Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-25МЗ
ОПП «Комп’ютерні науки»
спеціальності F3 «Комп’ютерні науки»
_____ Галіченко Т.О.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнова Т.В.
«____» _____ 2026 р.

Нормоконтроль проекту
_____ Коваленко А.С.
«____» _____ 2026 р.

Рецензент _____
_____

Підписано КЕП
Центральноукраїнський національний технічний університет
Галіченко Тетяна Олександрівна
Сертифікат: 514B5C86A1E5DA110400000094B13D008C8B1006
Дійсний з 12.08.2026 20:46:11 по 12.08.2027 20:46:10

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _F_____*"Інформаційні технології"*_____
Спеціальність F3 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 1 » вересня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Галіченко Тетяні Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI*

2. Керівник роботи *Смірнова Тетяна Віталіївна, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 44-13 від 17.08.2026 року

3. Строк подання студентом роботи до захисту *2.12.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*5. Впровадження системи в промислову експлуатацію*
2. Перегляд аналогічних існуючих систем.	*6. Наукова новизна.*
3. Опис і обґрунтування проектних рішень.	*7. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*8. Висновки.*

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2026	16.11.2026

8. Дата видачі завдання « 1 » вересня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2026 р.	
3.	Розробка моделі компонента	20.10.2026 р.	
4.	Розробка структур даних	25.10.2026 р.	
5.	Розробка алгоритмів системи	30.10.2026 р.	
6.	Програмування алгоритмів	10.11.2026 р.	
7.	Розрахунки з охорони праці та техніки безпеки	15.11.2026 р.	
8.	Оформлення ПЗ	17.11.2026 р.	
9.	Попередній захист роботи	2.12.2026 р.	

Дата видачі завдання
« 1 » вересня 2026 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 1 » вересня 2026 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Галіченко Т.О. Дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI. ФЗ Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Метою розробки є дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Об'єктом дослідження є процес адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Предметом дослідження є методи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Методи дослідження базуються на методах кодування інформації, методах хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерні науки, відмовостійкість, розподілені обчислення, хмарна тестова платформа GENI

ABSTRACT

Halichenko T.O. Research and Software Implementation of a System for Adaptive Management and Fault Tolerance in Distributed Computing within the GENI Cloud Testbed. F3 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This final qualification thesis, submitted for the second (master's) level of higher education, presents the development of software designed for a system of adaptive management and fault tolerance in distributed computing within the GENI cloud testbed.

The objective of this development is to research and implement a system for adaptive management and fault tolerance in distributed computing within the GENI cloud testbed.

The object of the research is the process of adaptive management and ensuring fault tolerance for distributed computing within the GENI cloud testbed.

The subject of the research consists of methods for adaptive management and ensuring fault tolerance for distributed computing within the GENI cloud testbed.

The research methodology draws upon techniques in information encoding, cloud computing, mathematical statistics, and software development.

The outcome of the work is the software implementation of a system for adaptive management and fault tolerance in distributed computing within the GENI cloud testbed.

During the development of the software model, an analysis of existing hardware and software tools was conducted. All components of the developed software are described in detail.

A user-friendly interface has been developed, and instructions for operating the software tools are provided. The program can be used on PCs running the Windows 10/11 operating system.

The program was developed in a Python environment.

Keywords: computer science, fault tolerance, distributed computing, GENI cloud tested

ВІДОМІСТЬ ПРОЕКТУ

		Текстові документи			
1	А4	Пояснювальна записка	94	–	
2	А4	Робоча програма	24	–	Додаток А
		Графічні документи			
3	А4	Наукова новизна	1	–	
4	А4	Структурна схема	1	–	
5	А4	Функціональна схема роботи системи	1	–	
6	А4	Діаграма взаємодії процесів	1	–	
7	А4	Блок-схема основної програми	1	–	
8	А4	Блок-схема роботи підпрограми	1	–	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 9
ВСТУП..... 10
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 13
1.1 Призначення системи..... 13
1.2 Область застосування 13
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 15
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 15
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 21
2.3 Розгорнута постановка завдання 22
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 24
3.1 Опис функціонування системи 24
3.2 Розробка структурної схеми..... 26
3.3 Розробка функціональної схеми 34
3.4 Розробка діаграми процесів 37
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 39
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 39
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 68
6 НАУКОВА НОВИЗНА 74

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 75
7.1 Вступ 75
7.2 Аналіз умов праці 76
7.3 Техніка безпеки та протипожежна профілактика 79
7.4 Розрахункова частина 81
7.5 Висновки до розділу 84
8 ОСНОВНІ ВИСНОВКИ 85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM	–	електронно-обчислювальна машина
ATM	–	Asynchronous Transfer Mode – асинхронний спосіб передачі даних
BER	–	Bit Error Rate – імовірність ушкодження одного біта
CBWFQ	–	class based weighted fair queueing
DiffServ	–	Differentiated Services – механізм який залежно від вимог до якості обслуговування записує в IP заголовки пакетів спеціальні мітки
DSCP	–	DiffServ CoSde Point
ICMP	–	Internet CoSntrol Message Protocol – міжмережний протокол управляючих повідомлень
ISP	–	Internet Service Provider – провайдер
LLQ	–	low latency queueing
MPLS-TE	–	Multiprotocol Label Switching Traffic Engineering
PQ	–	priority queueing
RIO	–	RED with Input Output
RTT	–	Round Trip Time – час між відправкою запиту та отриманням відповіді
STM	–	Synchronous Transfer Mode – синхронний спосіб передачі даних
QoS	–	Quality of Service – якість обслуговування
WFQ	–	Weighted Fair Queuing – механізм планування пакетних потоків даних
WRED	–	Weighted random early detection – взвішане значення довжини черги, у якості фактора, визначаючого ймовірність відкидання пакета

ВСТУП

Актуальність теми. Для забезпечення надійних послуг у хмарі, відмовостійкість, мабуть, є найважливішим фактором. Вроджена чутливість до збоїв перешкоджає продуктивності та надійності хмарних сервісів. Як наслідок, відмовостійкість стає необхідною характеристикою для підтримки надійності, що важко забезпечити через динамічну архітектуру та складні взаємозалежності. Для вирішення питань надійності хмари в літературі було розроблено багато відмовостійких підходів. У даній магістерській роботі представлено нещодавнє дослідницьке дослідження, яке має на меті класифікувати різні архітектури відмовостійкості та стійкості до вторгнень. Крім того, вона надає ретельний критичний аналіз існуючої відмовостійкості та стійкості до вторгнень, а також комбіновані підходи, спрямовані на підвищення надійності, доступності та виконання хмарних сервісів. Магістерська робота також містить порівняння структури досліджуваних систем на основі різних важливих критеріїв, таких як вартість, тривалість виконання, надійність, безпека, використання ресурсів, споживання енергії та коефіцієнт відмов. Це дослідження має на меті всебічно розглянути цю тему, щоб дослідники могли отримати уявлення з існуючих моделей у літературі та надати глибший погляд на деякі складні проблеми та перспективи. Це сприятиме розробці високостійких відмовостійких та стійких до вторгнень алгоритмів планування для поточних та майбутніх хмарних застосувань**.**

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

– Дослідження системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

– Програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Об'єктом дослідження є процес адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Предметом дослідження є методи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Методи дослідження базуються на методах кодування інформації, методах хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

– Розроблено вітчизняний продукт адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVIII науково-технічній конференції здобувачів вищої освіти «Наука-виробництву», основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №17.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Хмарні обчислення стали одним із найпоширеніших методів надання обчислювальних послуг кінцевим споживачам. Надані ресурси можуть бути використані як утиліта, де споживачі платять за послуги, які вони використовують, що є ключовим компонентом цієї парадигми [1, 2, 3, 4, 5, 6]. Наприклад, встановлення великомасштабних віртуальних машин (ВМ) для запуску хмарних програм є економічно ефективним та високомасштабованим [7, 8].

Наукові обчислення є одним з основних застосувань, що підтримуються хмарними системами, де дані додатків та підзадачі можуть розподілятися та оброблятися спільно між кількома організаціями [9]. Для підтримки дедалі складніших обчислювальних процесів були введені наукові робочі процеси для автоматизації та ефективного управління такими додатками [10]. Системи управління науковими робочими процесами, спочатку розроблені в grid-середовищі, підтримують контроль, виконання та моніторинг цих робочих процесів [11]. З розвитком хмарних обчислень дослідники все частіше досліджують хмарні системи наукових робочих процесів завдяки їхній масштабованості та гнучкості [12].

1.2 Область застосування

Незважаючи на ці переваги, хмарні системи управління робочими процесами залишаються вразливими до збоїв через їхню складність, динамічну поведінку ресурсів та розподілену архітектуру [13, 14]. Такі збої можуть негативно вплинути на виконання завдань, завершення робочого процесу та загальну надійність системи [15]. Наприклад, перевантажені машини або збої,

пов'язані з мережею, пам'яттю або системними компонентами, можуть перервати виконання завдань та поширити затримки на залежні завдання [16]. Ці проблеми стимулювали розробку відмовостійких підходів до планування, які покращують стійкість робочого процесу. Наприклад, робота [17] розглядає проблеми георозподіленого хмарного планування робочих процесів за допомогою оптимізованого розміщення даних та відмовостійкого планування завдань, демонструючи покращення ефективності виконання та енергоспоживання. Окрім випадкових збоїв, хмарні наукові робочі процеси також стикаються із загрозами безпеці, що виникають внаслідок вторгнень та зловмисної діяльності, що ще більше мотивує потребу в інтегрованих відмовостійких та вторгненостійких механізмах планування.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Flexera One

Найкраще підходить для керування хмарою разом із ліцензіями на програмне забезпечення.

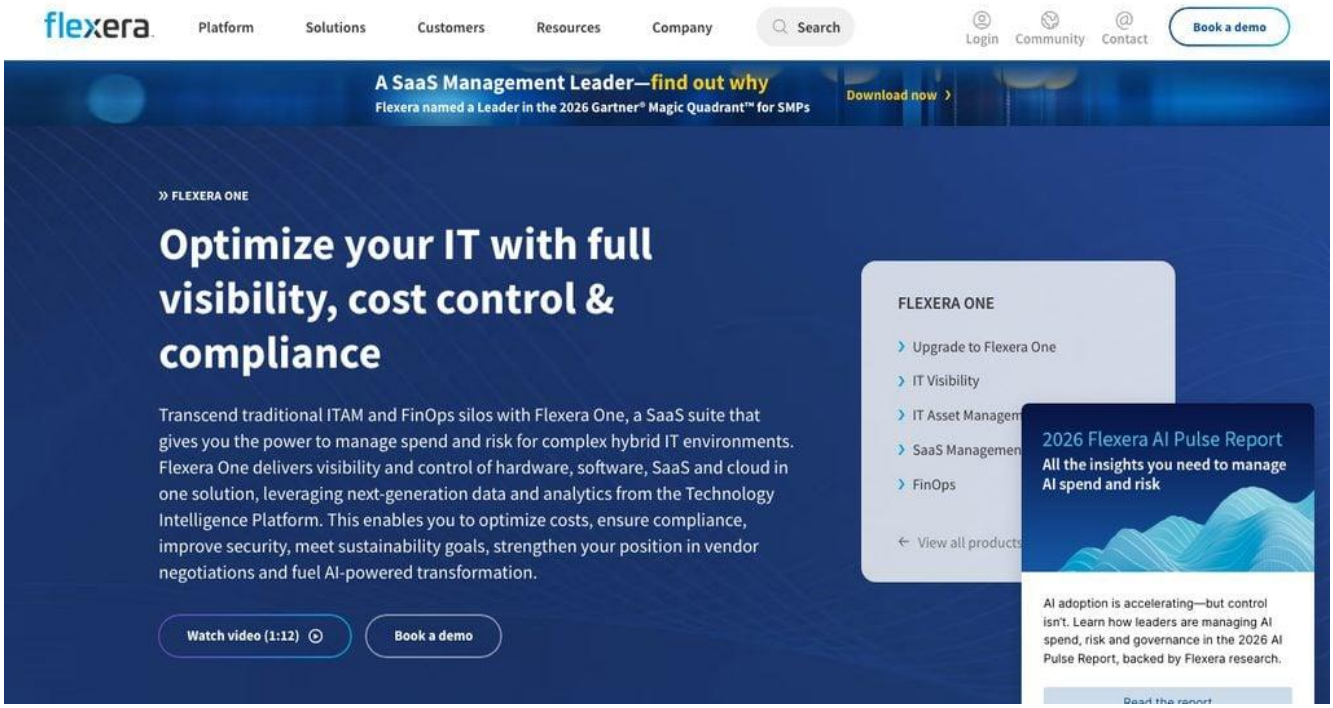


Рисунок 2.1 – Інтерфейс користувача Flexera One

Flexera One об'єднує FinOps, управління IT-активами та управління SaaS на одній платформі, що важливо, оскільки більшість підприємств витрачають гроші як на ліцензії, так і на екземпляри. Flexera також придбала Spot у NetApp, ще більше консолідувавши категорію.

Переваги:

– Хмара, SaaS та управління ліцензіями разом.

– Сильне управління.

Недоліки:

– Цикл продажів підприємства.

– Ціноутворювана ціна.

– Важкий для невеликого маєтку.

Ціна: Вказана; ліцензування для підприємств.

HPE Morpheus

Найкраще підходить для гібридних комплексів, що охоплюють хмарні та локальні системи.

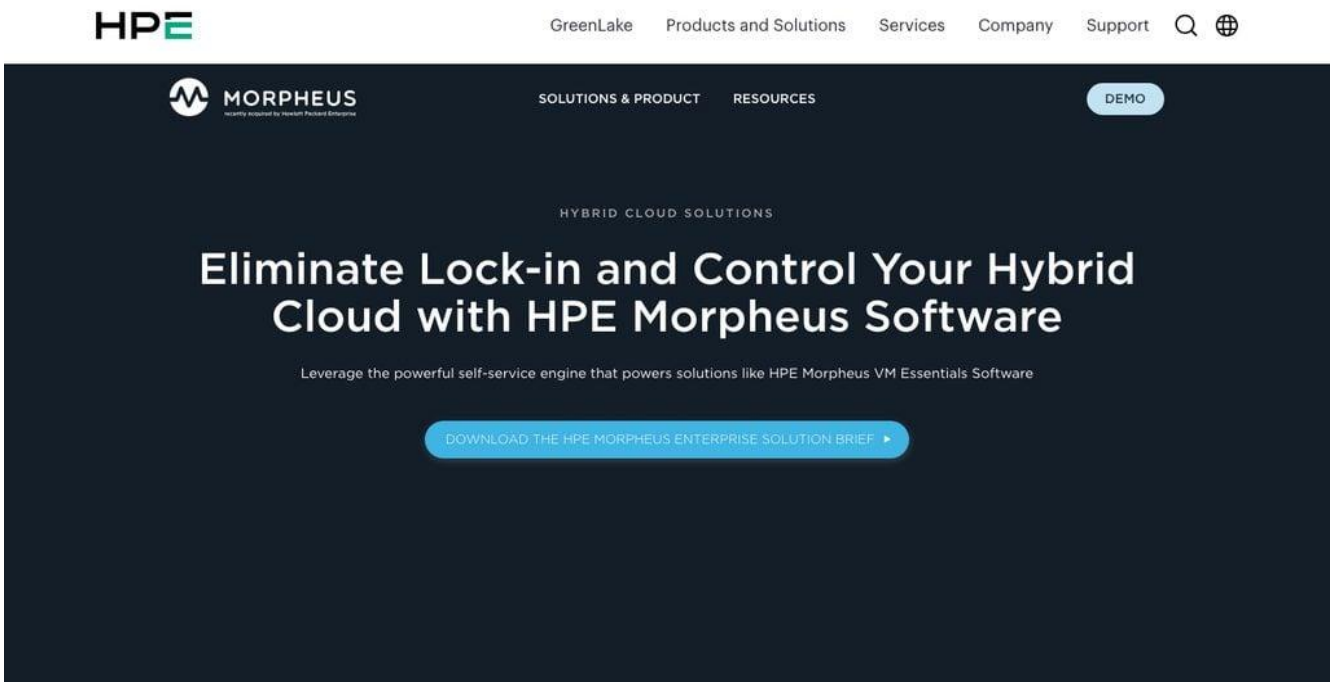


Рисунок 2.2 – Інтерфейс користувача HPE Morpheus

Morpheus Data, який тепер продається як HPE Morpheus, організовує надання ресурсів та управління публічними хмарами, VMware та серверними інфраструктурами з однієї площини керування. Це класичне визначення платформи керування хмарою.

