Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи управління розподіленим обчислювальним процесом в гетерогенних хмарних системах

(тема)

здобувач 2 року навчання,
групи СПзм-23-1
Євгенія БІТЮКОВА
(власне ім'я, прізвище)

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. Андрій БУГРІЙ
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (прізвище, ініціали)

2025 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)
"____" _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бітюковій Євгенії Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи управління розподіленим обчислювальним процесом в гетерогенних хмарних системах

затверджена наказом по університету від " 07 " квітня 2025 р. № 53стз

2. Термін подання студентом роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи

1. Технології управління розподіленими обчисленнями
2. Архітектури гетерогенних хмарних систем
3. Провайдери хмарних сервісів
4. Методи та моделі управління розподіленим обчислювальним

4. Перелік питань, що потрібно опрацювати в роботі

1 Аналіз предметної області
2 Методи управління розподіленим обчислювальним процесом в гетерогенних хмарних системах
3 Опис розробленого методу та експериментальні дослідження
4 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 14 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	Аналіз предметної області	08.04.25-18.04.25	
2	Розробка моделей	19.04.25-01.05.25	
3	Реалізація алгоритмів	02.05.25-10.05.25	
4	Розробка структури програмних засобів	11.05.25-21.05.25	
5	Розробка програмних модулів	22.05.25-02.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	03.06.25-05.06.25	
7	Подання кваліфікаційної роботи керівникові та її попередній захист	06.06.25-10.06.25	
8	Подання кваліфікаційної роботи на рецензування	11.06.25-12.06.25	

Дата видачі завдання 07 квітня 2025 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Андрій БУГРІЙ
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 56 с., 19 рис., 1 табл, 1 дод., 21 джерел.

ХМАРНІ ОБЧИСЛЕННЯ, ПЛАНУВАННЯ ЗАВДАНЬ, МЕТАЕВРИСТИЧНІ АЛГОРИТМИ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ОПТИМІЗАЦІЯ РЕСУРСІВ, МАСШТАБОВАНІСТЬ.

Робота присвячена вдосконаленню методів планування завдань у хмарних обчисленнях. Запропоновано двоетапний підхід, який передбачає створення віртуальних машин на основі кластеризації історичних даних та використання гібридного генетичного алгоритму мурашиних колоній для ефективного призначення завдань. Проведено експериментальне моделювання для оцінки ефективності запропонованого підходу. Результати показали значне скорочення часу виконання завдань, підвищення пропускної здатності та покращення масштабованості хмарних обчислень. Запропонований метод може бути корисним для покращення продуктивності та ефективності управління ресурсами у великих розподілених системах.

Запропоновані методи можуть бути використані для покращення роботи центрів обробки даних, зменшення часу очікування завдань та зниження витрат на обслуговування. Оптимізоване управління ресурсами сприятиме підвищенню ефективності обчислювальних систем у різних сферах, таких як наукові дослідження, фінанси, медицина та логістика.

ABSTRACT

Master’s thesis: 56 pages, 18 figures, 3 tables, 1 appendice, 21 sources.

CLOUD COMPUTING, JOB SCHEDULING, METAHEURISTIC ALGORITHMS, GENETIC ALGORITHM, RESOURCE OPTIMIZATION, SCALABILITY.

The work is devoted to improving task scheduling methods in cloud computing. A two-stage approach is proposed, which involves the creation of virtual machines based on historical data clustering and the use of a hybrid ant colony genetic algorithm for efficient task assignment. Experimental simulations are conducted to evaluate the effectiveness of the proposed approach. The results show a significant reduction in task execution time, increased throughput, and improved scalability of cloud computing. The proposed method can be useful for improving the performance and efficiency of resource management in large distributed systems.

The proposed methods can be used to improve the performance of data centres, reduce task waiting times, and reduce maintenance costs. Optimized resource management will contribute to increasing the efficiency of computing systems in various fields, such as scientific research, finance, medicine, and logistics.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ВМ – віртуальна машина

ОС – операційна система

ПЗ – програмне забезпечення

ШІ – штучний інтелект

ACO – Ant Colony Optimization

AIGA – Adaptive Incremental Genetic Algorithm

API – Application Programming Interfase

DIS – Distributed Interactive Simulation

DEVS – Discrete Event System Specification

IaaS – Infrastructure as a Service

EKS – Elastic Kubernetes Service

IDC – International Data Corporation

FCFS – First-Come First-Served

FPA – Pollination-Based Algorithm

GA – Genetic Algorithm

HAGA – Hybrid Ant Genetic Algorithm

HPC – High Performance Computing

PSO – Planning Scheduling and Optimization

SaaS – Simulation as a Service

VM – Virtual Machine

QoS – Quality of Service

ВСТУП

Розвиток хмарних технологій призвів до того, що більше ресурсів стає доступними. Нещодавній сплеск попиту на хмарні послуги вимагає подальшого вдосконалення хмарних центрів обробки даних. Як наслідок, для хмарних обчислень необхідне ефективне планування завдань. Щоб забезпечити рівномірний розподіл навантаження на системи з підвищеною масштабованістю та продуктивністю, центри обробки даних повинні мати відповідний механізм планування завдань. Ефективна стратегія планування завдань намагається оптимізувати результат, скоротити час відповіді, використовувати менше ресурсів і зберегти енергію шляхом узгодження відповідних ресурсів з робочим навантаженням.

Запропонована методика використовує двоетапний підхід до планування завдань. На першому етапі віртуальні машини створюються за допомогою методів класифікації та кластеризації на основі історичних даних завдань, а на другому етапі використовується гібридний генетичний алгоритм мурашок для планування найкращої віртуальної машини для завдання шляхом поєднання переваг генетичних алгоритмів із значеннями феромонів з алгоритмів мурашиних колоній. Запропонований підхід забезпечив економічно ефективне планування завдань із коротким терміном виконання.

Поява та швидкий розвиток хмарних обчислень інтенсивно змінили парадигму інформаційних технологій. Хмарні обчислення знайшли своє широке застосування в науці та техніці. Хмарні системи пропонують нові рішення для наукових експериментів і доступ до даних, розподілених між центрами обробки даних. Оптимізація управління хмарними ресурсами стає важливою сферою досліджень. Це дослідження зосереджено на плануванні завдань і обробці наукових даних у гетерогенному хмарному середовищі, де більшість завдань вимагає великої кількості ресурсів і значної

обчислювальної потужності. Наш підхід пропонує стратегію оптимізації планування завдань для різних віртуальних машин і центрів обробки даних на основі метаевристичного алгоритму Evolution Strategies. Як важлива властивість системи, масштабованість зіграла важливу роль у виборі алгоритму. Ми створили модель і додали політику брокера «Перша найдовша робота». Порівняно зі стандартним генетичним алгоритмом наш підхід показав покращення виміряних показників. Після тестування під різними навантаженнями запропонована стратегія дала багатообіцяючі результати та досягла кращого розмаху, більшого середнього використання ресурсів, кращої пропускної здатності, меншого середнього часу виконання, меншого ступеня дисбалансу та масштабованості.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Технології хмарних обчислень та задача розподілення завдань за ресурсами

Надання обчислювальних ресурсів на вимогу є метою галузі хмарних обчислень, яка швидко розвивається. Їх масштабоване, динамічне та адаптоване середовище є причиною значного зростання. Крім того, хмарні технології пропонують абстракції для віртуальних ресурсів і приховують технічні аспекти управління. Хмарні обчислення еволюціонували від традиційних хмарних обчислень протягом останніх кількох років, пропонуючи такі переваги, як послуги на вимогу та відкритий доступ для мобільних. Планування дозволяє найкращим чином розподілити ресурси між призначеними завданнями за обмежений проміжок часу, щоб досягти бажаного рівня обслуговування.

Алгоритм планування завдань займається вибором необхідних ресурсів, які забезпечують повне виконання завдань зі скороченим часом очікування та виконання. Він складається з набору вказівок і політик для розподілу завдань між відповідними ресурсами, такими як ЦП, пам’ять і пропускна здатність, щоб максимізувати продуктивність і використання ресурсів. Перший рівень планування завдань складається з набору завдань (званих хмарлетами), які надсилають користувачі хмари та які необхідні для виконання. Відповідальним є наступний рівень планування для розподілу завдань між відповідними ресурсами для досягнення найвищого рівня використання ресурсів із найкоротшим часом виконання.

Проміжок виконання – це загальна кількість часу, необхідна для виконання всіх завдань від початку до кінця. Когорта віртуальних машин, які використовуються для виконання завдань призначення, становить третій рівень віртуальних машин. Метаевристичні алгоритми використовуються в

інформатиці та математичній оптимізації для пошуку, генерації або вибору оптимального рішення (алгоритм часткового пошуку), яке може забезпечити адекватне вирішення проблеми пошуку, особливо коли дані відсутні або неповні, або обчислювальна потужність дуже мала. На відміну від мета-евристик, які занадто великі, щоб їх можна було повністю перерахувати чи досліджувати іншим чином, метаевристики беруть вибірку підмножини рішень. Найбільш використовуваними алгоритмами оптимізації для планування завдань є PSO та GA.

Складність у плануванні завдань полягає в плануванні в центрах обробки даних, вирішальній області для дослідження. Ефективні механізми планування завдань необхідні для балансування навантаження, масштабованості та продуктивності центрів обробки даних. Кілька додатків працюють одночасно в хмарних центрах обробки даних. Паралельне програмування неможливо реалізувати без належного розподілу ресурсів.

Неефективне планування призводить до тривалого часу виконання, величезних витрат і недовикористання доступних ресурсів, що все впливає на загальну продуктивність хмари. Використання ефективних алгоритмів планування дозволяє досягти кращих цілей використання ресурсів. Зіставляючи завдання за віртуальними машинами, алгоритми планування зменшують вартість і час виконання робочих процесів. Мета процедури планування полягає у створенні алгоритму, який визначає час і ресурс, за який буде виконано кожне завдання. У поточному середовищі хмарні обчислення стали найбільш широко використовуваною парадигмою розподілених обчислень. Якість обслуговування (QoS) для користувачів забезпечується хмарними обчисленнями. Вкрай важливо, щоб завдання були широко розподілені між доступними ресурсами, щоб забезпечити якість обслуговування користувачів.

Як результат, у системах хмарних обчислень процес планування вважається надзвичайно важливим. Оптимальне планування завдань у хмарі є важливою вимогою хмарних обчислень. За правильного планування може

спостерігатися втрата ресурсів, а також збільшення часу.

Оскільки хмарні обчислення працюють як модель оплати за користування, втрата ресурсів може означати, що ви платите за невикористання. Таким чином, найкращий алгоритм планування завдань покращує використання ресурсів і скорочує загальний час, необхідний для виконання завдання. Незважаючи на те, що для цього існує багато різних типів алгоритмів, не існує жодного існуючого алгоритму, який забезпечує оптимальне вирішення проблеми розподілу завдань і планування за поліноміальний час. Рішення, засновані на великому пошуку, неможливі через дорогий характер загального планування.

