S}$, а $m = \overline{1, M}$ - фізична машина, на якій знаходиться $Z_m = \overline{1, Z}$ віртуальних машин та

$P = \{P_1, ..., P_n, ...P_N\}, n = \overline{1, N}, P_n = \{e_1, ...e_i, ...e_L\}, l = \overline{1, L}$ - безліч шляхів, де N - кількість шляхів між компонентами i і $i+1, L$ - кількість ребер у дорозі n, які з'єднують VM_z і VM_{z+1}.

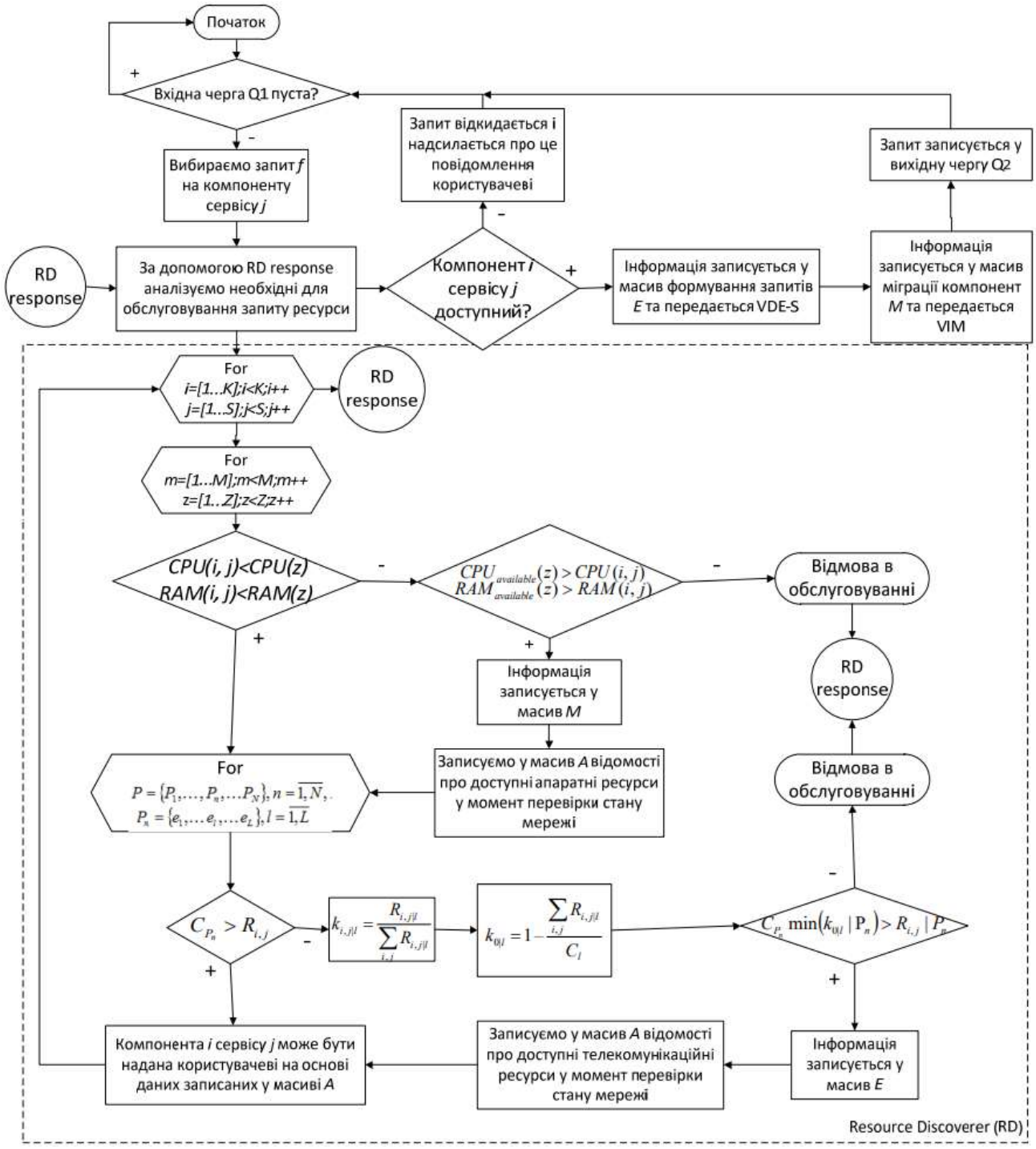


Рис. 2.8. Блок-схема алгоритму роботи методу балансування навантаження із забезпечення доступу фізичних ресурсів та з використанням функцій мережевої віртуалізації

Спочатку перевіряється вхідна черга запитів на компоненти сервісу. Якщо там міститься $f = \overline{1,F}$ запитів на ту чи іншу компоненту сервісу j, запускається аналізатор, який визначає, які ресурси необхідні для його обслуговування (таким чином формується таблиця значень CPU, RAM, вільної пропускної спроможності (C)), які забезпечать надання доступу до компоненту за час $t < t_{кр}$ і тип VM для кожного компонента сервісу одночасно перевіряється, компонент сервісу доступний. Кожен компонент сервісу використовує для своєї роботи апаратні потужності (CPU, RAM) фізичної та віртуальної машин, на яких він розташований. Resource discoverer перевіряє, z та VM має достатні апаратні ресурси, щоб надати користувачеві ще один екземпляр компонента. Якщо ресурсів достатньо, то перевіряємо, чи вистачить ресурсів каналу для того, щоб встановити з'єднання компонента з кінцевим користувачем.

$$C_{P_n} > R_{i,j} \tag{2.20}$$

де $R_{i,j}$ - швидкість інформаційного потоку, що генерує i-й компонент сервісу j. Усі параметри, які будуть доступні в момент перевірки стану мережі Resource discoverer, будуть записані в масив $\overline{A}$, формуватиме чергу на обслуговування. Туди вноситься пропускну здатність каналу, якої вистачить надання компонента. Resource discoverer повідомляє про це оркестратор і компонент переноситься у вихідну чергу запитів Q_2.

Менеджер телекомунікаційних ресурсів перевіряє стан каналів, і визначає оптимальний шлях передачі та обміну компонентами найменш завантаженими каналами. Оскільки в систему надходять запити на компоненти сервісів, які передаються різними маршрутами, то можна визначити частку заняття пропускної спроможності каналу кожним компонентом, щодо загальної кількості маршрутів, що проходять цим каналом.

Для початку визначимо ваговий коефіцієнт швидкості інформаційного потоку i - ого компонента j-ого сервісу щодо їх загальної кількості, трафік яких передається по l - ому каналу зв'язку:

$$k_{i,j|l} = \frac{R_{i,j|l}}{\sum_{i,j} R_{i,j|l}} \tag{2.21}$$

Тоді частка незайнятої пропускної спроможності l - ого каналу складе:

$$k_{0|l} = 1 - \frac{\sum_{i,j} R_{i,j|l}}{C_l} \tag{2.22}$$

де C_l - смуга пропуску l - ого каналу.

Значення частки заняття пропускної спроможності $\Pr_{i,j|l}$ виражається пропорційно $k_{i,l}$:

$$\Pr_{i,j|l} = k_{i,j|l} \cdot (1 - k_{0|l}) \cdot 100\% \tag{2.23}$$

Вважатимемо, що $\Pr_{i,j|l}$ відповідає пріоритету компоненти по відношенню до інших, трафік яких передається по l - му каналу зв'язку. Це слід розуміти таким чином, що завершення передачі цього компонента призведе до вивільнення найбільшої пропускної спроможності каналу, що, своєю чергою, дозволить обслуговувати чергу запитів з найбільшою ефективністю.

Вважатимемо, що певна смуга пропускання маршруту використовується неефективно, коли $\min\left(k_{0|l} \mid P_n\right)$ максимальне. Тоді, за допомогою VDE switeher manager'a та Migration manager здійснюється процес завантаження частини смуги, що не використовується, і відбувається її перерозподіл між компонентами з більшою потребою.

Якщо при оцінці вільних потужностей виявиться, що ресурсів для надання ще одного екземпляра компонента або наступного компонента недостатньо, тобто виконуються умови

$$CPU_{available}(z) > CPU(i,j) \tag{2.24}$$

$$RAM_{available}(z) > RAM(i,j) \tag{2.25}$$

$$C_{P_n} \min\left(k_{0|l} \mid P_n\right) > R_{i,j} \mid P_n \tag{2.26}$$

то здійснюється реплікація z-ї віртуальної машини на іншу фізичну машину PM_m. Інформація про необхідність міграції компонента надсилається оркестратору та

Migration manager, який здійснює переміщення перевантаженоїVM_zу співпраці з VDE switcher manager'ом. Якщо міграція неможлива, настане відмова обслуговування, і RD починає весь аналіз ресурсів спочатку.

Максимальне значення показника вільних віртуальних та телекомунікаційних ресурсів передаватиметься оркестратору, який у разі потреби здійснювати міграцію додатків на менш завантажені PM.

Висновки до 2-го розділу

1. Однією з основних причин погіршення якості обслуговування користувачів у хмарних мережах є затримка, що безпосередньо залежить від часу пошуку оптимального маршруту між компонентами сервісу. Перевантаження та міграція віртуальних машин призводять до змін у оптимальному маршруті передачі, що в свою чергу викликає подовження часу пошуку каналів та, отже, збільшення часу передачі та затримок. У даній роботі пропонується метод пошуку маршруту, що базується на критерії найменшого часу проходження. Цей метод використовує дані про поширення інформації та зміни в топології мережі для розрахунку оптимального маршруту передачі. Отримані дані є єдиною метрикою маршруту, яка враховує компроміс між властивостями трафіку та вибором оптимального маршруту, що важливо при обмеженій можливості одночасного існування потоків із максимальним сервісом на загальних лініях.

2. Оцінка стійкості у мережах центрів обробки даних вимагає забезпечення максимальної стійкості зв'язків між компонентами сервісу. Стійкість зв'язків у цьому контексті визначається як стійкість топологічних структур під час передачі або міграції компонентів, забезпечуючи фізичний і логічний маршрут при реплікації компонентів. Робота пропонує модель надання сервісу, що ґрунтується на методі пошуку маршруту з урахуванням стабільності структури віртуалізованого центру обробки даних. Ця модель вибирає стратегію, що базується на методі пошуку маршруту з урахуванням стійкості структури ЦОД та враховує інтенсивність запитів, що надходять до віртуальних машин.

3. Великі обсяги веб-сервісів та неефективний розподіл фізичних ресурсів

призводять до збільшення затримок у передачі. У даній роботі пропонується новий метод ефективного розподілу фізичних ресурсів, який підвищить ефективність балансування навантаження за допомогою інтегрованої архітектури управління та технології NVF. Метод полягає в оцінці максимального інтегрального показника доступних ресурсів фізичної машини та використанні його для міграції додатків на менш завантажені сервери. Якщо кількість запитів зростає, використання віртуального менеджера інфраструктури може створити програмний аналог фізичного мережевого пристрою за допомогою технології NFV. Цей метод дозволяє балансувати навантаження на глобальному та локальному рівнях, зменшуючи тривалість обслуговування та втрати запитів, і забезпечує оптимальне використання ресурсів.

3 МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ РОЗПОДІЛУ ІНФОРМАЦІЙНИХ ПОТОКІВ У ХМАРНИХ СЕРВІСАХ

У цьому розділі було виконано моделювання та дослідження розподілу інформаційних потоків у центрах обробки даних хмарної інфраструктури. Для цього була створена імітаційна модель формування структури центру обробки даних хмарної інфраструктури, використовуючи інструментарій Matlab. Ефективність використання методу пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних була досліджена на основі розробленої імітаційної моделі. Результати моделювання показали, що зі збільшенням стійкості структури час пошуку каналів для передачі запитів зменшується, що призводить до загального скорочення часу передачі сервісу в центр обробки даних хмарної інфраструктури.

Також було проведено моделювання інтегрованої системи керування, використовуючи функцію NVF. На основі отриманих залежностей була продемонстрована ефективність методу балансування навантаження, враховуючи доступність фізичних ресурсів, і використання функцій мережевої віртуалізації. Виявлено зниження затримки при наданні компонентів хмарного сервісу та здійснення контролю за ресурсами кожного сервера.

Крім того, було розроблено комплекс надання хмарних сервісів, який враховує управління оптичними ресурсами між центрами обробки даних хмарної інфраструктури. Це дозволяє моніторити та управляти завантаженістю каналів, оптимально розподіляти смугу пропускання для кожного логічного каналу одного фізичного каналу. Запропоновано використання математичної моделі надання сервісу, яка ґрунтується на оцінці стійкості структури, та інтегрованої архітектури системи управління ресурсами з використанням функції NVF. Це дозволяє підтвердити адекватність отриманих результатів. Запропоновано використання різних методів та алгоритмів у хмарних структурах як основи для забезпечення якості сервісу при розгортанні хмарних мереж.

3.1 Розробка імітаційної моделі структури ЦОД хмарної інфраструктури

Імітаційне моделювання процесу обслуговування інформаційних потоків завжди вимагало від розробника імітаційної моделі перевірки адекватності створеної моделі процесом, що відбувається у реальній системі масового обслуговування. Найпростіший спосіб визначити характеристики системи обслуговування полягає у отриманні експериментальних даних про процес обслуговування. Аналіз цих даних дозволяє визначити, які параметри системи обслуговування необхідно змінити у тому, щоб підвищити якість обслуговування, тобто оптимізувати процес [11, 20].

Сучасні системи масового обслуговування містять велику кількість компонентів, кожен з яких є складною системою, яка також має свої параметри та характеристики. Загалом, всі ці компоненти впливають на характеристики якості обслуговування системи. Тому для створення адекватної імітаційної моделі та адекватного оцінювання результатів моделювання необхідно врахувати всі компоненти, що беруть участь у процесі обслуговування.

Велика кількість абонентів, додатків та сесій, що генеруються цими додатками, та їх різноманітність значно впливають на характеристики трафіку, що надходить до системи обслуговування. Тому, щоб змоделювати такий трафік, необхідно застосувати потужний математичний апарат, який дозволив більш-менш точно описати характеристики такого трафіку. Зрозуміло, що найбільш ефективним способом моделювання у такій ситуації є розробка спеціального програмного забезпечення.

Завдяки програмній реалізації імітаційної моделі можна повністю реалізувати всі необхідні функції моделі, а й забезпечити контроль над її роботою. Програмне забезпечення дозволяє за допомогою графічного інтерфейсу користувача динамічно змінювати параметри моделі, тим самим оцінити поведінку моделюється в конкретній ситуації, яка може виникнути в реальній системі обслуговування. Крім того, програмне забезпечення за допомогою графічної оболонки дозволяє в реальному режимі часу оцінювати всі параметри моделі. Це

можливо здійснювати за допомогою графіків, діаграм, списків та таблиць, які оновлюються в реальному режимі часу.

Моделювання структур, що динамічно змінюються, вимагає від розробника імітаційної моделі великих зусиль. Необхідно врахувати всі вимоги та умови, за яких має змінюватися структура мережі. Важливим при цьому є вибір характеристик і параметрів трафіку, що передається за такою структурою.

Особливо непросто здійснювати моделювання структур хмарних мереж [19, 26]. Для розробки імітаційної моделі динамічно змінної структури мережі центру обробки даних, що є основою хмарної мережі, використано програмне середовище Matlab. Це середовище розробки програмного забезпечення має всі необхідні засоби для створення повноцінних, ефективних, багатопоточних динамічно змінних структур. За допомогою створення масивів можна згенерувати граф мережі, що складається з вузлів та ребер, який дозволить наочно відобразити структуру мережі.

Як вузли, в даній імітаційній моделі, виступатимуть віртуальні машини, а як ребер - логічні канали, які з'єднують їх між собою. Запити до віртуальних машин надсилаються за логнормальним законом розподілу (міжпакетний інтервал). Для спрощення приймемо, що інтенсивність надходження запитів на обслуговування від одного користувача в середині сегмента та його тривалість (4 зап/год, 4000 с.), одна віртуальна машина має лише один сервіс. Введено, що є два діючі стани кожного вузла (активний (1) або пасивний (0)).

Розробка імітаційної моделі відбувалася у кілька етапів:

• Створення динамічно змінної структури. У програмному середовищі Matlab рандомізовано створювалася матриця суміжності, яка вказувала на кількість вузлів та зв'язку між ними. Зв'язки між вузлами генерувалися випадково, проте їхня кількість не перевищувала 50% від максимальної кількості (максимальною кількістю зв'язків вважалося повнодоступне з'єднання вузлів між собою, тобто кожен з усіма). Після формування матриці проводилася оцінка ступеня вершини, коефіцієнта кластеризації та посередництва, а також визначалася довжина найкоротших шляхів та центральні вузли. Оцінка цих параметрів дозволить

визначити найбільш завантажені вузли, тобто віртуальні машини, якими проходить велика кількість запитів. У зв'язку з обмеженістю апаратних ресурсів комп'ютера, на якому проводилося моделювання, кількість вузлів дорівнювала 20;

• Створення генератора трафіку. Створювався масив запитів, які прямували до вузлів графа за логнормальним законом розподілу інтенсивністю 4 зап/год. Розмір запитів коливався від 1500 біт до 4500 байт, а тривалість обслуговування 4000 с.

• Оцінка взаємопов'язаності вузлів між собою. Задаємо, що кожне ребро має параметр ваговий коефіцієнт, що вказує на його максимальну пропускну здатність (від 1 – 10 Мбіт/с). Оскільки модель описана в п.2.2 працює на основі аналізу внутрішньої мережі центру обробки даних, відстань між вузлами (віртуальними машинами) не більше 25 м .

• Перевірка стабільності структури. Для кожної вершини визначаємо її завантаженість (на основі параметрів отриманих на першому та другому етапі моделювання та з використанням методу Нороса) та проводимо оцінку стійкості структури та затримки на надання сервісу. У разі зміни структури визначаємо загальний час передачі, приймаючи, що час комутації кожному вузлі трохи більше 0,03 мс.

3.2 Моделювання структури центру обробки даних хмарної інфраструктури та її вплив на параметри QoS

Для оцінки ефективності моделювання структури центру обробки даних необхідно випадковим чином сформувати зв'язки (згідно з умовами моделі мережі) між вузлами та оцінити ймовірність взаємопов'язаності спочатку довільних пар вузлів між собою, а потім загальний взаємозв'язок мережі, який визначатиметься як сума зв'язаності всіх пар вузлів [6, 18]. Оцінка завантаженості кожної віртуальної машини, протягом часу T, буде проводитися з використанням методу Нороса, з урахуванням, що кожна VM, згідно з теорією систем масового обслуговування, буде системою типу G\G\1. Для спрощення вважається, що коефіцієнт варіації

інтервалів між запитами та тривалості їх обслуговування постійними величинами.

У нашій моделі ми оцінюємо стійкість структури з погляду параметра ймовірності того, що між двома сегментами в наступний момент часу існуватиме з'єднання, тобто існувати хоча б одне ребро, яке буде «ключовою ланкою» для з'єднання цих сегментів [11, 16]. Однак така ймовірність існування ребра (з'єднання) безпосередньо залежатиме від ймовірності відмови певного шляху всередині сегмента (тобто, між віртуальними машинами існувати хоча б один маршрут), а також від ймовірності блокувань всередині кожного сегмента.

