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 підвищення ефективності використання ресурсів в системах хмарних обчислень

(тема)

Виконав:

студент II курсу, групи КСМзм-22-1

Якименко А.С.
(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 Якименко Анастасії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи підвищення ефективності використання ресурсів 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 ресурсів.

Постачальники хмарних послуг (AWS, GCP, Azure).

Методи розподілу завдань за хмарними ресурсами.

4. Перелік питань, що потрібно опрацювати у роботі

Аналіз предметної області.

Методи підвищення ефективності систем хмарних обчислень.

Результати експериментальних досліджень.

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: 58 с., 15 рис., 5 табл., 1 дод., 63 джерела.

БАГАТОРІВНЕВИЙ ПАРАЛЕЛИЗМ, ХМАРНІ ПЛАТФОРМИ, СЕРВІСИ, ВИСОКОПРОДУКТИВНІ ОБЧИСЛЕННЯ, МОДЕЛІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ.

Хмарні обчислення є багатообіцяючою платформою для запуску великих програмних систем на основі моделі оплати за використання. У хмарних обчисленнях зниження споживання енергії та забезпечення безпеки для планування робочого процесу є ключовими напрямками досліджень. Основний фокус існуючих алгоритмів, а саме методу рою частинок (particle swarm optimization, PSO), Round Robbin(RR), SJF, Min-Min, Min-Max тощо, є на основі часу виконання та вартості виконання програм. Однак ці алгоритми не змогли адекватно визначити енергоспоживання, використання ресурсів і безпеку в плануванні робочого процесу. Для вирішення цієї проблеми пропонується багатоцільова структура планування. Запропоновано фреймворк, який виконує динамічне планування робочого процесу за допомогою універсальної унікальної ідентифікації — Блейка (UUID-Blake), алгоритмів Manhattan Distance-Partition around (MD-PAM), Linear Scaling-Crow Search Optimization (LS-CSO), Anova-Recurrent Neural Network.

ABSTRACT

Master’s thesis: 58 pages, 15 figures, 5 tables, 1 appendice, 63 sources.

MULTI-LEVEL PARALLELISM, CLOUD PLATFORMS, SERVICES, HIGH-PERFORMANCE COMPUTING, PARALLEL COMPUTING MODELS.

Cloud computing is a promising platform for running large software systems based on a pay-per-use model. In cloud computing, reducing energy consumption and providing security for workflow scheduling are key research areas. The main focus of the existing algorithms, namely particle swarm optimization (PSO), Round Robbin(RR), SJF, Min-Min, Min-Max, etc., is based on the execution time and cost of the programs. However, these algorithms failed to adequately determine energy consumption, resource utilization, and security in workflow planning. To solve this problem, a multi-objective planning framework is proposed. A framework is proposed that performs dynamic planning of the workflow using the universal unique identification — Blake (UUID-Blake), Manhattan Distance-Partition around (MD-PAM), Linear Scaling-Crow Search Optimization (LS-CSO), Anova-Recurrent Neural algorithms Network.

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

Одним з основних напрямків розвитку хмарних обчислень є надання середовищ високопродуктивних обчислень, де задіяно багато віртуальних машин (ВМ), мереж, сховищ даних и програмних систем. Ці ресурси виділяються користувачам із колективного пулу ресурсів із обмеженим керуванням або взаємодією [1]. Три основні послуги, які надає хмарна технологія, включають програмне забезпечення як послугу (SaaS), платформу-як- послуги (PaaS), а також інфраструктуру як послугу (IaaS) [2], з яких більшість видатною моделлю є IaaS.

Ця модель надає користувачеві попередньо налаштовані віртуальні машини (ВМ) із хмарного середовища [3] [4]. Нескінченна кількість обчислювальних ресурсів і засобів зберігання може бути досягнута за допомогою віртуальних машин [5]. Зі збільшенням попиту на такі хмарні сервіси кількість користувачів збільшується з кожним днем [6]. Через це зросло непередбачуване використання ресурсів віртуальних машин, тому що сервери хмарних центрів обробки даних використовуються або надмірно, або недостатньо, що призводить до незбалансованого стану [7] [8]. Як наслідок, це погіршує загальну продуктивність і використання ресурсів у хмарі серверів.

Для максимального використання ресурсів і виконання завдань за мінімальний час необхідна розробка ефективних алгоритмів робочого процесу [9]. Визначним підходом до моделювання додатків із високою обробкою даних є робочий процес, який виконується в доменах хмарних обчислень [10]. Прямий нециклічний граф (Direct Acyclic Graph, DAG) позначає робочий процес, де обчислювальні роботи ілюструються вузлами графа, а залежності між завданнями графа ілюструються ребрами графа [11]. Наукові методи мають величезний вплив на розмір DAG [12]. Якщо розмір графа є невеликим, застосування методу є простим, в іншому випадку задача

набирає великого розміру і стає складною [13]. Методологія, яка виконує відображення завдань робочого процесу на гетерогенних та розподілених ресурсах обчислювальної системи, називається плануванням робочого процесу (Workflow Scheduling, WFS). Щоб задовольнити обмеження, визначені користувачем, для виконання завдань робочого процесу виділяється відповідна кількість ресурсів [14].

Для вирішення проблеми складних хмарних обчислень пропонується багатоцільова структура планування. Запропоновано фреймворк, який виконує динамічне планування робочого процесу за допомогою універсальної унікальної ідентифікації – Блейка (UUID-Blake), алгоритмів Manhattan Distance-Partition around (MD-PAM), Linear Scaling-Crow Search Optimization (LS-CSO), Anova-Recurrent Neural Network.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості наукових застосувань в хмарних обчисленнях

Необхідно визначити набір критеріїв до наукових програмних систем, щоб краще моделювати наукові застосування, також визначити відповідні обчислювальні ресурси для виконання та визначити кращу стратегію планування, наприклад:

- паралельна модель: визначає, чи буде програма визначена як одне послідовне завдання чи як кілька паралельних завдань, які можуть виконуватися на багатопроцесорних машинах або розподілених вузлах обробки;

- залежність завдань: пов’язана з потоком введення/виведення між завданнями; вказуючи, що завдання не може почати виконуватися, доки не будуть доступні вихідні дані досліджень, від яких воно залежить;

- використання ресурсів: у ньому вказується, що програми є інтенсивними за введенням/виведенням, інтенсивними по даним та інтенсивними за використанням центрального процесору.

Більшість наукових програмних систем мають дуже залежну структуру завдань, яка призначена для виконання в розподіленому обчислювальному середовищі. Така форма завдання у цих застосуваннях відноситься до наукових робочих процесів. На рисунках 1.1 – 1.5 показано набір популярних наукових робочих процесів, які імітують реальні наукові програми.

Cybershake (рисунок 1.1) та Broad Band (рисунок 1.2) використовуються для досліджень землетрусів, Montage (рисунок 1.3) задіяно в астрономічних дослідженнях, LIGO (рисунок 1.4) застосовується для виявлення гравітаційних хвиль, а Epigenomics (рисунок 1.5) – для дослідження біологічних питань. Кожен тип наукового дослідження вирішуватиметься шляхом виконання певного набору завдань, як позначено

кольорами на рисунку 1.1, залежно від відповідної програми. Крім того, для виконання кожного завдання можна використовувати одне або кілька паралельних завдань.

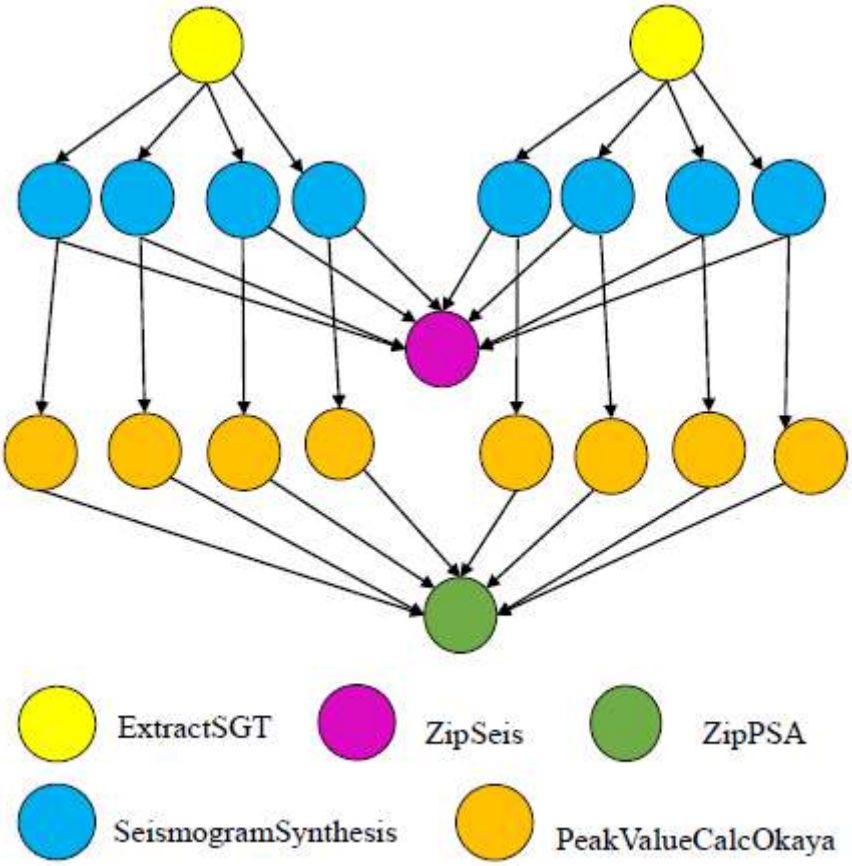

Рисунок 1.1 – Дослідження програмної системи Cybershake

В результаті, основні застосування, що інтенсивно працюють із процесором, і базові програми, пов’язані з пам’яттю, вимагають більше фізичної пам’яті для виконання.

Інтенсивнне використання процесора та пам'ті программами вимагають більше обчислювальних ресурсів для виконання завдання, а обчислювальні процеси з інтенсивним введенням/виведенням складаються із завдань, які споживають і генерують величезну кількість даних, і, отже, витрачають більшу частину свого часу на виконання операцій введення/ виведення. При цьому, в кластерних системах зазвичай використовуються мережні сховища.

