Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи віртуалізації систем імітаційного моделювання у хмарному середовищі

(тема)

Виконав:

студент II курсу, групи СПм-22-3

Прівалов Б.В.
(прізвище, ініціали)

Спеціальність

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

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

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Волк М.О.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Привалову Богдану Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи віртуалізації систем імітаційного моделювання у хмарному середовищі

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

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

3. Вхідні дані до роботи
Моделі хмарних обчислень.
Постачальники хмарних послуг (AWS, GCP, Azure).
Методи оцінки продуктивності хмарних платформ.
Опис систем імітаційного моделювання

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз предметної області.
Моделі та методи віртуалізації систем імітаційного моделювання у хмарному середовищі.
Інтеграція методу в хмарне середовище та проведення експериментів.
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	01.04.24 – 04.04.24	
2	Розробка моделей	05.04.24 – 15.04.24	
3	Реалізація алгоритмів	16.04.24 – 28.04.24	
4	Розробка структури програмних засобів	29.04.24 – 15.05.24	
5	Розробка програмних модулів	16.05.24 – 25.05.24	
6	Оформлення матеріалів кваліфікаційної роботи	26.05.24 – 05.06.24	
7	Подання кваліфікаційної роботи керівникові та	07.06.24 – 12.06.24	
	попередній захист		
8	Подання кваліфікаційної роботи на	13.06.24 – 15.06.24	
	рецензування		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Волк М.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 56 с., 13 рис., 3 табл., 1 дод., 50 джерел.

РОЗПОДІЛЕНІ СИСТЕМИ, КОМП’ЮТЕРНІ РЕСУРСИ, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ХМАРНІ ОБЧИСЛЕННЯ, МАСШТАБУВАННЯ, ВІРТУАЛЬНІ МАШИНИ

У кваліфікаційній роботі розглядаються актуальні питання підвищення ефективності систем розподіленої імітації на платформі хмарних віртуальних середовищ. Однією з задач процесу моделювання складних систем є масштабування завдань, яке полягає у збільшенні кількості ресурсів при зростанні потоків завдань. В роботі проводиться дослідження моделей, які використовуються для комп'ютерного моделювання, зокрема послідовна та паралельна модель з різними методами синхронізації. Розглядається хмарна модель імітаційного середовища. Дана модель дозволяє гнучке керування ресурсами, динамічне призначення та звільнення віртуальних машин. Експерименти проводилися з моделювання 3D-об'єктів на різній кількості віртуальних машинах. Використання розподіленого середовища моделювання дозволило збільшити розміри симуляції у випадку використання множини обчислювальних вузлів. Зафіксоване зростання продуктивності із збільшенням кількості обчислювальних вузлів. Результати дослідження можуть бути використані при розробленні нових методів розподілу ресурсів та технологій розподілених обчислень в хмарних системах, моделей ефективного управління обчислювальними вузлами, системах розподіленого імітаційного моделювання.

ABSTRACT

Master’s thesis: 56 pages, 13 figures, 3 tables, 1 appendice, 50 sources.

DISTRIBUTED SYSTEMS, COMPUTER RESOURCES, SIMULATION MODELING, CLOUD COMPUTING, SCALING, VIRTUAL MACHINES

The qualification paper examines topical issues of improving the efficiency of distributed simulation systems on the platform of cloud virtual environments. One of the tasks of the process of modeling complex systems is the scaling of tasks, which consists in increasing the number of resources when the flow of tasks increases. The paper examines the models used for computer simulation, in particular the serial and parallel model with different synchronization methods. The cloud model of the simulation environment is considered. This model allows flexible management of resources, dynamic assignment and release of virtual machines. Experiments were conducted on the modeling of 3D objects on various virtual machines. The use of a distributed simulation environment made it possible to increase the size of the simulation in the case of using a set of computing nodes. Recorded performance growth with increasing number of computing nodes. The results of the research can be used in the development of new methods of resource allocation and technologies of distributed computing in cloud systems, models of efficient management of computing nodes, systems of distributed simulation modeling.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10
1.1 Шаблони проєктування програмного забезпечення для систем віртуальної симуляції 10
1.2 Проблеми паралельного та розподіленого моделювання 14
1.3 Аналіз даних в технологіях хмарних обчислень 16
1.4 Хмарні алгоритми аналізу даних 17
1.5 Хмарні обчислювальні середовища у контексті моделювання 20
2 МОДЕЛІ ТА МЕТОДИ ВІРТУАЛІЗАЦІЇ СИСТЕМ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ У ХМАРНОМУ СЕРЕДОВИЩІ 24
2.1 Використання обчислювальних хмарних рішень у віртуальному моделюванні 24
2.2 Архітектура системи хмарних обчислень 28
2.3 Аналіз сучасних систем моделювання для хмарних обчислень 30
3 ІНТЕГРАЦІЯ МЕТОДУ В ХМАРНЕ СЕРЕДОВИЩЕ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ 33
3.1 Аналіз та вибір фрейворку моделювання хмарних систем 33
3.2 Опис експериментальної частини дослідження 36
3.3 Результати тестів 37
ВИСНОВКИ 42
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 45
ДОДАТОК А 50
Графічний матеріал кваліфікаційної роботи 50

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІТ – інформаційні технології

ПЗ – програмне забезпечення

ПК – персональний комп'ютер

ЦП – центральний процесор

AI – artificial intelligence

API – Application Programming Interface

AWS – Amazon Web Services

DIS – Distributed Interactive Simulation

GCP – Google Cloud Platform

GDI – Graphic Design Interface

HPC – High Performance Computing

IaaS – Infrastructure as a Service

IoT – Internet of Things

HLA – High Level Architecture

HTTP – HyperText Transfer Protocol

FOM – Federation Object Model

MIPS – Million Instructions Per Second

MVC – Model-View-Controller

NM – Network

NPB – NAS parallel benchmark

PaaS – Platform as a service

PDU – Protocol Data Unit

RTI – Run-Time Infrastructure

SaaS – Software as a service

SM – Simulation

WEB – World Wide Web

ВСТУП

Інформаційні технології розвиваються дуже швидко, що призводить до зростання можливостей усього людства та задовольняє соціальні та економічні вимоги нашого життя. Одночасно з цим технологічна еволюція сучасних комп'ютерних ресурсів має окремі проблеми. Тактові частоти сучасних процесорів вже певний період часу мають обмеження у 4-5 ГГц. Це пов'язано з обмеженнями елементної бази і технологій виробництва процесорів. Тому сучасне прискорення досягається архітектурними рішеннями, збільшенням множини ядер процесорів і забезпеченням технологіями паралельного виконання програм та розподілом функціональних сервісів за віддаленими ресурсами [1].

Загалом, виділимо наступні типи моделювання. По-перше, це фізичне моделювання, що використовує фізичні реальні системи за відповідними сценаріями. Такі процеси часто звуть натурними експериментами. Другій від моделювання – віртуальне, яке реалізується взаємодією програмних об'єктів в комп’ютерному середовищі моделювання. Користувач може бути задіяний у цьому процесі для прийняття рішень, моторики та реалізації комунікативних (інтерфейсних) навичок. По-трете – це конструктивне моделювання. Реалізується у вигляді комп’ютерної програми, команди якої базуються на різних об’єктах моделювання. Вони моделюють реальні процеси в контексті якогось визначеного сценарію моделювання [2].

Найбільш відомим вважається віртуальне моделювання, яке має більш високу точність, відповідність реальному світу (реалістичність), ніж конструктивне та інші види моделювання. Але, у разі великих, складних та/або динамічних систем, може складатися ситуація, в якої обчислювальної продуктивності одного комп'ютера недостатньо. У такої ситуації, є можливість переходу до іншого типу моделювання. Це може призвести до іншої проблеми, коли багато об’єктів, що об’єднані в один, призводять до

втрати точності і необхідності перегляду характеристик. Віртуальне моделювання можна проводити на множені комп’ютерних ресурсів. Тоді обчислювальна продуктивність одного ресурсу не є обмежуючим фактором.

Метою роботи є підвищення ефективності методів віртуалізації розподілених систем імітаційного моделювання в хмарному середовище шляхом розробки архітектурних моделей фреймворків симуляторів.

Для реалізації мети робити необхідно вирішити наступні задачі:

- провести дослідження шаблонів проектування програмного забезпечення для систем віртуальної симуляції;
- провести аналіз сучасних хмарних сервісів та симуляторів хмарних систем;
- запропонувати архітектуру фрейворку моделювання хмарних систем;
- провести експерименти по оцінці ефективності запропонованих рішень.

Об'єктом досліджень є процес імітаційного моделювання в хмарних середовищах.

Предмет досліджень: моделі та методи віртуалізації розподілених систем імітаційного моделювання в хмарному середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Шаблони проєктування програмного забезпечення для систем віртуальної симуляції

Основа кожної програми, яка реалізує виконання віртуальної симуляції, використовує один з існуючих шаблонів розробки програмного забезпечення (англ. software design patterns) [3]. Шаблон програмування розбиває програмне забезпечення на функціональні програмні модулі. Кожен з наявних модулів може розроблятися одночасно різними програмістами або фірмами. Один з відомих у світі шаблонів – MVC (модель–представлення–контролер, англ. Model-view-controller) – поділяє прикладне програмне забезпечення на три модуля: модель даних, інтерфейс та контролер [4,5]. Кожен модуль, у свою чергу, також може складатися з окремих підпрограм або функцій. Наприклад, окремий інтерфейс може реалізовувати взаємодію з апаратною частиною, іншою програмою або користувачем. Під час циклу виконання програми задіяні різні інтерфейси пристроїв введення, а обчислена фізика процесів, логіка роботи, відтворена графіка відображаються на екрані монітору. При класичному підході такі складові циклу виконання відтворюються послідовно. Коли закінчує працювати певний модуль системи, починається робота іншого.

В якості прикладу, розглянимо звичайний цикл функціонування системи моделювання з використанням шаблону MVC. Будемо враховувати, що кожен елемент з шаблону програмування в свою чергу може складатися з декількох модулів (рисунок 1.1).

На рисунку також представлено факт, що показує різну кількість підмодулів (функцій, підпрограм, бібліотек тощо) в своєму складі. Так інтерфейс програмної системи містить модулі Graphic Design Interface (GDI), який виконує графічну візуалізацію на екрані монітора, модуль взаємодії з

іншими програмними компонентами за допомогою мережи Network (NM) і модуль взаємодії з зовнішніми пристроями External Units (EU). До останніх відноситься клавіатура, маніпулятор миша та інше. І також контролер, наприклад, може складатися з модуля моделювання – simulation (SM), штучного інтелекту artificial intelligence (AI) та якихось інших програмних компонент, наприклад, прикладних бібліотек з моделями реальних фізичних явищ physic (PH).