Після формування всіх необхідних умов за допомогою аналітичних та статистичних методів оцінки ймовірності встановлення з'єднання та його надійності оцінюємо необхідний нам параметр.

Нехай зафіксуємо в момент часу T деякий стан мережі (кількість віртуальних машин тощо). Розглянемо зафіксований стан сегмента A (рис 3.1), у якому на даний час T знаходиться 20 вузлів.

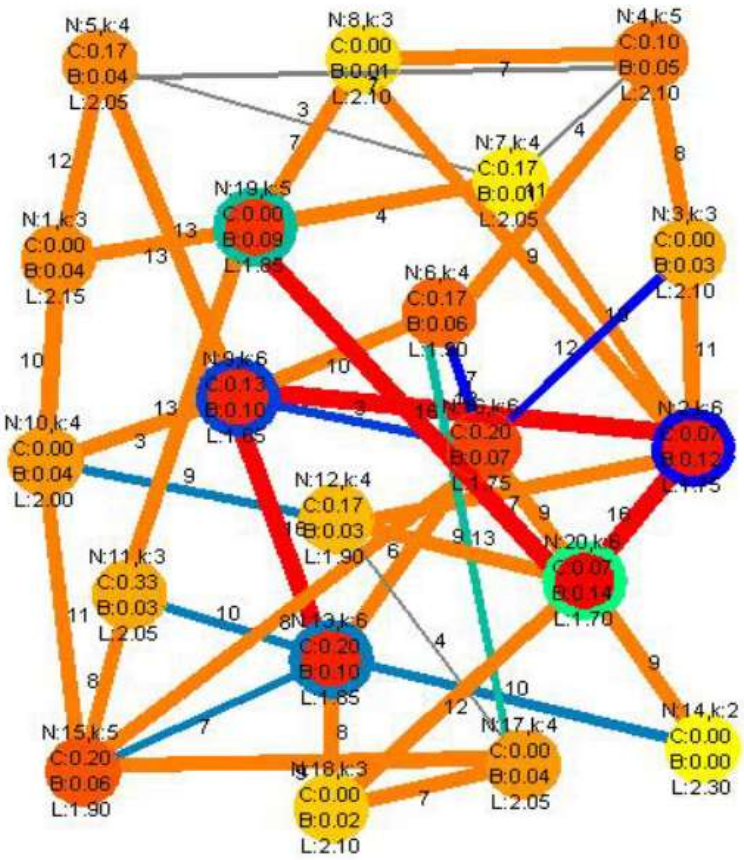


Рис. 3.1. Структура змодельованої мережі центру обробки даних

Задана кількість вузлів незначна, що пов'язано з такими обмеженнями: складність розрахунків щодо збільшення кількості вузлів зростає, а також ПК, на якому проводилося моделювання, має невисоку продуктивність. Червоним кольором показані найбільш завантажені ребра, а в середині кожної вершини – її параметри (ступеня вершини, коефіцієнт посередництва тощо).

K=

0	1,4	8,2	0	4,1	5,2	1,2	3,0	3,4	6,2	5,1	8,5	3,5	9,8	9,9	0	4,7	9,7	7,8	7,5
4,4	0	9,5	0	5,7	0	5,4	7,1	9,3	6,7	4,0	9,9	5,6	7,4	8,1	8,0	1,4	0	5,0	4,8
6,6	1,4	0	0	4,4	0,7	0	0	9,3	3,8	0	5,0	5,9	0	2,6	5,5	5,5	6,8	8,6	8,0
0	0	0	0	0	6,4	0	8,7	0,4	0,3	0	9,0	3,2	0,2	7,2	2,4	0	0,5	0,6	7,6
7,9	9,2	8,0	0	0	0	6,8	0	4,5	0	8,1	0,7	6,3	0	0	7,3	3,5	0	6,4	6,2
1,5	0	6,5	8,9	0	0	0	0	0	0	0	7,7	6,9	2,0	0	9,6	0	1,5	3,1	0
0,4	0,3	0	0	6,4	0	0	8,5	5,2	0	1,8	4,7	5,6	0	2,4	3,4	0	8,1	8,8	9,9
4,0	5,7	0	2,3	0	0	1,5	0	8,1	0	0,3	7,9	0	9,6	5,0	9,5	5,1	2,8	8,0	0
0,6	8,5	7,7	5,9	3,4	0	5,5	8,5	0	2,2	0	2,9	0	1,7	9,3	1,3	0	5,4	0	2,8
9,5	7,3	2,7	7,9	0	0	0	0	0,3	0	0	6,0	1,9	5,1	7,5	0	7,0	0	0	7,0
5,2	5,3	0	0	3,3	0	6,1	1,7	0	0	0	6,4	0	2,8	7,4	0,8	0	0	6,4	0
5,8	6,5	5,8	5,5	2,2	1,5	6,7	5,3	9,9	2,0	9,3	0	8,6	0	0	2,4	0,7	0	2,5	0
1,1	2,3	0,2	2,2	4,1	6,7	5,8	0	0	0,8	0	0,5	0	3,2	1,5	0	2,2	2,3	2,6	1,4
4,8	3,3	0	2,5	0	5,2	0	1,7	2,1	1,9	9,3	0	3,7	0	0	4,8	4,4	0	9,3	4,2
6,7	7,4	8,1	0,6	0	0	6,5	5,0	4,3	4,4	8,3	0	1,7	0	0	7,3	1,2	2,6	4,6	1,4
0	6,8	2,8	8,5	0,4	9,2	7,1	0,7	8,4	0	5,5	4,4	0	3,4	7,7	0	5,9	0	0	8,0
3,5	2,1	7,2	0	7,7	0	0	3,0	0	9,3	0	8,4	0	2,6	9,8	3,0	0	8,5	1,0	0,4
0,1	0	8,3	1,2	0	3,2	8,6	9,0	9,8	0	0	0	0,3	0	7,4	0	4,2	0	5,8	0
7,0	9,1	2,9	9,0	7,0	7,1	9,0	5,3	0	0	5,0	7,7	9,2	1,3	6,1	0	2,8	0,6	0	0
7,4	1,4	7,4	0,6	5,3	0	1,4	0	2,6	3,9	0	0	2,5	7,9	6,3	1,5	2,7	0	0	0

V=

0	1,0	5,7	0	19,7	18,0	2,5	5,4	16,9	10,9	5,1	6,3	15,5	5,7	4,0	0	16,4	10,8	15,1	4,0
0,1	0	9,3	0	12,9	0	5,3	17,1	1,7	23,7	2,3	7,5	15,2	16,7	1,9	13,2	23,8	0	21,9	4,6
6,5	13,5	0	0	13,8	6,8	0	0	12,2	24,0	0	4,8	5,6	0	23,5	1,3	10,5	18,0	11,5	10,9
0	0	0	0	0	19,4	0	20,4	15,7	22,4	0	23,8	18,4	18,5	21,4	23,4	0	13,2	1,6	23,0
17,9	7,6	14,0	0	0	0	7,3	0	6,2	0	20,2	5,9	17,7	0	0	5,9	11,7	0	4,7	23,3
8,8	0	0,4	15,3	0	0	0	0	0	0	0	19,6	20	25,0	0	23,8	0	6,9	12,6	0
17,1	17,6	0	0	10,2	0	0	20,6	6,6	0	0,7	12,4	13,0	0	13,2	14,5	0	24,0	21,9	17,3
11,5	17,7	0	10	0	0	22,2	0	4,4	0	21,5	7,1	0	4,4	24,7	6,5	3,8	24,9	16,1	0
9,6	12,1	22,5	24,5	14,5	0	13,7	13,0	0	23,8	0	4,9	0	4,5	2,0	1,9	0	13,3	0	23,3
11,3	10,9	10,9	10,1	0	0	0	0	9,1	0	0	10,7	20,8	15,7	0,2	0	22,3	0	0	23,6
7,1	16,7	0	0	0,6	0	10,3	7,7	0	0	0	19,7	0	0	6,7	23,9	0	0	10,2	0
10,5	18,6	1,8	0,1	11,9	6,8	13,9	19,2	4,2	6,7	12,7	0	12,7	0	0	20,1	4,9	0	13,1	0
19,6	22,8	9,7	9,0	6,3	14,4	24,3	0	0	18,0	0	7,9	0	15,1	1,1	0	0,8	23,6	12,8	18,5
22,0	23,4	0	13,4	0	5,3	0	5,4	13,8	10,4	23,5	0	15,6	0	0	7,9	10,1	0	9,5	19,8
11,4	21,7	13,4	13,2	0	0	20,5	4,5	7,8	12,5	3,1	0	4,4	0	0	15,6	24,6	7,0	3,0	14,5
0	0,3	7,0	22,6	9,1	20,9	21,8	19,1	21,9	0	18,1	19,5	0	13,7	18,0	0	17,0	0	0	10,7
2,2	23,0	10,8	0	14,3	0	0	24,4	0	24,4	0	11,4	4,3	0,6	3,1	11,4	0	18,1	9,6	1,3
19,3	0	7,8	7,0	0	23,3	21,2	16,7	22,4	0	0	0	15,3	0	19,9	0	4,0	0	4,2	0
18,3	3,8	20,4	8,2	11,0	21,5	18,5	22,1	0	0	17,0	7,5	12,6	15,4	1,8	0	8,6	8,1	0	0
9,3	0	23,3	8,4	21,4	0	20	0	15,1	15,8	0	0	21,1	2,3	12,4	18,0	10,6	0	0	0

Для оцінки зв'язності вузлів необхідно знайти безліч незалежних маршрутів, які б задовольняли критерії мінімальної близькості та максимального вагового коефіцієнта. Для цього потрібно визначити ймовірність зв'язаності між двома випадково обраними вузлами A і B, причому кількість обраних пар набагато більша за кількість вузлів надалі дозволить судити про взаємозв'язок в цілому.

Приймемо, що максимальна відстань між вузлами сегмента не перевищує 25 м, а максимальний коефіцієнт ребра не більше 10 Мбіт. Генеруємо матриці

відстаней між вузлами (V) та вагових коефіцієнтів (K).

Вважаємо, що кількість ребер між вузлами становить не менше 70% кількості ребер повнозв'язкової структури із заданою кількістю вузлів. Така кількість ребер задано для того, щоб сегмент характеризувався високою зв'язністю та в процесі генерації графа до кожного з вузлів можна було доступно через кілька шляхів.

Після генерування потрібних нам матриць та їх аналізу знаходимо всі можливі маршрути. Вони можуть складатися з одного ребра (це означає, що A і з'єднані безпосередньо) або здійснювати передачу через транзитні вузли (це означає, що між A і B є додаткова вершина C). Якщо існує безпосереднє з'єднання між A і B, то ймовірність того, що в такій системі дані два випадкові вузли будуть з'єднані залежить від характеристик ребра.

При кожному запуску моделі всі матриці, а відповідно і всі результати прийматимуть різні значення, оскільки з'єднання між вузлами генеруються випадковим чином.

Оптимальних маршрутів може бути кілька, тому вибираємо тільки той маршрут, де сумарне значення вагових коефіцієнтів ребер і відстаней між вузлами для одного маршруту буде більше суми вагових коефіцієнтів ребер і відстаней між вузлами іншого маршруту, можна визначити як:

$$\sum\left(\max\left[k_{1.1}+k_{1.2}\right],\min\left[S_{1.1}^{2}+S_{1.2}^{2}\right]\right)>\sum\left(\max\left[k_{2.1}+k_{2.2}\right],\min\left[S_{2.1}^{2}+S_{2.2}^{2}\right]\right) \qquad (3.1)$$

За такою умовою порівнюємо маршрути між собою та вибираємо найкращий маршрут для з'єднання. За такої кількості вузлів ймовірність того, що маршрут між довільною парою вузлів лежить через два ребра дуже велика. З цього випливає, що від кількості вузлів у сегменті (а їх у нашому випадку небагато) залежить, наскільки буде складним маршрут (через скільки проміжних вузлів він проходитиме). Відповідно, збільшення кількості ребер на маршруті зменшить ймовірність зв'язаності між собою вузлів. Для цієї моделі можливість того випадкові два вузли не взаємопов'язані між собою досить мала. За такою моделлю можна оцінити зв'язність довільної пари вузлів і говорити про чи зв'язність вузлів в цілому.

Для даної моделі можливість незв'язності вузлів оцінюється:

Таблиця 3.2.

Імовірність незв'язності вузлів мережі

1,159 E-06	2,26E-06	1,05E-10	2,71E-11	5,42E-11
2,48E-13	4,06E-19	4,93E-11	1,02E-13	3,13E-18
1,58E-08	1,01E-14	6,13E-26	1,81E-13	2,92E-11
6,54E-11	1,34E-06	3,34E-15	3,59E-11	4,27E-09

Звідси випливає, що у такому маленькому сегменті майже всі вузли взаємопов'язані між собою. Однак із зростанням кількості вузлів не взаємопов'язаність вузлів зростає:

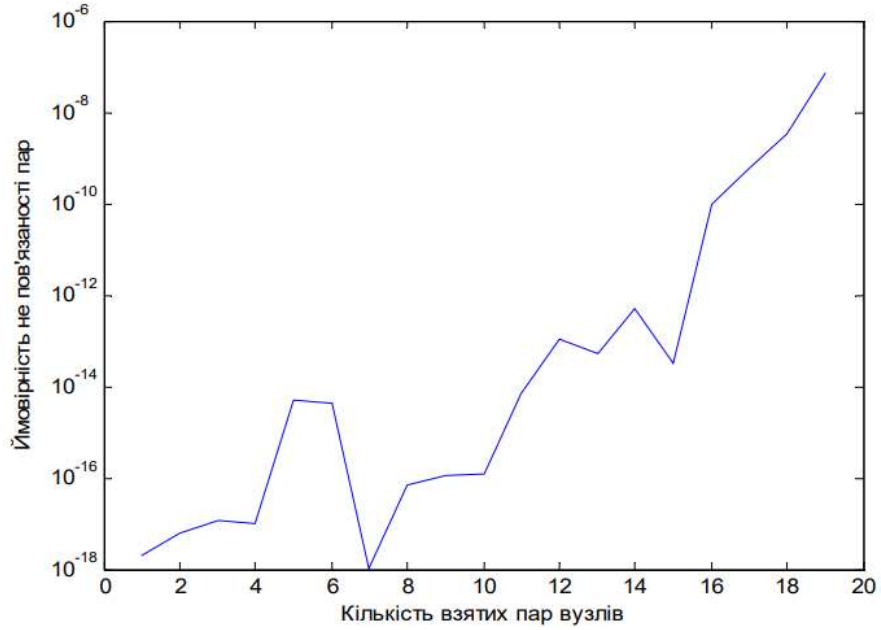


Рис. 3.2. Залежність ймовірності незв'язності вузлів від кількості

З наведеного графіка та таблиці значень можна простежити, що чим більше вузлів, тим ймовірність незв'язності зростає. Це свідчить у тому, що у великих сегментах зв'язаність між собою вузлів зменшується.

Мережа стає більш розрідженою, а для оцінки маршруту критерієм мінімальної близькості потрібно більше часу.

3.3 Дослідження ефективності застосування методу пошуку маршруту з урахуванням стійкості структури віртуалізованого ЦОД на основі розробленої імітаційної моделі

Потік трафіку між центрами обробки даних підпорядковується законам самоподібності і надходить обслуговування з нерівномірними інтервалами часу надходження. Виникає проблема ефективного розподілу ресурсів оптичного тракту відповідно до вимог трафіку. З одного боку, якщо ресурси каналу розподіляються відповідно до пікової швидкості, це призведе до нерівномірному розподілу смуги пропускання, оскільки її частина використовуватиметься передачі малогабаритних потоків і, у разі надходження високо пріоритетного трафіку, зможе бути звільненою. З іншого боку, не можна забезпечити відповідний рівень параметрів QoS, якщо ресурси пропускної спроможності розподіляються відповідно до середньої швидкості надходження пакетів. Тим більше, необхідно буде здійснювати прогнозування інтервалів надходження запитів та, відповідно, їх обслуговування до центру обробки даних, що не дозволить повною мірою забезпечити динамічну гнучкість та роботу такої системи. У такому випадку необхідно проводити агрегацію потоків трафіку перед його надходженням до буфера центру обробки даних, здійснювати перерахунок маршруту для кожного з агрегованих пакетів і, при необхідності, перерозподіляти ресурси каналів зв'язку відповідно до вимог трафіку. Як наслідок, це призведе до ускладнення процесів маршрутизації та виділення спектру, а також збільшить затримку на надання сервісу з кінця до кінця.

У хмарній мережі кожен сервіс - це набір атомарних компонентів, які потребують достатньо вільної обчислювальної потужності та пропускної спроможності каналів. У таких мережах процес міграції окремого компонента потребує додаткових мережних ресурсів. Фізично хмарної інфраструктури застосовують оптоволоконні системи зі спектральним ущільненням [10, 21].

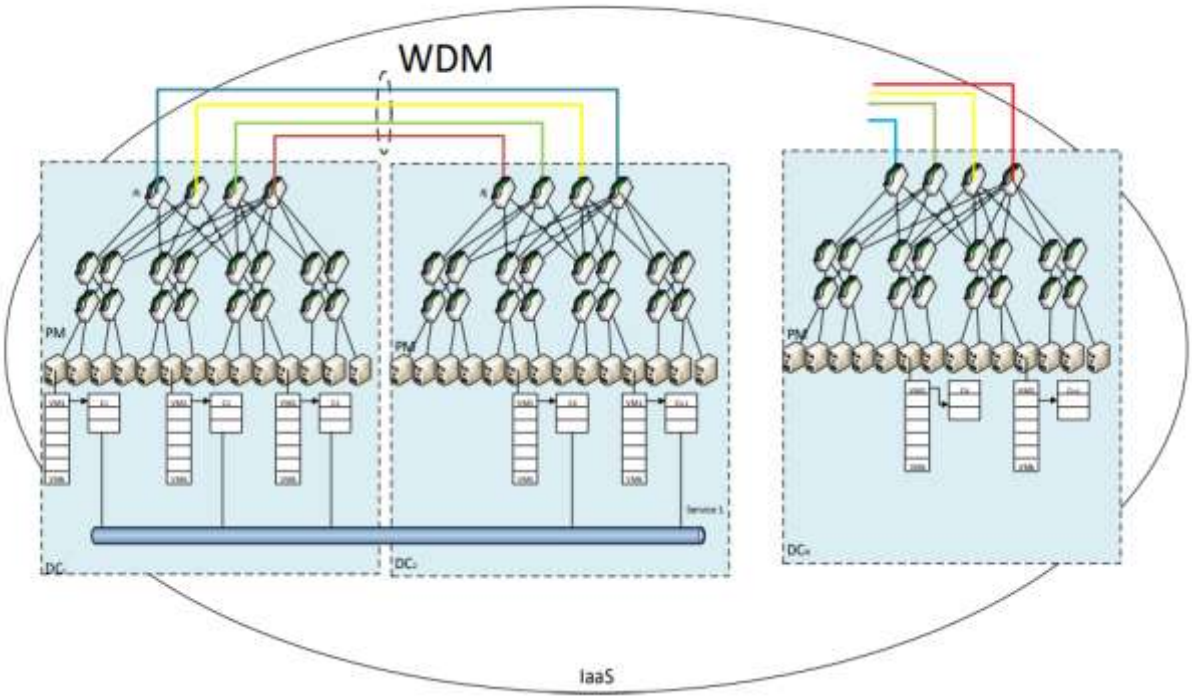


Рис. 3.3. Архітектура хмарної мережі

Сьогодні існує велика кількість програмних засобів, що дозволяють проводити моделювання як окремих мережевих пристроїв, так і мережі в цілому. Великі проблеми викликають саме моделювання таких хмарних систем. Основними засобами моделювання, які використовуються вченими у всьому світі для тестування їх гіпотез та розробок, є: NS (Network Simulator), OPNET, CloudSim, OmNET++. Всі ці засоби дозволяють досліджувати параметри функціонування мережевих вузлів, систем, протоколів і дозволяють впроваджувати власні зміни у конфігурацію моделі того чи іншого пристрою, що дозволяє провести дослідження власних розроблених вченими алгоритмів чи протоколів. Їхньою перевагою є те, що необов'язково будувати чи орендувати цілі дата-центри для дослідження, а можна використовувати програмні аналоги серверів, маршрутизаторів або комутаторів, розгортаючи на них необхідні програми або сервіси досліджуючи мережеві характеристики та параметри. Проте ці кошти базуються на принципі моделювання дискретних подій. Істотним недоліком цих засобів є те, що вони використовують статистичні методи для розрахунку стану системи у певний момент часу. Таким чином, час роботи реальної мережі може моделюватися

протягом декількох секунд, що не дозволяє простежити та адекватно оцінити зміни тих чи інших параметрів мережі. Особливо це простежується під час моделювання процесів, що працюють у реальному масштабі часу. Наприклад, моделювання топологічної зміни структури мережі, яка вимагає аналізу стійкості та надійності роботи мережевих компонентів, що входять до її складу, формування та обслуговування черг пакетів у маршрутизаторі та, відповідно, оцінка затримки при передачі з кінця в кінець для такої мережі.