Переваги:

– Справжня гібридна оркестрація.

– Самообслуговування.

– Сильний RBAC.

Недоліки:

– Тепер у портфоліо HPE.

– Котирування цін для підприємств.

CloudBolt

Найкращий варіант для спроби справжнього CMP без дзвінка від продавця.

CloudBolt займається наданням ресурсів, управлінням та оптимізацією витрат у гібридних хмарах та Kubernetes, і, що незвично для цієї категорії, пропонує безкоштовний рівень до ста ресурсів. Це робить його найлегшою CMP для чесної оцінки.

Переваги:

– Звільніть до 100 ресурсів.

– Гібрид та Kubernetes.

– Справжній приціл CMP.

Недоліки:

– Вказано платну ціну.

– Менші, ніж існуючі підприємства.

Ціни: Безкоштовно до 100 ресурсів; платно з урахуванням ціни.

Scalr

Найкраще підходить для управління тераформами у великих масштабах.

Scalr – це плавна заміна Terraform Cloud із забезпеченням дотримання політик, оцінкою витрат та покращеною поведінкою GitOps. Він керує інфраструктурою як кодом, а не через хмарну консоль, як це насправді працює в багатьох сучасних житлових комплексах.

Переваги:

– Дешевше, ніж Terraform Cloud.

– потужний політичний механізм.

– чистий GitOps.

Недоліки:

– Актуально лише якщо ви використовуєте Terraform.

– не широкий CMP.

Ціна: від приблизно 50 доларів на місяць.

Apptio Cloudability

Найкраще для звітності фінопсихологів, яку прийматиме фінансова команда.

Хмарність, частина IBM Apptio, створена для демонстрації та повернення коштів: розподіл витрат на багатохмарні технології між командами та продуктами у формі, яку визнає фінансовий відділ. Рекомендації з оптимізації надаються разом зі звітністю.

Переваги:

– Найкращий розподіл витрат та повернення коштів.

– Багатохмарний.

– Звітність фінансового рівня.

Недоліки:

– Ціноутворювальні ціни для підприємств.

– Звітність, а не практичний контроль.

Ціна: Вказана.

nOps

Найкраще для маєтків AWS, які хочуть автоматизованої економії.

nOps зосереджується на AWS та автоматизує управління зобов'язаннями, планування та оптимізацію розмірів, стягуючи частку заощаджень, а не фіксовану ліцензію. Така модель підходить командам, які хочуть отримати результат, а не ще одну панель інструментів.

Переваги:

– Ціна вказана з урахуванням економії.

– Потужна автоматизація AWS.

– Безкоштовний рівень.

Недоліки:

– AWS-орієнтований.

– Модель розподілу заощаджень потребує моделювання перед підписанням.

Ціноутворення: Безкоштовний рівень; оплата залежить від частки заощаджень.

CloudZero

Найкращий варіант за ціною на одного клієнта, функцію або команду.

CloudZero розподіляє витрати на хмарні технології на одиниці, зрозумілі для бізнесу – вартість на клієнта, на функцію, на середовище, – а не на екземпляр. Для SaaS-компанії, яка розраховує валовий прибуток, саме ця структура є продуктом.

Переваги:

– Економічний погляд на вартість хмарних послуг з точки зору одиниць.

– Сильне розподілення Kubernetes.

Недоліки:

– Запропонована ціна.

– Потрібна дисципліна тегування, щоб бути корисною.

Ціна: Вказана.

Vantage

Найкращий варіант для швидкого перегляду витрат на багатохмарні системи.

Vantage охоплює AWS, Azure, GCP та довгий список SaaS та постачальників інфраструктури, з безкоштовним рівнем та ціноутворенням, прив’язаним до відстежуваних витрат. Це найшвидший з цих сервісів для створення корисної картини без проекту впровадження.

Переваги:

– Дуже швидко налаштувати.

– Широке охоплення постачальників.

– Безкоштовний рівень.

Недоліки:

– Прозорість витрат, а не забезпечення чи управління.

Ціноутворення: Безкоштовний рівень; оплачується приблизно з 1,5% від відстежуваних витрат

Kubex

Найкраще підходить для Kubernetes та штучного інтелекту, оптимізуючи розмір робочого навантаження.

Kubex – це перейменований Densify, який тепер зосереджений на автоматизованій оптимізації ресурсів для робочих навантажень Kubernetes та штучного інтелекту – автоматичному масштабуванню та зміні розміру без ручного налаштування. Назву Densify більше не підтримується.

Переваги:

– Автоматизоване змінення розміру Kubernetes.

– Фокус на робочому навантаженні ШІ.

– Безкоштовний рівень.

Недоліки:

– Вужчий, ніж повноцінний CMP.

– Нещодавній ребрендинг від Densify.

Ціноутворення: безкоштовний рівень; платний рівень з пропозицією.

Finout

Найкращий варіант для перегляду хмарних сервісів, Kubernetes та SaaS за єдиною ціною.

Finout об'єднує витрати на хмарні технології, Kubernetes, платформу даних та SaaS в одну модель розподілу без необхідності попередньо перемаркувати все, що зазвичай є перешкодою для проектів FinOps.

Переваги:

– Працює з існуючими тегами.

– Широке охоплення джерел.

– Сильний розподіл.

Недоліки:

– Запропонована ціна.

– Орієнтований на звітність, а не на забезпечення.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програмних модулів системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у межах тестової платформи GENI (Global Environment for Network Innovations) було обрано мову програмування Python. Вибір цього інструментарію зумовлений високими вимогами до швидкості прототипування, читаності коду, гнучкості інтеграції з хмарними API та потужною екосистемою бібліотек для роботи з мережевими протоколами, телеметрією та алгоритмами адаптивного управління.

Python є високорівневою об'єктно-орієнтованою мовою програмування загального призначення з мінімалістичним синтаксисом і підтримкою кількох парадигм (структурної, об'єктно-орієнтованої, функціональної).

У контексті розробки системи відмовостійкості розподілених обчислень ключовими перевагами Python є:

– Динамічна типізація та вбудовані структури даних: забезпечують швидку обробку та трансформацію великих масивів логів, метрик стану хмарних вузлів і параметрів маршрутизації.

– Підтримка багатопотоковості та асинхронності: дозволяє ефективно реалізовувати механізми моніторингу в реальному часі, паралельного опитування вузлів тестової платформи GENI та швидкого реагування на збої.

– Модульність і масштабованість: чітка організація коду у функції, класи та пакети дає змогу легко нарощувати функціональність комплексу – від модулів збору телеметрії до інтелектуальних агентів адаптивного управління.

– Широка інтеграційна екосистема: наявність готових бібліотек для взаємодії з віртуалізованими середовищами, мережевими протоколами та

розподіленими базами даних значно спрощує розгортання та тестування платформи.

Завдяки своїй стабільності, кросплатформовості та розвиненим засобам взаємодії з різними операційними середовищами, Python забезпечує створення надійного, гнучкого та легко адаптованого програмного ядра для управління відмовостійкістю розподілених обчислень у хмарних тестових інфраструктурах.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Проблеми безпеки в хмарних наукових робочих процесах

Виконання наукових робочих процесів у хмарах має численні переваги, такі як легкий доступ до ресурсів, масштабованість на вимогу, підвищене використання ресурсів та економія коштів [12]. Незважаючи на ці проблеми, існують значні проблеми безпеки.

По-перше, логічні конструкції наукових робочих процесів вразливі до атак. Наукові робочі процеси зазвичай моделюються як орієнтовані ациклічні графи (DAG). Зв'язок залежності між вузлами в графі призведе до проблеми успадкування помилок, тобто помилки, що виникають на будь-якому етапі, будуть успадковані кінцевим результатом [18].

По-друге, порівняно зі звичайними бізнес-процесами, наукові процеси мають високі характеристики обчислювальної складності [19], тривалий час виконання [9] та високу дослідницьку цінність. Вони вважаються більш складними та важливими з таких причин:

– Для виконання наукових процесів потрібна велика кількість віртуальних машин. Віртуальні машини можна легко перетворити на цілі атак у хмарах [20], тому чим більше віртуальних машин, тим більший масштаб атаки.

– Наукові процеси можуть виконуватися місяцями або навіть роками [21], і це дає зловмисникам достатньо часу для атаки [22].

– Наукові процеси зазвичай застосовні до основних галузей наукових досліджень, таких як атмосферна наука, фізика високих енергій та біоінформатика [23], і будь-яка невелика помилка може завдати величезної шкоди [24].

Зрештою, хмари застосовують модель обслуговування з багатокористувацьким співіснуванням. Різні орендарі розділені логічними

межами, досягнутими технологіями віртуалізації на основі спільного використання одного фізичного об'єкта. Хоча гнучке управління ресурсами реалізується за допомогою цієї моделі, вона також створює власні ризики. З одного боку, зловмисники можуть через бічні канали порушити логічну межу та атакувати інших орендарів у тому ж фізичному об'єкті [25]. З іншого боку, у віртуалізованому середовищі також існує кілька вразливостей, таких як вразливості витоку віртуальної машини [26]. Як тільки зловмисник отримує права керування через ці вразливості для всього віртуалізованого середовища, він може контролювати всі віртуальні машини орендарів [27, 28].

Визначена проблема слугує каталізатором для цього систематичного огляду, зосередженого на алгоритмах стійкості до відмов та вторгнень, розроблених для планування робочих процесів у хмарному середовищі. Алгоритми класифікуються на основі типу стійкості, яку вони пропонують, з акцентом на їхні цілі та властивості. Крім того, ми висвітлюємо покращення, досягнуті завдяки поєднанню методів стійкості до відмов та методів стійкості до вторгнень. Цей аналіз надає порівняльний огляд схем, виділяючи їхні відмінні риси, щоб надихнути на подальші дослідження в цій галузі.

Через особливості наукових робочих процесів, вирішення проблем безпеки наукових робочих процесів у хмарах є досить складним завданням. Багато наукових операцій є обчислювально ресурсоємними [29], що вимагає використання віртуальних машин. Однак зловмисники можуть легко атакувати ці віртуальні машини [20]. Аналогічно, реалізація наукових робочих процесів займає багато часу [9]. Це дає зловмисникам достатньо часу для запуску атаки. Крім того, науковий робочий процес зазвичай є типом DAG, який особливо чутливий до атак, оскільки проміжні помилки успадковуються в кінцевому результаті [18].

У кількох наукових галузях проміжні дані наукового робочого процесу також містять конфіденційні дані. Користувачам буде завдано серйозної шкоди, якщо ці дані будуть зламані [30]. Це одна з основних загроз безпеці планування наукового робочого процесу, тобто проникнення неавторизованих суб'єктів у

хмарну систему. З огляду на ці проблеми, стає зрозуміло, що відмовостійкість та стійкість до вторгнень є двома основними факторами, які впливають на якість послуг, що надаються постачальниками хмарних послуг, а також на послуги, що використовуються споживачами.

3.2 Розробка структурної схеми

Широке використання хмарних обчислювальних послуг стало наслідком можливості гнучкого та еластичного розподілу ресурсів на основі вимог користувачів на основі оплати за використання. Користувачам більше не потрібно купувати, керувати або оновлювати свої обчислювальні ресурси. Натомість вони можуть отримати різні інформаційно-технологічні можливості, такі як обчислення, сховище, пам'ять, мережі, програмне забезпечення тощо, як послуги за низькою ціною та без клопоту, пов'язаного з володінням або обслуговуванням ресурсів. З іншого боку, постачальники хмарних послуг отримують вигоду завдяки масштабу використання своїх послуг, а також ідеї багатокористувацького обслуговування. Багатокористувацька архітектура дозволяє хмарному постачальнику спільно використовувати ресурси між кількома користувачами, тим самим сплачуючи менше. Користувач також може запитувати виділені ресурси, хоча це буде коштувати дорожче.

Незважаючи на те, що хмарні обчислювальні сервіси використовуються для величезної кількості застосувань, існують побоювання щодо збоїв і, отже, зниження продуктивності в хмарі. Збої впливають на надійність, а також на доступність сервісів і потребують ефективного вирішення. Тому відмовостійкість є одним з ключових елементів Цілей рівня обслуговування (SLO) для користувачів хмарних сервісів. FT – це здатність пристрою або системи продовжувати свою роботу у випадку, якщо деякі з його компонентів могли вийти з ладу. Існуючі підходи до відмовостійкості в хмарі базуються на традиційних розподілених обчислювальних підходах, які загалом поділяються на дві категорії: реактивні та проактивні. Реактивні підходи включають

протоколи відновлення на основі протоколів контрольних точок та реплікації, і ці підходи обробляють збої після їх виникнення. На противагу цьому, проактивні підходи намагаються оцінити виникнення збоїв до їх виникнення. Приклади включають підходи до міграції завдань та самовідновлення. З цих двох категорій реактивні підходи знайшли ширше впровадження в хмарних обчисленнях завдяки своїй простоті та легкості реалізації).

У даній магістерській роботі досліджується застосування реактивних підходів до відмовостійкості, тобто контрольних точок та реплікації, у хмарі інфраструктури як послуги (IaaS). Контрольні точки – це добре відомий метод, за якого стан активної віртуальної машини (ВМ) періодично зберігається під час нормального виконання. У разі збою завдання потрібно відкотити до останньої збереженої контрольної точки замість перезапуску; таким чином запобігаючи втратам під час виконання. У хмарних обчисленнях контрольні точки широко використовуються для періодичного збереження стану активної віртуальної машини для відновлення на фізичній машині, відмінній від тієї, на якій вона розміщена. Віртуальна машина може отримати свій останній збережений стан без помилок, якщо збій трапляється до його завершення. З іншого боку, реплікація завдання або ресурсу передбачає збереження їхньої копії. Тут копія може бути використана для заміни основного завдання або ресурсу у разі збою. Реплікація базується на використанні надмірності для обробки збоїв і тому використовується в хмарних обчисленнях скрізь, де ресурси, загалом, доступні в надлишку.

Кожен із двох підходів, тобто контрольні точки та реплікація, має свої переваги та Недоліки:. Контрольні точки використовують значно менше ресурсів порівняно з реплікацією, оскільки періодично потрібно зберігати лише стан завдання. Однак, контрольні точки пов'язані з накладними витратами як під час збереження контрольних точок, так і під час відновлення. Крім того, контрольна точка дозволяє відновитися після одного збою, оскільки одночасні збої машин, на яких розміщена віртуальна машина та зберігається її контрольна точка, неможливо обробити. Для порівняння, k-реплікація, тобто збереження k

реплік, гарантує відновлення після k-1 збоїв, хоча вона передбачає використання k-кратного перевищення фактично необхідних ресурсів. Таким чином, очевидно, що вибір підходу стійкості до відмов має бути обґрунтованим для досягнення найкращих результатів з мінімальним споживанням ресурсів.

У цій роботі пропонується адаптивний підхід до вибору відповідного підходу до FT (Field Transmission – Перетворення фактів) у середовищі хмарних обчислень. Запроектована схема адаптується до конкретного середовища впровадження, оскільки враховує фактори, що відображають як інтереси користувача, так і можливості постачальника хмарних послуг. По-перше, історичні дані фізичних машин у хмарі використовуються для класифікації машин на основі ризику їхнього виходу з ладу. Ця класифікація враховує зареєстрований час простою машини, характер несправності, тип фізичної машини та її призначення. Ми використали алгоритм k найближчих сусідів (KNN) для кластеризації фізичних машин у категорії збоїв високого, середнього та низького ризику. Потім кожній машині було присвоєно мітку ризику залежно від кластера, в якому вона знаходиться. Згодом нечітка логіка була використана для розробки правил для визначення відповідного підходу до відмовостійкості для віртуальних машин, розміщених на певній фізичній машині. Нечіткі правила базуються на трьох вхідних змінних.

