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 пріоритетного планування в системах хмарних обчислень

(тема)

Виконав:

студент II курсу, групи КСМзм-22-1

Сєрих О.О.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма

Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Саранча С.М.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 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-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Сєрих Олександру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи пріоритетного планування в системах хмарних обчислень

затверджена наказом по університету від " 03 " листопада 2023 р. № 244 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 15 січня 2024 р.

3. Вхідні дані до роботи

Моделі хмарних обчислень.

Постачальники хмарних послуг (AWS, GCP, Azure).

Методи оцінки продуктивності хмарних платформ.

4. Перелік питань, що потрібно опрацювати у роботі

Аналіз предметної області.

Моделі та методи пріоритетного планування в хмарних системах.

Експериментальна частина.

Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Презентація 12 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	07.11.23 – 15.11.23	
2	Розробка моделей	16.11.23 – 30.11.23	
3	Реалізація алгоритмів	01.12.23 – 08.12.23	
4	Розробка структури програмних засобів	09.12.23 – 13.12.23	
5	Розробка програмних модулів	14.12.23 – 25.12.23	
6	Оформлення матеріалів кваліфікаційної роботи	26.12.23 – 02.01.24	
7	Подання кваліфікаційної роботи керівникові та	03.01.24 – 10.01.24	
	попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	11.01.24 – 15.01.24	
	вання		

Дата видачі завдання 06 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Саранча С.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 55 с., 15 рис., 3 табл., 1 дод., 28 джерел.

ПРІОРИТЕТНЕ ПЛАНУВАННЯ, ХМАРНІ ОБЧИСЛЕННЯ, МОДЕЛІ РОЗПОДІЛЕННЯ РЕСУРСІВ, ЕНЕРГОСПОЖИВАННЯ, ВІРТУАЛЬНІ МАШИНИ.

Ефективне планування робочих процесів у хмарних обчисленнях все ще є складною проблемою, оскільки робочі процеси, що надходять в хмарну систему, мають різну потужність обробки завдань і залежать від різних різнорідних ресурсів. Неефективне планування робочих процесів для віртуальних ресурсів у хмарному середовищі призводить до порушень угод про рівень обслуговування та високого споживання енергії, що впливає на якість обслуговування хмарного провайдера. Багато існуючих авторів розробили алгоритми планування робочого циклу, що стосуються операційних витрат і створення, але все ж є напрямки для вдосконалення процесу планування в хмарній парадигмі, оскільки це недетермінована поліноміально складна проблема. В цьому дослідженні був розроблений багатоцільовий алгоритм планування робочого процесу з пріоритетом завдань на основі алгоритму пошуку зозулі. Моделювання було проведено на workflowsim. Для оцінки ефективності запропонованого підходу запропонований алгоритм порівняно з існуючими: Max-Min, FIFO, мінімальний час завершення, Min–Min, безпека розподілу ресурсів з ефективним плануванням завдань у гібридних системах хмарних обчислень на базі машинного навчання та Round Robin. Наш запропонований підхід перевершує ефективність завдяки мінімізації споживання енергії на 15% і зменшенню випадків порушення угоди про рівень обслуговування на 22%.

ABSTRACT

Master’s thesis: 55 pages, 15 figures, 3 tables, 1 appendice, 28 sources.

PRIORITY SCHEDULING, CLOUD COMPUTING, RESOURCE DISTRIBUTION MODELS, ENERGY CONSUMPTION, VIRTUAL MACHINES.

Effective workflow scheduling in cloud computing is still a challenging problem because the workflows entering the cloud system have different task processing power and depend on different heterogeneous resources. Inefficient scheduling of workflows for virtual resources in the cloud environment leads to SLA violations and high energy consumption, which affects the quality of service of the cloud provider. Many existing authors have developed workflow scheduling algorithms dealing with operational costs and creation, but there are still directions to improve the scheduling process in the cloud paradigm, as it is a non-deterministic polynomially complex problem. In this study, a multi-objective workflow scheduling algorithm with task priority based on the cuckoo search algorithm was developed. The simulation was carried out on workflowsim. To evaluate the effectiveness of the proposed approach, the proposed algorithm is compared to the existing ones: Max-Min, FIFO, minimum completion time, Min–Min, security of resource allocation with effective task scheduling in hybrid cloud computing systems based on machine learning and Round Robin. Our proposed approach outperforms by minimizing energy consumption by 15% and reducing SLA violations by 22%.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІТ – інформаційні технології

США – Сполучені Штати Америки

MOPS – мільйон операцій за секунду

ПЗ – програмне забезпечення

ПК – персональний комп'ютер

ЦП – центральний процесор

API – Application Programming Interface

AWS – Amazon Web Services

CRM – Customer Relationship Management

EC2 – Elastic Compute Cloud

GCP – Google Cloud Platform

HPC – High Performance Computing

IaaS – Infrastructure as a Service

IoT – Internet of Things

HTTP – HyperText Transfer Protocol

MIPS – Million Instructions Per Second

MPI – Message Passing Interface

MPICH – Message Passing Interface CHameleon

NPB – NAS parallel benchmark

PaaS – Platform as a service

SaaS – Software as a service

SOAP – Simple Object Access Protocol

SSD – Solid-State drive

WDS – бездротова розподільча система (англ., Wireless Distribution System)

USD – United States dollar

ВСТУП

Хмарні обчислення – це нова сфера, яка змінила вимоги до обчислювальних систем від локальних серверів до хмарних систем, використовуючи величезну кількість різнорідних розподілених ресурсів. Клієнту потрібен високошвидкісний доступ до Інтернету, щоб скористатися всіма можливостями таких систем. Хмарні системи класифікуються на публічні хмари, приватні хмари, гібридні хмари та хмари спільноти відповідно до вимог користувачів. Приватна хмара розгортається для надання або зберігання високозахищених даних, тоді як публічна хмара відкрита для громадськості, щоб отримати послуги на основі оплати за використання. Гібридна хмарна система є частково приватною та частково загальнодоступною, що означає, що захищена інформація зберігатиметься приватно, а решта – публічно. Хмарна система спільноти призначена для групи людей, які працюють в одній спільноті.

Залежно від послуг, доступних у хмарі [1], вони класифікуються: платформа як послуга (PaaS), програмне забезпечення як послуга (SaaS) і інфраструктура як послуга (IaaS). IaaS надає клієнтам різні ресурси, такі як сховище, мережі та потужності обробки. PaaS надає різні платформи, такі як Windows, UNIX, Linux тощо, а SaaS надає корпоративне програмне забезпечення клієнтам. Оскільки в хмарній системі потрібно виконати багато завдань, і вони здебільшого залежать одне від одного, або функціонально, або на основі спільних ресурсів, які вони використовуватимуть, що називається робочим процесом, це виклик для постачальників послуг. Розклад робочого процесу — це недетермінована поліноміальна (NP) задача жорсткої оптимізації. Добре розроблений алгоритм планування робочого процесу є важливим, оскільки численні завдання можна виконувати без шкоди для порушень Service level agreement (SLA) та зменшення споживання енергії. Бюджет, енергія, безпека, обмеження термінів, і відмовостійкість є

важливими параметрами для планування робочого процесу. Виконання завдання в хмарному середовищі переважно двофазне. Ці етапи включають надання різних ресурсів і зіставлення ресурсів із завданням. Забезпечення ресурсами в основному виконується за допомогою динамічного планування та статичного планування. У статичному плануванні необхідні конфігурації добре відомі, тому їх складно реалізувати відповідно до вимог. Однак у динамічному плануванні масштабованість ресурсів використовується для вирішення невизначених завдань. Масштабованість конфігурації ресурсу є проблемою для дослідників. Розроблено різні алгоритми для управління масштабованістю, розглядаючи важливі параметри. Параметри дослідження класифікуються за трьома категоріями: однооб'єктивні, біоб'єктивні та багатооб'єктивні залежно від потреб користувача [2].

Ефективне планування робочих процесів у хмарних обчисленнях все ще є складною проблемою, оскільки робочі процеси, що надходять в хмарну систему, мають різну потужність обробки завдань і залежать від різних різнорідних ресурсів. Багато існуючих авторів розробили алгоритми планування робочого циклу, що стосуються операційних витрат і створення, але все ж є напрямки для вдосконалення процесу планування в хмарній парадигмі, оскільки це недетермінована поліноміально складна проблема. В цьому дослідженні був розроблений багатоцільовий алгоритм планування робочого процесу з пріоритетом завдань на основі алгоритму пошуку зозулі. Моделювання було проведено на workflowsim. Для оцінки ефективності запропонованого підходу запропонований алгоритм порівняно з існуючими: Max-Min, FIFO, мінімальний час завершення, Min–Min, безпека розподілу ресурсів з ефективним плануванням завдань у гібридних системах хмарних обчислень на базі машинного навчання та Round Robin. Наш запропонований підхід перевершує ефективність завдяки мінімізації споживання енергії на 15% і зменшенню випадків порушення угоди про рівень обслуговування на 22%.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості використання хмарних систем

Розподілені системи хмарних обчислень є більш масштабованими, ніж централізовані хмарні системи, але повідомлення, що передається між фізичними машинами (PM), можуть призвести до проблем мережі та пропускної здатності в хмарній системі. Централізована динамічна консолідація є більш енергоефективною в хмарній системі. Динамічну консолідацію класифікують на два типи. Непродуктивна, у якій алгоритм розглядає загальну суму поточного використання ресурсів, але не прогнозує використання ресурсів у майбутньому. З іншого боку, прогнозна динамічна консолідація передбачає майбутні робочі навантаження на основі методів прогнозування.

Розміщення віртуальної машини (VM) на основі робочого процесу є важливим завданням у хмарних обчисленнях, оскільки воно оптимізує робочі навантаження та зменшує витрати та порушення SLA. Відповідно до дослідницьких статей, опублікованих дослідниками, проблеми з розміщенням віртуальної машини можна класифікувати як «свіже розміщення», «початкове розміщення» або «статичне розміщення». У цьому типі розміщення запити постійно надходитимуть на розгортання віртуальних машин на порожніх хостах, а також запити на видалення віртуальних машин. У такому типі ситуації дуже важливо відповісти системі запитів за короткий проміжок часу.

У другому випадку, консолідованому розміщенні або динамічній консолідації, деякі завдання на основі робочих навантажень будуть завершені, а деякі нові робочі навантаження будуть виникати. Це постійний процес. Тому оптимізація необхідна для вирішення проблем споживання енергії та порушення SLA [3]. У процесі планування система відобразить

виконання взаємозалежних завдань у розподілених системах. Планувальник виділить відповідні та необхідні ресурси системі на основі робочого процесу, щоб система мінімізувала час виконання. Як конкретні рішення проблеми, так і її рішення можуть бути більш практичними через накладні витрати на створення розкладу, які є дуже високими.

Немає відомих алгоритмів для генерації найбільш оптимальних рішень у поліноміальному часовому плануванні, тому розподілені ресурси вважаються NP-складними проблемами [4]. Різні типи метаевристичних алгоритмів використовуються в хмарних системах, такі як генетичний алгоритм (GA), оптимізація рою частинок (PSO) і алгоритми штучної бджолиної колонії використовуються для вирішення планування завдань робочого процесу в хмарних середовищах [5] . Більшість дослідників зосередилися на таких параметрах, як робочий діапазон, надійність, обчислювальна вартість і вартість обробки. Вони ще мають, однак зосередьтеся на інших метриках, таких як використання пам’яті, вартість електроенергії та використання мережі [6]. Багато з існуючих авторів запропонували різні алгоритми планування та звернулися до різних параметрів, але планування робочого процесу в хмарних обчисленнях є складною NP проблемою і все ж існує шанс покращити процес планування шляхом точного налаштування параметрів, які впливають на процес планування. Тому в цій кваліфікаційній роботі ми використали вороній алгоритм пошуку для вирішення процесу планування, враховуючи пріоритети завдань і їхні залежності, щоб точно відобразити ці робочі процеси на відповідних віртуальних ресурсах.