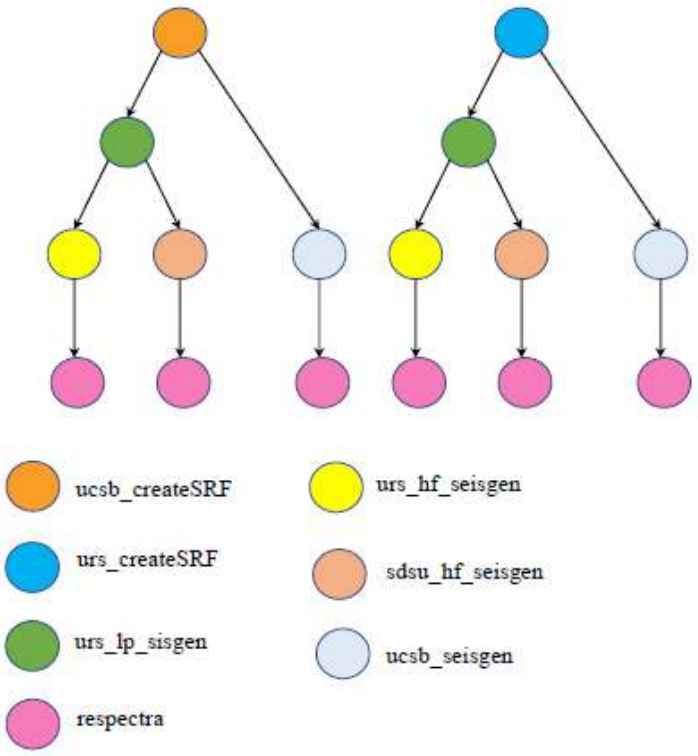

Рисунок 1.2 – Дослідження програмної системи Broadband

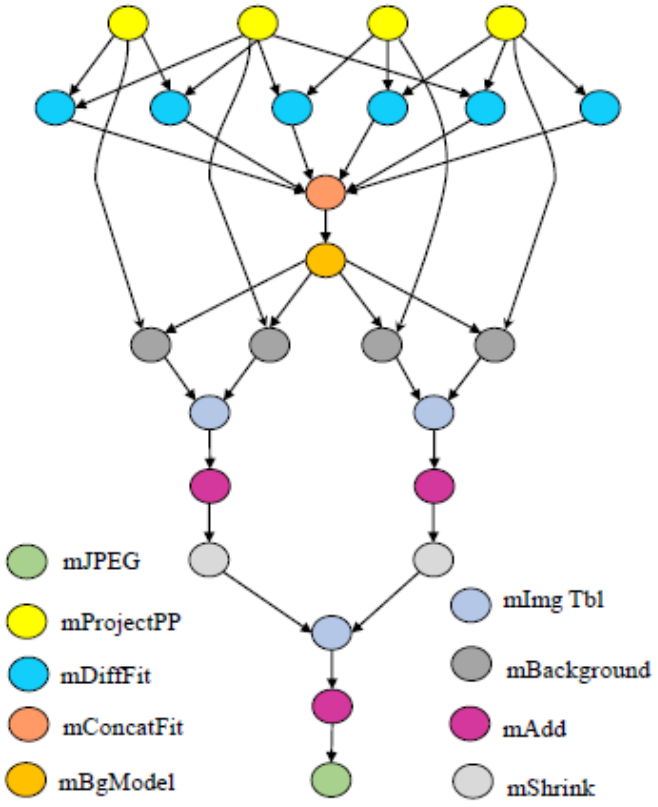

Рисунок 1.3 – Дослідження програмної системи Montage

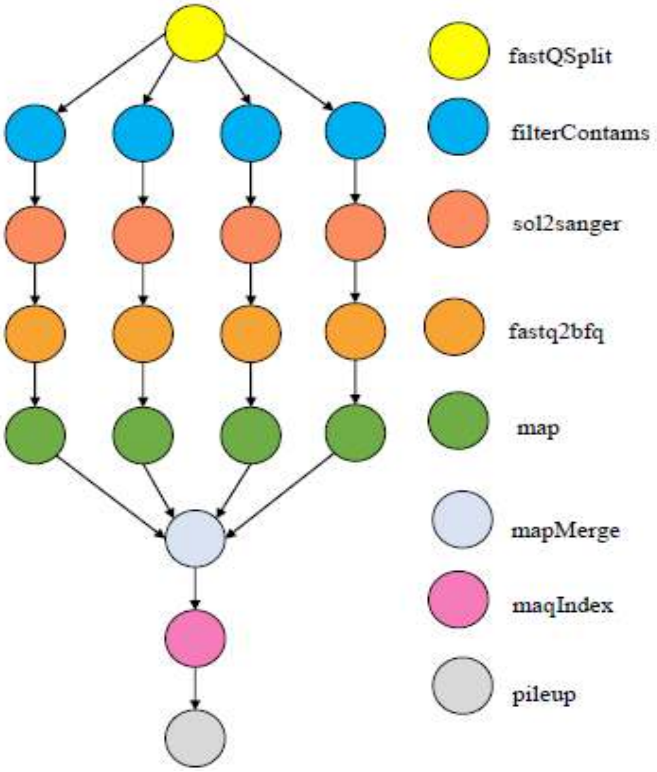

Рисунок 1.4 – Дослідження програмної системи Epigenomics

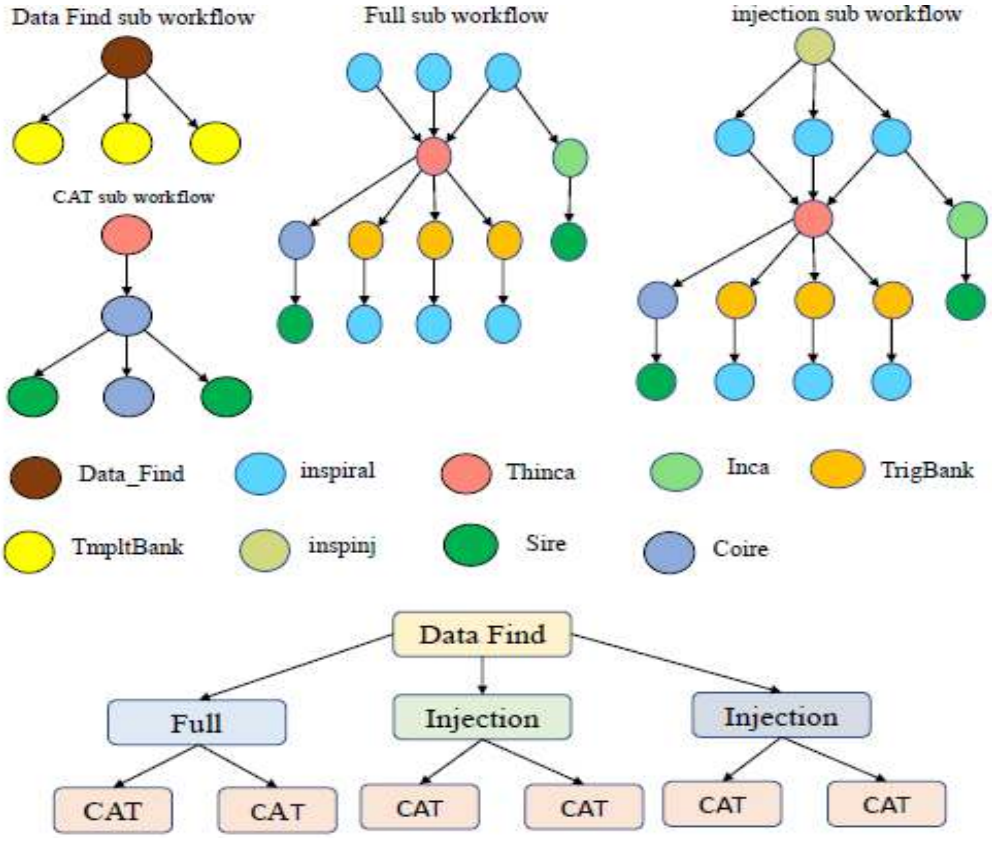

Рисунок 1.5 – Дослідження програмної системи LIGO

У плануванні робочого процесу існують різні моделі обчислювального процесу, а саме:

- паралельний обчислювальний процес,
- обчислювальний процес із розгалуженням;
- випадковий обчислювальний процес.

Існує декілька методологій для підвищення ефективності процесу WFS, хоча основні принципи хмарних обчислень, такі як еластичність і неоднорідність хмарних ресурсів, в існуючих роботах не розглядаються [15].

Більшість процедур планування в хмарних середовищах зосереджено на одному робочому процесі, а проблема непослідовних надходжень у великомасштабних WFS ігнорується [16]. Крім того, деякі важливі обмеження, такі як споживання енергії, використання ресурсів і безпека, не розглядаються в більшості робіт [17]. Безпечна передача завдань відповідним машинам зі зниженим енергомпоживанням є ключовим завданням у розподіленому обчислювальному середовищі з інтенсивним використанням даних. Існуючі роботи не змогли вирішити ці проблеми [18]. У даній роботі розробляється фреймворк, який розглядає обмеження енергії, недостатнє/надмірне використання ресурсів і обмеження безпеки для планування динамічних навантажень. У запропонованій структурі інтегруються LS-CSO та ANOVA-RNN з технікою хешування UUID-BLAKE для вирішення вищезгаданих проблем.

1.2 Аналіз стану наукових праць за напрямком досліджень

В роботі [19] розробили систему управління для оцінки кількох організацій у гібрідній хмарній моделі. Для розподілу ресурсів було продемонстровано підхід ідеального розподілу (Ideal Distribution Approach, IDA) у поєднанні з евристичним центрованим алгоритмом Enhanced IDA (EIDA). Ці алгоритми визначили досяжний та ідеальний графік виконання робочого процесу та ефективно зменшили застосування Makespan (MS) і

улучшили єфективность у випадку обмеження часу виконання. У зв'язку з необхідністю відповідати умовам SLA та скорочувати час і витрати на обчислення, алгоритм IDA ефективно знаходив рішення. Ефективність алгоритму IDA була підвищена завдяки включенню фази балансу навантаження. Аналогічно з алгоритмом IDA, Enhanced IDA (EIDA) досягла збільшення ефективності застосування MS. Експериментальні результати показали, що EIDA має кращі результати щодо вартості виконання MS, ніж існуючі алгоритми, однак, використання ресурсів було недостатньо ефективним.

Для наукових досліджень із інтенсивним використанням даних Ніколас Хазекампет та інші [20] запропонували комбіноване керування статичним і динамічним зберіганням даних та коду застосувань. Було сформульовано трирівневий підхід: по-перше, щоб проаналізувати вимоги до пам'яті робочого процесу перед виконанням, алгоритм статичного аналізу вніс достовірне передбачення успіху чи невдачі; по-друге, ігноруючи взаємоблокування, яке виникало під час виконання, алгоритм керування онлайн-сховищем мав відповідати вимогам до зберігання для майбутніх завдань; і по-третє, споживання пам'яті для окремих завдань було обмежено онлайн-алгоритмом керування сховищем, який дозволяв надійні гарантії статичного оцінювання разом із алгоритмами динамічного керування. Як свідчать результати, виконання цих методів у трьох складних робочих процесах є кращим за існуючі методи. Тим не менш, у цій схемі був потрібен підвищене обчислювальне навантаження робочий процес застосування MS.

В роботі [21] запровадили структуру, яка складалася з численних кооперативних агентів. Тут були враховані фази планування завдань (Task Scheduling, TS) разом із забезпеченням ресурсами, а QoS, наданий споживачеві, обмежений. Усі TS разом із процесами надання ресурсів були створені за допомогою інтегрованої моделі, і вона також служила інструментом керування для користувальницьких додатків та ефективного

використання хмарних ресурсів. Він мав складний процес планування через те, що він покладався на підзавдання, і він добре виконував залежні одночасні завдання. Що стосується MS, використання ресурсів і вартості, разом із споживанням енергії, у результатах було досягнуто високоякісну продуктивність у порівнянні з іншими моделями управління хмарними ресурсами. Однак вплив зв'язку між завданнями не перевірявся.

Робота [22] презентувала ідею управління робочим процесом людського інтелекту (human intelligence workflow management, HIWM) для динамічного розподілу ресурсів, зберігання, обробки роботи та обчислення операцій для швидкого надання послуг доповненої реальності (Augmented Reality, AR) на різноманітних інтелектуальних мобільних пристроях, які покладалися на людське ставлення та були застосовні в майбутньому веб-середовищі. Покладаючись на опис метаданих разом із запитами користувачів AR, було виконано попередню обробку, щоб зменшити час відповіді служби в HIWM. Була продемонстрована модель динамічного розподілу роботи для обробки великих даних для сервісів AR. Він також використовує хмарну інфраструктуру залежно від можливостей комп'ютера. Що стосується результатів, заснованих на запитах сервісу AR, 40-56% часу роботи було скорочено порівняно з існуючими моделями, тоді як неефективність виникає в процесі складного робочого процесу.

В роботі [23] розглядається механізм, орієнтований на робочий процес виконання TS у хмарному середовищі. Ця процедура робочого циклу виконувала величезну кількість взаємозалежних завдань кластером обчислювальних ресурсів із різнорідними можливостями. Тут суттєвим моментом був ефективний розподіл завдань, внаслідок чого склад виконання MC був знижений. Крім того, щоб вказати набір взаємозалежних завдань, на DAG було виконано подовження алгоритму max-min. Що стосується MS, ця методика перевершила існуючі методи, коли тестування виконувалося в стандартних наукових робочих процесах.

Під час виконання робочого процесу ігнорувалися деякі основних принципів хмарних обчислень, що було тут негативною стороною.

В роботі [24] запропонований HEFT-ACO компроміс між вартістю та робочим часом призначений для покращення вимог до якості обслуговування. Автори в першу чергу стурбовані пошуком найкращого рішення для призначення завдань віртуальній машині. Для призначення завдань розклад робочого процесу розділено на дві частини, перша з яких використовує HEFT для планування завдань на основі рангу, а друга з яких використовує ACO для планування готових завдань. Результати експерименту показали, що HEFT-ACO перевершив поточні алгоритми з точки зору вартості та MS. Однак енергоспоживання та безпека не здійснювалися ефективно.

На рисунку 1.6 наведено огляд оцінок робочого процесу та цілей планування, які використовуються в різних алгоритмах. Це також показує, що автори розглядали робочі процеси SIPHT і LIGO як статичні з обмеженими параметрами, що призвело до низької продуктивності планування.