Метаевристичні методи, такі як PSO, вирішують ці проблеми, надаючи майже оптимальне рішення за короткий період часу. Для покращення продуктивності алгоритмів планування пропонується двоетапна стратегія. Модель класифікації завдань використовується на першому етапі для ідентифікації завдань відповідно до попередніх даних планування. Це робиться за допомогою кластеризації CLARANS, коли спочатку кластеризуються різні старі завдання в попередньо визначену кількість кластерів, а потім на їх основі створюююбся віртуальні машини за допомогою CloudSim. На наступному етапі завдання зіставляються з необхідними віртуальними машинами, а потім завдання зіставляються з конкретними типами віртуальних машин за допомогою метаевристичної техніки гібридного генетичного алгоритму (HAGA), щоб виконати завдання за мінімальний проміжок часу.

1.2 Обробка великих даних та провайдери хмарних послуг

Дані з їхніми джерелами, управлінням і якістю мають вирішальне значення в цифровому середовищі. Наука про дані та аналітика даних стають все більш складними. Дані є паливом для алгоритмів і багатьох цифрових процесів. Алгоритми дозволяють більш оптимально обробляти великі обсяги

даних. Технології великих даних передбачають використання високомасштабованої розподіленої інфраструктури для паралельної обробки, зберігання, передачі та доступу до віддалених ресурсів для спільної роботи великої кількості територіально віддалених дослідників. За оцінками International Data Corporation (IDC) ринок хмарних обчислень зростає в усіх регіонах світу [1].

Окрім провідних глобальних постачальників послуг та інфраструктури хмарних обчислень, Amazon, Microsoft і Google, існують національні та регіональні постачальники хмарних послуг. Зберігання даних на дислокованих серверах призводить до необхідності створення національної електронної інфраструктури в хмарі [2], [3] для досягнення економічної конкурентоспроможності, ефективного контролю даних та прискорення інновацій у науці та техніці. Мета полягає в тому, щоб максимально перемістити робоче навантаження в хмару, щоб забезпечити вимоги до надійності, продуктивності та ефективності. Це призводить до багатьох невідомих і майбутніх проблем у хмарі та IT [4], оскільки екосистема та складність послуг і даних зростають експоненціально.

Концепція неоднорідності у високорозподілених центрах обробки даних розвивалася паралельно з розвитком хмарних обчислень. Неоднорідність центру обробки даних проявляється в різноманітності потужностей ресурсів і можливостей для забезпечення належної якості обслуговування (QoS).

Через зростаючий інтерес необхідна інфраструктура високопродуктивних обчислень (HPC) забезпечується хмарними обчисленнями через центри обробки даних. Застосування та переваги цієї міждисциплінарної технології науки про дані різноманітні. При застосуванні підходу до аналізу даних слід враховувати кілька питань такі, як надійність, збір і зберігання даних. Прикладами використання є геноміка, фізика елементарних частинок і прогноз погоди.

Основною властивістю, яка визначає якість системи хмарних

обчислень, є масштабованість. Постійним завданням є досягнення вищої продуктивності та більшої масштабованості в управлінні даними та хмарною інфраструктурою з оптимізованим розподілом робочого навантаження. Адаптивне керування ресурсами в реальному часі є обов'язковим у великомасштабних розподілених архітектурах. Проблема полягає в тому, що деякі системи не масштабуються до обсягу даних, який запитує програма.

Іншою проблемою є визначення відповідних показників під час оцінки масштабованості хмарних інфраструктур. Масштабованість слід розуміти в контексті точних методів оптимізації. Можна підключити велику кількість різнорідних ресурсів до системи, продуктивність якої зростає майже лінійно та відповідає численним цілям оптимізації, застосованим до метрик QoS Cloud за допомогою відповідних алгоритмів. Нове використання методів штучного інтелекту (ШІ) і програмно-визначений підхід до управління ресурсами створює простір для швидкого та гнучкого налаштування та управління динамічними потребами робочих процесів. Ці технології спрямовані на оптимальне використання ресурсів для динамічно запитуваних послуг за допомогою інтелектуального та централізованого підходу.

Основне дослідницьке питання полягає в тому, як оптимізувати центроване управління ресурсами з урахуванням масштабованості в гетерогенних хмарних середовищах за допомогою підходу планування завдань. В роботі проводиться оцінка продуктивності та аналізу масштабованості систем гетерогенної хмарної інфраструктури, яка виконує ресурсомісткі завдання.

Алгоритм Evolution Strategies обраний через його властивість адаптуватися до вимог сучасних систем і різних навантажень, а також надає потенціал для інтелектуального управління ресурсами. Моделювання – це обраний підхід, який використовується для моделювання великомасштабного середовища та тестування нашого рішення. Він пропонує середовище, яке забезпечує хорошу гнучкість для передбачення поведінки системи та тестування алгоритмів у відтворюваний спосіб.

Пропонується новий підхід до розподілу завдань із використанням природного алгоритму Evolution Strategies, який досі не використовувався в області керування та оптимізації хмарних ресурсів. Алгоритм Evolution Strategies натхненний теорією еволюції та природного відбору, який використовує випадкову мутацію, рекомбінацію та відбір на основі значення функції відповідності та застосовує ці кроки до популяції індивідуумів, що містять рішення-кандидати, щоб розробити кращі рішення під час повторної процедури, тим самим алгоритм Evolution Strategies застосовується у розподілі завдань для віртуальних машин (VM). Іншою метою запропонованої моделі є оптимізація планування завдань через центральний брокер. Тому для оптимізації розподілу ресурсів додатково використовується політика «Спочатку найдовше завдання». Такий підхід обрано через складність властивостей задачі у створеному навантаженні. Брокер забезпечує найтриваліший пріоритет роботи при виконанні ресурсомістких завдань.

Надається широка оцінка продуктивності алгоритму та порівняння з найсучаснішим рішенням за допомогою хмарного симулятора. У роботі застосовано підхід до планування завдань і було досліджено використання алгоритму стратегій еволюції як альтернативи популярному генетичному алгоритму, найбільш використовуваній стратегії з тієї ж групи еволюційних алгоритмів. Результати алгоритму Evolution Strategies і політики Evolution Strategies with Longest Job First порівнюються з результатами генетичного алгоритму на основі показників оцінки продуктивності, як-от час виконання, середнє використання ресурсів, пропускна спроможність, ступінь дисбалансу, аналіз масштабованості, аналіз розподілу навантаження. Підходи на основі стратегій еволюції показали, що вони добре масштабуються з кількістю доступних віртуальних машин і демонструють кращі результати, ніж генетичний алгоритм.

Основний внесок цієї роботи можна підсумувати таким чином: модель, побудована на основі фактичних даних спостережень і реальної хмарної

системи для масштабованого використання ресурсів. Нова адаптивна метаевристика на основі алгоритму Evolution Strategies для проблеми планування завдань у гетерогенних хмарних центрах обробки даних. Оцінка на основі моделювання та аналіз продуктивності та масштабованості запропонованого алгоритму.

1.3 Проблема масштабованості гетерогенних хмарних систем

Масштабованість є основною проблемою проектування кожної системи. Всі компоненти системи тісно пов'язані між собою і спільно використовують апаратні ресурси, що при великому потоці даних і великій кількості користувачів ускладнює масштабування і створює проблеми з продуктивністю. Майбутні центри обробки даних вимагатимуть адекватної обробки в хмарі, щоб ефективно керувати новими проблемами, пов’язаними з масштабованістю щодо ефективного аналізу великих обсягів даних.

Масштабованість є часто використовуваним терміном у літературі, але не існує єдиного визначення масштабованості. Її слід вважати багатокритеріальною задачею оптимізації [5]. Запропоновано кроки для аналізу в контексті розробки програмного забезпечення, які також можуть бути застосовані для аналізу масштабованості системи. Ці кроки включають визначення критичних випадків використання, вибір репрезентативних сценаріїв масштабованості, визначення вимог до масштабованості, планування досліджень вимірювань, виконання вимірювань, оцінку даних і представлення результатів. На масштабованість можуть впливати різноманітні властивості та фактори. Тому важливо дослідити, як на нього впливає певний фактор у певному середовищі та в який момент. Горизонтальне масштабування та вертикальне масштабування є двома способами виконати масштабованість і досягти пропорційного зростання ємності відповідно до збільшення навантаження на систему. Горизонтальне масштабування досягається шляхом оновлення системи шляхом додавання

додаткових екземплярів сервера до існуючого середовища. При вертикальному масштабуванні (або масштабуванні) до існуючих вузлів додається більше обчислювальної потужності або пам’яті, або це передбачає перехід на сервер з більшою потужністю. Масштабування вимагає рішень для досягнення рівномірного навантаження. Якщо навантаження на систему зменшується, ємність слід зменшити (зменшити), щоб скоригувати експлуатаційні витрати. Метрики для вимірювання масштабованості гетерогенних і однорідних обчислень відрізняються. Зусилля для визначення масштабованості системи ґрунтувалися на показниках системного навантаження, продуктивності системи, використання ресурсів [6], ефективності та прискорення [7]. Однак при оцінці потенціалу масштабованості системи також необхідно враховувати інші атрибути. Це включає продуктивність, зручність використання, вартість, працездатність, надійність, безпеку, доступність, розширюваність і функціональність.

Масштабованість є важливою функцією хмари. Його слід розглядати з точки зору вимог до ресурсів та управління. Масштабованість є критично важливою нефункціональною вимогою до хмари, яку слід відрізняти від еластичності хмари. Масштабованість системи — це здатність системи підтримувати збільшення робочого навантаження шляхом використання додаткових ресурсів протягом деякого часу. При цьому еластичність – це здатність системи адаптуватися до змін робочого навантаження автономно в будь-який момент часу [8]. Функція хмарного автомасштабування забезпечує необхідний QoS [9]. Із запровадженням нових технологій у найважливіших частинах інфраструктури центрів обробки даних виникає необхідність аналізувати масштабованість у ще складніших середовищах. Досягнення масштабованості між декількома мережевими центрами обробки даних у Хмарі необхідно розглянути. Традиційні методи, які використовуються для ефективного керування хмарними системами, передбачають різні підходи до керування ресурсами, планування та евристики. Управління ресурсами в хмарних середовищах має забезпечувати механізми глобального планування,

локального планування, профілювання попиту, оцінки використання, ціноутворення, масштабування додатків, керування робочим навантаженням, керування хмарою та дослідження вимірювань [10].

На масштабованість впливає вибір алгоритму управління ресурсами для більш ефективного використання всіх можливостей сайтів і ефективнішого розміщення завдань. Управління ресурсами передбачає динамічний розподіл обчислювальних, мережевих ресурсів і ресурсів зберігання, необхідних для виконання програм.

Надання ресурсів і моніторинг ресурсів є лише частиною проблем управління ресурсами. Дані моніторингу з територіально розподілених ресурсів можуть служити для реагування в реальному часі та розподілу ресурсів там, де і коли потрібні ресурси. Надання має аналізувати аспекти неоднорідності, швидкості та обсягу згенерованих даних у часі.