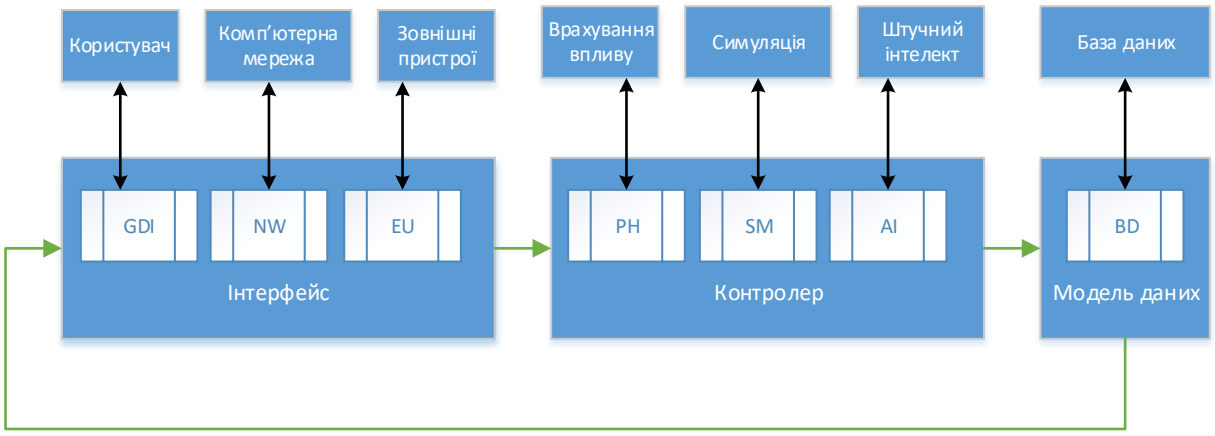

Рисунок 1.1 – Послідовне виконання процесу симуляції

Для виконання процесу моделювання з забезпеченням реалістичного зображення у користувачів, які спостерігають за процесом моделювання, формування кадрів екрана повинно відбуватися з певною частотою. Для отримання 50-60 кадрів в секунду, обчислення одного циклу симуляції повинні виконуватися приблизно за 17 мілісекунд. Вважається, що нижня межа для нормального сприйняття людиною повинна бути не меньш ніж 16 кадрів за хвилину [4]. При таких умовах реалістичність зображення буде стабільним. Таблиця 1.1 допомагає провести порівняльний аналіз залежності кількості обрахованих кадрів в залежності від кількості об'єктів моделювання та кількості циклів моделювання. Експеримент було проведено в рамках проекту моделювання системи візуалізації сцен 3D об'єктів, який було виконано на віртуальних машинах, з наступними параметрами: 4 ядра процесора Intel Core i7-9700K/3,6GHz/ з 12MB оперативної пам'яті.

Таблиця 1.1 – Порівняння кількісних параметрів для однієї віртуальної машини

Кількість циклів	Кількість об'єктів	Час формування кадру, сек.	Кількість кадрів в секунду
10	100	16	60
10	500	18	55
10	1000	22	45
15	100	24	42
15	500	28	35
15	1000	32	31
20	100	33	30
20	500	45	22
20	1000	57	17

Аналізуючи даних таблиці 1.1, можна зробити наступні висновки. Зростання кількості ітерацій моделювання та збільшення кількості об'єктів 3д сцени, що моделюється, спостерігається зростання часу обчислення кожного кадру зображення. Таким чином, кількість кадрів, що система може відобразити на екрані, зменшується. Такі ж самі висновки можна зробити відносно інших модулів системи. Збільшення кількості інтерфейсів взаємодії з іншими учасниками симуляції, розміру нейромереж в системах машинного навчання, тощо, призводить наприкінці до аналогічного результату.

Впровадження архітектури багатоядерних процесорів доводить, що послідовні системи моделювання становляться неефективними. Рішенням цього процесу може бути в асинхронному циклі виконання, коли окремим модулям не потрібно чекати завершення виконання інших. Це можна реалізувати, наприклад, коли замість очікування використовується останній результат, який зберігається на спільному ресурсі (рисунок 1.2). Але подібна модель може бути дуже складною для реалізації в реальній системі. І все

одно, деякі програми системи буде необхідно виконувати послідовно, бо це обумовлюється характером процесу, що моделюється.

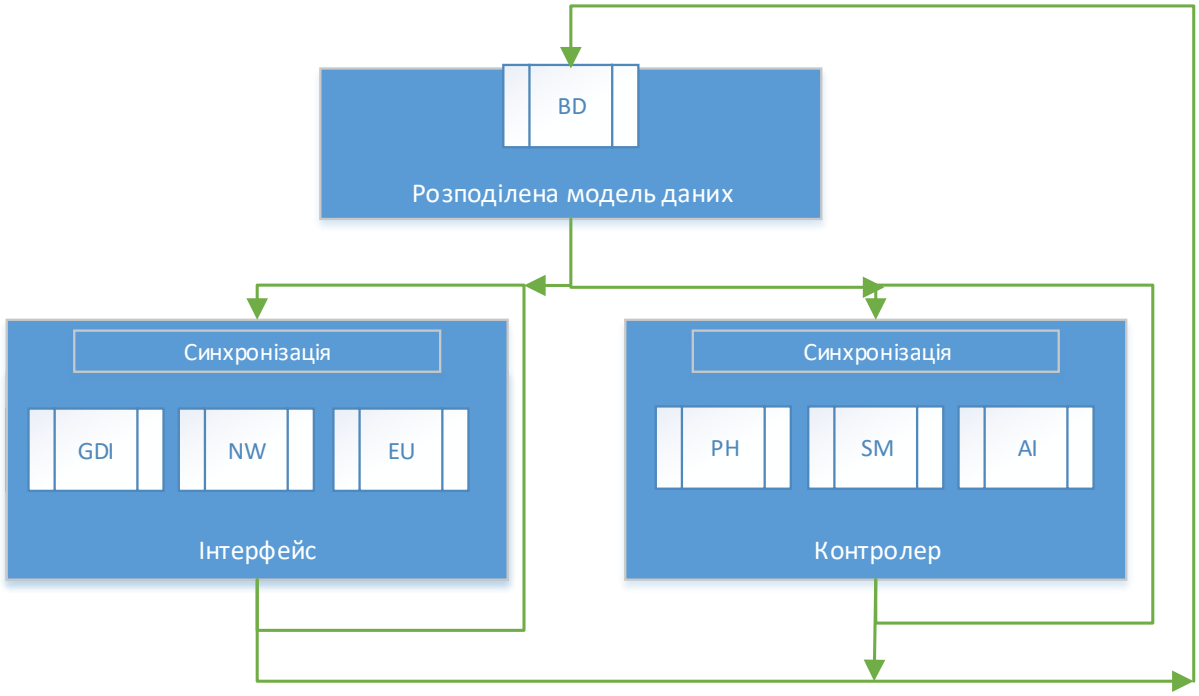

Рисунок 1.2 – Паралельне виконання процесу симуляції

Коли ми кажемо про процес комп’ютерного моделювання, використання однорангової мережів поєднанні з технгологією клієнт-сервер є найбільш поширеною практикою використання архітектури багатокористувацької програмної системи. Головна перевага однорангової мережі зв'язку полягає у відсутності центральних серверів. У цьому випадку вартість створення та підтримки не є високою. На жаль, такі рішення не масштабуються. Швидія з’єднань та обміну даними обмежує пропускну здатність, а впровадження синхронізації окремих модулів за допомогою сокетів є складною задачею. Повільність в обробки даних або підключенні одного з клієнтів може призвести до значних затримок в процесі розподіленого моделювання.

1.2 Проблеми паралельного та розподіленого моделювання

Під час паралельного та розподіленого моделювання найчастішими проблемами є синхронізація та підтримка фіксованого навантаження для всіх завдань. Ця задача пов'язана з питанням балансування навантаження в розподіленій системі[6]. Архітектура клієнт-сервер при централізованому управлінні, яка найчастіше використовується, вимагає одного серверу – комп'ютеру, який використовується для завдань сервера та розподіляє послуги доступу між клієнтами. На боці клієнтів використовуються клієнтські програми (наприклад, WEB-сайти), які взаємодіють із виділеним сервером. Програмні сервіси постійно обмінюються повідомленнями, пакетами даних та отримують конфігурацію змін, які треба зробити в локальному віртуальному середовищі моделювання [7].

Сервер виступає як посередник у процесах обміну повідомленнями між учасниками обчислювального процесу і можуть брати активну участь в розподілених процесах симуляції. Коли сервер приймає відповідне рішення щодо поточних подій, які відбуваються, можливі деякі затримки, що спричиняються мережею. Це може ускладнити процес моделювання та позбавити його реалістичності. Таким чином, в результатах моделювання можуть виникати ситуації нестабільності та збоїв частини об'єктів на сервері і клієнтських пристроях.

Одним з рішень цієї та подібних проблем є вдосконалення механізмів прогнозування, які оцінюють поточні та майбутні властивості об'єктів на базі отриманих даних (наприклад, швидкість, подібність, об'єм, напрямок). Традиційна архітектура мережних програмних систем, у тому числі і симуляторів, передбачає залучення одного серверу в моделі клієнт-сервер. Більш складні випадки, коли одного сервера недостатньо для реалізації функцій обробки чи зберігання певної кількості даних та з'єднань. У такому випадку потрібно мати багато серверів, які можуть бути незалежними. Статус стану моделювання зберігається на кожному сервері окремо або у

загальній базі даних. У першому випадку користувач обирає сервер для входу, у другому випадку – є комп'ютери або віртуальні машини, які не пручаються переміщенню на сервера та перенаправляють завдання користувача до найменш завантаженого ресурсу [7]. В архітектурі симуляторів найбільш часто використовуються два типи: розподілене інтерактивне моделювання (DIS) і архітектура високого рівня (HLA).

DIS виступає таким стандартом для обміну інформацією між розподіленими симуляторами, який засновано на децентралізованому керуванні, тому він не має окремого центрального серверу. Симулятори (програми) приєднуються до процесу розподіленого моделювання та можуть залишити цей процес у будь-який момент у майбутньому. Параметри моделювання зберігаються в повідомленнях серверу під назвою Protocol Data Unit (PDU). Симулятори обмінюється повідомленнями через один з доступних протоколів транспортного рівня (TCP або UDP), при цьому використовуючи багатоадресні або широкомовні методи та протоколи. Поточна, сьома версія DIS, визначає 72 різних типи PDU [8].

Архітектура високого рівня (HLA) — це спеціальна архітектура, яка була розроблена для гетерогенних комп’ютерних систем моделювання. Це рішення припускає об'єднання в одному обчислювальному пространстві зв'язок між незалежними від апаратними та програмними платформами. Одними з них можуть бути тренажери, інші – програми, які разом відповідно до HLA називається федератами. Вся система зв’язана засобами інфраструктури часу виконання (RTI) і створює так звану федерацію. Компоненти системи використовують усі дані про компоненти, які підтримують об’єктну модель об’єднання (FOM). Інформація взаємодії пересилається між симуляторами. Застосування описаних типів архітектури в багатьох симуляторів призводить до такого факту: віртуальне середовище відтворюється на кожному федераті. Призначення архітектури, подібної DIS і HLA – організація синхронізації різних симуляторів, що дозволяє розподілити віртуальне середовище між різними симуляторами.

1.3 Аналіз даних в технологіях хмарних обчислень

Видобуток даних[1-3] – це технологія, яка знаходить цінну інформацію з великих обсягів даних шляхом їх аналізу, у тому чмслі, залучая методи розподіленого та паралельного моделювання. База процесу інтелектуального аналізу даних на основі технології хмарних обчислень об'єднується з традиційним інтелектуальним аналізом даних, який складається з підготовки даних, інтелектуального аналізу даних, результатів оцінки – трьох етапівдослідження систем. З розвитком інформаційної ери, яка є наслідком «великих» даних, завдання інтелектуального аналізу даних створює нове через старі знання, акцентуючи увагу на великій базі даних ефективної та масштабованої технології інтелектуального аналізу даних.

Хмарні обчислення[4-5] розподіляють завдання на велику кількість ресурсів наявних комп’ютерів або інших ресурсівдля того, щоб усі програми мали доступ до обчислювальних потужностей, пристроїв зберігання та інформаційних послуг відповідно до потреб. У той же час, хмарні обчислення – це розвиток розподілених обчислень, грід-обчислень та паралельних обчислень [6].Хмарні обчислення зазвичай складаються з наступних 3 рівнів обслуговування: Saas, Iaas, Paas.

Моделі обслуговування показани на рисунку 1.3.

Існують певні вимоги до технології хмарних обчислень, об’єднують обчислювальні вузли і вузли зберігання даних. Планування завдань призначає та виконує завдання щодо збереження обладнання, відповідних блоків вхідних файлів, наскільки це можливо. Цей метод максимально використовує паралельні завдання для читання вхідних даних на локальній машині, ефективно зменшуючи мережевий потік даних[7].

Розподілені обчислення є одним із ефективних засобів вирішення завдань масового інтелектуального аналізу даних і вдосконалення масового інтелектуального аналізу даних[8].