Algorithm	Type of Workflow used						Scheduling Objectives							
	Montage	SIPHT	LIGO	Cybershake	Epigenomics	Random workflow	Makespan	Resource Utilization	Deadline	Cost	Reliability	Security	Energy Consumption	Budget
BTS [26]	√	√	√	√	√	√	-	-	√	√	-	-	-	-
EMO WFS in cloud [8]	√	√	√	√	√	-	√	-	-	√	-	-	-	-
NMMWS [27]	√		-	-	-	-	√	√	-	-	-	-	-	-
HMOPSO for WFS [28]	√	√	√	√	√	-	-	-	-	√	-	-	√	-
RSO [29]	-	-	-	-	√	-	√	√	-	√	-	-	-	-
HMO Workflow Scheduling in IaaS Cloud [2]	√	√	√	√	-	-	√	-	-	-	-	-	-	-
GRP-HEFT [30]	√	√	-	√	-	-	√	√	-	√	-	-	-	√
Task Replication [31]	√	√	√	√	-	-	-	-	-	√	√	-	-	-
BDCWS [6]	√	-	√	-	√	-	-	-	√	√	-	-	-	√
MOWS [9]	√	-	-	√	-	-	√	-	-	-	-	√	-	-
TOPSIS [32]	√	-	√	-	√	-	-	√	√	√	-	-	-	-
ALO-PSO [33]	√	-	-	-	-	-	√	-	-	√	√	√	-	-
GMPSO [34]	√	√	-	√	√	-	√	√	-	-	-	-	-	-
REEWS [35]	-	-	-	-	-	√	-	-	-	-	√	-	√	-
CPCS [36]	√	√	-	√	-	√	-	-	√	-	√	√	-	-

Рисунок 1.6 – Оцінки обчислювального процесу та цілі, що використовуються в алгоритмах управління науковими застосуваннями

Автор BTS врахував усі робочі процеси, включаючи випадкові робочі процеси, але не взяв до уваги багатоцільові параметри, такі як час виконання, використання ресурсів і споживання енергії. Існуючі алгоритми планування не стосувалися безпечного планування; безпека є основною проблемою в хмарних обчисленнях з точки зору доступу неавторизованих осіб і зміни даних (завдань), що може призвести до іншої атаки на заплановані к виконанню завдання.

Щоб усунути вищезазначені обмеження, пропонується підхід, відповідно котрому, на початку створюється безпечна схема розподілу в процесі планування шляхом дозволу авторизованих користувачів у хмарне середовище за допомогою техніки хешування UUID-BLAKE, після чого відбувається звернення до моніторингу віртуальної машини за допомогою A-RNN для покращення планування продуктивності з точки зору використання ресурсів і споживання енергії.

1.3 Постановка задач дослідження

Метою роботи є підвищення ефективності виконання складних наукових застосувань в системах хмарних обчислень шляхом зменшення енергоспоживання та кількісті обчислювальних ресурсів

Для реалізації мети роботи необхідно вирішити настіпня задачі:

- запропонувати ресурсоефективну структуру на основі планування робочого процесу для хмарних обчислень із використанням UUID-BLAKE, MH-PAM, A-RNN і LS-CSO;

- розробити безпечну політику планування робочого процесу шляхом генерації хеш-кодів. Хеш-код генерується за допомогою алгоритму UUID-BLAKE для підвищення автентичності хмарного сервера, дозволяючи законному користувачеві отримати до нього доступ;

- зменшити тривалість створення та накладні витрати на систему за рахунок об’єднання подібних завдань в пули або кластери. Процес кластеризації виконується за допомогою MH-PAM;

- віртуальні машини відстежувать за допомогою A-RNN, який передбачає аналіз використання ресурсів. На основі наявності ресурсів створюється динамічний робочий процес. Щоб подолати недолік RNN, включена функція ядра радіального базису ANOVA;

- за допомогою алгоритму LS-CSO вибирати найбільш відповідні віртуальні машини для виконання динамічного робочого процесу. Можливості кожної віртуальної машини оцінюється, а завдання призначаються на найбільш підходящій ресурс. Завдяки цьому споживання енергії зменшується, а ресурси використовуються ефективно.

2 МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ СИСТЕМ ХМАРНИХ ОБЧИСЛЕНЬ

2.1 Загальні підходи

У парадигмі хмарних обчислень планування робочого процесу є ключовим питанням і створює постійну складну проблему. У хмарній інфраструктурі використовується велика кількість віртуальних машин. Таким чином, вибір найбільш підходящої віртуальної машини для кожного завдання є складним процесом. Отже, у роботі запропоновано ресурсоефективне та безпечне планування робочого процесу на хмарному сервері з використанням MH-PAM, LS-CSO та ARNN із технікою хешування UUID-BLAKE. Запропонована структура фреймворку представлена на рисунку 2.1.

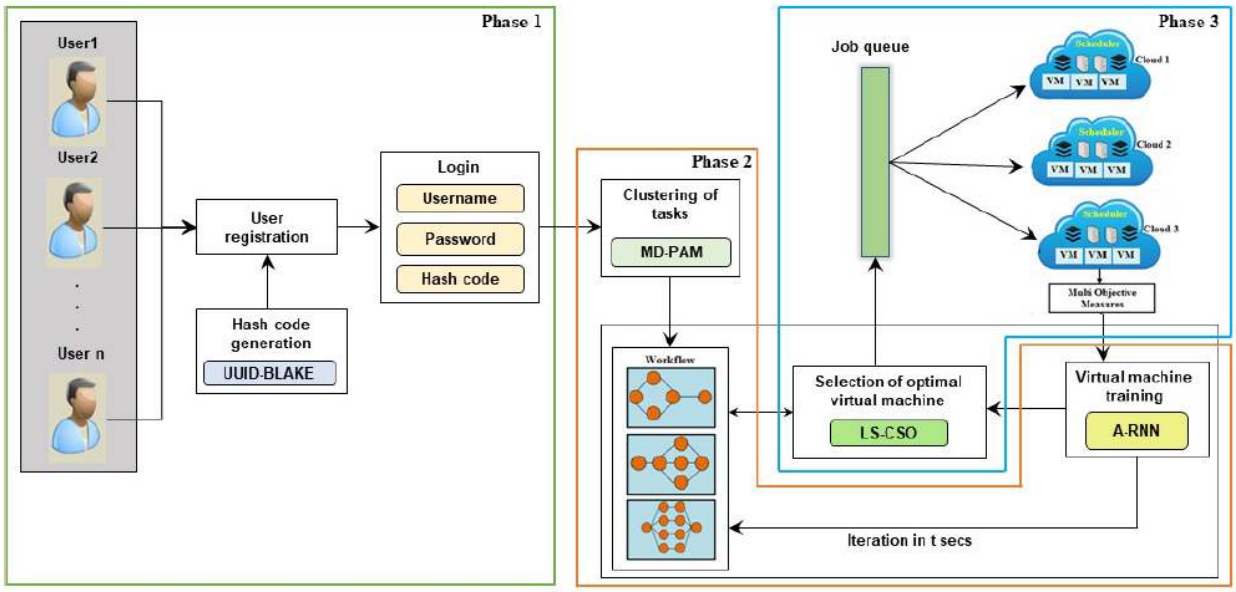

Рисунок 2.1 – Мультіоб'єктний фрейворк на базі LS-CSO

Процес роботи системи будується на трьох важливих фазах. Перший етап забезпечує безпечну авторизацію користувачів на хмарному

сервері. На другому етапі роботи, яку запитують користувачі, плануються на сервері таким чином, щоб загальний час, а також енергія, необхідна для виконання завдання, були мінімальними. Потім третій етап зосереджується на ефективному виборі віртуальних машин, які підходять для завдання, яке запитує користувач. Завдяки цьому споживання ресурсів, споживання часу, а також споживання енергії можна різко скоротити.

2.2 Автентифікація користувачів

На першому етапі проходить автентифікація користувача. Спочатку користувачі реєструються на відповідному хмарному сервері, надаючи свої дані та вимоги до свого хмарного сервера, такі як використання пропускної здатності, жорсткий диск, оперативна пам’ять, процесор з деталями оцінки та деякі інші деталі.

Після завершення процесу реєстрації для кожного зареєстрованого користувача генерується хеш-код. Цей процес генерації хеш-коду покращує автентичність хмарного сервера, дозволяючи законному користувачеві отримати доступ до хмарного сервера. У цій роботі хеш-код генерується за допомогою алгоритму хешування UUID-BLAKE. Тут універсальний унікальний ідентифікатор (UUID) об’єднано з існуючим алгоритмом хешування BLAKE для покращення складності хеш-коду.

Хеш-функція BLAKE-32 використовується для хешування повідомлення ms, і перед цим повідомлення доповнюється 66 бітами або більше, потім воно стає кратним 512. Подання бітів у повідомленні розглядається як останні 64 біти є двійковим представленням довжини в бітах незаповненого повідомлення. Це повідомлення про заповнення знову розбивається на 512-бітні блоки та повторно вводиться до функції стиснення. Зі старим хеш-значенням, який був хешованим 64-бітними бітами лічильника доповнюється 128-бітами. Крім того, старе хеш-

значення вказано як вектор ініціалізації для першого блоку. Під час доповнення лічильника останній блок встановлюється на нуль. У цій роботі UUID надається як вхідні дані для алгоритму BLAKE для забезпечення складності хеш-коду [37].

Нижче наведено кроки, включені в алгоритм BLAKE.

Функція стиснення BLAKE має значення «4» як вхідні дані, які виражаються як значення ланцюжка *(H)=H0,H1, ... , H7*– це внутрішні стани хеш блока повідомлення – *block(ms) = ms0, ms1, ... , ms15*.

Алгоритм починається з розділення повідомлення користувача *ms* на блокі по 512 байт. За потреби останній блок доповнюється нулем.

Стан *(S) = S0, ... ,S3* вибирається користувачем за бажанням і встановлюється на нульове значення, якщо база не потрібна. Це лише використовується завданнями, такими, як наприклад, рандомізоване хешування.

Counter *(C) =C0,C1* – лічильник. Якщо останній блок не містить жодного біта з вихідного повідомлення, лічильник скидається до нуля.

Функція стиснення *H, ms, S, C* є

$$H' = compress(H, ms, S, C). \tag{2.1}$$

Ініціалізація, циклічна ітерація та фіналізація — це три етапи, які проходить функція стиснення.

Початкові стани $L_0, \ldots, L_{15}$ представлено матрицею 4X4:

$$L = \begin{pmatrix} L_0 & L_1 & L_2 & L_3 \\ L_4 & L_5 & L_6 & L_7 \\ L_8 & L_9 & L_{10} & L_{11} \\ L_{12} & L_{13} & L_{14} & L_{15} \end{pmatrix}, \tag{2.2}$$

512-бітовий стан *L* підтримується у функції стиснення, яка

представлена у вигляді матриці 4x4 із 32-бітових слів. Поточний хеш, значення стану, значення таймера C і 256-бітна константа *C* використовуються для ініціалізації цього стану. Задано початковий стан функції стиснення

$$\begin{pmatrix} L_0 & L_1 & L_2 & L_3 \\ L_4 & L_5 & L_6 & L_7 \\ S_0 \oplus C_0 & S_1 \oplus C_1 & S_2 \oplus C_2 & S_3 \oplus C_3 \\ S_4 \oplus C_4 & S_5 \oplus C_5 & S_6 \oplus C_6 & S_7 \oplus C_7 \end{pmatrix}. \quad (2.3)$$

Матриця стану повторюється протягом 10 разів після ініціалізації. Настійно рекомендується використовувати простіші цикли під час проектування BLAKE, оскільки доведено, що це підвищує безпеку.

Функція, яка виконується на кожному циклі складається з 8 станів, а саме: $\lambda_0, \ldots, \lambda_7$, які відповідають за зміну даних (плутання) алгоритму BLAKE.

$$\lambda_0(L_0 \quad L_4 \quad L_8 \quad L_{12}) \qquad \lambda_1(L_1 \quad L_5 \quad L_5 \quad L_{13})$$
$$\lambda_2(L_2 \quad L_6 \quad L_{10} \quad L_{14}) \qquad \lambda_3(L_3 \quad L_7 \quad L_{11} \quad L_{15})$$
$$\lambda_4(L_0 \quad L_5 \quad L_{10} \quad L_{15}) \qquad \lambda_5(L_1 \quad L_6 \quad L_{11} \quad L_{12})$$
$$\lambda_6(L_2 \quad L_7 \quad L_8 \quad L_{13}) \qquad \lambda_7(L_3 \quad L_4 \quad L_9 \quad L_{14})$$

$$a \leftarrow a + b + (ms_{\sigma_{Ru}(2i)} \oplus C_{\sigma_{Ru}(2i+1)}), \quad (2.4)$$

$$d \leftarrow (d \oplus a) >> 16, \quad (2.5)$$

$$c \leftarrow (c + d), \quad (2.6)$$

$$b \leftarrow (b \oplus c) >> 12, \quad (2.7)$$

$$a \leftarrow a + b + (ms_{\sigma_{Ru}(2i+1)} \oplus C_{\sigma_{Ru}(2i)}), \quad (2.8)$$

$$d \leftarrow (d \oplus a) >> 8, \quad (2.9)$$

$$b \leftarrow (b \oplus c) >> 7. \quad (2.10)$$

Тут константи вказані як C_j для j = 0,…, 15 і перестановки Z_{16} позначено як σ_{Ru} для Ru=0,1,…,9. У документації до алгоритму BLAKE вони пропонуються обидва. Нарешті, вихідне хеш-значення H формулюється наступним чином. Останній крок виконується функцією стиснення після десяти/чотирнадцяти ітерацій λ.После чего новое значение H' виділяється з $L_0,\ldots,L_{15}$ с значением цепочки:

$$H' \leftarrow H_i \oplus S_{i \bmod 4} \oplus H_i \oplus H_{i+8}, для\, i = 0,1,...,7 . \quad (2.11)$$

Після успішної генерації хеш-коду, він зберігається як на хмарному сервері, так і в системі користувача. Коли під час входу в хмару користувач повинен пройти автентифікацію, он повинен ввести ім'я користувача, пароль та хеш-код.

