Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# АТЕСТАЦІЙНА РОБОТА

## Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель розподілу завдань по віртуальним серверам

(тема)

Виконав:
студент II курсу, групи СПзм-18-2
Башилов В.С.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: ст. викл. Мартовицький В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ __________ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри ____________
(підпис)
“____”______________ 20__ р.

# ЗАВДАННЯ

## НА АТЕСТАЦІЙНУ РОБОТУ

студентові Башилову Владиславу Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель розподілу завдань по віртуальним серверам

затверджена наказом по університету від “ 30 ” березня 2020 р. № 43 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2020 р.

3. Вхідні дані до роботи 1) об’єкт дослідження – центр обробки даних з хмарним обчислювальним середовищем; 2) стек протоколів: OpenStack; 3) мережна топологія: «Fat-Tree».

4. Перелік питань, що потрібно опрацювати в роботі
   1) принципи віртуалізації і побудови віртуального оточення
   2) підходи до оптимізації розподілу завдань по віртуальним серверам
   3) математична постановка задачи розподілу ресурсів
   4) формалізація завдання розподілу ресурсів
   5) експериментальні дослідження
   6) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (слайдів) слайд-презентація – 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім’я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Огляд принципів віртуалізації і побудови | 01.04.20-10.04.20 | |
| | віртуального оточення | | |
| 2 | Аналіз підходів до оптимізації розподілу | 11.04.20-17.04.20 | |
| | завдань | | |
| 3 | Математична постновка задачи розподілу | 18.04.20-24.04.20 | |
| 4 | Розробка моделі розподілу ресурсів та про- | 25.04.20-05.05.20 | |
| | ведення експериментів | | |
| 5 | Оформлення матеріалів атестаційної роботи | 06.05.20-11.05.20 | |
| 6 | Подання атестаційної роботи керівникові та | 12.05.20-13.05.20 | |
| | її попередній захист | | |
| 7 | Подання атестаційної роботи на рецензування | 14.05.20-15.05.20 | |
| | | | |

Дата видачі завдання 30 березня 2020 р.

Студент ____________________
(підпис)

Керівник роботи ____________________ ст. викл. Мартовицький В.О.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ


Пояснювальна записка атестаційної роботи: 82 с., 20 рис., 1 табл., 1 дод., 27 джерел.

АДМІНІСТРУВАННЯ, ВІРТУАЛІЗАЦІЯ, ВІРТУАЛЬНА МАШИНА, ІНФРАСТРУКТУРА ЯК СЕРВІС, МІГРАЦІЯ СЕРВЕРІВ, ОБЧИСЛЮВАЛЬНЕ НАВАНТАЖЕННЯ, РОЗПОДІЛ РЕСУРСІВ, ХМАРНІ ОБЧИСЛЕННЯ.

Метою атестаційної роботи є розробка моделі розподілу завдань по віртуальним серверам хмарного обчислювального середовища центру обробки даних.

У ході виконання атестаційної роботи обрано абстракції для опису ресурсних запитів і фізичних ресурсів центру обробки даних стосовно розподілу завдань між віртуальними серверами. Розроблено математичну модель центру обробки даних, в рамках якої сформульована математична постановка задачі розподілу ресурсів серверів, що допускає міграцію віртуальних машин і реплікацію елементів зберігання даних.

Наведено результати порівняння алгоритмів розподілу ресурсів. Запропонована триетапна методика, яка на відміну від відомої двоетапної дозволяє більш повно врахувати реальні процеси розподілу ресурсів ЦОД.

# ABSTRACT

Master's thesis: 82 pages, 20 figures, 1 tables, 1 appendix, 27 sources.

COMPUTER LOADING, INFRASTRUCTURE AS A SERVICE, RESOURCE DISTRIBUTION, SERVER MIGRATION, SYSTEM ADMINISTRATION, VIRTUAL MACHINE, VIRTUALIZATION.

The major goal of this thesis is to develop a model of task distribution on virtual servers in cloud computing environment of the data processing center.

During the attestation work, abstractions were selected to describe the resource requests and physical resources of the data processing center regarding the distribution of tasks between virtual servers. A mathematical model of the data processing center has been developed, within the framework of which a server resource allocation problem mathematical formulation has been formulated, which allows the virtual machines migration and data storage elements replication.

The results of resource allocation algorithms comparison are given. A three-stage technique is proposed, which, in contrast to the known two-stage method, allows to take more fully into account the real processes of data center resource allocation.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ВМ – віртуальна машина

ЗВМ – застосунок віртуальних машин

ІТ – інформаційні технології

МВМ – монітор віртуальних машин

ОЗП – оперативний запам’ятовуючий пристрій

ОС – операційна система

ПЗ – програмное забезпечення

ПК – персональний комп’ютер

ЦОД – центр обробки даних

ЦП – центральний процесор

IaaS – «інфраструктура як сервіс» (англ., Infrastructure as a Service)

PaaS – «платформа як сервіс» (англ., Plaatform as a Service)

SaaS – «програмне забезпечення як сервіс» (англ., Software as a Service)

SAN – мережа зберігання даних (англ., Storage Area Network)

SLA – угода про рівень надання послуги (англ., Service Level Agreement)

Гаряча точка – сервер у стані, коли використання будь-якого з його ресурсів перевищує граничне значення, попередньо задане адміністратором

Storage-елемент – елемент зберігання даних, який відображається на сховища даних ЦОД

Fat-Tree – топологія комп’ютерної мережі типу «потовщене дерево»

First Fit – алгоритм пакування у ємність типу «перший придатний за спаданням»

Hyper-V – система апаратної віртуалізації для x64-систем на основі гіпервізора

OpenStack – комплекс проектів вільного програмного забезпечення, призначених для використання при створенні інфраструктурних хмарних сервісів та хмарних сховищ, як публічних, так і приватних

Random Fit – алгоритм пакування у ємність типу «перший випадковий придатний»

# ВСТУП

Серверна віртуалізація – це архітектура програмного забезпечення (ПЗ), яке відповідає за те, щоб кілька операційних систем працювали на базі одного фізичного сервера [1]. При цьому ПЗ кожного сервера самодостатньо і відокремлено від будь-яких фізичних пристроїв. Це ж ПЗ сприймає доступні ресурси як ресурси одного фізичного сервера, хоча по факту отримує невеликий пул ресурсів. Віртуальні сервери працюють як імітація фізичного обчислювального обладнання.

У віртуальному сервері емулюється апаратне забезпечення: процесори, дискові накопичувачі, оперативна пам’ять. Операційні системи, встановлені на кожен віртуальний сервер, не «бачать» один одного і при цьому функціонують так, як якщо б вони були встановлені на звичайному комп’ютері. На одній апаратній платформі можна запускати кілька операційних систем (ОС) і в різних пропорціях розподіляти між ними фізичні ресурси.

З віртуалізацією нерозривно пов’язаний гіпервізор – програмне забезпечення, яке розгортається на сервері і безпосередньо взаємодіє з його фізичними ресурсами. Гіпервізор відповідає за те, щоб віртуальні машини «бачили» ці ресурси як власні.

Віртуалізація середовища підвищує гнучкість і адаптивність IT-інфраструктури організації, знижує витрати на її утримання, робить робочі навантаження мобільними, а ресурси – доступними. Наслідками віртуалізації серверів буде зростання автоматизації бізнес-процесів, поліпшення керованості і економічності інфраструктури, мінімізація аварійних і призначених для технічного обслуговування простоїв.

У ситуації з серверами віртуалізація вирішує відразу кілька важливих завдань [1]:

- дозволяє оптимізувати споживання обчислювальних ресурсів і ресурсів зберігання. До появи технології віртуалізації в дата-центрах накопичувалось багато обладнання, яке використовувалося неефективно. Поки одні машини працювали в половину (або менше) свого потенціалу, інші були перевантажені завданнями і часто зупинялися. Віртуалізація вирішила цю проблему, і тепер робочі навантаження рівномірно розподіляються між декількома машинами;

- стримує зростання кількості серверів. Можна зменшувати кількість серверів, необхідних в роботі різних завдань і застосунків, встановивши замість них одну машину і запустивши на ній потрібне число ОС (наприклад, сімейства Windows);

- знижує експлуатаційні витрати на утримання фізичного обладнання. Оскільки серверів менше, організація може заощадити на енергоспоживанні і кондиціонуванні приміщень (знизиться тепловиділення, тому можна буде використовувати менш потужні установки);

- спрощує міграцію даних. Під час передавання даних на інший сервер скорочується час виконання організаційних робіт: ІТ-фахівцю достатньо оновити драйвери на основній («хостовій») ОС, а гостьові продовжать працювати в колишньому режимі, так як не прив’язані до фізичного обладнання. Для користувачів ресурсів віртуальних машин таке переміщення залишиться непоміченим;

- підвищує продуктивність прикладного ПЗ. Якщо старі застосунки на 100% споживали фізичні ресурси однієї машини, зараз ця ситуація виключена. Працюючі віртуальні машини можуть автоматично переміщатися на менш навантажені сервери, щоб знизити навантаження на більш навантажені;

- підвищує доступність серверів і скорочує простої устаткування, оскільки для відновлення системи до початкового стану в разі збою потрібно менше часу. Віртуальні сервери підтримують технологію створення віртуальних знімків і спроможні робити резервне копіювання даних за заздалегідь складеним розкладом;

- спрощує роботу з віртуальним середовищем. Потрібно менше технічних фахівців, які займаються обслуговуванням системи. Адміністратори цінують віртуалізацію за те, що вона дозволяє дистанційно керувати віртуальними серверами незалежно від їх кількості та територіального розташування.

Мінус віртуалізації полягає в тому, що для переходу на цю технологію організації з високою ймовірністю доведеться купувати нове обладнання – з більш швидкими процесорами, збільшеним об’ємом дискового простору і оперативної пам’яті. Процесори в нових серверах повинні підтримувати технології віртуалізації, якщо використовуються системи на кшталт MS Hyper-V. Однак ці витрати окупаються протягом найближчих декількох років за умови, що компанія розвивається, а обсяги даних, якими вона оперує, безперервно зростають. До того ж, потужний сервер коштує дешевше, ніж кілька менш потужних машин.

Особливої уваги заслуговує організація роботи дата-центрів (ЦОД – центрів обробки даних) [2], де віртуалізація забезпечує високу гнучкість IT-інфраструктури і дозволяє їй краще відповідати потребам клієнтів і їх бізнесу. В умовах ЦОД критично важливим є оперативне визначення оптимального рішення з урахуванням поточних і прогнозованих значень стану системи. Необхідно уникнути невиправданих процедур перерозподілу ресурсів між завданнями та переміщень віртуальних машин, звести до мінімуму помилки вибору або ігнорування умов.

# 1 ПРИНЦИПИ ВІРТУАЛІЗАЦІЇ І ПОБУДОВИ ВІРТУАЛЬНОГО ОТОЧЕННЯ

## 1.1 Сутність віртуального оточення

Віртуалізацією в сфері інформаційних технологій (ІТ) називають процес ізоляції комп’ютерних ресурсів один від одного, що дозволяє зменшити залежності між ними. Це спрощує керування змінами в системі за рахунок їх локалізації в тому чи іншому шарі ізольованих за допомогою віртуалізації ресурсів. Технології віртуалізації зазвичай відносять до головних інновацій початку століття в сегменті стандартних серверів [1].

На думку більшості аналітиків, в середньостроковій перспективі саме віртуалізація буде перш за все впливати на зміни самої ІТ-інфраструктурі, характер її експлуатації та адміністрування. Результати дослідження розгортання серверів віртуалізації показали, що дана технологія чинить серйозний вплив на трансформацію мережевої інфраструктури, серверів та системи зберігання, а також на способи побудови центрів обробки даних (ЦОД) і керування ними. Різке зростання обсягів корпоративної інформації, що спостерігається останнім часом, пов’язане не тільки з розробкою нових бізнес-застосунків, але і з виникненням вкрай ресурсомістких напрямків. Серед них лідируючі позиції належать надвеликих баз даних в наукомістких галузях (генетиці, медицині, геології, астрономії та космічних дослідженнях) і системам обробки медіаконтенту (відео, анімація). Крім того, поява величезних архівів електронної пошти, об’ємних баз даних систем та інших подібних застосунків призвело до того, що загальний обсяг інформації, що зберігається корпоративної інформації оцінюється сьогодні в багато сотень петабайт і постійно збільшується.

В таких умовах застосування технології віртуалізації ресурсів зберігання – єдиний вихід, що дозволяє не тільки домогтися консолідації розрізнених «острівців даних», а й відображати практично необмежену кількість різнорідних носіїв, пристроїв пам’яті і систем зберігання у вигляді унітарного сховища даних (так званого віртуального пулу), керування яким здійснюється централізовано за допомогою єдиного вузла доступу.

У широкому сенсі, поняття віртуалізації означає приховування справжньої реалізації будь-якого процесу або об’єкта від справжнього його уявлення для того, хто ним користується. Продуктом віртуалізації є щось зручне для використання, має насправді більш складну або зовсім іншу структуру, відмінну від тієї, яка сприймається при роботі з об’єктом. Іншими словами, відбувається відділення уявлення від реалізації чого-небудь. У комп’ютерних технологіях під терміном «віртуалізація» зазвичай розуміється абстракція обчислювальних ресурсів і надання користувачеві системи, що «інкапсулює» (приховує в собі) власну реалізацію. Тобто, користувач працює зі зручним для себе поданням об’єкта, і для нього не має значення, як об’єкт влаштований в дійсності.

Отже, віртуалізація в обчисленнях – процес подання набору обчислювальних ресурсів, або їх логічного об’єднання, який дає певні переваги перед оригінальною конфігурацією. Сутність віртуалізації полягає в перетворенні апаратного забезпечення. Віртуальні машини використовують загальні апаратні ресурси, не заважаючи один одному, завдяки чому з’являється можливість без обмежень одночасно встановлювати кілька віртуальних систем і виконувати кілька застосунків на одному комп’ютері (рисунок 1.1).

Прикладом віртуалізації є симетричні мультипроцесорні комп’ютерні архітектури, які використовують більше одного процесора. Операційні системи зазвичай конфігуруються таким чином, щоб кілька процесорів представлялися як єдиний процесорний модуль. Ось чому програмні застосунки можуть бути написані для одного логічного (віртуального) обчислювального модуля, що значно простіше, ніж працювати з великою кількістю різних процесорних конфігурацій.


Рисунок 1.1 – Робота декількох віртуальних систем на одній фізичній

З точки зору користувача, віртуальна машина (ВМ) – це конкретний екземпляр якогось віртуального обчислювального середовища («віртуального комп’ютера»), створений за допомогою спеціального програмного інструменту. ВМ являє собою повністю ізольований програмний контейнер, здатний виконувати власну ОС і застосунки, як фізичний комп’ютер. Віртуальна машина працює абсолютно так само, як фізичний комп’ютер, і містить власні віртуальні (тобто програмні) центральний процесор (ЦП), оперативний запам’ятовуючий пристрій (ОЗП), жорсткий диск і мережеву інтерфейсну карту. Операційна система, застосунки та інші комп’ютери в мережі не здатні відрізнити віртуальну машину від фізичного комп’ютера. Крім того, сама віртуальна машина «вважає» себе матеріально існуючим комп’ютером. Проте, він складається виключно з програмного забезпечення й не містить апаратних компонентів.

## 1.2 Переваги віртуалізації

Основні переваги віртуальних машин перед фізичними обумовлени такими відмінними їх рисами [1, 2]:

- сумісність;
- ізольованість;
- інкапсуляція;
- незалежність обладнання.

Сумісність полягає в тому, що як і фізичний комп’ютер, віртуальна машина працює під керуванням власної гостьової операційної системи і виконує власні застосунки. Вона також містить всі компоненти, стандартні для фізичного комп’ютера (материнську плату, відеокарту, мережевий контролер тощо). Тому віртуальні машини повністю сумісні з усіма стандартними операційними системами, програмами та драйверами пристроїв на базі x86. Віртуальну машину можна використовувати для виконання будь-якого програмного забезпечення, придатного для фізичного комп’ютера на базі x86.

Віртуальні машини можуть використовувати загальні фізичні ресурси одного комп’ютера і при цьому залишатися повністю ізольованими один від одного, якби вони були окремими фізичними машинами. Наприклад, якщо на одному фізичному сервері запущено чотири віртуальних машини, і одна з них дає збій, це не впливає на доступність інших трьох машин. Ізольованість – важлива причина набагато більш високої доступності та безпеки застосунків, які виконуються в віртуальному середовищі, в порівнянні із застосунками, виконуваними в стандартній, невіртуалізованій системі.

Віртуальна машина являє собою програмний контейнер, що зв’язує, або «інкапсулює» повний комплект віртуальних апаратних ресурсів, а також ОС і все її застосування в програмному пакеті. Завдяки інкапсуляції віртуальні машини стають неймовірно мобільними і зручними в керуванні. Наприклад, віртуальну машину можна перемістити або скопіювати з одного пункту до іншого так само, як будь-який інший програмний файл. Крім того, віртуальну машину можна зберегти на будь-якому стандартному носії даних: від компактної карти Flash-пам’яті USB до корпоративних мереж зберігання даних (SAN).

Також важливим є той факт, що віртуальні машини повністю незалежні від базового фізичного обладнання, на якому вони працюють. Наприклад, для віртуальної машини з віртуальними компонентами (ЦП, мережевою картою, контролером SCSI) можна задати налаштування, які абсолютно не збігаються з фізичними характеристиками базового апаратного забезпечення. Віртуальні машини можуть виконувати різні операційні системи (Windows, Linux і ін.) на одному і тому ж фізичному сервері.