Збір інформації про обчислювальну техніку, мережевий трафік, стан і хід виконання багатьох завдань, що виконуються одночасно, може допомогти оцінити масштабованість системи. Парадигма хмарних обчислень базується на віртуалізації, що застосовується на різних рівнях. Управління ресурсами базується на різних алгоритмах планування та метаевристиках для керування використанням спільних ресурсів.

1.4 Методи планування завдань за обчислювальними ресурсами

Методи планування завдань, які засновані на генетичних алгоритмах мурашиної колонії, пропонуються в світлі значної позитивної переваги зворотного зв’язку, яку має оптимізація мурашиної колонії (ACO) щодо швидкості конвергенції. Метод швидко знаходить ідеальне рішення та перетворює його на початковий феромон ACO, використовуючи можливості глобального пошуку генетичного алгоритму. Дослідження моделювання демонструють, що цей алгоритм працює краще, ніж генетичний метод і його ACO за тих самих обставин і навіть ефективніше у великомасштабних

середовищах.

Запропонований метод ефективний у середовищі хмарних обчислень. Проте на швидкість конвергенції суттєво впливає обраний батьківський феромон [11]. Інші автори запропонували адаптивний інкрементний генетичний алгоритм, який зменшує генетичну модель, для чого розбиває завдання на менші групи. Нова методика мутації застосована для автоматичного регулювання кросинговеру та частоти мутації. Запропонована модель показала підвищену ефективність у плануванні завдань, але вона не враховує аспект балансування навантаження віртуальних машин [12].

Також запропонована модель, яка викликає функцію планування через регулярні проміжки часу. Модель порівнює ресурси із завданнями та доступними ресурсами ітеративно, щоб отримати найкращий графік [13]. Планування є однією з ключових тем дослідження хмарних обчислень, яку необхідно розглянути для досягнення оптимальної продуктивності. Метою планування є призначення завдань ресурсам з оптимальним навантаженням для кожної віртуальної машини. Було виявлено, що метаевристичні методи забезпечують майже оптимальні рішення за розумний проміжок часу для проблем NP і забезпечують різні алгоритми хмарного планування, засновані на загальних мета-евристичних методах, а саме: оптимізація мурашиної колонії (ACO), генетичний алгоритм (GA) і оптимізація рою частинок (PSO).

Існує метод інтелектуального рою бактерій із визначенням пріоритетів завдань. Ця концепція скорочує час простою віртуальних машин, використовуючи хемотаксичну та фуражну поведінку бактерій. Запропонований алгоритм скорочує час виконання робочого процесу та зменшує енергію [14], поєднуючи генетичний алгоритм і оптимізацію колонії мурашок. Час виконання значно скорочено порівняно зі стандартним генетичним алгоритмом, а порушення SLA значно зменшені порівняно з генетичним алгоритмом (GA) та адаптивним інкрементним генетичним алгоритмом (AIGA) [15]. Було проведено аналіз планування завдань, і було помічено, що метаевристичний алгоритм планування використовувався для

оптимізації продуктивності алгоритмів планування завдань у середовищі хмарних обчислень. Для аналізу продуктивності розглядаються такі алгоритми, як генетичний алгоритм (GA), алгоритм на основі запилення (FPA), алгоритм оптимізації колонії мурах (ACO) і генетичний алгоритм на основі відбору турнірів (TS-GA) [16].

У роботі представлено нові підходи до економічно оптимізованої конфігурації хмарних ресурсів за допомогою виявлення аномалій, машинного навчання та оптимізація рою. Це комплексне рішення, яке працює в замкнутому циклі без будь-якого зовнішнього контролю чи ініціалізації, вивчаючи, як використовується система, і усуваючи аномальні обставини, які вони виникають. Експерименти показали, що за 10 місяців було досягнуто 85% економії [17]. Запроваджено двоетапну стратегію (класифікація завдань і планування), щоб максимізувати продуктивність планування завдань у хмарних середовищах. Щоб класифікувати завдання на основі даних попереднього планування, перший етап передбачає створення значної кількості віртуальних машин (ВМ) різних типів за допомогою моделі класифікатора завдань, заснованої на конструкції класифікатора Байєса. Це прискорює процес планування завдань під час створення віртуальних машин (VM).

На другому етапі завдання динамічно призначаються конкретним віртуальним машинам. У результаті запропоновано динамічний алгоритм планування завдань. Генетичний алгоритм працює краще, ніж підхід динамічного планування, незважаючи на те, що перший перемагає звичайні алгоритми, такі як Max-Min і Max-Min. Спочатку розробляються основні кроки, потім принцип алгоритму мурашиної колонії (ACO) і, нарешті, стратегія планування, призначена для скорочення тривалості виконання завдань і віртуальних машини. Використання алгоритму оптимізації мурашиної колонії та меншої кількості запущених віртуальних машин може збільшити використання центрів обробки даних, зберігаючи при цьому якість часу виконання завдань. Це значно зменшує кількість віртуальних машин,

знижує витрати центрів обробки даних і встановлює баланс між обсягом виробництва та якістю обслуговування порівняно зі звичайними жадібними алгоритмами [18].

Загальна архітектура системи запропонованої методології зображена на рисунку 1.1. Робота системи складається з двох основних етапів, які відбуваються один за одним. Двома етапами запропонованої методології є вдосконалене створення віртуальної машини для історичних типів завдань і розширений генетичний алгоритм з феромоном ACO.

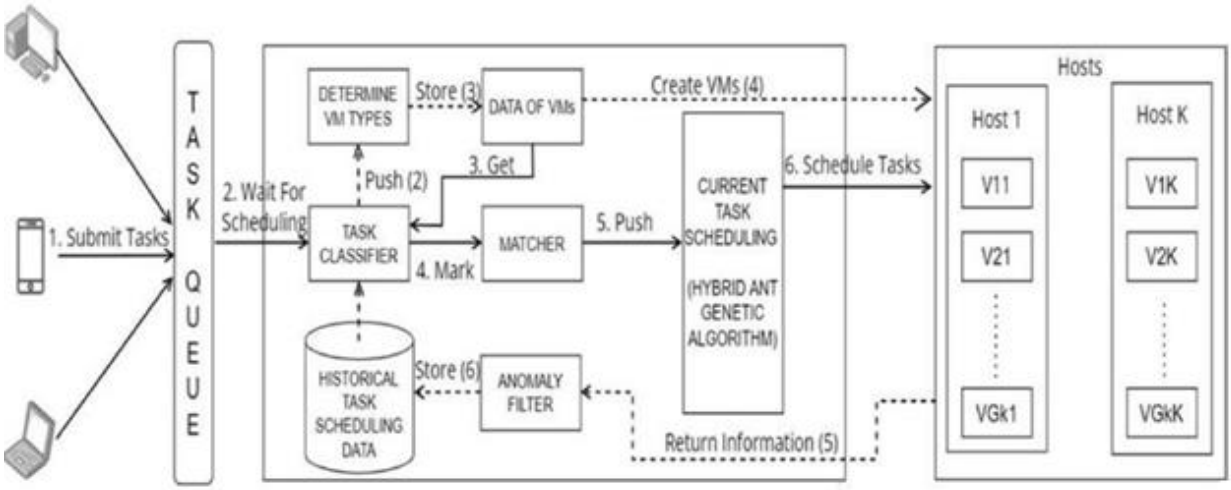

Рисунок 1.1 – Оптимізована двоетапна задача планування

На першому етапі вхідні завдання групуються для різних типів віртуальних машин, на основі яких віртуальні машини створюються на хостах. На другому етапі вхідні завдання класифікуються та зіставляються з відповідними типами віртуальних машин. Потім виконується планування завдань за допомогою метаевристичного алгоритму, який називається гібридним генетичним алгоритмом. Ці види алгоритмів, які допомагають розв'язувати NP-складні задачі. Після планування завдань дані знову зберігаються в наборі історичних даних, щоб їх можна було використати знову на першому етапі. Набір даних містить інформацію про завдання та виділений їм ЦП, а також виділену пам’ять.

1.5 Постановка мети та завдань дослідження

Підсумуємо основні недоліки існуючих рішень.

Висока складність обчислень – традиційні алгоритми потребують значних ресурсів, що обмежує їх застосування у великомасштабних системах.

Довгий час виконання завдань – існуючі методи не забезпечують швидке призначення ресурсів, що призводить до затримок.

Низька адаптивність – багато алгоритмів не враховують динамічні зміни в навантаженні, що знижує ефективність системи.

Обмежена масштабованість – сучасні підходи не завжди забезпечують ефективний розподіл ресурсів при зростанні навантаження.

Метою роботи є підвищення ефективності управління ресурсами у хмарних обчисленнях шляхом оптимізації процесу розподілу завдань, що дозволяє скоротити час виконання, покращити балансування навантаження та збільшити пропускну здатність системи.

Об'єкт дослідження: процеси управління розподілом завдань у хмарних обчислювальних системах.

Предмет дослідження: методи та алгоритми планування завдань для підвищення ефективності використання ресурсів у хмарних середовищах.

Завдання дослідження:

а) аналіз існуючих методів планування завдань у хмарних системах;

б) розробка двоетапного підходу для оптимізації розподілу завдань;

в) впровадження метаевристичних алгоритмів для покращення ефективності управління ресурсами;

г) оцінка продуктивності запропонованого методу через експериментальне моделювання;

д) порівняння отриманих результатів із традиційними алгоритмами планування.

2 МЕТОДИ УПРАВЛІННЯ РОЗПОДІЛЕНИМ ОБЧИСЛЮВАЛЬНИМ ПРОЦЕСОМ В ГЕТЕРОГЕННИХ ХМАРНИХ СИСТЕМАХ.