Після успішної генерації хеш-коду користувач повинен ввести ім'я користувача, пароль і хеш-код. Потім ці три дані перевіряються сервером перевірки. Якщо всі введені дані правильні, то система дозволяє користувачеві отримати доступ до хмарного сервера. Таким чином, користувачі можуть використовувати хмарні ресурси для завершення своєї роботи.

2.3 Планування розподілу завдань за ресурсами

Після надсилання завдань найважливіші та необхідні атрибути, такі як кількість процесорів, необхідних для виконання завдання, кількість інструкцій N вхідні, а такі розміри, як великі, середні та малі завдання D, витягуються з робочого процесу. Ці атрибути значно сприяють ефективній кластеризації завдань. Математичне представлення для атрибутів WF наведено нижче:

$$WF = \{ N_{CPU}, N_{Ins}, D \}.$$

Після цього завдання групуються на основі цих атрибутів. Цей процес кластеризації різко зменшує накладні витрати системи шляхом призначення одного або кількох малих завдання в один блок виконання, який називається завданням. У цьому запропонована структура, кластеризація подібних завдань виконується за допомогою методу медоїд (Manhattan Distance-base Partitioning around Medoid, MD-PAM). PAM є свого роду алгоритм K-медоїдів. Алгоритм k- медоїдів це підхід до кластеризації, пов'язаний із k-середніми алгоритмами кластеризації. Таким чином, PAM на основі K-медоїдів є менш чутливим до шуму та викидів до k-значення, оскільки він використовує медоїд як центри кластерів замість mean, який використовується в k-means. В алгоритмі PAM, мінімальна відстань між медоїд і завдання розраховуються в термінах Манхеттенської відстані (MD), тим самим формується більше схожих завдань на кластер. Алгоритм MD-PAM радикально покращує точність кластеризації. Метод розподілу наведено нижче.

Етап1. На початковому етапі оптимальний медоїд k ініціалізується шляхом додавання кількості об'єктів з мінімальною відстанню до всіх інших об'єктів. Під об'єктами розуміються завдання. Відстані між об'єктами *i* та *j* відповідно A*i* та B*j* обчислюються за допомогою формули відстані, яка задана як

$$MD = \sum_{i=1}^{n} \left| A_i - B_j \right|. \qquad (2.12)$$

Етап 2. Об'єкт $i(i \in O)$ розглядається як кандидат, який додається до k. Потім TG_i обчислюється загальний приріст TG_i для кожного об'єкта:

$$TG_i = \sum_{j \in O} Max\{H_i - \vartheta(j,i),0\}, \tag{2.13}$$

де j є об'єктом O, крім i. У випадку, $H_i > \vartheta(j,i)$ якість кластеризації покращується.

Етап 3. Після обчислення загального приросту всіх об'єктів в O, об'єкт G, який має максимум, TG_g вибирається, що математично сформулюється як

$$k = k \cup \{G\}, \tag{2.14}$$

$$O = O - \{G\}. \tag{2.15}$$

Ці кроки повторюються, доки всі об'єкти v не будуть обрані.

Етап 4: Фаза обміну покращує якість кластеризації шляхом оптимізації набору вибраних об'єктів, які закінчуються, враховуючи всі пари обміну $(v,\emptyset) \in k \cup O$ та обчислення ефекту $\xi v\phi$ на сумі відмінностей між об'єктами та їх центрами кластерів шляхом заміни v і ϕ, а потім перенесення ϕ з множини O до k. Ефект $\xi v\phi$ розраховується за виразом

$$\xi_{v\phi} = \sum \{v_{tv\phi} | t \in O\}, \tag{2.16}$$

де, $v_{tv\phi}$ позначає внесок кожного об'єкта t в множину O для заміни v і O. Якщо $(t,v) > H_t$ або $\vartheta(t,v) = H_t$, потім можна обчислити $v_{tv\phi}$ як

$$v_{tv\phi} = \begin{cases} Min\{\vartheta(t,\phi) - H_t, 0\} & \vartheta(t,v) > H_t \\ Min\{\vartheta(t,\phi,\varepsilon_t) - H_t\} & \vartheta(t,v) = H_t \end{cases}. \tag{2.17}$$

Етап 5. Пара (v, ϕ)з мінімумом $\varepsilon_{k\phi}$ вибирається, щоб визначити, чи виконано заміну чи ні. Якщо $\varepsilon_{k\phi} < 0$, відбувається процес обміну і

повертається до початкової фази обміну. В іншому випадку за значення вибирається медоїда.

Етап 6. На наступному етапі об’єкти, які є найближчими до медоїдів з мінімальними відмінностями між об’єктами, групуються в кластери, і це визначається як

$$C_P = \{C_1, C_2, C_3, \ldots, C_n\}, \tag{2.18}$$

де $C_1, C_2, C_3, \ldots, C_n$ означає кількість сформованих кластерів. Цей кластер містить більше подібних завдань, щоб завдання можна було виконувати ефективно з обмеженим споживанням ресурсів і часу.

Після успішної кластеризації кількість запитів збирається з числа користувачів, і ці запити надсилаються до систем керування робочим процесом (WMS). Зазвичай велике робоче навантаження вимагає значної кількості обчислювальних ресурсів, у якийсь момент сервер виснажує свої ресурси, і він не може обробляти вхідні запити. Отже, щоб впоратися з цим, було використано WMS. Таким чином, WMS забезпечує ефективне використання ресурсів хмари та керує обмеженим споживанням енергії. WMS знаходиться на головній машині, яка планує завдання робочого циклу на основі доступних ресурсів. Крім того, він контролює виконання та керує вхідними даними, проміжними файлами та виводом файлів результатів.

Спеціальна програма Mapper показує стан робочого циклу та створює виконуваний робочий процес на основі доступності віртуальних машин, які відстежуються та прогнозуються ARNN. Ця програма відображення робочого процесу визначає відповідне програмне забезпечення разом із апаратними ресурсами, необхідними для впровадження схеми розподілу. Таким чином, при обмеженому ET виконується кількість завдань. Крім того, для оптимізації продуктивності Mapper реструктурує робочий процес.

2.4 Моніторинг віртуальних машин

На цьому етапі віртуальні машини відстежуються раніше призначення робочих процесів віртуальним машинам на основі статистичних даних машин і та обчислених характеристик. Таким чином, віртуальні машини контролюються за допомогою A-RNN (рисунок 2.2). Традиційний RNN використовує для активації sigmoid або softmax функції класифікації. Проте загасаючий градієнт існує у процесі зворотного поширення і є недоліком цих функцій активації, що призводить до затримки навчання а також низької продуктивність класифікації. Подолати вищезазначені проблеми можна за рахунок радіального базиса ANOVA в функції ядра, який добре підходить для більш глибоких мереж та поєднується з алгоритмом RNN. Крім того, зменшується частота помилок.

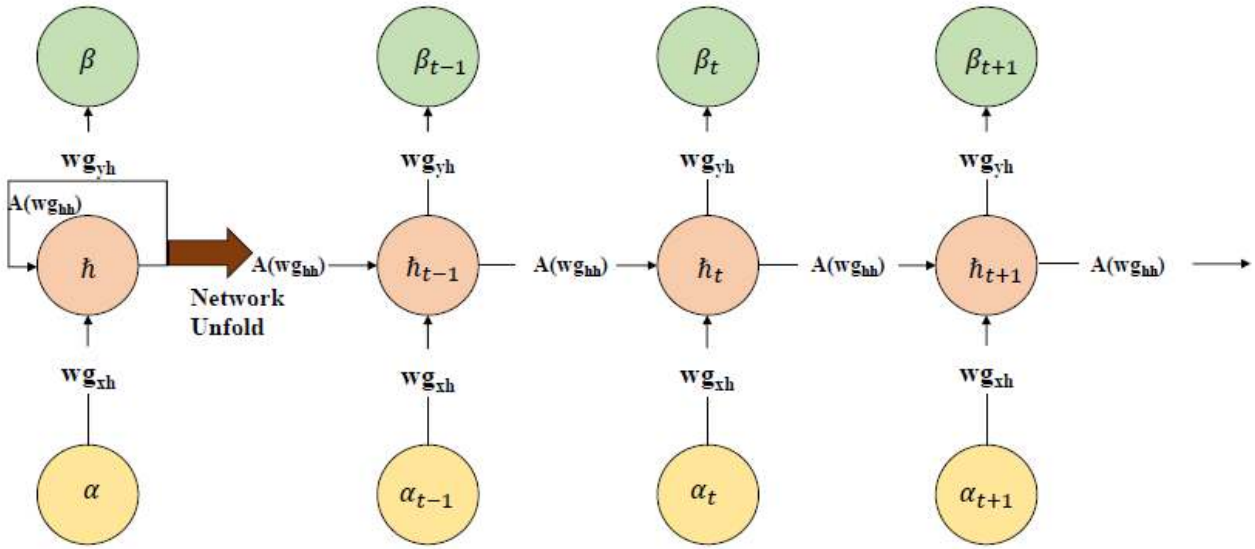

Рисунок 2.2 – Структура A-RNN класифікатора

RNN – це нейронна мережа, в якій результат від попереднього кроку (2.18) використовується як вхідні дані для наступного кроку. Прихований стан є важливою особливістю RNN, яка зберігає інформацію про послідовність. Кожен прихований шар (HL) у традиційній нейронній мережі

має власний набір ваг і упереджень. Наприклад, ваги та зміщення HL1, HL2 і HL3 є (W1, B2), (W2, B2). і (W3, B3) відповідно. Це означає, що кожен HL незалежний від іншого. Таким чином, вони не запам'ятовують значення попереднього шару. З іншого боку, RNN перетворює незалежну активацію на залежну, призначаючи однакові ваги та зміщення всім рівням. Таким чином, шляхом подачі кожного значення до наступного HL, складність збільшення параметра та запам’ятовування кожного попереднього виходу зменшується.

Крок 1: шляхом ітерації наступної послідовності від *t=1* до *T*, ANOVA-RNN вхідні дані $\alpha_{In} = \{\alpha_1, \alpha_2, \ldots \alpha_t\}$ з урахуванням прихованої послідовності векторів $\hbar_{Lyr} = \{\hbar_1, \hbar_2, \ldots \hbar_t\}$ разом із послідовністю вихідних векторів $\beta_{out} = \{\beta_1, \beta_2, \ldots \beta_t\}$. HL вимірюється як

$$\hbar_{Lyr} = \theta_{act}\left\lfloor wg_{a\hbar}\alpha_t + wg_{\hbar\hbar_{t-1}}\hbar_{t-1} + B_a \right\rfloor, \tag{2.19}$$

де ваги матриці (наприклад, значення ваги, приховане введенням, вказано як $wg_{a\hbar}$, значення ваги HL прихованого шару позначається як $wg_{\hbar\hbar}$, вектор зміщення позначається як B_a, а радіальна базова ядерна функція ANOVA пропонується як

$$\theta_{act} = \sum_{k=1}^{n} Exp\left(-\sigma(\alpha_{X^k} - \alpha_{Y^k})^2\right), \tag{2.20}$$

де вхідні дані позначаються як α_X та α_Y.