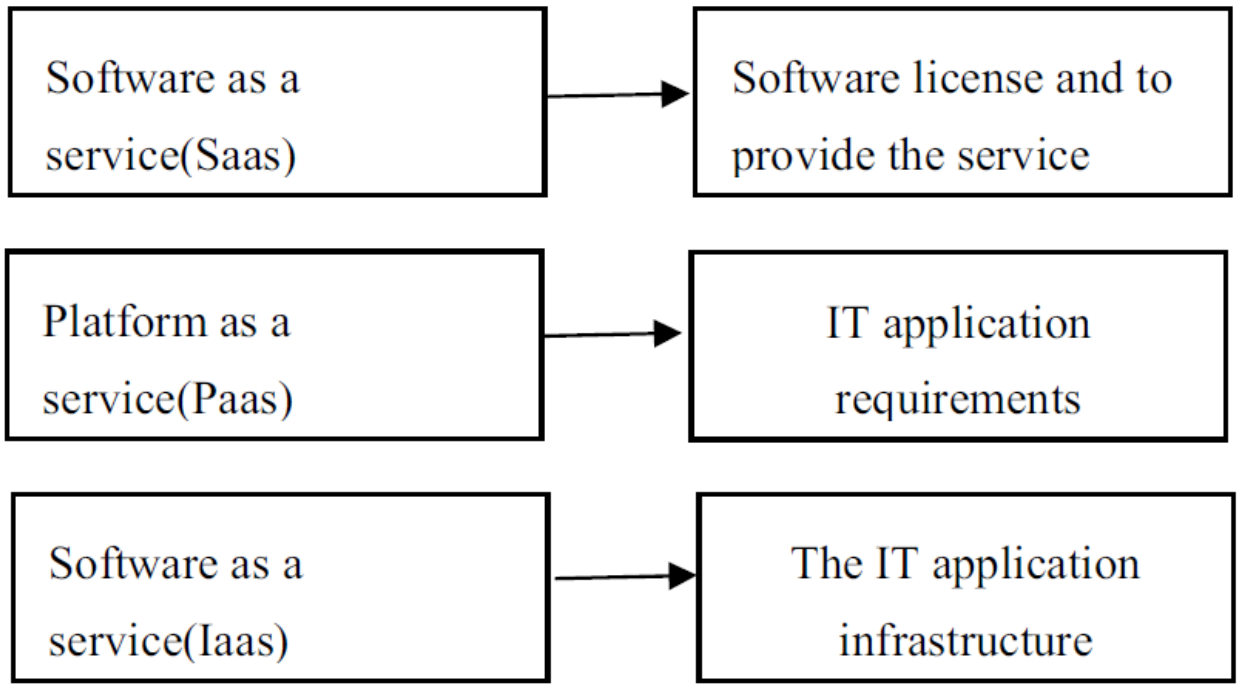

Рисунок 1.3 – Сервісна модель хмарних обчислень

Платформа хмарних обчислень забезпечує розподілене зберігання файлів і можливість паралельних обчислень, що є хорошим рішенням для розподіленої пам'яті, що використовується в розподілених обчисленнях і паралельних обчисленнях у двох рівнях вмісту[9].

Хорошою основою для побудови платформи інтелектуального аналізу даних хмарних обчислень є здатність ядра підтримки розподіленої файлової системи та розподілених паралельних обчислень[10]. Серед популярних розподілених файлових систем присутні файлова система Google (GFS), розподілена файлова система (HDFS), файлова система (KFS), яка може ефективно вирішити проблему зберігання великих даних.

1.4 Хмарні алгоритми аналізу даних

Алгоритм інтелектуального аналізу даних - це ядро системи керувіання розподеленої симуляції. Треба використовувати лише найефективніші алгоритми інтелектуального аналізу даних, щоб краще виконувати завдання

інтелектуального аналізу даних. Але через різноманітність алгоритмів інтелектуального аналізу даних існує також багато типів даних, вимоги до різних типів алгоритмів інтелектуального аналізу даних неоднакові. Найбільш часто використовуваний алгоритм інтелектуального аналізу даних має такі категорії.

Основна мета алгоритму класифікації базується на існуючих наборах даних для пошуку інших даних, аналізі існуючих наборів даних і відкритті нових даних, а потім пошуку принципу класифікації даних. Цей принцип можна використовувати для класифікації даних після додавання. Алгоритми класифікації підходять для реляційних даних, що складаються з кортежу.

Основною метою кластерного аналізу є пошук значущої моделі розподілу нових із потенційних даних. Процес полягає в тому, що існуючі дані не визначаються правилами групування заздалегідь, а поділяються на різні групи для аналізу відповідно до характеристик самих даних. Кластерний аналіз також використовується для реляційних даних, що складаються з кортежу.

Основна мета правил асоціації полягає в тому, щоб знайти цікаву асоціацію або кореляцію між наборами елементів у великих обсягах даних. Правил асоціації для типу даних відносно більше, в основному підходить для даних типу транзакції, типу транзакції та типу зв’язку. Правило асоціації, яке найкраще підходить для обробки типу змінної – це логічний і числовий тип.

Алгоритм паралельного видобутку даних є однією з ключових технологій, яка може ефективно використовувати основні можливості, надані платформою гучних обчислень. Загальний процес алгоритмів паралельного видобутку даних показано на рисунку 1.4.

Система інтелектуального аналізу даних зараз зазвичай заснована на хмарних обчисленнях, побудована на «хмарі», прозоро надаючи послуги інтерфейсу для різноманітних користувачів терміналів, забезпечуючи відкритий інтерфейс для програм, заснованих на розробці системи. Користувач може використовувати непрямі різні служби за допомогою

виклику відкритого інтерфейсу, наданого системою. Користувачеві не потрібно знати, як система має досягти, не потрібно турбуватися про обчислювальну потужність і ємність системи, потрібно лише вибрати відповідний алгоритм для обробки даних і, зрештою, способу виконання завдання для області розгортання системи для отримання результатів аналізу даних[11].

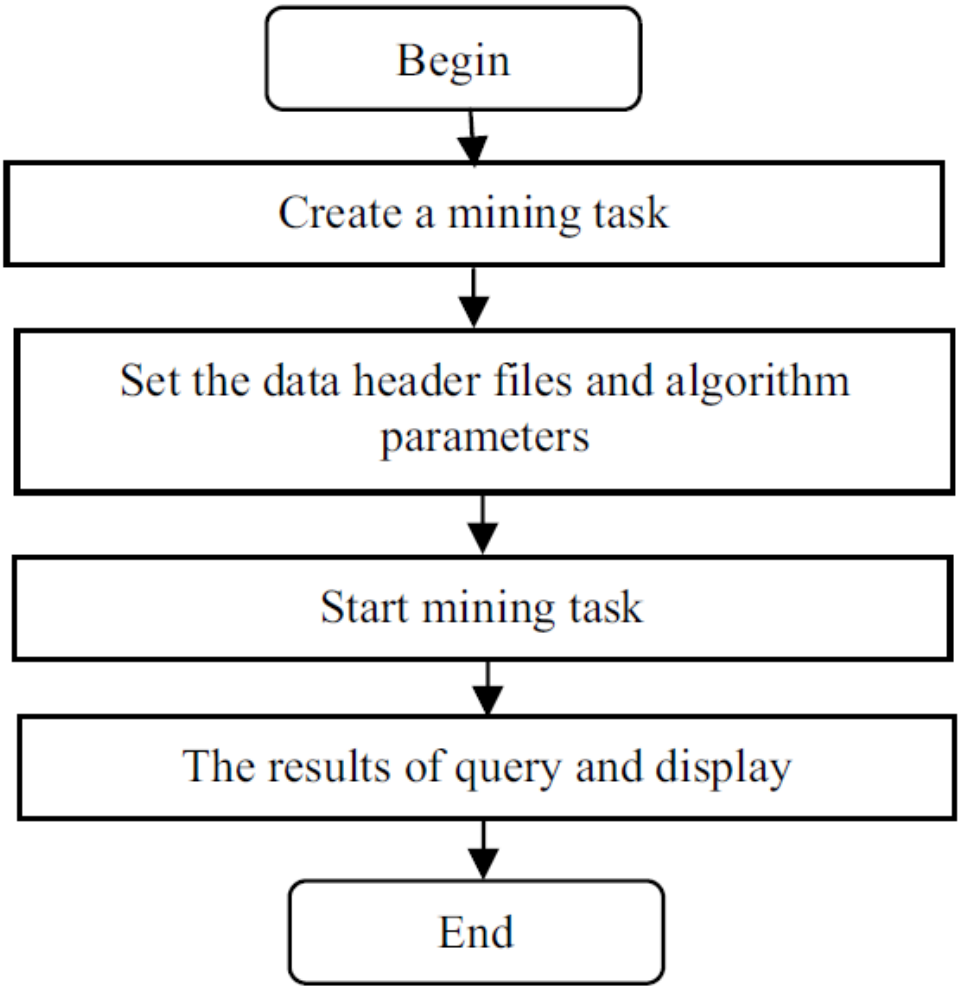

Рисунок 1.4 – Алгоритми паралельного аналізу даних з виконанням загальних етапів процесу розподіленого моделювання

Система інтелектуального аналізу даних, заснована на хмарних обчисленнях, може використовувати спосіб оплати за вимогою. Підприємства або окремі особи можуть отримати послугу безпосередньо через цю платформу, їм не потрібно купувати дороге програмне

забезпечення. Дані здебільшого зберігаються в хмарі зберігання після того, як настання хмарної ери, завдяки чому стали можливими дані на базі платформи хмарних обчислень, що базуються на інструментах видобутку.

1.5 Хмарні обчислювальні середовища у контексті моделювання

Швидкий розвиток у сфері інформації та обчислювальної техніки призвів до появи нових моделей хмарних обчислень. За матеріалами Національного Визначення Інституту стандартів і технологій (NIST), «Хмарні обчислення — це модель для повсюдного, зручного мережевого доступу на вимогу до спільного пулу конфігурованих обчислювальних ресурсів (наприклад, мереж, серверів, пристроїв зберігання даних, програм та служб), які можна швидко підготувати та використати з мінімальними зусиллями керування або взаємодії з постачальником послуг» [17]. Ця модель реалізована через використання моделей обслуговування та розгортання. Трійка базові моделі обслуговування включає надання інфраструктури як послуги (IaaS), платформа як послуги (PaaS) і програмне забезпечення як послуги (SaaS).

Моделі розгортання мають справу зі способом в якій інфраструктура системи Cloud Computing розгортається, володіє та керується постачальниками послуг та споживачами. Це призводить до концепції приватної, публічної та гібридної хмарної архітектури [22].

Типова сервісна архітектура системи хмарних обчислень зображено на рисунку 1.5. Видно, що система хмарних обчислень має багаторівневу архітектуру з апаратним рівнем у центрі обробки даних, що формує основу загальної системи. Фактично вона складається з фізичних апаратних пристроїв включаючи процесор, пам'ять, сховище, канали передачі даних. Використовуючи фізичне обладнання, рівень інфраструктури, включає в себе технології віртуалізації для надання інфраструктури як а Сервіс (IaaS) (наприклад, Amazon Web Services (AWS) [3] і Rackspace [4]), який зазвичай

складається з пулу віртуальних машин (ВМ), які можуть бути надані IT-службі на вимогу споживачів. Оскільки IT-ресурси фізичного рівня є обмежені, віртуалізація є дуже важливою технологією, яка дає IT-споживачам уявлення про необмежені обчислення ресурси в їхньому розпорядженні.