## 1.3 Керування віртуальним оточенням

Інструменти для створення ВМ дозволяють організовувати і запускати довільне число віртуальних машин, яке обмежується лише фізичними ресурсами реального ПК. Власне інструмент для створення ВМ (застосунок віртуальних машин, ЗВМ) – це звичайна програма, що встановлюється, як і будь-яка інша, на конкретну реальну операційну систему. Ця реальна ОС іменується «хазяйською», або хостовою, ОС (від англ. host – «головний», «базовий», «провідний») [3].

Всі завдання з керування віртуальними машинами вирішує спеціальний модуль в складі застосунка ВМ – монітор віртуальних машин (МВМ). Монітор грає роль посередника у всіх взаємодіях між віртуальними машинами і базовим обладнанням, підтримуючи виконання всіх створених ВМ на єдиній апаратній платформі і забезпечуючи їх надійну ізоляцію.

Користувач не має безпосереднього доступу до МВМ. У більшості програмних продуктів йому надається лише графічний інтерфейс для створення і налаштування віртуальних машин. Коли віртуальна машина створена і запущена, у користувача може виникнути повна ілюзія того, що він працює з автономним комп’ютером, що має власні процесор, оперативну пам’ять, відеосистему і (як правило) «стандартний» набір зовнішніх пристроїв, включаючи флопі-дисковод і пристрій читання CD/DVD.

Насправді, віртуальна машина не має доступу до фізичних ресурсів реального комп’ютера. Робота з ними покладена на МВМ. У спрощеному вигляді архітектура системи, в якій використовуються віртуальні машини, виглядає як такі взаємодії:

- хостова ОС і монітор віртуальних машин поділяють між собою права на керування апаратними компонентами комп’ютера; при цьому хостова ОС займається розподілом ресурсів між власними застосунками (включаючи і консоль ВМ);
- монітор ВМ контролює розподіл ресурсів між запущеними віртуальними машинами, створюючи для них ілюзію безпосереднього доступу до апаратного рівня (цей механізм називається саме віртуалізацією);
- гостьові ОС в межах виділених їм ресурсів керують роботою застосунків, запущених на ній.

Таким чином, кінцевий користувач ВМ отримує в своє розпорядження фізичні ресурси ПК в якості сервісу (послуги). Характеристиками і параметрами послуги виступають обсяг наданого ОЗП, обсяг виділеного дискового простору, частка продуктивності процесора і набір застосункових пристроїв.

Віртуалізація надає безліч переваг, як для інфраструктури підприємств, так і для кінцевих користувачів. За рахунок віртуалізації забезпечується істотна економія на апаратному забезпеченні, облуговуванні, підвищується гнучкість IT-інфраструктури, спрощується процедура резервного копіювання та відновлення після збоїв. Віртуальні машини, будучи незалежними від конкретного обладнання одиницями, можуть поширюватися в якості встановлених шаблонів, які можуть бути запущені на будь-якій апаратній платформі підтримуваної архітектури.

## 1.4 Технологічні оосбливості застосування віртуалізації

ВМ можуть бути основними компонентами набагато більш масштабного рішення – віртуальної інфраструктури. На відміну від ВМ, які використовують апаратні ресурси одного комп'ютера, віртуальна інфраструктура – об'єднані апаратні ресурси всієї IT-інфраструктури, включаючи комп'ютери, мережеві пристрої і об'єднані сховища (рисунок 1.2) [3]. Віртуальна інфраструктура являє собою динамічний розподіл фізичних ресурсів відповідно до потреб підприємства.

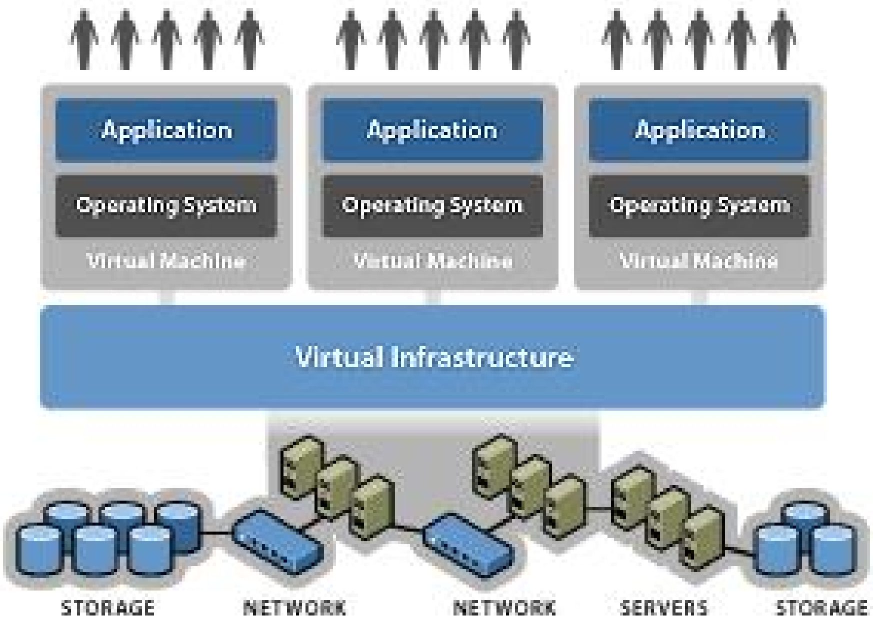


Рисунок 1.2 – Схематичне представлення віртуальної інфраструктури

Відокремлюючи програмне середовище від вихідної апаратної інфраструктури, технологія віртуалізації дозволяє об'єднати ряд серверів, інфраструктур зберігання і мереж в єдиний пул ресурсів, який динамічно, безпечно і надійно розподіляється між застосунками в міру необхідності. За допомогою цього якісно нового підходу до організації підприємства отримують можливість створити обчислювальну інфраструктуру з максимальною ефективністю, доступністю, автоматизацією і гнучкістю, що складається з недорогих серверів, які відповідають галузевим стандартам.

При описі віртуалізації платформ поняття віртуалізації розглядалося у вузькому сенсі, переважно стосовно процесу створення віртуальних машин. Однак, якщо розглядати віртуалізацію в широкому сенсі, можна прийти до поняття віртуалізації ресурсів, узагальнюючим в собі підходи до створення віртуальних систем. Віртуалізація ресурсів дозволяє концентрувати, абстрагувати і спрощувати керування групами ресурсів, таких як мережі, сховища даних і простору імен. Основними видами віртуалізації ресурсів є такі:

- об’єднання;
- агрегація;
- концентрація.

Під об’єднанням мається на увазі організація декількох фізичних або логічних об’єктів в пули ресурсів (групи), що представляють зручні інтерфейси користувача. Приклади такого виду віртуалізації:

- багатопроцесорні системи, що представляються одна потужна система;
- RAID-масиви і засоби керування томами, що комбінують декілька фізичних дисків в один логічний;
- віртуалізація систем зберігання, використовувана при побудові мереж зберігання даних SAN (Storage Area Network);
- віртуальні приватні мережі (VPN) і трансляція мережевих адрес (NAT), що дозволяють створювати віртуальні простори мережевих адрес і імен;
- кластеризація комп’ютерів і розподілені обчислення (grid computing).

При поділі ресурсів (англ. – partitioning) в процесі віртуалізації відбувається поділ будь-якого одного великого ресурсу на кілька однотипних об’єктів, зручних для використання. В мережах зберігання даних це називається зонуванням ресурсів («zoning»).

Віртуалізація ресурсів, на відміну від віртуалізації платформ, має ширший і розпливчастий сенс і являє собою масу різних підходів, спрямованих на підвищення зручності кількість звернень від користувачів з системами в цілому.

В даний момент програми, що працюють на серверах в ІТ-інфраструктурі компаній, створюють невелике навантаження на апаратні ресурси серверів (в середньому 5-15 відсотків). Віртуалізація дозволяє мігрувати з цих фізичних серверів на віртуальні і розмістити їх все на одному фізичному сервері, збільшивши його завантаження до 60-80 відсотків і підвищивши тим самим коефіцієнт використання апаратури, що дозволяє істотно заощадити на апаратурі, обслуговуванні і електроенергії.

На сьогоднішній день проекти по віртуалізації ІТ-інфраструктури активно впроваджуються багатьма провідними компаніями, що займаються системною інтеграцією і є авторизованими партнерами провайдерів систем віртуалізації. В процесі віртуалізації ІТ-інфраструктури створюється віртуальна інфраструктура – комплекс систем на основі віртуальних машин, що забезпечують функціонування всієї ІТ-інфраструктури, що володіє багатьма новими можливостями при збереженні існуючої схеми діяльності ІТ-ресурсів. Вендори різних платформ віртуалізації готові надати інформацію про успішні проекти з впровадження віртуальної інфраструктури в великих банках, промислових компаніях, лікарнях, освітніх установах. Безліч переваг віртуалізації операційних систем дозволяють компаніям економити на обслуговуванні, персоналі, апаратне забезпечення, забезпеченні безперебійної роботи, реплікації даних і відновлення після збоїв. Також ринок віртуалізації починає наповнюватися потужними засобами керування, міграції та підтримки віртуальних інфраструктур, що дозволяють використовувати переваги віртуалізації найбільш повно.

Серед проблемних моментів, пов’язаних з віртуалізацією, слід зазначити неможливість емуляції всіх пристроїв та особливі вимоги до застосункових апаратних ресурсів.

В даний момент всі основні пристрої апаратних платформ підтримуються вендорами систем віртуалізації, однак якщо потрібно, наприклад, віртуалізувати будь-які контролери або пристрої, які не підтримуються ними, доведеться відмовитися від віртуалізації такого оточення.

Використання різних технік віртуалізації дозволило наблизити показники швидкодії віртуальних машин до реальних, проте для того, щоб фізичний хост зміг запускати хоча б пару віртуальних машин, потрібна достатня для них кількість апаратних ресурсів. Деякі платформи віртуалізації вимогливі до конкретного апаратного забезпечення. Наприклад, дуже якісна платформа компанії VMware, ESX Server, була б поза конкуренцією, якби не висувала жорстких вимог до апаратного забезпечення [3].

Незважаючи на перелічені та цілком переборні недоліки, віртуалізація продовжує набирати обертів і у найближчому майбутньому очікується істотне розширення, як ринку платформ віртуалізації, так і засобів керування віртуальними інфраструктурами.

## 1.5 Віртуалізація як підґрунтя хмарних обчислень

«Хмарні» обчислення є однією з ключових концепцій, що дозволяє вивести (дематеріалізувати) IT-активи компаній з основного фонду. Володіння IT-ресурсами передбачає зазвичай їх недостатню експлуатацію і зростаючу вартість обслуговування. Стверджується, що було б розумним перевести IT-активи в категорію «на вимогу», виключивши їх з капітальних витрат [2, 3].

Хмара – це пул абстрактних і керованих обчислювальних інфраструктур, які базуються на технології віртуалізації, з високим ступенем масштабованості, здатних надавати послуги з розміщення та виконання програм кінцевих клієнтів, оплачувані по фактичному споживанню. Хмара виступає метафорою віддаленого обчислювального ЦОД, до якого надається доступ на основі оплати pay-as-you-go (оплата за фактичне використання сервісу обчислення). Таким чином, програмне забезпечення фактично надається користувачеві як сервіс. Користувачеві хмарних обчислень не потрібно піклуватися ні про інфраструктуру, ні про фактичне програмне забезпечення, «хмара» успішно приховує всі технічні та програмні деталі.

Поява першої технології, близькою до сучасного розуміння терміна «cloud computing», приписується компанії Salesforce, заснованій у 1999 році. Саме тоді і з'явилася перша пропозиція нового виду продукту «Програмне забезпечення як сервіс» (англ. – Software as a Service, SaaS). Певний успіх Salesforce в цій області сприяв появі інтересу у великих компаній IT-індустрії, які спішно повідомили про свої дослідження в області хмарних технологій. Перше бізнес-рішення під назвою «Amazon Web Services» було запущено в 2005 році компанією Amazon, яка з часів кризи, активно займалася модернізацією своїх ЦОД. Наступним свою технологію поступово ввела компанія Google, почавши з 2006 року пропонувати SaaS-сервіси під назвою «Google Apps», а потім і моделі надання платформи як сервісу (PaaS) під назвою «Google App Engine». І, нарешті, свою пропозицію анонсувала компанія Microsoft, презентувавши її на конференції PDC 2008 під назвою «Azure Services Platform». Висока зацікавленість найбільших гравців ринку IT демонструє певний статус хмарних обчислень як сучасного тренда.

На даний момент більшість хмарних інфраструктур розгорнуто на серверах ЦОД, використовуючи технології віртуалізації, що фактично дозволяє будь-якому призначеному для користувача застосунку використовувати обчислювальні потужності, абсолютно не замислюючись про технологічні аспекти. В такому випадку «хмару» можна розуміти як єдиний доступ до обчислень з боку користувача. З поняттям хмарних обчислень часто пов'язують такі технології надання сервісу (Everything as a service), як «Програмне забезпечення як сервіс» (англ. – Software as a Service, SaaS), «Інфраструктура як сервіс» (англ. – Infrastructure as a Service, IaaS) і «Платформа як сервіс» (англ. – Plaatform as a Service, PaaS).

SaaS – модель розгортання програми, яка має на увазі надання застосунку кінцевому користувачеві як послуги на вимогу (on demand). Доступ до такого застосунку здійснюється за допомогою мережі, а найчастіше за допомогою Інтернет-браузера.

IaaS – модель надання комп’ютерної інфраструктури як сервісу. Замість покупки серверів, ПЗ, спеціального мережевого устаткування, користувач може отримати ці ресурси у вигляді аутсорсингу (outsource). Фактично, як зауважують багато експертів, це всього лише деяка еволюція сервісів хостингу.

PaaS – модель мережевого надання обчислювальної платформи як сервісу, яка пропонує розгортання і підтримку веб-застосунків і сервісів без необхідності покупки і управління шарами апаратного і програмного забезпечення.

Таким чином, ці технології при спільному використанні дозволяють користувачам хмарних обчислень скористатися обчислювальними потужностями і сховищами даних, які за допомогою певних технологій віртуалізації і високого рівня абстракції надаються їм як послуги.

## 1.6 Постановка задачи дослідження

Хмарні ЦОД повинні надавати зручний мережевий доступ до загального пулу обчислювальних ресурсів, керування якими повинно здійснюватися з мінімальними зусиллями з адміністрування або втручанням сервіс-провайдера. Гнучкість в керуванні досягається за рахунок використання платформ віртуалізації, які дозволяють створювати віртуальні машини на фізичних серверах, також їх переміщати і знищувати. При цьому потрібно враховувати ряд проблем, що виникають в сучасних ЦОД, які можна частково вирішити шляхом переміщення віртуальних машин між фізичними хостами. До таких проблем відносяться: велике енергоспоживання; нерівномірність тепловиділення; необхідність виконання SLA-угод тощо.

В роботі потрібно опрацювати питання методики оптимального розміщення віртуальних серверів ЦОД, яка на відміну від відомої двоетапної методики дозволяє більш повно врахувати реальні процеси розподілу ресурсів ЦОД.

Дана методика має включати у себе три етапи: початкове, динамічне і статичне розміщення віртуальних серверів.

На етапі первинного розміщення вирішується завдання багатокритеріальної оптимізації закріплення хмарних серверів по фізичним серверам в незавантаженому ЦОД, з урахуванням тільки вимог хмарних серверів, місткості фізичних машин, платформних обмежень, а також електроживлення та охолодження.

Під динамічним розміщенням розуміється перерозподіл хмарних серверів між фізичними серверами відповідно до зміни стану системи. При цьому контролюються такі параметри системи, як енергоспоживання, температура процесора, нестача обчислювальних ресурсів для віртуальних машин. При виході параметрів за допустимі межі приймається рішення про переміщення віртуального сервера на фізичний.

Третій етап методики має місце тоді, коли система виходить за межі зміни параметрів, розрахованих на першому етапі. Для отримання рішення за прийнятний час запропоновано використовувати комбінацію методів послідовних поступок і обмежень.

Відповідно до цього, потрібно вирішити ряд завдань:

- обрати абстракції для опису ресурсних запитів і фізичних ресурсів центрів обробки даних;
- розробити математичну модель центру обробки даних, що дозволить описувати широкий клас їх архітектур;
- сформулювати математичну постановку задачі розподілу ресурсів, яка допускає міграцію віртуальних машин і реплікацію елементів зберігання даних;
- розробити підхід розподілу навантаження в хмарному

обчислювальному середовищі, який ґрунтується на аналізі рівномірності використання ресурсів.

# 2 ПІДХОДИ ДО ОПТИМІЗАЦІЇ РОЗПОДІЛУ ЗАВДАНЬ ПО ВІРТУАЛЬНИМ СЕРВЕРАМ У ХМАРНОМУ ОБЧИСЛЮВАЛЬНОМУ СЕРЕДОВИЩІ

## 2.1 Загальні відомості

Хмарні ЦОД дозволяють користувачам орендувати у провайдера обчислювальні ресурси на вимогу і оплачувати їх фактичне споживання. За визначенням Національного інституту стандартів і технологій США «хмарні обчислення є моделлю надання зручного мережевого доступу до загального пулу обчислювальних ресурсів, які можуть бути швидко виділені і звільнені з мінімальними зусиллями з адміністрування або втручанням сервіс-провайдера [3]. Таке гнучке керування ресурсами стало можливим завдяки технологіям віртуалізації, що дозволяє розміщувати кілька віртуальних машин на одній фізичній машині. Крім цього віртуалізація дозволяє здійснювати «живу міграцію» віртуальних машин, тобто переміщати віртуальну машину з одного фізичного сервера на інший без її відключення. Крім цього жива міграція допомагає виконувати SLA-угоди, збалансувати навантаження між фізичними серверами ЦОД, а також розміщувати віртуальні машини на меншій кількості фізичних серверів. Технології віртуалізації реалізовані в гіпервізорах, таких як VMware ESX, Microsoft Hyper-V або Xen, які дозволяють створювати, знищувати і переміщати віртуальні машини. У зв’язку з цим виникає завдання розміщення віртуальних машин на фізичних серверах. При цьому потрібно враховувати ряд проблем, що виникають в сучасних ЦОД, які можна частково вирішити, шляхом переміщення віртуальних машин.