Крок 2: Далі, щоб оцінити ефективність віртуальних машин, підраховується вихідний $\alpha = wg_{\alpha\beta}\hbar_t + B_a$ $\beta_{out} = \sigma_S\left\lfloor wg_{\alpha\beta}\hbar_t + B_a \right\rfloor$рівень. Використовуючи функцію сигмоїдної активації (σ_S), активується вихідний шар, потім проводиться оцінювання:

$$\beta_{out} = \sigma_S\left\lfloor wg_{\alpha\beta}\hbar_t + B_a \right\rfloor, \tag{2.21}$$

$$\alpha = wg_{\alpha\beta}\hbar_t + B_a . \quad (2.22)$$

Крок 3. Наступне рівняння використовується для обчислення сигмоїдної функції активації

$$\sigma_S = \frac{1}{1+\varepsilon^{-\alpha}} . \quad (2.23)$$

Крок 4: Згодом, шляхом розрахунку різниці між початковим значенням α_a і прогнозованим значенням α_p, аналізується значення втрат, яке визначається як

$$Lost = (\alpha_a - \alpha_p)^2 . \quad (2.24)$$

Що стосується попередніх значень функцій, віртуальні машини навчаються ефективно, якщо значення втрати моделі дорівнює нулю (*Lost=0*). Оновлюючи значення ваги, виконується зворотне поширення, якщо значення втрати *Lost! =0*.

Нарешті, використання ресурсів віртуальної машини ефективно визначається за допомогою ANOVA-RNN за t секунд для кожної ітерації, завдяки цьому робочі процеси генеруються відповідно до використання ресурсів для кластерного завдання.

2.5 Вибір віртуальних машин

Після цього, за допомогою алгоритму LS-CSO, вибираються найбільш відповідні віртуальні машини для виконання кластерного завдання. Тут аналізуються ресурси та можливості кожної віртуальної машини; таким чином є відповідні віртуальні машини виділені для виконання кластерних

завдань. Отже, цей процес різко зменшує споживання ресурсів та споживання енергії, а також прискорює загальний процес виконання.

Розгляними процес вибору віртуальних машин за допомогою алгоритма LS-CSO, який є новим метаевристичним алгоритмом оптимізації. Алгоритм заснован на ідеї ворон зберігати надлишок їжі в схованках разом із відновленням запасів, коли це необхідно (популяційний підхід). У CSO, розгорнувши алгоритм LS, ймовірність поінформованості підвищується в алгоритмі LS-CSO (рисунок 2.3). Таким чином, оптимальний VM вибирається точно. Нижче наведені основні кроки CSO.

Algorithm 1 Pseudocode for LS-CSO algorithm

Input:Number of virtual machines
Output: Selection of optimal virtual machines
Begin
Initialize the positions $VM = \{vm_1, vm_2, vm_3, \ldots\ldots vm_n\}$ randomly.
Evaluate the memory of each vm by the initial position.
Compute the fitness value for each vm.
Update the position of vm according to the random another vm.
Improve the awareness probability by using,
$g(h) = \frac{s(x) - \min s(x)}{\max s(x) - \min s(x)}$
if $R_j \geq AP_{j,Itr}$
$x_{i,Itr} = x_{i,Itr} + R_i \times Fl_{i,Itr} \times (Mem_{j,Itr} - x_{i,Itr})$
else
Relocate the VM randomly
endif
Check the feasibility of new position
Update the VMs memory
The process continue until the best solutions obtained
End

Рисунок 2.3 – Псевдокод алгоритму LS-CSO

Крок 1. Ініціалізація. Спочатку кожна доступна віртуальна машина ініціалізується випадковим чином у *d*-вимірності. *N* означає розмір пулу ресурсів. У цьому випадку кожен елемент наступної матриці позначає VM. Процес ініціалізації виражається як

$$VM = \begin{pmatrix} vm_1^1 & vm_2^1 & \cdots & vm_d^1 \\ vm_1^2 & vm_1^2 & \cdots & vm_1^2 \\ \vdots & \vdots & \vdots & \vdots \\ vm_1^N & vm_2^N & \cdots & vm_d^N \end{pmatrix}. \quad (2.25)$$

Кожна пам'ять віртуальної машини ініціалізується. Передбачається, що віртуальні машини зберігають свої ресурси на своїх початкових позиціях, оскільки вони не мають досвіду на початковій ітерації

$$Mem = \begin{pmatrix} M_1^1 & M_2^1 & \cdots & M_d^1 \\ M_1^2 & M_1^2 & \cdots & M_1^2 \\ \vdots & \vdots & \vdots & \vdots \\ M_1^N & M_2^N & \cdots & M_d^N \end{pmatrix}. \quad (2.26)$$

Крок 2. Оцінка функції придатності. Щоб проаналізувати кожну окрему віртуальну машину, розраховується функція відповідності; тоді її значення приймається як початкове значення пам’яті. Кожна віртуальна машина збирає свою пам'ять у стані, якому відповідає значення параметру Mem_j.

Крок 3. Оновлення стану віртуальної машини. VM оновлює свою позицію, вибираючи випадкову іншу VM, наприклад *xi* генеруючи випадкове значення *xj,* аналізує відповідне значеннння Mem_j. Якщо це значення перевищує ймовірність усвідомлення *AP*, то віртуальна машина *xi* слідуватиме за *xj*, щоб дізнатися *Memj*. Для того, щоб підвищити ймовірність усвідомлення, використовується техніка лінійного масштабування, яка виконується за допомогою виразу:

$$g(h) = \frac{s(x) - \min s(x)}{\max s(x) - \min s(x)}. \quad (2.27)$$

Крок 4. Оновлення до нової позиції. Віртуальна машина оновлює свою позицію, вибираючи випадкову іншу віртуальна машину так, що на парі *xji* проходить зайняття Mem_j.

Потім новий *xj* розраховується наступним чином:

$$x_{i+1} = \begin{cases} x_i + R_i \times Fl_i \times (Mem_j - x_i), \text{якщо } R_j \geq AR_j \\ x_i, \quad \text{уншому випадку} \end{cases}. \tag{2.28}$$

Крок 5: Перевіряється доцільність нових позицій кожної окремої віртуальної машини. Віртуальна машина оновлює свою позицію, якщо її нова позиція можлива. Інакше віртуальна машина зберігає своє поточне положення.

Крок 6. Оцінка функції відповідності нових позицій кожної окремої віртуальної машини, розраховується функція відповідності.

Крок 7: Оновлення пам’яті для кожної позиції у матриці (2.26):

$$Mem_{i+1} = \begin{cases} x_{i+1}, \text{якщо } F(x_{i+1}) \leq F(Mem_i) \\ Mem_i, \quad \text{уншому випадку} \end{cases}, \tag{2.29}$$

де значення цільової функції пропонується як *F*. Віртуальна машина оновлює свою пам’ять новою позицією, якщо значення функції відповідності нової позиції віртуальної машини краще, ніж значення функції відповідності запам’ятаної позиції. Тому оптимальні віртуальні машині формулюються як

$$vm_{opt} = \{vm_1, vm_2, \ldots vm_n\}. \tag{2.30}$$

Після визначення найбільш підходящих віртуальних машин, кластеризовані завдання впорядковуються в чергу для виконання. Таким

чином, завдання плануються за віртуальними машинами таким чином, що процес виконання споживає мінімальну кількість енергії.

Таким чином, ми разглянули повний цикл управління розподілом програмних завдань за обчислювальними ресурсами, який включає наступні етапи: автентифікація користувача та завдання, кластерізація завдань, розподіл завдань за віртуальними машинами.

На прикинці було розроблено алгоритм розподілу завдань за ресурсами в хмарном середовищі.

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

В цьому розділі наведено результати експериментів, які ілюструють повну оцінку кінцевого результату для запропонованої системи. Результати експериментів підкреслюють ефективність її роботи. Моделювання проводилося за допомогою середовища Cloudsim та наборів входних даних HCSP [39] та GWA-T-12 Bitbrains [40], які є публічними та до них є доступ в Інтернеті.

3.1 Конфігурація експериментального кластеру та оцінка продуктивності алгоритму BLAKE

Моделювання відбувалось на 64 - розрядних віртуальних машинах, які працювали на платформі Azure (фірми Microsoft) з 8 ядрами кожен. Таблиці 3.1 містить конфігурацію для кожної віртуальної машини.

Таблиця 3.1 – Сервіси інфраструктури

Хмарна платформа	Тип	Ядра	Процесор	Пам'ять	Тип пам'яті	Віртуалізація	Мережа
Azure	A10	8	Intel Core i7-11800H	64 GB	DDR3	Hyper-V	10 gigabit ethernet

BLAKE2B добре працює на 64-розрядних процесорах на Intel Core i7-11800H та може обробляти 1 гігабайт на секунду (рисунок 3.1) На рисунку можна побачити, що BLAKE2B працює краще, ніж SHA-1, BLAKE2S, MD5, SHA-512 на процесорах Intel. Навіть самі швидкі алгоритми MD5 та SHA-512 показують результати трішки гірше ніж запропонований алгоритм.

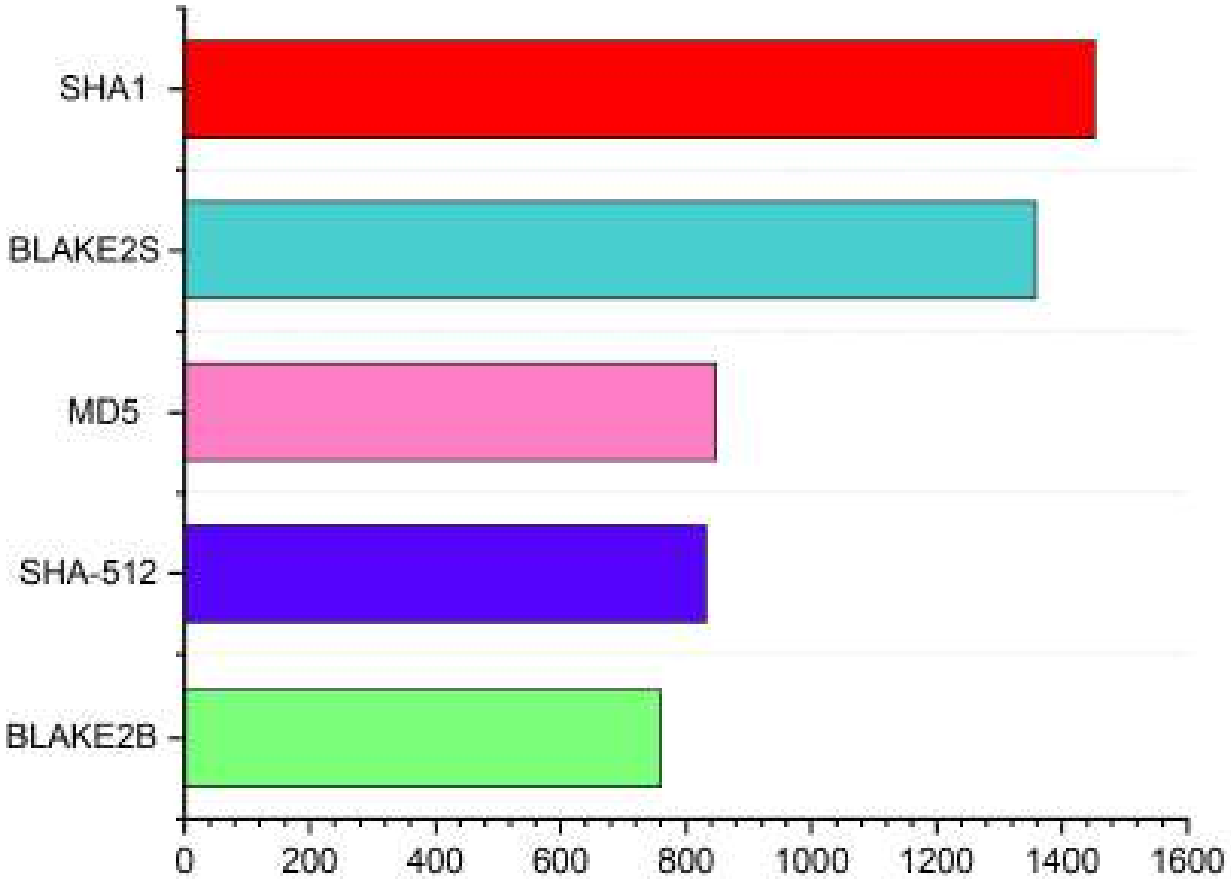

Рисунок 3.1 – Порівняння часу виконання різних алгоритмів хешування з розробленою модифікацією BLAKE2B

Таблиця 3.2 визначає якісне порівняння різних сервісів, що надають відомі алгоритми хешування в порівнянні з розробленим алгоритмом BLAKE2B. У випадку BLAKE2B автентифікація є важливою під час завантаження на сервер даних, тоді як для інших методів хешування автентифікація не є важливою. З даних таблиці також можна зробити висновки, що розроблений метод є більш якісним, зокрема він більш безпечніший та має особливості зберігання даних при передачі.

Загалом, можна зробити висновки, що метод BLAKE2B є швидшим, як показано на рисунку 3.1, і безпечнішим, ніж інші методи хешування, та володіє якісними характеристиками, які у сукупності ліпше, ніж у ушних методів, які було розглянуто в роботі.