Перша – це рівень пріоритету віртуальної машини, визначений кінцевим користувачем.

По-друге, ми розглядаємо мітку ризику фізичної машини, призначену алгоритмом кластеризації.

По-третє, остання вхідна змінна береться як рівень доступності сховища.

Тут перша вхідна змінна відображає інтерес кінцевого користувача, тоді як дві інші змінні є показником можливостей постачальника хмарних послуг. Нечіткі правила спрямовані на отримання результату, тобто контрольної точки або реплікації, після всебічного врахування інтересів обох зацікавлених сторін. Результати моделювання підтвердили, що запропонований підхід перевершує підходи контрольної точки або реплікації, коли вони виконуються окремо.

Основні моменти такі:

– Запропоновано новий підхід до класифікації та відмовостійкості на основі нечітких правил.

– Використання системи нечіткого висновку враховує реальні вимоги користувачів та можливості постачальника хмарних послуг.

– Запропонований підхід є адаптивним і, отже, може використовуватися динамічно залежно від фактичного стану системи.

GENI

Користувачі GENI можуть отримувати доступ до ресурсів з об'єднаних тестових платформ за допомогою свого облікового запису GENI. У багатьох випадках вони можуть додати ці ресурси до свого сегмента GENI, щоб запустити експеримент, який охоплює кілька тестових платформ по всьому світу:

– Дослідники використовують CloudLab для створення власних хмар, експериментуючи з новими архітектурами, які стануть основою для наступного покоління обчислювальних платформ. Користувачі GENI можуть отримати доступ до CloudLab з порталу GENI та увійти, використовуючи свій обліковий запис GENI.

– Chameleon – це масштабна платформа для дослідження концепцій глибоко програмованих хмарних сервісів, дизайну та основних технологій. Вона дозволяє користувачам досліджувати проблеми, починаючи від створення програмного забезпечення як послуги (ПЗ як послуга) і закінчуючи підтримкою віртуалізації ядром. Користувачі GENI можуть отримати доступ до Chameleon з порталу GENI та увійти, використовуючи свій обліковий запис GENI.

– Emulab – це тестовий стенд для емуляції мережі, який надає дослідникам широкий спектр середовищ для розробки, налагодження та оцінки своїх систем. Emulab реалізує API GENI Aggregate Manager; користувачі GENI можуть включати ресурси Emulab до своїх зрізів.

– ORBIT – це емулятор бездротової мережі, розроблений для відтворюваних експериментів, а також підтримує реалістичну оцінку протоколів

і застосувань. Користувачі GENI можуть отримати доступ до ORBIT з порталу GENI та увійти, використовуючи свій обліковий запис GENI.

– WITest призначений для досліджень, освіти та інформаційно-просвітницької роботи, зосередженої на впровадженні та експериментуванні з протоколами, додатками та послугами бездротових мереж. Користувачі GENI можуть отримати доступ до WITest з порталу GENI та увійти, використовуючи свій обліковий запис GENI.

– Віртуальна стіна iMinds складається з понад 350 серверів, які можна використовувати як базове обладнання або віртуалізувати за допомогою XEN. Віртуальна стіна iMinds реалізує API GENI Aggregate Manager; користувачі GENI можуть включати ресурси Віртуальної стіни до своїх сегментів.

– SAVI – це канадська дослідницька платформа, яка підтримує експериментальні дослідження в галузі хмарних обчислень, орієнтованих на прикладні мережі, інтегрованих бездротових/оптичних мереж доступу та архітектур майбутнього Інтернету. Користувачі GENI можуть отримати доступ до SAVI та використовувати її з порталу GENI, використовуючи свій обліковий запис GENI.

– PlanetLab – це глобальна дослідницька мережа для вивчення мережевих явищ, створення нових мережевих протоколів, оцінки нових та існуючих мережевих сервісів і розгортання нових мережевих сервісів у глобальному масштабі. PlanetLab реалізує API GENI Aggregate Manager; користувачі GENI можуть включати ресурси PlanetLab до своїх сегментів.

«Шари» – це сукупність фізичних чи віртуальних ресурсів (обчислювальних, мережевих та систем зберігання даних), які об'єднуються та налаштовуються як єдина інфраструктура для виконання завдань конкретного користувача (або групи). Такі середовища повністю ізольовані одне від одного, що фактично означає створення гнучких, програмно-конфігурованих віртуальних систем під конкретні потреби.

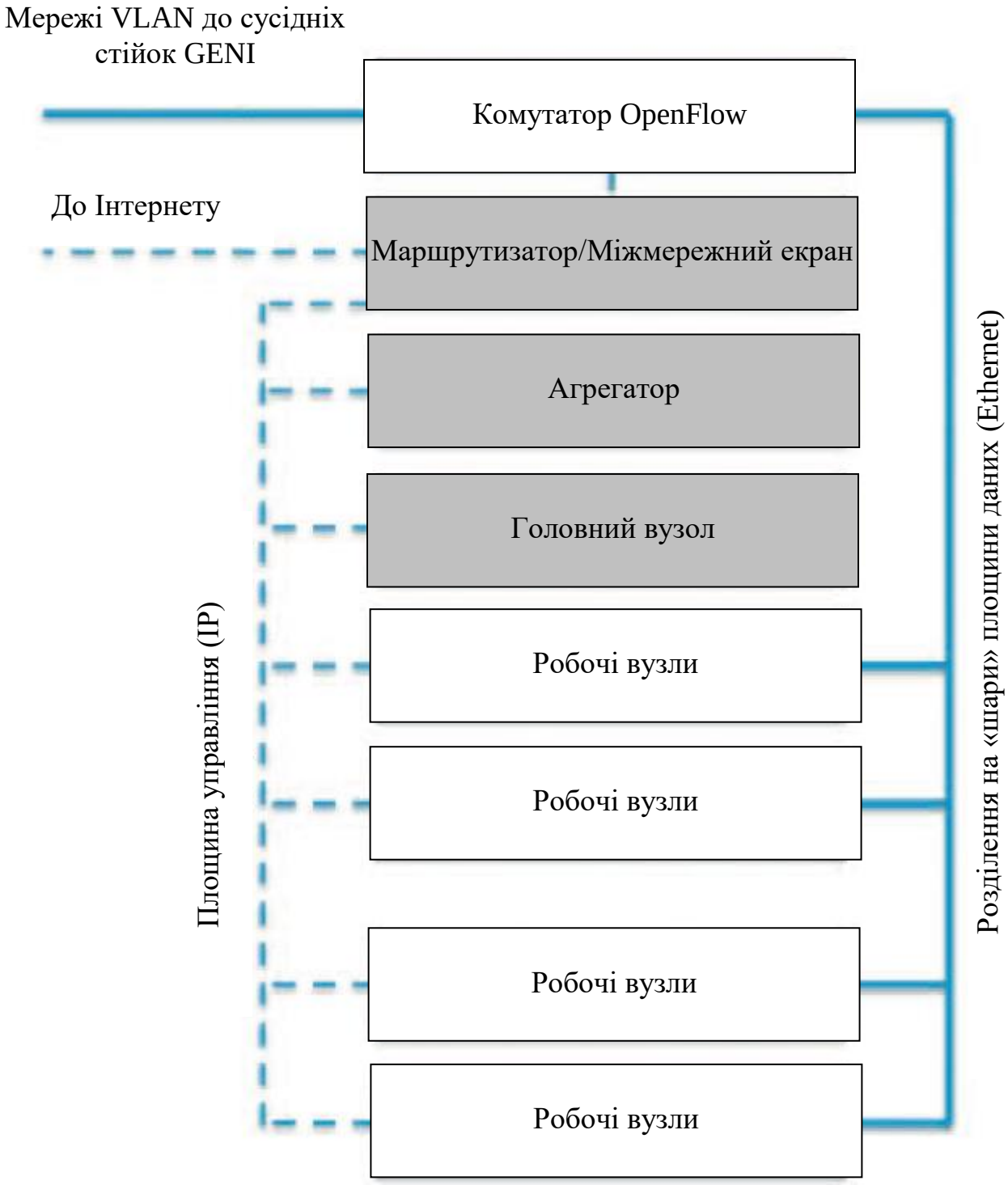


Рисунок 3.1 – Структурна схема системи

У проєкті GENI базовими елементами є спеціальні стійки з обчислювальними потужностями, СЗД та мережевим обладнанням. Для віртуалізації та створення мережевих «шарів» у межах однієї стійки зазвичай

застосовують технологію VLAN. Проте для зв'язку між різними стійками VLAN має Недоліки:, оскільки не гарантує достатнього рівня ізоляції за продуктивністю та програмним контролем. Тому тут ефективніше використовувати сучасні підходи на базі SDN та OpenFlow.

Для забезпечення взаємодії між територіально віддаленими компонентами активно використовується експериментальна мережа Internet2, яка також функціонує на основі технології OpenFlow.

GENI дає змогу проводити масштабні експерименти з обчислювальним і мережевим обладнанням – наприклад, моделювати поведінку складних великомасштабних мереж, а також тестувати новітні мережеві архітектури й сервіси. Крім того, інфраструктура GENI надає дослідникам із будь-якої точки світу віддалений доступ до розподілених обчислювальних ресурсів мережі.

Серед ключових тенденцій розвитку сучасних мереж і розподілених хмар виділяються міжнародна федерація випробувальних стендів (FIDC), а також активна розробка програмно-конфігурованих пунктів обміну трафіком (SDX) і програмно-конфігурованих інфраструктур (SDI).

Питання створення федерації є переважно організаційним, а не технологічним. Головні завдання тут – узгодити правила спільного використання ресурсів та запровадити єдиного довіреного брокера. Основою безпеки та управління мають стати три принципи («3A»): Authentication (автентифікація), Authorization (авторизація) та Accounting (облік споживаних ресурсів).

Технічна реалізація відносно проста, натомість справжнім викликом є розробка політик інтеграції ресурсів та досягнення домовленостей між різними організаціями й інститутами.

Ключову роль у побудові федерацій відіграють вузли SDX (Software Defined eXchanges). Вони слугують фізичними чи віртуальними майданчиками для безпечного обміну трафіком між різними мережами SDN відповідно до їхніх внутрішніх політик. Своєю чергою, концепція SDI (Software Defined Infrastructure) поєднує ці ресурси, мережі та вузли SDX в єдину систему, яку

користувачі можуть гнучко задіювати для створення мультидоменних програмно-конфігурованих середовищ.

Сьогоднішня мережева інфраструктура складається з розрізнених «острівців» SDN, тоді як завдання SDX – об'єднати їх в єдину систему. Масове впровадження SDN та SDX здатне докорінно змінити саму архітектуру сучасного Інтернету.

Наразі SDX залишається переважно концептуальною технологією, і повноцінних комерційних рішень ще немає, проте вже існує низка робочих прототипів. Один із них розроблений Міжнародним центром iCAIR спільно з партнерами та розгорнутий на майданчику StarLight у Чикаго. Цей хаб забезпечує обмін трафіком на фізичному рівні (оптичні потоки) та рівні L2 між численними національними й міжнародними науковими мережами. Масштаби цієї площадки вражають: до неї підключено майже 30 оптичних потоків по 100 Гбіт/с, велика кількість каналів 40G, а також кілька сотень 10-гігабітних ліній.

Поява StarLight SDX стала ключовим кроком у реалізації ініціативи iGENI, яка спрямована на створення глобальної тестової мережі на базі технологій OpenFlow/SDN. Цю інфраструктуру планують розгорнути на основі всесвітньої оптичної мережі **GLIF** (Global Lambda Integrated Facility), яка охоплює зокрема й Україну, причому StarLight виступає одним із її головних вузлів.

StarLight SDX є великим віртуальним комутатором, побудованим на базі кількох фізичних SDN/OpenFlow-комутаторів. Його першочергове завдання – зв'язати між собою численні локальні мережі SDN, а також підтримувати традиційні мережеві сервіси рівня L2 та безпосередньо оптичні послуги.

Для забезпечення взаємодії між вузлами SDX залучаються три окремі транспортні мережі.

У межах проєкту планується застосування SDN-контролера, який підтримує широкий спектр мережевих сервісів і додатків. Серед них – маршрутизація рівня L2/L3 з підтримкою якості обслуговування (QoS), багатопотокова маршрутизація, фільтрація трафіку, трансляція адрес (NAT),

балансування навантаження, мережева віртуалізація, захист від DDoS-атак, верифікація мереж, а також інтеграція з системами управління центрами обробки даних (ЦОД) тощо.

Сучасні мережеві концепції, що базуються на програмованості, відкривають для українських компаній чудові перспективи перетворитися на повноцінних гравців ринку мережевих технологій, тоді як раніше Україна виступала переважно в ролі споживача готових закордонних рішень.

3.3 Розробка функціональної схеми

Функціональну схему розроблюваної системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI наведено на рисунку 3.2. Програмний комплекс складається з низки взаємопов'язаних функціональних блоків:

1. Блок інтерфейсу користувача: забезпечує взаємодію дослідника або оператора з системою.

2. Блок визначення параметрів QoS: призначений для аналізу поточних показників якості обслуговування, до яких належать смуга пропускання (Bandwidth), затримка (Delay), варіація затримки (Jitter) та втрати пакетів (Packet Loss).

3. Блок примусового завдання параметрів QoS: здійснює конфігурацію та примусове встановлення цільових значень параметрів якості обслуговування.

4. Блок моделювання завантаженості мережі: моделює інтенсивність трафіку та навантаження в тестовому середовищі GENI за заданих умов QoS.

5. Блок моніторингу мережі: виконує постійний аналіз поточного стану розподіленої хмарної платформи.

6. Блок дослідження можливостей механізмів WRED (Weighted Random Early Detection): досліджує методи запобігання перевантаженням шляхом керованого відкидання пакетів з урахуванням їх пріоритетів.

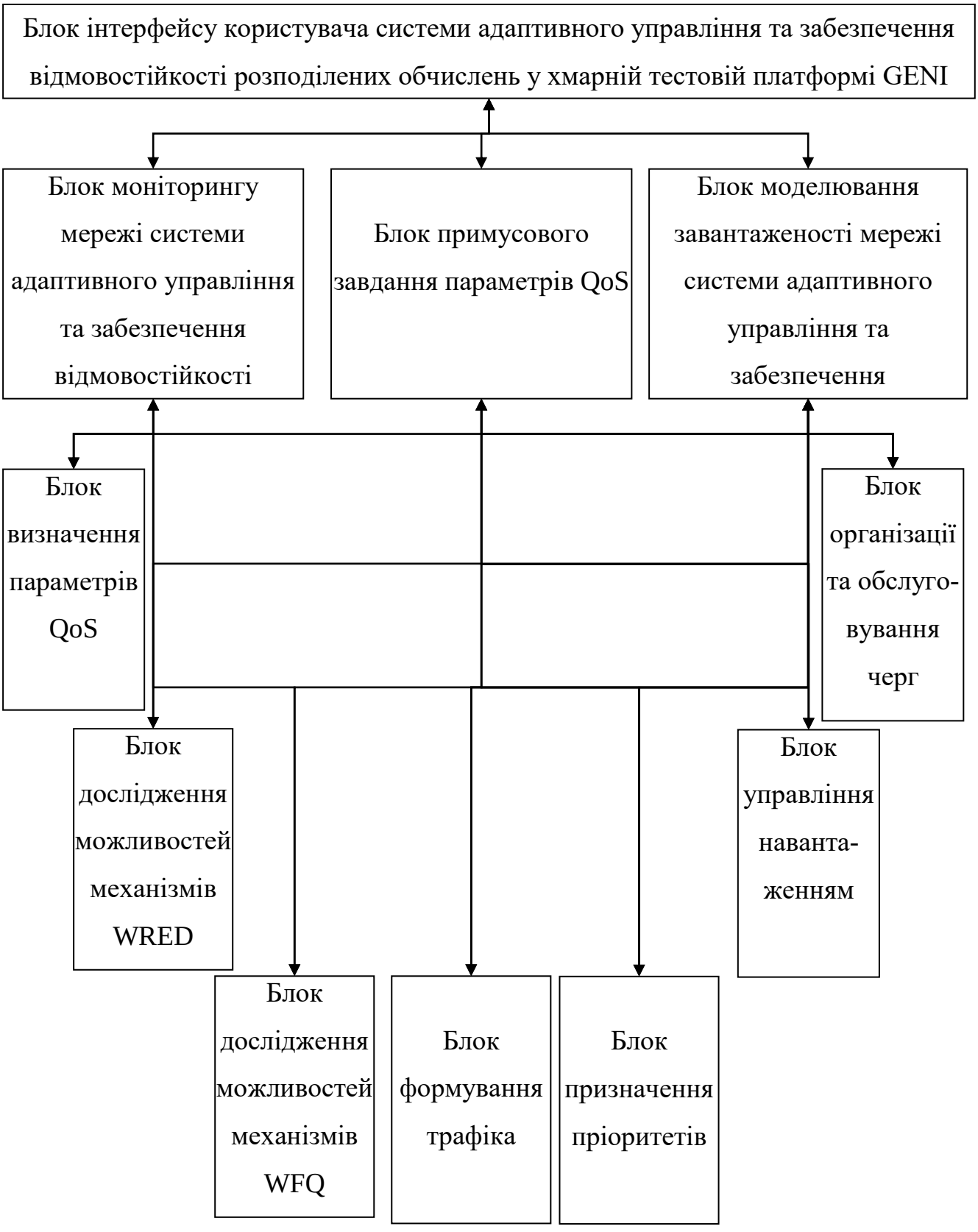


Рисунок 3.2 – Функціональна схема системи

7. Блок дослідження можливостей механізмів WFQ (Weighted Fair Queuing): реалізує управління перевантаженням на основі зваженого справедливого розподілу смуги пропускання між потоками.