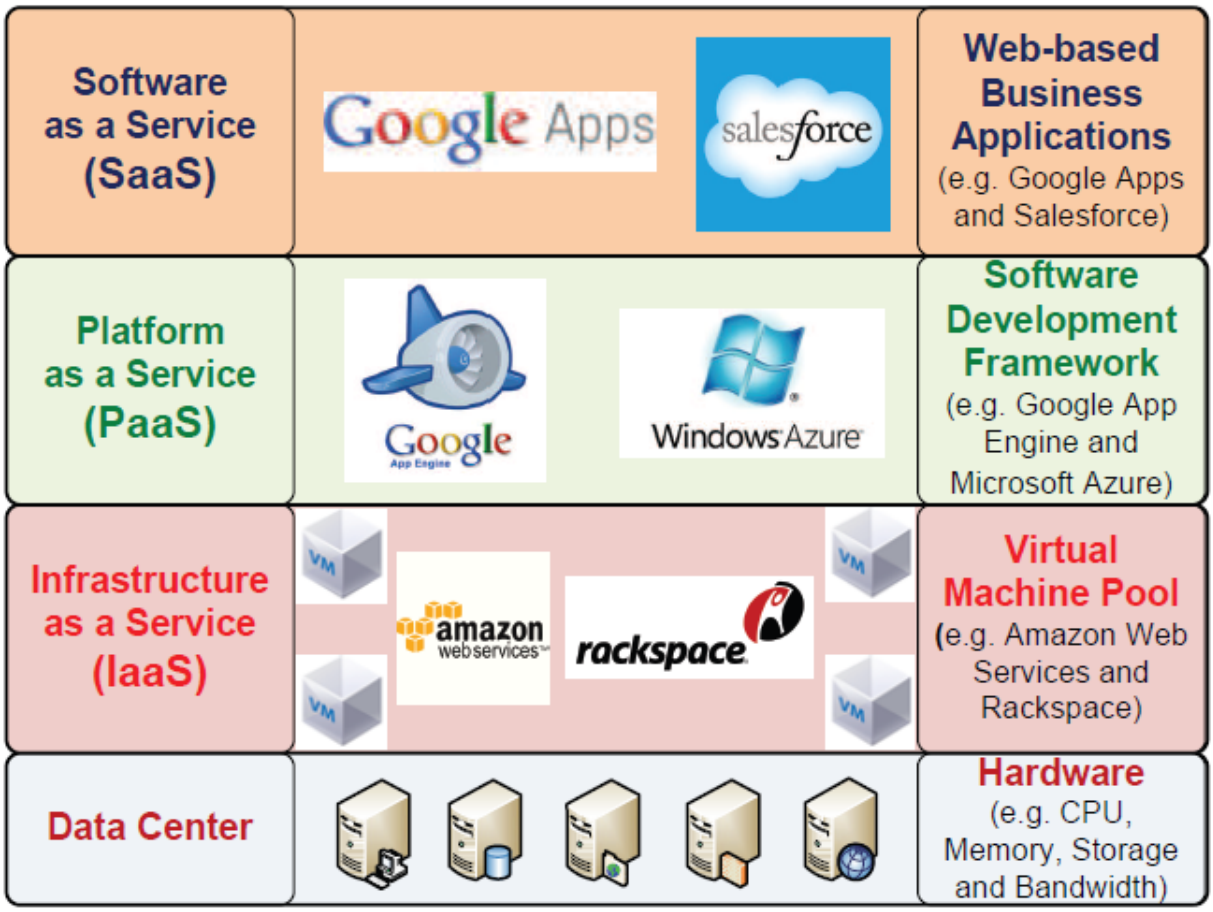

Рисунок 1.5 – Архітектура служб хмарних обчислень

Далі йде рівень платформи, який забезпечує платформу для створення і розробки програмного забезпечення, яке може бути пізніше доставлено через Інтернет. Отже, цей шар полегшує і надає платформу як послугу (PaaS) (наприклад, Google App Engine [5] і Windows Azure [6]), використовуючи компоненти та сервіси рівня інфраструктури. Він включає послуги для спільної розробки, тестування, розгортання, розміщувати та підтримувати програмне забезпечення та додатки в спільному інтегроване середовище

розробки.

Нарешті, найвищий рівень програмного забезпечення, який забезпечує шлях для надання готового до використання програмного забезпечення та додатків для бізнес-потреб споживачів хмарних IT. Отже, цей рівень полегшує та надає програмне забезпечення як послугу (SaaS) (наприклад, Google Apps [7] і Salesforce [8]), використовуючи компоненти та служби рівня платформи. SaaS, что є по суті послугою, яка надає засоби доступу до програмного забезпечення або додаток через Інтернет. Деякі з переваг такої послуги полягають у тому, що програмне забезпечення управляється з центральним розташуванням, користувачам не потрібно виконувати програмне забезпечення можна інтегрувати оновлення та різні частини програмного забезпечення Інтерфейси прикладного програмування (API).

Переваги хмарних обчислень для підприємств є зниження капітальних і експлуатаційних витрат IT та обчислювального обладнання. Фізичне IT-обладнання не потрібне придбати за високу ціну, натомість їх можна придбати як послугу на основі оплати за використання. Оскільки програмне забезпечення придбається у постачальника хмарних послуг як послуга, клієнтам не потрібно турбуватися про оновлення програмного забезпечення.

Сховище та пам'ять більше не обмежені ресурси у моделі хмарних обчислень, як зараз клієнти мають поняття необмежених ресурсів на вимогу. Дані клієнтів тепер більш надійно зберігаються в хмарі оскільки немає страху втратити дані у випадку збою апаратного забезпечення в приміщеннях клієнтів. Оскільки дані клієнта зберігаються централізовано, це дозволяє спростити групову співпрацю між працівники організації. Також можна отримати доступ до даних з різних незалежних пристроїв, оскільки дані не прив’язані до певного пристрою.

Різноманітні переваги, згадані вище, мають бути належним чином перевірені, перш ніж їх зможуть використовувати клієнти. У той же час постачальники послуг повинні тестувати різні моделі обслуговування, щоб вони могли досягти максимальний прибуток. Крім того, хмарні програми

мають різний склад, конфігурацію та розгортання вимог. Отже, кількісна оцінка продуктивності цих хмарних додатків з різними вимогами є складним завданням. Ця проблема особливо загострюється, коли Cloud дата-центр знаходиться в робочому стані. Це може бути ризиковано через точки зору як постачальників послуг, так і клієнтів, оскільки будь-яке погіршення продуктивності призведе до клієнтів віддалення від постачальників послуг. Рішення цього проблема полягає у використанні інструментів моделювання, які можуть оцінити продуктивність хмарних програм перед розгортанням та справжньою установкою.

Перевага цього підходу полягає в тому, що служби можуть тестуватися в повторюваному та контрольованому середовищі, вільному від реальної вартість. Крім того, це дає можливість налаштувати продуктивність вузьких місць перед розгортанням на реальних хмарах [9]. Однак, завдяки наявності численних засобів моделювання для симуляції середовищ хмарних обчислень дуже важливо провести аналіз для вибору та необхідно провести критичну оцінку таких інструментів для правильного моделювання та оцінки.

2 МОДЕЛІ ТА МЕТОДИ ВІРТУАЛІЗАЦІЇ СИСТЕМ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ У ХМАРНОМУ СЕРЕДОВИЩІ

2.1 Використання обчислювальних хмарних рішень у віртуальному моделюванні

Використання хмарних рішень в обчислювальних процесах віртуального моделювання (рисунок 2.1), запропоноване в роботі, передбачає сумісне використання тонких клієнтів та засобів віртуалізації. Паралельно з цім наразі спостерігається і використання товстих клієнтів з сучасними швидкими процесорами та новими відеокартами. Використання тонких клієнтів можливе за умов швидкої обробки програмних завдань клієнта окремим сервером. Тоді середовище моделювання наразі може бути названим хмарним середовищем моделюванням. Прикладом можна вважати задачі комп'ютерної графіки, які потребують багато масових обчислень.

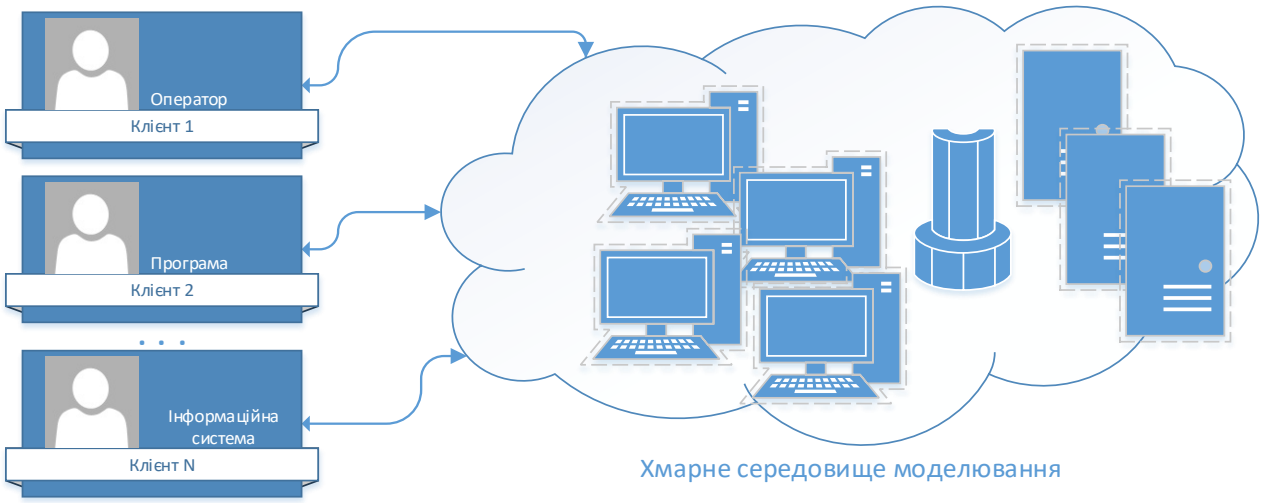

Рисунок 2.1 – Середовище імітаційного хмарного моделювання

Фірма NVIDIA випустила новий продукт з назвою NVIDIA GRID, який пропонує такі послуги. Запропоновані рішення дозволяють відправляти дані

на обробку зображень від віддаленого клієнта до виділеного серверу. Як результат, з боку клієнта програмне забезпечення отримує стабільний відеопотік, що декодується клієнтом та одразу виводиться на екран. В такому сенсі симуляція не залежать від платформи серверу і не диктує умови про наявне програмне забезпечення на боці клієнта.

Особливістю хмарної системи моделювання є реалізація асинхронного управління, що дозволяє гнучку політику масштабування системи в горизонтальному сенсі. Так можлива задавати додаткові віртуальні вузли у випадку потреб, що виникають. Важливим фактором роботи є врахування асинхронного режиму, який гарантує кожному вузлу відповідний інтерфейс, який об'єднує різні федерати та дозволяє взаємодіяти об’єктам, які віддалено керуються іншими віртуальними машинами.

Об'єкти (федерати) закріплюються за обчислювальними вузлами за територіальною, функціональною, платформною або іншими ознаками. В випадку знаходження об'єкту в зоні окремого вузла, цим об'єктом керує одна з віртуальних машин цього вузла. Масштаби розташування об'єктів для конкретного вузла можуть відрізнятися. Наприклад, вузли, які містять щільніше заповнення об’єктами симуляції, мають більшу продуктивність. Ініціалізація вузлів може проходить як статично, так і у динамічному режимі. При реалізації процесу моделювання та етапу аналізу результатів, можна організувати задовільний розподіл ресурсів за рахунок ефективного призначення вузлів. Розподіл можна повторювати багато разів для кожного завдання, кожної зміни состава ресурсів, стратегії управління та сценарію, що моделюється. У випадку динамічного призначення, постійний аналіз продуктивності та завантаження обчислювальних вузлів, а також можливі процедури ініціювання нових та звільнення вузлів, які не відповідають вимогам SLA або продуктивності процесу моделювання.

Подібний механізм реалізує масштабування обчислювальних ресурсів з урахуванням складності моделі системи [7]. Перевага динамічного розподілу полягає в гнучкій політиці управління та значної економії обчислювальних

ресурсів у випадку, коли їх використання є невід'ємною частиною процесу моделювання.