Таблиця 3.2 – Порівняльна таблиця між BLAKE2B та іншими методами з точки зору сервісної інфраструктури

Послуги	BLAKE2B	MD5	SHA1	BLAKE2S	SHF-12
Безпека даних	+	+	+	+	+
Цілісність даних в БД	+	+	+	+	+
Цілісність даних при передачі	+	-/+	-	+	-/+
Автентифікація при зава-нтажені даних на сервер	+	-/+	-/+	-/+	-/+
Автентифікація при обміні даними між серверами	+	-	-	-/+	-

3.2 Аналіз продуктивності MH-PAM

Запропонований метод MH-PAM оцінюється з точки зору часу кластеризації, а результати порівнюються з різними існуючими методами, такими як середній зсув, нечіткі C-середні значення, K-середні значення та PAM, щоб визначити гідність методу.

Рисунок 3.2 ілюструє продуктивність MH-PAM. Показані чотири кластеризовані групи, що сформовані на основі довжини завдання для ефективного скорочення часу виконання.

На рисунку 3.3 пропонований час для кластера MH-PAM порівнюється з часом, які витрачають інші методи, таких як Means, Shift, Fuzzy C-Means, K-means і PAM. Час кластеризації — це кількість часу, необхідна класифікатору для кластеризації завдань. Завдяки групуванню одного чи кількох невеликих завдань в єдину одиницю виконання завдань, кластеризація зменшує кількість робіт і споживання енергії. Це означає, що запропонований алгоритм працює краще, ніж поточний алгоритм.

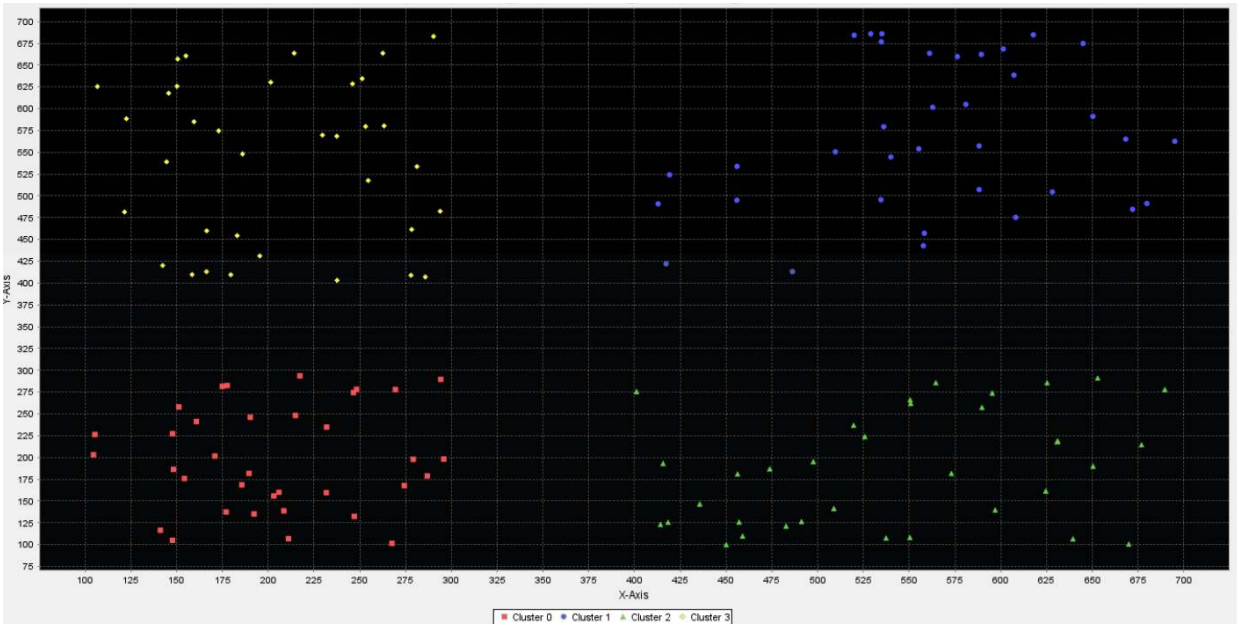

Рисунок 3.2 – Кластерізація завдань з використанням методу MH-PAM

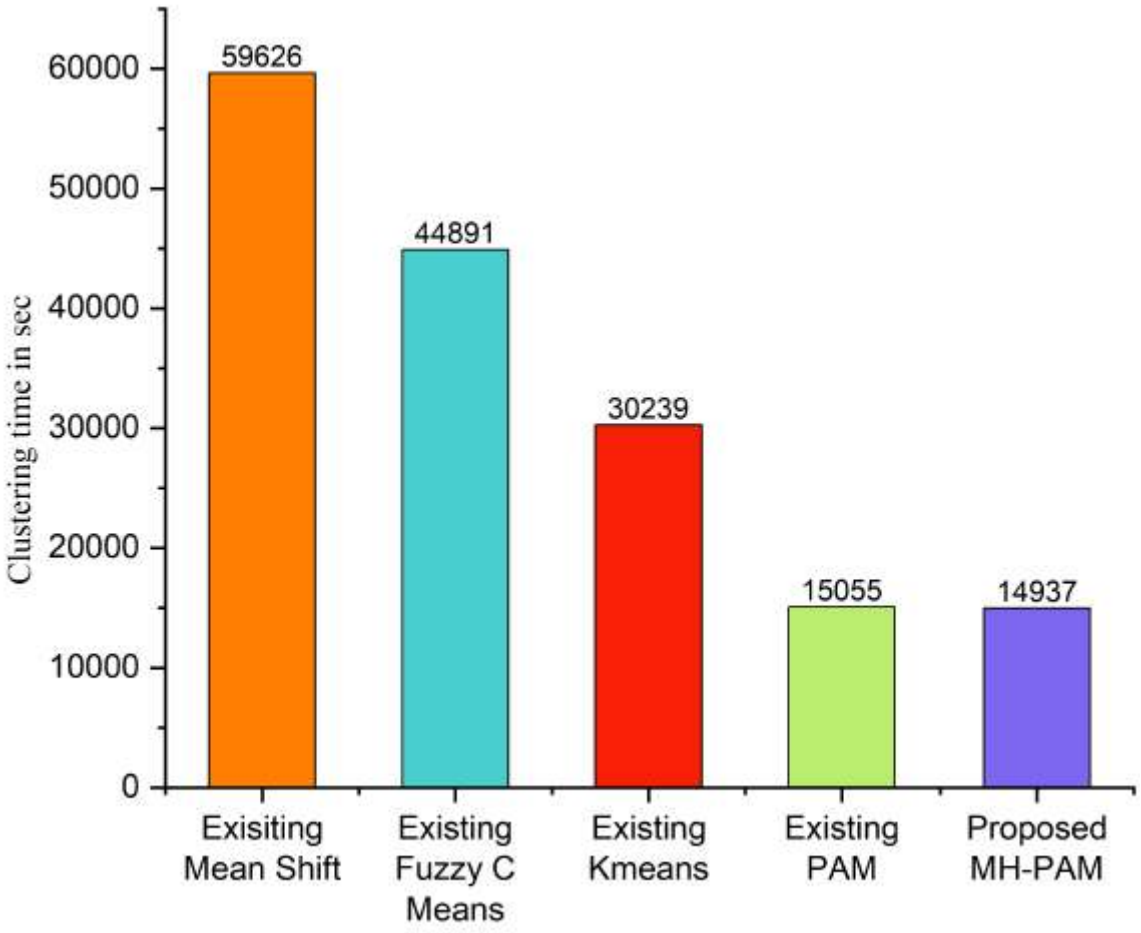

Рисунок 3.3 – Порівняння часу виконання різних алгоритмів планування

3.3 Аналіз ефективності алгоритму A-RNN

Такі показники, як похибка, точність, специфічність, чутливість, запам’ятовування та час навчання, аналізуються для Sermetrics, та порівнюються з сучасними алгоритмами, як Deep Neural Network (DNN), RNN, Deep Belief Network (DBN) і Convolution Neural Network (CNN).

Що стосується специфічності, чутливості та точності, оцінка ефективності A-RNN разом із існуючими підходами, такими як RNN, DBN, DNN та CNN, наведена в таблиці 3.3. Запропонований алгоритм A-RNN забезпечує високі показники точності, чутливості та специфічності, такі як 97,62%, 97,60% і 97,65%. тоді як існуюча система досягає низького рівня точності, чутливості та специфічності, який коливається між 93,66%-95,93%, 93,49%-95,81% і 93,82%-96,05% відповідно. Таким чином, запропонований алгоритм працює більш безпечно разом із керуванням надмірним використанням енергії.

Таблиця 3.3 – Оцінка ефективності запропонованого A-RNN

Техніка	Показники продуктивності (%)		
	точність	чутливість	специфічність
Запропонований A-RNN	97,62	97,60	97,65
RNN	95,93	95,81	96.05
ДБН	91,46	91,50	91,43
DNN	89,98	89,96	90.01
CNN	93,66	93,49	93,82

Порівняльна оцінка метричних значень, отриманих запропонованим алгоритмом A-RNN разом з існуючими методами, проілюстрована на рисунку 3.4. Якщо система хоче бути надійною та ефективною, значення показників повинні залишатися високими. Таким чином, у порівнянні з

існуючими RNN, DBN, DNN і CNN, запропонована методика A-RNN гарантує вищу точність, чутливість, а також рівень специфічності. Таким чином, запропонована система дає точні результати у WFS і ефективно управляє енергією.

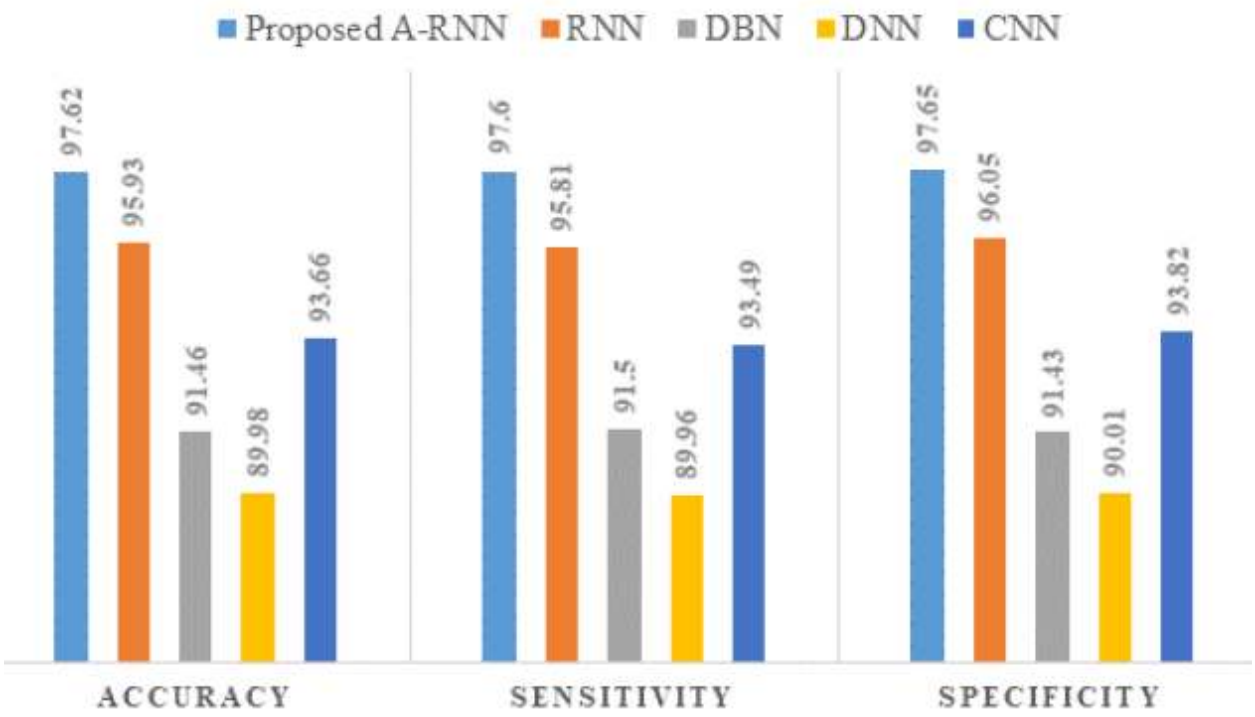

Рисунок 3.4 – Порівняння запропонованого A-RNN

У таблиці 3.4 запропонований A-RNN алгоритм разом з іншими існуючими показниками продуктивності моделей RNN, DBN, DNN і CNN, такими як точність, повнота та F-міра, включено. Значення схеми відоме завдяки високому рівню точності, повноти та F-міри. Для точності, нагадаємо, разом із F-мірою, запропонована методика отримала 97,60%, 97,61% та 97,59%; тоді як існуючий підхід досяг середніх показників 92,43%, 92,67% і 92,66%, що є порівняно низьким. Таким чином, запропонований A-RNN мінімізує величезні ускладнення та покращує стійкість WFS завдяки підвищеній безпеці та низькому споживанню енергії.