Було створено дві підмережі з фізичних машин (серверів), які локально об'єднані у два різні центри обробки даних. В основі роботи кожного окремо взятого сервера покладено модель розгортання віртуальних машин на серверах та надання сервісу. Для тестування ефективності запропонованих методів та алгоритмів запропоновано архітектуру побудованої мережі, представлену на рисунку 3.4.

У зв'язку з неможливістю дослідження транспортної мережі НТЦМТ розроблено програмний аналог, який за функціональністю відповідає реальній WDM системі.

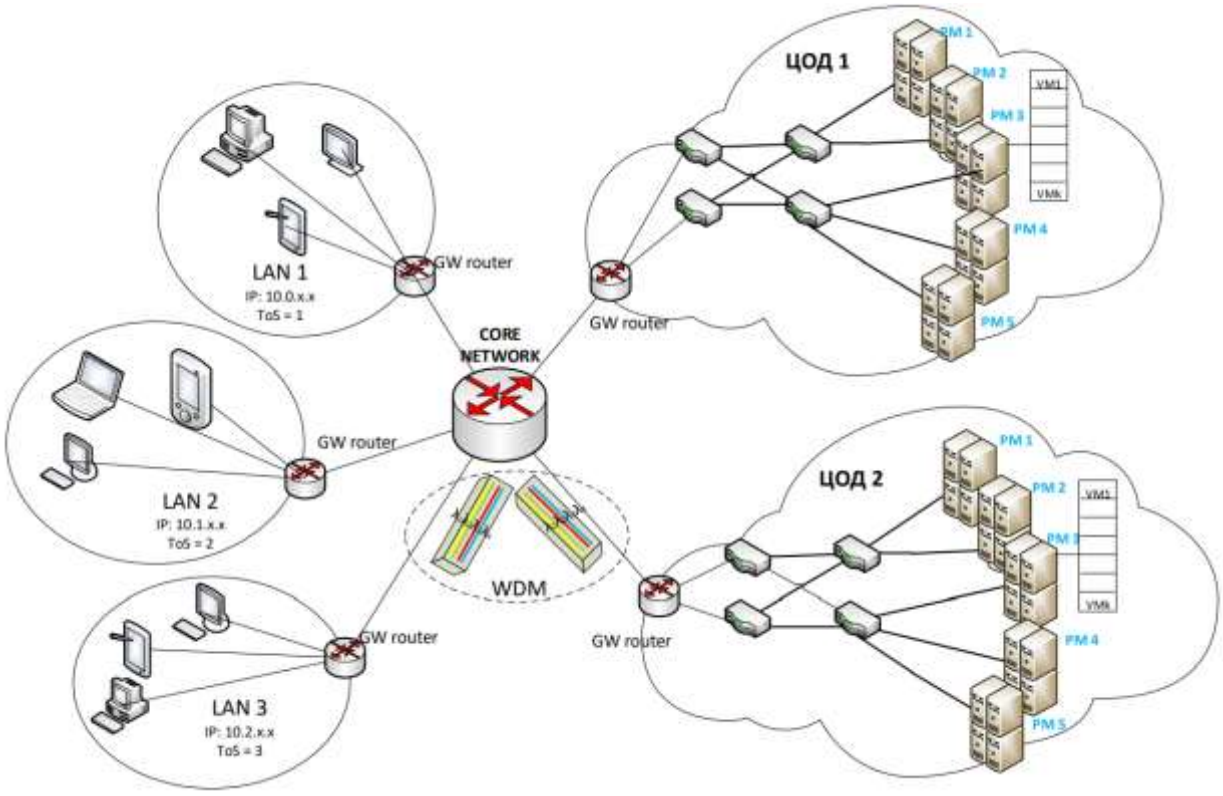


Рис. 3.4. Тестове середовище для надання хмарних сервісів з використанням

розподілених дата-центрів хмарної інфраструктури

У мережі згенеровано набір сервісів, які розгортаються на створеній хмарній інфраструктурі центрів обробки даних, та функціонують паралельно. Параметри цих сервісів представлені у таблиці 3.3, які розгортання на розробленої інфраструктурі на рисунку 3.5.

Таблиця 3.3.

Параметри розгорнутих на розробленій інфраструктурі сервісів

Номер сервісу	Назва хмарного сервісу	Кількість компонентів сервісу	Максимальний час обробки запиту на компонент сервісу
Service 1	Docs online	2	100 мс
Service 2	Video streaming	2	400 мс
Service 3	Data Base	2	1 мс
Service 4	Mail	2	400 мс
Service 5	Computing	3	2 мс

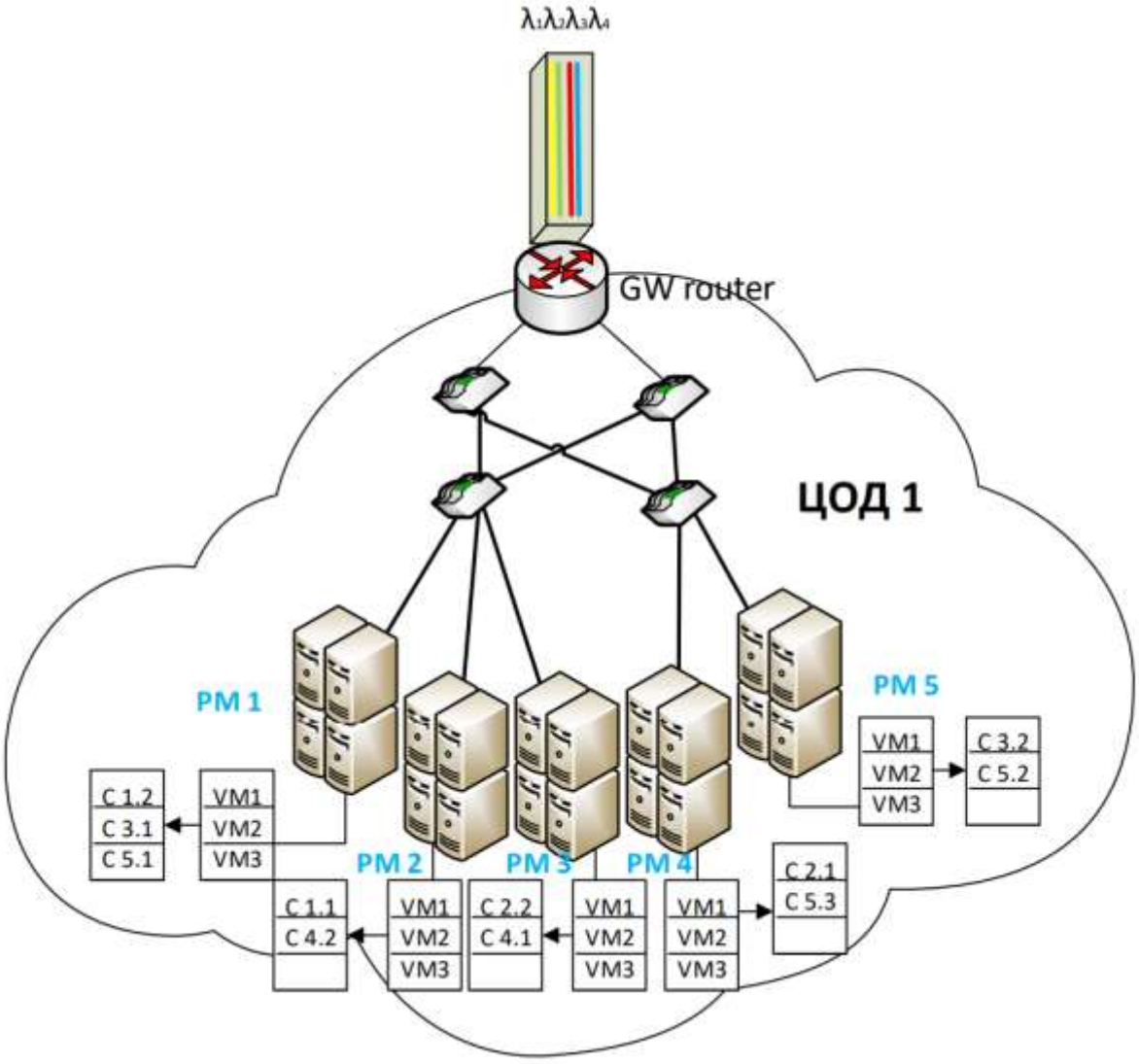


Рис. 3.5. Принцип розгортання компонентів сервісу на ЦОД 1

Кожен компонент сервісу встановлювався окрему віртуальну машину. Тривалість існування віртуальних машин не обмежена. Для перевірки і тестування

роботи системи всі компоненти сервісів на кожному фізичному сервері запущені одночасно, а запити, які надходять на кожну з них і гіпервізора серверів, що маршрутизуються. Загальний контроль та управління такою системою здійснює оркестратор відповідно до архітектури, запропонованої у другому розділі. Продуктивність конкретної віртуальної машини розраховується як відношення одиниці часу до тривалості обслуговування запиту цією машиною на загальну кількість всіх віртуальних машин, встановлених на сервері. Структурна схема розробленого комплексу з використанням засобів UML відображена на рисунку 3.6.

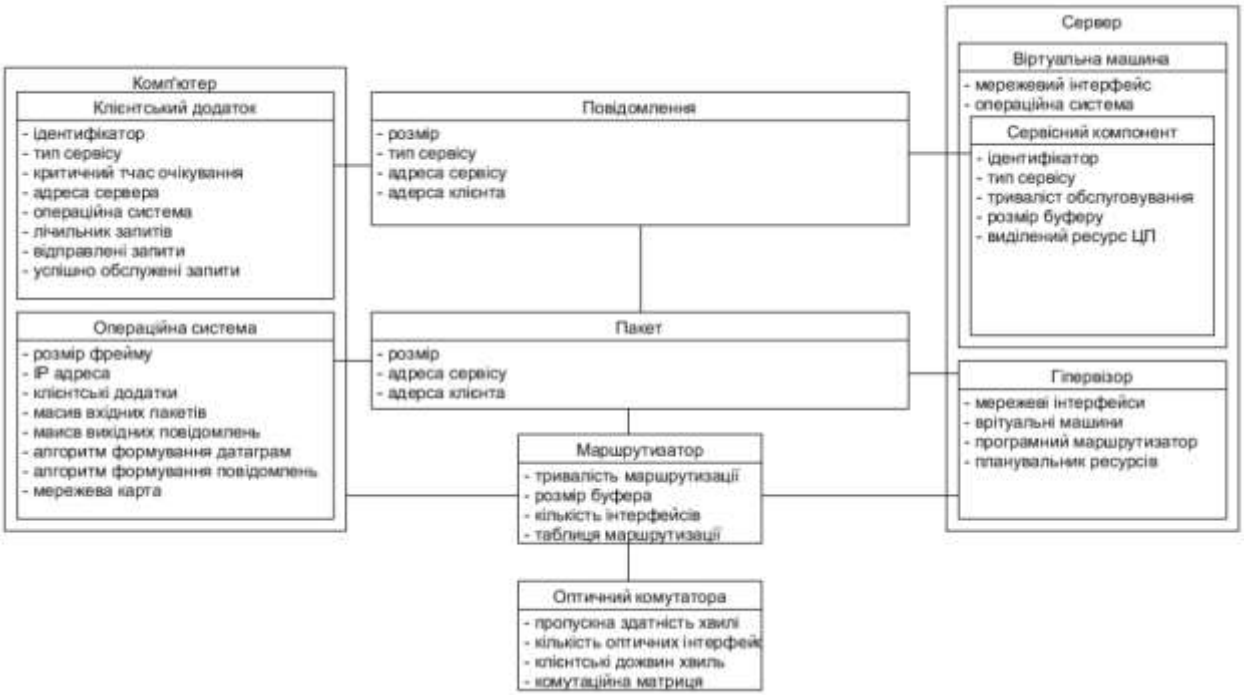


Рис. 3.6. Структурна схема розробленого комплексу

3.4 Імітаційне моделювання інтегрованої системи керування з використанням функції NVF

Для дослідження ефективності запропонованих у другому розділі рішень було розроблено імітаційну модель хмарної інфраструктури. Імітаційне моделювання процесу обслуговування завжди вимагало від розробника імітаційної моделі перевірки адекватності створеної моделі процесом, що відбувається у реальній системі масового обслуговування. Найпростіший спосіб визначити

характеристики системи обслуговування полягає у отриманні експериментальних даних про процес обслуговування. Аналіз цих даних дозволяє визначити, які параметри системи обслуговування необхідно змінити у тому, щоб підвищити якість обслуговування, тобто оптимізувати процес [12, 19].

Сучасні системи масового обслуговування містять велику кількість компонентів, кожен з яких є складною системою, яка також має свої параметри та характеристики. Загалом, всі ці компоненти впливають на характеристики якості обслуговування системи. Тому для створення адекватної імітаційної моделі та адекватного оцінювання результатів моделювання необхідно врахувати всі компоненти, що беруть участь у процесі обслуговування.

Велика кількість абонентів, додатків та сесій, що генеруються цими додатками, та їх різноманітність значно впливають на характеристики трафіку, що надходить до системи обслуговування. Тому, щоб змоделювати такий трафік, необхідно застосувати потужний математичний апарат, який дозволив більш-менш точно описати характеристики такого трафіку. Зрозуміло, що найбільш ефективним способом моделювання у такій ситуації є розробка спеціального програмного забезпечення.

Завдяки програмній реалізації імітаційної моделі можна повністю реалізувати всі необхідні функції моделі, а й забезпечити контроль над її роботою. Програмне забезпечення дозволяє за допомогою графічного інтерфейсу користувача динамічно змінювати параметри моделі, тим самим оцінити поведінку моделюється в конкретній ситуації, яка може виникнути в реальній системі обслуговування. Крім того, програмне забезпечення за допомогою графічної оболонки дозволяє в реальному режимі часу оцінювати всі параметри моделі. Це можливо здійснювати за допомогою графіків, діаграм, списків та таблиць, які оновлюються в реальному режимі часу.

Більшість симуляторів телекомунікаційних систем та мереж операціонують за дискретною моделлю подій, яка не завжди повністю відтворює усі особливості процесів, що відбуваються в мережі [17, 21]. Основоположенням для розробки наведеної архітектури є модель розгортання віртуальних машин на фізичних

серверах та надання хмарних сервісів. Структурна схема розробленої моделі, яка була створена з використанням інструментів UML, зображена на рисунку 3.7.

Концептуальна модель надання хмарного сервісу користувачу з розгортанням інфраструктури та запропонованою системою управління та методом балансування відображена на рисунку 3.8.

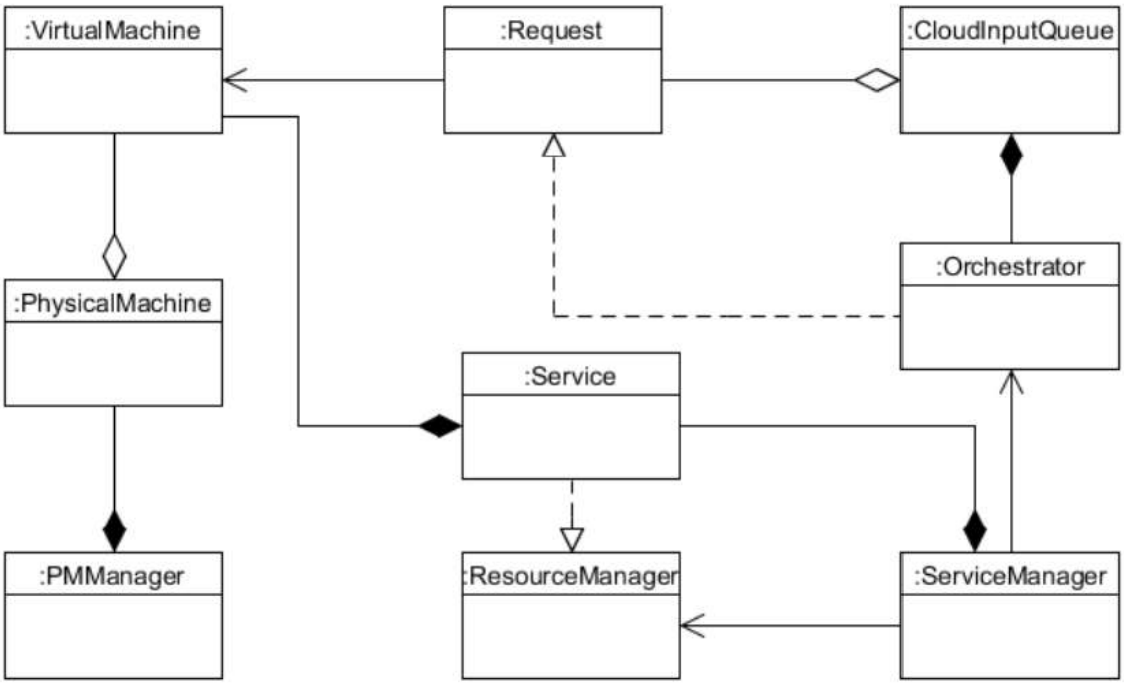


Рис. 3.7. Структурна схема імітаційної моделі

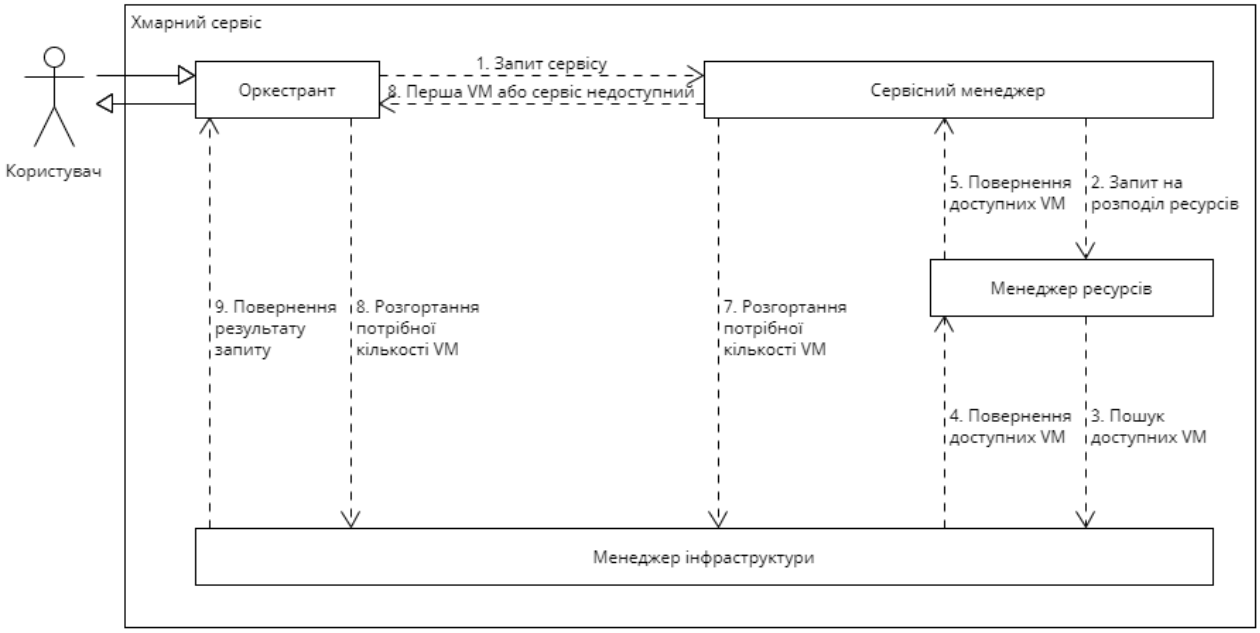


Рис. 3.8. Модель надання сервісу

Для дослідження ефективності та адекватності роботи розробленого програмно-апаратного комплексу відповідно до процесів реальної хмарної мережі необхідно провести його тестування. Тестування системи здійснювалося у три етапи.