Велике енергоспоживання. Відомо, що в неактивному стані увімкнений сервер споживає значну кількість електроенергії [4]. Тому в такій ситуації для економії електроенергії необхідно переміщати все віртуальні машини з незавантаженого сервера на інші з вимкненням останнього.

Нерівномірність тепловиділення. Відомо також, що температура кожного сервера повинна бути збережена нижче порогового значення, оскільки перегрів компонентів може призвести до апаратних збоїв. Крім цього нерівномірне тепловиділення серверів в ЦОД призводить до появи «гарячих точок», що значно збільшує витрати на охолодження [4]. При виявленні локальних гарячих точок віртуальні машини з високими навантаженнями переміщують для полегшення ситуації і зменшення витрат на охолодження.

Необхідність виконання SLA-угод. Зі збільшенням навантаження на віртуальні машини, розміщені на одному сервері, часто виникає ситуація, при якій кілька віртуальних серверів конкурують за обчислювальні ресурси, що негативно відбивається на показниках якості роботи застосунків. Переміщення віртуальних машин з проблемного хоста дозволяє їм отримати необхідні ресурси для роботи і збільшити показники якості.

Нерівномірність використання ресурсів. Для підвищення коефіцієнта використання фізичних серверів розміщують віртуальні машини таким чином, щоб мінімізувати залишок невикористаних ресурсів.

В літературі по розміщенню віртуальних машин розрізняють три різних режими роботи з користувачами: бронювання, доступ на вимогу і наявний ринок (spot market) [5]. У даній роботі буде розглянуто тільки доступ на вимогу як найбільш поширений. В такому доступі існує два типи розміщення віртуальних машин: початкове розміщення та міграція віртуальних машин [3, 6]. Незважаючи на те, що в деяких випадках можуть застосовуватися схожі алгоритми, початкове розміщення та міграція, як правило, розглядаються як окремі теми. В існуючих роботах не розглянуто взаємодію між первинним і динамічним розміщенням, особливо в ситуації, коли необхідно враховувати безліч суперечливих критеріїв. Тому дана робота присвячена розробці методики багатокритеріального розміщення віртуальних машин в хмарному ЦОД.

Розглянемо традиційну дворівневу систему керування хмарним ЦОД

[3], що складається з локальних і глобального менеджерів, представлених на рисунку 2.1. Локальний менеджер розміщений на кожному фізичному вузлі і є модулем монітора віртуальних машин. Його завданням є постійний моніторинг завантаження процесора вузла і зміна розмірів віртуальних машин відповідно до їх потреби в ресурсах. Система моніторингу ЦОД вимірює системну інформацію, включаючи використання ресурсів, енергоспоживання і тепловиділення, збираючи їх в центральне сховище. Глобальний менеджер аналізує дану інформацію і відповідно до неї оптимізує розміщення віртуальних серверів.

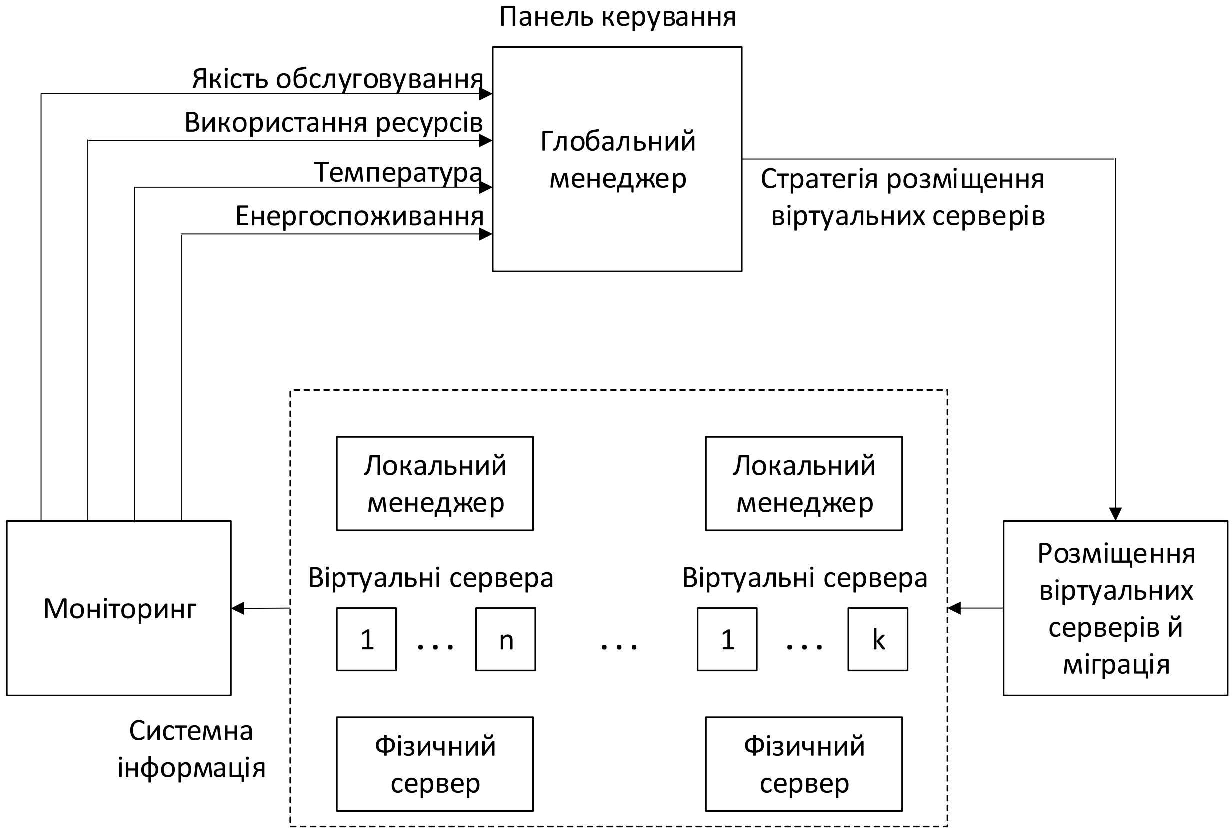


Рисунок 2.1 – Дворівнева система керування хмарним ЦОД

Запити на ресурси від користувачів виражаються в замовленні віртуального сервера з певним набором ресурсів, і кожний користувальницький застосунок запущений на своєму віртуальному сервері. Ресурси на фізичних серверах поділені на частини (слайси) за кількома вимірюваннями відповідно до запитів на ресурси віртуальних серверів. Кількість споживаних кожним віртуальним сервером ресурсів обмежено розміром віртуального сервера, який можна уявити як k-розмірний вектор, в якому кожна компонента відповідає одному з ресурсів.

## 2.2 Методика призначення віртуальних серверів ЦОД

При розподілі віртуальних машин в ЦОД необхідно відповісти на наступні питання.

Питання 1. Яким чином розподілити ресурси, з огляду на велику кількість фізичних і віртуальних серверів?

Питання 2. Як знайти оптимальне рішення, з огляду на кілька суперечливих цілей?

Питання 3. Коли потрібно здійснювати розміщення або переміщення віртуальних серверів?

Дану проблему пропонується вирішувати в три етапи: початкове, динамічне і статичне розміщення віртуальних серверів.

При первинному розміщенні вирішується завдання закріплення хмарних серверів по фізичним серверам в незавантаженому ЦОД, з урахуванням тільки вимог хмарних серверів, місткості фізичних машин, платформних обмежень, а також електроживлення та охолодження.

Під динамічним розміщенням розуміється перерозподіл хмарних серверів між фізичними серверами відповідно до зміни стану системи або вимог на зміну ресурсів хмарних серверів від замовників. Динамічне розміщення повинно проводитись швидко з мінімальним числом міграцій. Статичне розміщення передбачає врахування планових змін в системі розподілу ресурсів ЦОД, воно має довгостроковий ефект, проводиться набагато рідше, ніж динамічне розміщення, а також вимагає значних змін в поточному плані розміщення, тобто велике число міграцій віртуальних серверів. Статичне розміщення також проводиться в разі, якщо динамічний не дає бажаного ефекту. Розглянемо ці етапи більш детально.

### 2.2.1 Первинне розміщення

При первинному розміщенні вирішується завдання закріплення хмарних серверів по фізичним серверам в незавантаженому ЦОД. Постановка завдання для початкового розміщення приведена в [7]. Під оптимальним розміщенням хмарних серверів розуміється таке закріплення їх за фізичними серверами, при якому мінімізуються енергоспоживання, невикористані ресурси, нерівномірність тепловиділення і порушення SLA-угод. Критерії, отримані в ряді робіт експериментально, мають такий вигляд [3, 5]:

- енергоспоживання

$$f_{power}(u_{CPU}) = \begin{cases} p_0 + (p_1 - p_0)u_{CPU}, & u_{CPU} > 0; \\ 0, & u_{CPU} = 0, \end{cases}$$

де $p_0$ – споживання електроенергії незавантажені сервером; $p_1$ – споживання електроенергії цілком завантаженим сервером;

- невикористані ресурси

$$f_{resource}(u_{CPU}, u_{RAM}) = 1 - u_{CPU} \cdot u_{RAM},$$

де $u_{CPU}$ – завантаження процесора; $u_{RAM}$ – завантаження пам’яті.

Цей критерій відображає наскільки повно завантажені ресурси різних типів;

- тепловиділення

$$f_1(T) = 1 - \frac{1}{1 + e^{(T - T_s)}},$$

де T – поточна температура процесора; $T_s$ – безпечна температура;

- порушення SLA-угод

$$f_{SLA}(u_{CPU}) = 1 - \frac{1}{1 + e^{u_{CPU} - 0,9}},$$

де $u_{CPU}$ – завантаження процесора, максимальне значення якого приймається рівним 0,9.

Для вирішення цієї багатокритеріальної задачі пропонується використовувати метод послідовних поступок. Для цього критерії розташовують і нумерують в порядку зменшення важливості. Важливість критеріїв можна визначити виходячи з того, як часто кожен із критеріїв згадувався в літературі. Таким чином, критерії слід розташувати в такий спосіб: енергоспоживання, нерівномірність тепловиділення, порушення SLA-угод і невикористані ресурси. Рішення задачі оптимізації проводиться спочатку по найважливішому критерію. На наступному кроці шукається рішення найкраще для наступного за важливістю критерія, а по найважливішому критерію вводиться поступка, яка характеризує допустиме відхилення цього критерію від його мінімального значення. На третьому кроці шукається рішення по третьому критерію, при заданих поступках по першому і другому і так до тих пір, поки не буде розглянуто останній за важливістю критерій. Отримане в результаті рішення є оптимальним за Парето.

### 2.2.2 Динамічне розміщення

Після того, як віртуальні машини були розподілені, необхідна корекція їх розміщення у зв’язку зі змінами навантажень, кількості ресурсів ВМ і запитів на додавання чи видалення ВМ. Робота з керування ресурсами лягає на глобальний менеджер (рисунок 2.1), який приймає рішення щодо міграції віртуальних серверів і вмикання чи вимикання фізичних серверів відповідно до даних системи моніторингу. Ці дані можуть відноситися до рівня фізичних ресурсів, таких як температура, енергоспоживання, завантаження процесора, пам'яті тощо. На рівні віртуальних ресурсів можуть використовуватися показники завантаження ресурсів віртуальних машин. На рівні застосунків можуть збиратися показники якості обслуговування, такі як середній час відгуку, максимальний час відгуку для заданої частки запитів, пропускна здатність тощо. У зв'язку з різномаїттям застосунків, що розміщуються на віртуальних серверах у ЦОД, важко виокремити загальні показники продуктивності рівня застосунків. Тому обмежимося такими показниками, одержуваними з рівня віртуалізації і рівня системи моніторингу ЦОД, як температура процесора, енергоспоживання і завантаження ресурсів віртуальних машин.

Глобальний менеджер повинен виконувати три основні функції:

- відслідковувати умови, при яких необхідно вжити заходів щодо міграції ВМ і вмикання/вимикання фізичних серверів;
- вибирати віртуальні машини для міграції;
- вибирати вузли призначення.

Динамічне розміщення віртуальних серверів найчастіше необхідним є в наступних трьох випадках; підвищення температури процесора; низька енергоефективність; боротьба за ресурси серед віртуальних машин. Для цього контролер періодично перевіряє три згадані вище параметри з використанням даних датчиків, і якщо вони перевищують допустимий рівень, то контролер активізується для того, щоб виконати дії щодо можливої міграції віртуальних машин. При цьому необхідно враховувати, що ці показники можуть короткочасно коливатися у зв'язку з динамічною зміною навантаження на хмарні застосунки.

На глобальний контролер покладено функцію вибору віртуальних машин, що визначає, які віртуальні машини повинні бути перенесені з урахуванням плюсів і мінусів можливої міграції. У разі підвищення температури процесора сервера слід вибрати віртуальні машини з високим завантаженням процесора щоб ефективніше знизити його температуру. У той саме час, міграція ВМ також вимагає процесорного часу і операцій введення-виведення, що, в свою чергу, збільшує споживання енергії і температуру сервера. Процес динамічної міграції віртуальної машини полягає в копіюванні файлу пам’яті і образу на хост призначення, а також відстеження, які сторінки пам’яті були модифіковані. Тому для міграції вибирається віртуальний сервер з меншим об’ємом пам’яті і інтенсивністю модифікацій сторінок, що може скоротити час і накладні витрати на міграцію.

У разі конкуренції за ресурси алгоритму вибору віртуальних машин потрібно визначити, які віртуальні сервери конкурують за ресурси. Також враховуються накладні витрати на міграцію: конкуруючі віртуальні машини упорядковуються за зростанням їх обсягу пам’яті, а потім віртуальні машини з пам’яттю меншого обсягу вибираються для міграції.

Низька енергоефективність розпізнається, коли контролер виявляє сервер, що простоює, або сервер з низьким завантаженням ресурсів; тоді всі віртуальні машини, що працюють на сервері, повинні бути переміщені, а сам сервер вимкнений.

Для того, щоб визначити новий хост призначення для обраної віртуальної машини, у функції вибору хоста обчислюються наступні параметри, які також можна розрахувати, використовуючи критеріальні функції першого етапу: температура сервера; енергоспоживання; продуктивність.

Розміщення віртуальної машини на «найхолоднішому» сервері може допомогти збалансувати розподіл температури в ЦОД. Використовуючи модель температури, отриману в результаті обробки системних даних моніторингу, функція вибору може спрогнозувати температуру хоста призначення після міграції ВМ.

З точки зору економії енергії, переважно поміщати віртуальну машину на найбільш завантажений сервер, який ще має достатньо ресурсів. Алгоритм вибору хоста може використовувати моделі енергоспоживання для прогнозування майбутнього енергоспоживання кандидатів в хости призначення.

Щоб уникнути виникнення нового конфлікту за ресурси і гарантувати продуктивність кожної ВМ, перевага віддається хостам, які мають найбільшу кількість вільних ресурсів, достатніх для розміщення мігруючих віртуальних машин.

Результати вибору сервера за різними критеріями можуть відрізнятися один від одного. Для вирішення цього завдання пропонується використовувати узагальнений критерій у вигляді адитивної згортки.

### 2.2.3 Статичне розміщення

Необхідність в цьому етапі виникає кожен раз, коли система виходить за межі зміни параметрів, розрахованих на першому етапі. При цьому вирішується завдання першого етапу. У цьому випадку метод послідовних поступок, що вимагає довільного вибору величин пробних поступок, число яких може бути великим, призводить в ряді завдань до значних часових витратах. Тому для вирішення подібних багатокритеріальних задач найчастіше використовуються методи формування узагальненого критерію, які в даному випадку не дозволять в повній мірі врахувати вимоги хмарних сервісів і характер використовуваних критеріальних функцій. Тому в даній роботі запропонований інший підхід з використанням таких методів, як метод послідовних поступок і обмежень, і їх комбінації. Для цього системним адміністратором задається вектор переваг приватних критеріїв, на підставі якого визначається розмір компромісної поступки за менш важливими критеріями, що дозволяє знайти Парето-оптимальне компромісне рішення, що дає мінімальні відносні відхилення критеріїв від своїх оптимальних значень [7]. Узагальнена схема пропонованої методики наведена на рисунку 2.2.