2.1 Моделі оцінки ефективності управління розподіленим обчислювальним процесом

Серед кількох характеристик для оцінки продуктивності та масштабованості виділимо наступні показники для вимірювання продуктивності нашої запропонованої моделі. Makespan – це часто використовувана метрика [19, 20] для ефективності планування на паралельних машинах щодо час, необхідний для виконання поданих завдань. Мінімізація makespan пов’язана з призначенням завдань віртуальним машинам (VM). Він визначається як максимальний час виконання (*Tmax*), необхідний для завершення завдань *j* на одній призначеній віртуальній машині *i*:

$$T^{VM_i} = \sum_{j=1}^{m} T_i^{task_j}, \text{де } i \in \overline{VM}, \tag{2.1}$$

$$T_{max}^{VM} = max\left(T^{VM_1}, T^{VM_2}, \ldots, T^{VM_n}\right). \tag{2.2}$$

Середнє використання ресурсів (U_R) є важливим показником ефективності використання ресурсів у хмарній системі. Оптимізація середнього використання ресурсів означає досягнення рівного балансу в хмарних центрах обробки даних. Цей кількісний показник розраховується за допомогою наведеного нижче рівняння. *n* позначає кількість ресурсів VM:

$$U_R = \frac{\sum_{i=1}^{n} T^{VM_i}}{T_{max}^{VM} \times n}. \tag{2.3}$$

Пропускна здатність (V) визначається як кількість завдань N, оброблених за одиницю часу (секунда). Мета полягає в тому, щоб максимізувати пропускну здатність, яка характеризує ефективність центрального процесору.

$$V = \frac{N}{T_{max}^{VM}}. \tag{2.4}$$

Середній час виконання (Average Execution Time – *Tavg*) – показник ефективності використання ресурсів у хмарній системі. Це час, витрачений на виконання завдання при активному використанні ресурсу віртуальної машини як хмари. Цей кількісний показник розраховується за допомогою наступного рівняння. *Tavg* позначає час виконання завдання j, а m позначає кількість завдань.

$$T_{avg} = \frac{\sum_{j=1}^{m} T^{task_j}}{m}. \tag{2.5}$$

Ступінь дисбалансу (D), згідно з наведеною нижче формулою, є параметром, який впливає на ефективність балансування навантаження між віртуальними ресурсами. Формула базується на часі виконання на всіх віртуальних машинах:

$$D = \frac{T_{max} - T_{min}}{T_{avg}}. \tag{2.5}$$

Останнім параметром можна визначити кількість віртуальних машин, які використовувались протягом певного часу з урахуванням часу обслуговування завдань та часу простою.

2.2 Аналіз попередніх запусків пакетів завдань за допомогою генетичного алгоритму

Перший етап запропонованої методології пов’язаний із завчасним створенням необхідних віртуальних машин для різних типів завдань, щоб зменшити затримку у виконанні. Він застосовує три кроки, а саме попередню обробку, кластеризацію та створення віртуальної машини.

Попередня обробка використовує історичні дані про попередні виконання завдання, які містять деталі наданого процесора та наданої пам’яті з набору даних трасування кластера. На цьому етапі виконується попередня обробка набору даних для отримання лише необхідних стовпців даних.

Кластери завдань отримують шляхом застосування алгоритму кластеризації CLARANS до попередньо оброблених даних. Цей алгоритм є більш ефективним, ніж найпопулярніший алгоритм k-середніх, оскільки він не надає значення викидам (тобто точкам даних, які є незвичними). Оптимальні параметри для алгоритму знаходяться за допомогою алгоритму на рисунку 2.1.

```
INPUT: Range values
OUTPUT: Optimal parameters
max_score ← zero
Initialize a variable key
for each k and v in cluster object items
    if v [0] >max_score
        max_score ← v [0]
        key ← k
end for
print key //Optimal parameters
print max_score //Calinski Harabasz Score
```

Рисунок 2.1 – Алгоритм пошуку оптимальних параметрів та їх значень

Даними для кожного кластера є дані, необхідні для створення віртуальної машини.

Другий крок передбачає створення віртуальних машин, параметри яких

отримані на попередньому етапі кластеризації. Віртуальні машини створюються на хостах у середовищі CloudSim за допомогою Eclipse. Ці віртуальні машини попередньо створені, тому немає потреби в будь-якому додатковому створенні віртуальної машини кожного разу під час виконання хмарлета.

Класифікатор завдань виконує свої завдання в два етапи. На першому етапі він аналізує історичні дані планування завдань і визначає кількість типів віртуальних машин, які відповідають конкретним завданням, застосовуючи наївний класифікатор Байєса. Другий етап зосереджується на зіставленні поданого завдання з найбільш підходящим типом віртуальної машини. Наївний класифікатор Байєса можна використовувати для виконання класифікації наборів даних, як показано на рисунку 2.2.

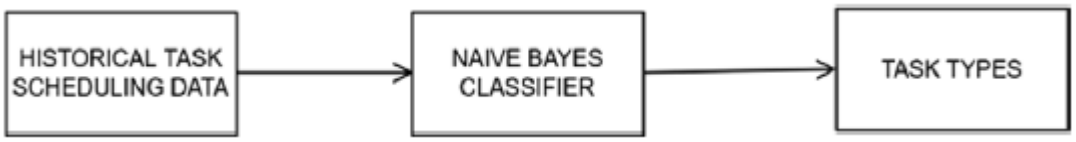

Рисунок 2.2 – Розробка модуля класифікатора завдань

Розширений генетичний алгоритм із феромоном ACO полягає у наступному. Оскільки завдання надходять динамічно, усі вони впорядковуються в чергу завдань. Класифікатор відокремлює спеціальні завдання від звичайних завдань і спочатку отримує спеціальні завдання, а потім звичайні завдання. Потім він визначає віртуальні машини, які підходять для вибраних завдань. Брокер порівнює позначені завдання з конкретними віртуальними машинами потрібного типу, пропускаючи їх. Вибрані віртуальні машини використовуються для планування та перенесення завдання. Деталі поточного планування надсилаються назад до активного планувальника завдань і пізніше архівуються як історичні дані. Результати повертаються користувачам, як показано на рисунку 2.3.

Алгоритм планування завдань (брокер) – найважливіша частина, де

реалізовано запропонований генетичний алгоритм з гібридним мурашиним алгоритмом. Цей процес поєднує як принципи генетичного алгоритму, так і оптимізацію мурашиної колонії.

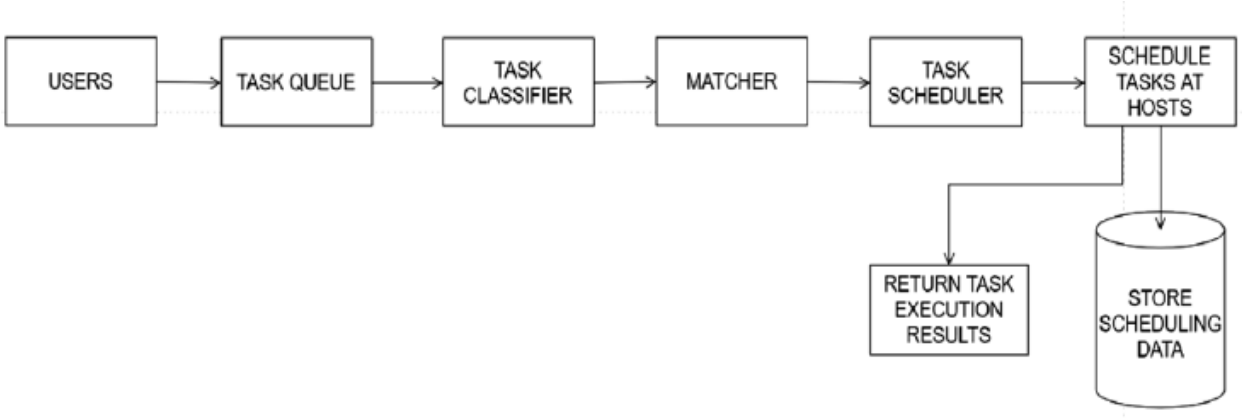

Рисунок 2.3 – Покращений генетичний алгоритм

Він включає ініціалізацію популяції, відбір, кросинговер, мутацію, додавання феромонів і випаровування. Структура модуля представлена на рисунку 2.4.

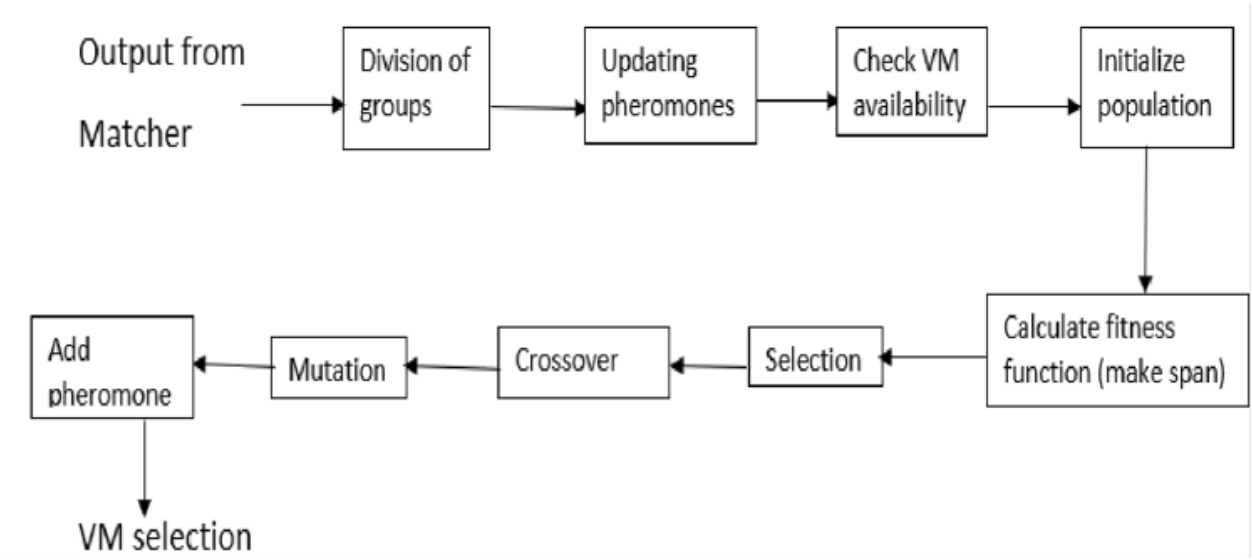

Рисунок 2.4 – Структура модуля планування завдань

На вхід поступає пара Task-VM. Результат роботи: остаточні пари Task-VM після оптимізації.

З попереднього кроку отримано пару завдання-ВМ, яка повідомляє нам

про те, якій ВМ відповідатиме завдання. Проблема тут полягає в тому, що багато завдань можуть відповідати певній віртуальній машині, і, отже, балансування навантаження виконується неправильно. Деякі машини можуть працювати протягом великого проміжку часу, тоді як інші можуть працювати дуже обмежено. Далі описано кроки, які виконуються під час впровадження алгоритму.

У таблиці 2.1 показано завдання та номер віртуальної машини, з яких ми отримуємо пару завдання-VM. Кодування – це процес, за допомогою якого завдання та пари віртуальних машин надаються як вхідні дані для алгоритму планування завдань. Після процесу кодування ми створюємо популяцію з багатьох особин. Для цього використовується алгоритм, що зображено на рисунку 2.5 для ініціалізації популяції.