1.2 Аналіз літератури в предметній області

В роботі досліджуються два енергоефективних метода консолідації. Це алгоритми максимальної потужності і мінімальної енергії, які найкраще підходять для збільшення ефективності. Вони покращують

існуючий алгоритм, додаючи метод розширеної свідомої консолідації завдань і техніку максимального використання (MaxUtil), щоб зменшити споживання енергії та порушення SLA. Результати їхніх експериментів показують, що запропонований алгоритм кращий з точки зору енергозбереження, порушення SLA та міграції віртуальних машин. Алгоритм вибирає найкращий сервер на основі енергоспоживання та потужності центрального процесора, наявного на сервері.

Існуючі розробки використовують верхнє порогове значення, зберігаючи вільні ресурси для майбутнього використання, керуючись вимогами, що постійно змінюються, і зменшуючи порушення SLA [7]. Дослідники запропонували онлайн-алгоритм планування робочого процесу для керування науковим робочим процесом багатохмарного середовища. Запропонований алгоритм заснований на адаптивному розподілі та консолідації ресурсів під назвою OWS-A2C. Науковий робочий процес виконується в локальній перспективі, а потім, відповідно до вимог кількох програм, екземпляри розподіляються відповідно до процесу. Вони порівнюють результати запропонованого алгоритму з результатами GAINM, PHGS і PHA2CI і показують, що середнє використання ресурсу запропонованим алгоритмом менше, ніж інші, і споживає менше енергії. Дослідники зосередилися на двох основних параметрах: низькій вартості та тривалості планування робочого процесу.

Існує алгоритм оптимізації під назвою оптимізація рою частинок імунної мутації, який використовує рій імунних частинок для покращення швидкості та якості оптимізації [8]. Запропонований метод заснований на кодованому режимі, обчисленні афінності, концентрації антитіл, функції стимулювання, клонуванні антитіл, схрещуванні та мутації. Відповідно до експериментальних результатів, показаних дослідниками, і результату запропонованого алгоритму, очевидно, що алгоритм кращий з точки зору тривалості створення та вартості обчислень. Вони порівняли свій алгоритм із хмарною інфраструктурою

PSO та Iass для завдань із обмеженням термінів для хмарного провайдера, щоб показати, що запропонований алгоритм є більш ефективним з точки зору економії коштів та скорочення терміну виготовлення.

Було запропоновано алгоритм планування на основі робочого процесу з обмеженим бюджетом, названий алгоритмом планування, керованим завданнями (TDSA) [9], щоб оптимізувати час виконання в плануванні робочого процесу. Вони використали два нових механізми: динамічний механізм розподілу суббюджету для завдання та планування завдань на основі дублювання завдань у механізмі планування.

Вони використовували метод розподілу підбюджетних коштів, щоб повернути невикористаний бюджет для планування. У механізмі планування завдань на основі запитів вони розбивають неактивні слоти в ресурсах для вибіркового дублювання завдань. Результати експерименту показують, що він оптимізує запас до 17,5% і використання ресурсів до 31,6%. Дослідники порівняли свій алгоритм з жадібним алгоритмом наданням ресурсів і модифікованим неоднорідним часом найранішого завершення (GRP-HEFT).

Алгоритм, запропонований дослідниками, засновано на зниженні споживання енергії в хмарі за допомогою перевідображення критичних завдань (RMREC), яке в основному вважається часом виконання або вартістю в умовах бюджетного обмеження. Вони розклали алгоритм на дві різні фази: зменшення споживання енергії та переналаштування критичних завдань. Щоб досягти поставленого завдання, попереднє зіставлення між завданнями та віртуальними машинами базується на найменшому споживанні енергії. На другому етапі процесу критичні завдання перенаправляються на віртуальні машини з меншими витратами на частку та споживанням енергії. Дослідники порівняли запропонований ними алгоритм з алгоритмом мінімізації споживання енергії з бюджетним обмеженням і мінімізації споживання енергії з балансуванням

бюджетного навантаження. Результат показує, що запропонований алгоритм перевершує мінімізацію витрат і енергії [11].

Є ряд праць, де дослідники зосередилися на ефективному плануванні на основі класифікації завдань і порогу. Вони розділили алгоритми на етапи, у яких перший етап полягає в обробці завдань робочого процесу, щоб уникнути вузьких місць залежностей і тривалого часу виконання. У другому випадку вони використовували PSO, щоб вибрати найкращий графік робочого процесу. Дослідники порівняли запропонований ними алгоритм із GA, PSO, енергоефективним алгоритмом балансування навантаження з плануванням робочого процесу, оптимізацією Firefly та покращеним роєм частинок і виявили, що запропонований дослідниками алгоритм набагато кращий у енергозбереженні, розширенні та балансуванні навантаження. Необхідно додатково проаналізувати подальші дослідження щодо міграції віртуальної машини та адаптивних порогових вимог, щоб отримати більше ефективності в споживанні енергії [12].

У наступної статті дослідники використовували підхід кластеризації, щоб мінімізувати порушення SLA. Вони зосередилися на прогнозуванні порушень SLA та порушенні передбачення робочого навантаження, використовуючи підхід класифікації Naive Bayer. Вони порівнюють результати запропонованого алгоритму з результатами активних віртуальних машин без прогнозування та з авторегресійним інтегрованим ковзним середнім підходом для продуктивності віртуальної машини. З експериментальних результатів ясно, що порушення SLA у запропонованому алгоритмі менше, ніж в інших методах, і він працює краще, ніж інші сучасні методи [13].

Пропонується нове забезпечення ресурсами для різних завдань і планування робочого процесу. Дослідники назвали запропоновані алгоритми новим забезпеченням ресурсами та плануванням робочого процесу, GRP-HEFT. Вони намагалися мінімізувати кількість часу в хмарі

за певного бюджету. Вони порівнюють алгоритм з іншими найсучаснішими робочими процесами, такими як мультиоб'єктивне планування колоній мурашок, PSO та GA. Коли вони порівняли тривалість створення та часову складність, вони виявили, що запропонований алгоритм перевершує інші більш ніж на 13%. Часова складність алгоритму становить O (mN), а в гіршому випадку – O (mN2).

Ще одна перевага запропонованого алгоритму полягає в тому, що він є детермінованим, що означає, що кожен запуск з однаковими вхідними даними отримує той самий makespan.

Подальша дослідницька робота для використання екземплярів Spot була проведена в роботі [14]. Дослідники зосередилися на алгоритмі планування на основі робочого процесу з обмеженим бюджетом, щоб зменшити тривалість роботи. Оскільки модель виставлення рахунків у більшості систем хмарних обчислень базується на годинах, і це повсякденна задача для планування робочого процесу, це призводить до збільшення терміну виконання, а також до нездійсненних рішень.

Пропонований алгоритм базується на погодинному циклі розрахунків. Крім того, оскільки обмеження даних застосовуються до завдань робочого процесу в хмарі, існує ймовірність неактивних слотів у ресурсах хмари. Кілька робіт можна використати для цих неактивних слотів, щоб повторювані завдання були виконані на вже використаних ресурсах, щоб скоротити час виконання. Це мінімізує тривалість робочого процесу, одночасно забезпечуючи бюджетні обмеження. Запропонована модель TDSA базується на двох важливих факторах: алгоритмі динамічного суб-бюджетування, і перший з них здебільшого відповідає за відновлення невикористаного бюджету та його перерозподіл, а другий базується на механізмі планування дублювання завдань, щоб використовувати незадіяні системні слоти на ресурсах для вибіркового дублювання завдання.

Результат алгоритму показує, що алгоритм TDSA зменшує пробіг на 17,4% і використання ресурсів на 31,6% [15]. Оскільки управління робочим процесом і планування в хмарних обчисленнях є дуже актуальними, дослідники зосередилися на розумному управлінні хмарою для підвищення продуктивності.

Вони порівняли запропонований алгоритм з алгоритмами GA. Вони запропонували алгоритми модифікованої оптимізації колонії мурашок (MOACO) і алгоритму погоні за гепардом (CCA), щоб збільшити використання ресурсів ЦП і мінімізувати час відповіді. Дослідники використали динамічну модель прогнозування ресурсів для підвищення продуктивності [16]. Вони розробили нові алгоритми, щоб мінімізувати проміжок часу, використовуючи два механізми: динамічний підбюджетний розподіл, у якому планується невикористаний бюджет, і планування на основі дублювання завдань, у якому незадіяні слоти використовуються для дублювання завдання. Пропонований алгоритм TDSA порівнюється з різні базові алгоритми та зменшує робочий діапазон до 17,4%. Дослідники не акцентували увагу на невизначеності робочого процесу [10].

Дослідники намагалися мінімізувати тривалість планування на основі робочого процесу, зменшивши витрати на виконання в умовах обмежень термінів і бюджету. Пропонований алгоритм заснований на GA. Вони використовували метод вирівнювання зверху вниз, щоб призначити пріоритет завданням. Вони розробили метод двовимірного кодування. Метод використовується для отримання різного потомства для збільшення різноманітності популяції в хмарі.

Зменшили витрати на планування робочого процесу вдалось в [17]. та[18], де запропонували підхід до планування завдань, який стосується makespan, коефіцієнта довжини планування (SLR) і прискорення. Цей підхід використовує різнорідні ресурси та механізми пріоритетних завдань у висхідних і низхідних оптимістичних таблицях витрат. Весь

експеримент проводився на робочих процесах. Він порівнював базові алгоритми, і запропонований підхід домінував над існуючими алгоритмами. Гібридизований підхід, тобто генетичні та змодельовані алгоритми відпалу, які використовуються для вирішення makespan для ефективного планування робочого процесу в хмарному середовищі [19].

Усі масштабні моделювання проводяться на робочих процесах workflowsim і запропонованому підході в порівнянні з існуючими підходами та покращеними параметрами за допомогою цього підходу.

Метаевристичний підхід був розроблений для вирішення різнорідних ресурсів у плануванні в хмарній парадигмі, спрямований на мінімізацію часу створення [20]. Весь шаблон планування базується на пріоритетах завдань і вставці завдань у відповідні слоти на основі дублювання завдань. Він оцінив робочі процеси в реальному часі та визначив, що HH-LiSch перевершує існуючі алгоритми за вказаним параметром.

Використовується також підхід до планування робочого процесу, який використовується при плануванні в хмарній парадигмі [21]. Дискретний PSO використовується як методологія в цій парадигмі шляхом швидкого генерування початкового рою. Оцінюється за існуючими підходами та імпровізованими параметрами, тобто SLR та ефективністю.

Ефективний алгоритм планування завдань, змодельований з використанням гібридного підходу, тобто GA, та алгоритмів імітованої нормалізації [22]. Цей гібридний підхід забезпечує баланс між розвідкою та експлуатацією. Він оцінює сучасні алгоритми та адресує параметри, тобто SLR і прискорення.

Алгоритм планування бі-цільового завдання був розроблений з використанням моделювання нормалізації [23]. Цей підхід покращує шаблони планування в порівнянні з HEFT і порівнює робочі процеси в реальному часі та імпровізовані параметри SLR, грошову вартість,

робочий діапазон і ефективність. Розроблено двоцільовий підхід до планування завдань, щоб вирішити питання щодо тривалості та надійності [24]. Цей підхід було розроблено та змодельовано за допомогою дискретного пошуку зозулею, який балансує між локальним і глобальним пошуком. Він порівнювався з існуючими підходами та розглядав згадані параметри. Порівняння методів наведено у таблиці 1.1.