На першому етапі виконувався аналіз роботи мережі, при її функціонуванні згідно з розробленою архітектурою хмарної мережі. На цьому етапі аналізується використання фізичних ресурсів серверів, проводився порівняльний аналіз їх завантаженості, затримки проходження пакетів з кінця в кінець, кількість опрацьованих та необроблених запитів. Особлива увага приділялася сервісам із найгіршою якістю надання, тобто з найбільшою кількістю не опрацьованих запитів.

На другому етапі проводився моніторинг інфраструктури системи та тривалості обслуговування запитів. Використання інтегрованої архітектури управління дозволило контролювати доступні апаратні та програмні ресурси.

На третьому етапі запускалися всі запропоновані вище методи для сервісів, в яких тривалість обслуговування запитів є найбільшою. Досліджувалась якість надання хмарних сервісів для кінцевих користувачів. Під якістю надання сервісу розуміється збільшення затримки та часу обробки запитів компоненти сервісу.

Тестування проводилося без застосування запропонованих рішень та з їх застосуванням. Інтенсивність надходження запитів на сервер лише з одним розгорнутим сервісом та однією віртуальною машиною наведено на рисунку 3.9.

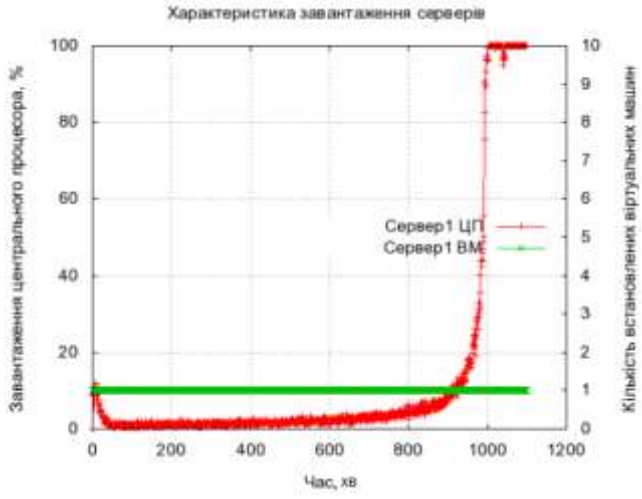


Рис. 3.9. Характеристика завантаження першого сервера

На рисунку 3.10 відображено інтенсивність надходження запитів на надання сервісів ЦОД 1 для користувачів лише однієї LAN протягом 20 годин роботи системи.

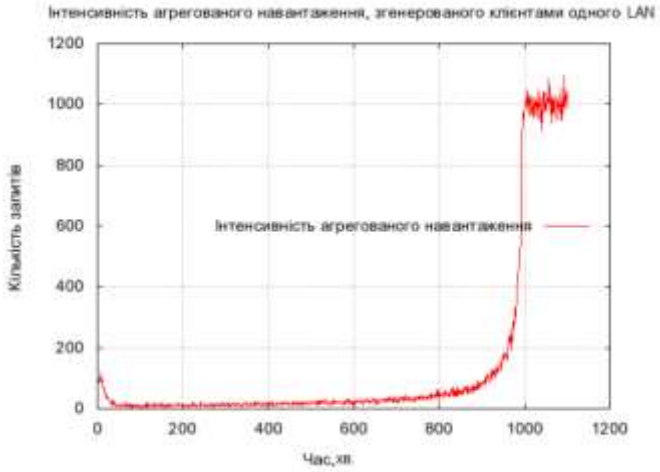


Рис. 3.10. Інтенсивність надходження запитів на надання послуг ЦОД 1

Як видно із рисунків 3.9 - 3.10 завантаженість сервера зростає за статечним законом розподілу, особливо в той час, коли користувачі перебувають у гуртожитках та намагаються отримати доступ до хмарних сервісів. Очевидно, що високий рівень завантаженості негативно впливає на кінцеву якість сприйняття послуги, оскільки збільшується час обробки запитів на надання цього сервісу (в даному випадку простежується різке збільшення часу відгуку від 0,05 до 0,35 с (рисунок 3.11)), за рахунок переповнення буферів доступних віртуальних машин.

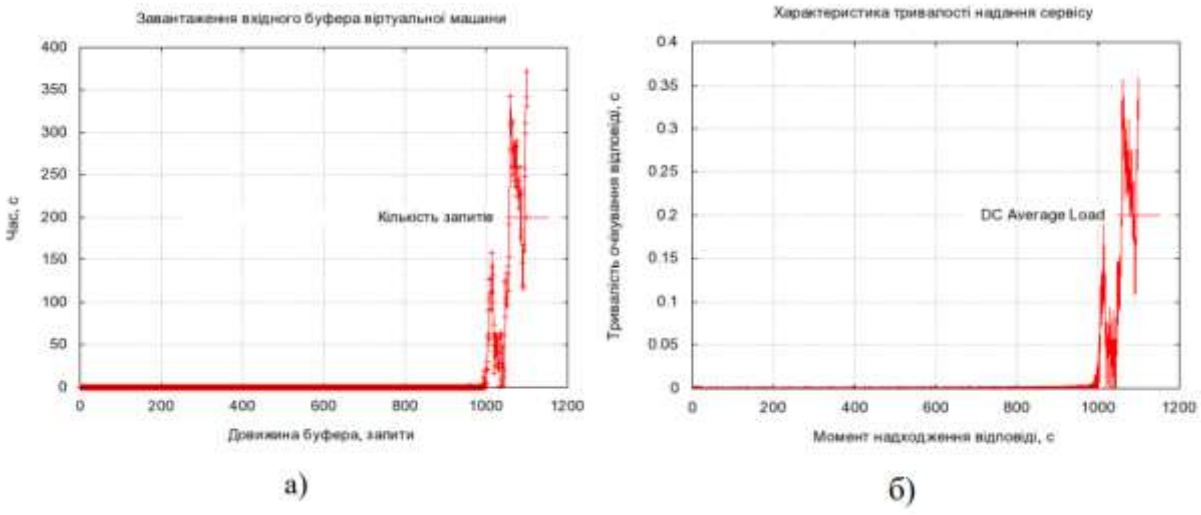


Рис. 3.11. Характеристика тривалості надання сервісу а) та завантаженістю вхідного буфера VM; б) та моменту надходження відповіді

Після розгортання всіх компонентів сервісів на серверах ЦОД 1 проводився аналіз завантаженості всіх серверів, залежно від інтенсивності запитів на конкретні компоненти сервісів (рисунок А.1 додатка А).

Дослідження показали, що віртуальні машини, на яких розміщені компоненти сервісу Computing, перевантажені, та у разі надходження додаткової кількості запитів необхідно здійснювати їхню міграцію (рисунок А.2 г додатка А).

Як наслідок, збільшується тривалість обслуговування запиту та його надання кінцевому користувачеві. З рисунка А.3 додатка А видно, що, якщо першого сервісу затримка надання сервісу у середньому становить близько 0,1 з. (рисунок А.3а), для другого – 0,07с. (рисунок А.3б), для третього – 0,09с. (рисунок А.3в), для четвертого – 0,06 с. (рисунок А.3г), за той самий час спостереження, то для п'ятого сервісу (рисунок А.3д) затримка на його надання вже після 100 хв. доступності сервісу, перевищує максимально допустимий рівень QoS (згідно з рекомендаціями ITU-T) і продовжує зростати за логарифмічним законом.

Враховуючи таке суттєве погіршення якості надання сервісу, згідно з методами, описаними в другому розділі, відбувається реплікація компонентів п'ятого сервісу на сервери з незадіяними віртуальними ресурсами. В результаті роботи розробленого програмно-апаратного комплексу підтверджується необхідність ефективного балансування навантаження з урахуванням доступності компонентів та стійкості структури мережі ЦОД та зменшення тривалості обробки запитів на надання сервісу кінцевим користувачам та, як наслідок, підвищення якості надання послуг.

3.5 Оцінка ефективності методу балансування навантаження на основі аналізу доступних компонентів хмарного сервісу

У мережі вручну створено набір сервісів, які розгортаються на створеній інфраструктурі. Інфраструктура системи представлена у вигляді матриці: по горизонталі – кількість фізичних серверів, по вертикалі – віртуальні машини, розгорнуті на кожному сервері. Параметри цих сервісів представлені у таблиці 3.4,

які розгортання на розробленої інфраструктурі на рисунку 3.12.

Таблиця 3.4.

Параметри сервісів

Назва хмарного сервісу	Колір	Вимоги до обчислювальних ресурсів	Адреси атомарних сервісів
1	Синій	{59, 59, 59}	Instance 1 {1001, 2001, 3001}
2	Чорний	{20, 20, 20}	Instance 2 {1002, 1003, 2002}
3	Фіолетовий	{20, 20, 20}	Instance 3 {2003 3002, 3003}
4	Блакитний	{20, 20, 20}	Instance 4 {4001, 4002, 4003}
5	Зелений	{20, 20, 20}	Instance 5 {4004, 5001, 5002}
6	Бузковий	{20, 20, 20}	Instance 6 {5003, 5004, 6001}

Рис. 3.12. Інфраструктура хмарної системи

Для генерації трафіку використовуються генератори, що базуються на логнормальному (щодо інтервалу надходження запитів) та експоненціальному (щодо інтенсивності надходження) законах розподілу. Це поєднання дозволяє отримати мультисервісний трафік з характеристиками, що наближаються до трафіку реальної хмарної мережі.

Усі сервіси та генератори трафіку для кожного типу сервісу активовані одночасно. Тривалість існування віртуальних машин не обмежена. Моделювання виконується на трьох етапах.

На першому етапі проводиться аналіз роботи мережі згідно з існуючою

архітектурою та принципами хмарної мережі. На цьому етапі оцінюється використання фізичних ресурсів серверів, виконується порівняльний аналіз їх завантаженості, затримки передачі пакетів, кількості оброблених та необроблених запитів. Особлива увага приділяється сервісам, які мають найбільшу кількість необроблених запитів в розробленому сценарії.

Другий етап включає моніторинг інфраструктури системи та тривалості обслуговування запитів. Використання інтегрованої архітектури управління дозволяє контролювати доступні апаратні та програмні ресурси.

Третій етап включає алгоритм балансування навантаження для сервісів, де тривалість обслуговування запитів є найбільшою. З взаємодією алгоритму балансування навантаження та інтегрованого управління інфраструктурою очікується зменшення тривалості обслуговування запитів, підвищення продуктивності та якості мережі, а також розвантаження сервера.

Моделювання проводилося на два етапи: до використання запропонованих рішень та після їх застосування. Інтенсивність надходження запитів для кожного з сервісів представлена на рисунку 3.13.

На рисунку 3.14 а, б, в, г, д, е відображено тривалість обслуговування запитів надання кожного типу сервісу. У момент переходу з першого етапу на другий спостерігається зниження цього параметра та здійснюється контроль за ресурсами кожного сервера.

Параметри кожного з сервісів та тривалість обслуговування запитів на надання кожного з них наведено у таблиці 3.5.

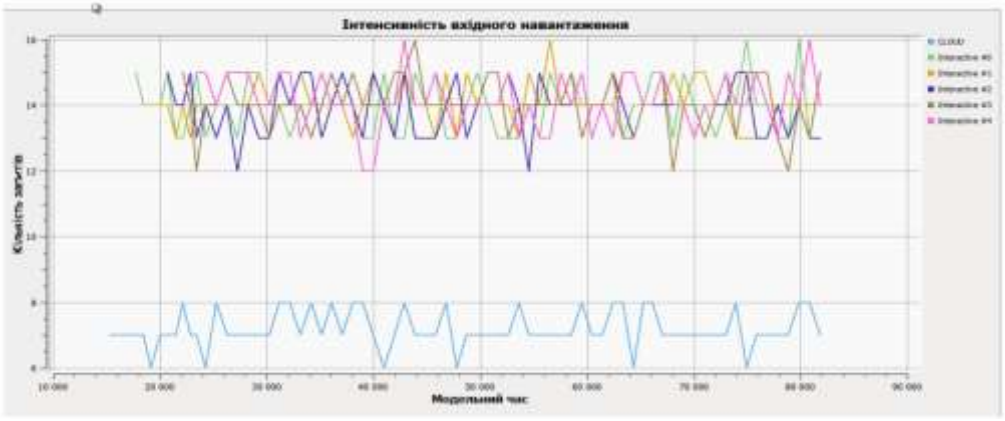
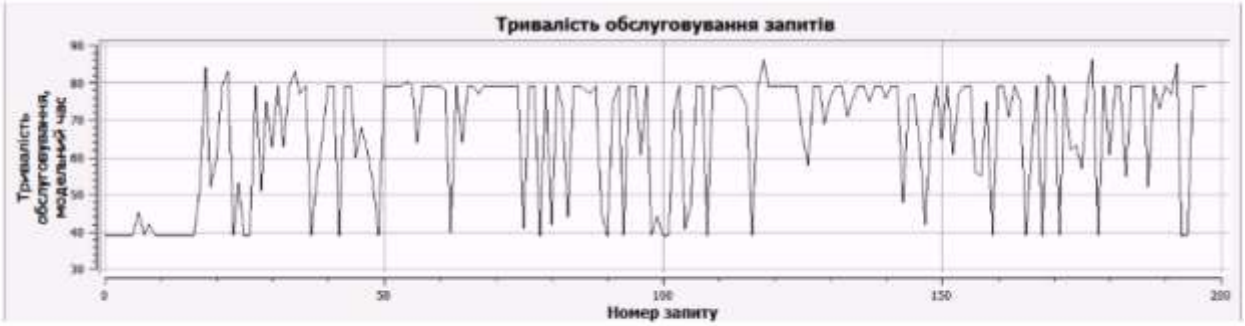
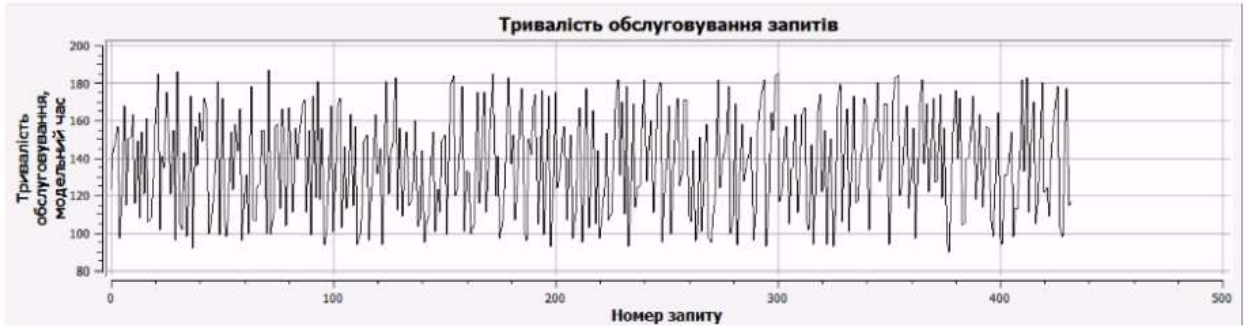


Рис. 3.13. Інтенсивність надходження запитів

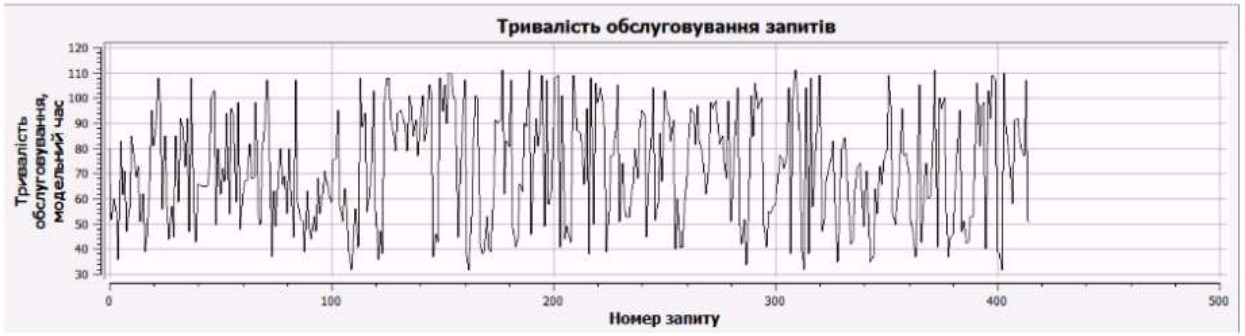


а)

б)

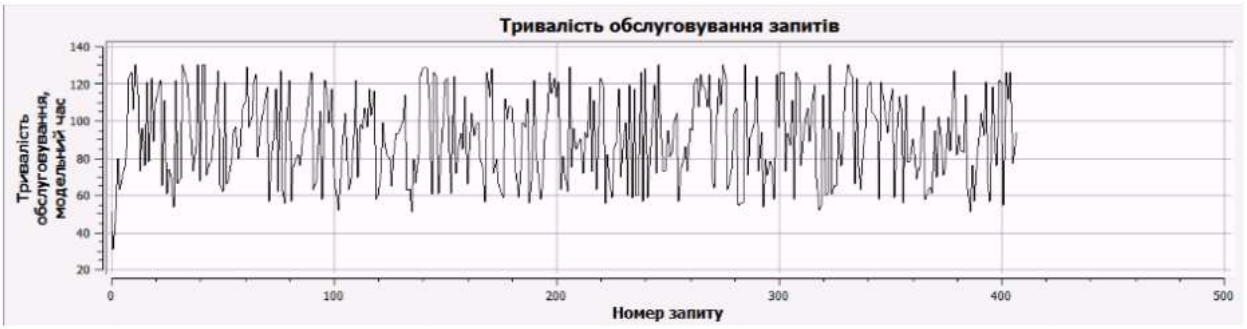


в)



г)

Рис. 3.14. Тривалість обслуговування запитів на надання кожного типу сервісу



д)

е)

Продовження Рис. 3.14. Тривалість обслуговування запитів на надання кожного типу сервісу

Таблиця 3.5.