З іншого боку, виникає перенавантаження при тестуванні і аналізі поточного та прогнозованого стану вузлів. Перевага динамічного розподілу віртуальних машин полягає в більшій стійкості до відмов. При несподіваній втрати зв'язку з вузлом, стартує нова віртуальна машині та процес моделювання продовжується. Використовуються як стандартні обчислювальні блоки, так і спеціальні та спеціалізовані блоки. Моделі симуляторів підтримують фізику навколишнього світу, логіку функціонування об'єктів та штучний інтелект. Виділені віртуальні машини можуть використовувати графічні процесори (GPU), спецпроцесори або інші апаратні компоненти, які використовують віддалені мобільні пристрої для організації хмари [9].

Існують готові комерційні платформи, наприклад SpatialOS від Improbable (рисунок 2.2), які засновані на розподіленій обробці в системах хмарних обчислень для віртуального моделювання.

Такі рішення базуються на комерційній платформі Google Cloud. Моделювання в SpatialOS керується віртуальними вузлами, названими Workers. При зростанні потреб в обчислювальних потужностях моделювання, підключаються нові віртуальні машини на обчислювальних вузлах.

Є два типу робочих: керовані та зовнішні. Життєвим циклом керованих робочих керує SpatialOS. Зовнішні працівники виступають клієнтами які самі визначають, коли підключається або відключається до системи SpatialOS. Платформа SpatialOS слідкує за усіма типами працівників. Події, які відбувається в одному з пр оцівників, надсилаються в SpatialOS, а вона, в свою чергу, інформує про подію усі інші вузли. Для клієнтів обробка подій включає тільки відображення 3D-моделей, текстур і анімації. Керовані працівники займаються внутрішніми обчисленнями на базі інформації, яка призначається їм системою керування платформи SpatialOS.

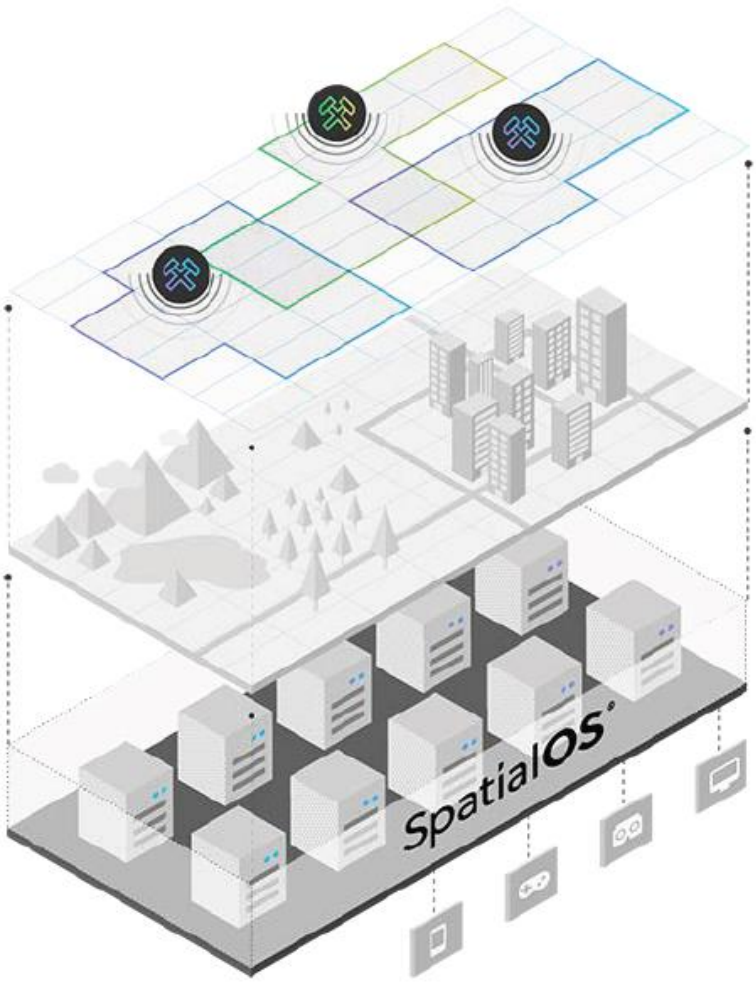

Рисунок 2.2 – Модель SpatialOS

Зовнішні працівники знають лише ту інформацію, яка потрібна об'єкту симуляції, обслуговуванням якого вони займаються. Також можливе налаштування способу взаємодії обчислювальних вузлів та віртуальних машин. У випадку статичного розподілу ресурсів встановлюються координати, атрибути та номер вузла. При динамічному розподілі визначення максимальної кількості вузлів та віртуальних машин, а також горизонтальне масштабування на основі зростаючої кількості керованих працівників [15].

Кількість працівників не обмежується, тому що реалізовані механізми горизонтального масштабування. Система може виділити необмежну кількість вузлів та віртуальних машин (обмеження можуть бути тільки за тарифним планом провайдера). Статичний розподіл більш корисно використовувати для відомих завдань, динамічний – для невідомих.

2.2 Архітектура системи хмарних обчислень

Хмара — це обчислювальна модель, заснована на Інтернеті, залучення дата центрів, чиї обчислювальні ресурси включають обчислювальну потужність, розширення ємності зберігання та віртуалізацію, а також для надають послуги користувачам (рисунок 2.3). З масивним збільшенням обсягу даних, диверсифікацією та персоналізованим видобутком даних з високим попитом, традиційні централізовані методи видо бутку даних не можуть адаптуватися. Хмарні обчислення стають ефективним способом вирішення проблеми масивності інтелектуальний аналіз даних через його величезну ємність для зберігання та обчислювальну здатність еластичних змін.

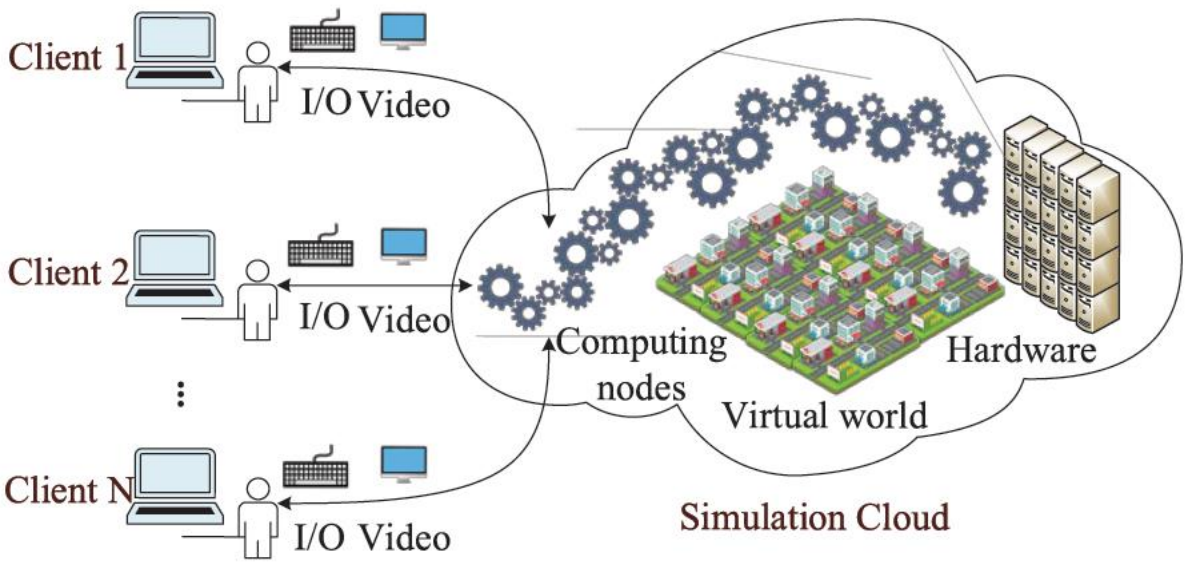

Рисунок 2.3 – Імітаційна хмарна модель

Фреймворк хмарних обчислень Hadoop — це широко використовувана архітектура розподіленої системи з відкритим кодом [12]. Користувачі можуть легко створити приватну хмарну платформу. Оскільки йому не потрібно розуміти розробку розподілених додатків, що лежать в основі справи, користувач може повністю використовувати можливості кластерних

обчислень і високошвидкісного зберігання.

Поточний аналіз і обробка даних хмарних обчислень широко використовують розподілену структуру розробки для роботи з подібними MapReduce. Вона може виконувати паралельно масивні завдання збору та аналізу даних у великій кількості комп'ютерів. Ця модель може значно абстрагувати складні операції у великомасштабних кластерних паралельних обчисленнях у процесі виконання двох функцій: Map та Reduce.[12].

На етапі Map структура Map/Reduce розбиває вхідні дані на велику кількість сегментів даних, і кожен фрагмент даних призначається для завдання Map. Кожному завданню з Map буде призначено ключ-значення для обчислення, щоб створити проміжний результат, а потім усі проміжні результати з однаковим значенням ключа значення передаються до функції збору результату після обчислення.

На етапі Reduce кожне завдання Reduce приймає два кортежі Key-Value як вхідні дані. Два кортежі викличуть Reduce функцію для злиття зі значенням і встановлення меншого значення, і кожен виклик функції Reduce має лише 0 або 1 вихідне значення. На кожному етапі виконання завдання підтримується відмовостійкість. Якщо на одному або декількох вузлах з'являється помилка в розрахунку процесу, буде автоматично перерозподіл завдань на інші вузли.

У роботі розроблено систему інтелектуального аналізу даних на основі технології хмарних обчислень, загальну структуру як показано на рисунку 2.4. Вузли в системі поділяються на дві категорії: MainCtrlNode і WorkNode. MainCtrlNode у системі складається з NameNode, сховища даних, JobTracker, SecondaryNameNode, бібліотеки алгоритмів інтелектуального аналізу даних. WorkerNode складається з Task-Tracker, DataNode, який відповідає за фактичне зберігання та обчислювальну роботу. NameNode керує метаданими файлової системи, яка є головним сервером розподіленої файлової системи та реалізовує відкриту, закриту, операцію, перейменування файлової системи.

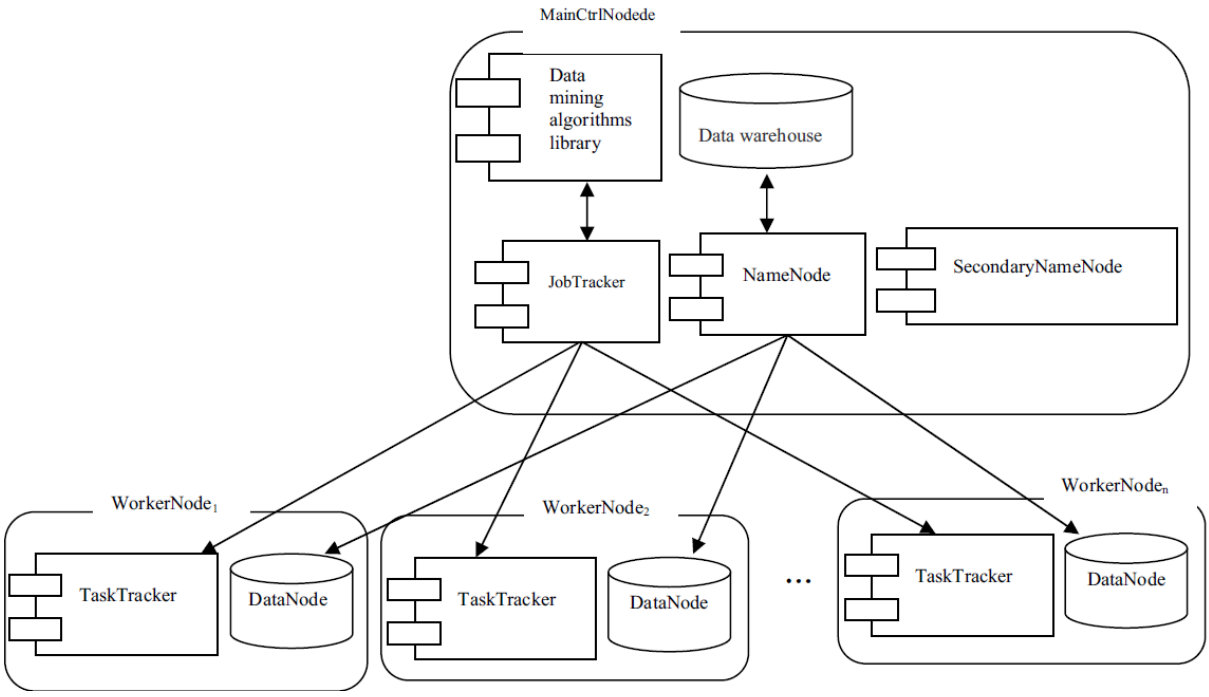

Рисунок 2.4 – Архітектура системи аналізу даних на основі технології хмарних обчислень

DataNode відповідає за обробку запитів клієнтів на читання та запис, для зберігання фактичних даних, відповідно до команди NameNode, виконує копіювання блоку даних, видалення, створити роботу. Ми застосовуємо інтелектуальний аналіз даних, який буде використовуватися в наборі даних для завантаження в сховище даних, NameNode автоматично блокуватиме файли та резервне зберігання даних для кожного DataNode. SecondaryNameNode допомагав NameNode обробляти файли зображень і журнал транзакцій.