Таблиця 1.1 – Показники ефективності для сучасних методів планування

Назва методології	Параметри, що використовуються
ECTC	Енергоспоживання, міграції, порушення SLA
OWC-A2C	Використання ресурсів, витрати на виконання
IMPSO	Час виконання, вартість
TDSA	Час виконання, використання ресурсів
RMREC	Споживання енергії
PSO	Makespan, енергоспоживання, балансування навантаження
Clustering approach	Прогноз порушення SLA, навантаження
GRP-HEFT	Час виконання завдання
DESA	Час виконання завдання, балансування навантаження
MOACO, CCA	Завантаження ЦП, час відгуку
TDSA	Час виконання завдання
PMHEFT	Час виконання завдання, енергоспоживання
SLAAEERM	Енергоефективність, відповідність SLA.
TDA	Час виконання завдання, прискорення
Hybrid GA	Час виконання завдання
HH-LiSch	Відповідність SLA, час виконання завдання, прискорення
HDPSO	Відповідність SLA, час виконання завдання, прискорення
TSAA	Відповідність SLA, прискорення
Bi-objective	Вартість, прискорення, час розгортання
HDCSA	Час виконання завдання, прискорення

В таблиці враховується безпека розподілу ресурсів із ефективним плануванням завдань у методології хмарних обчислень і гібридного машинного навчання (RATS-HM). Параметрами, що розглядаються, є використання ресурсів, споживання енергії та час відгуку [27]. ми проаналізували різні алгоритми планування завдань і робочих процесів, які враховують одноцільові та багатоцільові підходи, але оскільки планування в хмарних обчисленнях все ще є проблемою, оскільки це NP-складна проблема. Крім того, це пріоритетне планування робочого процесу шляхом мінімізації енергоспоживання, порушення SLA не сформульовано існуючими авторами. Тому, щоб подолати цю прогалину та вирішити ці параметри під час ефективного планування робочих процесів на віртуальних ресурсах, ми сформулювали запропоновану нами багатоцільову модель планування робочого процесу за допомогою пошукової оптимізації зозулі.

1.3 Постановка мети та завдань дослідження

Метою роботи є збільшення ефективності робочого процесу в системах хмарних обчислень шляхом розробки метода пріоритетного планування завдань за хмарними ресурсами. Нижче наведено основні задачі та обмеження роботи:

- багатоцільова модель планування робочого процесу розроблена за допомогою воронячого алгоритм пошуку;
- пріоритети завдань обчислюються на основі тривалості завдання та потужності обробки VM;
- масштабне моделювання проводиться на workflowsim шляхом генерації випадкових робочих навантажень;
- оцінити запропонований підхід в порівнянні з базовими підходами для визначення ефективності щодо відповідних показників, тобто порушення SLA та споживання енергії;

2 МОДЕЛІ ТА МЕТОДИ ПРІОРИТЕТНОГО ПЛАНУВАННЯ В ХМАРНИХ СИСТЕМАХ

2.1 Моделі планування в хмарних системах

У цьому розділі описано основні проблеми та моделі, на основі яких було розроблено подільші методи. Щоб виявити споживання енергії та порушення SLA, ми враховуємо наступні параметри.

По перше, це час виконання завдання. У зв'язку з розподіленою природою завдання, час виконання усього завдання дорівнює максимальному часу виконання одної з програм в завданні:

$$T_{task} = Max(t_i^{finish} - t_i^{start}), i = \overline{1,N} \quad (2.1)$$

де N – кількість програм в завданні $task$, t^{finish} – час завершення виконання програми, t^{start} – час початку виконання програми.

По друге, це споживання енергії на виконання завдань. Цей параметр використовується для розрахунку загального енергоспоживання серверів, доступних у центрі обробки даних.

Споживання енергії: у хмарних обчисленнях існує два типи споживання енергії. Фіксоване енергоспоживання (E_f) та динамічне споживання енергії (E_d). Фіксоване споживання енергії через простої сервера, а динамічне споживання енергії пов’язане з енергією, яка споживається під час виконання будь-якого завдання та використання інших інфраструктур у хмарі [26]:

$$E_{full} = (\rho) \cdot E_f + (1 - \rho) \cdot E_d \quad (2.2)$$

де ρ – частина простою ресурсу, яка для конкретного ресурсу визначається як

$$\rho_i = \frac{T_{task} - (t_i^{finish} - t_i^{start})}{T_{task}} \quad (2.3)$$

Динамічне споживання енергії в основному базується на системі охолодження (E_{col}), комунікаційного ресурсу ($E_{network}$), ресурсу зберігання (E_{storg}), та енергії, що витрачається на обчислення (E_{CPU}):

$$E_d = E_{CPU} + E_{storg} + E_{network} + E_{col}. \quad (2.3)$$

Фіксоване споживання енергії можна оцінити таким же чином (тоді, вочевидь, фіксоване споживання повинно бути менше за динамічне), або використовувати статистичні амортизаційні характеристики дата центру.

Споживання енергії можна додатково розділити на час простою та активний час. Цільову функцію енергоефективності з урахуванням розподілу енергії (2.2) та характеристики простою ресурсів (2.3) можна сформулювати наступним чином:

$$min(E_{full}) = min\left(\sum_{i=1}^{N}(\rho_i \cdot E_f + (1-\rho_i)\cdot E_d)\right), \quad (2.4)$$

за умови однорідності хмарних ресурсів. В іншому випадку, E_f і E_d доведеться обчислювати для кожного ресурсу окремо.

Споживання енергії буде мінімальним, коли час простою дорівнює 0. Споживання енергії в режимі простою можна мінімізувати за допомогою динамічного масштабування напруги та частоти конкретного основного процесора.

2.2 Ступінь забезпечення SLA

Мета визначення цього параметра – визначити відсоток порушень SLA шляхом порівняння очікуваного часу виконання з фактичним часом виконання. Порушення SLA є відношення загального попиту до виділених ресурсів, основної пам’яті, сховища та пропускної здатності. Порушення SLA у всіх завданнях з урахуванням процесорів, на яких виконуються програмні компоненти завдання:

$$SLA = \sum_{k=0}^{N} \frac{A_k^{CPU} - E_k^{CPU}}{A_k^{mem}}. \qquad (2.5)$$

Щоб прискорити процес планування завдань, спочатку ми обчислюємо пріоритети завдань, використовуючи довжину завдання та продуктивність віртуальних машин. Пріоритет розраховується за допомогою рівняння:

$$Pr = \frac{T_k^{size}}{VM_n^{pro}} \qquad (2.6)$$

2.3 Розроблення методу пріоритетного планування

Хмарні постачальники відіграють важливу роль у хмарних обчисленнях, оскільки їм потрібно керувати максимальними цілями з мінімальними ресурсами. З іншого боку, користувачі хмари зосереджуються на продуктивності ресурсів протягом мінімального проміжку часу [25]. Програма робочого циклу в хмарних обчисленнях складається з набору завдань із залежністю одних завдань від інших завдань, як-от батьківсько-дочірнє завдання, де дочірнє завдання не може бути виконано, доки не буде виконано батьківське завдання. Ці залежності ускладнюють виконання

завдань. Запропонований нами алгоритм базується на визначенні пріоритетів завдань на основі порушення SLA та очікуваного часу, необхідного для виконання роботи. Це можливо, лише якщо ми отримаємо віртуальні машини, які є більш ефективними на момент введення завдань у процес. Знову ж таки, необхідно знати можливе порушення SLA у відсотках, таким чином алгоритм знайде точну віртуальну машину, де має бути виконане завдання. Це зменшить кількість порушень SLA, а також зменшить робочий діапазон і загальне споживання енергії.

У запропонованому методі припустимо, що є N – кількість віртуальних машин $VM = \{VM_1, VM_2, \ldots, VM_N\}$, K – кількість завдань $T = \{T_1, T_2, \ldots, T_K\}$, з кількістю фізичних ресурсів $PMs = \{P_1, P_2, \ldots, P_I\}$. Крім цього є множина датацентрів $D = \{D_1, D_2, \ldots, D_J\}$. Завдання залежать від інших завдань і ресурсів, на яких вони будуть виконуватися. Завдання згруповані в різні рівні залежно від зв'язності. Всередині групи завдання розставляються за пріоритетністю, щоб мінімізувати порушення SLA. Завданням із максимальною довжиною та потужністю обробки призначаються вищі пріоритети в межах групи/рівня. Самостійне завдання буде виконано першим на основі пріоритетів. Щоб усунути порушення SLA та зменшити споживання енергії та тривалість створення, необхідно визначити віртуальну машину, виконання на якої займатиме менше часу та без будь-яких порушень SLA. Щоб визначити найкращу ВМ, ми використали алгоритм пошуку зозулі. Алгоритм пошуку зозулі дасть найкращі віртуальні машини після кількості випадкових ітерацій.

Сформулюємо задачу методу. Визначити задачу планування робочого процесу можна, розглядаючи робочий процес як орієнтований ациклічний граф (DAG) (рисунок 2.1), вершинами якого виступають завдання $T = \{T_1, T_2, \ldots, T_K\}$. Завдання розбиті по рівням (групам) на основі пріоритетів. Параметри завдань представлені у таблиці 2.1. К цім параметрам відносяться час виконання, об'єм оперативної та зовнішньої пам'яті, та трафік.

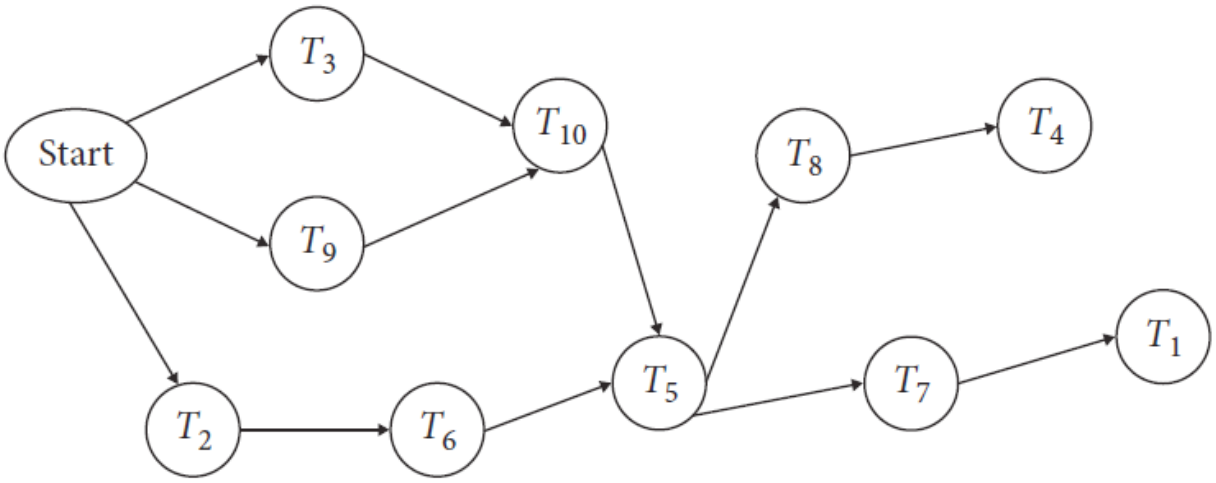

Рисунок 2.1 – Приклад DAG із 10 завданнями та п’ятьма рівнями глибини

Таблиця 2.1 – Приклад множини завдань

Завдання	Пріоритет	Час виконання(мс)	Час роботи з пам'яттю(мс)	Час роботи зберігання(мс)	Пропускна здатність(кб/с)
T_1	6	23	2	5	12
T_2	1	10	6	6	3
T_3	1	9	7	10	5
T_4	5	5	8	22	9
T_5	3	3	12	12	2
T_6	2	23	19	23	6
T_7	4	21	12	45	4
T_8	4	11	4	5	8
T_9	1	11	55	12	3
T_{10}	2	12	12	12	10

На графі DAG вказується на залежність, яка називається взаємозалежністю в робочих процесах. Ці робочі процеси виконуються на віртуальних ресурсах, і вони позначені як $VM = \{VM_1, VM_2, \ldots, VM_N\}$. У свою чергу, ці віртуальні ресурси знаходяться на фізичних ресурсах, позначених $PMs = \{P_1, P_2, \ldots, P_1\}$, а фізичні ресурси знаходяться в центрах обробки даних,

і вони позначаються як $D = \{D_1, D_2, \ldots, D_J\}$. Наведену вище суть завдання можна визначити таким чином T_k – взаємозалежні завдання, які плануються на VM_n віртуальні ресурси для визначення відсотка порушення SLA, робочого діапазону та споживання енергії.