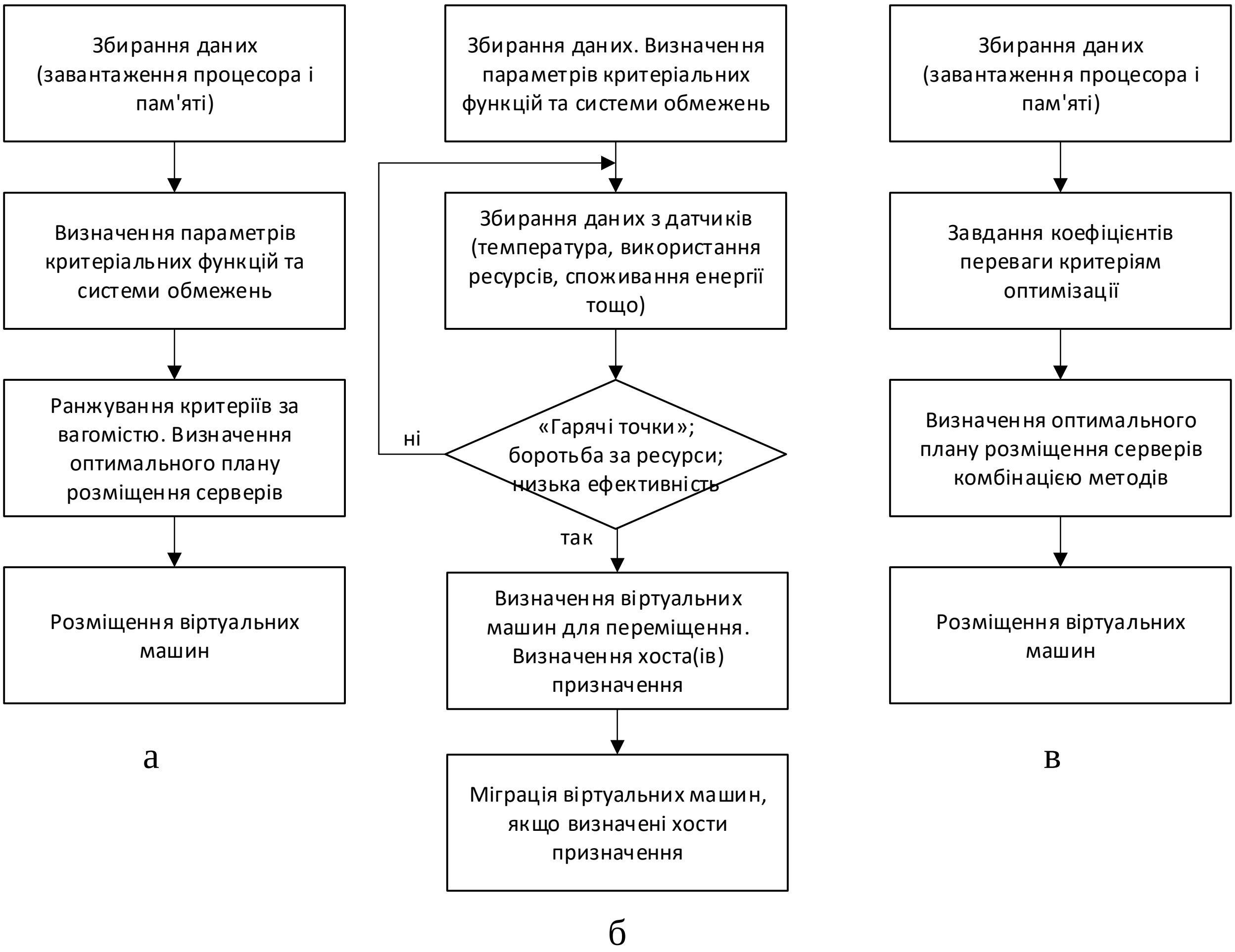


Рисунок 2.2 – Етапи методики оптимального розміщення віртуальних серверів: а) первинне розміщення; б) динамічне розміщення; в) статичне розміщення

## 2.3 Порівняння різних підходів до розподілу ресурсів у віртуальних середовищах ЦОД

Якість роботи планувальників ресурсів в центрах обробки даних визначає завантаження ресурсів ЦОД і можливість гарантованого забезпечення необхідної якості сервісу (SLA). Платформа OpenStack пропонує ряд алгоритмів, з яких слід звернути увагу на два: «призначення запиту на перший відповідний фізичний ресурс» та «вибір випадковим чином фізичного ресурсу з безлічі відповідних ресурсів» [8]. Ці алгоритми прийнятні за якістю одержуваних відображень запитів на фізичні ресурси ЦОД при їх завантаженні до 50%. При завантаженні фізичних ресурсів більше 50% якість відображень, одержуваних цими алгоритмами, значно погіршується. Для відображення ресурсних запитів на фізичні ресурси ЦОД потрібно вирішити три взаємопов'язаних NP-складних задачі:

- відображення віртуальних машин на обчислювальні ресурси ЦОД;
- відображення елементів зберігання даних (storage-елементів) на сховища даних ЦОД;
- відображення віртуальних каналів на фізичні ресурси мережі обміну ЦОД.

У разі високого завантаження фізичних ресурсів ЦОД алгоритми з OpenStack неприйнятні за якістю одержуваних відображень ресурсних запитів на фізичні ресурси ЦОД, а точні алгоритми неприйнятні з-за високої обчислювальної складності. Це обумовлює актуальність розробки евристичних алгоритмів, які дозволяють забезпечити потрібний баланс між точністю та обчислювальною складністю.

Для забезпечення високого завантаження в режимі використання ЦОД IaaS [9] потрібно рішення наступних завдань.

Завдання 1. Усунення сегментації фізичних ресурсів за рахунок міграції віртуальних ресурсів в ЦОД.

Завдання 2. Усунення перевантаження каналів обміну, що входять у фізичне сховище даних. Дане завдання виникає, коли є віртуальні сховища даних з високою інтенсивністю зчитування і низькою інтенсивністю запису. Вона може бути вирішена, якщо допускається реплікація сховищ даних. В цьому випадку для забезпечення консистентності даних не потрібен канал з високою пропускною здатністю, і одна половина застосунків може працювати з віртуальним сховищем даних, а інша – з його копією, яка розташовується в іншому фізичному сховищі даних.

Завдання 3. Можливість завдання SLA для всіх типів ресурсів ЦОД. Для цього обчислювальні ресурси, сховища даних, мережеві ресурси повинні розглядатися як плановані типи ресурсів, і їх планування має відбуватися узгоджено в сенсі дотримання угод про якість сервісу.

У таблиці 2.1 наведено порівняння відомих алгоритмів [10-23] з точки зору можливості вирішувати завдання 1-3. Як видно з таблиці, жоден з алгоритмів не дозволяє вирішувати всі перераховані вище завдання в сукупності.

У цьому розділі математично формулюється задача відображень ресурсних запитів на фізичні ресурси ЦОД і пропонуються алгоритми її вирішення, які дозволяють вирішувати всі перераховані вище завдання в сукупності для моделі обслуговування IaaS.

Таблиця 2.1 – Порівняльна характеристика існуючих методів та алгоритмів

| Розглянуті роботи | Обчислювальні ресурси як тип ресурсів | Сховища даних як тип ресурсів | Мережеві ресурси як тип ресурсів | Можливості | | |
|---|---|---|---|---|---|---|
| | | | | відображення віртуальних каналів на фізичні | реплікація | міграція |
| Application placement on a cluster of servers [10] | + | – | – | – | – | – |
| Cloud Storage and Online Bin Packing [11] | – | + | – | – | – | – |
| Efficient Resource Scheduling in Data Centers using MRIS [12] | + | + | – | – | – | – |
| Quantifying Load Imbalance on Virtualized Enterprise Servers [13] | + | – | – | – | – | + |
| On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach [14] | + | – | – | – | – | + |
| Optimal Mapping of Virtual Networks with Hidden Hops [15] | – | – | – | + | – | – |
| Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration [16] | + | – | – | + | – | + |
| A Virtual Network Mapping Algorithm based on Subgraph Isomorphism [17] | + | – | – | + | – | – |

| | | | | | | |
|---|---|---|---|---|---|---|
| Algorithms for Assigning Substrate Network Resources to Virtualized network Resources [18] | + | – | – | – | – | – |
| Virtual network embedding with coordinated node and link mapping [19] | + | – | – | + | – | – |
| Virtual Network Embedding Through Topology Aware Node Ranking [20] | + | – | – | + | – | – |
| Coupled placement in modern data centers [21] | + | + | + | ±1) | – | – |
| Server storage virtualization: integration and load balancing in data centers [22] | + | + | + | ±1) | – | + |
| Joint VM placement and routing for data center traffic engineering [23] | + | – | + | + | – | + |

Примітка. 1) Алгоритми застосовні лише для ЦОД, що мають ієрархічну (деревоподібну) топологію.

# 3 МАТЕМАТИЧНА ПОСТАНОВКА ЗАДАЧИ РОЗПОДІЛУ РЕСУРСІВ

Модель фізичних ресурсів ЦОД будемо задавати графом $H = (P \cup M \cup K, L)$, де P – множина обчислювальних вузлів, M – множина сховищ даних, K – множина комутаційних елементів мережі обміну ЦОД, L – множина фізичних каналів передачі даних.

На множині P визначені функції vh(p) і qh(p), що задають продуктивність обчислювального вузла (операцій за секунду) і наявний обсяг всієї пам’яті обчислювального вузла (байт).

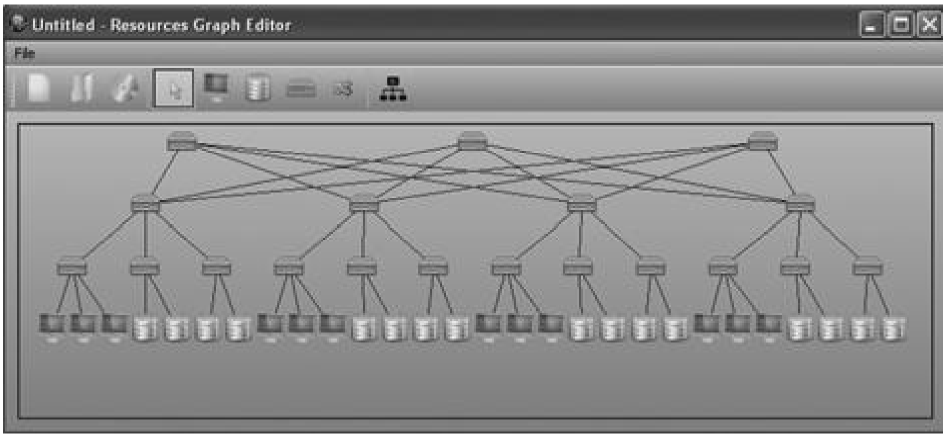


Рисунок 3.1 – Стандартна топологія ЦОД «Fat-Tree»

На множині M визначені функції uh(m) і type(m), що задають обсяг всієї пам’яті сховища даних (байт) і його тип.

На множині K визначена функція th(k), що задає пропускну здатність комутаційного елемента (байт/с), а на множині L – функція rh(l), що задає номінальну пропускну здатність каналу передачі даних (байт/с).

Функції vh(p), qh(p), uh(m), type(m), th(k), rh(l) формують розмітку графа H.

Приклад стандартної топології ЦОД «Fat-Tree» [17] наведено на рисунку 3.1: кожен кореневої комутатор з’єднаний з усіма комутаторами першого рівня; кожен комутатор першого рівня з’єднаний з декількома комутаторами другого рівня; кожен комутатор другого рівня об’єднує або кілька обчислювальних вузлів, або кілька сховищ даних.

Ресурсний запит будемо ставити графом $G = \left(W \cup S,\ E\right)$, де W – множина віртуальних машин, використовуваних застосунками, S – множина storage-елементів, E – множина віртуальних каналів передачі даних між віртуальними машинами і storage-елементами запиту.

На множині W визначені функції v(w) і q(w), що задають необхідну застосункам продуктивність віртуальних машин (операцій за секунду) і обсяг пам’яті.

На множині S визначені функції u(s) і type(s), що задають необхідний обсяг пам’яті (байт) і тип storage-елемента.

На множині E визначена функція r(e), що задає необхідну пропускну здатність віртуального каналу (байт/с).

Функції v(w), q(w), u(s), type(s), r(e) формують розмітку графа G. Ресурсні запити можуть бути двох типів:

- слабкопов’язаного типу:

$$[(\left\{W_i\right\}_{i=1}^{N}, \left\{S_j\right\}_{j=1}^{K}), E = \varnothing];$$

- сильнопов’язаного:

$$[(\left\{W_i\right\}_{i=1}^{N}, \left\{S_j\right\}_{j=1}^{K}), E = \left\{(W \subset \left\{W_i\right\}_{i=1}^{N}, S \subset \left\{S_j\right\}_{j=1}^{K})\right\}].$$

Призначенням ресурсного запиту будемо називати відображення $A: G \to H = \{W \to P,\ S \to M, E \to \left\{K,\ L\right\}\}$, що задовольняє наступним обмеженням.

Обмеження 1. Віртуальна машина w може бути призначена на виконання на обчислювальному вузлі p, якщо вірні умови

$$\sum_{w \in W_p} \nu(w) \le \nu h(p) \text{ та } \sum_{w \in W_p} \nu(w) \le \nu h(p).$$

Тут $W_p$ – множина віртуальних машин, призначених на виконання на обчислювальному вузлі p. Іншими словами, продуктивність обчислювального вузла повинна бути не менше необхідної сумарної продуктивності призначених на нього віртуальних машин і обсяг пам’яті обчислювального вузла повинен бути не менше необхідної сумарної пам’яті призначених на нього віртуальних машин. Планування виконання віртуальних машин на обчислювальному вузлі здійснюється локальним планувальником обчислювального вузла. Локальний планувальник повинен гарантовано забезпечувати для кожної віртуальної машини необхідну продуктивність.

Обмеження 2. Віртуальний канал e може бути відображений на фізичний канал l, якщо є вірною умова

$$\sum_{e \in E_l} r(e) \le rh(l).$$

Тут $E_l$ – множина віртуальних каналів, відображених на фізичний канал l.

Обмеження 3. Віртуальний канал e може проходити через комутаційний елемент k, якщо є вірною умова

$$\sum_{e \in E_k} r(e) \le \tau h(k).$$

Тут $E_k$ – множина віртуальних каналів, що проходять через комутаційний елемент k.

Обмеження 4. Storage-елемент s може бути розміщений в сховище даних m, якщо є вірною умова

$$\sum_{s \in S_m} u(s) \le uh(m).$$

Тут Sm – множина storage даних, що розміщені у сховищі m; тип сховища даних збігається з типом storage-елемента.

Реплікацією називається відображення $R: H \to H$, яке дублює дані деякого $m \in M$ і створює віртуальний канал підтримки консистентності даних (m’, $l_1$, $k_1$, ..., $k_{n-1}$, $l_n$, m); $k_i \in K$, $l_i \in L$, $m' \in M$, m’– репліка сховища m.

Якщо storage-елемент є реплікацією деякої бази даних s і потрібно забезпечення консистентності даних, то в граф запитуваних ресурсів G додається вершина s’. До графу G також додається віртуальний канал між вершинами s і s’, пропускна здатність якого визначається виходячи з вимоги забезпечення консистентності реплікації і бази даних.

Залишковим графом доступних ресурсів називається граф $H_{res}$, для якого перевизначені функції:

$$\nu h_{res}(p) = \nu h(p) - \sum_{w \in W_p} \nu(w),\ qh_{res}(p) = qh(p) - \sum_{w \in W_p} q(w),$$

$$uh_{res}(m) = uh(m) - \sum_{s \in S_m} u(s),\ \tau h_{res}(k) = \tau h(k) - \sum_{e \in E_l} r(e),$$

$$rh_{res}(l) = rh(l) - \sum_{e \in E_k} r(e).$$

В якості вихідних даних завдання призначення ресурсних запитів на фізичні ресурси задані:

1) множина запитів Z = {($G_i$, $T_i$)}, що надійшли центральному планувальнику; тут $T_i$ – час, на який запитуються ресурси запитом $G_i$;

2) залишковий граф доступних ресурсів $H_{res} = (P \cup M \cup K, L)$.

Потрібно: з множини Z розмістити на виконання у ЦОД максимальне число запитів. При цьому повинні бути вірними умови:

$$\sum_{w\in W_p} \nu(w) \le \nu h(p), \ \sum_{w\in W_p} q(w) \le qh(p);$$

$$\sum_{e\in E_l} r(e) \le rh(l); \ \sum_{e\in E_k} \tau(e) \le \tau h(k);$$

$$\sum_{s\in S_m} u(s) \le uh(m); \ \forall s \in S_m : type(s) = type(m).$$

Входом алгоритму призначення запитів на фізичні ресурси є залишковий граф доступних ресурсів $H_{res}$ і множина ресурсних запитів $\{G_i\}$. Множину $\{G_i\}$ формує диспетчер задач. До неї, крім новоприбулих запитів, можуть входити і запити, які виконуються, і для яких допустима міграція. Диспетчер завдань також визначає час запуску планувальника.

Виходом алгоритму призначення запитів на фізичні ресурси є множина призначень ресурсних запитів на фізичні ресурси $\left\{A_i : G_i \to H, i = \overline{1,n}\right\}$ і множина реплікацій $\{R_i\}$, i = 0,1, ...

# 4 МОДЕЛЬ РОЗПОДІЛУ РЕСУРСІВ СЕРВЕРІВ МІЖ ЗАВДАННЯМИ У ПРОЦЕСІ ФУНКЦІОНУВАННЯ ХМАРНОГО ОБЧИСЛЮВАЛЬНОГО СЕРЕДОВИЩА

У статті [24] була доведена актуальність проблеми розподілу ресурсів в хмарних обчислювальних середовищах і запропонована методика початкового виділення ресурсів на основі методу аналізу ієрархій для підвищення ефективності використання ресурсів хмарного обчислювального середовища (рисунок 4.1).