2.3 Аналіз сучасних систем моделювання для хмарних обчислень

Щоб переконатися, що хмарна інфраструктура обчислень здатна надавати послуги відповідно до узгоджених стандартів та послуги надаються бажаної якості, важливо, щоб вони були перевірені раніше ніж розгортаються. Цього можна досягти за допомогою моделювання і інструментів моделювання, які можуть тестувати інфраструктуру, послуги та

застосування в різних умовах попиту. Існує чисельна кількість симуляторів та засобів моделювання, тому проведемо їх всебічний огляд і критичну оцінку. Таким чином, у цьому підрозділі представлено роботу, яка вже є існує в літературі в області огляду та огляду інструментів моделювання та імітації для тестування продуктивності систем хмарних обчислень.

Чжао та ін. вказують на те, що існує два види хмар обчислювальних симуляторів, а саме симуляторів, заснованих тільки на програмном забезпеченні та тренажери, засновані як на програмному, так і на апаратному забезпеченні [10]. Вони розглядають одинадцять тренажерів і згадують їх короткий опис разом з їх основними характеристиками. Нарешті вони забезпечили порівняння на основі критеріїв базової платформи, мови програмування та апаратно/програмного забезпечення.

Дуже схожа робота Oujani et al. забезпечує порівняння між вісьмома хмарними симуляторами (які однакові як у статті Чжао та ін.) [11]. Критерії їх порівняння також базується на трьох функціях, а саме базовій платформі, мови програмування та композиція апаратно/програмного забезпечення.

В іншому огляді Malhotra та ін. CloudSim симулятор і всі його варіанти (CloudAnalyst, GreenCloud, NetworkCloudSim, EMUSIM і MDCSim) розглядаються та порівнюються [12]. Вони порівнюють варіанти CloudSim на основі критеріїв платформи, мови програмування, мережевих функцій, типу симулятора (на основі подій/пакетів) і доступності (з відкритим або комерційним кодом).

Ще одна цікава стаття Сакелларі та ін. класифікує дослідження хмарних обчислень на три категорії, а саме: математичне моделювання хмарних систем, хмарне моделювання, програмне забезпечення та хмарні випробувальні стенди [13]. Вони наочно демонструють актуальність і застосування цих підходів та їх відносні переваги та недоліки. Для математичного моделювання вони оцінюють рішення на основі критеріїв функції QoS/продуктивності та енергоефективності. Хмарне програмне забезпечення для моделювання порівнюється на основі критеріїв функцій

енергоефективності, QoS, мови програмування (Java/C++) і доступності (з відкритим кодом/комерційним). Нарешті, Cloud Testbeds було класифіковано як комерційні (наприклад, Amazon EC2 і S3, Google Apps), наукові (наприклад, Open Cirrus, Open Cloud) і програмні інфраструктури (наприклад, OpenStack). Ці Cloud Testbeds оцінюються за послуги, які вони можуть підтримувати, а саме IaaS, PaaS і SaaS та їх доступності (з відкритим вихідним кодом/комерційним).

На основі цього аналізу видно, що дослідження обмежені оглядом підходів до моделювання та тестування продуктивності систем Cloud Computing. Слід зазначити, що весь огляд загалом надає інформацію про різні доступні підходи та їх особливості. Однак вони не дають чіткого пояснення щодо того, який підхід підходить для конкретної ситуації. Тому ми розглядаємо додатковий огляд існуючих підходів, новизна яких полягає в забезпеченні рейтинга цих підходів на основі їх популярності в дослідницькому співтоваристві.

3 ІНТЕГРАЦІЯ МЕТОДУ В ХМАРНЕ СЕРЕДОВИЩЕ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Аналіз та вибір фрейворку моделювання хмарних систем

З метою критичної оцінки та порівняння існуючих та запропонованого інструментів хмарного моделювання розробимо систему оцінювання. Цей каркас складається з набору критеріїв, які використовуватимуться для порівняння різних підходів та категорій класифікації. Внаслідок чого було вирішено наявність багатьох інструментів моделювання/моделювання скласти короткий список і скоротити набір таких інструментів. В аналізі представлено лише 10 таких інструментів. Це Eucalyptus, ns2, CloudSim, Opnet, GreenCloud, OpenStack, Open Cloud, Open Cirrus, CloudAnalyst та iCanCloud.

Розглядаючи критерії в існуючій літературі, слід зазначити, що найбільш частими є ліцензії, категорії моделювання, наявність відкритого коду та бібліотек підтримки, сервіси та інше. Першими були різні інструменти, яки шукали в Google Scholar і номер пошуку результатів був відзначені для кожного з них. Індекс популярності складався за шкалою від 0 до 10. Симулятор, який мав максимальну кількість, мав 10 балів (це в цьому випадку був Eucalyptus). Індекс популярності інших симуляторів розраховувався як ((Кількість пошуків/Максимальна кількість пошуків)*10).

Ми отримали такі результати.

Eucalyptus: це платформа програмного забезпечення з відкритим кодом для хмарних обчислень, які реалізують інфраструктуру як а Сервіс (IaaS) [14]. Евкаліпт дає користувачам можливість запускати та контролювати розгорнуті екземпляри віртуальних машин на різноманітних фізичних обчислювальних ресурсах. Це забезпечує сумісність з популярними веб-сервісами Amazon (AWS) API, включаючи Elastic Cloud Compute (EC2) і

Simple (S3).

ns2: це, перш за все, симулятор мережі з дискретними подіями і використовується в науково-педагогічній діяльності [15]. Це безкоштовне програмне забезпечення, загальнодоступне за ліцензією GNU GPLv2 для дослідження, розробки та використання. Незважаючи на те, що ns2 є дуже точним і популярним тренажером в області моделювання комп'ютерних мереж, було виявлено, що він не дуже дуже підходить для моделювання систем хмарних обчислень. Однак виявлено, що деякі дослідники використовували ns2 у своїй роботі при моделюванні хмарних обчислень, але вони мають лише обмежені функції на основі хмарного симулятора під назвою GreenCloud на платформі ns2.

CloudSim: це розширюваний набір інструментів моделювання, який дозволяє моделювати та імітувати середовища хмарних обчислень [9]. Набір інструментів CloudSim підтримує моделювання та створення однієї або кількох віртуальних машин на змодельованому вузлі центру обробки даних, робочі місця та їх відображення на відповідні віртуальні машини [16]. Це також дозволяє моделювати декілько центрів обробки даних для дослідження об’єднання та пов’язаних політик для міграції віртуальних машин для забезпечення надійності та автоматичного масштабування додатків [17].

Opnet: це симулятор комерційної мережі, який виконує керування продуктивністю програми в порядку забезпечення продуктивності програми для користувачів і задовольнення запитів бізнесу [18]. З точки зору перегляду систем хмарних обчислень, Opnet не має можливість впровадження та тестування інфраструктури як службі, однак вона має можливості для тестування додатків в хмарних системах [19].

GreenCloud: це середовище моделювання для енергетично залежних центрів обробки даних хмарних обчислень на основі платформи ns2. Тренажер створений для фіксації енергії, споживаної компонентами центру обробки даних (сервери, комутатори сховища даних), а також на рівні пакетів пересилки даних в реалістичних налаштуваннях і робочому навантаженні

розподілів ресурсів [20].

OpenStack: це хмарна операційна система, яка контролює великі пули обчислювальних ресурсів, сховищ і мережевих ресурсів у всьому центрі обробки даних, забезпечує інформаційну панель, що дає адміністраторам контроль, одночасно надаючи повноваження своїм користувачам надавати ресурси через веб-інтерфейс [21]. Система була розроблена як спільна програма на базі Linux проекту із трьома напрямками, а саме обчислення, служба зберігання та зображень. Це забезпечує сумісність з популярними комерційними хмарними провайдерами, такими як Amazon Веб-сервіси, Rackspace і HP Cloud.

Open Cloud: це хмарний тестовий стенд із понад 30 членами, зокрема Cisco, NASA та університети США Штатів і Японії, що реалізують проекти переважно в областях проміжного ПЗ хмарних обчислень Big Data [22]. Ця система забезпечує моделювання усіх трьох хмарних послуг, а саме IaaS, PaaS і SaaS.

Open Cirrus: Це дуже великий хмарний випробувальний стенд, що включає об'єднані гетерогенні розподілені центри обробки даних [23]. Open Cirrus — це спільна ініціатива, яку спонсорує Hewlett Packard, Intel і Yahoo у співпраці з іншими, більш ніж вісімью організаціями та університетами світу [24]. Вважається найбільшим випробувальним стендом та складається з десяти сайтів у Північній Америці, Європі та Азії і складається з кількох тисяч ядер і пов'язаних з ними центрами зберігання даних.

9) CloudAnalyst: це інструмент для імітації великого масштабу хмарних програм з метою вивчення поведінки таких програм під час різних конфігурацій розгортання [25]. CloudAnalyst допомагає розробникам уявлення про те, як географічно поширювати програми серед хмарних інфраструктур і додаткових послуг, наприклад оптимізацію продуктивності програм. Система базується на платформі CloudSim і надає графічний інтерфейс користувача для легшого вивчення хмарних програм.

iCanCloud: це платформа моделювання, яка є орієнтованою на

моделювання широкого діапазону систем хмарних обчислень та їх основних архітектур [26]. Вона має можливість моделювати великі середовища (тисячі вузлів) і розподілені програми із настроюваним на різних рівнях деталізації. Симулятор iCanCloud використовує популярну OMNET++ структуру, що використовується для моделювання комп'ютерних мереж.

3.2 Опис експериментальної частини дослідження

Експериментальна частина для нашого дослідження поєднує дані експериментів та дані, отримані при використанні стандартних методів. На початку експериментів, було визначено множину програм тестування для створення середовищі моделювання. Експерименти проводилися для вимірювання продуктивності обраних програмних завдань на різноманітних хмарних системах. Система управління використовувала горизонтальну масштабованість хмарних систем для досягнення найліпшої продуктивності при виконанні еталонних програм. Статистику для експериментів було отримано шляхом моделювання на базі пакету CloudSim, який є у вільному доступі. Було використано метод стратифікованої випадкової вибірки для аналізу вхідних параметрів. Для аналізу даних ми обрали статистичні методи перевірки гіпотез і довірчих інтервалів, для визначення коректності отриманих у ході експериментів результатів.

Для проведення симуляції обрано відбувалось на 4 віртуальних машин (конфігурація A10), які розгорнуто на платформі Azure фірми Microsoft з 32 ядрами. В таблиці 3.1 міститься задіяна конфігурація кожної з віртуальних машин у хмарному кластері.

Для експериментів було обрано операційну систему CentOS 6.5. Для оцінки масштабованості, швидкодії та продуктивності з урахуванням використання розробленого методу та алгоритмів, було обрано гібридні програмні системи з обміном даними.