Коли завдання надходять у систему, глобальна система перевіряє пріоритет і залежність завдання. Відповідно до пріоритету завдання та залежності робочого процесу система переставляє завдання та генерує новий DAG [18], як показано на рисунку 2.1. Коли нове завдання додається до системи, глобальний брокер повторно генерує DAG на основі доступних завдань та залежності завдань. Цей процес триватиме, доки не залишиться завдань для виконання.

Ми використали алгоритм пошуку зозулі, щоб знайти найефективніші віртуальні машини на основі доступних ресурсів для віртуальних машин. Це допоможе передати завдання у віртуальну машину, де воно зможе виконувати свої завдання без порушення SLA. Це також економить енергію та скорочує час виконання. Ми використали мультиевристичний алгоритм пошуку зозулі. Сінь-Ше та Суаш Деб розробили алгоритм, заснований на виношковому паразитизмі деяких видів зозуль і стратегії кладки. Цей алгоритм дасть нам найоптимальнішу віртуальну машину, яка потрібна для планування робочого процесу. Алгоритм ґрунтується на поведінці зозулі при розмноженні та польотах зозулі. Пропонована архітектура показана на рисунку 2.2, а послідовність виконання методу показана на рисунку 2.3.

Для довжини кроків використовується випадкове блукання, яке називається «польоти збору», і воно розподіляється відповідно до розподілу ймовірностей. Параметри задань послідовно враховуються для вибору і оцінки призначення завдання за обчислювальним ресурсов у такому порядку. На першрму кроці враховується пріоритет завдань і створюється граф. Потім визначається єфективність віртуальних машин та обирається датацентр для розмішення віртуальної машини. Потім час виконання, об'єм оперативної та зовнішньої пам'яті, та трафік використовуються для перевірки SLA.

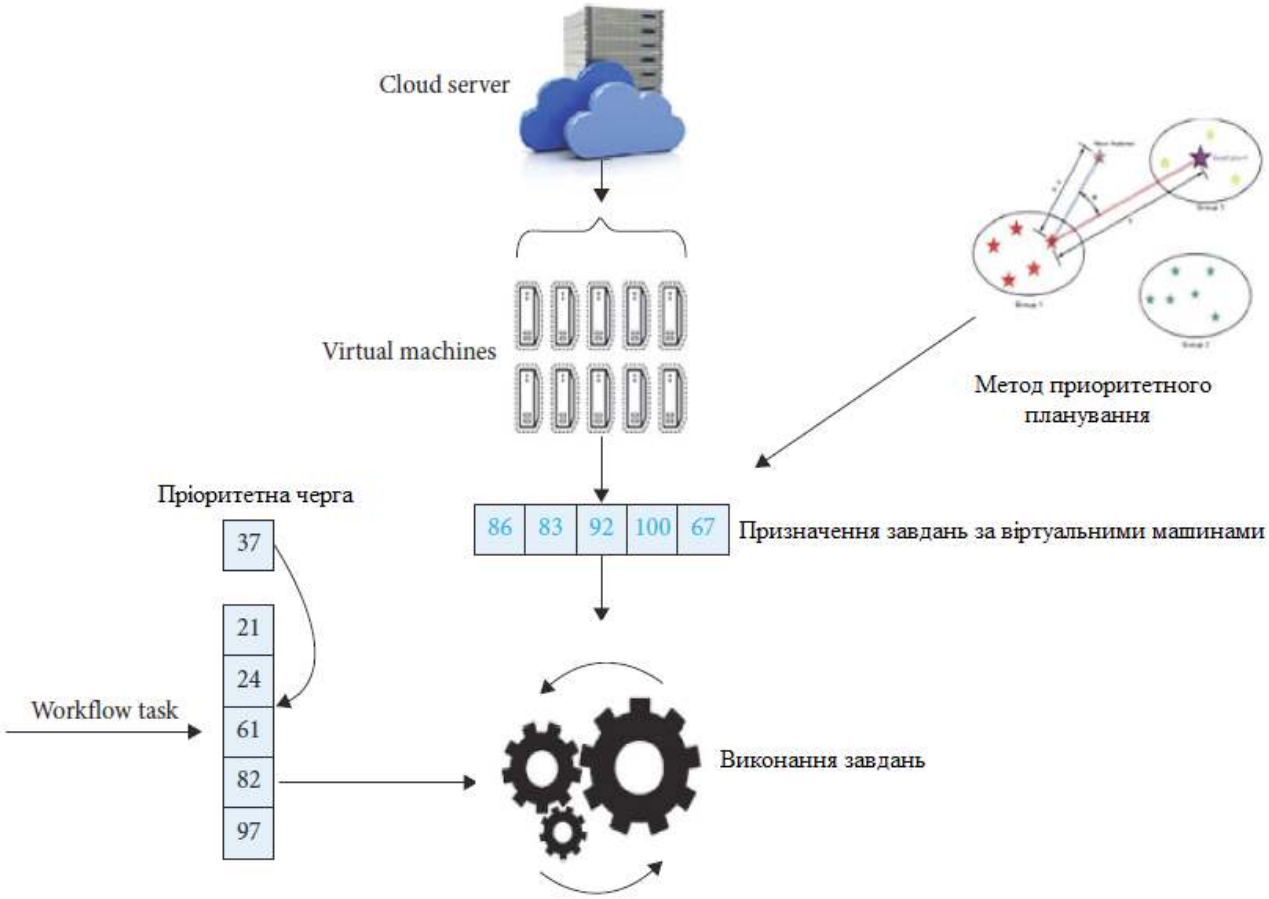

Рисунок 2.2 – Архітектура системи, що пропонується

Розрахунок часової складності запропонованого підходу полягає у наступному. Спочатку в методі усі *K* завдання надходять від різних різнорідних користувачів хмари. Для всіх завдань розраховуються пріоритети, а часова складність для розрахунку всіх пріоритетів завдань позначається *O(k)*. Після оцінки пріоритетів усі ці завдання повинні бути передані планувальнику, який отримає повну інформацію о завданні та ресурсах. Таким чином, часова складність для передачі завдань планувальнику *O(m)*. Після збору пріоритетів планувальник відповідає за створення схеми призначення завдань за віртуальними машинами, враховуючи ці пріоритети, і їх часову складність *O(n)*.

Рисунок 2.3 – Етапи методу пріоритетного планування

Тому, загальну часову складність усього описаного процесу слід представити як суму складностей цих робочих процесів: $O(k+m+n)$.

3 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

3.1 Опис експериментального фреймворка

Ефективність запропонованого методу оцінимо за допомогою фреймворку моделювання робочого процесу. Він підтримує моделювання та моделювання великомасштабних хмарних середовищ із хмарними програмами як завданнями. Завдання із їхніми залежностями попередньо визначаються у файлі montage.xml. Існує дві категорії файлів: 25 і 100 умовних робочих навантажень. На основі вивчення існуючих алгоритмів ми визначили набір значень як цільове порушення SLA. Цей набір виправлено для всіх алгоритмів, які ми використовували для виявлення порушень у нашому тестуванні. Результати часу виконання, порушення SLA та споживання енергії ефективні, оскільки значення фіксуються для всіх алгоритмів і порівнюються для всіх алгоритмів.

Наявна робота розширюється, використовуючи пріоритетне планування завдань. Завдання вводяться в систему на основі їхніх залежностей і розміщуються в DAG на різних рівнях. Завдання вводяться в процес тільки на основі їх рівня, тобто найбільш самостійні завдання вводяться в систему першими. Тоді програмні системи будуть розташовані в порядку мінімального часу завершення, щоб мінімізувати порушення SLA. Потім, на другому етапі, завдання будуть призначені віртуальній машині, яка швидко виконає роботу. У результаті найкраще підходить віртуальна машина буде призначена хмарному завданню. У результаті буде мінімальна втрата ресурсів, і ресурси будуть використані для інших процесів.

Було використано різну кількість віртуальних машин і різну кількість завдань, щоб перевірити валідність запропонованого нами методу. Ми виконали шість рівнів тестування з 5, 10 і 15 віртуальними машинами, а також з 25 і 100 завданнями із різною залежністю. Результат нашого методу

перевірено на найпопулярніших і поширених алгоритмах, таких як Max–Min, first-come-first-served (FCFS), minimal completion time (MCT), Min–Min, RATS-HM і Round Robin. Ми порівнюємо результати щодо енергоспоживання, робочого діапазону та порушень SLA. Наші спостереження чітко показують, що це зменшує порушення SLA та час виконання. У той же час це знижується енергоспоживання системи.

Кожного разу, коли завдання вводиться в систему, воно буде розміщено в DAG. Коли завдання надходить до системи від різних клієнтів, спочатку вони будуть відсортовані на основі залежності завдання. Потім система витягує завдання та перевіряє SLA кожного завдання. Після завершення процесу метод знайде найефективніші віртуальні машини для завершення роботи. Цей процес триватиме для всіх завдань.

Усе моделювання було проведено на робочих процесах і випадково згенерований робочий процес DAG із симулятора, враховуючи всі вищезгадані в цьому розділі. Детальні параметри конфігурації комп'ютера для нашого моделювання, представлені в таблиці 3.1.

Таблиця 3.1 – Конфігурація комп'ютера для моделювання

Параметр	Значення
Кількість завдань	25-100
Кількість віртуальних машин	5-15
Фізична оперативна пам'ять	16Gb
Ємкість зовнішньої пам'яті	8Tb
Пропускна здатність мережі	1,2 Mbps
Гіпервізор	Xen
Кількість датацентрів	15

3.2 Розроблення алгоритмів для реалізації методу пріоритетного планування

Ми використали програмне забезпечення симулятора обчислювального процесу, щоб протестувати наш метод у порівнянні з існуючими алгоритмами. Розробка алгоритмів на основі методу планування виконувалась на основі алгоритму пошуку зозулі, результатом роботи алгоритму є план виконання.

Перший алгоритм – це алгоритм пріоритезації завдань.

Крок 1. Ініціалізація завдань $T = \{T_1, T_2, \ldots, T_K\}$.

Крок 2. Для кожного завдання T_i з множини T_i

Крок 3. Знайти завдання з найвищим пріоритетом, який обчислити за допомогою рівняння (2.6).

Крок 4. Якщо є декілька завдань з одним пріоритетом, вибрати завдання з мінімальним часом обчислення (з найменшою складністю) або з мінімальною зв'язністю.

Крок 5. Помістіть завдання в чергу.

Крок 7. Якщо є ще якесь завдання, перейдіть до кроку 2.

Крок 8. Повернення остаточного DAG.

Наступним етапом методу застосовуємо модифікований алгоритм зозулі для пошуку множини віртуальних машин. Вхідні дані: віртуальні машини $VM = \{VM_1, VM_2, \ldots, VM_N\}$ із їхньою швидкістю, пропускною спроможністю, обсягом оперативної пам’яті. Вихідні дані: упорядковані віртуальні машини на основі ефективності.

Крок 1. Введення віртуальних машин $VM = \{VM_1, VM_2, \ldots, VM_N\}$.

Крок 2. Створення початкової популяції.

Крок 3. Виконувати до $t<max$ поколінь або задоволення критерія зупинки.

Крок 4. Отримання VM_j випадковим чином.

Крок 5: Оцінка придатності віртуальної машини для завдання з черги.

Крок 6: Якщо віртуальна машина підходить до виконання завдання, розміщення його на цій віртуальній машині.

Крок 7: Якщо для віртуальній машині ще не обрано ресурс, вибір обчислювального ресурсу $PMs = \{P_1, P_2, \ldots, P_1\}$ випадковим чином.