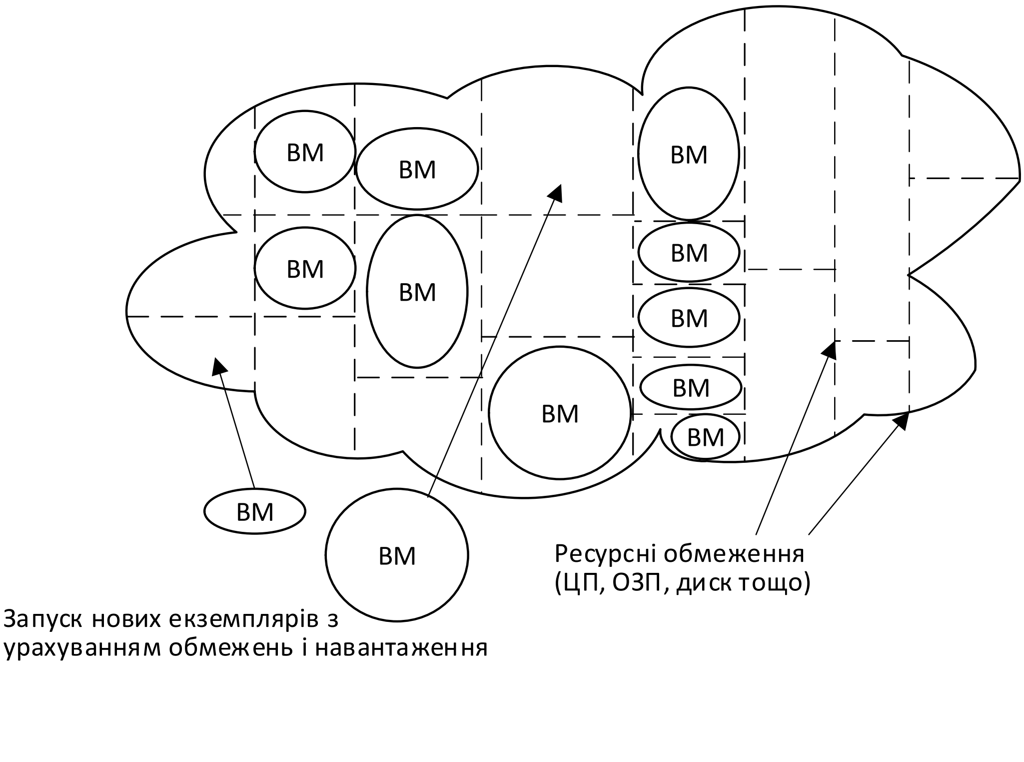


Рисунок 4.1 – Первинне виділення ресурсів у хмарному обчислювальному середовищі з використанням методу аналізу ієрархій

Суть зазначеної методики полягає в тому, щоб запустити черговий екземпляр (віртуальну машину, що надається як сервіс всередині хмарної обчислювального середовища) на хості з найменшою вірогідністю нераціонального використання ресурсів, що забезпечує більш ефективне використання ресурсів хмарного обчислювального середовища в цілому.

В даному випадку під ефективністю хмарного обчислювального середовища $E_{cl}$ розуміється частка примірників із загального їх числа, функціонуючих без втрати продуктивності через перевантаження будь-яких ресурсів.

Показник ефективності $E_{cl}$ є комплексним показником, що залежить від сукупності часткових показників:

$E_{CPU}$ – відсоток примірників із задоволеною потребою в ресурсах центрального процесора;

$E_{RAM}$ – відсоток примірників із задоволеною потребою в ресурсах пам’яті;

$E_{storage}$ – відсоток примірників із задоволеною потребою в дискових ресурсах;

$E_{network}$ – відсоток примірників із задоволеною потребою в мережевих ресурсах;

T – час функціонування хмарного обчислювального середовища.

Таким чином, можна визначити наступну залежність:

$$E_{cl} = F(E_{CPU}, E_{RAM}, E_{storage}, E_{network}, T) \rightarrow \max .$$

При цьому необхідно врахувати таке обмеження:

$$R_{used} < R_{exist} ,$$

де $R_{used}$ – загальний обсяг використаних ресурсів хмарного обчислювального середовища; $R_{exist}$ – загальний обсяг наявних ресурсів хмарного обчислювального середовища.

Максимальне значення показника $E_{cl}$ відповідає максимальному значенню продуктивності існуючих екземплярів при максимальній місткості хмарного обчислювального середовища, тобто здатності без втрати продуктивності або з мінімальною її втратою забезпечувати функціонування застосункових примірників з урахуванням існуючих ресурсних обмежень.

Проте, рішення задачі початкового виділення ресурсів не забезпечує подальшого раціонального розподілу ресурсів в ході функціонування хмарного обчислювального середовища. З урахуванням того, що всередині примірників (віртуальних машин – будемо вважати, що терміни «екземпляр» та «віртуальна машина» в контексті хмарного обчислювального середовища є синонімами) реалізовані програми та програмні комплекси, такі, як СКБД, веб-сервіси, ERP-системи тощо [25], параметри цих примірників, пов’язані з навантаженням, істотно змінюються в часі. З цього випливає, що показник ефективності функціонування хмарного обчислювального середовища також змінюється у часі, й для забезпечення потрібного значення показника ефективності $E_{cl}$ є необхідним постійний моніторинг використання ресурсів та вчасний їх перерозподіл між віртуальними машинами задля запобігання втрат продуктивності у різних ланках хмарного обчислювального середовища, іншими словами, балансування навантаження віртуальних машин на існуючих ресурсах хмарного обчислювального середовища.

Як механізм балансування навантаження в хмарному обчислювальному середовищі запропоновано використовувати підхід на основі «живої міграції», під якою розуміється технологія міграції віртуальних машин між хостами або ресурсами зберігання з нульовим часом простою застосунків.

Черговість процесу динамічної міграції в разі зміни хоста (фізичного сервера, на якому виконується віртуальна машина) виглядає наступною послідовністю кроків.

Крок 1. Зупинка віртуальної машини.

Крок 2. Передача параметрів віртуальної машини з сервера початкового розташування на сервер цільового розташування.

Крок 3. Передача образу оперативної пам’яті з сервера початкового розташування віртуальної машини на сервер цільового розташування.

Крок 4. Створення віртуального домену та розміщення способу оперативної пам’яті в оперативній пам’яті сервера цільового розташування.

Крок 5. Запуск віртуальної машини на сервері цільового розташування.

За необхідності зміни сховища, образ віртуальної машини переноситься з одного ресурсу зберігання на інший.

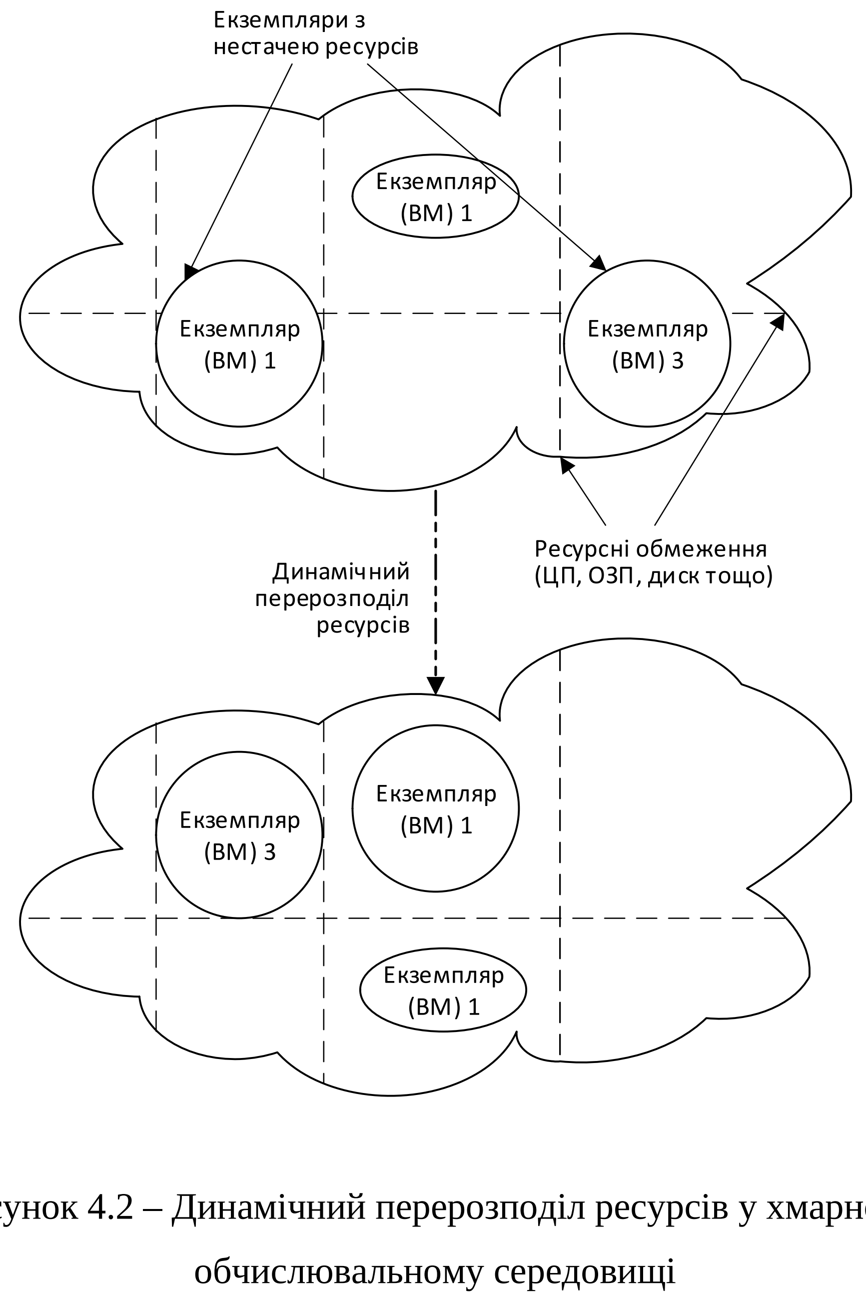


Рисунок 4.2 – Динамічний перерозподіл ресурсів у хмарному обчислювальному середовищі

Для того, щоб ефективно балансувати навантаження в хмарному обчислювальному середовищі, необхідно прогнозувати його зміну в часі, оскільки грунтуючись тільки на поточному навантаженні досить складно запобігти нестачі ресурсів – можна тільки усунути її за фактом. Більш того, в деяких ситуаціях перенесення віртуальних машин не потрібно, а лише підвищує накладні витрати на міграцію [24].

Виходячи з вищесказаного, процес перерозподілу ресурсів в хмарному обчислювальному середовищі можна розділити на два етапи.

Етап 1. Прогноз навантаження.

Етап 2. Виявлення хостів з підвищеним навантаженням і динамічний перерозподіл ресурсів.

В даній роботі розглянуто рішення задачі на другому етапі процесу розподілу ресурсів.

## 4.1 Формалізація завдання розподілу ресурсів

Для вирішення завдання розподілу ресурсів запропоновано використовувати концепцію нерівномірності $N_R$ використання ресурсів p-го сервера ($N_R^p$), представлену в [24]. Нехай n – число ресурсів, що розглядаються, а $r_i$ – прогнозоване завантаження i-го ресурсу сервера p. Визначимо прогнозовану нерівномірність завантаження ресурсів сервера p як:

$$N_R^p = \sqrt{\sum_{i=1}^{n}\left(\frac{r_i - \overline{r}}{\overline{r}}\right)^2},$$

де $\overline{r}$ – середнє прогнозне завантаження всіх ресурсів сервера p.

На практиці не для всіх типів ресурсів зниження продуктивності позначається на ефективності функціонування системи в цілому, тому необхідно враховувати тільки значні ресурси в розрахунку (такі, як завантаження ЦП, оперативної пам’яті, дискової підсистеми, мережі тощо). Шляхом мінімізації показника нерівномірності вдається поєднати різні типи навантаження і домогтися поліпшення загального використання серверних ресурсів хмарного обчислювального середовища (рисунок 4.2).

Алгоритм виконується періодично для отримання оцінки стану виділених ресурсів, яка заснована на прогнозі потреб примірників. Сервер називається «гарячою точкою», якщо використання будь-якого з його ресурсів вище «допустимої позначки», попередньо заданої адміністратором хмарного обчислювального середовища для кожного типу ресурсів. Це говорить про те, що хост перевантажений, тому частина примірників повинна бути переміщена з нього на інші хости. «Температура» $t^*$ гарячої точки визначається як квадратична сума використання всіх її ресурсів вище «допустимої позначки» [25]:

$$t^* = \sum_{r \in R} (r - r_t)^2 ,$$

де R – набір перевантажених ресурсів сервера p; $r_t$ – «допустима відмітка» ресурсу r (враховуються тільки перевантажені ресурси).

Температура гарячої точки відображає ступінь перевантаження сервера. Якщо сервер не є гарячою точкою, його температура дорівнює нулю.

Різні типи ресурсів мають різні допустимі позначки. Наприклад, для завантаженого ЦП та використаної оперативної пам’яті вони можуть бути визначені як 90% і 80% відповідно. Таким чином, сервер стає гарячою точкою при досягненні цього навантаження.

## 4.2 Алгоритм балансування навантаження у хмарному обчислювальному середовищі

Алгоритм балансування навантаження є сукупністю наступних кроків.

Крок 1. Сортування списку серверів-«гарячих точок» p за спаданням температури, $t^*$ (тобто найгарячіша точка стає першою в списку). Мета цієї дії – виключити всі гарячі точки, якщо це можливо, або, принаймні, підтримувати їх температуру настільки низькою, наскільки це можливо.

Крок 2. Визначення для кожного сервера p списку примірників e, для яких повинна бути виконана міграція.

Крок 3. Сортування списку примірників e на основі результуючої температури хоста $t^{*}_{result}$, яка визначається після міграції примірника (віртуальної машини). Метою є виконання міграції того примірника, який знизить температуру сервера $p_n$ до мінімального значення: $t^{*}_{result}(p_n) \rightarrow \min$.

Крок 4. Вибір віртуальної машини e', міграція якої дозволить максимально знизити нерівномірність сервера.

Крок 5. Визначення для знайденої віртуальної машини e' можливості знаходження цільового сервера $p_{target}$ для її розміщення. Сервер вважається придатним для міграції, якщо після перенесення на нього віртуальної машини e' нерівномірність цільового сервера виявиться меншою за нерівномірність поточного.

Крок 6. Вибір сервера $p^{'}_{target}$ з цього списку, нерівномірність якого буде максимально знижена після перенесення даної віртуальної машини. Варто звернути увагу, що нерівномірність цільового сервера може бути і збільшена. В цьому випадку необхідно вибрати сервер, для якого таке збільшення буде мінімальним.

Крок 7. Якщо цільовий сервер $p^{'}_{target}$ знайдений, ініціюється міграція віртуальної машини на цей сервер і оновлюється прогноз навантаження для всіх причетних серверів. В іншому випадку, зі списку вибирається наступна віртуальна машина і відбувається пошук цільового сервера для неї.

Структурна схема алгоритму представлена на рисунку 4.3. До тих пір, поки можна знайти цільової сервер для будь-якої віртуальної машини сервера-«гарячої точки», виконання алгоритму для даного сервера триває. Як тільки це стає неможливим, здійснюється перехід до наступної «гарячої точки».

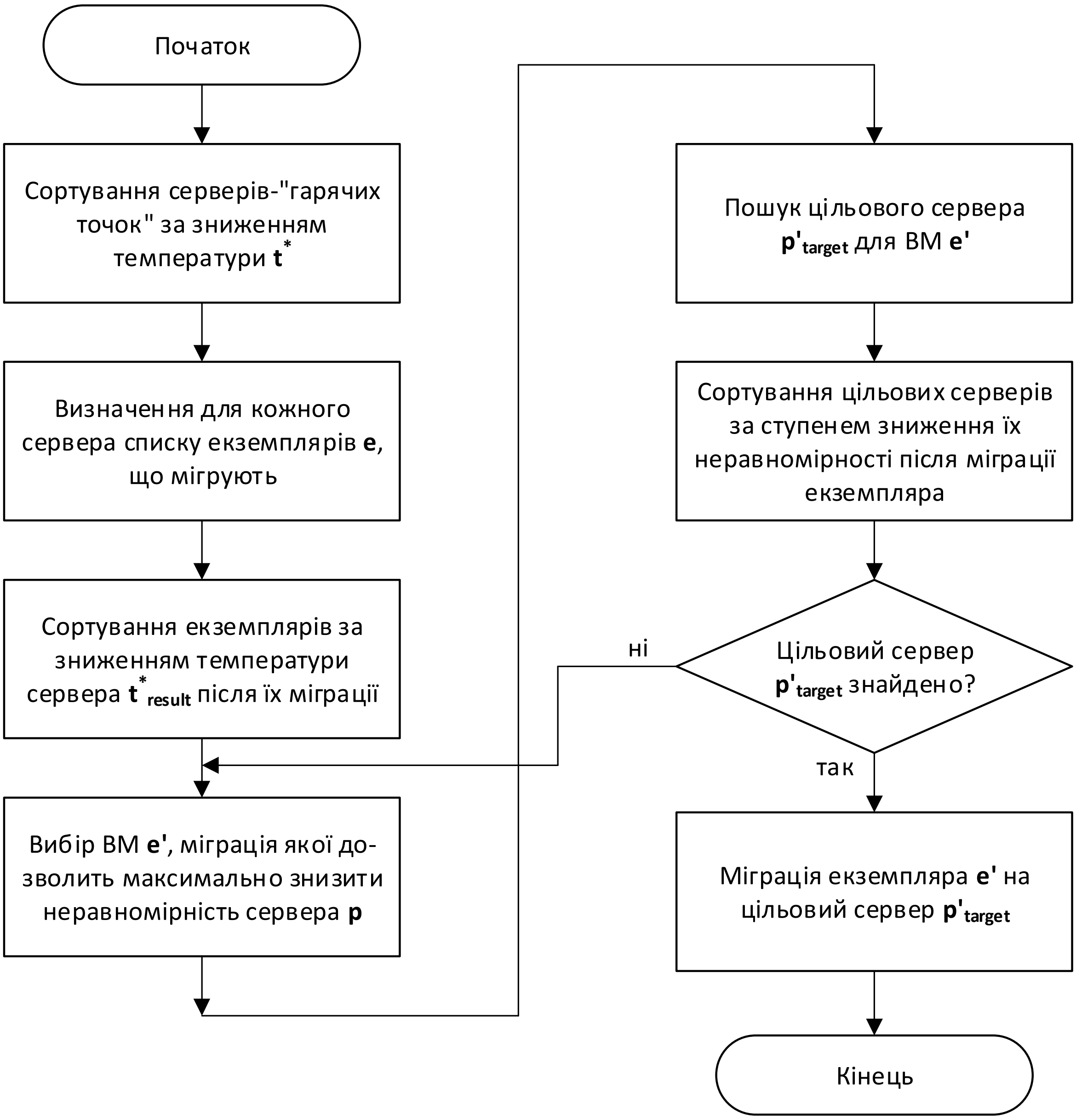


Рисунок 4.3 – Алгоритм балансування навантаження у хмарному обчислювальному середовищі

Прогнозування навантаження з використанням штучної нейронної мережі перед запуском алгоритму дозволяє не враховувати несуттєві, нетривалі піки навантаження, що веде до значного зниження числа помилкових спрацьовувань алгоритму усунення гарячих точок і, таким чином, зниження накладних витрат на переміщення віртуальних машин.