Таблиця 2.1 – Опис завдання

Частота процесора (МГц)	Обсяг пам'яті (Mb)	Кількість циклів	Кількість інструкцій (MIPS)	Номер комп'ютера
1024	900	30	170	1
1024	300	15	255	2
850	150	125	2040	4
256	300	100	320	3
512	900	50	51	2
2048	50	20	20	5
128	412	10	482	5
450	2000	5	3021	6
50	300	20	450	4

Селекція і мутація застосовуються в кожній ітерації генетичного алгоритму для отримання нового рішення. За допомогою розрахунку

значення придатності оцінюється ефективність нового рішення, щоб оцінити його життєздатність. Алгоритм на рисунку 2.5 використовує час створення діапазону для визначення значення придатності.

```
Input: Task data, VM data
Output: Fitness value of individuals
1  Initialize new list, Execution time
2  for i=0 to task data size do
3      find Execution time and add to list
4  end for
5  Initialize a Map, BusyTime
6  for i=0 to task data size do
7      Put (existing busy time of ith VM + execution time of task i)
8  end for
9  fitness ← 0
10 for all values in BusyTime do
11     fitness ← maximum(values)
12 end for
```

Рисунок 2.5 – Функція відбору генетичного алгоритму

Під час відбору вибираються найбільш придатні рішення, а інші відхиляються. Доступні різні стратегії вибору, включаючи турніри, рулетку та сортування. Вибір сортування використовується для вибору двох найбільш підходящих особин для подальшої обробки. Використовується алгоритм на рисунку 2.6 для виконання цього вибору.

```
Input: fitness value of individuals
Output: Fittest / Selected individuals
1  maxfit←MAX_VALUE
2  for i← 0 to individuals.length do
3      if maxfit >= ith individual fitness
4          maxfit= ith individual fitness
5          maxfitindex=i
6      end if
7  end for
```

Рисунок 2.6 – Функція селекції генетичного алгоритму

Запропонований алгоритм використовує техніку одноточкового кросинговеру для генерації нових нащадків із тих, що були обрані на етапі відбору. На цьому кроці вибирається випадкова точка в найкраще підігнаній особині та змінюється їх вміст до цієї точки з другим найкраще підігнаним варіантом. Одноточковий кросинговер називається так тому, що використовується лише одна точка. Етапи реалізації цього процесу показано в алгоритмі на рисунку 2.7.

```
Input:  Random Crossover Point
Output: Cross overed Individuals
1   for i←0 to crossoverPoint do
2       swap(fittestGenes[i], secondFittestGenes[i])
3   end for
```

Рисунок 2.7 – Функція одноточкового кросинговеру

Процес мутації призводить до утворення нових характеристик у хромосомах. Запропонований алгоритм використовує стратегію мутації обміну для пошуку нових характеристик. Обираються дві точки та обмінюються їх значення, якщо це завдання можна виконати на віртуальній машині іншого вибраного завдання. Детальні кроки для цього представлені в авлгоритмі на рисунку 2.8.

```
Input: fittest individuals
Output: mutated individual
1   Mutated ← false
2   for i← 0 to geneLength and !mutated do
3       for j ←i+1 fittest geneLength and !mutated do
4           firstSet ← TaskVMdata(i)
5           SecondSet ← TaskVmdata(j)
6           if firstSet contains fittestgenes(j) && secondSet contains fittestgenes(i) then
7               swap fittestgenes of i and j
8               mutated ← true
9           end if
10      end for
11  end for
```

Рисунок 2.8 – Функція мутації

Два цикла програми забезпечують повний обхід варіантів мутації.

На останньому етапі працює алгоритм (рисунок 2.9), який додає феромони для відслідкування шляхів завдань.

```
Input: task-VM pair
Output: pheromone values
1   for i=0 to gene length do
2       Calculate executionTime[]
3       Find index of pheromone w.r.t fittest gene
4       Pheromone[i]+= executionTime[i]
5   end for

Input: Pheromone values
Output: Evaporated Pheromone value
        1     for i<- 0 to pheromone.length do
        2        pheromone[i]-=(pheromone[i]/partial_taskdata_size)
        3     end for
```

Рисунок 2.9 – Додавання та випаровування феромонів

3 ОПИС РОЗРОБЛЕНОГО МЕТОДУ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Метаевристичний метод управління розподіленими обчисленнями

Метаевристичні алгоритми не залежать від проблем і дають хорошу продуктивність для задач оптимізації високого рівня з різних областей, як у наукових дослідженнях, так і в практичній реалізації. Часто використовувана гібридна метаевристика поєднує метаевристичні алгоритми з ідеями та компонентами інших алгоритмів. У цьому підрозділі пропонується метаевристичний метод з урахуванням неоднорідності, заснований на принципах алгоритму Evolution Strategies, для вирішення проблеми планування завдань у хмарних обчисленнях.

Алгоритм Evolution Strategies спрямований на підвищення ефективності шляхом вибору відповідних кандидатів із цільовою функцією з домену. Тобто в цьому домені вибрані існуючі відповідні ресурси для прийняття обробки завдань специфічних характеристик. Цілі оптимізації полягають у мінімізації тривалості створення, максимальному використанні ресурсів і виконанні всіх завдань, призначених різнорідним хмарним ресурсам через ітеративний процес стратегій розвитку.

Стратегії еволюції – це метаевристичний алгоритм на основі популяції. Техніка Evolution Strategies використовує ідеї процесу біологічної еволюції, природного відбору та розмноження. Еволюція є результатом природного відбору, коли виживають лише ті особини, які найкраще демонструють боротьбу за життєві ресурси. Цей метод глобальної оптимізації належить до області еволюційних обчислень. Сфера еволюційних обчислень — це гілка штучного інтелекту, яка використовує групу еволюційних алгоритмів, до якої належать алгоритм стратегій еволюції та генетичний алгоритм. Обидва методи, швидше за все, знайдуть рішення, близьке до оптимального, і мають

застосування в багатьох різних областях. Основна відмінність між генетичним алгоритмом і алгоритмом стратегії еволюції полягає у використанні механізмів відбору.

Алгоритми Evolution Strategies використовують відбір і оператор мутації, щоб отримати загальне рішення ttest, тоді як генетичні алгоритми використовують відбір, кроссовер і мутацію для пошуку оптимального розподілу.

Популяція в багаточленному алгоритмі Evolution Strategies *(μ, λ)* складається з μ батьків, які після процесу мутації дають потомство λ з умовою $\lambda > \mu$. Кожен із λ нащадків має свій рівень відповідності. Використовуючи оператори мутації, створені λ діти формують популяцію наступного покоління. Оператор мутації змінює вибране рішення для створення нащадків і таким чином підтримує різноманітність популяції. Нащадки створюються шляхом випадково згенерованої зміни в батьківському організмі. Для переходу в нове покоління спостерігають потомство. Від них окремі особини передаються наступним поколінням. Кожна ітерація представляє крок покоління. Під час процесу оцінювання особи для наступної ітерації алгоритму оцінюються шляхом застосування функції фітнеса, яка визначає, наскільки оптимальним/придатним є рішення. Порівнюються числові значення цієї функції для кожної особини, які описують придатність особин. Тому в *(μ, λ)* - стратегіях еволюції особини виживають лише одне покоління. Ця процедура триває до тих пір, поки не буде виконана умова зупинки на задану кількість поколінь.

Прагнучи знайти та зберегти оптимальне рішення для кожного покоління, треба додати принцип елітарності до існуючих основних кроків методу Evolution Strategies.

Реалізація підходу *(μ, λ)* - еволюційних стратегій до розподілу завдань містить такі етапи.

Етап 1. Ініціалізація популяції. Метод, який використовується для ініціалізації популяції – це випадкова ініціалізація, що генерує випадкові

рішення, які прагнуть до оптимальності популяції. Популяція визначається як пари хмарлетів (зчитаних із файлу робочого навантаження) і віртуальних машин. Кожен елемент сукупності є вектором, який містить n параметрів $X = (x_1, x_2, x3, \ldots, x_n)$, де $x_i = (task, VM)$, $x_i \in X$.

Етап 2. Оцінка популяції. Кожен індивід популяції оцінюється за допомогою функції допасованості (F), яка враховує ємність віртуальних машин і довжину хмарлетів, визначених як

$$F = \frac{Length(task)}{Speed(VM)}. \quad (2.1)$$

Етап 3. Селекція. μ батьки вибираються з популяції для подальшого процесу створення потомства для кожного обраного батька.

Етап 4. Мутація. Хмарлети унікальні, і VM можуть відрізнятися. Для створення нащадків використовуємо оператор мутації випадкового скидання, де випадкова віртуальна машина з набору допустимих значень призначається випадково вибраному хмарлету. У цьому дослідженні мутації відбуваються в 10% окремої одиниці популяції. Вибираючи параметри для алгоритму Evolution Strategies, ми провели попереднє дослідження та протестували різні комбінації параметрів. Ми застосували правило 1/7 до співвідношення батьківських нащадків, оскільки воно дало оптимальні результати порівняно з іншими перевіреними комбінаціями параметрів пари «батьківські нащадки». Відповідно до попередніх досліджень це співвідношення є оптимальним і рекомендованим для підтримки високого селективного тиску.

Етап 5. Елітарність. Принцип елітарності використовується для того, щоб якомога довше зберегти в популяції найбільш оптимальне рішення-кандидат. Метод елітарності дозволяє зберігати та передавати тестові рішення з кожної сукупності.

Наприкінці алгоритму загальним найкращим рішенням є тестове рішення з кожного покоління, а хмарлет призначається вибраній віртуальній машині.

На рисунку 3.1 представлено алгоритм відповідно до розробленого методу.

Input: $(\mu, \lambda, \text{GenerationSize}, \text{CloudletList}, \text{VirtualMachineList})$
Output: S_{best} (Mapping of cloudlets to appropriate VMs)

```
1: begin
2: Population = InitializePopulation(Random, pairs < Cloudlet, VM >)
3: EvaluatePopulation(Population)
4: S_best = FindBest(Fitness)
5: add S_best to BestSolution
6: while ¬StopCondition(GenerationSize) do
7:     for i = 0 to λ do
8:         Parent = SelectParent(Population, μ)
9:         Child_i = Mutate
10:        add Child_i to Children
11:    end
12:    EvaluatePopulation(Children)
13:    S_best = FindBest(Fitness)
14:    add S_best to BestSolution
15:    Population = Children
16: end
17: return best S_best from BestSolution
```

Рисунок 3.1 – Псевдокод алгоритму створення схеми призначення

Список хмарлетів надсилається центральному брокеру для розповсюдження в центрах обробки даних і прив’язки хмарлетів до призначених віртуальних машин. Політику посередника планування «Найдовше завдання спочатку» було застосовано для оптимізації балансування навантаження та керування використанням ресурсів для обробки інтенсивних обчислювальних завдань.

На рисунку 3.2 показана діаграма метаевристичного алгоритму планування завдань Evolution Strategies із політикою брокера центру обробки даних «Спочатку найдовша робота». Перевірка підходів на основі стратегій еволюції пояснюється в наступному розділі.