Крок 8: Якщо параметри ресурсу задовольняють параметрам віртуальної машини і завданням, перехід до кроку 3.

Крок 10: Знайти новий обчислювальний ресурс для розміщення віртуальної машини.

Крок 11. Якщо такого ресурсу не знайдено, видалити завдання з віртуальної машини VM_j та виключити її з подальшого розгляду.

Крок 12. Перехід до кроку 4.

Крок 13. Повернення схеми завдань, віртуальних машин та ресурсів.

На останньому етапі враховується енергоефективність розміщення завдань та віртуальних машин на обчислювальних ресурсах. Вхід: усі завдання в DAG. Результат: час виконання, порушення SLA та загальне енергоспоживання.

Крок 1. Ініціалізація завдань

Крок 2. Очікування будь-якого завдання з черги.

Крок 3. Якщо є нові завдання, вставити його у відповідній позиції в DAG і згенеруйте нову DAG на основі алгоритму 1.

Крок 4. Знайдіть віртуальну машину з мінімальним або нульовим порушенням SLA

Крок 5. Розрахуйте ефективність віртуальної машини за допомогою модифікованого алгоритму пошуку зозулі.

Крок 6. Виконайте завдання

Крок 7. Якщо будь-яке нове завдання переходить до кроку 3

Крок 8. З’ясуйте робочий діапазон, порушення SLA та енергоспоживання.

3.3 Результати моделювання

В роботі було використано програмне забезпечення симулятора робочого процесу, щоб протестувати наш алгоритм у порівнянні з існуючими алгоритмами. Ми розробили наш алгоритм на основі методу планування, заснованого на GA, з використанням алгоритму пошуку зозулі, і отриманням плану призначення завдань за ресурсами виконання. Для експериментів було використано центри обробки даних з набором хостів, із однаковими налаштуваннями, наприклад, 10 хостів і 20 віртуальних машин на кожному.

Завдання вводилися в систему випадковим чином. Запропонований алгоритм перевіряє залежність завдань і знаходить завдання на основі пріоритету. Завдання з найвищим пріоритетом буде виконано першим. Подібним чином наш алгоритм знайде найкращі віртуальні машини на основі наявних ресурсів, щоб мінімізувати потужність споживання. Далі ми порівняємо наш метод на основі часу виконання, порушення SLA та споживання енергії.

Взаємовиключні завдання розміщуються на одному рівні, щоб їх можна було виконувати паралельно. Наша програма приймає всі завдання по одному рівню. Оскільки нашою основною метою є мінімізація порушень SLA та споживання енергії, запропонований нами алгоритм перевіряє SLA кожного рівня, робочий діапазон і споживання енергії та виконує завдання на їх основі. Запропонований нами алгоритм перевіряє віртуальну машину з набором ефективності для завдання, де порушення дорівнює нулю або близько до нуля. Це мінімізує порушення SLA та зберігає найефективніші ресурси вільними для інших завдань.

3.3.1 Розрахунок енергоспоживання за різними сценаріями

Програма моделювання розраховує енергоспоживання системи для обробки завдання на основі часу процесора, використовуваної оперативної

пам’яті та пропускної здатності, що використовується в системі. Після завершення процесу програма відобразить загальне енергоспоживання системи. Розрахунок енергоспоживання віртуальних машин проводиться на основі статичного та динамічного енергоспоживання. Тестування його з іншим набором віртуальних машин і завдань. Експериментальні результати показують, що наш алгоритм перевершує добре відомі та поширені алгоритми.

Спочатку для розрахунку споживання енергії ми розглянули 5 віртуальних машин і 25 завдань і робоче навантаження, яке згенеровано випадковим чином з workflowsim. Згенероване споживання енергії для вищезазначених віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, RATS-HM і PSAMHWFSA становить 1756,2, 3243, 1876,5, 1987,8, 1875,35 і 435,76 відповідно. З рисунку 3.1 видно, що запропонований підхід перевершує існуючі алгоритми з точки зору споживання енергії для вищенаведеного сценарію.

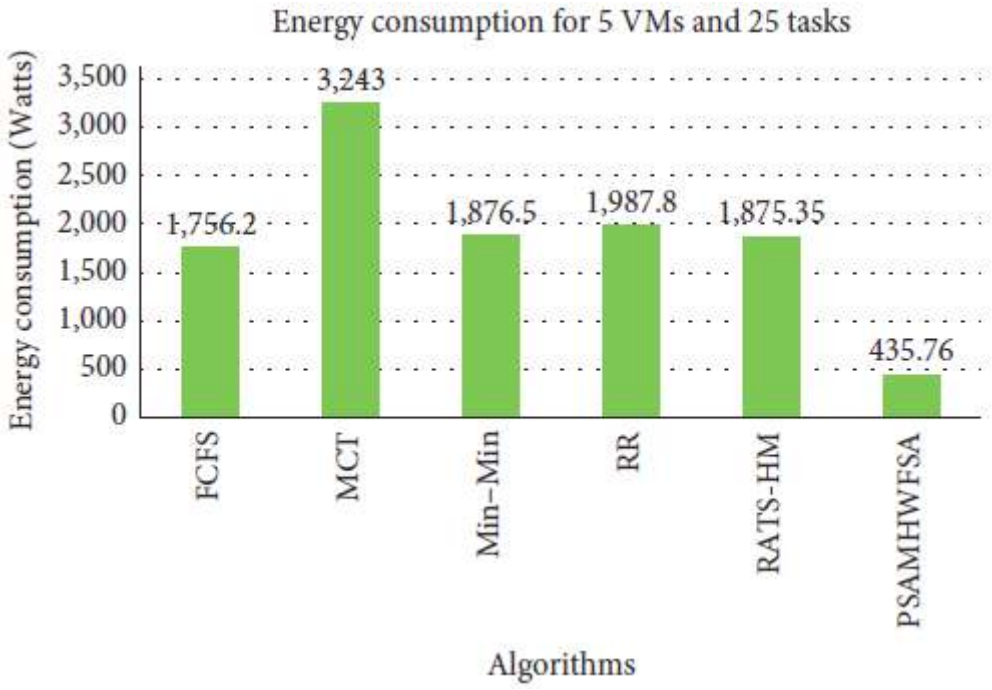

Рисунок 3.1 – Розрахунок енергоспоживання для 5 віртуальних машин і 25 завдань

В наступному експерименті розрахунку споживання енергії було розглянуто 10 віртуальних машин і 25 завдань та робоче навантаження, згенероване випадковим чином з workflowsim. Згенероване споживання енергії для вищезазначених віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, RATS-HM і PSAMHWFSA становить 2976,3, 1123,5, 1089,7, 5323, 2133,4 і 335,87 відповідно. З рисунку 3.2 можна побачити, що запропонований підхід значно перевершує існуючі методи з точки зору споживання енергії для другого сценарію.

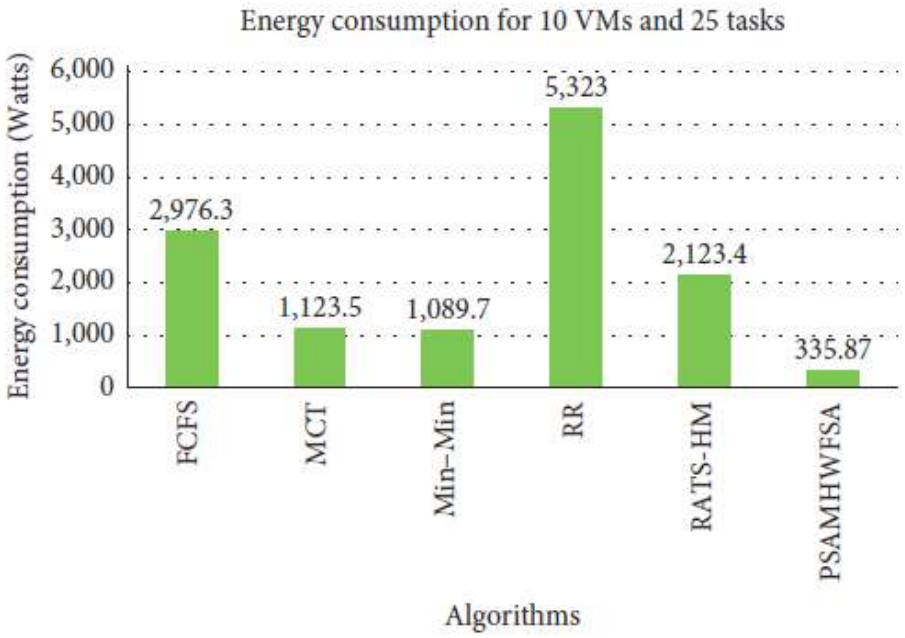

Рисунок 3.2 – Розрахунок енергоспоживання для 10 віртуальних машин і 25 завдань

Далі збільшимо кількість завдань. Виконаємо розрахунок споживання енергії для 10 віртуальних машин і 100 завдань. Розрахунки споживання енергії для такої комбінації віртуальних машин і завдань для алгоритмів FCFS, MCT, Min–Min, RR, RATS-HM і PSAMHWFSA становить 6864, 8000, 6874,3, 7012, 4786,2 і 2032,8 відповідно. На рисунку 3.3 можна побачити, що наш метод дає більш ефективне рішення по споживанню енергії для сценарію з 10 віртуальними машинами та 100 завданнями.

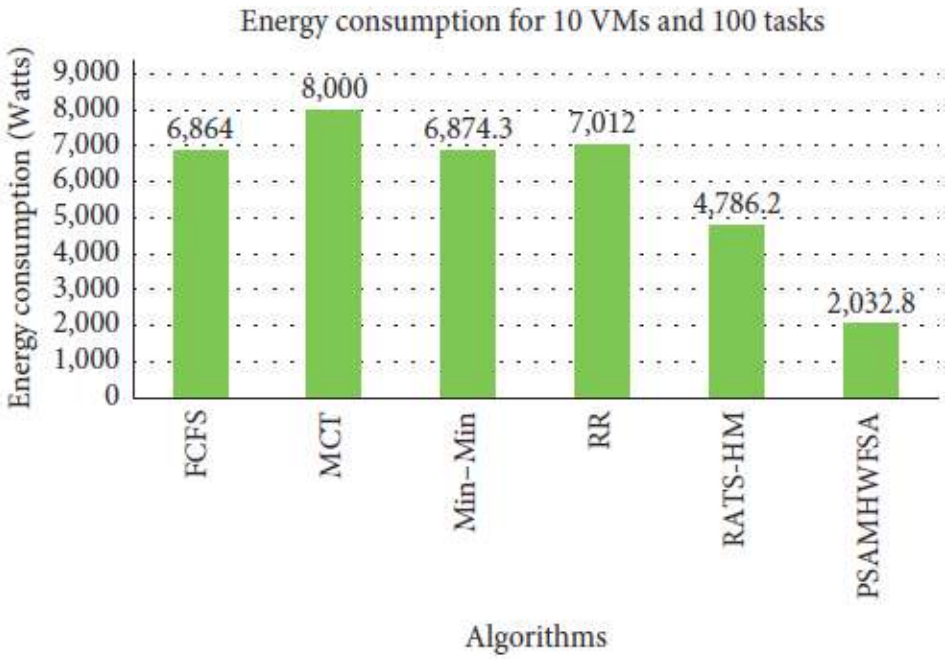

Рисунок 3.3 – Розрахунок енергоспоживання для 10 віртуальних машин і 100 завдань

В наступному екмперименті було збільшено кількість віртуальних машин. Для розрахунку споживання енергії було задано на 5 віртуальних машин більше – 15 віртуальних машин і 100 завдань, а робоче навантаження було автоматично згенеровано програмою моделювання випадковим чином з workflowsim. Згенероване споживання енергії для 15 віртуальних машин і 100 завдань було порівняно з методами FCFS, MCT, Min–Min, RR, RATS-HM і PSAMHWFSA та становло наступні значення: 5654, 7654, 5632, 6543, 4675 і 1987 відповідно для кожного методу. Діаграма, яка представлена рисунку 3.4 показує, що використання запропонованого методу для розподілу завдань та віртуальних машин за ресурсами, дає кращі варіанти стосовно споживання енергії для вищенаведеного сценарію в порівнянні з відомими методами розподілу.