Тривалість обслуговування запитів на надання послуг

Назва хмарного сервісу	Адреса атомарного сервісу	Тривалість надання сервісу	Кількість запитів на надання сервісу, що надійшли в систему	Кількість опрацьованих запитів
1	Instance 1 {1001, 2001, 3001}	70 мод.сек.	212	212
2	Instance 2 {1002, 1003, 2002}	150 мод.сек.	394	391
3	Instance 3 {2003 3002, 3003}	130 мод.сек.	367	365
4	Instance 4 {4001, 4002, 4003}	80 мод.сек ..	405	404
5	Instance 5 {4004, 5001, 5002}	90 мод.сек.	374	372
6	Instance 6 {5003, 5004, 6001}	60 мод.сек.	332	332

З результатів, поданих на рисунку 3.14 й у таблиці видно, що найбільше запитів надходить другого сервіс і тривалість їх обробки, загалом, близько 150 секунд модельного часу. Одна секунда модельного часу дорівнює одній

мікросекунді реального часу моделювання стану системи. Відповідно, апаратних та програмних ресурсів для надання ще одного компонента другого сервісу недостатньо. Буфери віртуальних машин переповнені запитами. У разі оркестратор приймає рішення про міграцію компонентів сервісу в інший фізичний сервер. На рисунку 3.15 наведено вид інфраструктури мережі після міграції та включення алгоритму балансування навантаження.

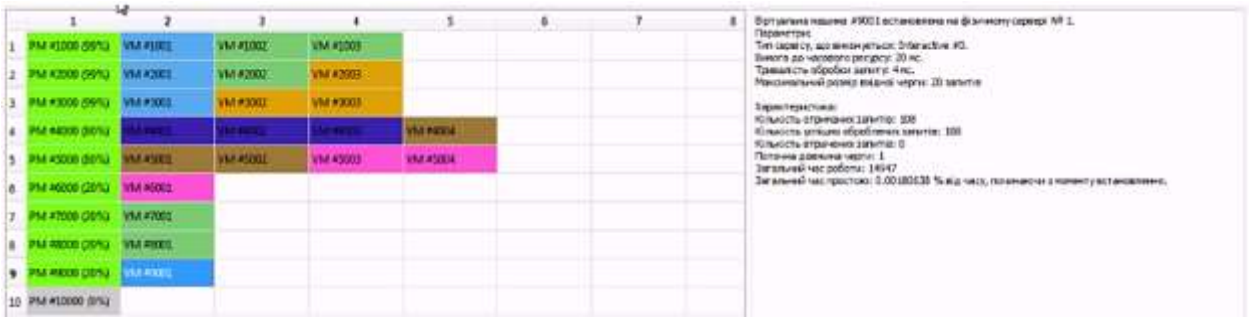


Рис. 3.15. Інфраструктури мережі після міграції та включення алгоритму балансування навантаження

Після включення алгоритму балансування навантаження тривалість обслуговування запитів зведено у таблиці 3.6.

Таблиця 3.6.

Тривалість обслуговування запитів на надання послуг при включенні алгоритму балансування навантаження

Назва хмарного сервісу	Адреса атомарного сервісу	Тривалість надання сервісу	Кількість запитів на надання сервісу, що надійшли до системи	Кількість опрацьованих запитів
1	Instance 1 {1001, 2001, 3001}	70 мод.сек	407	407
2	Instance 2 {1002, 1003, 2002}	50 мод.сек	739	739
3	Instance 3 {2003 3002, 3003}	130 мод.сек	711	706
4	Instance 4 {4001, 4002, 4003}	80 мод.сек	695	692
5	Instance 5 {4004, 5001, 5002}	90 мод.сек	732	728
6	Instance 6 {5003, 5004, 6001}	60 мод.сек	740	740

На рисунку 3.16 відображено тривалість обслуговування запитів на надання другого типу сервісу після переходу на менш завантажену фізичну машину. Як видно, спостерігається зниження цього параметра та здійснюється контроль за ресурсами кожного сервера.

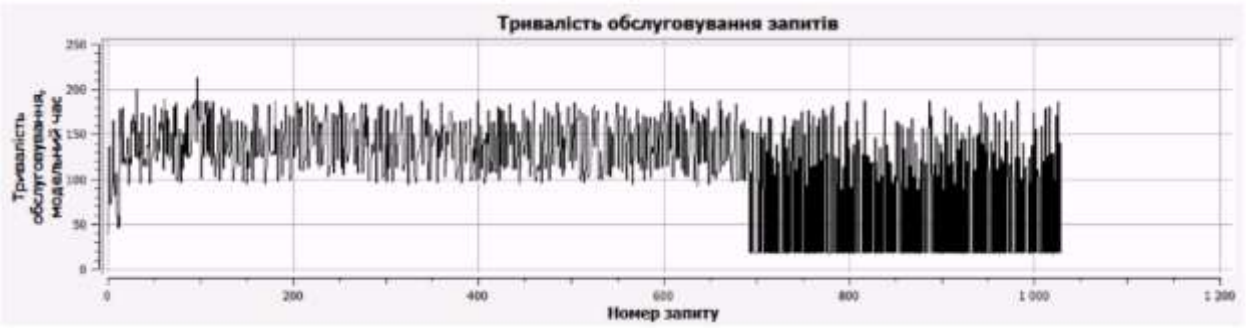


Рис. 3.16. Тривалість обслуговування запитів на надання другого типу сервісу з урахуванням запропонованих рішень

Аналіз отриманих результатів свідчить про те, що реалізація балансування навантаження за допомогою інтегрованої архітектури управління з використанням технології NVF призводить до значного скорочення тривалості обслуговування запитів, приблизно втричі. Завдяки комплексній оцінці телекомунікаційних та програмно-апаратних ресурсів запропонований метод дозволяє ефективно зменшити час затримки при наданні сервісу користувачам та розвантажити сервер, який має найвище навантаження. Це стає особливо важливим у випадках, коли в мережі присутній значний обсяг різноманітних сервісів і передається великий обсяг трафіку.

Моніторинг якості обслуговування, представлений на рисунку 3.14, демонструє, що після переміщення компонентів сервісу середня тривалість обробки запитів та затримка пакетів з кінця до кінця зменшилися з 150 до 55 мс, що складає майже втричі менше початкового значення.

3.6 Дослідження ефективності використання запропонованих рішень та їх вплив на якість надання хмарних сервісів

Для підтвердження ефективності застосування запропонованих у дисертаційній роботі рішень було досліджено їх вплив на якість надання хмарних сервісів з використанням розробленого програмно-апаратного комплексу. на 77 хв. тестування роботи системи оркестратор на основі аналізу завантаженості буфера (рисунок Б.1 додатка Б) було прийнято рішення про реплікацію третьої компоненти п'ятого сервісу на основі запропонованого методу в другому розділі і розгортання на другому сервері ще однієї віртуальної машини з відповідним компонентом.

Архітектура системи після такої реплікації зображена на рисунку 3.17. Слід зазначити, що доступ користувачів до цього компонента та його репліки здійснюється на основі методу, запропонованого у п.2.1.

Відповідно зміна завантаженості центрального процесора серверів 2 та 4 зображена на рисунку Б.2а та б додатка Б.

З рисунків видно, що завантаженість сервера 4 після балансування навантаження зменшилася вдвічі, тоді як середня завантаженість другого сервера становила 60%. На основі аналізу інтенсивності надходження запитів на компоненти п'ятого сервісу (рисунок Б.3 додатка Б) на 178 хв, 811 хв. та 812 хвилині тестування роботи системи оркестратор приймає рішення про аналогічні реплікації компонентів цього сервісу на сервери в межах ЦОД 1. Зміна завантаженості серверів 2, 3, 4, 5 в результаті балансування можна спостерігати на рисунку Б.4 а-г додатка Б.

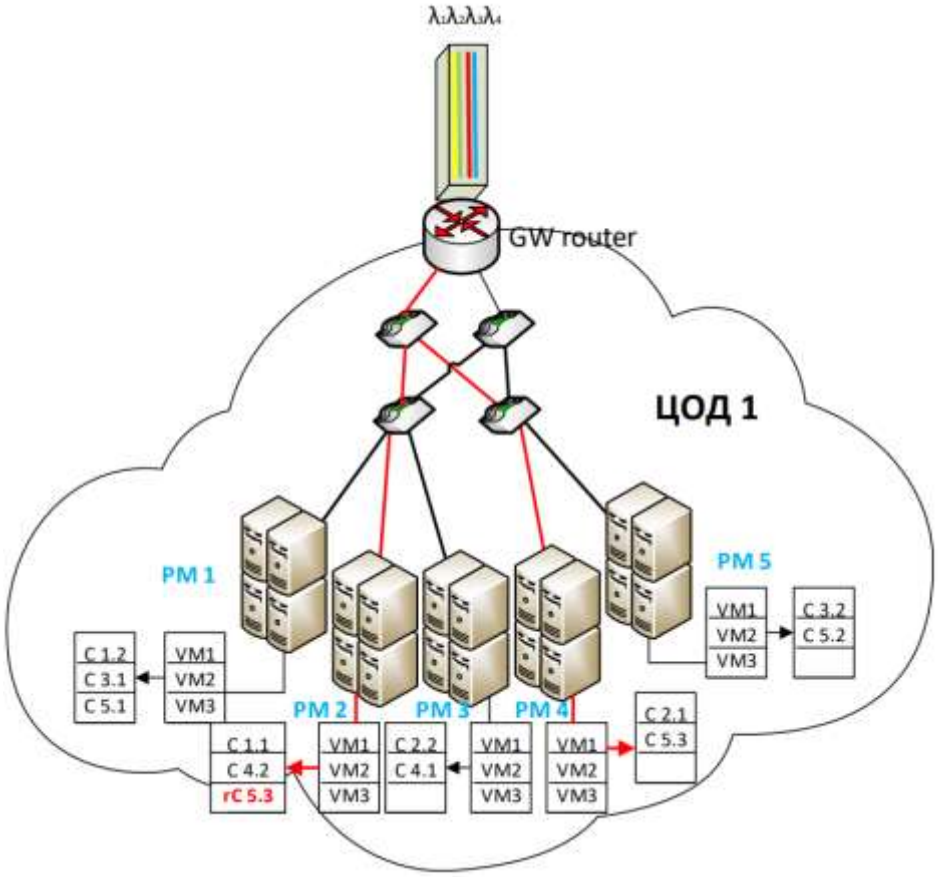


Рис. 3.17. Архітектура системи після реплікації

В результаті дослідження завантаженості віртуальних машин серверів (рисунок Б.5 додатка Б) та аналізі інтенсивності надходження запитів на компоненти п'ятого сервісу оркестратор приймає рішення про реплікацію компонентів цього сервісу на ЦОД 2 у 1227 та 1229 хвилинах роботи системи та здійснює управління ресурсами як у межах кожного з них (за допомогою методу та архітектури описаних у другому розділі), так і між ними (на основі системи управління оптичними ресурсами).

Тому архітектура системи набуде вигляду представленого на рисунку 3.18.

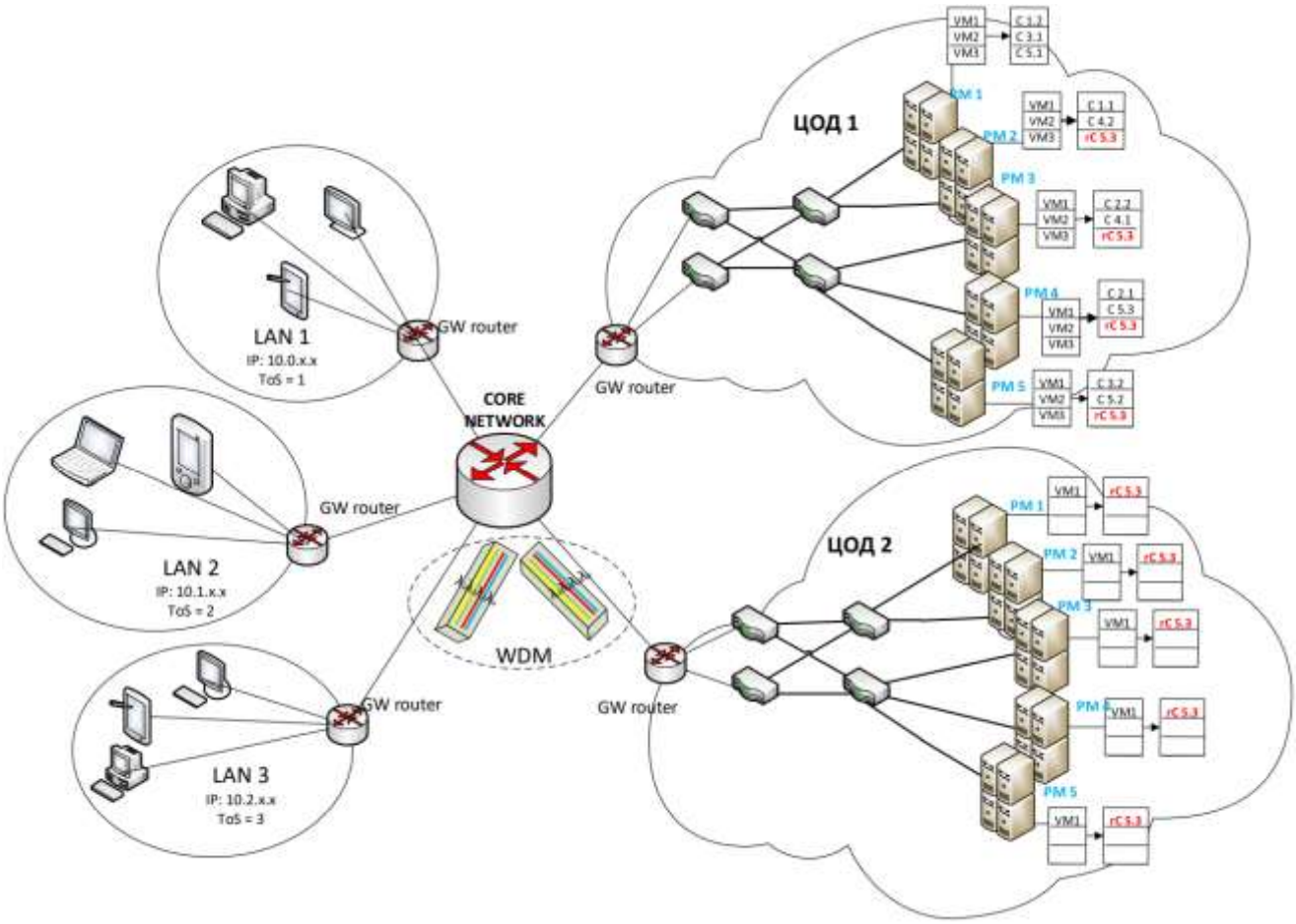


Рис. 3.18. Розташування компонентів сервісів інфраструктури досліджуваної мережі

Завантаженість кожного сервера в результаті роботи розробленої системи на основі запропонованих у дисертаційній роботі рішень дещо збільшилася, проте тривалість обслуговування запитів, тобто затримка на надання сервісів, зменшувалася, а потім і рівень QoS підвищився. Це з графіків, наведених на рисунку Б.6 додатка Б.

Відповідно до представленого на рисунку Б.6 додатка Б даних, чітко розпізнаються моменти активації розроблених методів та алгоритмів. Завдяки успішній інтеграції системи управління як на локальному, так і на глобальному рівні, вдалося підтримувати необхідний рівень якості надання сервісів в середньому на рівні 0,02 сек. Детальне розглядання інтенсивності надходження запитів від користувачів на п'ятий сервіс та тривалості їх обслуговування (рисунок Б.7 додатка Б) показує, що в кожний момент балансування навантаження

тривалість затримки на надання сервісу зменшується.

У перший момент (на 77 хв. рисунка Б.7а), затримка на надання сервісу зменшується з 1 сек. до 0,02 сек., що становить майже 80%. У другий момент (на 178 хв.), затримка на надання сервісу зменшується з 1 сек. до 0,04 сек. на протязі 811 та 812 хв. (рисунок Б.6б). Після значного та тривалого інтенсивного надходження запитів, затримка на надання сервісу зменшилася більш ніж на 85%. Це свідчить про ефективне застосування методу балансування навантаження з урахуванням доступності фізичних ресурсів у межах одного центру обробки даних і методу локального розподілу оптичних ресурсів під час передачі запитів на компоненти сервісу між центрами обробки даних.

Висновки з 3-го розділу

1. З метою оцінки ефективності моделі надання сервісу, яка базується на методі пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних, було створено імітаційну модель структури центру обробки даних. Результати моделювання вказують на те, що при стабільності структури мережі спостерігається зменшення затримки, що сприяє прискоренню надання сервісу кінцевому користувачеві та забезпечує відповідний рівень якості обслуговування (QoS). Застосування методу пошуку маршруту із врахуванням стійкості структури дозволило зменшити час затримки на 35%, що призводить до ефективного прискорення надання хмарних сервісів.

2. Внаслідок моделювання структури центру обробки даних у хмарній інфраструктурі виявлено, що зменшення затримки викликає зменшення часу пошуку каналів для передачі запитів, що в свою чергу сприяє скороченню загального часу передачі сервісу. Загальне зменшення затримки та покращення якості надання сервісів користувачам мережі досягнуті на рівні 12%. Виявлено, що зі збільшенням стійкості структури зменшується час пошуку каналів для передачі запитів, призводячи до скорочення загального часу передачі хмарного сервісу.

3. Проведено імітаційне моделювання хмарної інфраструктури, що базується на моделі розгортання віртуальних машин на фізичних серверах та наданні сервісу

через метод балансування навантаження з аналізом доступних компонентів хмарного сервісу. Результати вказують на те, що реалізація інтегрованої архітектури управління з використанням технології NVF дозволяє значно зменшити тривалість обслуговування запитів на приблизно 3 рази. Завдяки комплексній оцінці телекомунікаційних та програмно-апаратних ресурсів, використання запропонованого методу призводить до зменшення часу затримки надання сервісу та розвантаження найбільш завантаженого сервера. Це має важливе значення в умовах великої кількості сервісів та значного обсягу трафіку в мережі. Моніторинг якості обслуговування свідчить про зниження середньої тривалості обробки запитів та затримки пакетів з 150 до 55 мс після перенесення компонентів сервісу.