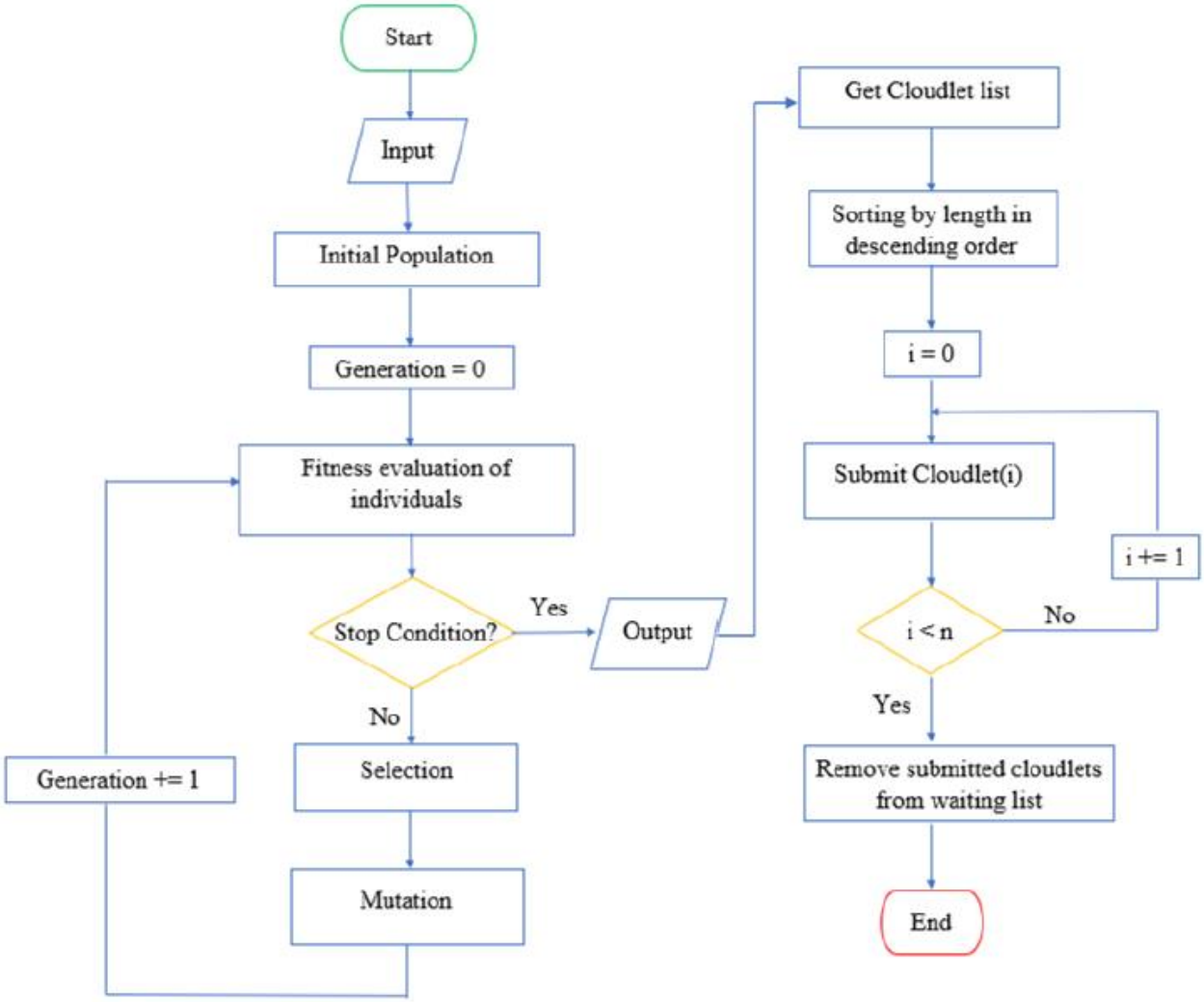

Рисунок 3.2 – Блок-схема планування завдань на основі стратегій розвитку та політики брокера «Спочатку найдовшої роботи»

Цю політику невипереджувального балансування навантаження було адаптовано для визначення пріоритетів процесів, які мають більшу довжину хмарлету. Cloudlets сортуються в порядку спадання попередньо призначеної довжини інструкцій і надсилаються до віртуальних машин, визначених процедурою Evolution Strategies. Ця централізовано визначена політика балансування навантаження «Спочатку найдовше завдання» максимізує використання системи за високих навантажень.

3.2 Експериментальні результати, аналіз ефективності та масштабованості

У цьому підрозділі показано емпіричні результати запропонованого нами алгоритму Evolution Strategies під час вирішення проблеми планування завдань у гетерогенному середовищі хмарних обчислень.

Підхід складається з двох наступних реалізацій, заснованих на принципах алгоритму *(μ, λ)* - Evolution Strategies:

- планування завдань Evolution Strategies;
- планування завдань Evolution Strategies із балансуванням навантаження за найдовшу роботу.

Щоб більш точно виміряти продуктивність запропонованого нами підходу, ми застосували моделювання з набором даних, описаним у розділі 2. Крім того, продуктивність запропонованих алгоритмів порівнювалася з генетичним алгоритмом, більш часто використовуваним алгоритмом з тієї ж групи алгоритмів. Основними цілями запропонованого алгоритму для планування завдань і балансування навантаження в контексті розглянутих метрик є покращення використання та розподілу хмарних ресурсів, зменшення часу створення, затримки та неузгодженості між завантаженнями ресурсів ВМ. Було виконано тести з різною кількістю завдань (1000, 5000, 10000, 15000 і 20000 завдань), щоб оцінити продуктивність і масштабованість запропонованого метаевристичного алгоритму. Кожну політику оцінювали 10 разів і дали середні результати. Результати трьох алгоритмів порівнювали на основі наступних показників оцінки ефективності, представлених у розділі 2:

- час виконання;
- середнє використання ресурсів;
- пропускна здатність;
- середній час виконання;
- ступінь дисбалансу;

- аналіз масштабованості;

- аналіз розподілу навантаження.

Наш алгоритм враховує характеристики завдання та вибирає хост із найбільш відповідним екземпляром віртуальної машини. Наступні рисунки візуалізують показники ефективності для кожної політики.

Час виконання представляє різницю в часі між часом початку та завершення списку завдань. Зменшений робочий діапазон свідчить про кращу продуктивність. Графік на рисунку 3.3 показує значення часу виконання для різних завдань.

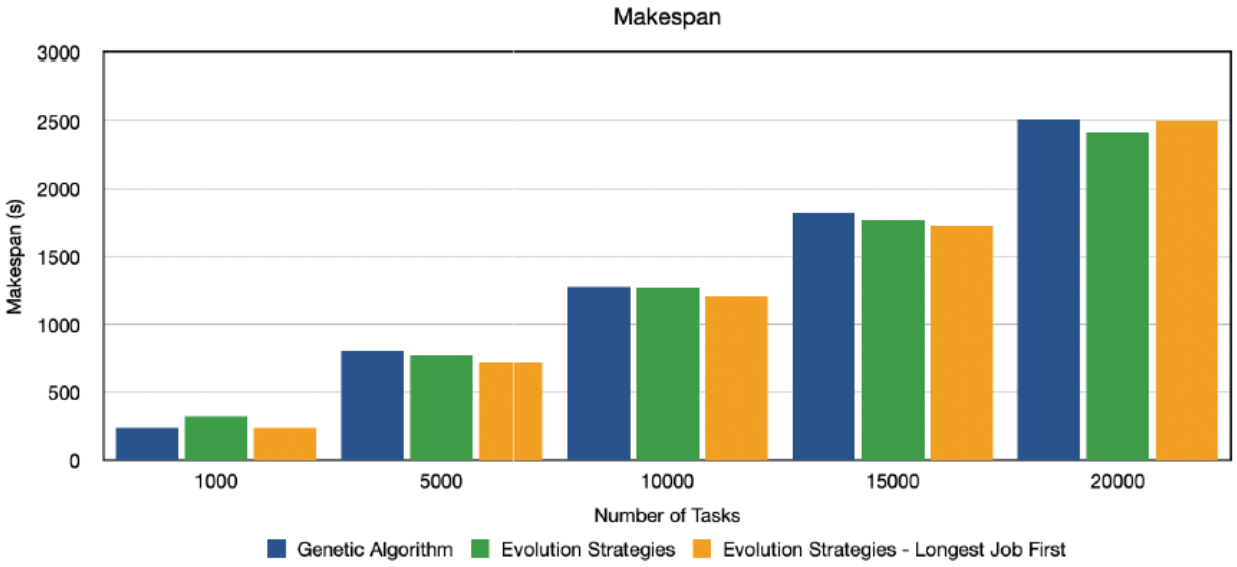

Рисунок 3.3 – Час виконання завдань

На рисунку показано, що алгоритми на основі стратегій еволюції перевершують генетичний алгоритм. Порівняно з двома політиками підхід «Стратегії еволюції з найдовшою роботою спочатку» показує кращі результати загалом із найнижчим значенням тривалості роботи.

Значення вказують на очікуване збільшення загального робочого часу через збільшення робочого навантаження.

Порівняльний аналіз використання ресурсів між Evolution Strategies і найсучаснішим алгоритмом показано на рисунку 3.4. Використання ресурсів показує доступність ресурсів і є хорошим показником загальної ефективності

використання ресурсів. Підходи стратегій еволюції перевершують генетичний алгоритм, при цьому стратегії еволюції з найдовшою роботою спочатку мають вищу цінність використання ресурсів, ніж інші підходи.

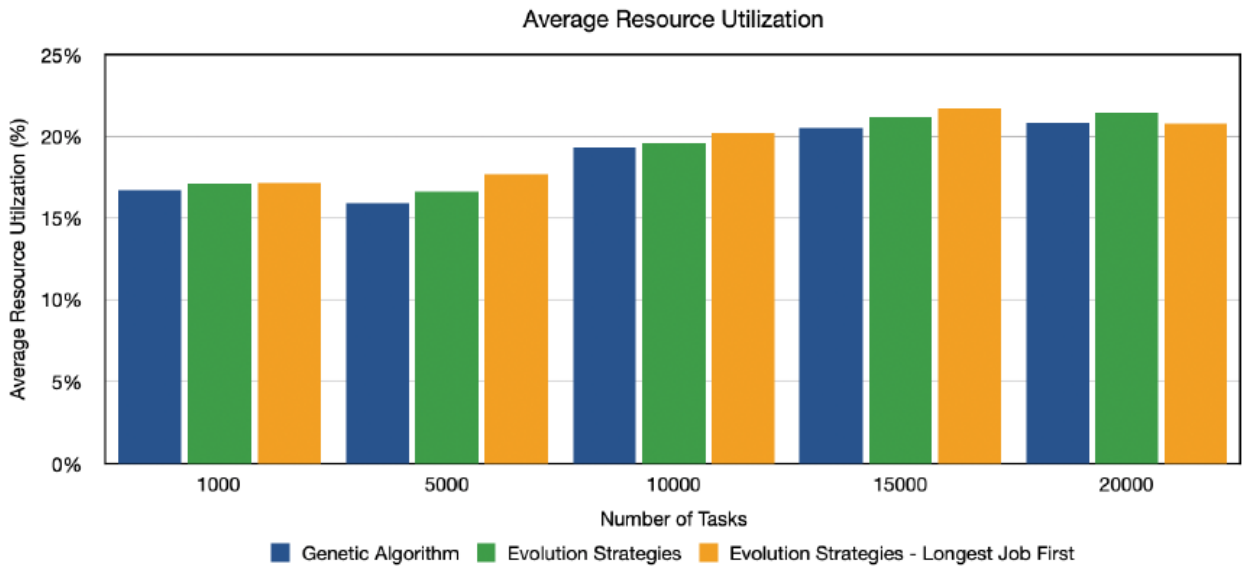

Рисунок 3.4 – Середнє використання ресурсів

Спостережувана система центру обробки даних у хмарі має велику ємність ресурсів і призначена для використання в різноманітних наукових дослідженнях. Такі відсотки відповідають призначенню центрів обробки даних, де використовувані завдання становлять лише частину навантаження. Можна помітити, що значення використання ресурсів зростає з більшою кількістю завдань.