Таким чином, на усіх комбінаціях входних даних запропонований метод надав кращі результати з споживання енергії, що показали експеременти, які порівняли об'єми споживання електроенергії з різними методами розподілу завдань за ресурсами.

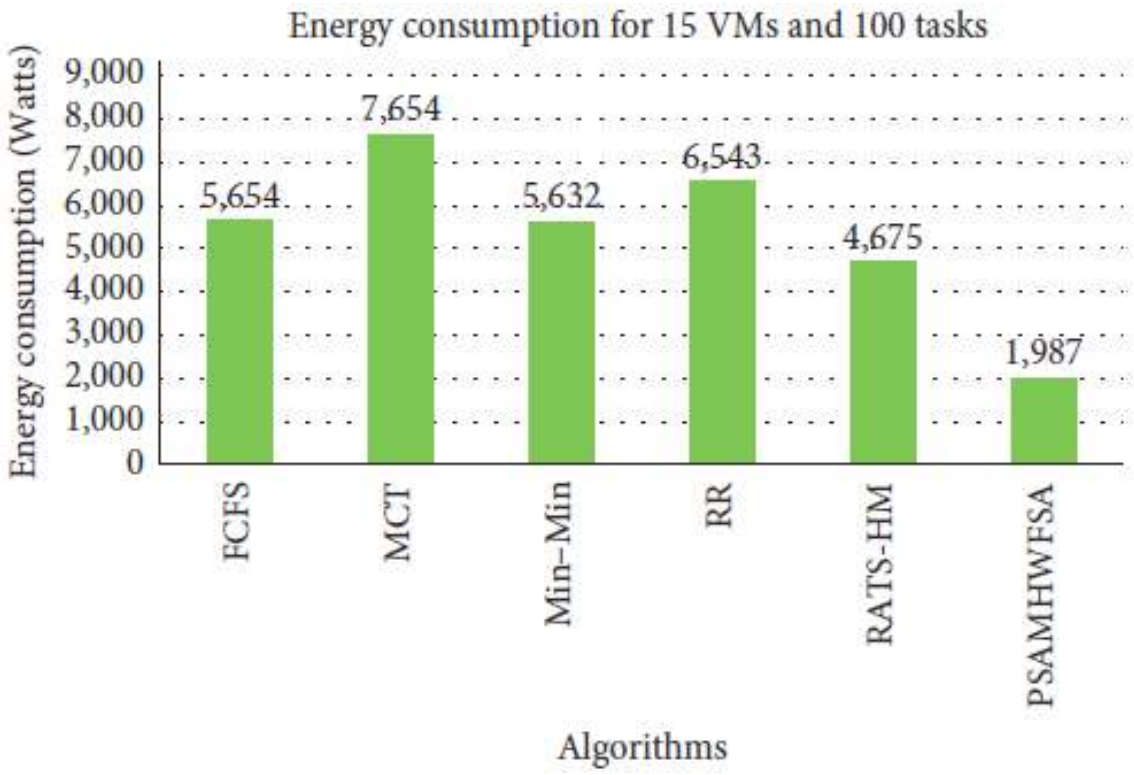

Рисунок 3.4 – Розрахунок енергоспоживання для 15 віртуальних машин і 100 завдань

3.3.2 Розрахунок часу виконання завдань

Розрахунок часу виконання з використанням різних сценаріїв. Він використовується для розрахунку максимального часу, необхідного для виконання всього завдання, введеного в систему. Це визначає максимальний час виконання усіх віртуальних машин, що використовуються в процесі. Ми обчислюємо тривалість виконання завдань на основі часу входу та часу виходу кожного завдання для кожної віртуальної машини. Потім знаходимо час, який є максимальним серед віртуальних машин. Це надає максимальний час для виконання завдань. Експериментальне тестування виконувалося з різними наборами віртуальних машин і завдань. Наші експериментальні результати показують, що запропонований нами метод перевершує добре відомі та звичайні алгоритми.

Спочатку для розрахунку часу виконання ми розглянули 5 віртуальних машин і 25 завдань і робоче навантаження, згенероване випадковим чином з workflowsim. Згенерований робочий діапазон часу виконання для вищезгаданих віртуальних машин і завдань для різних методів розподілу FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM і PSAMHWFSA становить 285,14, 356,72, 287,88, 267,32, 186,22, 183,88 і 27,32 відповідно. З рисунка 3.5 можна зробити висновок, що запропонований підхід перевершує існуючі методи з огляду на розрахунок для описаного сценарію.

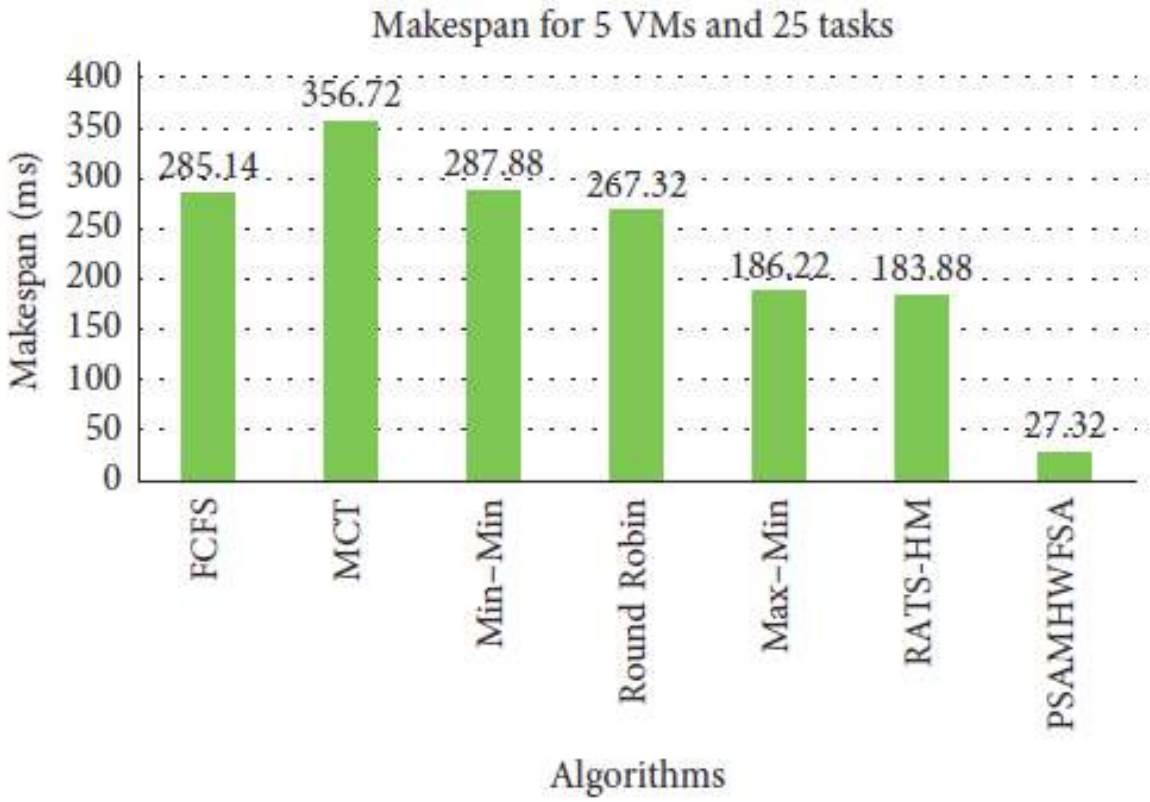

Рисунок 3.4 – Розрахунок часу виконання для 5 віртуальних машин і 25 завдань

У другому експерименті для розрахунку часу виконання боло взято 10 віртуальних машин і 25 завдань. Робоче навантаження було згенеровано випадковим чином. Згенерований час виконання для вищезгаданих віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, Max–Min, RATS-

HM і PSAMHWFSA алгоритмів: 276.87, 387.23, 298.12, 245.36, 174,34, 163,99 і 29,36 відповідно. З рисунку 3.6 видно, що запропонований підхід перевершує існуючі алгоритми з огляду на розрахунок для поточного сценарію.

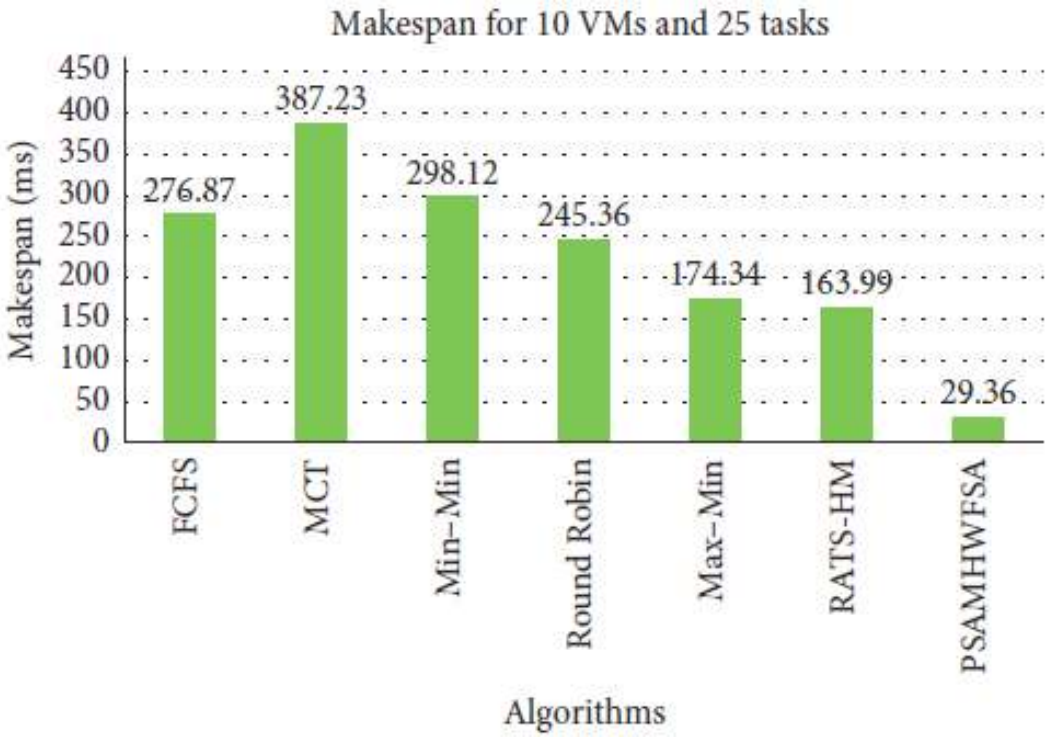

Рисунок 3.6 – Розрахунок часу виконання для 10 віртуальних машин і 25 завдань

У третьому експерименті для розрахунку часу виконання було задано 10 віртуальних машин та 100 завдань з робочим навантаженням, згенерованим автоматично системою моделювання випадковим чином з workflowsim. Обчислений час виконання було отримано для вищезгаданих кількостей віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM і PSAMHWFSA і становили 296,34, 321,42, 267,76, 283,24, 185,36, 172,46 і 31,57 відповідно. З рисунку 3.7 видно, що запропонований підхід перевершує існуючі алгоритми з огляду на розрахунок для наведеного вище сценарію.