Таблиця 3.4 – Оцінка продуктивності запропонованого A-RNN

Техніка	Показники продуктивності (%)		
	точність	повнота	F-міра
Запропонований A-RNN	97.60	97.61	97.59
RNN	95.81	95.18	95.76
DBN	91.45	91.50	91.68
DNN	89.96	89.69	89.36
CNN	93.49	93.38	93.85

На рисунку 3.5 показано порівняльну характеристику на основі даних таблиці 3.4.

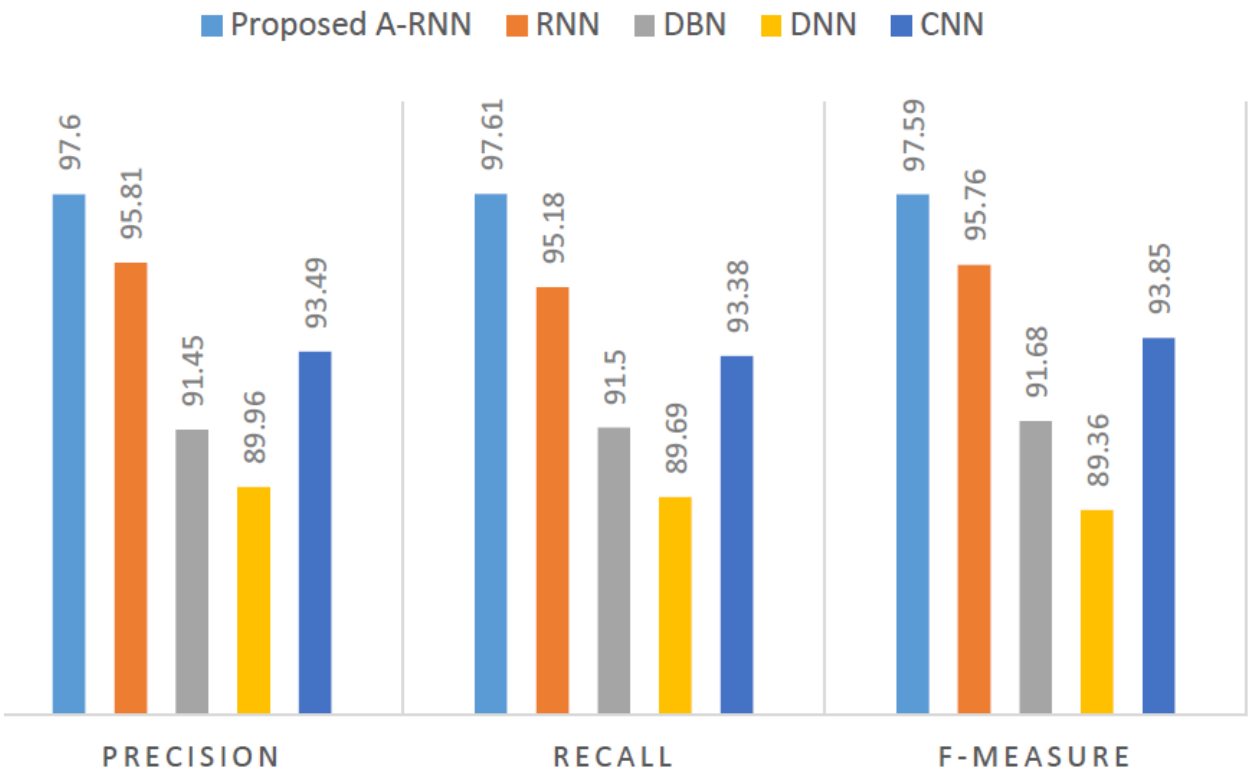

Рисунок 3.5 – Графічне представлення порівняння запропонованого A-RNN алгоритму щодо точності, повноти та F-міри

Порівняльна оцінка запропонованої методології з точки зору часу навчання представлена на рисунку 3.6. Запропонований A-RNN досягає високих діапазонів продуктивності від 97,6% до 97,59% для точності, повноти та F-міри, але для існуючих показників поточний RNN, DBN , DNN і CNN досягають низької продуктивності від 89,96% до 95,76%. Таким чином, запропонований підхід A-RNN перевершує інші існуючі методи та дає точні результати за надзвичайно складних умов.

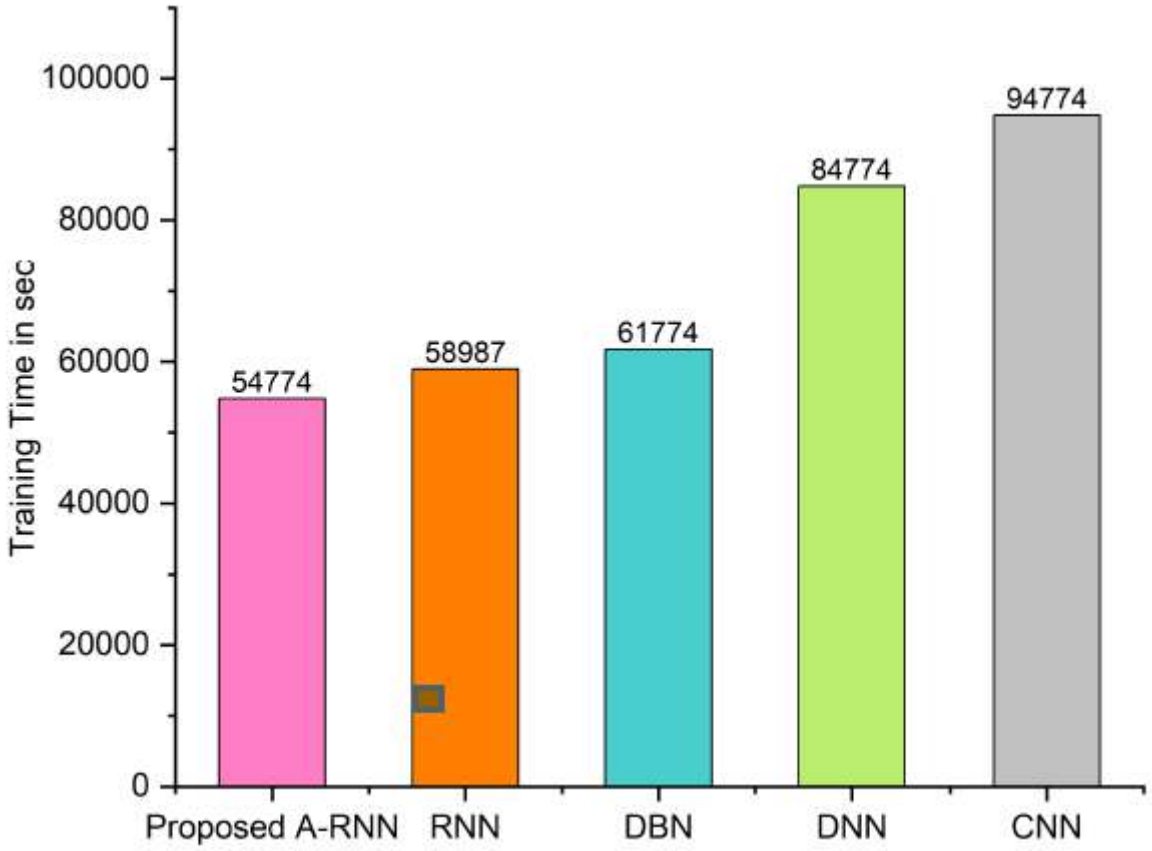

Рисунок 3.6 – Порівняння запропонованого A-RNN за часом навчання

Час навчання, витрачений запропонованим A-RNN алгоритмом, порівнюється з алгоритмами RNN, DBN, DNN і CNN на рисунку 3.6. Час навчання – це кількість часу, який витрачає класифікатор на навчання даних. Якщо час навчання системи великий, це вплине на ET усієї системи. Система повинна витрачати менше часу на навчання, якщо вона хоче бути

просунутою. Для завершення процесу навчання запропонованому A-RNN алгоритму потрібно 54774 мс; тоді як інші методи займають 58987 мс для RNN, 61774 мс для DBN, 84774 мс для DNN і 94774 мс для CNN. Таким чином, порівняно з існуючими підходами, запропонований A-RNN завершує весь процес навчання з малим часом обчислення.

На рисунку 3.7 порівнюється час тестування пропонованого A-RNN з часом існуючої роботи, наприклад RNN, DBN, DNN і CNN. Час тестування – це час, витрачений класифікатором на тестування файлів. Якщо час тестування великий, це погіршує продуктивність моделі. Кажуть, що модель є найкращою, коли тестування мінімальне.

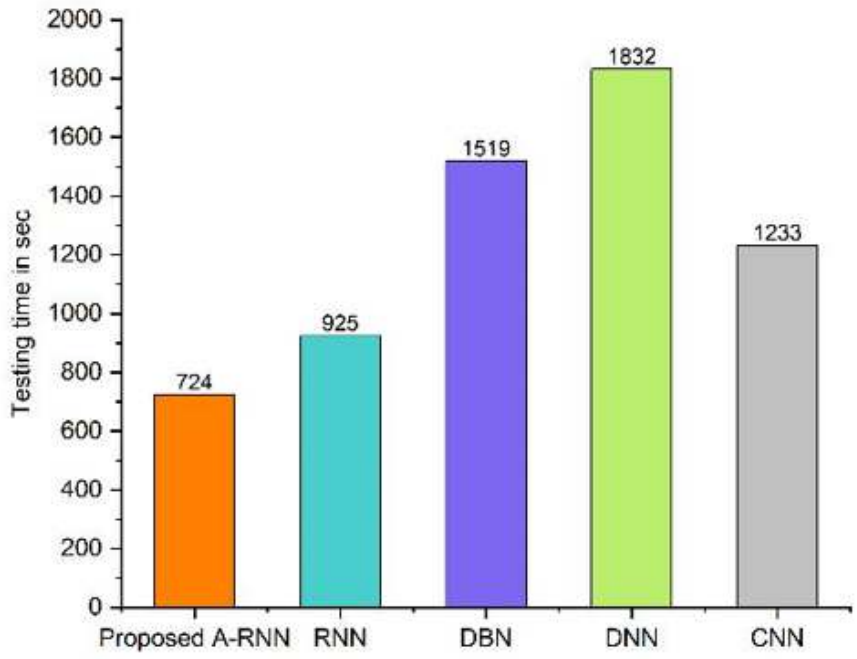

Рисунок 3.7 – Порівняння запропонованого A-RNN за часом тестування

Відповідно до цього запропонований A-RNN займає 724 мс, тоді як існуючі системи, такі як RNN, DBN, DNN і CNN, вимагають 925 мс, 1233 мс, 1519 мс і 1832 мс часу тестування. Таким чином, запропонований A-RNN завершує весь процес тестування з меншими витратами часу в порівнянні з існуючими.

ВИСНОВКИ

В кваліфікаційній роботі запропоновано нову структуру для наукового планування робочого процесу в хмарних обчисленнях. Ці рамки націлені на автентифікацію, споживання енергії, використання ресурсів, вартість та інше як багатоцільові параметри. Технологія реалізації з 3 етапів.

Перший е тап – фаза автентифікація, яка дозволяє користувачеві безпечне планування за допомогою методу Blake2b з UUID, на другому етапі завдання користувача спочатку кластеризуються в різні групи за допомогою методу MH-PAM за допомогою ресурсів A-RNN, які відстежуються для створення динамічного робочого процесу. На третьому етапі LS-CSO алгоритм використовується для оптимізації для планування робочого процесу на відповідних віртуальних машинах. Для підтвердження ефективності запропонованої системи наведено порівняльні експериментальні результати. Результати продемонстрували, що запропонована структура працює краще з точки зору багатоцільових метрик, які беруться до уваги, ніж поточні методи PSO, CSO та RR. Цей аналіз використовував загальнодоступні набори даних HCSP і GWA-T-12 Bitbrains. У подальшому роботу можна розширити за допомогою деяких передових нейронних мереж і виконання процесу планування робочого процесу в кількох хмарах з даними в реальному часі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kaushik Mishra, Jharashree Pati, and Santosh Kumar Majhi. A dynamic load scheduling in iaas cloud using binary jaya algorithm. Journal of King Saud University-Computer and Information Sciences, 34(8):4914–4930, 2022.