Пропускна здатність є вирішальним показником для оцінки та порівняння продуктивності запропонованого методу. Це важливий атрибут, який слід враховувати під час тестування та аналізу масштабованості при різних навантаженнях. На рисунку 3.5 показано досягнуту пропускну здатність для всіх порівнюваних політик. Результати свідчать про те, що запропоновані модифікації методів на основі генетичних алгоритмів та підходи стратегій розвитку, зокрема стратегії розвитку з підходом «найдовша робота спочатку», покращили пропускну здатність для кожного тесту в експерименті.

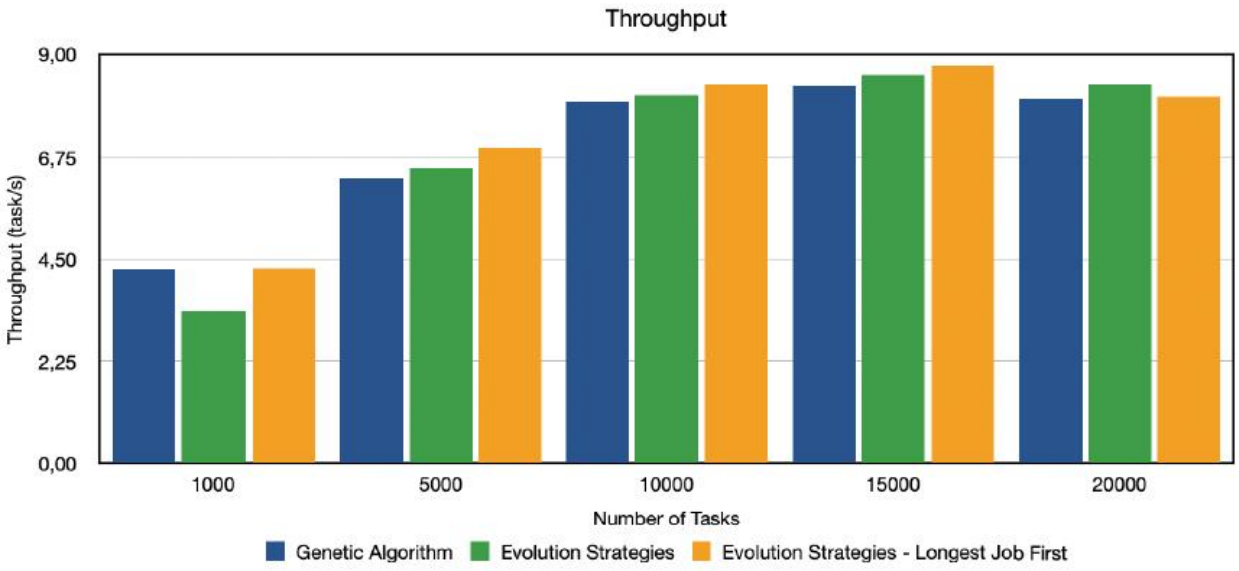

Рисунок 3.5 – Пропускна здатність системи

Параметри, пов’язані з часом, мають значний вплив на забезпечення високої продуктивності для користувачів хмарними системами. Наприклад, рисунок 3.6 не показує значних відхилень у середньому часі виконання хмарлетів, виміряному в секундах, між підходами генетичного алгоритму та стратегій еволюції. З збільшенням розмірності завдань, запропонований алгоритм починає показувати кращі результати, тому його ефективно використовувати на задачах великого розміру.

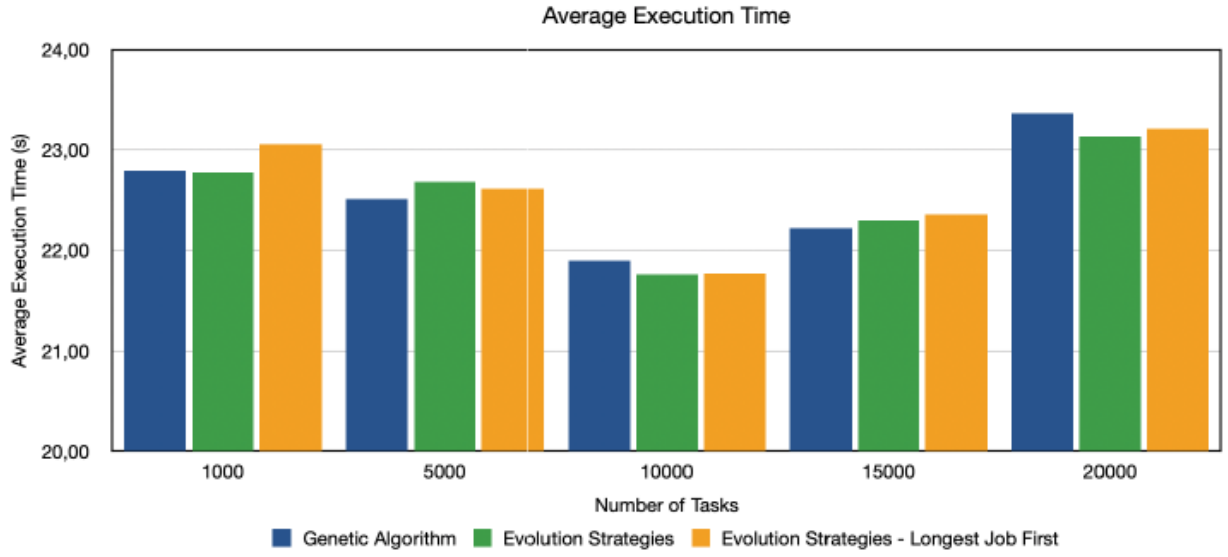

Рисунок 3.6 – Середній час виконання завдань

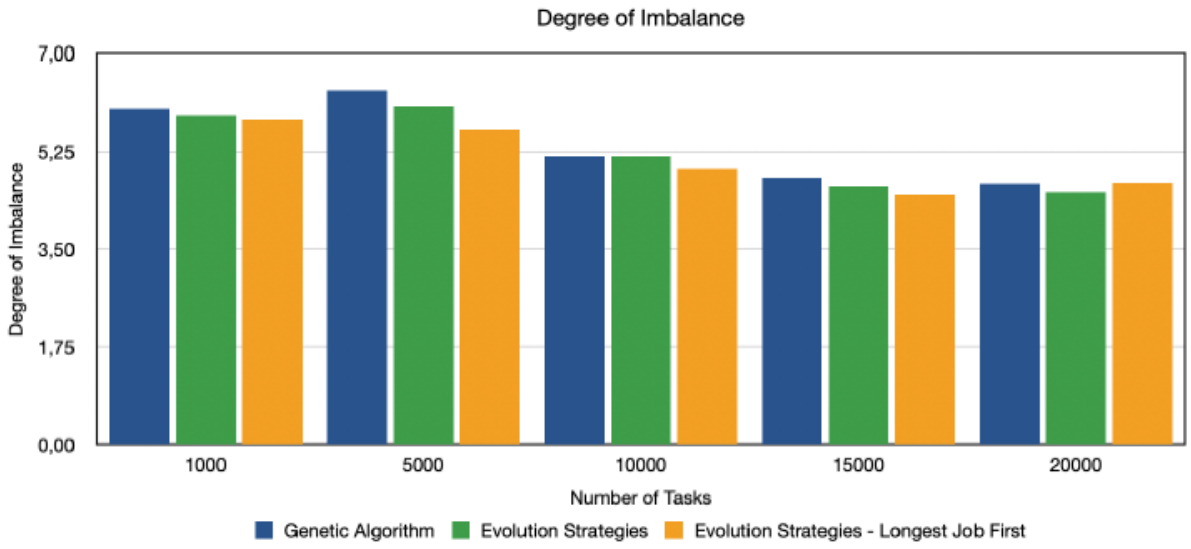

Рисунок 3.7 – Ступінь дисбалансу

Система успішно обробляє збільшену кількість завдань з точки зору масштабованості. На рисунку 3.8 показано, що екземпляри віртуальних машин у хмарній системі періодично масштабуються.

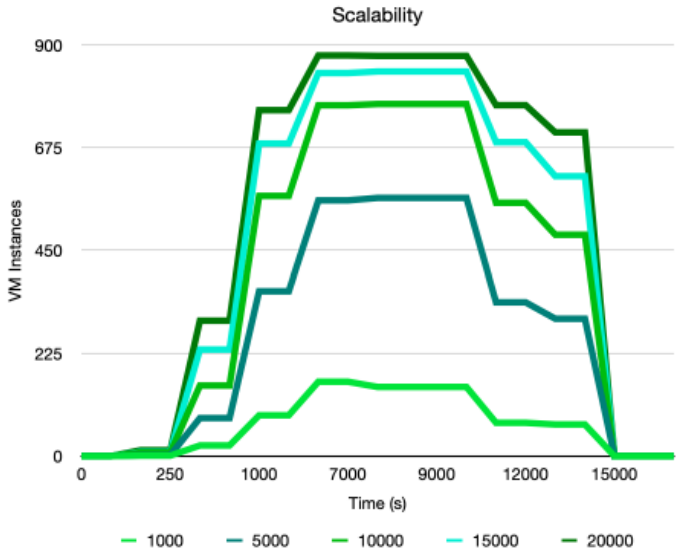

Рисунок 3.8 – Масштабування віртуальних машин у часі

Відповідно до виконання завдання ресурсна потужність надається динамічно (додається та видаляється). Масштабованість досягається в усіх 5

центрах обробки даних (по горизонталі). Ступінь дисбалансу та віртуальні машини на хостах усіх центрів обробки даних (по вертикалі). Підходи «Стратегії еволюції» та «Стратегії еволюції з найдовшою роботою спочатку» демонструють схожу ефективність масштабованості.

Ці підходи мають загальний принцип розподілу завдань, який слідує алгоритму стратегії еволюції. Така поведінка масштабованості очікується, оскільки алгоритми на основі Evolution Strategies призначають завдання віртуальним машинам, а брокер розподіляє завдання між призначеними віртуальними машинами.

Алгоритми на основі стратегій розвитку впливають на зміни в кількості віртуальних машин і типі віртуальних машин і, таким чином, впливають на масштабованість. Принцип «Спочатку найдовше завдання» впливає на балансування навантаження та визначає пріоритетність найтриваліших завдань, але не впливає на кількість і вибір віртуальних машин, які будуть використовуватися.