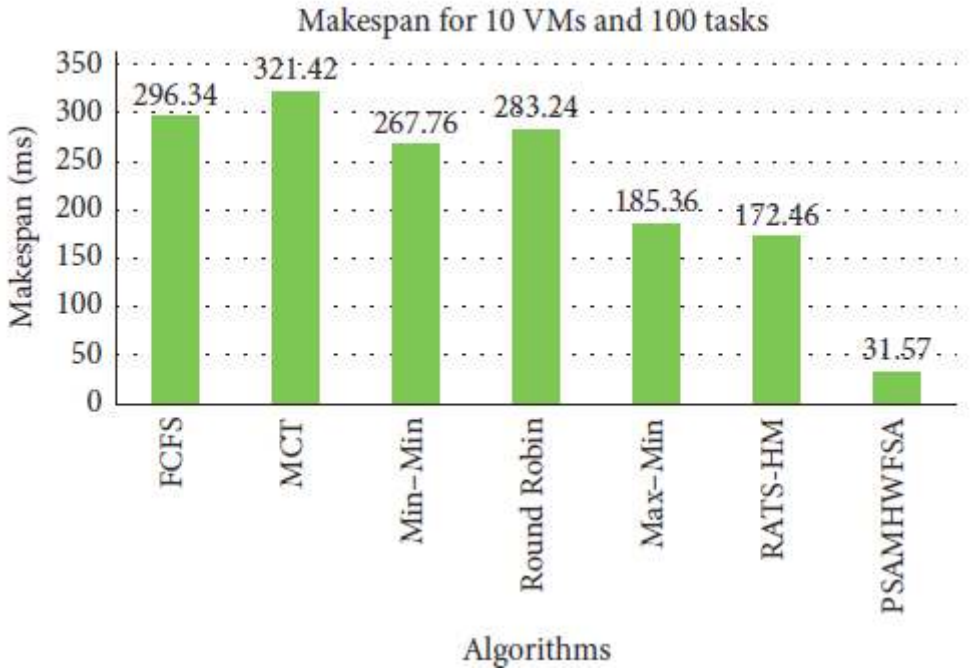

Рисунок 3.6 – Розрахунок часу виконання для 10 віртуальних машин і 100 завдань

І останній експеримент з розрахунку часу виконання проводився для комбінації 15 віртуальних машин і 100 завдань. Робоче навантаження згенероване випадковим чином з workflowsim. Значення часу виконання для вищезгаданих віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM і PSAMHWFSA становлять 297,35, 456,33, 302,35, 257,13, 185,65, 175,38 і 32,46 відповідно. З рисунка 3.7 видно, що запропонований підхід перевершує існуючі алгоритми з огляду на розрахунок для сценарію 15 віртуальних машин і 100 завдань.

3.3.3 Розрахунок порушення SLA за різними сценаріями.

Далі були проведені експерименти, які перевіряли порушення SLA кожного завдання на основі попередньо визначених SLA. Перед виконанням будь-якого завдання було перевірено порушення SLA, і VM використовувалась для виконання завдання, де порушення дорівнює нулю

або мінімально. Щоразу, коли завдання вводилося в процес, воно знаходило відповідний SLA відповідно до угоди. Потім метод перевіряв неактивні віртуальні машини, де завдання можна виконати без порушень або з мінімальними порушеннями. Було використано модифікований метод пошуку зозулі, щоб знайти найефективніші віртуальні машини.

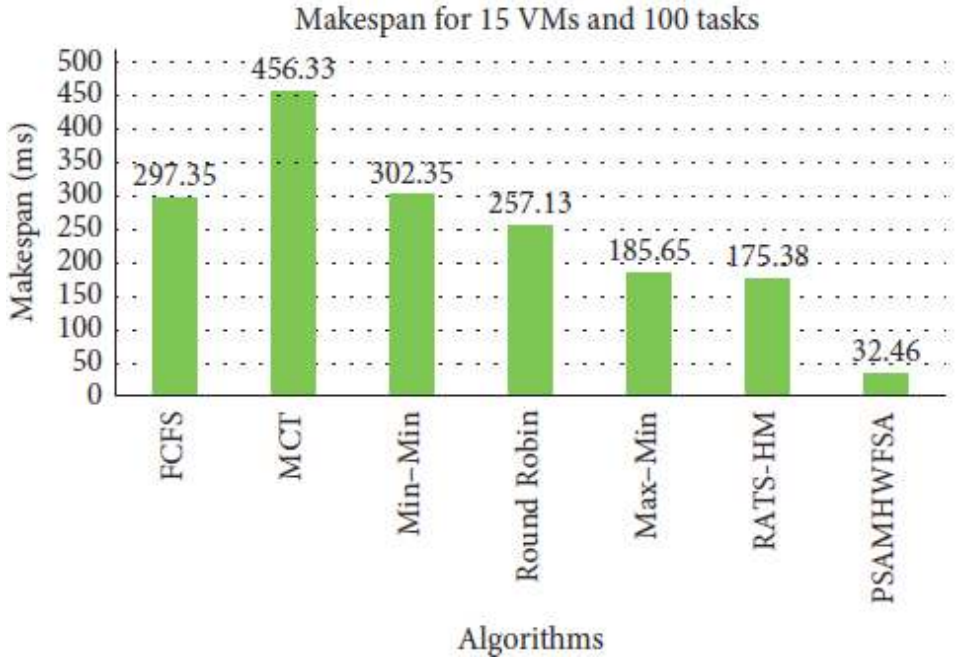

Рисунок 3.7 – Розрахунок часу виконання для 15 віртуальних машин і 100 завдань

Потім система знаходить той варіант, де є мінімальне або нульове порушення. Виходячи з результату, зрозуміло, що запропонований нами алгоритм перевершує добре відомі та звичайні алгоритми.

Спочатку для розрахунку порушення SLA було розглянуто 5 віртуальних машин і 25 завдань і робоче навантаження, згенероване випадковим чином з workflowsim. Згенеровані makespans для вищезгаданих віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM і PSAMHWFSA становлять 24, 31, 26, 32, 27, 18 і 5 відповідно. З рисунка 3.8 видно, що запропонований підхід перевершує існуючі алгоритми з огляду на порушення SLA для наведеного вище сценарію.

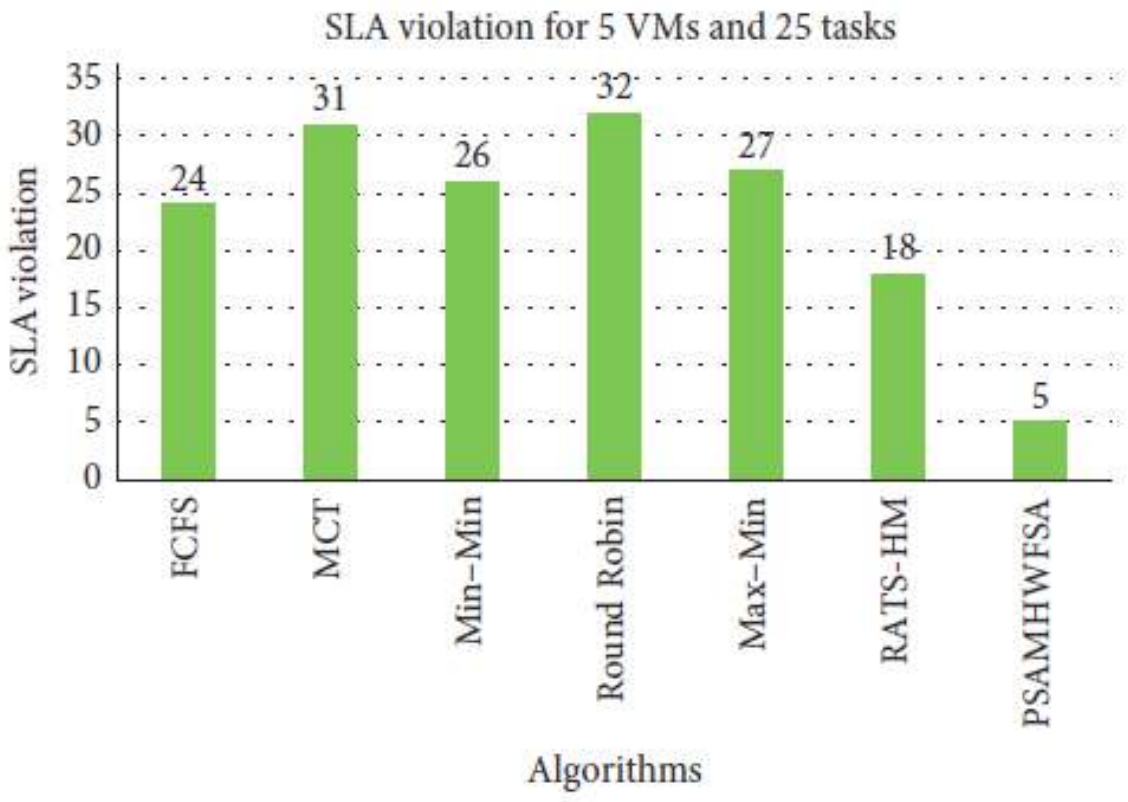

Рисунок 3.8 – Розрахунок порушення SLA для 5 віртуальних машин і 25 завдань

Для розрахунку порушення SLA В другому експерименті було розглянуто 10 віртуальних машин і 25 завдань з робочим навантаженням, яке було згенеровано випадковим чином програмою workflowsim. Обчислено порушення SLA для згадані 10 віртуальних машин і 25 завдань для методів розподілу FCFS, MCT, Min–Min, RR, Max– Min, RATS-HM і PSAMHWFSA становлять 26, 36, 22, 38, 23, 21 і 4 відповідно. На рисунку 3.9 показані результати моделювання, які показують більшу ефективність запропонованого методу в порівнянні з існуючими по значенням порушення SLA для наведеного вище сценарію.

У третьому експерименті щодо розрахунку порушення SLA задіяно 10 віртуальних машин та 100 завдань. Робоче навантаження згенеровано випадковим чином з workflowsim. Було розраховано перевищення SLA для 10 віртуальних машин і 100 завдань.

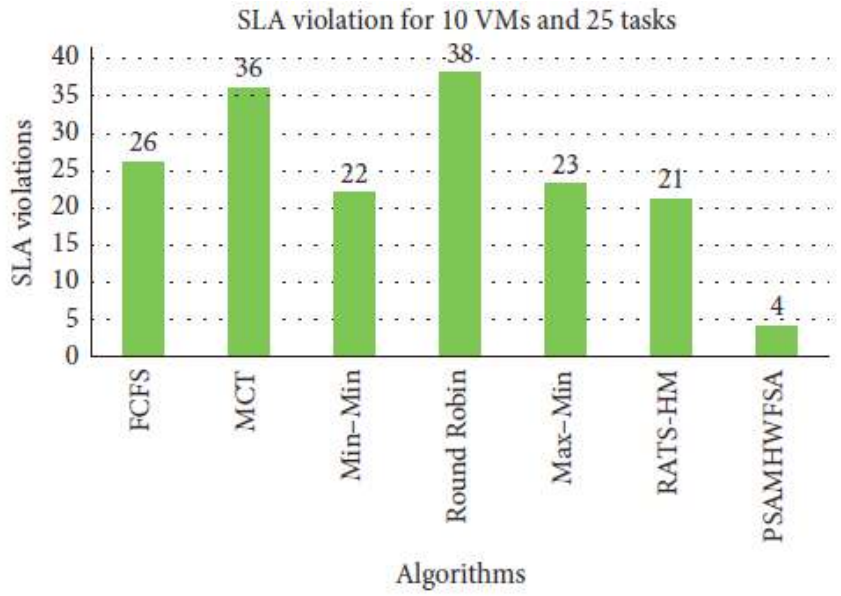

Рисунок 3.9 – Розрахунок порушення SLA для 10 віртуальних машин і 25 завдань

З рисунка 3.10 видно, що запропонований підхід перевершує існуючі алгоритми FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM по параметру порушення SLA для наведеного вище сценарію.