8. Блок призначення пріоритетів: реалізує явні та неявні методи класифікації і пріоритезації трафіку (зокрема на основі полів IP TOS).

9. Блок організації та обслуговування черг: керує буферизацією та порядком передачі пакетів у комутаторах і маршрутизаторах.

10. Блок управління навантаженням: забезпечує стабілізацію роботи мережі в умовах перевантажень.

11. Блок формування трафіка: відповідає за процедури сегментації, вирівнювання та маніпулювання пакетами для мінімізації затримок.

Детальний огляд ключових компонентів системи:

– Організація та обслуговування черг: Після призначення пріоритетів алгоритми обслуговування визначають черговість передачі пакетів із буферів пам'яті. У разі перенавантаження мережі сучасні служби QoS гарантують першочергову передачу важливих даних. Використання вдосконалених алгоритмів, таких як WFQ, дозволяє ефективно розподіляти пропускну здатність між різними додатками.

– Управління навантаженням: Для запобігання критичним збоям у системі застосовується механізм WRED, який заповнює прогалину між простим відкиданням пакетів і повним переповненням черг, знижуючи інтенсивність відправки трафіку джерелом до настання колапсу.

– Формування трафіку (Traffic Shaping): Комплекс процедур із вирівнювання та сегментації пакетів забезпечує дотримання часових параметрів передачі, запобігаючи виникненню мережевих ланцюжків і знижуючи загальну затримку в розподіленій інфраструктурі GENI.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерської роботи, наведена на рисунку 3.3. Детальний розгляд процесної діаграми дозволяє простежити характер взаємодії внутрішніх компонентів та маршрутизацію інформаційних потоків у розробленій системі. У процесі проєктування застосовується модель графічного представлення потоків даних (Data Flow Diagram, DFD), яка наочно відображає рух, перетворення та збереження інформації в інформаційній системі (зокрема, параметрів трафіку, метрик якості обслуговування QoS та станів вузлів розподілених обчислень).

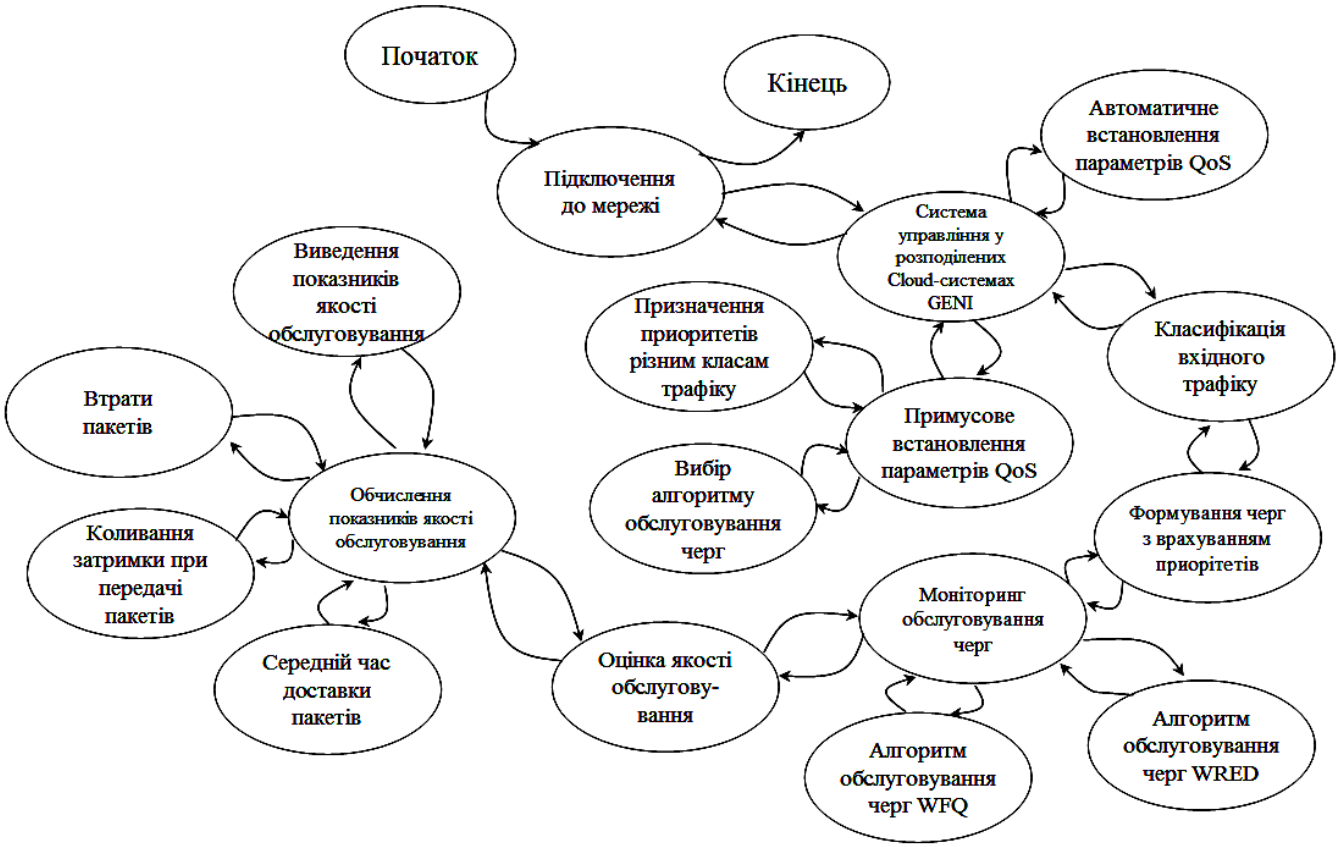


Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми взаємодії процесів є фундаментальним інструментом структурного проєктування для візуалізації обробки даних. Традиційно розробка

розпочинається з побудови контекстної діаграми потоків даних, яка ілюструє загальну взаємодію розроблюваної системи адаптивного управління та відмовостійкості із зовнішнім середовищем (дослідником, компонентами хмарної платформи GENI та джерелами мережевого трафіку). У подальшому ця модель деталізується шляхом поетапної декомпозиції процесів і потоків даних з метою повної та всебічної демонстрації архітектури комплексу, що створюється.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Алгоритмічне забезпечення системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень призначене для координації обчислювальних завдань, раціонального використання доступних ресурсів і підтримання виконання експериментів за умов часткових відмов інфраструктури. Основним завданням є автоматизоване визначення стану обчислювальних вузлів, розподілення завдань відповідно до характеристик ресурсів, виявлення порушень працездатності та відновлення незавершених обчислень. Для цього передбачається поєднання алгоритмів адаптивного планування, моніторингу продуктивності, резервування стану завдань і керування відновленням. Особливістю досліджуваної системи є орієнтація на архітектуру експериментальної хмарної платформи GENI (Global Environment for Network Innovations), яка використовувала федеративну модель надання обчислювальних і мережевих ресурсів. Загальну послідовність функціонування системи наведено на рисунку 4.1. Основний алгоритм передбачає отримання опису розподіленого обчислювального експерименту, визначення необхідних ресурсів, формування конфігурації експериментального середовища та перевірку доступності виділених вузлів. Після підготовки інфраструктури виконується розподілення обчислювальних завдань між доступними виконавцями. У процесі роботи система збирає телеметрію, контролює виконання операцій і виявляє відхилення від очікуваного стану. За необхідності здійснюється перегляд плану обчислень. У разі відмови компонента визначається можливість повторного виконання завдання або відновлення з попередньо збереженого стану. Результати операцій передаються до підсистеми зберігання та використовуються для подальшого аналізу ефективності.

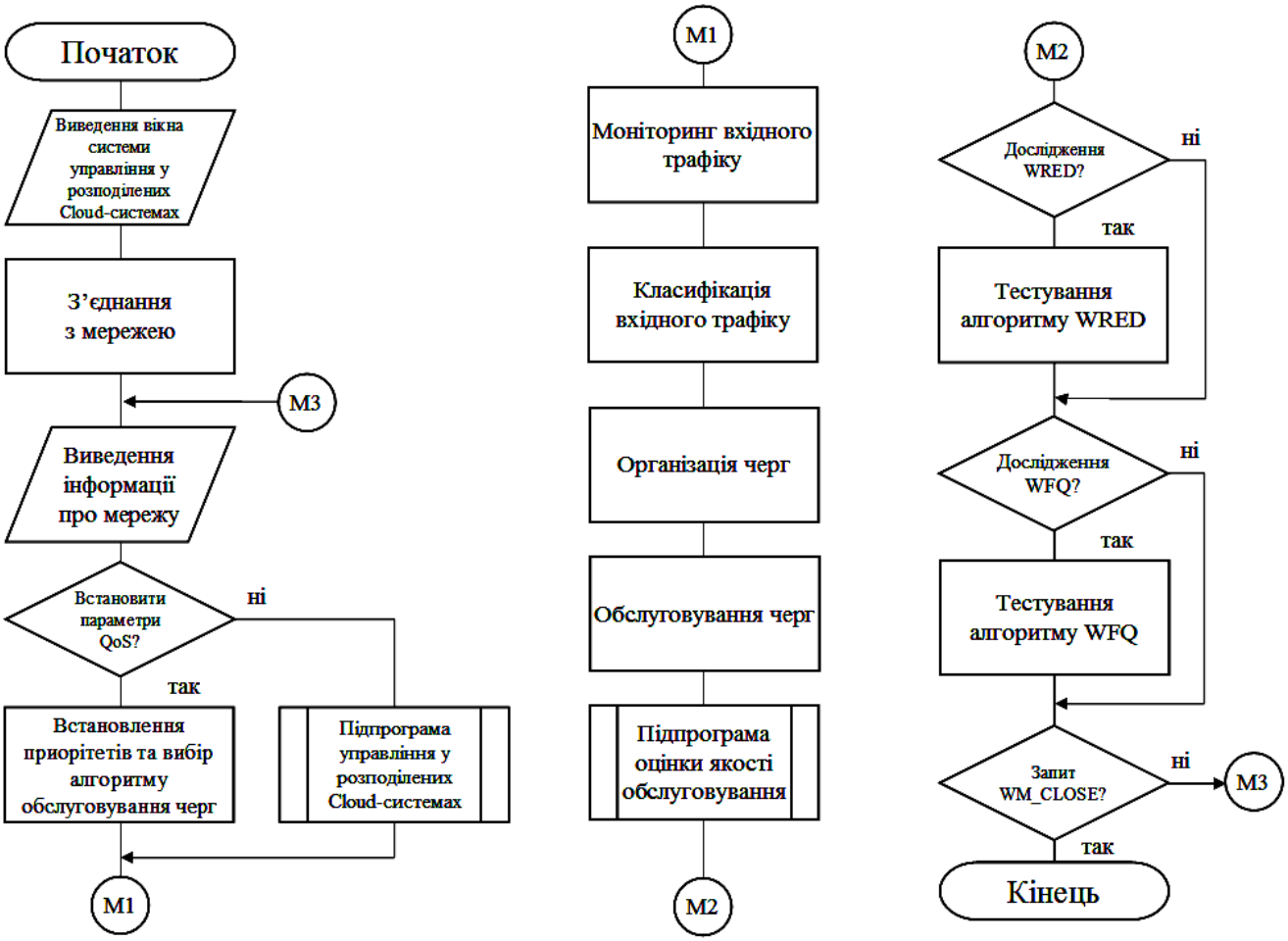


Рисунок 4.1 – Блок-схема основної програми

Архітектура GENI передбачала використання декількох рівнів керування ресурсами. Обчислювальні та мережеві ресурси надавалися окремими інфраструктурними агрегатами. Для організації експерименту використовувалася логічна область ізоляції Slice, яка містила виділені ресурси Sliver. Опис необхідної конфігурації формувався у вигляді Resource Specification (RSpec). Взаємодія з агрегатами здійснювалася через GENI Aggregate Manager API. Така модель дозволяла дослідникам формувати експериментальні топології з обчислювальних вузлів і мережевих з'єднань.

Водночас сама платформа не забезпечувала універсального прикладного планувальника розподілених завдань. Тому алгоритми адаптивного керування та відновлення необхідно розглядати як функції окремого програмного комплексу, який використовує виділені ресурси експериментального середовища.