## 4.3 Побудова алгоритмів призначення запитів на фізичні ресурси ЦОД

Для рішення сформульованої задачі пропонуються три розроблені алгоритми:

- алгоритм із загальним планувальником для всіх типів ресурсів;
- алгоритм з використанням незалежних планувальників для кожного типу ресурсів;
- алгоритм на основі схеми мурашиних колоній.

Алгоритм із загальним планувальником для всіх типів ресурсів заснований на поєднанні жадібних стратегій і стратегій обмеженого перебору: відповідно до трьох жадібних критеріїв вибирається спочатку черговий запит, потім – елементи запиту і фізичні ресурси для них. Якщо на черговому кроці не вдалося розмістити елемент запиту, то викликається процедура обмеженого перебору. Якщо елемент є storage-елементом, то викликається процедура реплікації і тільки в разі неуспішного завершення процедури реплікації викликається процедура обмеженого перебору. Для процедури обмеженого перебору задається параметр, який дозволяє регулювати обчислювальну складність і точність алгоритму.

При використанні в ЦОД незалежних планувальників для кожного типу ресурсів алгоритм призначення запитів на фізичні ресурси складається з трьох кроків.

Крок 1. Призначення віртуальних машин в мінімальне число обчислювальних вузлів.

Крок 2. Призначення storage-елементів в мінімальне число фізичних сховищ даних.

Крок 3. Для отриманих відображень віртуальних машин і storage-елементів на фізичні ресурси побудова відображення віртуальних каналів запитів на фізичні канали мережі обміну.

Алгоритми призначення віртуальних машин і storage-елементів засновані на поєднанні жідібних стратегій і стратегій обмеженого перебору.

Відповідно до трьох жадібних критеріїв вибирається спочатку черговий ресурсний запит, потім – черговий елемент запиту (віртуальна машина/storage-елемент) і фізичний ресурс для нього (обчислювальний вузол/сховище даних). Якщо на черговому кроці не вдалося розмістити елемент запиту, то викликається процедура обмеженого перебору. Для процедури обмеженого перебору задається параметр, який дозволяє регулювати обчислювальну складність і точність алгоритму. Алгоритм відображення віртуальних каналів на фізичні ресурси заснований на вирішенні задачі про знаходження k найкоротших шляхів в графі [19]. Процедура реплікації storage-елементів викликається, якщо не вдалося відобразити віртуальний канал на фізичні канали.

Алгоритм на базі схеми мурашиних колоній може бути використаний замість алгоритму із загальним планувальником для всіх типів ресурсів. Алгоритм заснований на схемі мурашиних колоній [26] для відображення віртуальних машин і storage-елементів на обчислювальні вузли і сховища даних. Далі для отриманих відображень віртуальних машин і storage-елементів на фізичні ресурси будуються відображення віртуальних каналів запитів на фізичні канали мережі обміну за допомогою алгоритму знаходження найкоротшого шляху [27]. На відміну від алгоритмів, заснованих на поєднанні жадібних стратегій і стратегій обмеженого перебору, даний алгоритм не вимагає підбору евристичних процедур для приватних завдань (задані обмеження на можливі значення вихідних даних), але має більш високу обчислювальну складність.

## 4.4 Результати експериментального дослідження властивостей алгоритмів

Всі експерименти проводилися для моделей ЦОД з архітектурою «Fat-Tree».

### 4.4.1 Порівняння запропонованих алгоритмів з алгоритмами платформи OpenStack

Наведені результати порівняння запропонованих алгоритмів за якістю одержуваних відображень з алгоритмами платформи OpenStack. Якість відображень будемо оцінювати відсотком розміщених в ЦОД запитів з початково заданого набору. Всі експерименти проводилися для наборів слабкопов'язаних запитів, оскільки алгоритми з платформи OpenStack не підтримують призначення сильнопов'язних запитів.

У платформі OpenStack реалізовано два алгоритми призначення ресурсних запитів: First Fit і Random Fit. Для кожного з алгоритмів були згенеровані набори вихідних даних, які демонструють недоліки функціонування алгоритмів при високому завантаженні фізичних ресурсів ЦОД.

Для алгоритму First Fit були згенеровані вхідні дані з наступними характеристиками:

- кількість фізичних обчислювальних вузлів – 6n (n – натуральне) з продуктивністю в 210 ум. од.;
- загальне число віртуальних машин – 18n; запит містить:

а) 6n віртуальних машин з необхідною продуктивністю 31 ум. од.;

б) 6n віртуальних машин з необхідною продуктивністю 71 ум. од.;

в) 6n віртуальних машин з необхідною продуктивністю 106 ум. од.

При необмеженій кількості вузлів алгоритм First Fit призначає всі запити у 10n вузлів (рисунок 4.4). Оптимальне призначення всіх запитів можливе у 6n вузлів (рисунок 4.5). При обмеженні числа вузлів значенням 6n алгоритм First Fit виробляє призначення всіх віртуальних машин, крім 4n віртуальних машин з ємністю 106 ум. од., отримуючи при цьому завантаження ресурсів, яке дорівнює 65%. Запропоновані алгоритми успішно призначали всі віртуальні машини із завантаженням фізичних ресурсів 99%. При проведенні експериментів було з'ясовано, що якість відображень не залежить від значення n.

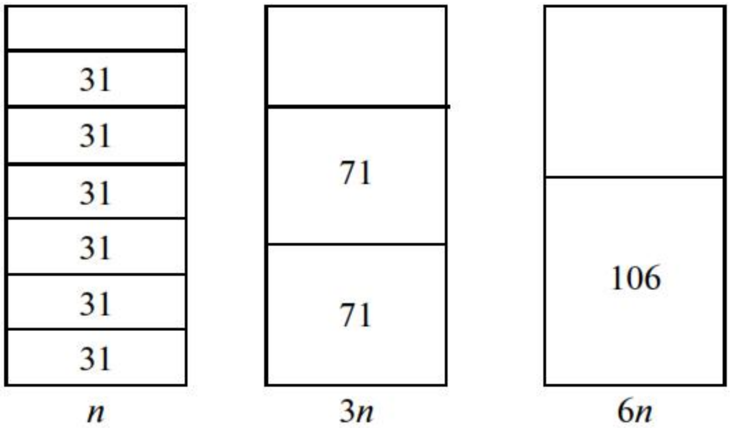


Рисунок 4.4 – Призначення ресурсів алгоритмом First Fit

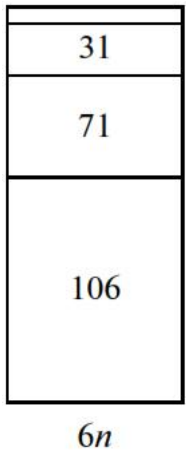


Рисунок 4.5 – Оптимальне призначення ресурсів

Для алгоритму Random Fit були використані вхідні дані з наступними характеристиками:

- кількість обчислювальних вузлів n + 1 – кожен з продуктивністю 1 млн ум. од .;
- загальна кількість віртуальних машин $n^3 + n$:

а) $n^3$ віртуальних машин з необхідною продуктивністю 1 млн/$n^3$ ум. од.;

б) n віртуальних машин з необхідною продуктивністю 1 млн ум. од.

При проведенні призначення запити з маленької ємністю заповнювали надані ресурси таким чином, що не залишалося обчислювальних вузлів, в яких не було б призначено хоча б однієї віртуальної машини. Після цього віртуальні машини, для яких уже була потрібна повна ємність обчислювального вузла, вже не могли бути призначені. Отримане завантаження ресурсів для алгоритму Random Fit дорівнювало в середньому 10%. Розроблені алгоритми дозволяли повністю призначити всі віртуальні машини із завантаженням 100%. При проведенні експериментів було з'ясовано, що якість одержуваних відображень не залежить від значення n.

Для демонстрації погіршення якості роботи алгоритмів платформи OpenStack в залежності від потенційно можливого завантаження фізичних ресурсів була застосована схема дослідження зі зростаючим навантаженням. Для описаних тестів із значенням n = 10 вироблялося видалення ресурсних запитів таким чином, щоб потенційно можливе завантаження фізичних ресурсів знижувалося з кроком 10%. Таким чином, було отримано дві серії тестових даних зі зростаючим завантаженням. Результати роботи алгоритмів наведені на рисунках 4.6, 4.7.

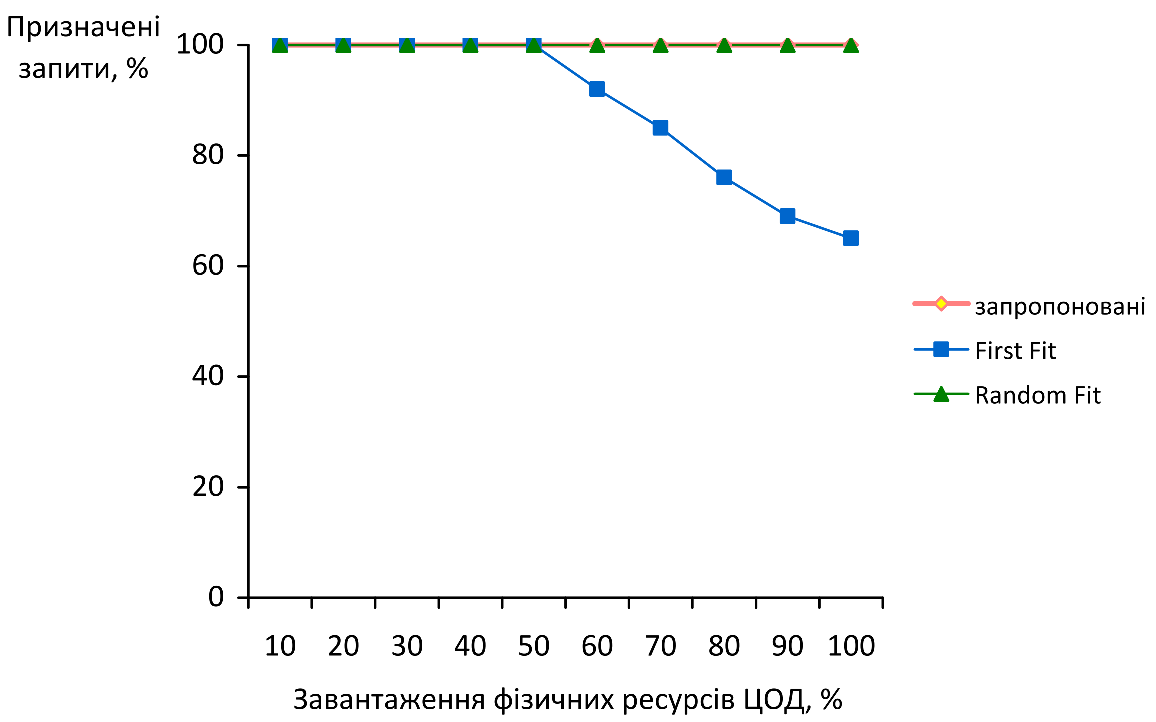


Рисунок 4.6 – Результати роботи всіх алгоритмів на даних для First Fit

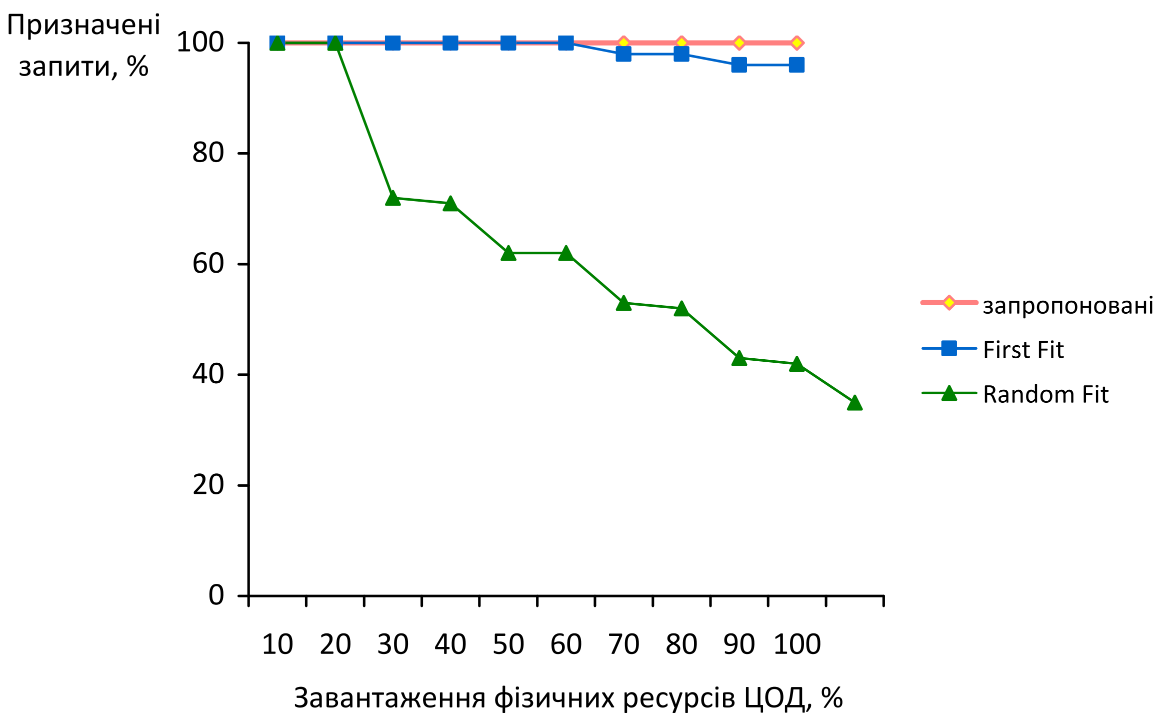


Рисунок 4.7 – Результати роботи всіх алгоритмів на даних для Random Fit

### 4.4.2 Дослідження ефективності процедури реплікації

Для демонстрації ефективності процедури реплікації використовувалися наступні параметри топології «Fat-Tree»:

- кореневий рівень архітектури складався з 10 маршрутизаторів з пропускною спроможністю 50000 ум. од. кожен. Канали передачі даних, що з’єднують маршрутизатори кореневого і другого рівнів, мали пропускну здатність 10000 ум. од.;
- другий рівень архітектури складався з 15 маршрутизаторів з пропускною спроможністю 16000 ум. од. кожен. Канали передачі даних, що з’єднують маршрутизатори другого і першого рівнів, мали пропускну спроможність 2000 ум. од.;
- кожен маршрутизатор другого рівня був з’єднаний з чотирма маршрутизаторами першого рівня з пропускною спроможністю 2000 ум. од. Кожен маршрутизатор першого рівня з’єднувався з двома обчислювальними вузлами або сховищами даних каналами передачі з пропускною

спроможністю 1000 ум. од .;

- всього в ЦОД було 60 обчислювальних вузлів з продуктивністю 1000 ум. од. та 60 сховищ даних з ємністю 1000 ум. од. Кожен маршрутизатор другого рівня мав доступ до чотирьох обчислювальних вузлів і чотирьох сховищ даних.

Для генерації набору запитів використовувалися чотири основних шаблони.

Шаблон 1 з невеликою кількістю віртуальних каналів: сім віртуальних машин, п’ять storage-елементов і 11 віртуальних каналів (в середньому по два віртуальних канали на один storage-елемент).

Шаблон 2 з великим числом віртуальних каналів: п’ять віртуальних машин, два storage-елемента і вісім віртуальних каналів (по чотири віртуальних канали на один storage-елемент).

Шаблон 3 з невеликою кількістю віртуальних машин: три віртуальних машини і два storage-елемента. Один зі storage-елементів був з’єднаний з усіма віртуальними машинами, а другий – з однією з них (всього чотири віртуальних канали).

Шаблон 4 з високою вимогою до пропускної здатності мережі: дві віртуальних машини, один storage-елемент і два віртуальних канали між ними. Значення запитуваної пропускної здатності віртуальних каналів вибиралися так, щоб їх сума дорівнювала 900 ум. од. (При пропускній спроможності каналу, що веде до сховища даних, у 1000 ум. од.).

Набори запитів ділилися на два основні класи та генерувалися за такими схемами. У першому класі варіювалося потенційно можливе завантаження каналів (при оптимальному розміщенні), що ведуть до сховищ даних, від 0,3 до 1,0. Завантаження обчислювальних вузлів і сховищ даних було фіксованим і дорівнювало 0,75. В даному класі були присутні тільки запити, згенеровані за першим та другим шаблонами.