4. Розроблено модель управління оптичними ресурсами між центрами обробки даних хмарної інфраструктури, яка базується на методі локального розподілу та управлінні сегментом WDM мережі. Це дозволяє оптимізувати використання оптичних ресурсів фізичного тракту, зменшити ймовірність блокування при прокладанні нових логічних каналів і спрощує систему управління. Застосування запропонованого методу призводить до зниження завантаженості мережевого обладнання та зменшення його енергоспоживання в півтора рази. Застосування алгоритму прокладання наскрізних тунелів дозволяє розвантажити довжину хвилі понад 60% і використовувати вільні довжини хвиль з високим порядковим номером для прокладання наскрізних тунелів між вузлами. Це є ключовим умовою в умовах значного обсягу трафіку в мережах.

5. Розроблено комплекс надання хмарних сервісів із гарантованим рівнем QoS, що практично підтверджує ефективність запропонованих методів та алгоритмів, враховуючи як програмну, так і апаратну складову. Використання розробленого програмно-апаратного комплексу підтверджує необхідність ефективного балансування навантаження з урахуванням доступності компонентів та стійкості структури мережі центрів обробки даних, що сприяє зменшенню тривалості обробки запитів на надання сервісу та підвищенню загальної якості обслуговування.

6. Успішна інтеграція системи управління на локальному та глобальному рівнях дозволяє підтримувати необхідний рівень якості надання сервісів в середньому на рівні 0,02 сек. Ефективне використання методу балансування навантаження, який базується на доступності фізичних ресурсів у межах одного центру обробки даних та методі локального розподілу оптичних ресурсів при передачі запитів на компоненти сервісу між центрами обробки даних, призводить до значного зменшення затримки у перший момент (на 77 хв.) та другий момент (на 178 хв.) моделювання, що визначається значущим та тривалим надходженням запитів.

ВИСНОВКИ

У даній роботі було вирішено завдання поліпшення характеристик надання хмарних сервісів, зокрема збільшення стійкості віртуальних топологій центру обробки даних хмарної інфраструктури з метою удосконалення алгоритмів балансування навантаження для підвищення ефективності хмарних сервісів, сформованих за допомогою дистанційно-векторних методів. Це стало актуальним в умовах зростання різноманітності потоків у сучасних гетерогенних мережах для відповіді на вимоги комунікаційних додатків.

Основні висновки дослідження включають:

1. Проведено аналіз методів надання хмарних сервісів зі хмарною архітектурою. Обрано взаємодію моделей DiffServ і IntServ як оптимальний спосіб забезпечення необхідної якості обслуговування (QoS) для передачі запитів. Основним критерієм QoS визначено затримку передачі, враховуючи рекомендації ITU-T Y.1540. Виявлено, що віртуальна структурна мережа не враховується при формуванні метрики, що призводить до погіршення QoS при міграції атомарних елементів хмарного сервісу.

2. Запропоновано метод пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних для поліпшення QoS та зменшення часу пошуку маршруту. Метод забезпечує компроміс між вибором оптимального маршруту та вимогами до параметрів QoS, ураховуючи ймовірність одночасного існування потоків із максимальними вимогами до якості обслуговування.

3. Удосконалений метод пошуку маршруту дозволив зменшити затримку в процесі пошуку маршруту на 12% та середню затримку на 35%. Це сприяло прискоренню процесу надання хмарних сервісів.

4. Розроблено метод балансування навантаження на основі віртуалізації мережевих функцій та оцінки доступності компонентів. Цей метод зменшує тривалість обслуговування запитів у 3 рази та дозволяє розподіляти навантаження між вузлами для оптимізації роботи розподіленої системи.

5. Запропоновано модель та метод управління мережевими ресурсами оптичної мережі між центрами обробки даних хмарної інфраструктури, що поліпшило часові параметри обслуговування та зменшило завантаженість оптичного мережевого тракту на 1,5 раза.

6. Розроблено комплекс надання хмарних сервісів із гарантованим рівнем QoS, що ефективно використовує методи балансування навантаження та управління мережевими ресурсами. Підтверджено, що інтеграція такого комплексу можливо досягти необхідного рівня якості надання послуг та значного зменшення затримок.

ПЕРЕЛІК ПОСИЛАНЬ

1. Priya, V., Kumar, C. S., & Kannan, R. (2019). Resource scheduling algorithm with load balancing for cloud service provisioning. Applied Soft Computing, 76, 416-424. doi:10.1016/j.asoc.2018.12.021

2. Milan, S. T., Rajabion, L., Ranjbar, H., & Navimipour, N. J. (2019). Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments. Computers & Operations Research, 110, 159-187. doi:10.1016/j.cor.2019.05.022

3. Hussein, M. K., & Mousa, M. H. (2020). Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization. IEEE Access, 8, 37191-37201. doi:10.1109/access.2020.2975741

4. Ala'anzy, M., & Othman, M. (2019). Load Balancing and Server Consolidation in Cloud Computing Environments: A Meta-Study. IEEE Access, 7, 141868-141887. doi:10.1109/access.2019.2944420

5. Dong, Y., Xu, G., Ding, Y., Meng, X., & Zhao, J. (2019). A 'JointMe' Task Deployment Strategy for Load Balancing in Edge Computing. IEEE Access, 7, 99658-99669. doi:10.1109/access.2019.2928582

6. Juarez, F., Ejarque, J., Badia, R.M.: Dynamic energy-aware scheduling for parallel task-based application in cloud computing. Future Gener. Comput. Syst. 78, 257–271 (2018)

7. Zhang, Y., Cheng, X., Chen, L., Shen, H.: Energy-efficient tasks scheduling heuristics with multi-constraints in virtualized clouds. J. Grid Comput. 16, 459–475 (2018)

8. Liu, X.-F., Zhan, Z.-H., Deng, J.D., Li, Y., Gu, T., Zhang, J.: An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans. Evol. Comput. 22(1), 113–128 (2018)

9. Naik, K., Gandhi, G.M., Patil, S.H.: Multiobjective virtual machine selection for task scheduling in cloud computing. In: Verma, N.K., Ghosh, A.K. (eds.) Computational Intelligence: Theories, Applications and Future Directions, pp. 319–331.

Springer, Singapore (2019)

10. Fatemeh, E., Babamir, S.M.: A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. Concurr. Comput.: Pract. Exp. 30(12), e4368 (2018)

11. Chawla, A., Ghumman, N.S.: Package-based approach for load balancing in cloud computing. Big Data Analytics, pp. 71–77. Springer, Singapore (2018)

12. Jana, B., Chakraborty, M., Mandal, T.: Task scheduling technique based on particle swarm optimization algorithm in cloud environment. In: Pant, M., Ray, K., Sharma, T.K., Rawat, S. (eds.) Soft Computing: Theories and Applications, pp. 525–536. Springer, Singapore (2019)

13. Guo, M., Guan, Q., Ke, W.: Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload. IEEE Access 6, 15178–15191 (2018)

14. Niknam, S., Wang, P., Stefanov, T.: Resource optimization for real-time streaming applications using task replication. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37(11), 2755–2767 (2018)

15. Gill, S.S., Buyya, R.: Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: from fundamental to autonomic offering. J Grid . Comput. 1–33 (2018)

16. Baker, T., et al.: Greeaodv: an energy efficient routing protocol for vehicular ad hoc networks. International Conference on Intelligent Computing. Springer, Cham. (2018)

17. Bala, M.: Proportionate resource utilization based VM allocation method for large scaled datacenters. Int. J. Inf. Technol. 10(3), 349–357 (2018)

18. Xie, Lei, et al. "A Novel Self-Adaptive VM Consolidation Strategy Using Dynamic Multi-Thresholds in IaaS Clouds." Future Internet 10.6 (2018): 52.

19. Zhao, H., et al.: Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE T. Parall. Distr. 29(6), 1385–1400 (2018)

20. Mondal, R. K., et al.: Load balancing of unbalanced matrix problem of the sufficient machines with min-min algorithm. Methodologies and application issues of contemporary computing framework, pp. 81–91. Springer, Singapore (2018)

21. Adhikari, M., Amgoth, T.: Heuristic-based load-balancing algorithm for IaaS cloud. Futur. Gener. Comput. Syst. 81, 156–165 (2018)

22. Alaei, N., Safi-Esfahani, F.: RePro-active: a reactive–proactive scheduling method based on simulation in cloud computing. J. Supercomput. 74(2), 801–829 (2018)

23. Aazam, M., Zeadally, S., Harras, K.A.: Offloading in fog computing for IoT: review, enabling technologies, and research opportunities. Futur. Gener. Comput. Syst. 87, 278–289 (2018)

24. Juarez, F., Ejarque, J., Badia, R.M.: Dynamic energy-aware scheduling for parallel task-based application in cloud computing. Futur. Gener. Comput. Syst. 78, 257–271 (2018)

25. Hussain, A., et al.: RALBA: a computation-aware load balancing scheduler for cloud computing. Clust. Comput. 21(3), 1667–1680 (2018)

26. The Three Types of Load Balancing You Meet in the Cloud [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://devcentral.f5.com/articles/the-three-types-of-load-balancing-you-meetin-the-cloud-30704.

27. Load balancing your load balancers for endless scalability [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://www.loadbalancer.org/blog/load-balancing-your-load-balancers-forendless-scalability/.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА
«РОЗРОБКА ТА АНАЛІЗ АЛГОРИТМІВ ДЛЯ ЕФЕКТИВНОГО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У ХМАРНИХ СЕРВІСАХ»

Виконав: студент групи ПДМ – 64, Луппа Олексій Андрійович

Керівник: к.т.н., доц. кафедри ІПЗ, Яскевич Владислав Олександрович

Київ - 2023

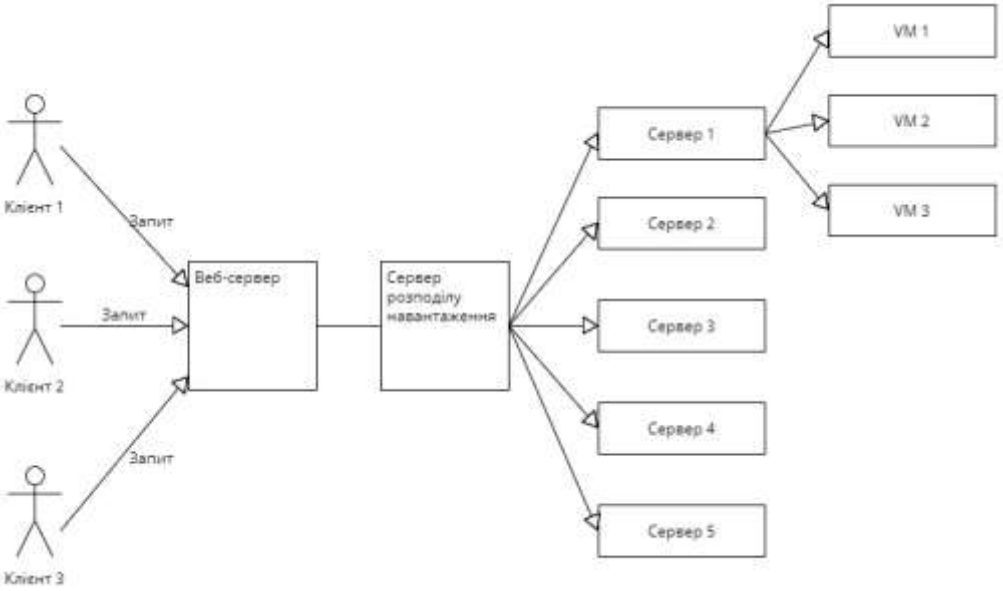
МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності хмарних сервісів за рахунок алгоритмів балансування навантаження.

Об'єкт дослідження: процес балансування навантаження у хмарних сервісах.

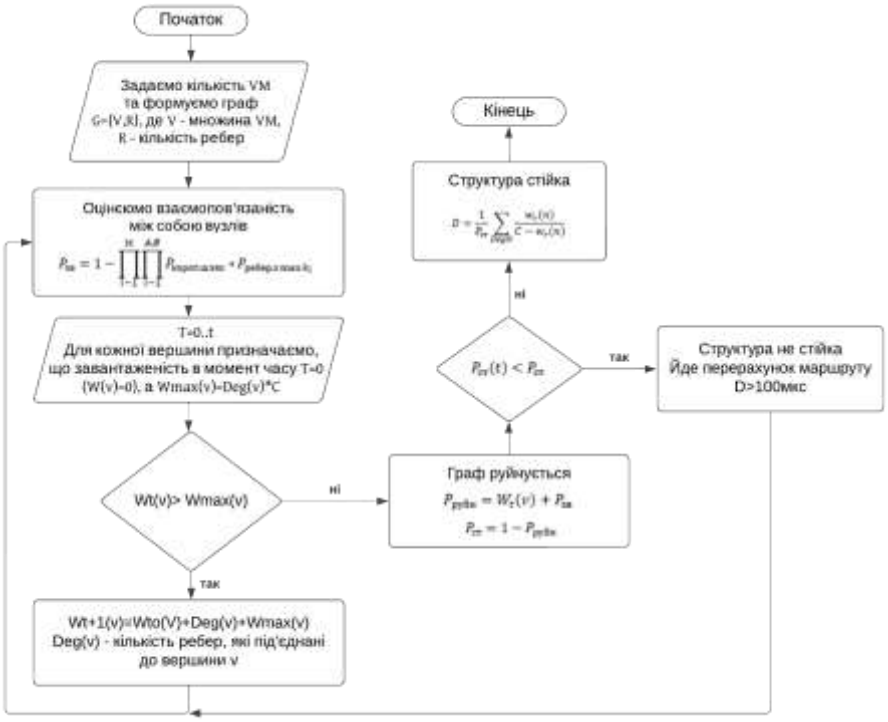
Предмет дослідження: алгоритми для ефективного балансування навантаження у хмарних сервісах.

АРХІТЕКТУРА РОЗПОДІЛЕНИХ СИСТЕМ «КЛІЄНТ -СЕРВЕР»

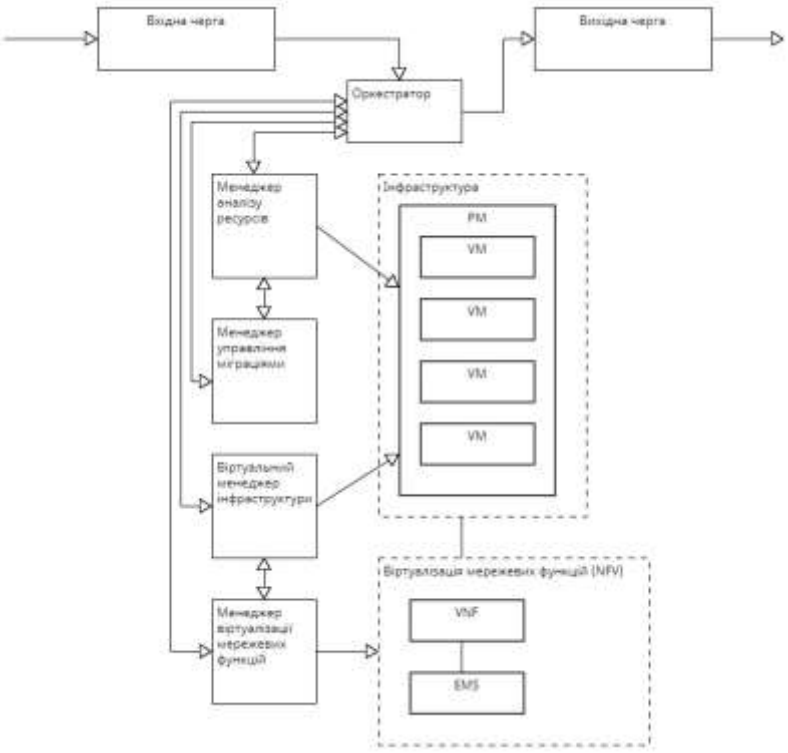


ТИПОВА ЛОГІЧНА ТОПОЛОГІЯ МЕРЕЖІ ЦЕНТРУ ОБРОБКИ ДАНИХ

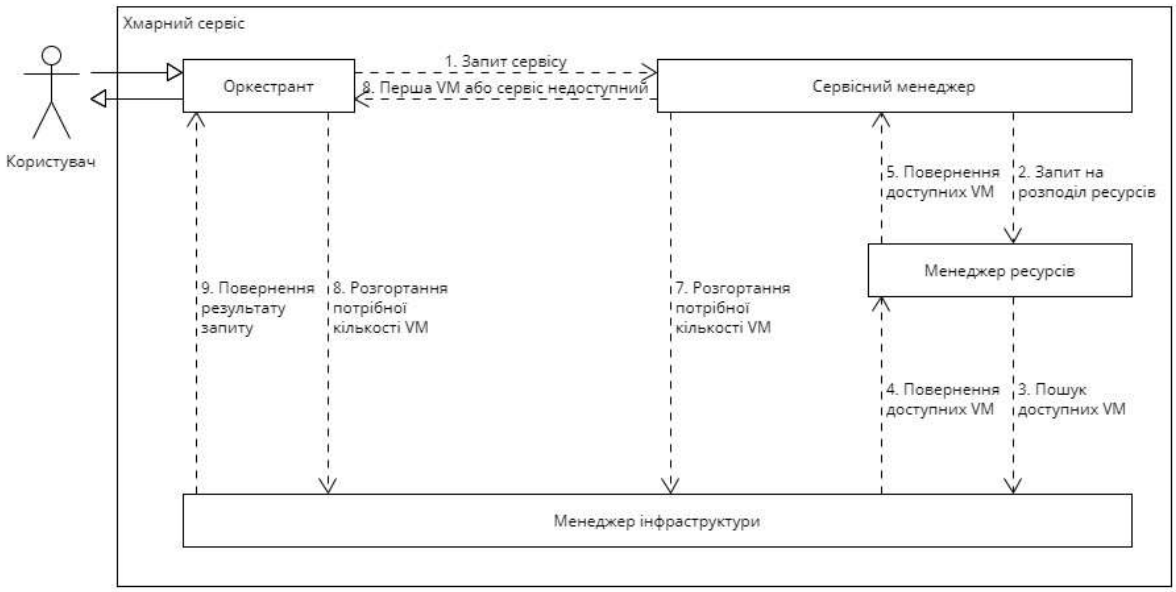
БЛОК-СХЕМА АЛГОРИТМУ РОБОТИ МЕТОДУ ПОШУКУ МАРШРУТУ З УРАХУВАННЯМ СТІЙКОСТІ СТРУКТУРИ ВІРТУАЛІЗОВАНОГО ЦЕНТРУ ОБРОБКИ ДАНИХ



КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ УПРАВЛІННЯ ХМАРНОЮ МЕРЕЖЕЮ



МОДЕЛЬ НАДАННЯ СЕРВІСУ



ТРИВАЛІСТЬ ОБСЛУГОВУВАННЯ ЗАПИТІВ НА НАДАННЯ СЕРВІСУ З УРАХУВАННЯМ ЗАПРОПОНОВАНИХ РІШЕНЬ

ВИСНОВКИ

1. Проведено аналіз методів надання хмарних сервісів зі хмарною архітектурою. Обрано взаємодію моделей DiffServ і IntServ як оптимальний спосіб забезпечення необхідної якості обслуговування (QoS) для передачі запитів. Виявлено, що віртуальна структурна мережа не враховується при формуванні метрики, що призводить до погіршення QoS при міграції атомарних елементів хмарного сервісу.