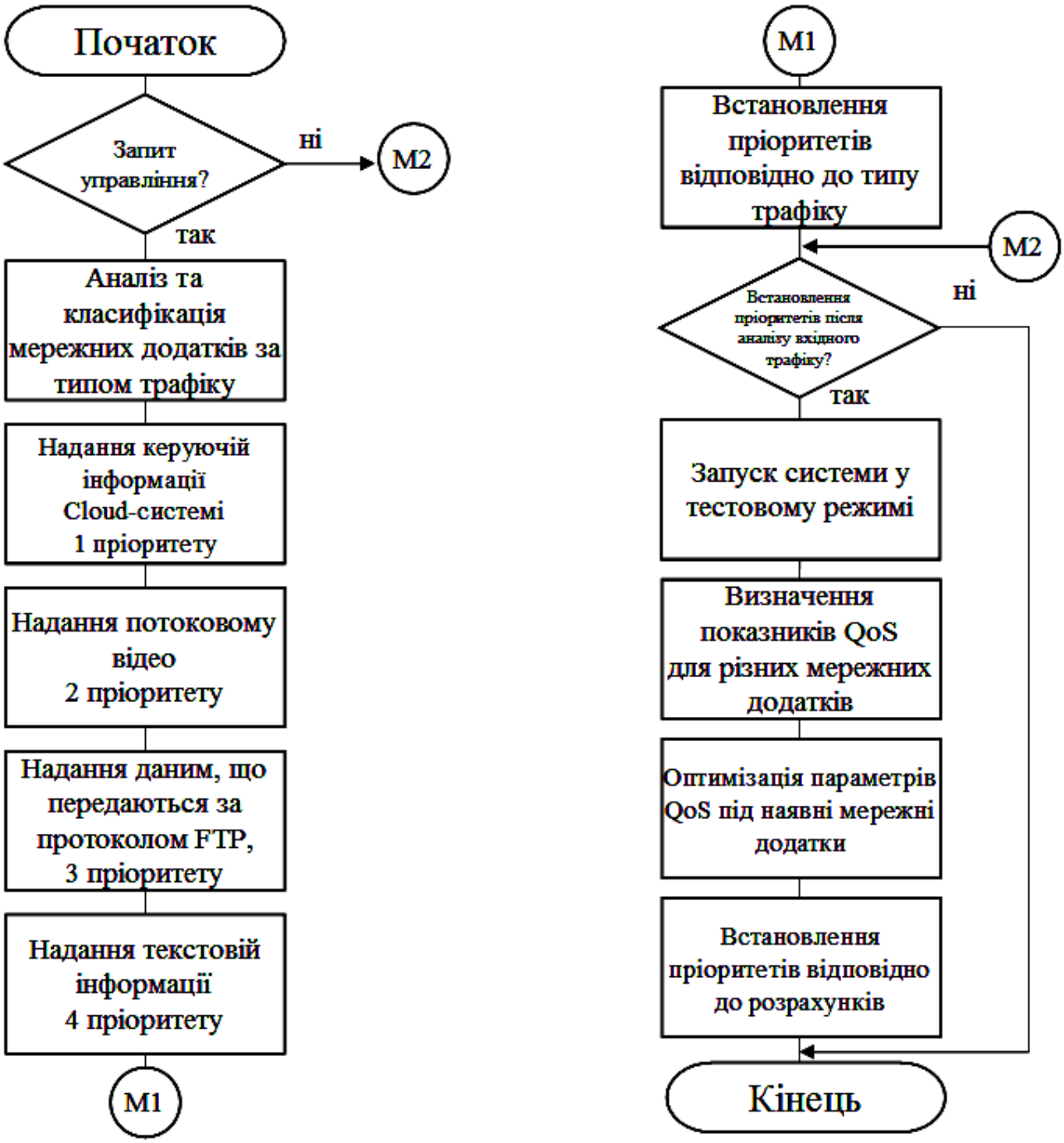


Рисунок 4.2 – Блок-схема роботи підпрограми

Слід враховувати актуальний стан досліджуваної платформи. Програмна модель, орієнтована на GENI, може досліджуватися на основі збережених архітектурних специфікацій, сумісного емулятора керувальних інтерфейсів або адаптації до іншої доступної експериментальної платформи. Це не змінює змісту

алгоритмів адаптивного управління, але визначає обмеження їхньої практичної перевірки.

Для реалізації системи доцільно передбачити керувальний сервер, службу планування завдань, компоненти моніторингу, агентів обчислювальних вузлів і сховище службового стану. Керувальний сервер приймає описи експериментів та координує їхнє виконання. Планувальник визначає виконавців із достатніми ресурсами. Агенти отримують завдання, запускають обчислювальні процеси та передають повідомлення про їхній стан. Підсистема моніторингу формує поточну картину доступності вузлів. Компонент відновлення аналізує виявлені відмови та визначає допустимі способи продовження обчислень. Зберігання службового стану поза короткочасними обчислювальними вузлами дозволяє зменшувати залежність керування від життєвого циклу окремого експерименту.

Окремим функціональним елементом є підпрограма адаптивного вибору виконавця та перевірки можливості відновлення завдання. Принцип її функціонування представлено на рисунку 4.2. Підпрограма отримує характеристики завдання, перелік доступних вузлів, поточні показники завантаження та відомості про попередні відмови. На першому етапі відбираються ресурси, які відповідають мінімальним технічним вимогам. Далі для допустимих кандидатів оцінюються очікувана тривалість виконання, доступність обчислювальних ресурсів і витрати на передавання даних. Після вибору виконавця формується рішення про призначення завдання. Якщо придатні вузли відсутні, завдання залишається в черзі. За наявності відмови додатково перевіряється можливість використання контрольної точки або повторного виконання.

Під час побудови алгоритмів необхідно відокремлювати керування інфраструктурою від керування прикладними обчисленнями. Операції GENI Aggregate Manager API стосувалися отримання інформації про ресурси, їхнього виділення, перевірки стану та звільнення. Запуск прикладних обчислювальних завдань потребував додаткового програмного забезпечення, встановленого на виділених вузлах. Відповідно, створення Sliver не означає автоматичної

готовності обчислювального агента. Система повинна окремо підтверджувати працездатність інфраструктурного ресурсу та готовність прикладного компонента до виконання завдань.

Алгоритм реєстрації розподіленого обчислювального експерименту

Початковим етапом роботи програмного комплексу є отримання опису експерименту. Він визначає склад обчислювальних завдань, їхні взаємозалежності, необхідні ресурси та умови завершення. Вхідними даними алгоритму є ідентифікатор експерименту, перелік операцій, характеристики вхідних даних і встановлені обмеження виконання.

На першому етапі перевіряється повнота опису. Для кожного завдання повинні бути визначені унікальний ідентифікатор, тип операції, приблизні вимоги до процесора й оперативної пам'яті та спосіб отримання вхідних даних. Якщо завдання використовує результат іншої операції, відповідна залежність реєструється в описі експерименту.

Наступним етапом є перевірка структури залежностей. Для цього обчислювальний процес може представлятися орієнтованим графом, вершини якого відповідають завданням, а ребра визначають порядок використання результатів. Для звичайного пакетного виконання доцільно використовувати орієнтований ациклічний граф (Directed Acyclic Graph, DAG).

Якщо в описі виявлено циклічну залежність, яка не передбачена моделлю обчислень, система повинна відхилити такий експеримент або повідомити про необхідність уточнення. Інакше частина завдань може залишатися в очікуванні без можливості переходу до виконання.

Після перевірки формується запис експерименту. Він містить перелік завдань, граф залежностей, поточний стан та ідентифікатор конфігурації ресурсів. Окремо визначаються умови успішного завершення.

Результатом алгоритму є перевірений опис обчислювального експерименту, придатний для визначення необхідної інфраструктури.

Алгоритм формування конфігурації ресурсів GENI

Для розгортання експерименту в архітектурі GENI необхідно визначити набір обчислювальних і мережевих ресурсів. Відповідна конфігурація історично представлялася за допомогою документів RSpec.

Вхідними даними алгоритму є характеристики завдань, необхідна кількість вузлів, вимоги до мережевої взаємодії та відомості про доступні інфраструктурні агрегати.

На першому етапі визначається мінімальна кількість обчислювальних ресурсів. Для цього враховуються кількість незалежних завдань, можливість паралельного виконання та обсяг пам'яті, необхідний кожному процесу.

Далі формується логічна топологія експерименту. Для розподіленого обчислення можуть передбачатися керувальний вузол, декілька виконавців і відповідні мережеві з'єднання. Конкретна кількість компонентів повинна визначатися потребами обчислювального процесу.

Після цього створюється опис необхідних ресурсів. У моделі GENI для цього використовувався Request RSpec. Він дозволяв задати параметри вузлів і мережеві зв'язки відповідно до можливостей обраного агрегата.

Під час перевірки необхідно враховувати, що різні агрегати могли підтримувати неоднакові типи ресурсів. Тому універсальна конфігурація не завжди могла бути застосована без змін до всіх постачальників інфраструктури.

Результатом алгоритму є перевірений опис необхідної топології та параметрів експериментального середовища.

Алгоритм резервування та підготовки обчислювальних вузлів

Після формування конфігурації система повинна забезпечити підготовку доступних обчислювальних ресурсів. Для історичної архітектури GENI відповідні операції виконувалися через Aggregate Manager API.

На початковому етапі перевіряється наявність дозволу користувача на керування відповідною областю Slice. Далі визначається набір агрегатів, у яких можуть бути отримані необхідні ресурси.

Після передавання запиту виконується аналіз відповіді інфраструктурного компонента. Якщо виділення ресурсів підтримується, система отримує відповідний опис фактично наданих вузлів.

У GENI для представлення виділених ресурсів використовувався Manifest RSpec. Він міг містити характеристики обчислювальних вузлів, параметри мережевих інтерфейсів і необхідні відомості для підключення.

Наступним етапом є перевірка фактичної доступності ресурсів. Успішна відповідь інфраструктурного API не повинна вважатися достатнім підтвердженням готовності прикладного середовища.

Для цього виконується підключення до вузла через дозволений службовий інтерфейс. Після встановлення зв'язку перевіряється наявність необхідних програмних компонентів і можливість запуску обчислювального агента.

Якщо вузол не відповідає встановленим вимогам, він позначається як непридатний до виконання завдань. При цьому необхідно відрізняти тимчасову недоступність від остаточної відмови виділення ресурсу.

Результатом алгоритму є перелік підготовлених обчислювальних вузлів із підтвердженими характеристиками та станом готовності.

Алгоритм контролю життєвого циклу експериментальних ресурсів

Особливістю GENI було надання ресурсів на обмежений період. Виділені Sliver належали до відповідних Slice та мали визначені часові обмеження використання. Після завершення дозволеного періоду ресурси могли звільнятися інфраструктурою.

Тому алгоритм керування повинен ураховувати не лише поточну доступність вузла, а й допустиму тривалість його використання.

Вхідними даними є відомості про виділені ресурси, час завершення їхнього використання та оцінка тривалості незавершених завдань.

На першому етапі визначається залишковий допустимий час роботи ресурсу. Далі він порівнюється з прогнозованою тривалістю обчислень та необхідним резервом для збереження результатів.

Якщо до завершення дозволеного періоду залишається недостатньо часу, система повинна обмежити призначення нових тривалих завдань. Для вже активних процесів може ініціюватися контрольоване створення контрольної точки.

Якщо інфраструктура підтримує продовження терміну резервування, відповідна операція може виконуватися через дозволений API. Водночас її успішність залежить від політики агрегата та доступності ресурсів.

За відсутності можливості продовження система повинна завершити необхідні операції, зберегти доступні результати та звільнити ресурси відповідно до регламенту.

Результатом алгоритму є контроль використання обмежених за часом ресурсів без необґрунтованого припущення про їхню постійну доступність.

Алгоритм запуску та реєстрації обчислювальних агентів

Після підготовки інфраструктури на обчислювальних вузлах необхідно активувати компоненти, які безпосередньо виконують завдання.

Вхідними даними алгоритму є характеристики вузла, параметри керувального сервера та конфігурація прикладного середовища.

На першому етапі перевіряється доступність необхідних програмних залежностей. Далі запускається агент виконання. Він повинен встановити захищене службове з'єднання з керувальним компонентом і передати відомості про свої ресурси.

Після реєстрації агент отримує унікальний ідентифікатор. Для нього зберігаються характеристики процесора, оперативної пам'яті, доступного дискового простору та мережевого інтерфейсу.

Окремо визначається максимальна кількість завдань, які можуть виконуватися одночасно. Це значення залежить від характеру обчислень. Наприклад, процеси з високим споживанням пам'яті не повинні призначатися лише за критерієм кількості вільних процесорних ядер.

Після завершення ініціалізації агент переходить до стану готовності. Якщо службове з'єднання втрачено, керувальний компонент повинен відобразити невизначений стан вузла до отримання додаткового підтвердження.

Результатом алгоритму є зареєстрований виконавець із перевіреними характеристиками та дозволом на отримання завдань.

Алгоритм формування черги розподілених завдань

Для організації виконання обчислювальних операцій використовується черга завдань. Її структура повинна враховувати залежності між операціями та поточний стан ресурсів.

Вхідними даними є граф обчислень, характеристики завдань і відомості про вже завершені операції.

На першому етапі визначаються завдання, усі попередні залежності яких виконані. Такі операції можуть переходити до стану готовності.

Завдання з незавершеними залежностями залишаються заблокованими. Вони не повинні передаватися виконавцям, оскільки необхідні вхідні дані ще не підготовлені. Після завершення окремого завдання система оновлює стан пов'язаних вершин графа. Якщо всі необхідні результати доступні, наступна операція додається до черги готових завдань.

Для впорядкування може використовуватися пріоритет, визначений критичністю операції або її положенням у загальному графі. Завдання, що належать до критичного шляху обчислення, можуть отримувати підвищену увагу.

Окремо необхідно контролювати повторне додавання. Одне завдання не повинно одночасно отримувати декілька незалежних призначень, якщо така поведінка не передбачена алгоритмом резервного виконання.

Результатом є актуальна черга операцій, готових до розподілення між обчислювальними вузлами.

Алгоритм оцінювання поточного навантаження вузлів

Адаптивне керування потребує актуальних характеристик обчислювального середовища. Для цього використовується періодичне отримання телеметрії від зареєстрованих агентів.

Вхідними даними є показники використання процесора, оперативної пам'яті, дискової підсистеми та мережевих ресурсів.

На першому етапі агент отримує локальні характеристики стану. Далі вони передаються керувальному компоненту з відповідною часовою позначкою.

Під час оброблення визначається актуальність повідомлення. Застарілі показники не повинні використовуватися як достовірне представлення поточного навантаження.

Для зменшення впливу короткочасних коливань можуть застосовуватися ковзні середні або експоненційне згладжування. При цьому важливо зберігати відомості про пікові значення, які можуть свідчити про тимчасове перевантаження.

Окремо контролюється кількість активних завдань. Навіть за невеликого поточного використання процесора вузол може бути непридатним до отримання нової операції через зарезервовану пам'ять або очікуваний обсяг введення-виведення.

Результатом алгоритму є актуалізований профіль навантаження кожного вузла, який використовується під час планування.

Алгоритм прогнозування тривалості виконання завдань

Для адаптивного розподілення обчислень необхідно оцінювати очікуваний час завершення операцій. Така оцінка дозволяє враховувати неоднакову продуктивність вузлів і змінний характер навантаження.

Вхідними даними є характеристики завдання, попередні результати виконання подібних операцій та поточний стан ресурсів.

На першому етапі формується набір ознак. До них можуть належати обсяг вхідних даних, оцінка обчислювальної складності, кількість доступних процесорних ядер і характеристики попередніх виконань.

Для початкового прогнозування може використовуватися статистична модель на основі середніх значень для відповідного класу завдань.

За наявності достатньої кількості спостережень можливе застосування регресійних алгоритмів машинного навчання. Наприклад, Random Forest Regressor

або Gradient Boosting можуть оцінювати залежності між параметрами завдання та фактичною тривалістю.

Під час навчання необхідно враховувати, що частина попередніх операцій могла бути перервана. Для таких записів повна тривалість успішного виконання невідома. Тому їх не слід безпосередньо використовувати як звичайні завершені спостереження.

Окремо контролюється невизначеність прогнозу. Якщо система не має достатніх даних про певний клас завдань, необхідно використовувати консервативну оцінку або відповідний резерв часу.

Результатом алгоритму є прогнозована тривалість виконання, яка передається до планувальника.

Алгоритм адаптивного розподілення завдань

Основним завданням планувальника є вибір виконавця, який забезпечує допустимі умови виконання конкретної операції з урахуванням доступних ресурсів.

Вхідними даними алгоритму є черга готових завдань, профілі навантаження вузлів, прогнозована тривалість обчислень і характеристики мережевих з'єднань.

На першому етапі формується перелік технічно придатних виконавців. Вузол відхиляється, якщо він не має достатнього обсягу пам'яті, необхідних програмних залежностей або перебуває в стані невизначеної доступності.

Далі для кожного допустимого кандидата оцінюється очікуваний час виконання. До нього може додаватися прогнозована затримка отримання вхідних даних.

Особливу увагу необхідно приділити локальності інформації. Якщо значний обсяг даних уже доступний на певному вузлі, передавання його іншому виконавцю може створювати додаткове мережеве навантаження.

Після обчислення оцінок формується впорядкований перелік кандидатів. Перевага надається ресурсу, який забезпечує прийнятне співвідношення між прогнозованою тривалістю та використанням доступних потужностей.

Водночас необхідно уникати систематичного призначення всіх операцій одному найпродуктивнішому вузлу. Для цього враховуються вже зарезервовані ресурси та прогнозоване навантаження від призначених, але ще не запущених завдань.

Якщо жодний кандидат не відповідає встановленим вимогам, операція залишається в черзі.

Результатом алгоритму є рішення про призначення завдання конкретному виконавцю або про відкладення його виконання.