У другому класі вхідних даних використовувалися тільки запити, згенеровані за третім і четвертим шаблонами. В даному класі варіювалася кількість запитів, згенерованих за четвертим шаблоном, від 0 до 30, при цьому потенційно можливе завантаження мережі зростало від 0,5 до 0,8. Завантаження обчислювальних вузлів і сховищ даних була фіксованою і дорівнювала 0,75.

Кількість запитів в наборі завжди дорівнювала 100. Всі алгоритми запускалися на кожному з наборів обох класів вхідних даних один раз із забороною реплікації, і один раз з її дозволом. Алгоритм на основі схеми мурашиних колоній виконував 70 ітерацій, параметр глибини перебору в алгоритмах із загальними і окремими планувальниками ресурсів дорівнював двом. На наступних рисунках зображена різниця в кількості розміщених запитів кожним з алгоритмів з забороною і дозволом операції реплікації.

На рисунках 4.8 і 4.9 представлені результати роботи алгоритму з окремими планувальниками ресурсів на першому і другому класах даних. На рисунках 4.10 і 4.11 показані результати роботи алгоритму з єдиним планувальником ресурсів на першому і другому класах даних.

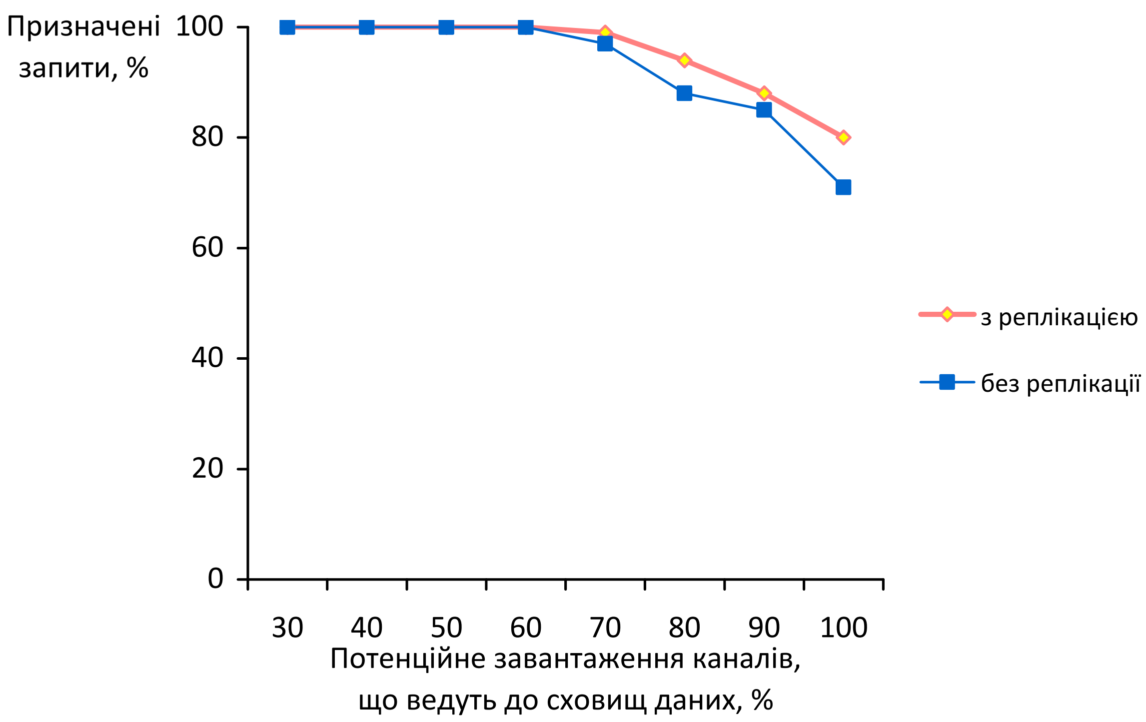


Рисунок 4.8 – Алгоритмим з окремими планувальниками на першому класі даних

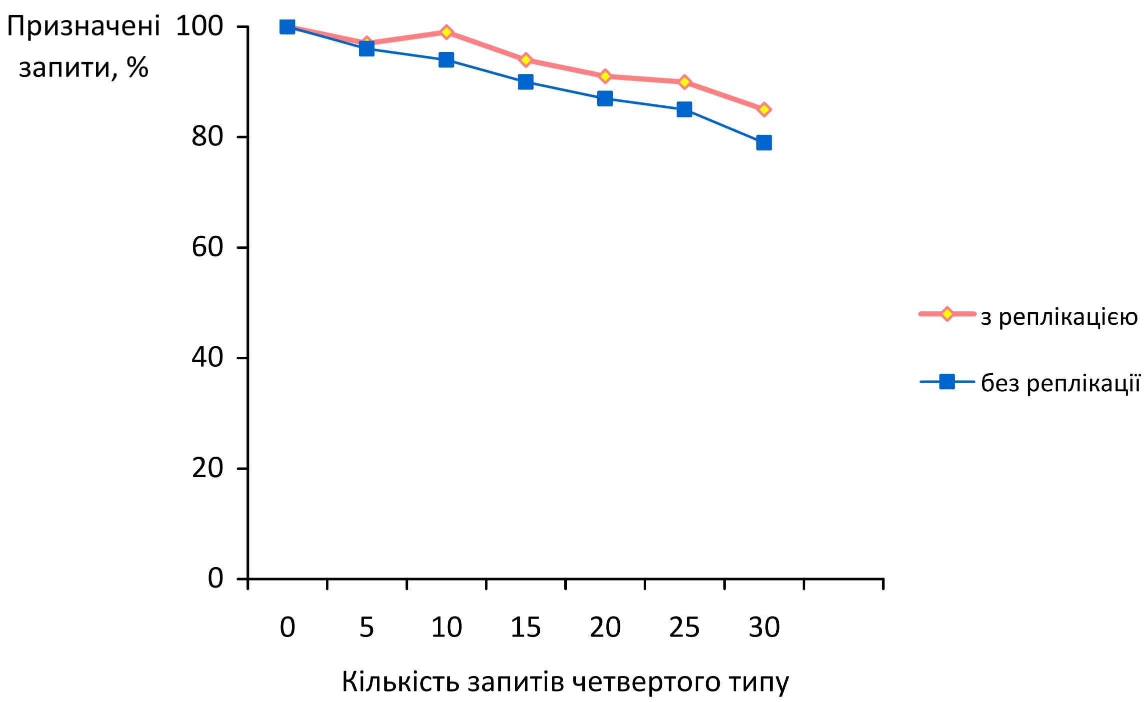


Рисунок 4.9 – Алгоритмим з окремими планувальниками на другому класі даних

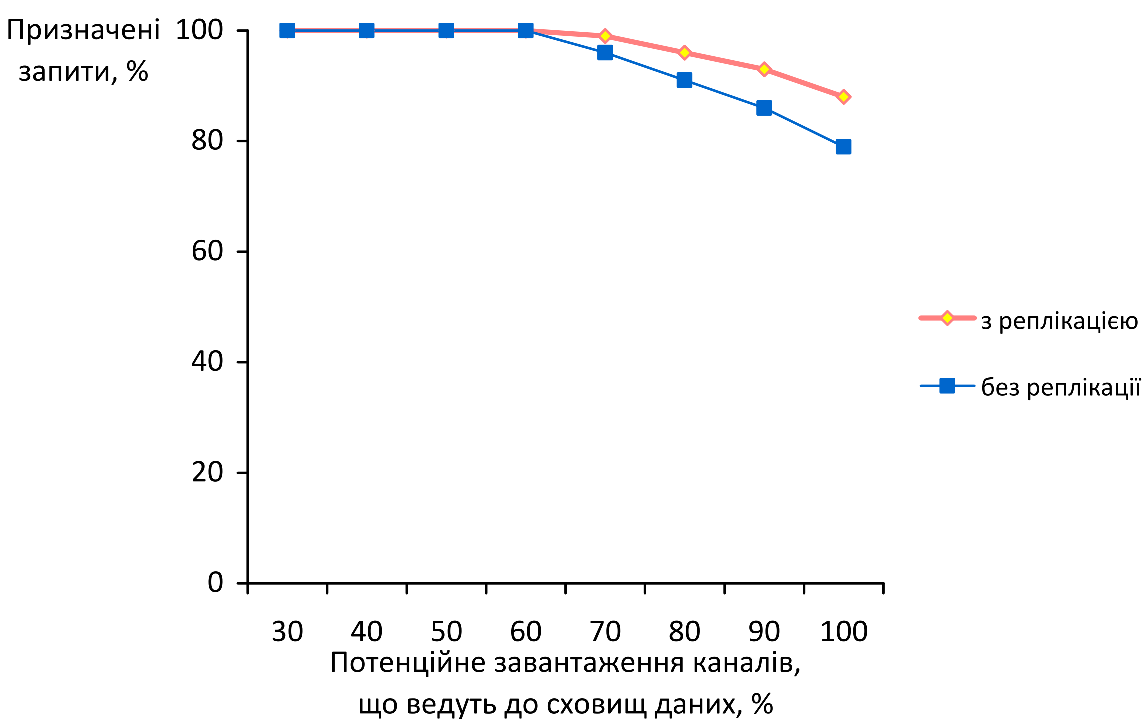


Рисунок 4.10 – Алгоритм з єдиним планувальником на першому класі даних

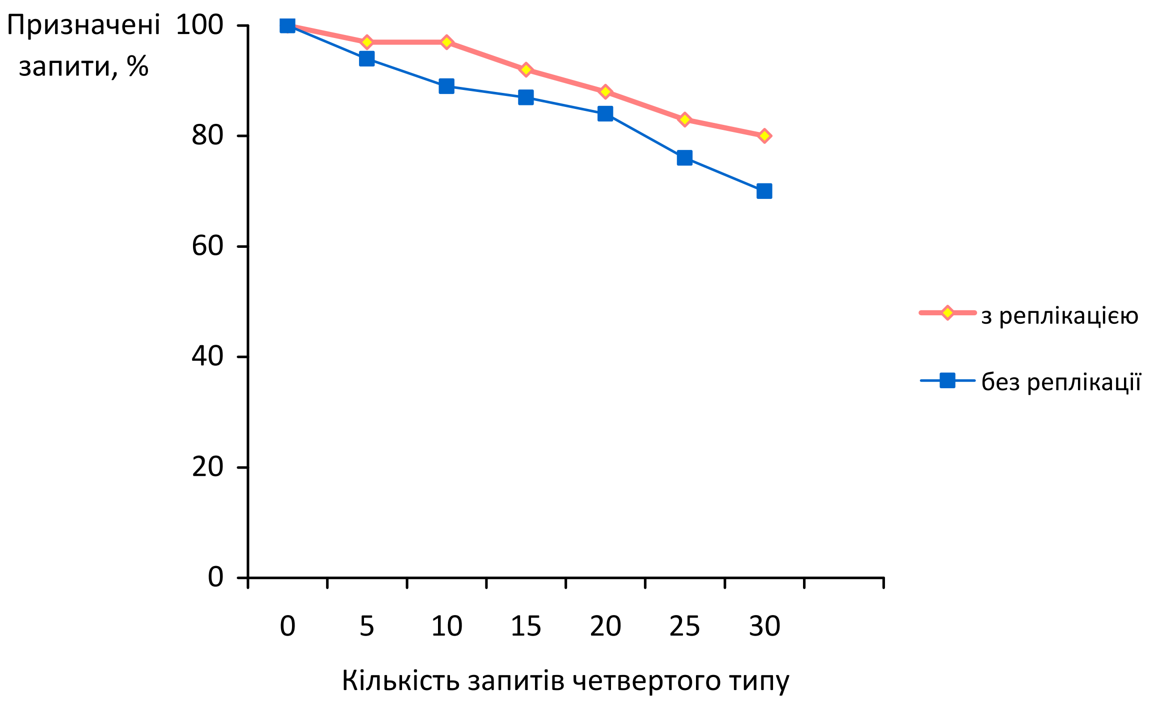


Рисунок 4.11 – Алгоритм з єдиним планувальником на другому класі даних

На рисунках 4.12 і 4.13 наведені результати роботи мурашиного алгоритму на першому і другому класах даних.

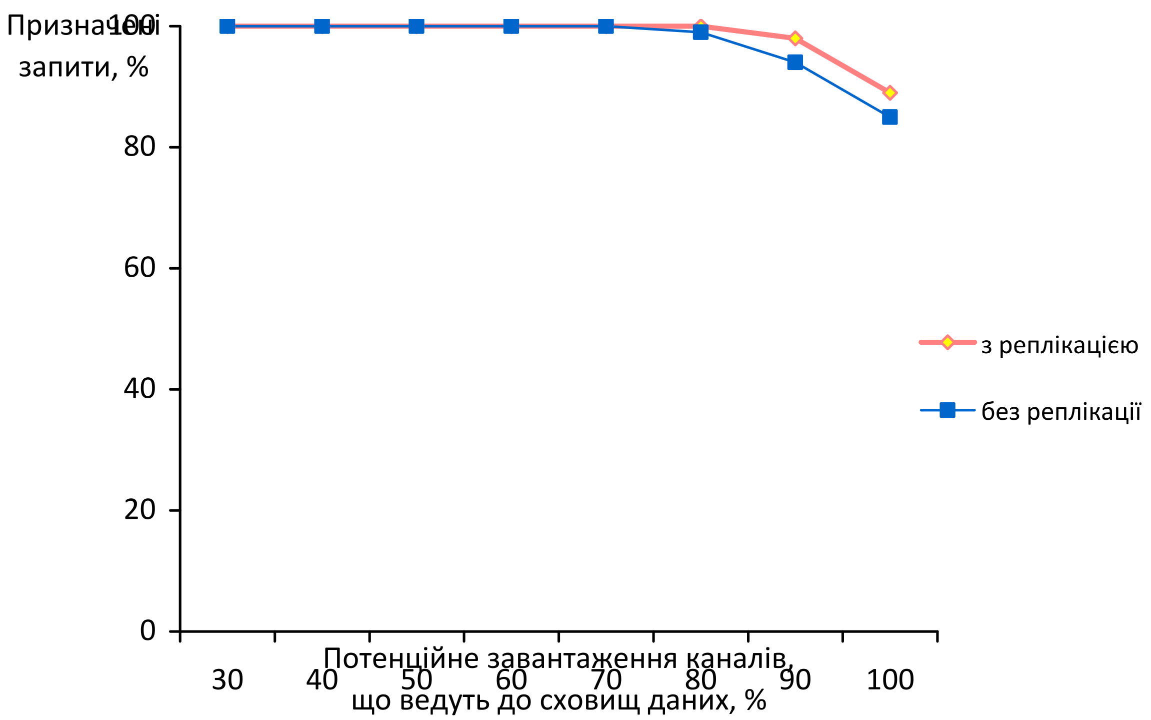


Рисунок 4.12 – Мурашиний алгоритм на першому класі даних

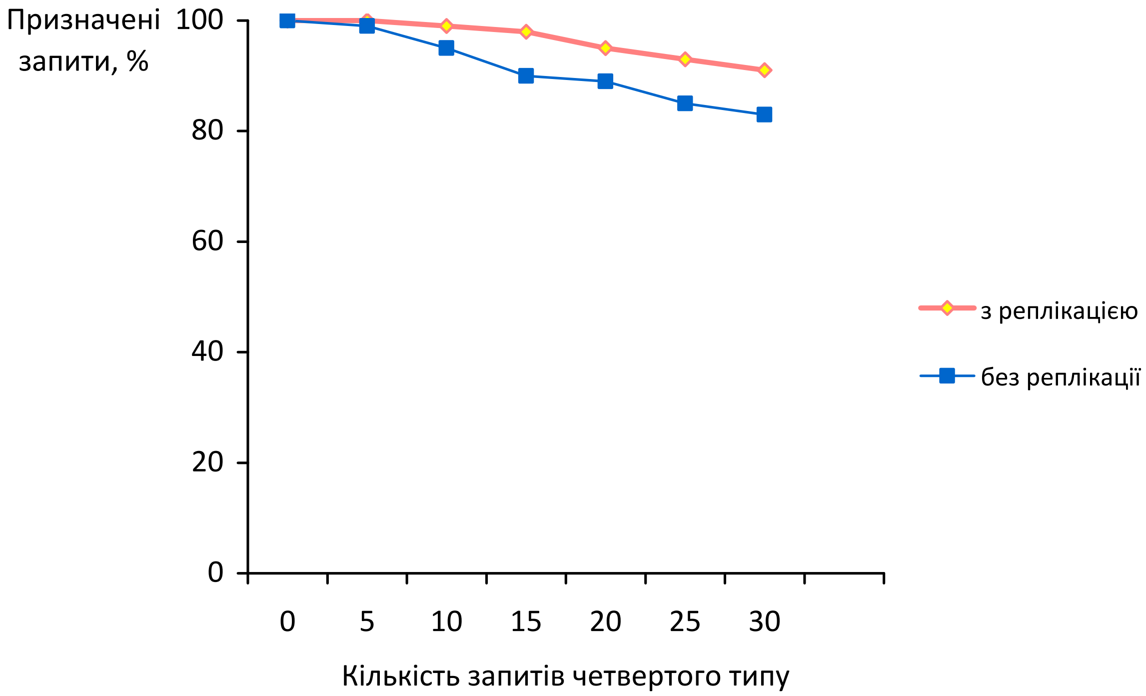


Рисунок 4.13 – Мурашиний алгоритм на другому класі даних

Дослідження алгоритмів на першому класі вхідних даних показали, що процедура реплікації дозволяє підвищити кількість розміщуваних запитів при потенційно можливому завантаженні каналів, що ведуть до сховищ даних, 0,7 і вище. Для другого класу вхідних даних аналогічні результати були отримані вже при наявності хоча б п’яти запитів з високою вимогою до пропускної здатності мережі обміну.