Таблиця 3.1 – Апаратна конфігурація для віртуальної машини

Хмарна платформа	Тип	Ядра	Процесор	Об'єм пам'яті	Тип пам'яті	Віртуалізація	Мережа
Azure	A10	8	IntelXeon E5-2670@ 2.6 GHz	32 GB	DDR3	Hyper-V	10 gigabit ethernet

Робоче навантаження на хмарну систему моделювання гарантувало ефективний розподіл розроблених програм. При цьому обчислювалась вартість використання хмарних ресурсів. Повністю навантаження на систему було реалізовано та обслуговувалось з використанням робочих вузлів та віртуальних машин A10 VM.

При аналізі статистичних даних оцінки навантаження було використано наступні критерії ефективності: швидкість і MOPS. Аналізуючи MOPS, можна отримати результати тестів, аналогічних бенчмаркам. Окрім цього, ми проаналізували обсяги прискорення на базі співвідношення часу, який було витрачено у випадку послідовного виконання програм, та часу, отриманого після распаралелювання та виконання паралельної програми.

Оцінку результатів досліджували шляхом повторних експериментів, що надало можливість збільшити точність оцінки та врахувати зміни середовища моделювання при повторних експериментах.

3.3 Результати тестів

Метою тестів було виявлення різниці в потужностях платформи моделювання на основі 1, 2 або 4 вузлів. Симуляція проводилася з використанням платформі SpatialOS, яка була описана висче. Сценарій експериментів використовував статичний метод при розподілі вузлів та віртуальних машин. Були визначено 4 позиції обчислювальних вузлів за координатами: <-250,-250>, <-250,250>, <250,-250> та <250,250>. Об'єкти

розміщувалися випадковим чином з рівномірним законом розподілу, при цьому виконувалось правило відстані вузлів один від одного: вузли розміщувалися на відстані 500 одиниць від позиції <0,0> (рисунок 3.1).

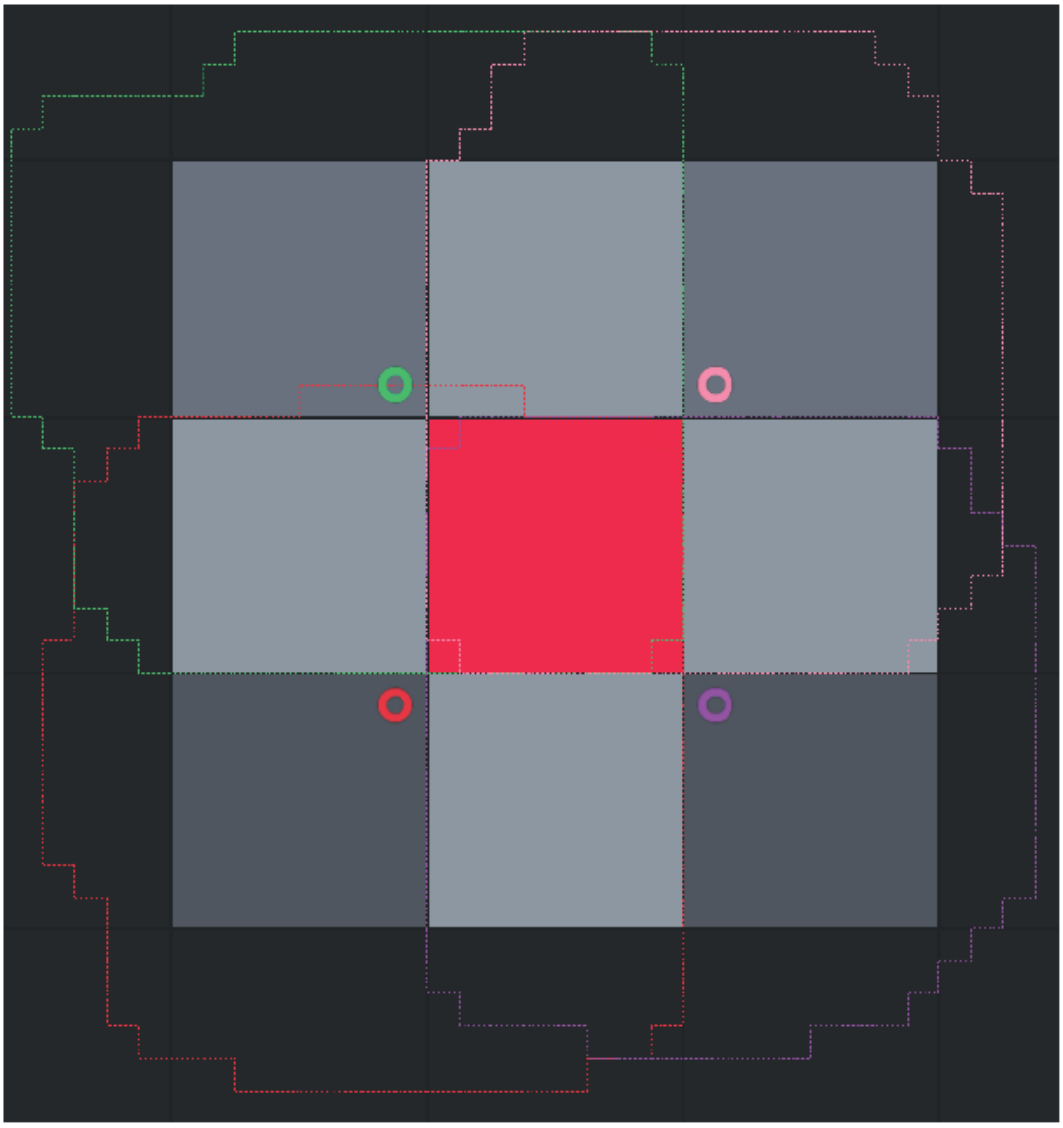

Рисунок 3.1 – Розміщення робочих вузлів і щільність об'єктів моделювання

Експерименти проводилися на конфігураціях: 1 – 4 вузлів та 300, 600, 900, 1200, 1500 об'єктів моделювання, з яких 50 виконували функції статичних предметів, а решта були динамічними об'єктами, які здійснювали рух у випадковому напрямку згідно рівномірного закону розподілу. Результати експериментів наведені у таблиці 3.2. Було помічено, що зміна параметрів ефективності системи від кількості об'єктів моделювання, кількості зв'язків та розміру пакетів даних, якими обмінюються федерати

моделі, пов'язано з продуктивністю системи нелілейними законами (згідно даних, отриманих експериментальним шляхом). Находження математичної моделі цього процесу може бути темою окремого дослідження.

Таблиця 3.2 – Порівняння кількості циклів моделювання, кількості об'єктів моделювання та кількості кадрів для різної кількості вузлів

Кількість об'єктів	Кількість робочих вузлів					
	1		2		4	
	Кількість кадрів в секунду	Кількість об'єктів на вузлу	Кількість кадрів в секунду	Кількість об'єктів на вузлу	Кількість кадрів в секунду	Кількість об'єктів на вузлу
300	45	300	55	150	60	75
600	31	600	52	300	58	150
900	15	900	34	450	49	225
1200	8	1200	17	600	25	300
1500	3	1500	9	750	13	375

Аналізуючи результати експериментів можна зробити висновки, що при збільшенні кількості кадрів, які система встигає обробити за одну секунду, не є лінійним, що спостерігається при збільшенні кількості обчислювальних вузлів. Таким чином знайдено поріг – на одному вузлу неможна обчислити більш ніж 760 об'єктів. На рисунку 3.2 показано діаграму, яка візуалізує дані, що представлені у таблиці 3.2. Застосування 4 вузлів з достатньою кількістю віртуальних машин дозволило реалізувати плавне моделювання 3D зображень, які містили більш ніж 1200 об'єктів на задовільному рівні, що є хорошим показником для середньостатистичних зображень, які виникають при візуалізації процесів моделювання.

Аналіз динаміки параметрів оцінювання показує, що найбільший приріст FPS відбувається при організації переходу з одного вузла на два вузли. Найменший приріст FPS помітний при переході з 2 вузлів на 4 вузли.

Таким чином, становиться очевидним факт, що при використанні більшої кількості вузлів та віртуальних машин, необхідно задіяти додаткові ресурси для синхронізації та обміну даними. Кількість та якість ресурсів визначає система керування розподіленим моделюванням.

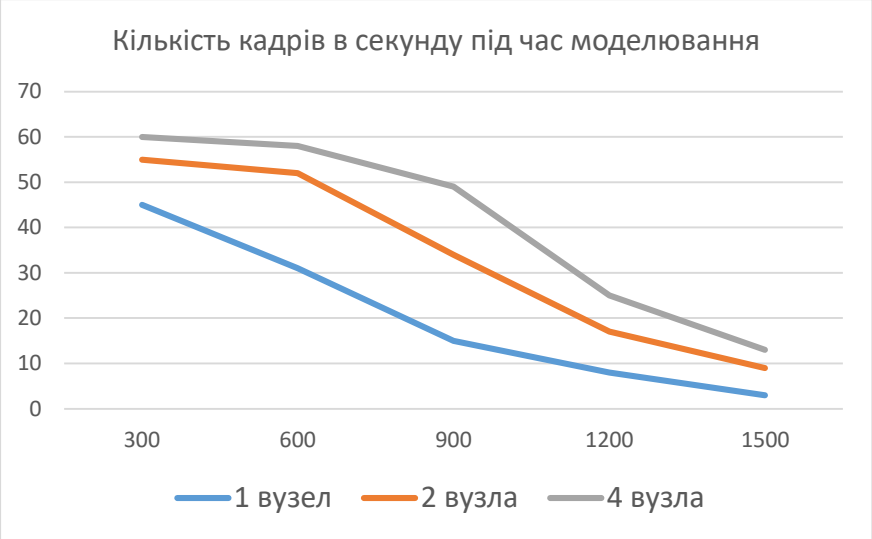

Рисунок 3.2 – Діаграма залежності кількості кадрів від кількості об'єктів для різної кількості обчислювальних вузлів

На рисунку 3.3 показано діаграму, що залежність візуалізує надмірність асоціації об’єктів, які задіяні в процесі моделювання та показує рівномірну підтримку параметрів середовища моделювання при масштабуванні.

Низьке збільшення потужності, ймовірно, є результатом високої надлишковості об’єктів, призначених вузлам, що показано на рисунку 3.3. У випадку 4 обчислювальних вузлів загальна сума всіх об’єктів у вузлах вдвічі перевищує кількість усіх об’єктів у моделюванні . Це викликано перекриттям зон відповідальності вузлів і високою щільністю розподілу об'єктів між усіма вузлами. У разі рівномірного розподілу об’єктів моделювання надмірність має бути меншою, а збільшення FPS – вищим.

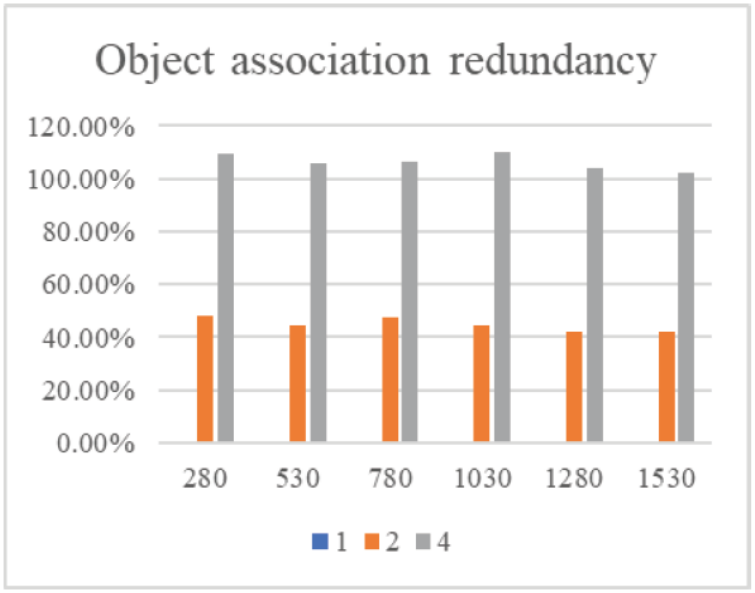

Рисунок 3.3 – Гістограма надмірності асоціації об’єктів

Повільне збільшення потужності кластеру відбувається в результаті високої надлишковості об’єктів, що моделюються на призначеній віртуальній машині, яка розташована на обчислюваному вузлу. У випадку, коли задіяно 4 обчислювальних вузли, сума всіх об’єктів моделювання в вузлах може вдвічі перевищувати кількість усіх об’єктів моделювання. Це є наслідком перекриття сегментів відповідальності віртуальних машин вузлів та високій щільності розподілених об'єктів за усіма вузлами (рисунок 3.4).