Алгоритм врахування мережевої топології GENI

Розподілені обчислення можуть потребувати інтенсивного обміну між вузлами. Для експериментів, ресурси яких розміщені в різних агрегатах, мережеві характеристики можуть істотно впливати на загальний час виконання.

Вхідними даними алгоритму є логічна топологія експерименту, доступні характеристики каналів і розміщення обчислювальних вузлів.

На початковому етапі визначаються зв'язки між виконавцями. Далі аналізується очікуваний обсяг передавання для кожної залежності між завданнями.

Для оцінювання мережевих витрат можуть використовуватися виміряні затримки та пропускна здатність відповідних з'єднань. Якщо такі показники недоступні, результат повинен позначатися як приблизний.

У GENI експериментальні мережеві топології могли використовувати з'єднання другого рівня моделі OSI. Водночас логічне виділення каналу не означало гарантованої постійної пропускної здатності.

Тому алгоритм не повинен припускати, що всі виділені з'єднання мають однакові характеристики. Для кожного маршруту необхідно враховувати доступні вимірювання.

Якщо два завдання інтенсивно обмінюються проміжними даними, доцільно розглядати їхнє розміщення на вузлах із меншою затримкою взаємодії.

Результатом алгоритму є додаткова характеристика мережевої доцільності розміщення, яка використовується під час адаптивного планування.

Алгоритм запуску обчислювального завдання

Після визначення виконавця необхідно забезпечити контрольований запуск обчислювального процесу.

Вхідними даними є ідентифікатор завдання, параметри виконання, адреси вхідних даних та відомості про вибраний вузол.

На першому етапі керувальний компонент перевіряє актуальність призначення. Далі створюється унікальна спроба виконання завдання. Вона дозволяє відокремлювати повторні запуски після відмов.

Після цього агент отримує параметри обчислення та перевіряє доступність необхідних даних. Якщо вхідна інформація відсутня або пошкоджена, завдання не повинно переходити до активного стану.

За успішної підготовки агент запускає відповідний процес. Для його виконання можуть використовуватися системні механізми запуску програм або спеціалізоване середовище ізоляції.

Після початку роботи агент передає підтвердження керувальному компоненту. У відповідному записі фіксуються час запуску, ідентифікатор виконавця та поточна спроба.

Якщо підтвердження не надійшло, система не повинна автоматично припускати, що процес не запущений. Необхідно перевірити фактичний стан спроби, щоб уникнути неконтрольованого дублювання.

Результатом алгоритму є запущене завдання з підтвердженим станом або повідомлення про неможливість його виконання.

Алгоритм моніторингу стану виконання

Під час обчислення система повинна контролювати не лише доступність вузла, а й фактичний стан прикладного процесу.

Вхідними даними є повідомлення агентів, відомості про активні завдання та поточні часові характеристики.

На першому етапі перевіряється надходження службових повідомлень про працездатність. Для кожного агента зберігається час останнього підтвердженого зв'язку.

Далі аналізуються повідомлення про стан конкретного завдання. Вони можуть містити ідентифікатор спроби, поточний етап роботи та доступні показники прогресу.

Для завдань, які підтримують визначення відсотка завершення, відповідне значення може передаватися до керувального компонента. Однак не всі алгоритми дозволяють достовірно оцінити залишковий обсяг роботи.

Окремо контролюється тривалість виконання. Якщо вона значно перевищує очікувану, формується повідомлення про можливе уповільнення.

Таке відхилення не повинно автоматично трактуватися як відмова. Причиною може бути складніший набір вхідних даних, тимчасове мережеве перевантаження або помилка прогнозу.

Результатом алгоритму є актуальний стан активних обчислень із можливістю визначення підозрілих відхилень.

Алгоритм виявлення відмов обчислювальних вузлів

Відмовостійкість передбачає своєчасне визначення компонентів, які більше не можуть забезпечувати виконання призначених операцій.

Вхідними даними є службові повідомлення агентів, відповіді інфраструктурних інтерфейсів та характеристики поточних завдань.

На першому етапі визначається час останнього підтвердженого зв'язку з вузлом. Якщо він перевищує встановлений інтервал, ресурс переводиться до стану підозрюваної недоступності.

Далі можуть виконуватися додаткові перевірки. Наприклад, керувальний компонент перевіряє доступність службового інтерфейсу або отримує відомості про стан виділеного ресурсу від інфраструктури.

Важливо розрізняти фізичну відмову вузла та втрату мережевого зв'язку. Якщо агент не відповідає, але обчислення продовжується, необґрунтований повторний запуск може призвести до одночасного виконання декількох копій завдання.

Тому за неоднозначного стану необхідно застосовувати механізм контрольованого підтвердження відмови або передачі права виконання новій спробі.

Окремо перевіряється строк дії виділеного ресурсу. Завершення часу резервування може призвести до недоступності вузла без його технічної несправності.

Результатом алгоритму є класифікований стан ресурсу з визначенням можливості подальшого виконання завдань.

Алгоритм збереження контрольних точок обчислення

Для зменшення втрат виконаної роботи доцільно передбачити механізм контрольних точок (Checkpointing). Його призначення полягає у збереженні проміжного стану, необхідного для продовження обчислення після відмови.

Вхідними даними є поточний стан завдання, його ідентифікатор та параметри зберігання.

На першому етапі визначається можливість створення контрольної точки. Не всі програми підтримують відновлення з довільного моменту. Для таких випадків необхідно реалізувати прикладне збереження стану або використовувати повторне виконання з початку. Якщо контрольні точки підтримуються, процес формує узгоджене представлення свого стану. Воно може містити проміжні результати, поточну позицію оброблення та іншу необхідну інформацію.

Після підготовки дані передаються до сховища, доступного незалежно від обчислювального вузла. Збереження лише на локальному диску тимчасового ресурсу не забезпечує відновлення після його втрати.

Наступним етапом є перевірка записаних даних. Для цього може використовуватися криптографічний контроль цілісності.

Після успішного підтвердження створюється запис метаданих контрольної точки. Він містить ідентифікатор завдання, номер спроби, версію стану та місце зберігання.

Результатом алгоритму є підтверджена контрольна точка, придатна для подальшого відновлення.

Алгоритм визначення періодичності контрольних точок

Надмірно часте збереження стану створює додаткове навантаження на дискову й мережеву підсистеми. Занадто великі інтервали призводять до втрати значного обсягу виконаної роботи у випадку відмови.

Для визначення доцільної періодичності необхідно враховувати витрати на створення контрольної точки та характеристики надійності вузлів.

Вхідними даними алгоритму є тривалість збереження стану, статистика попередніх відмов і очікувана тривалість обчислення.

На першому етапі визначається час, необхідний для формування та передавання контрольної точки. Далі оцінюється очікувана втрата обчислювальної роботи за різних інтервалів збереження.

Для початкового планування можуть використовуватися відомі аналітичні підходи до оптимізації періодичного Checkpointing, зокрема модель Young та її подальші уточнення.

За умов значного навантаження на сховище інтервал між контрольними точками може збільшуватися. Якщо ризик відмови або завершення доступного часу ресурсу зростає, доцільно розглядати частіше збереження.

Водночас така адаптація повинна враховувати мінімальний час, необхідний для завершення самої операції записування.

Результатом алгоритму є рекомендована періодичність створення контрольних точок, яка використовується компонентом керування виконанням.

Алгоритм відновлення завдання після відмови

Після підтвердження неможливості продовження обчислення на попередньому вузлі необхідно визначити спосіб відновлення.

Вхідними даними є стан завдання, відомості про останню підтверджену контрольну точку та перелік доступних виконавців.

На першому етапі перевіряється завершеність попередньої спроби. Якщо її стан невизначений, система повинна обмежити можливість одночасного записування результатів різними виконавцями.

Далі визначається наявність придатної контрольної точки. Якщо вона доступна та відповідає версії обчислювального завдання, формується запит на відновлення.

Для вибору нового виконавця повторно застосовується алгоритм адаптивного планування. При цьому вузол, на якому сталася відмова, тимчасово виключається з переліку кандидатів до уточнення його стану.

Після призначення нового ресурсу агент отримує контрольну точку та перевіряє її цілісність. Далі виконується відновлення прикладного процесу.

Якщо контрольна точка відсутня або несумісна з поточною версією програми, завдання може виконуватися повторно з початку за умови допустимості такої операції.

Для завдань із зовнішніми побічними ефектами необхідно передбачити окремі гарантії повторного виконання. Наприклад, повторний запис до зовнішньої бази даних може призвести до дублювання результатів.

Результатом алгоритму є нова контрольована спроба виконання або повідомлення про неможливість автоматичного відновлення.

Алгоритм запобігання конфліктам повторних запусків

Під час розподілених обчислень неможливо повністю виключити ситуацію, коли керувальний компонент втрачає зв'язок із виконавцем, але той продовжує виконувати завдання.

Тому необхідно забезпечити контроль права на публікацію результатів.

Вхідними даними алгоритму є ідентифікатор завдання, номер спроби та актуальна версія керувального стану.

На початковому етапі кожній новій спробі надається послідовно зростаючий номер покоління. Він використовується для визначення актуального виконавця.

Після призначення нової спроби попередня втрачає право підтверджувати завершення завдання. Навіть якщо застарілий вузол продовжить обчислення, його результат не повинен автоматично перезаписувати актуальні дані.

Для цього сховище службового стану повинно підтримувати умовне оновлення записів. Під час підтвердження перевіряється збіг номера покоління з поточним значенням.

Такий підхід відповідає принципу Fencing Tokens. Його застосування дозволяє зменшувати ризик конфліктів між застарілими та новими виконавцями.

Водночас для зовнішніх систем, які не перевіряють відповідні токени, необхідні додаткові механізми ідемпотентності або транзакційного захисту.

Результатом алгоритму є контрольоване підтвердження результатів лише від актуальної спроби виконання.

Алгоритм повторного розподілення навантаження

Адаптивне управління передбачає можливість перегляду призначення завдань у відповідь на зміну стану ресурсів.

Вхідними даними алгоритму є поточне завантаження вузлів, черга готових операцій та результати прогнозування тривалості.

На першому етапі визначаються ресурси, які мають значне перевантаження. Далі аналізується можливість передавання ще не запущених завдань іншим виконавцям.

Для активних процесів перенесення можливе лише за наявності відповідного механізму збереження та відновлення стану. Без такої підтримки примусове завершення обчислення може призвести до втрати виконаної роботи.

Наступним етапом є оцінювання очікуваної користі перенесення. Враховуються час створення контрольної точки, передавання даних, повторної ініціалізації та прогнозована тривалість завершення на новому вузлі.

Якщо витрати на перенесення перевищують очікуваний виграш, доцільно залишити завдання на поточному виконавці.

Окремо передбачається гістерезис. Він дозволяє уникати багаторазового переміщення завдань за незначних коливань навантаження.

Результатом алгоритму є оновлений план виконання, який не порушує цілісність активних обчислень.

Приклад програмної реалізації адаптивного планувальника

Для пояснення принципу вибору обчислювального вузла може використовуватися програмний алгоритм мовою Python. Наведений фрагмент демонструє оцінювання придатності виконавців за ресурсними характеристиками, доступністю та прогнозованою тривалістю виконання. Він не реалізує безпосереднє звернення до GENI Aggregate Manager API та є ілюстративною моделлю роботи планувальника.

```
from dataclasses import dataclass
from math import isfinite


@dataclass(frozen=True)
class ComputeNode:
    # Характеристики доступного обчислювального вузла
    node_id: str
    free_cpu: int
    free_memory_mb: int
    cpu_load: float
    network_delay_ms: float
    failure_risk: float
    ready: bool


@dataclass(frozen=True)
class ComputeTask:
    # Мінімальні вимоги та прогнозована тривалість
    task_id: str
    required_cpu: int
    required_memory_mb: int
    estimated_seconds: float
    priority: int


def select_compute_node(
    task: ComputeTask,
    nodes: list[ComputeNode]
) -> dict:
    """Вибір виконавця для розподіленого завдання."""

    # Перевіряємо коректність ресурсних вимог
    if (
        task.required_cpu <= 0
        or task.required_memory_mb <= 0
        or not isfinite(task.estimated_seconds)
        or task.estimated_seconds <= 0
    ):
        return {
            "status": "INVALID_TASK",
            "task_id": task.task_id
        }

    candidates = []

    for node in nodes:
        # Недоступні вузли не використовуємо
        if not node.ready:
```

```
            continue

        # Перевіряємо достатність процесора та пам'яті
        if node.free_cpu < task.required_cpu:
            continue

        if node.free_memory_mb < task.required_memory_mb:
            continue

        values = (
            node.cpu_load,
            node.network_delay_ms,
            node.failure_risk
        )

        # Відхиляємо некоректні або неповні вимірювання
        if not all(isfinite(value) for value in values):
            continue

        if not 0 <= node.cpu_load <= 1:
            continue

        if not 0 <= node.failure_risk <= 1:
            continue

        if node.network_delay_ms < 0:
            continue

        # Демонстраційна оцінка тривалості з урахуванням навантаження
        load_factor = 1.0 + node.cpu_load

        predicted_time = (
            task.estimated_seconds * load_factor
        )

        # Ураховуємо мережеву затримку в умовній оцінці
        predicted_time += node.network_delay_ms / 1000.0

        # Вищий ризик відмови збільшує оцінку витрат
        score = predicted_time * (1.0 + node.failure_risk)

        candidates.append((score, node.node_id))

    # Якщо придатних ресурсів немає, зберігаємо завдання в черзі
    if not candidates:
        return {
            "status": "WAITING_RESOURCES",
            "task_id": task.task_id
        }

    # Вибираємо вузол із найменшою умовною оцінкою
    candidates.sort()
    score, selected_node = candidates[0]

    return {
        "status": "PROPOSED",
        "task_id": task.task_id,
        "node_id": selected_node,
        "estimated_score": score
    }
```

У наведеному прикладі система послідовно аналізує доступних виконавців. Спочатку відхиляються вузли, які не відповідають мінімальним

ресурсним вимогам. Далі перевіряється коректність телеметрії. Для придатних ресурсів формується умовна оцінка, яка враховує прогнозовану тривалість, поточне навантаження та ризик відмови.

Отримані оцінки використовуються для визначення пріоритетного виконавця. Якщо жодний вузол не відповідає встановленим вимогам, завдання залишається в черзі очікування.

Слід враховувати, що показник failure_risk у цьому прикладі є вхідним параметром. Він не обчислюється автоматично з телеметрії. Для практичного застосування необхідно окремо розробити процедуру його оцінювання на основі історичних спостережень.

Також модель не враховує фактичний обсяг передавання вхідних даних і можливість конкурентного резервування ресурсів. Тому в повній реалізації вибір виконавця повинен супроводжуватися атомарним підтвердженням призначення та перевіркою актуальності стану вузла.

Алгоритм керування станами обчислювальних завдань

Для забезпечення узгодженості роботи всіх компонентів необхідно використовувати визначену модель життєвого циклу завдання.

Після реєстрації операція перебуває в стані очікування залежностей. Коли необхідні вхідні дані доступні, вона переходить до черги готових завдань.

Після вибору виконавця формується стан призначення. Далі агент підтверджує отримання операції та початок виконання.

Під час обчислення система може отримувати повідомлення про проміжний прогрес і створені контрольні точки. Вони не змінюють завершеність завдання, але використовуються для відновлення.

За успішного завершення агент передає відомості про результат. Керувальний компонент перевіряє його цілісність і відповідність актуальній спробі.

Лише після підтвердження результату завдання переходить до завершеного стану. Після цього можуть активуватися залежні операції.

У разі помилки формується стан відмови. Він використовується для визначення необхідності повторного виконання або завершення експерименту з помилкою.

Результатом алгоритму є узгоджена послідовність станів, яка запобігає передчасному завершенню та некоректному повторному виконанню операцій.

Алгоритм зберігання службових метаданих

Для підтримання відмовостійкості необхідно забезпечити збереження інформації про експерименти незалежно від окремих обчислювальних вузлів.

Вхідними даними алгоритму є стани завдань, результати призначення, відомості про контрольні точки та записи моніторингу.