Также слід зазначити, що на першому класі даних алгоритм з єдиним планувальником розміщує більше запитів, ніж алгоритм з окремими планувальниками, тоді як на другому класі даних ситуація зворотна. Мурашиний алгоритм при цьому працює ефективніше обох алгоритмів як на першому, так і на другому класах даних.

## 4.5 Умови ефективного застосування запропонованих алгоритмів

Для демонстрації типових областей застосування алгоритмів використовувалася мережа ЦОД з топологією «Fat-Tree» з наступними характеристиками:

- кореневий рівень архітектури складається з п'яти маршрутизаторів з пропускною здатністю 100000 ум. од.;

- другий рівень архітектури складається з 10 маршрутизаторів з пропускною спроможністю 20000 ум. од. Перший і другий рівні архітектури з'єднані в мережу дводольним графом, пропускна здатність каналів зв'язку становить також 20000 ум. од.;

- до кожного маршрутизатора другого рівня підключено по чотирьох маршрутизаторів третього рівня з пропускною спроможністю 5000 ум. од. кожен. Пропускна здатність каналів зв'язку між другим і третім рівнями мережі становить 5000 ум. од. на канал;

- сукупність одного маршрутизатора другого рівня і підключених до нього чотирьох маршрутизаторів третього рівня будемо називати сегментом топології «Fat-Tree». Листковий рівень топології кожного сегмента формується 10 обчислювальними вузлами і 10 сховищами даних, які підключені до маршрутизаторів третього рівня з розрахунку п'ять машин одного типу на маршрутизатор.

Дана топологія відображає топологію незалежних сегментів, що застосовується на практиці, з додатковим рівнем доступності. При такій організації роботи ЦОД сильно пов'язані запити розміщуються в одному сегменті для зменшення затримок передачі даних і підвищення продуктивності міжмережевої взаємодії.

Дослідження проводилися на двох класах даних.

Набори запитів з першого класу були отримані за допомогою генератора оптимального посегментного завантаження. Для заданої топології він дозволяє отримувати набори запитів, повне розміщення яких вимагає завантаження кожного типу ресурсів ЦОД не менше ніж на 95%. Набір запитів формується завданням різного числа завантажених сегментів (від 1 до 10). Відповідно потенційно можливе завантаження ресурсів ЦОД (при оптимальному відображенні запитів на фізичні ресурси) становить для цього класу від 9,5 до 95%.

Другий клас даних отримано тими ж методами, що і перший, але потенційне завантаження мережевих каналів листкового рівня для нього становить в середньому 50% з інтервалом розкиду 5%. Додатково введено обмеження на відповідність запитів шаблонам «важкого» і «легкого» типів. Математичне очікування необхідної продуктивності елементів запиту «важкого» типу становить 60% продуктивності фізичного ресурсу (обчислювального вузла або сховища даних). Відповідне значення для запитів «легкого» типу становить 20% продуктивності фізичних вузлів.

Для кожного класу вихідних даних генерувалося в середньому по 200 запитів на сегмент, що забезпечують його завантаження на 95% при оптимальному відображенні. Число віртуальних машин і сховищ даних в запиті – від 10 до 20, відношення числа віртуальних машин і сховищ даних до числа віртуальних каналів дорівнює 1.

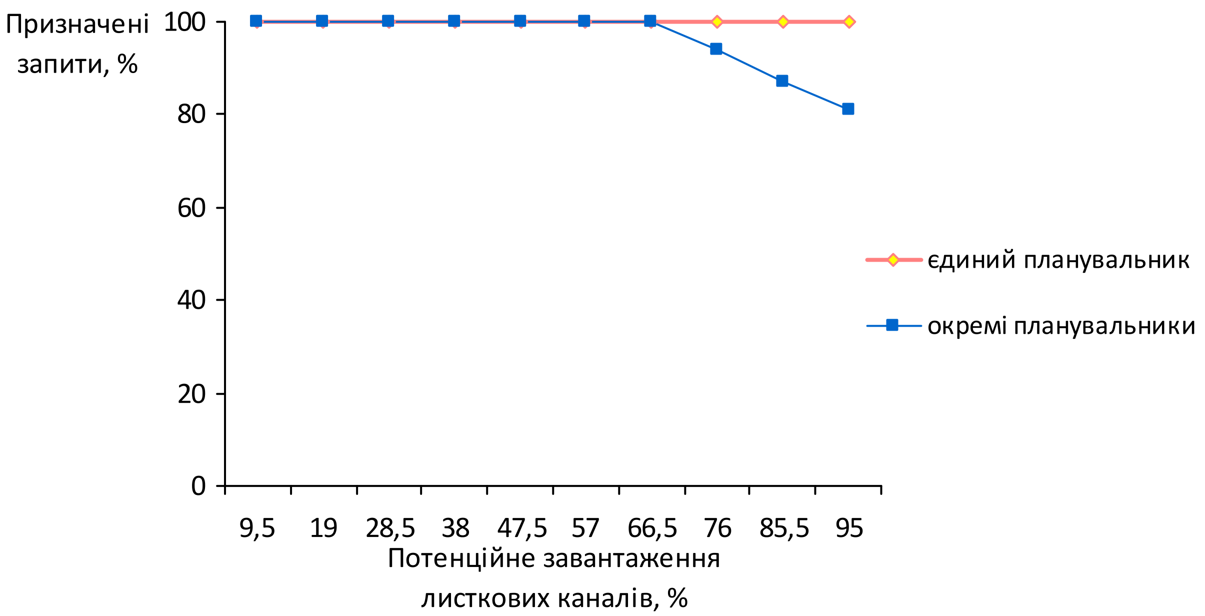


Рисунок 4.14 – Результати дослідження на даних першого класу

Дослідження алгоритмів на першому класі вихідних даних показали, що алгоритм з єдиним планувальником для всіх типів ресурсів дозволяє призначати запити в повному обсязі до 95% завантаження фізичних ресурсів

ЦОД. Алгоритм з окремими планувальниками для кожного типу ресурсів дає гірші результати в умовах, коли потенційне завантаження фізичних ресурсів ЦОД перевищує 70%.

Дослідження проводилися для алгоритмів з єдиним планувальником ресурсів і окремими планувальниками ресурсів. При проведенні дослідження глибина обмеженого перебору була обмежена значенням два. При збільшенні значення глибини алгоритми починають вести себе асимптотично однаково, проте час роботи поліноміально зростає.

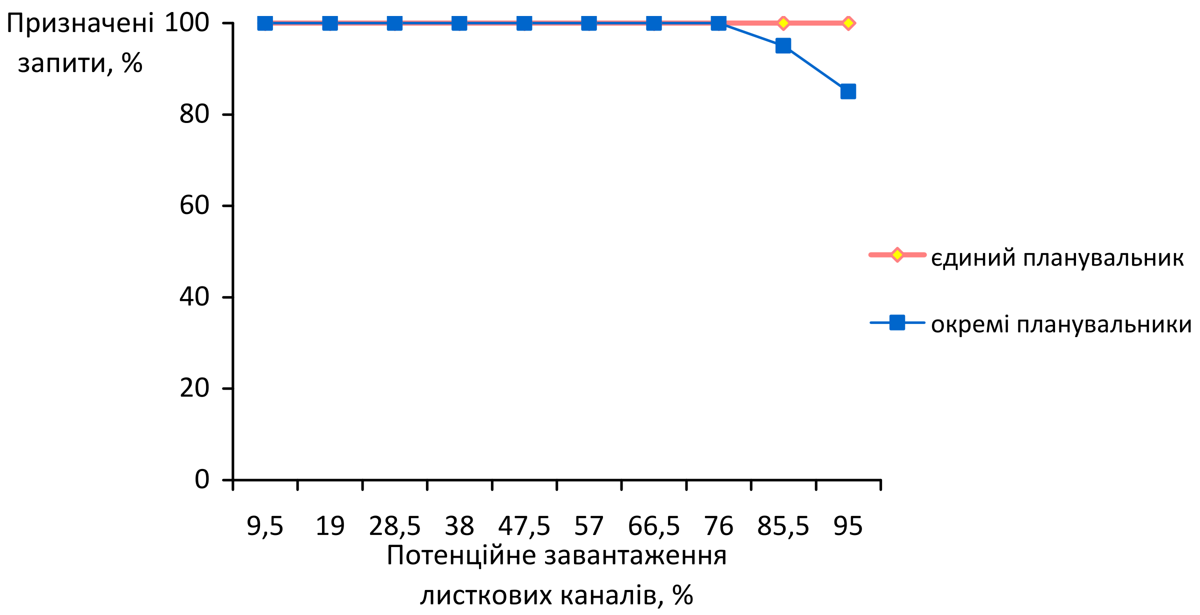


Рисунок 4.15 – Результати дослідження на даних другого класу

Дослідження алгоритмів на другому класі вихідних даних показали, що алгоритм з окремим планувальником для кожного типу ресурсів дозволяє призначати запити в повному обсязі аж до 95% завантаження обчислювальних вузлів і сховищ даних ЦОД. Алгоритм з єдиним планувальником для всіх типів ресурсів дає гірші результати в умовах, коли потенційне завантаження цих фізичних ресурсів перевищує 80%.

Результати порівняння алгоритмів наведені на рисунках 4.14 і 4.15.

# ВИСНОВКИ

Плюси і мінуси технології віртуалізації необхідно розглядати в контексті конкретної організації і тих завдань, які вона вирішує. Можливі ситуації, коли недоліки технології переважать її сильні сторони, але в світі швидко зростаючих даних це буде винятком з правил.

Обрано абстракції для опису ресурсних запитів і фізичних ресурсів центрів обробки даних. Розроблено математичну модель центру обробки даних, яка дозволяє описувати широкий клас їх архітектур. В рамках цієї моделі сформульована математична постановка задачі розподілу ресурсів, яка допускає міграцію віртуальних машин і реплікацію елементів зберігання даних. Наведено результати порівняння алгоритмів розподілу ресурсів для центрів обробки даних з єдиним планувальником для всіх типів ресурсів і для центрів обробки даних з окремими планувальниками для кожного типу ресурсів, а також результати порівняння розроблених алгоритмів з аналогічними алгоритмами платформи OpenStack.

Запропонована триетапна методика, яка на відміну від відомої двоетапної дозволяє більш повно врахувати реальні процеси розподілу ресурсів ЦОД.

Методи багатокритеріальної оптимізації, що використовуються на етапах методики, засновані на Парето-оптимальних рішеннях в порівнянні з використовуваними узагальненими критеріями на основі згортки, що дозволяє врахувати не тільки ваги критеріїв, але і їх взаємозв’язки.

Запропоновано підхід розподілу навантаження в хмарному обчислювальному середовищі, який ґрунтується на аналізі рівномірності використання ресурсів і прогнозі навантаження, який може надаватися при застосуванні інтелектуальних алгоритмів.

В результаті застосування механізму прогнозування зміни навантаженості хостів у хмарному обчислювальному середовищі і своєчасного перерозподілу ресурсів в динаміці функціонування застосунків всередині примірників вдалося досягти значного підвищення ефективності використання ресурсів в хмарному обчислювальному середовищі, підвищити здатність хмарного обчислювального середовища приймати нові екземпляри з мінімальним зниженням продуктивності вже функціонуючих програм. Так, використання запропонованого алгоритму дозволило зменшити втрату продуктивності примірників в тестовому хмарному середовищі в середньому на 12%.

Запропоновані в роботі підхід і алгоритми відображення ресурсних запитів на фізичні ресурси ЦОД мають такі особливості:

- допускають можливість міграції віртуальних ресурсів, що дозволяє усувати сегментацію фізичних ресурсів, яка може виникати в ході роботи ЦОД;
- допускають можливість реплікації сховищ даних з метою підвищення надійності зберігання даних і ефективності використання матеріальних ресурсів;
- допускають можливість розглядати всі типи ресурсів (обчислювальні ресурси, сховища даних, мережеві ресурси) як планований тип ресурсу. Це робить можливим завдання SLA для всіх типів ресурсів.

Експериментальні дослідження показали, що розроблені алгоритми дозволяють отримувати відображення ресурсних запитів на фізичні ресурси ЦОД, для яких можливо виконання заданих вимог SLA при завантаженні фізичних ресурсів до 95%.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Гультяев, А. К. Виртуальные машины: несколько компьютеров в одном [Текст]. – СПб. : Питер, 2006. – 224 с.
2. Li, X. Advanced Design and Implementation of Virtual Machines [Текст]. – CRC Press, 2017. – 465 p.
3. J. Xu and J. Fortes, "Multi-objective Virtual Machine Placement in Virtualized Data Center Environments" [Текст], Proceedings of the 2010 IEEE / ACM Conference on Green Computing and Communications, 179-188.
4. Y.-W. B. Chia-Hung Lien and /Vl.-B. Lin, "Estimation by software for the power consumption of streaming-media servers," [Текст] IEEE Trans, on Instrumentation and Measurement, vol. 56, no. 5, 2007.
5. Q. Tang, SKS Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous highperformance computing data centers: A cyber-physical approach", [Текст] IEEE Trans. Parallel Distrib. Syst, vol. 19, no. II, pp. 1458-1472, 2008.
6. K. Mills, J. Filliben and C. Dabrowski, "Comparing VM Placement Algorithms for On-Demand Clouds", [Текст] Proceedings of IEEE CloudCom 2011, Nov. 29-Dec. 1, 2011, Athens.
7. Ворожцов, А. С., Тутова, Н. В., Тутов, А. В. Оптимизация размещения облачных серверов в центрах обработки данных [Текст] // T-Comm – Телекоммуникации и транспорт, № 6. – 2015. – С. 4-8.
8. Pepple K. Deploying OpenStack. Sebastopol: O’Reilly, 2011 року. [Текст]
9. Amies A, Sluiman H., Tong QG et al. Developing and Hosting Applications on the Cloud [Текст]. Boston: IBM Press, 2012.
10. Urgaonkar B., Rosenberg AL, Shenoy P. Application Placement on a Cluster of Servers [Текст] // Intern. J. Foundations of Computer Science. 2007. T. 18. № 05. P. 1023-1041.

11. Bein D., Bein W., Venigella S. Cloud Storage and Online Bin Packing [Текст] // Proc. of the 5th Intern. Symp. on Intelligent Distributed Computing. Delft: IDC, 2011. P. 63-68.

12. Nagendram S., Lakshmi JV, Rao DV et al. Efficient Resource Scheduling in Data Centers Using MRIS [Текст] // Indian J. Computer Science and Engineering. 2011. V. 2. Issue 5. P. 764-769.

13. Arzuaga E., Kaeli DR Quantifying Load Imbalance on Virtualized Enterprise Servers [Текст] // Proc. of the First Joint WOSP / SIPEW Intern. Conf. on Performance Engineering. San Josa, CA: ACM, 2010. P. 235-242.

14. Mishra M., Sahoo A. On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach [Текст] // Cloud Computing (CLOUD), IEEE Intern. Conf. Washington: IEEE Press, 2011. P. 275-282.

15. Botero JF, Hesselbach X., Fischer A. et al. Optimal Mapping of Virtual Networks with Hidden Hops [Текст] // Telecommunication Systems. 2012. V. 51. № 4. P. 273-282.

16. Yu M., Yi Y., Rexford J. et al. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration [Текст] // ACM SIGCOMM Computer Communication Review. 2008. V. 38. № 2. P. 17-29.

17. Lischka J., Karl H. A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection [Текст] // Proc. of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures. Barcelona: ACM, 2009. P. 81-88.

18. Zhu Y., Ammar MH Algorithms for Assigning Substrate Network Resources to Virtual Network Components [Текст] // 25th Intern. Conf. on Computer Communications. Barcelona: INFOCOM, 2006. P. 1-12.

19. Chowdhury NMMK, Rahman MR, Boutaba R. Virtual Network Embedding with Coordinated Node and Link Mapping [Текст] // 28th Intern. Conf. on Computer Communications. Barcelona: INFOCOM, 2009. P. 783

20. Cheng X., Sen S., Zhongbao Z. et al. Virtual Network Embedding

Through Topology-aware Node Ranking [Текст] // ACM SIGCOMM Computer Communication Review. 2011. V. 41. № 2. P. 38-47.

21. Korupolu M., Singh A., Bamba B. Coupled Placement in Modern Data Centers [Текст] // IEEE Intern. Symp. on Parallel & Distributed Processing. NY: IPDPS, 2009. P. 1-12.

22. Singh A., Korupolu M., Mohapatra D. Server-Storage Virtualization: Integration and Load Balancing in Data Centers [Текст] // Proc. of the 2008 ACM / IEEE Conf. on Supercomputing. Austin: IEEE Press, 2008. P. 1-12.

23. Jiang JW, Tian L., Sangtae H. et al. Joint VM Placement and Routing for Data Center Traffic Engineering [Текст] // 31st Intern. Conf. on Computer Communications. Orlando: INFOCOM, 2012. P. 2876-2880.

24. Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment [Текст] – Zhen Xiao, Weija Song and Qu Chen – IEEE Transaction on Parallel and Distributed Systems (TPDS), 2013

25. Sanjib Mishra. Short Term Load Forecasting Using Computational Intelligence Methods [Текст], Department of Electronics and Communication Engineering National Institute Of Technology, Rourkela, 2008, с. 53-62.

26. Kostenko VA, Plakunov AV An Algorithm for Constructing Single Machine Schedules Based on Ant Colony Approach [Текст] // J. of Computer and Systems Sciences Intern. 2013. V. 52. № 6. P. 928-937.

27. Ahuja RK, Magnanti TL, Orlin JB Network Flows: Theory, Algorithms, and Applications [Текст]. New Jersey: Prentice Hall, 1993.