2. Yongqiang Gao, Shuyun Zhang, and Jiantao Zhou. A hybrid algorithm for multi-objective scientific workflow scheduling in iaas cloud. IEEE Access, 7:125783–125795, 2019.

3. Ying Xie, Yuanwei Zhu, Yeguo Wang, Yongliang Cheng, Rongbin Xu, Abubakar Sadiq Sani, Dong Yuan, and Yun Yang. A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment. Future Generation Computer Systems, 97:361– 378, 2019.

4. Yong Zhao, Youfu Li, Ioan Raicu, Shiyong Lu, Cui Lin, Yanzhe Zhang, Wenhong Tian, and Ruini Xue. A service framework for scientific workflow management in the cloud. IEEE Transactions on Services Computing, 8(6):930–944, 2014.

5. Zheyi Chen, Kai Lin, Bing Lin, Xing Chen, Xianghan Zheng, and Chunming Rong. Adaptive resource allocation and consolidation for scientific workflow scheduling in multi-cloud environments. IEEE Access, 8:190173–190183, 2020.

6. Naqin Zhou, Weiwei Lin, Wei Feng, Fang Shi, and Xiongwen Pang. Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment. Cluster Computing, pages 1–15, 2020.

7. Sarbjeet Singh, Maitreyee Dutta, et al. Critical path based scheduling algorithm for workflow applications in cloud computing. In 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring), pages 1–6. IEEE, 2016.

8. Zhaomeng Zhu, Gongxuan Zhang, Miqing Li, and Xiaohui Liu.

Evolutionary multi-objective workflow scheduling in cloud. IEEE Transactions on parallel and distributed Systems, 27(5):1344–1357, 2015.

9. Farzaneh Abazari, Morteza Analoui, Hassan Takabi, and Song Fu. Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm. Simulation Modelling Practice and Theory, 93:119–132, 2019.

10. Yuandou Wang, Hang Liu, Wanbo Zheng, Yunni Xia, Yawen Li, Peng Chen, Kunyin Guo, and Hong Xie. Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning. IEEE access, 7:39974–39982, 2019.

11. Jiagang Liu, Ju Ren, Wei Dai, Deyu Zhang, Pude Zhou, Yaoxue Zhang, Geyong Min, and Noushin Najjari. Online multi-workflow scheduling under uncertain task execution time in iaas clouds. IEEE Transactions on Cloud Computing, 9(3):1180–1194, 2019.

12. Zhongjin Li, Victor Chang, Haiyang Hu, Hua Hu, Chuanyi Li, and Jidong Ge. Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds. Information Sciences, 568:13–39, 2021.

13. Seyed Ziae Mousavi Mojab, Mahdi Ebrahimi, Robert Reynolds, and Shiyong Lu. icats: Scheduling big data workflows in the cloud using cultural algorithms. In 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), pages 99–106. IEEE, 2019.

14. Haibing Lu, Xi Chen, Qi Liu, Michele Samorani, Guojie Song, and Yanjiang Yang. Stochastic workflow authorizations with queueing constraints. IEEE Transactions on Dependable and Secure Computing, 18(4):1605–1619, 2020.

15. K Kanagaraj and S Swamynathan. Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud. Future Generation Computer Systems, 79:878–891, 2018.

16. Yang Cui and Zhang Xiaoqing. Workflow tasks scheduling optimization based on genetic algorithm in clouds. In 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pages 6–10. IEEE,

2018.

17. Kalka Dubey and Subhash Chander Sharma. An extended intelligent water drop approach for efficient vm allocation in secure cloud computing framework. Journal of King Saud University-Computer and Information Sciences, 34(7):3948–3958, 2022.

18. Longxin Zhang, Lan Wang, Zhicheng Wen, Mansheng Xiao, and Junfeng Man. Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems. IEEE Access, 8:205099–205110, 2020.

19. Kalka Dubey, Mahmoud Y Shams, Subhash Chander Sharma, Abdulaziz Alarifi, Mohammed Amoon, and Aida A Nasr. A management system for servicing multi-organizations on community cloud model in secure cloud environment. IEEE access, 7:159535–159546, 2019.

20. Nicholas Hazekamp, Nathaniel Kremer-Herman, Benjamin Tovar, Haiyan Meng, Olivia Choudhury, Scott Emrich, and Douglas Thain. Combining static and dynamic storage management for data intensive scientific workflows. IEEE Transactions on Parallel and Distributed Systems, 29(2):338– 350, 2017.

21. Ali Asghari, Mohammad Karim Sohrabi, and Farzin Yaghmaee. A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents. Computer Networks, 179:107340, 2020.

22. Hyun-Woo Kim, Jong Hyuk Park, and Young-Sik Jeong. Human-intelligence workflow management for the big data of augmented reality on cloud infrastructure. Neurocomputing, 279:19–26, 2018.

23. Harish Kumar Patnaik, Manas Ranjan Patra, and Rahul Kumar. A workflow based approach for task scheduling in cloud environment. Materials Today: Proceedings, 2021.

24. Ali Belgacem and Kadda Beghdad-Bey. Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost. Cluster Computing, 25(1):579–595, 2022.

25. Raza Abbas Haidri, Chittaranjan Padmanabh Katti, and Prem Chandra Saxena. Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. Journal of King Saud University-Computer and Information Sciences, 32(6):666–683, 2020.

26. Eun-Kyu Byun, Yang-Suk Kee, Jin-Soo Kim, Ewa Deelman, and Seungryoul Maeng. Bts: Resource capacity estimate for time-targeted science workflows. Journal of Parallel and Distributed Computing, 71(6):848–862, 2011.

27. Indrajeet Gupta, Madhu Sudan Kumar, and Prasanta K Jana. Efficient workflow scheduling algorithm for cloud computing system: a dynamic priority-based approach. Arabian Journal for Science and Engineering, 43:7945–7960, 2018.

28. Hatem Aziza and Saoussen Krichen. A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. Neural Computing and Applications, 32:15263–15278, 2020.

29. M Karpagam, K Geetha, and C Rajan. A reactive search optimization algorithm for scientific workflow scheduling using clustering techniques. Journal of Ambient Intelligence and Humanized Computing, 12:3199–3207, 2021.

30. Hamid Reza Faragardi, Mohammad Reza Saleh Sedghpour, Saber Fazliahmadi, Thomas Fahringer, and Nayereh Rasouli. Grp-heft: A budget-constrained resource provisioning scheme for workflow scheduling in iaas clouds. IEEE Transactions on Parallel and Distributed Systems, 31(6):1239–1254, 2019.

31. Samaneh Sadat Mousavi Nik, Mahmoud Naghibzadeh, and Yasser Sedaghat. Task replication to improve the reliability of running workflows on the cloud. Cluster Computing, 24:343– 359, 2021.

32. K Kalyan Chakravarthi, L Shyamala, and V Vaidehi. Topsis inspired cost-efficient concurrent workflow scheduling algorithm in cloud. Journal of King Saud University-Computer and Information Sciences, 34(6):2359–2369, 2022.

33. Jabir Kakkottakath Valappil Thekkepuryil, David Peter Suseelan, and Preetha Mathew Keerikkattil. An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing

environment. Cluster Computing, 24:2367–2384, 2021.

34. Haithem Hafsi, Hamza Gharsellaoui, and Sadok Bouamama. Genetically-modified multi-objective particle swarm optimization approach for high-performance computing workflow scheduling. Applied soft computing, 122:108791, 2022.

35. M. A. Vouk. Cloud computing : Issues, research and implementations. In Information Technology Interfaces, 2018. ITI 2018. 30th International Conference on, , June 2018. P. 31–40.

36. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: Toward a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1). 2019. P.50–55.

37. L. Johnson, A. Levine, and R. Smith. The 2019 horizon report, 2019. Austin, Texas, The New Media Consortium.

38. Foster, Y. Zhao, I. Raicu and S. Lu, “Cloud Computing and Grid Computing 360-Degree Compared,” Grid Computing Environments Workshop (GCE '08), 2008.

39. R. Buyya, R. Ranjan and R. N. Calheiros, Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. Proceedings of the Conference on High Performance Computing and Simulation, Leipzig, Germany. IEEE Press: New York, U.S.A., 21–24 June 2019. pp.1–11.

40. L. Vaquero, L. Rodero-Merino , J. Caceres and M. Lindner, A break in the clouds: towards a cloud definition. SIGCOMM Computer Communication Review, Volume 39, Number 1, January 2019. pp.50-55.

41. NIST definition of cloud computing, http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

42. L. Youseff, M. Butrico and D. Da Silva, Toward a Unified Ontology of Cloud Computing, Grid Computing Environments Workshop (GCE '08), 2008.

43. Six benefits of cloud computing. Available at http://web2.sys-

con.com/node/640237.

44. B. Hayes. Cloud computing. Communications of the ACM, 51(7). July 2018. pp.9–11.

45. B. P. Rimal, E. Choi and I. Lumb, A Taxonomy and Survey of Cloud Computing Systems. Fifth International Joint Conference on INC, IMS and IDC. 2019, pp 44–51.

46. Gathering clouds of XaaS! https://www.ibm.com/developerworks/ mydeveloperworks/blogs/sbose/entry/gathering_clouds_of_xaas?lang=en

47. B. Sonisky, chapitre 1, cloud computing bible,wiley publishing inc 2011

48. Q. Zhang, L. Cheng and R. Boutaba, Cloud computing : state of the art and research challenges. Journal of Internet Services and Applications, vol. 1, 2020 pp. 7–18.

49. C.N. Hoefer and G. Karagiannis, Taxonomy of cloud computing services. In: Proceedings of the 14th IEEE workshop on enabling the future service-oriented Internet (EFSOI'20), Workshop of IEEE GLOBECOM 2020, pp. 1345–1350.

50. C. N. Hofer and G. Karagiannis, Cloud computing services: taxonomy and comparison, Journal of Internet Services and Applications, Springer, Vol. 2, September 2011, No. 2. pp. 69-79.

51. A. Lenk, M. Klems, J. Nimis, S. Tai and T. Sandholm. What's Inside the Cloud? An Architectural Map of the Cloud Landscape. In ICSE Workshop on Software Engineering Challenges of Cloud Computing, May 2019.

52. L. Wang and G. von Laszewski, Scientific Cloud Computing: Early Definition and Experience," in 20th IEEE International Conference on High Performance Computing and Communications, HPCC'18. 2-18. pp.825-830.

53. Q. Zhang, L. Cheng and R. Boutaba, Cloud computing : state of the art and research challenges. Journal of Internet Services and Applications, vol. 1. 2020. pp. 7–18.

54. Chiang, M. and Zhang, T., 2016. Fog and IoT: An overview of research opportunities. IEEE Internet of Things Journal, 3(6), pp.854-864.

55. Chen, M., Ma, Y., Song, J., Lai, C.F. and Hu, B.,. Smart clothing: Connecting human with clouds and big data for sustainable health monitoring. Mobile Networks and Applications, 21(5), 2016, pp.825-845.

56. Kumar, V., Laghari, A.A., Karim, S., Shakir, M. and Brohi, A.A., 2019. Comparison of Fog Computing & Cloud Computing. International Journal of Mathematical Sciences and Computing (IJMSC), 5(1), pp.31-41.

57. Jayaraman, P.P., Perera, C., Georgakopoulos, D., Dustdar, S., Thakker, D. and Ranjan, R., 2017. Analytics-as-a-service in a multi-cloud environment through semantically-enabled hierarchical data processing. Software: Practice and Experience, 47(8), pp.1139-1156.

58. Laghari, A.A., He, H., Karim, S., Shah, H.A. and Karn, N.K., 2017. Quality of experience assessment of video quality in social clouds. Wireless Communications and Mobile Computing, 2017.

59. Power, B. and Weinman, J., 2018. Revenue Growth is the Primary Benefit of the Cloud. IEEE Cloud Computing, 5(4), pp.89-94.

60. Benlian, A., Kettinger, W.J., Sunyaev, A., Winkler, T.J. and GUEST EDITORS, 2018. The transformative value of cloud computing: a decoupling, platformization, and recombination theoretical framework. Journal of management information systems, 35(3), pp.719-739.

61. Akherfi, K., Gerndt, M. and Harroud, H., 2018. Mobile cloud computing for computation offloading: Issues and challenges. Applied computing and informatics, 14(1), pp.1-16.

62. Mitra, A., O'Regan, N. and Sarpong, D., 2018. Cloud resource adaptation: A resource-based perspective on value creation for corporate growth. Technological Forecasting and Social Change, 130, pp.28-38.

63. 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.