На першому етапі перевіряється структура службового повідомлення. Далі визначається запис, який потрібно створити або оновити.

Для зберігання може використовуватися PostgreSQL або інша СКБД із підтримкою транзакцій. Важливо забезпечити атомарність критичних змін стану.

Наприклад, підтвердження завершення завдання та активація залежних операцій повинні виконуватися так, щоб після відмови не виникав суперечливий стан графа обчислень.

Окремо зберігаються ідентифікатори спроб виконання. Це дозволяє встановлювати зв'язок між повторними запусками та фактичними результатами.

Для підвищення відмовостійкості доцільно передбачити резервне копіювання службового сховища. Якщо метадані розміщені на одному незахищеному вузлі, його відмова може призвести до втрати можливості відновлення всього експерименту.

Результатом алгоритму є узгоджена історія виконання та актуальний стан розподілених обчислень.

Алгоритм забезпечення відмовостійкості керувального компонента

Централізований планувальник спрощує координацію завдань, але може створювати єдину точку відмови. Тому необхідно передбачити механізм відновлення його роботи.

Вхідними даними алгоритму є стан керувального сервера, доступність службового сховища та відомості про активні завдання.

На початковому етапі перевіряється працездатність основного керувального компонента. Якщо він недоступний, необхідно визначити можливість його перезапуску.

У разі використання резервного екземпляра важливо запобігти одночасному активному керуванню декількома незалежними планувальниками.

Для цього може застосовуватися механізм вибору активного керувального вузла через спільне сховище стану або відповідний сервіс координації.

Після отримання права керування новий компонент завантажує актуальні записи завдань і перевіряє їхній стан.

Особливу увагу необхідно приділити активним обчисленням. Вони можуть продовжувати виконуватися під час недоступності керувального сервера. Тому після відновлення спочатку виконується узгодження стану з агентами, а не автоматичний повторний запуск усіх незавершених операцій.

Результатом алгоритму є можливість відновлення координування обчислень без необґрунтованого дублювання активних процесів.

Алгоритм завершення експерименту та звільнення ресурсів

Після завершення обчислювальних завдань система повинна забезпечити збереження результатів і коректне завершення використання експериментальної інфраструктури.

Вхідними даними є стан графа обчислень, перелік активних ресурсів і відомості про збережені результати.

На першому етапі перевіряється завершеність усіх обов'язкових операцій. Якщо частина завдань перебуває в стані помилки, формується відповідний підсумковий статус.

Далі контролюється передавання результатів до постійного сховища. Для цього можуть використовуватися контрольні значення, які дозволяють перевіряти цілісність отриманих файлів.

Після підтвердження збереження даних виконується зупинка прикладних агентів. Окремо перевіряється відсутність незавершених операцій передавання.

Для історичної архітектури GENI звільнення ресурсів здійснювалося через відповідні керувальні операції агрегатів. У сучасній реалізації необхідно використовувати еквівалентний механізм фактично доступної інфраструктури.

Результатом алгоритму є завершений експеримент зі збереженими результатами та звільненими тимчасовими ресурсами.

Особливості програмної реалізації системи

Для реалізації керувального компонента як можливе технологічне рішення може використовуватися Python. Він дозволяє організовувати асинхронну взаємодію, оброблення черг, виконання алгоритмів планування та аналіз телеметрії.

Для програмної взаємодії з історичними інтерфейсами GENI може розглядатися бібліотека geni-lib, яка підтримувала роботу з RSpec та відповідними компонентами федеративної інфраструктури.

Оскільки GENI більше не функціонує як доступна дослідницька платформа, практична реалізація повинна передбачати відокремлення алгоритмів планування від конкретного інфраструктурного API.

Для цього доцільно використовувати адаптер ресурсного середовища. Він надає керувальному компоненту операції отримання переліку ресурсів, резервування вузлів, перевірки стану та звільнення.

У варіанті історичного моделювання відповідні операції можуть реалізовуватися через емулятор GENI Aggregate Manager API. Для експериментальної перевірки на сучасній платформі може використовуватися інший адаптер, зокрема до FABRIC, якщо доступна відповідна інфраструктура та дозволи.

Алгоритми керування завданнями повинні залишатися незалежними від способу виділення ресурсів. Це дозволяє перевіряти їхню логіку в локальному середовищі та зменшує залежність програмного комплексу від конкретної дослідницької платформи.

Для збору телеметрії можуть використовуватися програмні агенти, які передають доступні системні показники. Для їхнього накопичення та візуалізації можливе застосування Prometheus і Grafana.

Конкретні засоби програмної реалізації повинні визначатися фактично розробленою архітектурою. Наведені технології є технічно обґрунтованими варіантами, а не підтвердженим переліком встановлених компонентів.

Алгоритм оброблення помилок та виняткових ситуацій

Під час роботи розподіленого комплексу можливі некоректні описи експериментів, відмови мережевих каналів, недоступність вузлів, помилки виконання обчислень і порушення збереження результатів.

Для кожного випадку необхідно визначити категорію помилки та допустиму реакцію.

На першому етапі перевіряється можливість повторного виконання операції. Тимчасові помилки мережевої взаємодії можуть оброблятися обмеженою кількістю повторних спроб.

Для зменшення навантаження на недоступний компонент доцільно застосовувати збільшення інтервалу між повторними зверненнями.

Якщо помилка пов'язана з некоректними вхідними даними, повторне виконання без їхнього виправлення зазвичай не має практичного значення. У такому випадку завдання переводиться до стану помилки з відповідним поясненням.

Окремо обробляються відмови, що виникають під час записування результатів. Якщо підтвердження не отримано, необхідно перевірити фактичну наявність об'єкта перед повторним записуванням.

За втрати зв'язку з виконавцем застосовується процедура уточнення його стану. Автоматичне повторне виконання допускається лише після перевірки можливості конфлікту з попередньою спробою.

Результатом алгоритму є контрольована реакція на помилки з можливістю подальшого відновлення відповідно до характеру порушення.

Забезпечення інформаційної безпеки розподілених обчислень

Керування ресурсами та виконання обчислювальних завдань потребують розмежування повноважень. Особливе значення має захист службових інтерфейсів і даних, які передаються між компонентами.

У GENI для доступу до федеративної інфраструктури використовувалися механізми облікових записів, сертифікатів та відповідних повноважень експериментатора. У програмній реалізації необхідно враховувати аналогічні вимоги до авторизації керувальних операцій.

Для взаємодії між агентами та керувальним сервером доцільно використовувати захищені канали. При цьому необхідно перевіряти автентичність обох сторін.

Окремо забезпечується захист службових облікових даних. Вони не повинні включатися безпосередньо до вихідного коду або доступних усім користувачам конфігураційних файлів.

Під час передавання контрольних точок і результатів обчислень доцільно застосовувати перевірку цілісності та відповідні механізми розмежування доступу.

Для зменшення впливу скомпрометованого агента необхідно обмежувати його повноваження конкретним експериментом і набором призначених завдань.

Результатом застосування відповідних механізмів є контрольований доступ до розподілених ресурсів і службових даних програмного комплексу.

Перевірка коректності алгоритмів та оцінювання ефективності

Для експериментального дослідження запропонованого алгоритмічного забезпечення необхідно передбачити перевірку правильності планування, виявлення відмов, збереження контрольних точок та відновлення обчислень.

Початковим етапом є перевірка реєстрації експериментів. Для цього формуються тестові графи завдань із відомими залежностями. Необхідно підтвердити правильність визначення готових операцій та неможливість передчасного запуску залежних завдань.

Далі перевіряється механізм розподілення ресурсів. Для цього можуть використовуватися синтетичні характеристики вузлів із різними показниками процесорної продуктивності, доступної пам'яті та мережевої затримки.

Під час перевірки планувальника необхідно підтвердити, що завдання не призначаються ресурсам, які не відповідають мінімальним технічним вимогам.

Окремо оцінюється адаптивність. Для цього створюються сценарії зі змінним навантаженням, у яких доступність вузлів і характеристики продуктивності змінюються протягом виконання експерименту.

Для порівняння можуть використовуватися статичне розподілення завдань, алгоритм Round Robin та запропонований адаптивний планувальник. Випробування повинні виконуватися за порівнянних умов.

Основним показником ефективності може бути загальний час завершення графа обчислень (Makespan). Додатково визначаються середній час очікування завдання, використання процесорних ресурсів і кількість перепризначень.

Для перевірки алгоритму прогнозування тривалості можуть використовуватися Mean Absolute Error та Root Mean Squared Error. Вони дозволяють оцінювати різницю між прогнозованою та фактичною тривалістю завершених операцій.

Особливу увагу необхідно приділити відмовостійкості. Для цього моделюються зупинка обчислювального агента, втрата мережевого зв'язку та недоступність керувального компонента.

Під час таких випробувань перевіряється правильність визначення стану вузла та можливість відновлення незавершених завдань.

Окремо перевіряється використання контрольних точок. Для цього тривале тестове обчислення переривається після збереження проміжного стану. Подальше відновлення повинно забезпечувати правильне продовження роботи.

Для визначення користі контрольних точок можуть вимірюватися обсяг повторно виконаної роботи, час відновлення та додаткові витрати на збереження стану.

Для порівняння відмовостійких стратегій доцільно використовувати сценарії повного повторного запуску та відновлення з останньої контрольної точки.

Важливим показником є частка завдань, які завершилися успішно після введення контрольованих відмов. При цьому необхідно враховувати кількість і тип збоїв, щоб забезпечити коректність порівняння.

Додатково досліджується стійкість до повторних запусків. Для цього можуть моделюватися затримки службових повідомлень та ситуації, коли попередній виконавець продовжує роботу після втрати зв'язку.

Очікуваним результатом є відсутність конфліктного підтвердження завершення від застарілої спроби. Для цього перевіряється правильність контролю поколінь і умовного оновлення стану.

Для оцінювання масштабованості можуть використовуватися експерименти зі збільшенням кількості завдань і виконавців. Під час перевірки вимірюються затримка планування, інтенсивність службових повідомлень та завантаження керувального сервера.

Окремо необхідно оцінити витрати на адаптивне керування. Якщо додатковий аналіз ресурсів потребує значного часу, його застосування може бути недоцільним для дуже коротких обчислювальних операцій.

Практична перевірка повинна виконуватися в доступному експериментальному середовищі. Оскільки GENI припинила роботу, використання її історичних інтерфейсів доцільно перевіряти через емулятор або окремий сумісний адаптер. За наявності доступу до FABRIC може бути досліджена можливість перенесення відповідних алгоритмів без зміни їхньої основної логіки.

Кількісні висновки щодо скорочення часу виконання, покращення використання ресурсів і підвищення стійкості до відмов можуть бути сформовані лише після проведення відповідних експериментів.

Розглянуте алгоритмічне забезпечення формує основу системи адаптивного управління розподіленими обчисленнями в архітектурі хмарної

тестової платформи GENI. Його функціонування передбачає послідовне керування ресурсами, планування завдань, моніторинг стану виконавців і контрольоване відновлення після відмов. Основне значення мають узгодженість службового стану, збереження контрольних точок і запобігання конфліктам повторних запусків. Відокремлення алгоритмів планування від інфраструктурних інтерфейсів створює можливість їхнього дослідження в сумісному експериментальному середовищі та подальшої адаптації до доступних платформ розподілених обчислень.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмне забезпечення системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI призначене для автоматизованого розподілення завдань між обчислювальними вузлами, контролю їхньої працездатності та відновлення обчислень після виникнення відмов.

Система забезпечує керування експериментальними ресурсами, моніторинг навантаження, адаптивне планування та збереження проміжних результатів.

Архітектура програмного комплексу орієнтована на принципи функціонування хмарної тестової платформи GENI (Global Environment for Network Innovations). Головне вікно програми показане на рисунку 5.1.

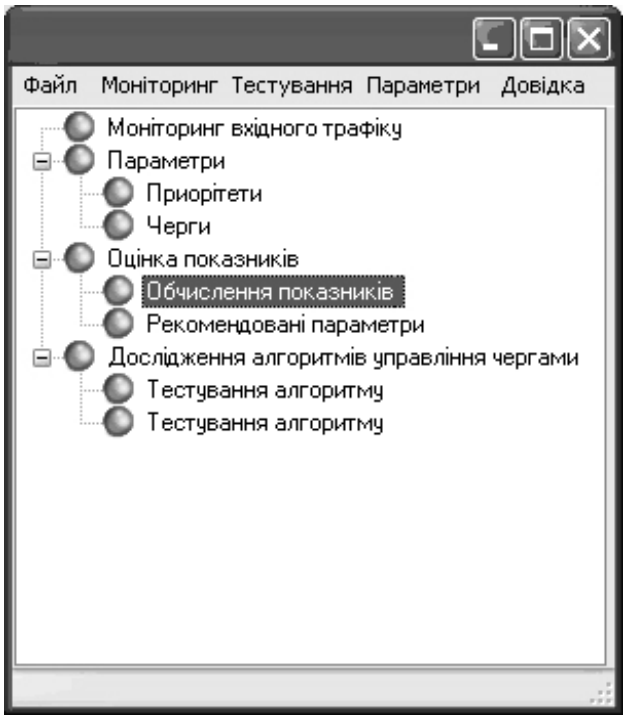


Рисунок 5.1 – Головне вікно розробленого ПЗ

Практичне впровадження системи передбачає підготовку керувального сервера, налаштування обчислювальних вузлів, встановлення програмних залежностей та організацію взаємодії між компонентами. Основними функціональними складовими є планувальник завдань, модуль керування ресурсами, агенти виконавців, підсистема моніторингу та компонент відновлення обчислень. Для збереження службової інформації передбачається окреме сховище станів завдань і метаданих контрольних точок.

Під час розгортання необхідно враховувати особливості платформи GENI. Її офіційна експлуатація завершилася 1 серпня 2023 року. Тому безпосереднє підключення до колишньої федеративної інфраструктури не може розглядатися як актуальний спосіб впровадження. Для практичної перевірки програмного комплексу доцільно використовувати емулятор ресурсного середовища GENI або адаптер до доступної дослідницької інфраструктури. Одним із можливих варіантів є FABRIC, яка підтримує експериментальні обчислювальні та мережеві ресурси, але використовує власні програмні інтерфейси.

Як можливе середовище реалізації алгоритмів планування та оброблення телеметрії розглядається Python. Його засоби дозволяють організовувати асинхронну взаємодію, виконувати обчислювальні процедури та керувати станами завдань. Конкретні версії операційної системи, інтерпретатора й програмних бібліотек визначаються фактичною реалізацією. Для ізоляції залежностей доцільно використовувати окреме програмне середовище.

Апаратні вимоги залежать від кількості одночасних обчислювальних завдань і складності їхнього виконання. Керувальний сервер повинен мати достатні ресурси для оброблення службових повідомлень, планування та зберігання поточного стану. Обчислювальні вузли повинні відповідати вимогам конкретних завдань до процесора, оперативної пам'яті та дискового простору. Для експериментів із великим обсягом проміжних даних необхідно додатково враховувати пропускну здатність мережевих каналів.

Підготовка ресурсного середовища починається з визначення топології експерименту. В історичній архітектурі GENI логічна область експерименту

позначалася як Slice, а виділені ресурси як Sliver. Опис необхідних вузлів і мережевих з'єднань формувався за допомогою Resource Specification (RSpec). Взаємодія з інфраструктурними агрегатами здійснювалася через Aggregate Manager API. Для локального моделювання необхідно забезпечити відповідне представлення цих об'єктів або реалізувати їхнє зіставлення з ресурсами іншої платформи.

Окремим етапом є налаштування адаптера інфраструктури. Він повинен забезпечувати отримання відомостей про доступні ресурси, створення експериментального середовища, перевірку стану вузлів та завершення їхнього використання. Для емулятора GENI можуть використовуватися відповідні моделі операцій Aggregate Manager API. Під час інтеграції з FABRIC потрібний окремий адаптер до її підтримуваних інтерфейсів. Це дозволяє зберігати незалежність алгоритмів планування від конкретної дослідницької платформи.