2. Запропоновано метод пошуку маршруту з урахуванням стійкості структури віртуалізованого центру обробки даних. Метод забезпечує компроміс між вибором оптимального маршруту та вимогами до параметрів QoS, ураховуючи ймовірність одночасного існування потоків із максимальними вимогами до якості обслуговування.

3. Удосконалений метод пошуку маршруту дозволив зменшити затримку в процесі пошуку маршруту на 12% та середню затримку на 35%. Це сприяло прискоренню процесу надання хмарних сервісів.

4. Розроблено метод балансування навантаження на основі віртуалізації мережевих функцій, метод зменшує тривалість запитів у 3 рази, з 150 до 55 мс.

5. Запропоновано модель та метод управління мережевими ресурсами оптичної мережі обробки даних хмарної інфраструктури, що поліпшило часові параметри обслуговування та зменшило завантаженість оптичного мережевого тракту на 1,5 раза.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Луппа О.А. Методи забезпечення приватності та інтеграції в сучасних додатках з урахуванням зменшення балансування навантаження // Зв'язок. – Київ: ДУТ, 2023 (ще не опублікована)

Тези доповідей:

1. Луппа О.А. Балансування навантаження на основі прогнозування використання ресурсів // Всеукраїнська науково-практична конференція «Telecommunication : problems and innovation». – Київ: ДУТ, 2023 (ще не опубліковано)
2. Луппа О.А. Адаптивні алгоритми балансування навантаження // Науково-практична конференція «Проблеми комп'ютерної інженерії». – Київ: ДУТ, 2023 (ще не опублікована)

Додаток А

Інтенсивність агрегованого навантаження, завантаженість сервісних компонентів та зміна тривалості обслуговування запитів

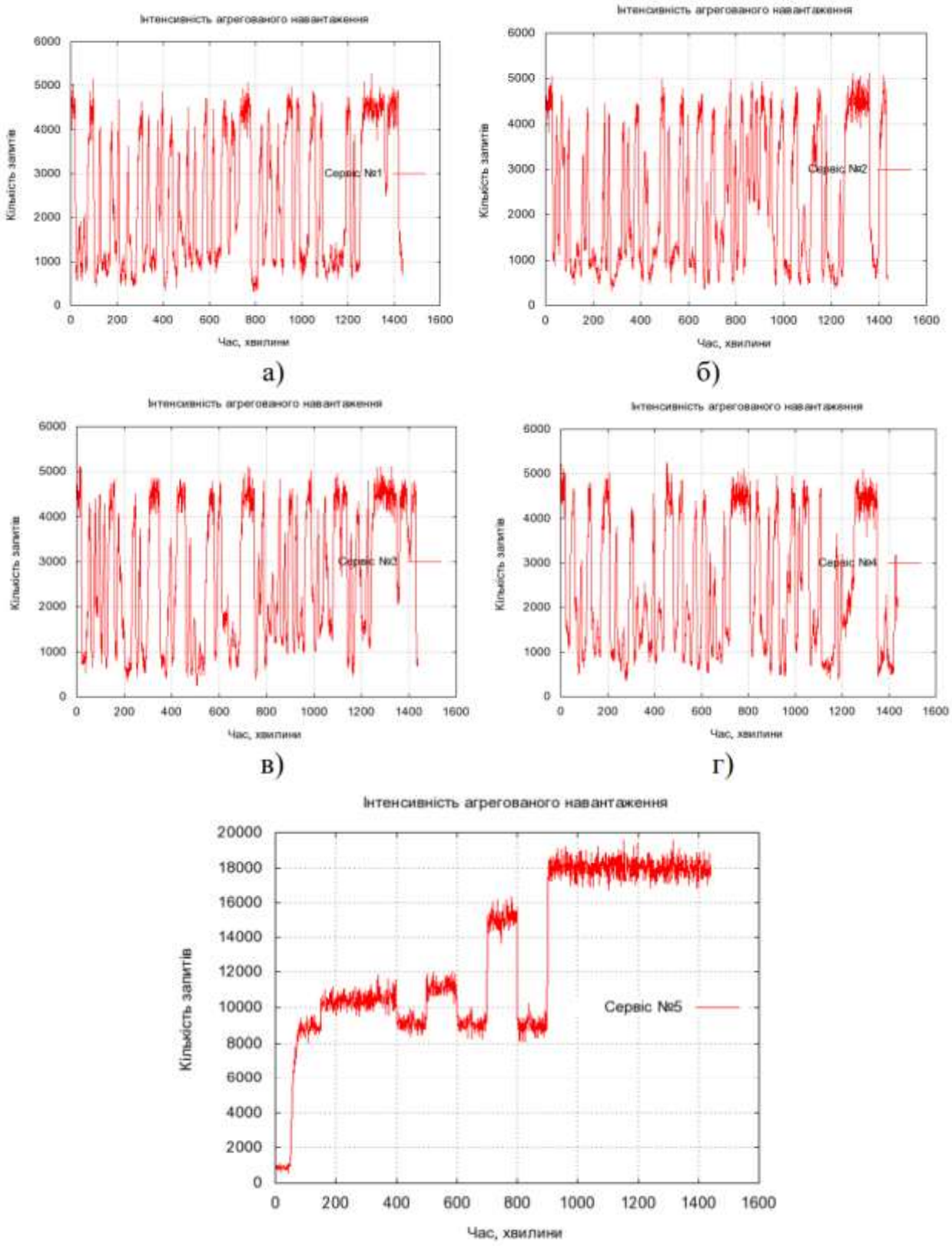


Рис. А.1. Інтенсивність агрегованого навантаження: а) на перший сервіс; б)

другого сервіс; в) третій сервіс; г) у четвертий сервіс; д) на п'ятий сервіс

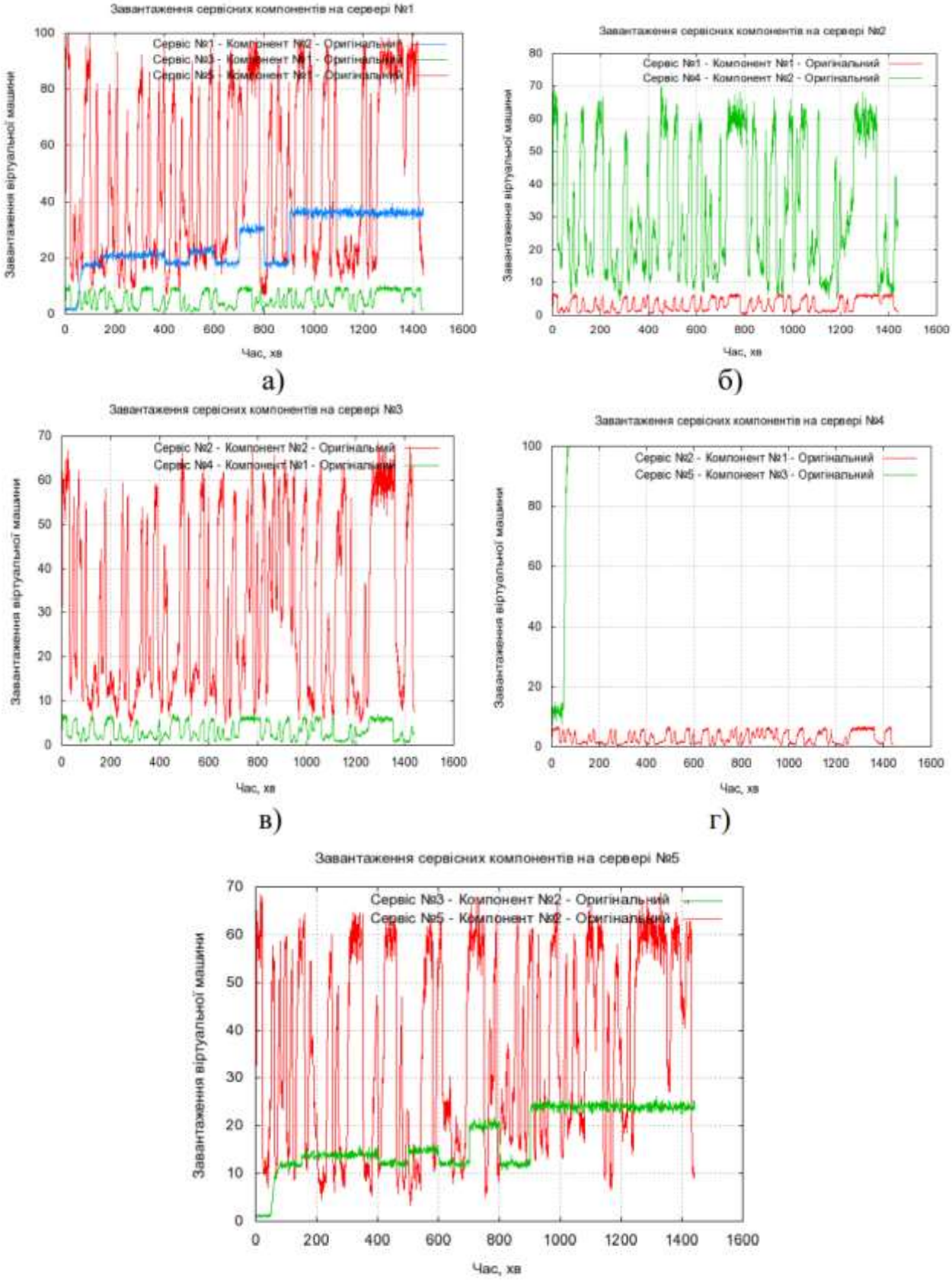
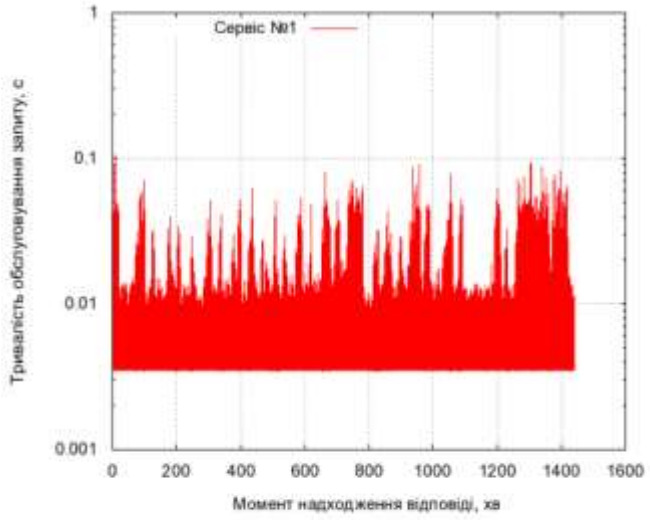
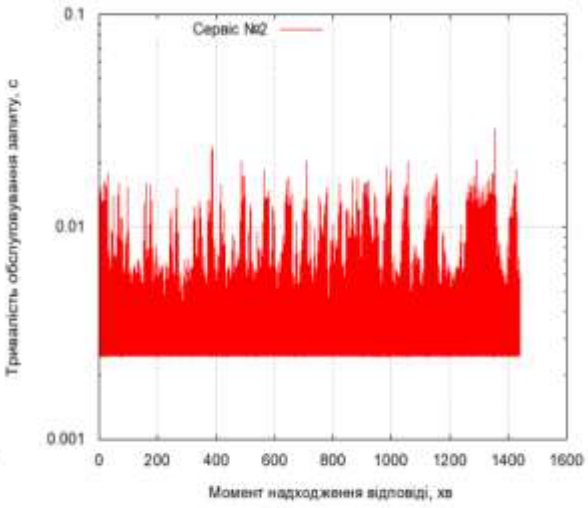


Рис. А.2. Завантаженість сервісних компонентів а) на першому сервері б) на

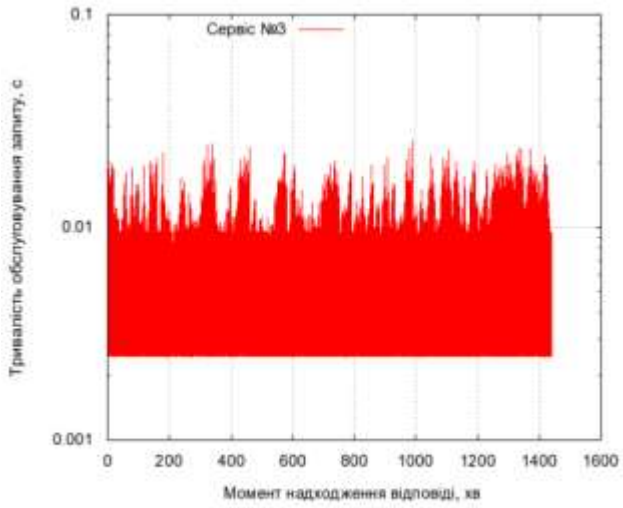
другому сервері в) на третьому сервері г) у четвертому сервері д) на п'ятому сервері



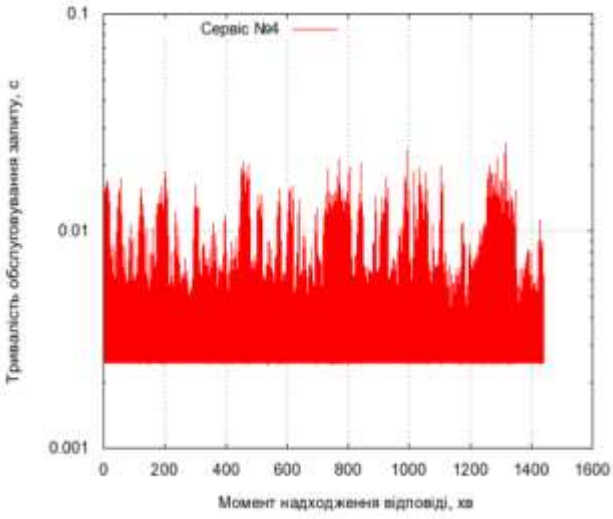
а)



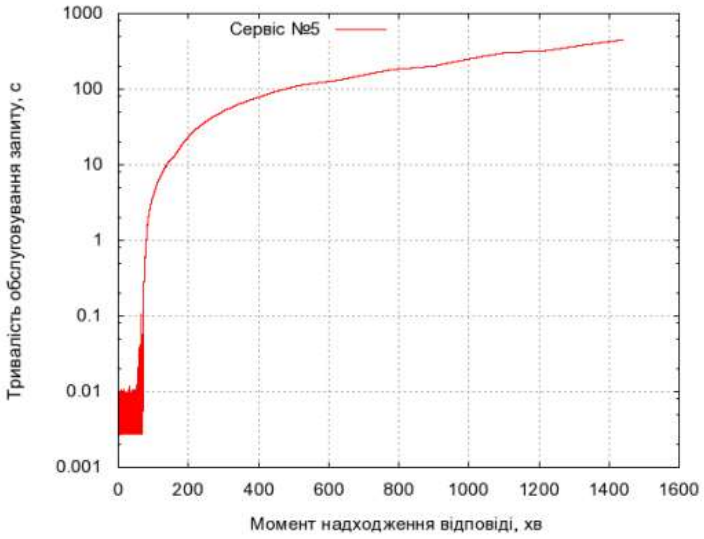
б)



в)



г)



д)

Рис. А.3. Зміна тривалості обслуговування запитів а) перший сервіс; б) другого сервіс; в) третій сервіс; г) у четвертий сервіс; д) на п'ятий сервіс

Додаток Б

Дослідження ефективності використання запропонованих рішень та їх вплив на якість надання хмарних сервісів

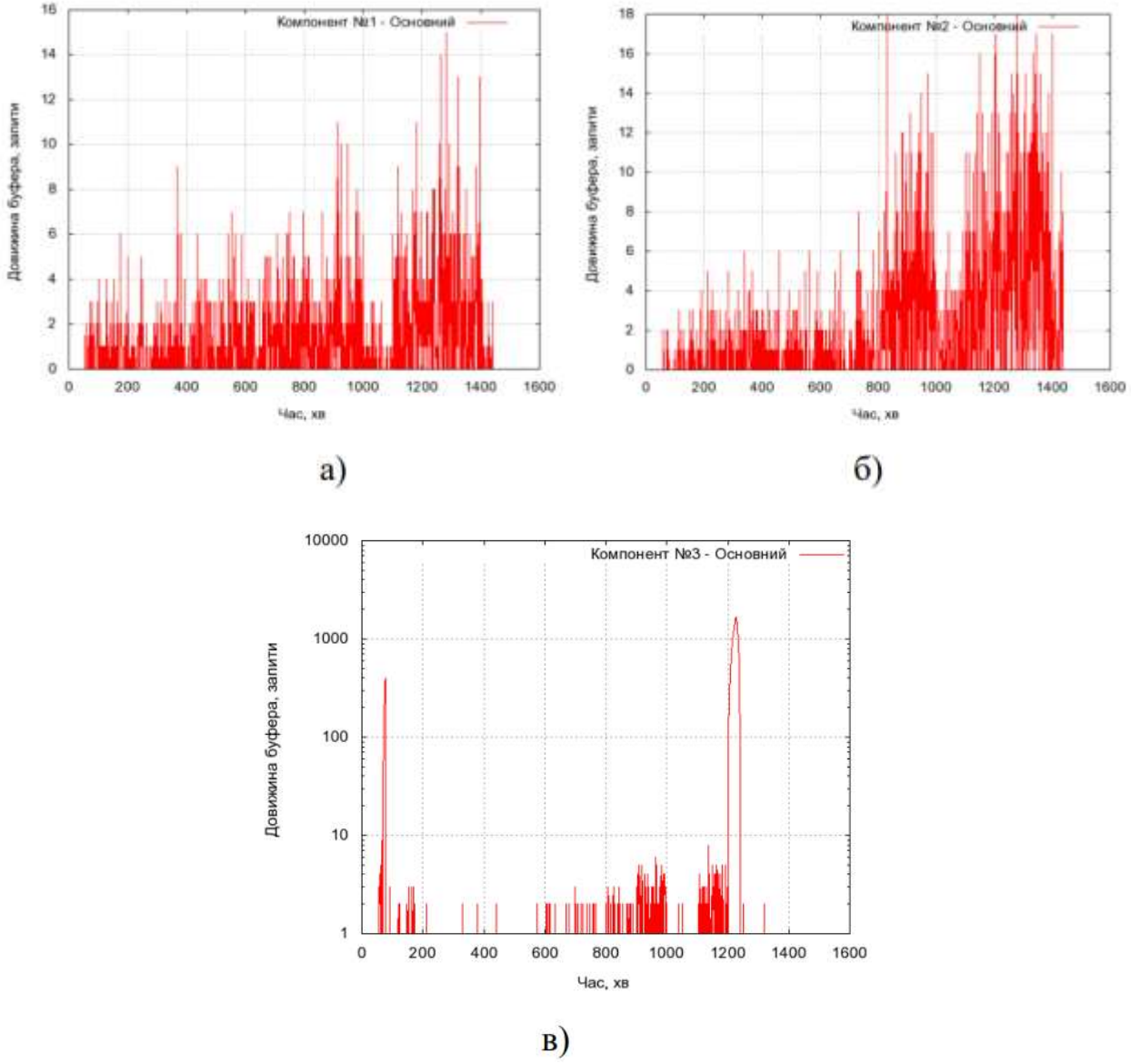
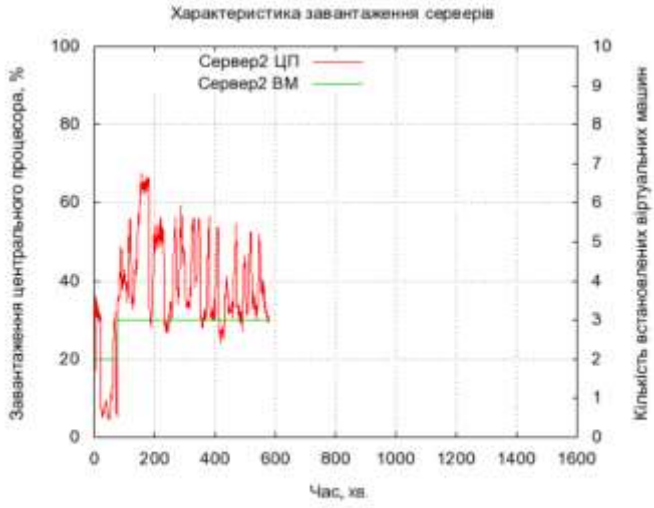
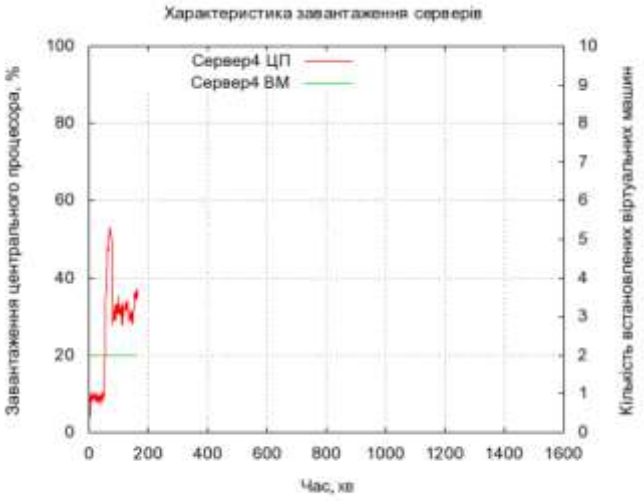