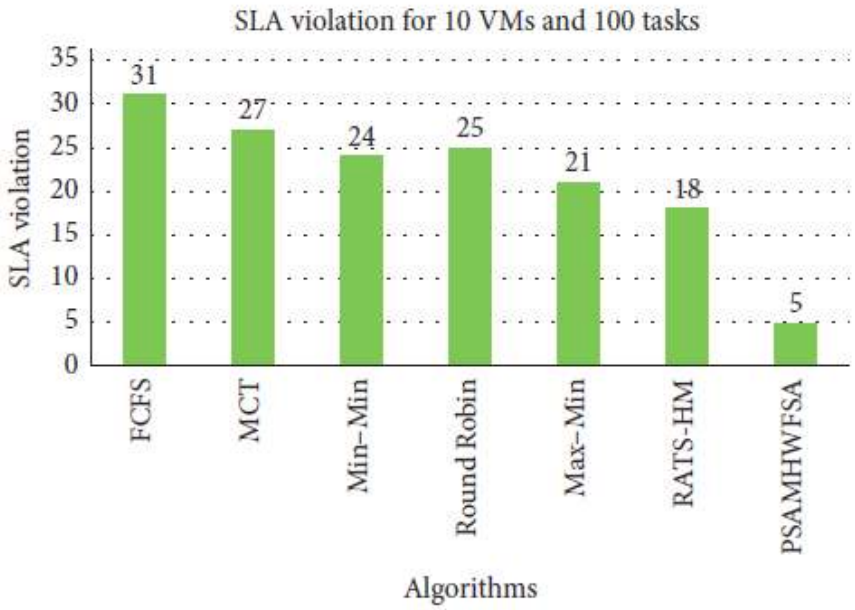

Рисунок 3.10 – Розрахунок порушення SLA для 10 віртуальних машин і 100 завдань

В останньому експерименті для розрахунку порушення SLA розглянули 15 віртуальних машин і 100 завдань. Робоче навантаження теж згенероване випадковим чином з workflowsim.

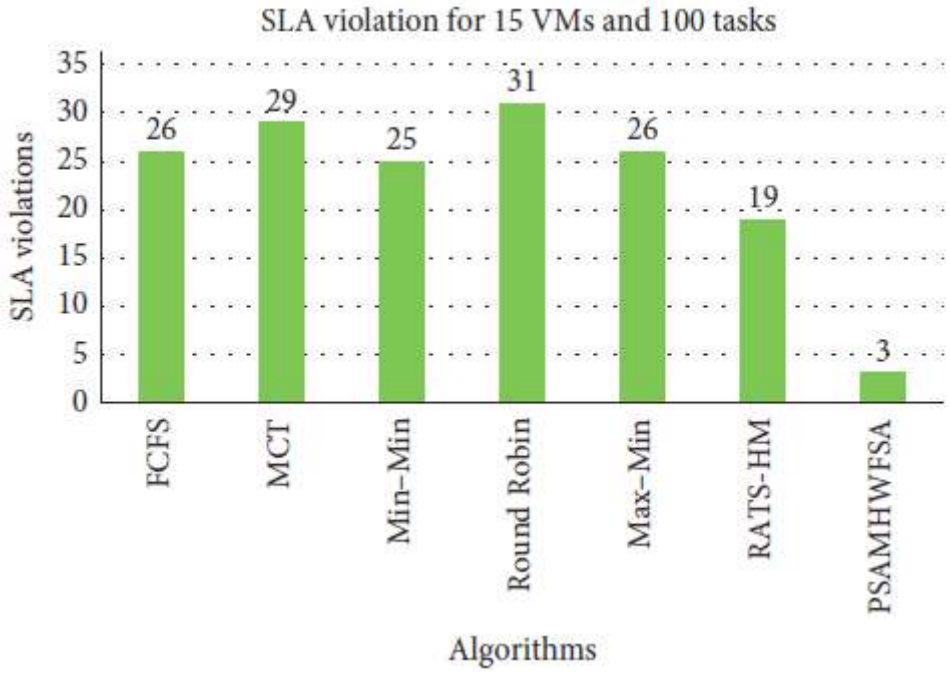

Рисунок 3.10 – Розрахунок порушення SLA для 15 віртуальних машин і 100 завдань

Результати обчислень для згаданих вище віртуальних машин і завдань для FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM і PSAMHWFSA становлять 26, 29, 25, 31, 26, 19 і 3 відповідно. Рисунок 3.11 демонструє, що розроблений метод перевершує існуючі алгоритми по критерію порушення SLA для сценарію 15 віртуальних машин і 100 завдань.

ВИСНОВКИ

Запропонований метод вирішує проблеми підвищення ефективності робочого процесу в хмарних обчисленнях. Метод включає використання трьох різних алгоритмів для визначення пріоритетів завдань, відображення продуктивності віртуальної машини та, нарешті, обчислення робочого діапазону, споживання енергії та порушення SLA.

Для реалізації методу було використано граф DAG для представлення різних завдань на основі часу, необхідного для їх виконання. Необхідний час залежить від часу, необхідного для ЦП, використання пам’яті, пам’яті та пропускної здатності каналів обміну даними.

Було промодельоване використання методу за допомогою симулятора робочого процесу. Виконано порівняння запропонованого методу із популярними алгоритмами на основі робочого процесу. Запропонований нами підхід моделюється шляхом обчислення пріоритетів завдань, а потім він передається планувальнику для створення розкладів на точних віртуальних машинах, мінімізуючи порушення SLA та споживання енергії.

Порівняння за різними алгоритмами планування, тобто Max–Min, FCFS, MCT, Min–Min, RATS-HM та Round Robin, з запропонованим методом показало, що він перевершує за вказаними параметрами, тобто порушення SLA на 22% та споживання енергії на 15%.

Запропонований метод можна вдосконалити, щоб мінімізувати час міграції. Запропонований метод можна вдосконалити, щоб мінімізувати час міграції. У цьому дослідженні використовувався метаевристичний підхід, але все ж дослідження проблем планування в хмарній парадигмі є дуже динамічними, і за такого підходу неможливо передбачити тип майбутнього робочого навантаження на хмарну систему. Тому, в майбутній роботі можливо застосувати техніку машинного навчання для ефективного планування завдань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. B. Gul, I. A. Khan, S. Mustafa, O. Khalid, S. S. Hussain, and D. Dancey, “CPU and RAM energy-based SLA-aware workload consolidation techniques for clouds,” IEEE Access, vol. 8, pp. 62990–63003, 2020.

2. E. I. Elsedimy and F. Algarni, “Toward enhancing the energy efficiency and minimizing the SLA violations in cloud data centers,” Applied Computational Intelligence and Soft Computing, vol. 2021, Article ID 8892734, 14 pages, 2021.

3. N. D. Vahed, M. Ghobaei-Arani, and A. Souri, “Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review,” International Journal of Communication Systems, vol. 32, no. 14, Article ID e4068, 2019.

4. F. Fakhfakh, H. H. Kacem, and A. H. Kacem, “Workflow scheduling in cloud computing: a survey,” in 2014 IEEE 18th International Enterprise Distributed Object Computing ConferenceWorkshops and Demonstrations, pp. 372–378, IEEE, Ulm, Germany, 2014.

5. Y. Gao, S. Zhang, and J. Zhou, “A hybrid algorithm for multiobjective scientific workflow scheduling in IaaS cloud,” IEEE Access, vol. 7, pp. 125783–125795, 2019.

6. Y. Wang, H. Liu, W. Zheng et al., “Multi-objective workflow scheduling with Deep-Q-Network-based multi-agent reinforcement learning,” IEEE Access, vol. 7, pp. 39974–39982, 2019.

7. S. Mustafa, K. Sattar, J. Shuja et al., “SLA-aware best fit decreasing techniques for workload consolidation in clouds,” IEEE Access, vol. 7, pp. 135256–135267, 2019.

8. Z. Chen, K. Lin, B. Lin, X. Chen, X. Zheng, and C. Rong, “Adaptive resource allocation and consolidation for scientific workflow scheduling in multi-cloud environments,” IEEE Access, vol. 8, pp. 190173–190183, 2020.

9. P. Wang, Y. Lei, P. R. Agbedanu, and Z. Zhang, “Makespandriven

workflow scheduling in clouds using immune-based PSO algorithm," IEEE Access, vol. 8, pp. 29281–29290, 2020.

10. F. Yao, C. Pu, and Z. Zhang, "Task duplication-based scheduling algorithm for budget-constrained workflows in cloud computing," IEEE Access, vol. 9, pp. 37262–37272, 2021.

11. L. Zhang, L. Wang, Z. Wen, M. Xiao, and J. Man, "Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems," IEEE Access, vol. 8, pp. 205099–205110, 2020.

12. N. Malik, M. Sardaraz, M. Tahir, B. Shah, G. Ali, and F. Moreira, "Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds," Applied Sciences, vol. 11, no. 13, Article ID 5849, 2021.

13. R. Anitha and C. Vidyaraj, "Workload and SLA violation prediction in cloud computing," in 2019 Third International Conference on Inventive Systems and Control (ICISC), pp. 582– 587, IEEE, Coimbatore, India, 2019.

14. H. R. Faragardi,M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 6, pp. 1239–1254, 2020.

15. M. N. Aktan and H. Bulut, "Metaheuristic task scheduling algorithms for cloud computing environments," Concurrency and Computation: Practice and Experience, vol. 34, no. 9, Article ID e6513, 2022.

16. Y. Hu, H. Wang, and W. Ma, "Intelligent cloud workflow management and scheduling method for big data applications," Journal of Cloud Computing, vol. 9, Article ID 39, 2020.

17. Y. Cui and Z. Xiaoqing, "Workflow tasks scheduling optimization based on genetic algorithm in clouds," in IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 6–10, IEEE, Chengdu, China, 2018.

18. R. N. Talouki, M. H. Shirvani, and H. Motameni, “A heuristicbased task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms,” Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 8, Part A, pp. 4902–4913, 2022.

19. R. N. Talouki, M. H. Shirvani, and H. Motameni, “A hybrid meta-heuristic scheduler algorithm for optimization of workflow scheduling in cloud heterogeneous computing environment,” Journal of Engineering, Design and Technology, vol. 20, no. 6, pp. 1581–1605, 2022.

20. M. H. Shirvani and R. N. Talouki, “A novel hybrid heuristicbased list scheduling algorithmin heterogeneous cloud computing environment for makespan optimization,” Parallel Computing, vol. 108, Article ID 102828, 2021.

21. M. H. Shirvani, “A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems,” Engineering Applications of Artificial Intelligence, vol. 90, Article ID 103501, 2020.

22. M. Tanha, M. H. Shirvani, and A. M. Rahmani, “A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments,” Neural Computing and Applications, vol. 33, pp. 16951–16984, 2021.

23. M. H. Shirvani and R. N. Talouki, “Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach,” Complex & Intelligent Systems, vol. 8, pp. 1085–1114, 2022.

24. Y. A. Alaie, M. H. Shirvani, and A. M. Rahmani, “A hybrid biobjective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach,” The Journal of Supercomputing, vol. 79, pp. 1451–1503, 2023.

25. M. Sohani and S. C. Jain, “A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing,” IEEE Access, vol. 9, pp. 62653–62664, 2021.

26. S.Mustafa, K. Bilal, S. U. R.Malik, and S. A.Madani, “SLA-aware energy efficient resource management for cloud environments,” IEEE Access, vol. 6, pp. 15004–15020, 2018.

27. P. K. Bal, S. K. Mohapatra, T. K. Das, K. Srinivasan, and Y.-C. Hu, “A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques,” Sensors, vol. 22, no. 3, Article ID 1242, 2022.

28. Саранча С.М., Якименко А.С., Баляба Ю.В., Сєрих О.О. Методи розподілення віртуальних машин за хмарними ресурсами. Проблеми інформатизації: Матеріали одинадцятої міжнародної науково-технічної конференції. –Баку – Харків – Бельсько-Бяла , 16 – 17 листопада 2023 року, с.50.