Центри обробки даних великої місткості отримують пропорційно більше завдань.

Таким чином, навантаження в центрах обробки даних збалансовано між 5 центрами обробки даних (DC) у всіх сценаріях тестування, як показано на рисунку 3.9. Це сприяє масштабованості системи. Аналізуючи оцінені випадки, показані на рисунках, можна зробити висновок, що екземпляри VM масштабуються протягом виміряних інтервалів на хостах і центрах обробки даних залежно від кількості поточних активних завдань.

Завдяки великій місткості центрів обробки даних і налаштуванням ресурсів, встановленим у симуляторі, були досягнуті оптимізовані значення виміряних показників. Майже в усіх сценаріях тестування підходи на основі стратегій еволюції показали кращі результати.

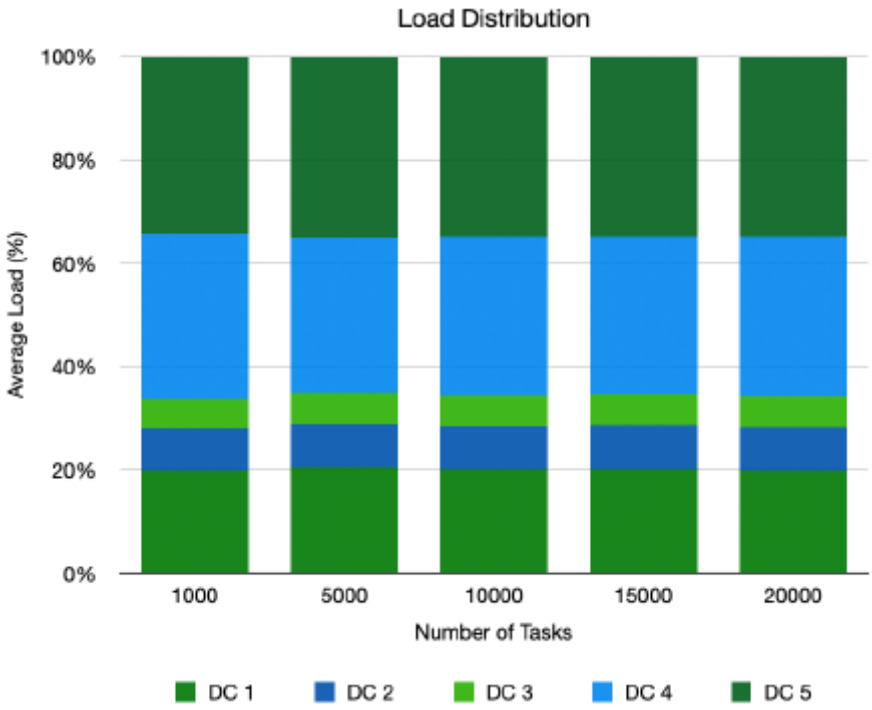

Рисунок 3.9 – Розподіл навантаження центру обробки даних

ВИСНОВКИ

Щоб досягти бажаного планування завдань з мінімальним часом виконання, а також покращити якість хмарного сервісу, запропонований метод використовує двоетапну структуру планування завдань і пов'язані з ним алгоритми. На основі історичних даних планування завдань попередньо створено достатню кількість віртуальних машин з різними атрибутами ресурсів. Це може заощадити значну кількість часу під час створення віртуальних машин і зменшити частоту відмов у плануванні завдань. Розроблено еволюційний метаевристичний алгоритм для покращення продуктивності планування завдань. Запропонований алгоритм виявляє завантаження віртуальної машини, використовуючи нову схему додавання та видалення феромонів VM. Виявляючи завантажені віртуальні машини за допомогою феромону, він покращив мутацію генетичного алгоритму та операцію кросинговеру. Запропонований алгоритм виключає віртуальні машини з рішення розміщення, якщо вони завантажені. Запропонований алгоритм здатний дати краще рішення з меншою обчислювальною потужністю через зменшений простір рішення. Час виконання скорочується, оскільки перевантажені віртуальні машини видаляються з простору рішень, оскільки віртуальні машини в просторі рішень можуть почати виконання наступних завдань раніше, ніж завантажені віртуальні машини.

Запропонована реалізація метаевристичного алгоритму *(μ, λ)*-Evolution Strategies, адаптованого для вирішення проблеми планування завдань у гетерогенних середовищах хмарних обчислень. Метод було дослідженго під час моделювання планування завдань у рамках CloudSim. Робоче навантаження моделювання було створено на основі завдань. Було проведено різні тести моделювання, щоб встановити та перевірити ефективність запропонованого підходу Стратегії розвитку. Експерименти показують, що планування завдань Evolution Strategies із реалізованою моделлю політики

брокера «Спершу найбільше завдання» є надійним. Він досягає значно кращої продуктивності, ніж метаевристики генетичного алгоритму та стратегії еволюції.

Запропонований алгоритм планування завдань може оптимально планувати завдання для віртуальних машин. Підхід, заснований на Evolution Strategies, мінімізує проміжок часу, скорочує середній час виконання та дисбаланс, збільшує використання ресурсів. Аналіз масштабованості алгоритму Evolution Strategies із політикою брокера центру обробки даних «Спершу найдовша робота» надав гарні результати. Також покращено розподіл навантаження центру обробки даних для алгоритму Evolution Strategies із політикою брокера «Спочатку найдовша робота», пропускну здатність системи і рівень масштабованості. Динамічний розподіл завдань і керування неоднорідними ресурсами центру обробки даних великої ємності покращують продуктивність системи за будь-яких сценаріїв [21].

Можна зробити висновок, що запропоноване рішення забезпечує масштабованість використання хмарних ресурсів у різних центрах обробки даних. Алгоритм Evolution Strategies займає важливе місце в штучному інтелекті та має потенціал, щоб відповісти на проблеми продуктивності стандартних методів навчання з підкріпленням.

У майбутніх дослідницьких роботах планується оптимізувати оператор допасованості алгоритму шляхом інтеграції більшої кількості характеристик робочого навантаження та характеристик ресурсів для кращих рішень. Алгоритм планування завдань, який балансує час призначення завдання, час виконання завдання, вартість і балансування навантаження, також можна розглядати в майбутніх роботах.

Майбутня робота також може включати тестування методу на реальній системі хмарних обчислень і вдосконалення запропонованого методу планування шляхом розгляду того, як зменшити споживання енергії, використання ресурсів і масштабування при збереженні якості обслуговування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. IDC Forecasts Worldwide `Whole Cloud' Spending to Reach $1.3 Trillion by 2025. Accessed: Mar. 2022. [Online], Available: https://www.idc.com/getdoc.jsp?containerId=prUS48208321

2. Croatian Scientific and Educational Cloud (HR-ZOO). Accessed: Mar. 2022. [Online]. Available: https://www.srce.unizg.hr/hr-zoo/en

3. Buyya R. A manifesto for future generation cloud computing: Research directions for the next decade. ACM Comput. Surv., vol. 51, no. 5, Sep. 2019, Art. no. 105.

4. P. Loncar, Modeling and simulation of heterogeneous resources in the cloud: (Work in progress) in Proc. IEEE 20th Int. Symp. Netw. Comput. Appl. (NCA), Nov. 2021, pp. 1–3.

5. Duboc L., Rosenblum D., Wicks T. A framework for characterization and analysis of software system scalability," in Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. (ESEC-FSE), Sep. 2007, pp. 375–384.

6. Gao J., Pattabhiraman P., Bai X., Tsai W. T. SaaS performance and scalability evaluation in clouds. in Proc. IEEE 6th Int. Symp. Service Oriented Syst. (SOSE), Dec. 2011, pp. 61–71.

7. Волк М.О., Гора М.В. Модифікований метод самовідновлення розподіленого програмного забезпечення в гетерогенних комп'ютерних системах. Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27). С. 5-17. DOI: https://doi.org/10.30837/ITSSI.2024.27.005

8. Herbst N., Kounev S., Reussner R. Elasticity in cloud computing: What it is, and what it is not. in Proc. Int. Conf. Autonomic Comput., Jun. 2017, pp. 23 – 27.

9. Qu C., Calheiros R. N., Buyya R. Auto-scaling web applications in

clouds: A taxonomy and survey," ACM Comput. Surv., vol. 51, no. 4, Jul. 2019. – pp. 1–33.

10. Jennings B., Stadler R. Resource management in clouds: Survey and research challenges. J. Netw. Syst. Manage., vol. 23. no. 3. Jul. 2015. – pp. 567–619,

11. Liu CY, Zou CM, Wu P. A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. In 2014 13th International Symposium on Distributed computing and applications to business,engineering and science, 2014; pp. 68–72.

12. Duan K, Fong S, Siu SW, Song W, Guan SSU. Adaptive incremental genetic algorithm for task scheduling in cloud environments. Symmetry. 2018;10(5):168.

13. Jang SH, Kim TY, Kim JK, Lee JS. The study of genetic algorithm-based task scheduling for cloud computing. Int J Control Autom. 2012;5(4). P.157–162.

14. Milan ST, Rajabion L, Darwesh A, Hosseinzadeh M, Navimipour NJ. Priority-based task scheduling method over cloudlet using a swarm intelligence algorithm. Clust Comput. 2020. 23. P. 63–71.

15. Ajmal MS, Iqbal Z, Khan FZ, Ahmad M, Ahmad I, Gupta BB. Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers. Comput Electr Eng. 2021;95: 107419.

16. Walia NK, Kaur N. Performance Analysis of the Task Scheduling Algorithms in the Cloud Computing Environments. In 2nd International Conference on intelligent engineering and management, 2021; pp. 108–113.

17. Osypanka P, Nawrocki P. Resource usage cost optimization in cloud computing using machine learning. IEEE Trans Cloud Comput. 2020;10(3):2079–89.

18. He Z, Dong J, Li Z, Guo W. Research on Task Scheduling Strategy Optimization Based on ACO in Cloud Computing Environment. In IEEE 5th Information Technology and Mechatronics Engineering Conference, 2020; pp.

1615–1619.

19. Mamchych O., Volk M. A unified model and method for forecasting energy consumption in distributed computing systems based on stationary and mobile devices. Radioelectronic and Computer Systems, [S.l.], v. 2024, n. 2, p. 120-135. DOI: https://doi.org/10.32620/ reks.2024.2.10

20. Volk M., Ivanisenko I. Simulation methods for load balancing in distributed computing. Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2017), Novi Sad, Serbia, September 27 –October 2, 2017, - p. 690-695

21. Волк М.О., Бугрій А.М., Бітюкова Є.В. Галушка О.І., Брестовицький Р.М., Соробей Б.В., Розподілене моделювання гетерогенних систем інтернету речей. Вісник Херсонського національного технічного університету. Том 2 No 1(92). 2025. - С. 45-50. DOI: https://doi.org/10.35546/kntu2078-4481.2025.1.2.6