Рис. Б.1. Завантаження буферів на надання: а) першого компонента п'ятого сервісу; б) другого компонента п'ятого сервісу; в) третього компонента п'ятого сервісу



а)



б)

Рис. Б.2. Характеристика завантаження: а) другого сервера; б) четвертого сервера

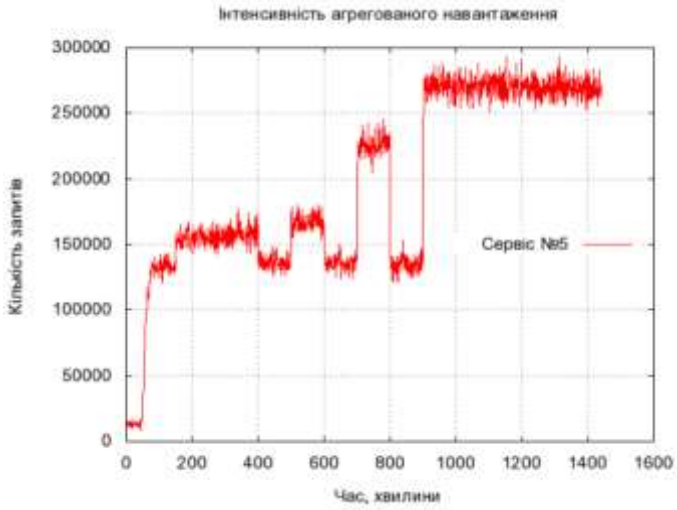


Рис. Б.3. Характеристика надходження запитів на компоненти п'ятого сервісу

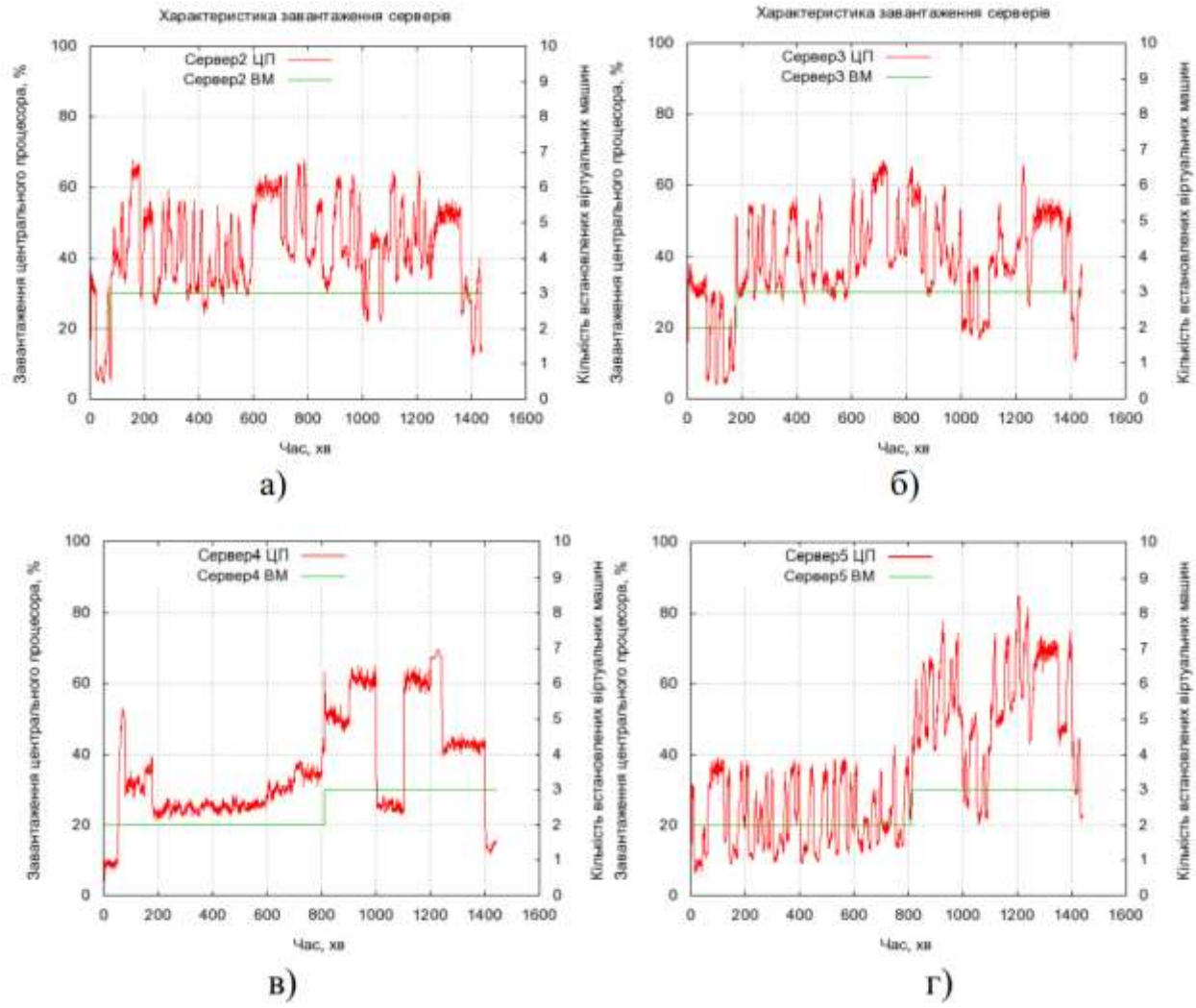


Рис. Б.4. Характеристика завантаженості: а) другого сервера; б) третього сервера; в) четвертий сервер; г) п'ятого сервера внаслідок балансування

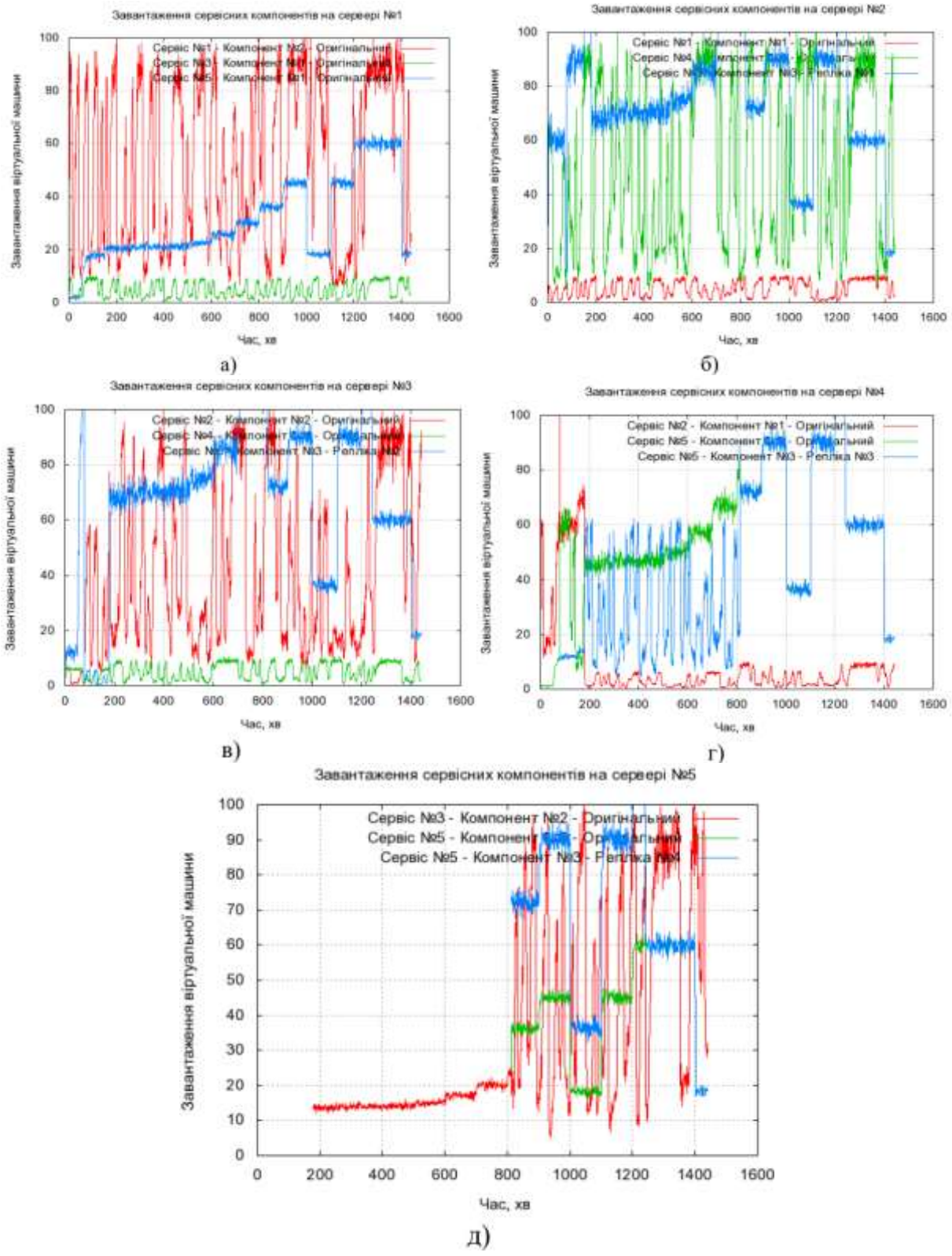


Рис. Б.5. Характеристика завантаження сервісних компонентів а) на першому сервері б) на другому сервері в) на третьому сервері г, д) на четвертому та п'ятому серверах

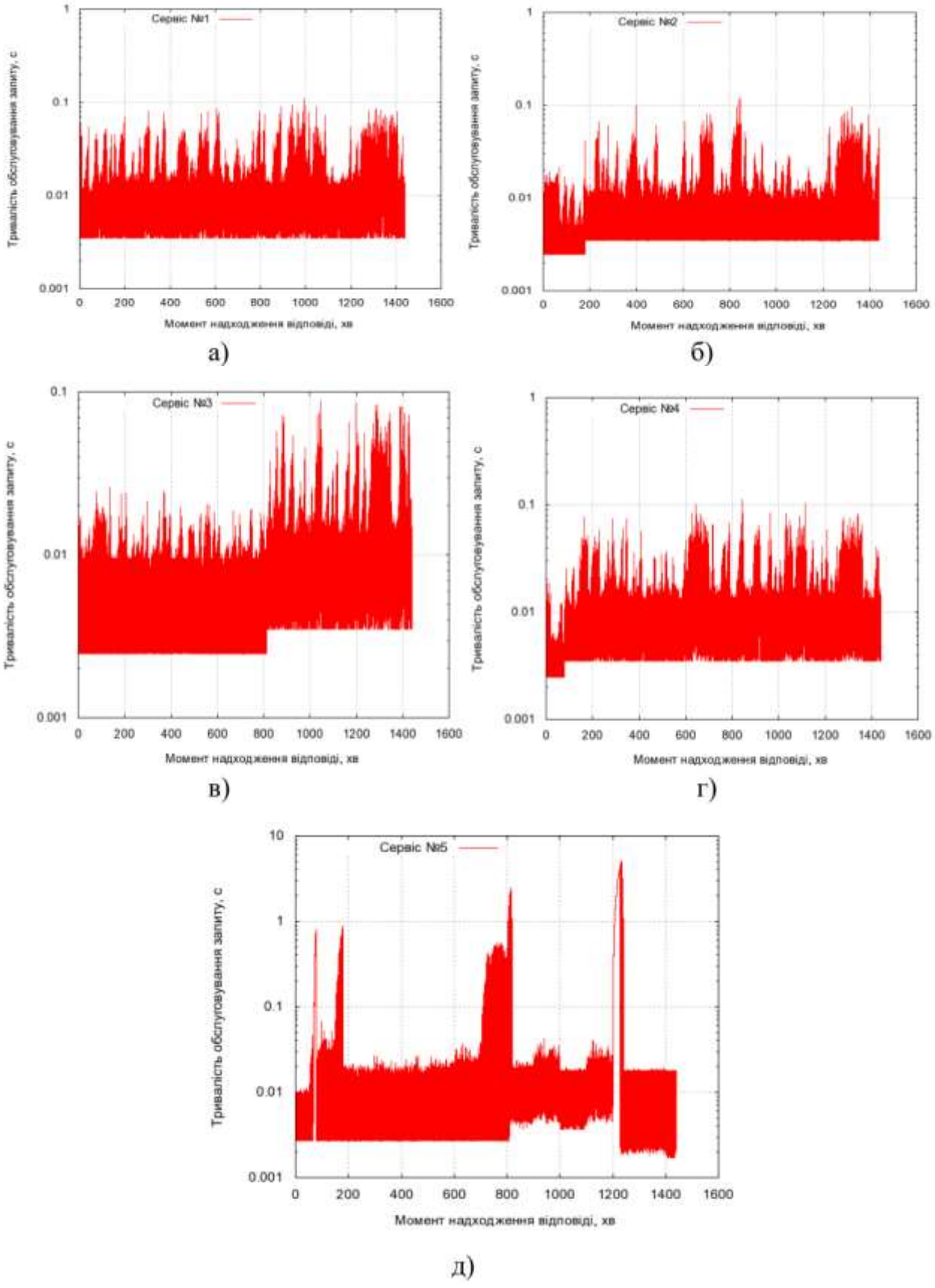


Рис. Б.6. Зміна тривалості надання: а) першого сервісу; б) другого сервісу; в) третього сервісу; г) четвертого сервісу; д) п'ятого сервісу після застосування запропонованих рішень

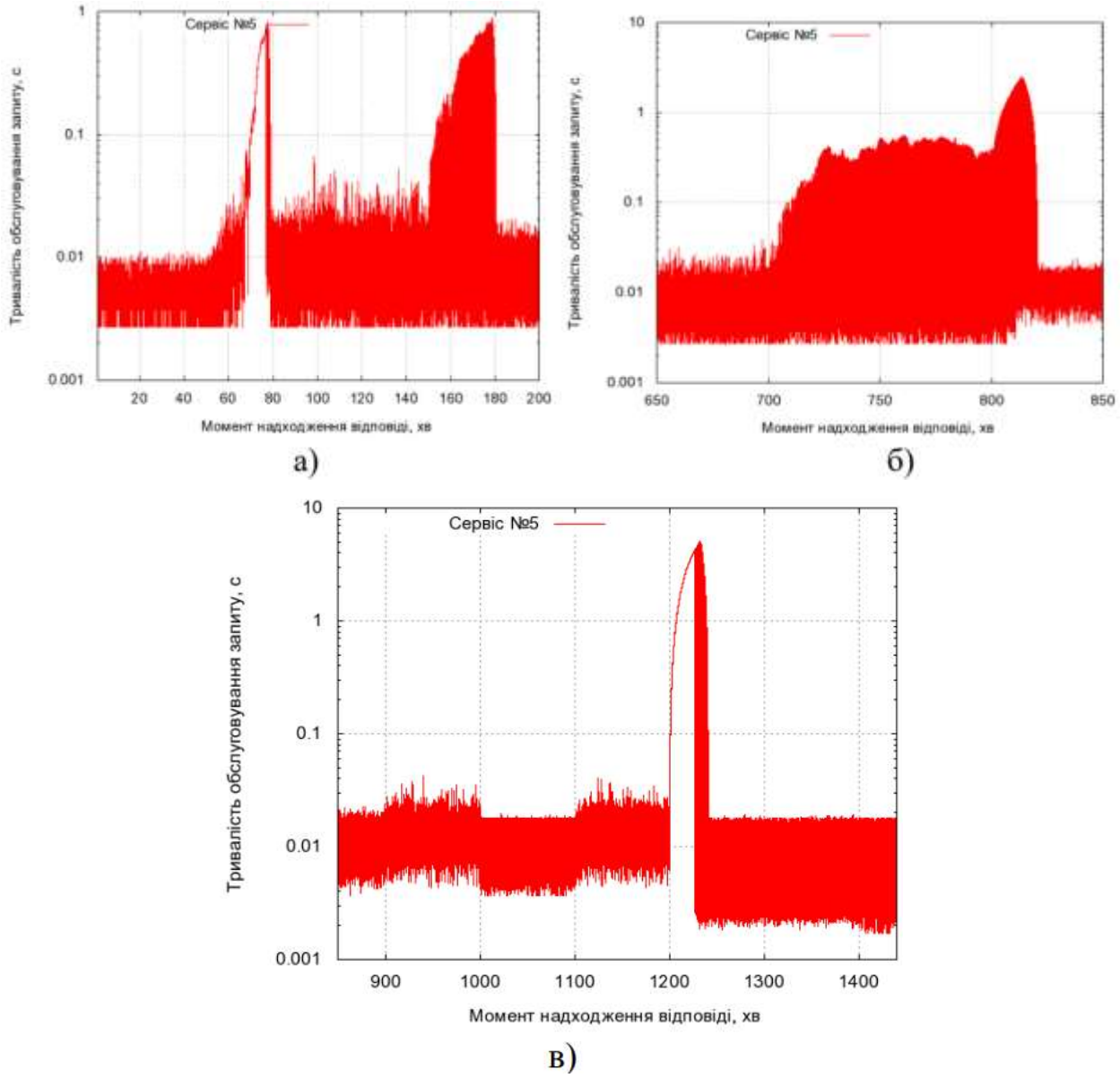


Рис. Б.7. Зміна тривалості надання п'ятого сервісу а) на 77 хв.; б) на 812 хв.; в) на 1221хв.