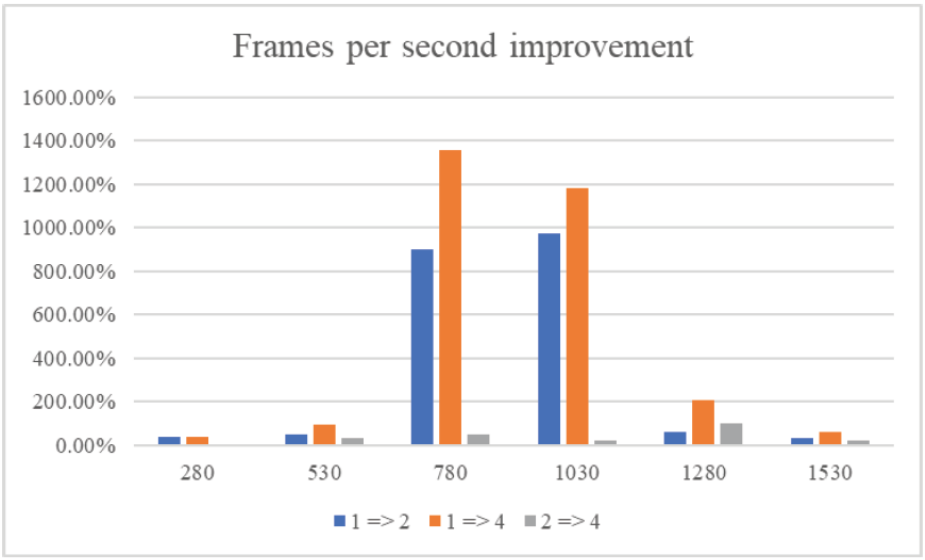

Рисунок 3.4 – Гістограма покращення обробки кадрів за секунду

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто концепцію створення системи розподіленого віртуального моделювання. Було показано та проаналізовано найпоширеніші проблеми, які виникають в процесі моделювання складних інформаційних систем. В роботі запропоновано рішення, яке використовує хмарні системи для проведення розподіленого моделювання сукупності складних об'єктів. При застосуванні великої кількості вузлів та віртуальних машин необхідно задіяти велику обчислювальну потужність не тільки для моделювання, а і для управління обчислювальним процесом, обміну повідомленнями між ними об'єктами моделювання. Тому досліджені залежності при масштабуванні з залученням додаткових хмарних вузлів, у загальному випадку, носить нелінійний характер [50].

При використанні розподілених платформ моделювання, в роботі призвело до подвійного розміру системи симуляції завдяки використанню чотирьох обчислювальних вузлів. При збільшенні кількості вузлів, зменшується швидкість збільшення продуктивності. При тестуванні деяких сценаріїв моделювання, коли ряд об'єктів не змінювали свого положення протягом деякого часу, можна було спостерігати зменшення продуктивності. Практичний досвід показав, що мінливе віртуальне середовище моделювання може призвести до значної різниці в навантаженні обчислювальних вузлів при різних сценаріях симуляції.

Подальшими дослідженнями можна розвинути сучасні методи та механізми призначення програмних завдань (об’єктів симуляції) за обчислювальними ресурсами, розробити більш ефективні моделі управління процесом розподіленого імітаційного моделювання, представлення завдань та модулів контуру управління хмарними системами.

В роботі розроблено архітектуру розподіленої системи моделювання з

використанням інтелектуального аналізу даних на основі технології хмарних обчислень. Масивна інформація про дані та потужні обчислювальні можливості й можливості обробки даних хмарних обчислень забезпечують потужну підтримку інтелектуального аналізу даних. Завдяки інтелектуального аналізу даних і технології хмарних обчислень в роботі пропонується архітектура платформи інтелектуального аналізу даних, заснована на хмарних обчисленнях.

Також в роботі обґрунтована необхідність створення основи для оцінки та порівняння інструментів моделювання та імітації для середовищ хмарних обчислень. З цією метою впроваджено структуру, яка складалася з шести критеріїв, а саме функцій API, вимог операційної системи, підтримуваних послуг, категорії інструменту, ліцензування та популярності у дослідницькому співтоваристві.

Проаналізовано десять найпопулярніших програм для моделювання, програмне забезпечення, фреймворки та тестові рішення. Була виконана оцінка і порівняння за шістьма критеріями. Одним із висновків був вибір інструмента моделювання обчислювального та мережевого обладнання та використано Eucalyptus як програмну основу для керування тестовою платформою приватної хмари.

Таким чином, в роботі підвищено ефективность методів віртуалізації розподілених систем імітаційного моделювання в хмарному середовищі шляхом розробки архітектурних моделей фреймворків симуляторів.

Для реалізації мети робити було вирішино наступні задачі:

- проведено дослідження існуючих шаблонів проектування програмних систем віртуальної симуляції;
- проведено аналіз хмарних сервісів сучасних провайдерів та програм симуляції хмарних систем;
- запропонувано архітектурне рішення для фрейворка імітаційного моделювання хмарних систем;
- проведено експерименти та оцінена ефективності запропонованих

рішень.

Крім того, за відсутності апаратного забезпечення є програмне забезпечення моделювання CloudSim – це найкращий вибір для моделювання різних оцінок ефективності, що можна отримати в процесі експериментів. В якості майбутньої роботи планується дослідити особливості CloudSim і Eucalyptus і порівняти їх детально шляхом імітації сценаріїв послуг Cloud Computing.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Farhanaaz, Sanju V. Compiling with MultiCores. 2nd International Conference for Innovation in Technology, INOCON 2023. P.1-8 DOI: 10.1109/INOCON57975.2023.10101054

2. Weatherly R. M., Wilson A. L., Canova B. S., Page E. H., Zabek A. A., Fisher M. C. Advanced Distributed Simulation though the Aggregate Level Simulation Protocol, *Proceedings of the 29thHawaii International Conference on Systems Sciences*.1996. P. 407-414. DOI: 10.1109/HICSS.1996.495488

3. Aratchige R., Manujaya K., Weerasinghe P. An Overview of Structural Design Patterns in Object-Oriented Software Engineering. Sofware Modeling. 2024. P.1-3. DOI: 10.13140/RG.2.2.16089.90724.

4. García R. MVC: Model–View–Controller. iOS Architecture Patterns. 2023. P.45-106. DOI: 10.1007/978-1-4842-9069-9_2.

5. Thakur R.N., Pandey U.S. The Role of Model-View Controller in Object Oriented Software Development. Nepal Journal of Multidisciplinary Research. No2. 2019. P.1-6. DOI: 10.3126/njmr.v2i2.26279.

6. Ivanisenko I.M., Volk M.O. Simulation methods for load balancing in distributed computing. Proceedings of IEEE East-West Design & Test Symposium (EWDTS’2017), Novi Sad, Serbia, September 27 – October 2, 2017. P. 690-695. DOI: 10.1109/EWDTS.2017.8110078

7. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and controlling system, 2016. Vol. 3/9 (81). pp. 45–53.

8. Peng Y., Dang W., Yin Q. Distributed simulation of MAS-based interactive applications with HLA. WIT Transactions on Information and Communication Technologies. No60. 2014. P.229-238. DOI: 10.2495/CTA140281.

9. Mamchych O., Volk M. Smartphone Based Computing Cloud and Energy Efficiency.12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2022, pp.1-5, DOI: 10.1109/DESSERT58054.2022.10018740

10. J Han,M Kamber. Data mining concepts andtechniques[M].Third Edition.San Francisco, CA,USA:Morgan Kaufmann Publishers,2012.

11. Shao feng-jing,Yu zhong-qing. Principle and algorithm of data mining[M].Beijing: Science Press,2009.

12. Shang Lin,Luo Bin. A data mining system based on Data Warehouse Framework[J]. Application Research of computers,2000,17(9):63-65.

13. Yang Yong,Dong zhen-jiang,Lu Ping. With the characteristics of cloud computing service delivery platform and its key technology research[J]. ZTE Communications,2011,17(5):55-57.

14. Wu zhu-hua. The analysis of the core technology of cloud computing[M].Bei Jing: People's Posts and Telecommunications Press, 2011.

15. Wang yi-jie,Sun wei-dong,Zhou Song. The key technology of distributed storage in cloud computing environment[J]. Journal of software,2012,23(4):962.

16. Wang Cong,Wang cui-rong,Wang xing-wei. The design of data center network architecture for Cloud Computing[J]. Research and development of computer,2012,49(2):286-293.

17. Hang He,Yi xiao-dong,Li shan-shan. Realization and evaluation of massive data processing platform for high performance computer[J]. Research and development of computer, 2012, 49:357-361.

18. M. A. Razzaq. Simulation and assessment of vertical scaling for a smart campus environment using the internet of things, IEEE Access 10 96322–96330.

19. H. Z. Fahmi, F. J. Lin, Nfv-enabled vertical scalability for iot slices, in: 22nd Asia- Pacific Network Operations and Management Symposium. IEEE access, vol. 10. pp. 96322-96330. 2022. doi: 10.1109/ACCESS.2022.3204042

20. M. Gusev. Scalable dew computing, Appl. Sci. Vol.12 (19) 9510. 2022.

P.1-10. https://doi.org/10.3390/app12199510

21. W. Brockelsby, R. Dutta. Traffic analysis in support of hybrid sdn campus architectures for enhanced cybersecurity. 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops. Paris, France. 2021. P. 41-48. DOI: 10.1109/ICIN51074.2021.9385530.

22. S. Tang. Coordinate-based efficient indexing mechanism for intelligent iot systems in heterogeneous edge computing, J. Parallel Distr. Comput. Vol.166.2022. P. 45–56. https://doi.org/10.1016/j.jpdc.2022.04.012

23. W. Han, S. Li, The method of allocating resources for ideological and political education in universities based on iot technology. Journal of Information & Knowledge ManagementVol. 21, No. Supp02, 2240011 (2022). https://doi.org/10.1142/S0219649222400111

24. W. Altwoyan, I. S. Alsukayti, A novel iot architecture for seamless iot integration into university systems, Int. J. Adv. Comput. Sci. Appl. 13 (4).

25. Y. Zheng, Q. Li, Y. Chen, X. Xie, W. Ma. Understanding mobility based on gps data. 10th International Conference on Ubiquitous Computing. 2008 P.312–321. https://doi.org/10.1145/1409635.1409677

26. Y. Zheng, X. Xie, W. Ma. Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull. Vol. 33(2). 2010. P. 32–39.

27. 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.

28. 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.

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

30. Foster, Y. Zhao, I. Raicu and S. Lu, “Cloud Computing and Grid Computing 360-Degree Compared,” Grid Computing Environments Workshop

(GCE '08), 2008.

31. 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.

32. 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.

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

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

35. Six benefits of cloud computing. Available at http://web2.sys-con.com/node/640237.

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

37. 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.

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

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

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

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

47. 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.

48. 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.

49. 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.

50. Волк М., Лабазов В.Г., Кипаренко Д.О., Привалов Б.В., Шумов Д.О., Самойлов І.А. "Розподілене комп'ютерне моделювання в системах хмарних обчислень". Вісник Херсонського національного технічного університету. №1. 2024. с. 211-217. **Фахове видання**