Після підготовки інфраструктури налаштовуються обчислювальні агенти. Для кожного вузла визначаються адреса керувального сервера, службові параметри підключення та доступні обчислювальні ресурси. Агент повинен мати можливість отримувати завдання, запускати відповідні процеси та передавати відомості про їхній стан. Права доступу необхідно обмежувати операціями, які потрібні для виконання визначеного експерименту.

Для зберігання службових метаданих як можливий варіант може використовуватися PostgreSQL. У відповідному сховищі передбачається реєстрація експериментів, станів завдань, ідентифікаторів виконавців і повторних спроб. Окремо зберігаються відомості про контрольні точки. Необхідно забезпечити транзакційне оновлення критичних записів, щоб після перезапуску керувального компонента не виникали суперечливі стани обчислень.

Наступним етапом є підготовка системи моніторингу. Для отримання показників завантаження процесора, оперативної пам'яті та мережевих ресурсів можуть використовуватися програмні агенти. Як можливі засоби накопичення й візуалізації телеметрії розглядаються Prometheus і Grafana. Перед запуском планувальника необхідно перевірити актуальність показників та правильність

зіставлення телеметрії з ідентифікаторами виконавців.

Для забезпечення відмовостійкості налаштовується механізм створення контрольних точок. Програми, які підтримують збереження проміжного стану, повинні передавати відповідні дані до сховища, доступного незалежно від обчислювального вузла. Для кожної контрольної точки зберігаються ідентифікатор завдання, номер спроби та відомості про цілісність. Якщо прикладна програма не підтримує відновлення стану, необхідно передбачити повторне виконання з початку.

Після цього визначаються параметри планувальника. До них належать допустима кількість одночасних завдань, мінімальні ресурсні вимоги, періодичність оновлення телеметрії та обмеження повторних спроб. Окремо встановлюються інтервали контролю працездатності агентів. Для запобігання конфліктам після відмов необхідно використовувати ідентифікатори спроб і механізм підтвердження права на публікацію результатів.

Рекомендована послідовність запуску починається з перевірки доступності службового сховища та ресурсного середовища. Далі запускається керувальний сервер і активується адаптер інфраструктури. Після цього ініціалізується моніторинг та реєструються обчислювальні агенти. Наступним етапом є запуск планувальника й компонента відновлення. Завершується підготовка перевіркою доступності користувацького інтерфейсу та можливості створення нового обчислювального експерименту.

Первинне тестування доцільно виконувати на невеликій експериментальній топології. Для цього створюється декілька обчислювальних вузлів і набір незалежних завдань із визначеними ресурсними вимогами. Після запуску перевіряється правильність реєстрації виконавців, розміщення операцій у черзі та їхнього призначення придатним ресурсам. Результати завершених завдань повинні зберігатися разом із відповідними службовими записами.

Для перевірки адаптивного планування використовується сценарій зі змінним навантаженням. У процесі його виконання контролюється оновлення характеристик вузлів і можливість перегляду призначення нових завдань.

Система повинна враховувати доступні ресурси та не передавати операції виконавцям, які не відповідають мінімальним вимогам. Зміна навантаження не повинна спричиняти неконтрольованого перенесення активних процесів.

Окремо перевіряється механізм відновлення після відмови. Для цього в лабораторному середовищі може моделюватися зупинка одного обчислювального агента. Після виявлення порушення необхідно перевірити формування відповідного стану завдання та вибір нового виконавця. Якщо існує придатна контрольна точка, відновлення повинно виконуватися з її використанням. За відсутності збереженого стану застосовується дозволена процедура повторного запуску.

Під час експлуатації можливі втрата зв'язку з агентом, помилки резервування ресурсів, недоступність службового сховища та пошкодження контрольних точок. Для діагностики використовуються журнали керувального компонента, записи виконання завдань і телеметрія вузлів. Якщо стан попередньої спроби залишається невизначеним, нове виконання не повинно автоматично отримувати право перезаписування результатів. Після усунення несправності здійснюється узгодження фактичного стану виконавців зі службовими метаданими.

Перед переходом до регулярної експлуатації необхідно перевірити збереження конфігурації після перезапуску, доступність резервних копій службових метаданих і правильність відновлення обчислювальних завдань. Додатково контролюються захист службових з'єднань, розмежування повноважень і можливість завершення експерименту зі звільненням тимчасових ресурсів. Оцінка експлуатаційної готовності повинна ґрунтуватися на результатах фактично виконаних перевірок у вибраному середовищі.

Для документування розробленого програмного забезпечення передбачається представлення відомостей про автора. Вони використовуються для ідентифікації розробника, встановлення походження програмного продукту та організації його подальшого технічного супроводження. Інформація про автора системи адаптивного управління та забезпечення відмовостійкості

розподілених обчислень представлена на рисунку 5.2.

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

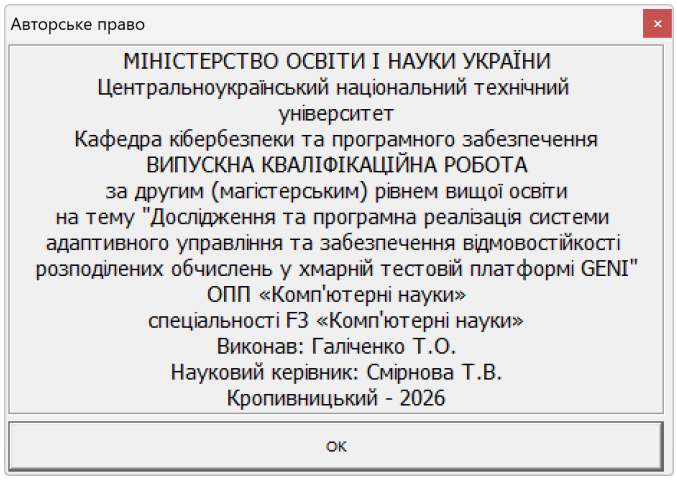


Рисунок 5.2 – Вікно розробника ПЗ

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Метою розробки є дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Об'єктом дослідження є процес адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Предметом дослідження є методи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Методи дослідження базуються на методах кодування інформації, методах хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

– Розроблено вітчизняний продукт адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

7.1 Вступ

Незважаючи на стрімке розповсюдження комп'ютерних мереж, з’ємні носії інформації не втрачають актуальності у ряді сфер професійної діяльності.

Стрімкий розвиток галузі інформаційних технологій (ІТ) не тільки у сфері управління виробництвом, в банківській систему, бізнесі, системі освіти, але також на транспорті, сфері обслуговування призвів до того, що десятки мільйонів людей у всьому світі виявились втягнутими у взаємодію людини з комп'ютером. Природно виникає запитання: настільки безпечною є ця взаємодія для людини? Адже відома аксіома про те, що будь-яка взаємодія людини та засобів праці двостороння.

Впровадження комп'ютерних технологій принципово змінило характер праці різних категорій фахівців. Працівники, використовують комп'ютерну техніку, на своєму досвіді оцінили її величезні можливості. Одночасно виникла певна безтурботність при її експлуатації.

Недотримання вимог безпеки призводить до того, що й через кілька днів роботи за комп'ютером співробітник починає відчувати певний дискомфорт: в нього виникає головний біль і різь у власних очах, з'являються почуття виснаження й дратівливість. В окремих людей порушується сон, погіршується зір, занедужують руки, шия, поперек тощо.

До недоліків умов праці користувачів комп’ютерної техніки можна віднести:

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог, мікроклімату на робочих місцях;

– низький рівень освітленості у приміщеннях і на робочих поверхнях апаратури;

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимам праці та відпочинку;

– надмірне виробничу навантаження працівників;

– відсутність навичок зниження впливу психоемоційного напруги.

Відповідно до ст.14 Закону «Про охорони праці» [1] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування коштів індивідуальної захисту працівників; відповідні вимоги охорони праці, умови праці в кожному робоче місце; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці.

7.2 Аналіз умов праці

Приміщення розташовано на третьому поверсі п'ятиповерхового будинку. У приміщенні розташовано 3 робочих місць з комп'ютерами (далі ПК). Відповідно до норм «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2] площа, що відводиться для робочого місця з комп'ютером повинна бути не менше 6 м^2, об'єм не менше 20 м^3. Розміри даного приміщень складають: довжина – 6 м, ширина – 4,5 м, висота – 3,5 м, тобто загальна фактична площа складає 27 м^2. Необхідна площа на 3 робочих місця із установленими ПК складає 18 м^2, що не перевищує фактичну. Обсяг кабінету на одного працюючого складає 31,5м^3, отже відповідає нормі ДСанПіН 3.3.2-007-98 – не менше 20 м^3 [2].

При роботі з ПК людина може піддатися впливу шкідливих та небезпечних факторів. Під шкідливими виробничими факторами розуміють фактори, тривалий вплив яких викликає розвиток професійних захворювань. Небезпечні виробничі фактори – вплив яких на працюючого викликає травму, тобто пошкодження організму. Шкідливі і небезпечні чинники, з якими стикається бібліограф при роботі з ПК, приведені в таблиці 7.1.

Таблиця 7.1 – Перелік шкідливих та небезпечних виробничих факторів

Найменування факторів	Можливі джерела їх виникнення	Характер дії
Небезпека ураження електричним струмом	Мережа живлення	Небезпечний
Пожежонебезпечність приміщень	Наявність матеріалів, що згорають і джерел запалення (електроапаратура)	Небезпечний та шкідливий
Іонізація повітря	Статична електрика випромінювання	Шкідливий
Підвищений рівень шуму	Шум створюється перетворювачем напруги ЕОМ, її технічною периферією, а також людьми, що працюють в приміщенні	Шкідливий
Несприятлива освітленість	Недостатнє штучне і природне освітлення	Шкідливий
Незадовільні параметри мікроклімату	Незадовільний стан системи опалення і вентиляції	Шкідливий
Психофізіологічні напруження	Монотонність праці, перенапруженість зорових аналізаторів, розумова напруженість, незручність і статичність пози	Шкідливий

По категорії вибухо – і пожежонебезпеки, згідно дане приміщення відноситься до категорії В – пожежонебезпечне, тому що присутні тверді матеріали, що горять, такі як дерев'яні столи, папір і інше. Виходячи з категорії пожежонебезпеки і поверховості будинку, ступінь вогнестійкості будівлі II. Згідно з ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» [13] ЕОМ повинні розташовуватись в будівлі не менше ніж II ступню вогнестійкості.

По ступені небезпеки поразки людей електричним струмом відділ, згідно, класифікується як приміщення з підвищеною небезпекою, тому що не виключена можливість одночасного дотику людини до маючих з'єднання з землею конструкціям будинку, з одного боку, і до металевих корпусів електроустаткування, що можуть виявити під напругою – з іншого.

Для забезпечення вищевказаних оптимальних метеорологічних умов у помешканні передбачена система опалення (загальне парове) в холодному періоді, та вентиляція і кондиціювання в теплий період року, згідно ДБН2.5–67–2013 «Опалення, вентиляція та кондиціювання» [4]. При виконанні замірів параметрів мікроклімату, значення їх відповідали оптимальним та допустимим параметрам відповідно до ДСанПіН3.3.2.007–98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно – обчислювальних машин» [2].

Припустимий рівень іонізації повітря помешкання відповідно до СН 21.52-80 повинен складати 1500 – 3000 один./м^3.

Нормування освітлення здійснюється відповідно до ДБН В.2.5 – 28 – 2006 «Природне та штучне освітлення». [5]

Відділ забезпечений комбінованим освітленням. В темний час доби передбачається загальне і/або місцеве рівномірне штучне, а в світлий – бокове одностороннє природне освітлення два віконних прорізи.

7.3 Техніка безпеки та протипожежна профілактика

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» [13] будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2 м від рівня підлоги, її кріплення здійснюється за допомогою металевих власників. Біля кожного стола організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу входять п'ять розеток і дві клеми заземлення. Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 В. Про це є відповідні написи на кожному розподільному щиті.

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов’язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об’єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [6], приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною

температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів по способі захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно правилам пристрою електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Однієї з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

7.4 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток 63х63х6 мм, (згідно з ДСТУ 2251-93 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною L=1,6 м, та горизонтальний електрод – металева полоса з перетином 60х5 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру прямокутником (рис. 7.1).

Розрахунок проведемо за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару ґрунту – чорнозем, нижнього шару ґрунту – глина (питомий опір ρ_2 = 40 Ом·м). Умовна товщина верхнього шару ґрунту: H=0,55 м. Відстань між вертикальними заземлювачами (електродами) A=3 м. Глибина закладення горизонтального контура заземлення t=0,6 м. Опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

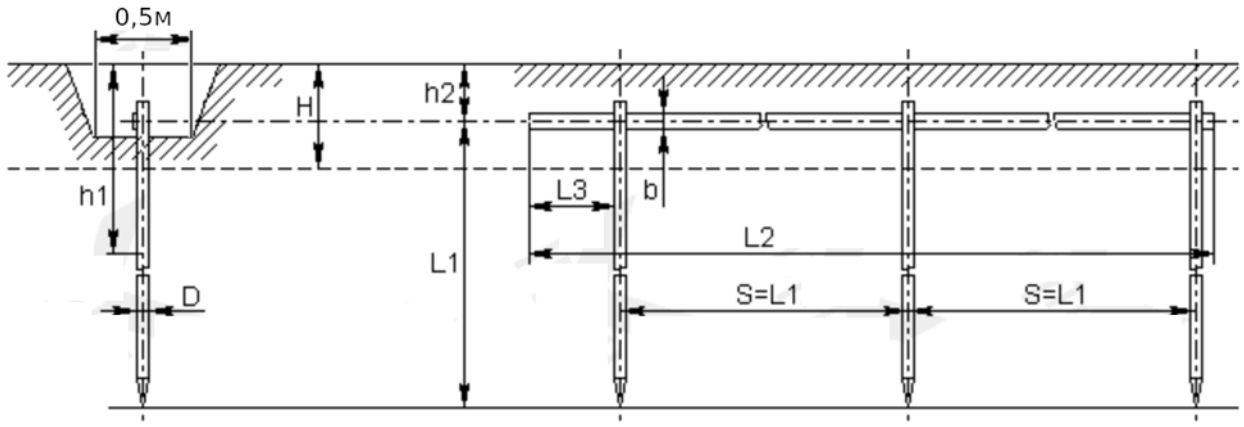


Рисунок 7.1 – Схема штучного заземлення

Виконаємо розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T = t + L/2 = 0,6 + 1,6/2 = 1,4 \text{ м.}$$

Розрахунковий питомий опір ґрунту (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі ґрунту):

$$\rho = \psi \cdot \rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де

ψ = 1,36 – табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому ґрунті [12];

$\rho_2 = 40$ Ом·м – табличне значення питомого опору нижнього шару ґрунту (глина) [12].

Еквівалентний діаметр вертикального електрода (кутка) [12]:

$$D_в = 0,95 \cdot К = 0,95 \cdot 63 = 59,85 \text{ мм} = 0,0598 \text{ м},$$

де К= 63 мм – розмір металевого кутка (заданий).

Відношення A/L=3/1,6 =1,87.

Опір розтіканню електричного струму одного електроду вертикального заземлювача з урахуванням заглиблення заземлювача [12]:

$$R_О = 0,366 \cdot (\rho/L) \cdot [\lg(2L/D_в) + (1/2) \cdot \lg((4T+L)/(4T-L))] =$$

$$= 0,366 \cdot (54,5/1,6) \cdot [(\lg(2 \cdot 1,6/0,0598) + (1/2) \cdot \lg((4 \cdot 1,4+1,6)/(4 \cdot 1,4-1,6))] = 23 \text{ Ом}.$$

Визначаємо коефіцієнт екранування вертикальних електродів $К_{ев}$=0,62 при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [12].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН}$ = 4 Ом:

$$N = R_О / (Кев \, R_{ЗН}) = 23 / (0,62 \cdot 4) = 9,3 \approx 9 \text{ шт}.$$

Визначаємо довжину з'єднуючої полоси:

$$L_П = 1,05 \cdot A \cdot N = 1,05 \cdot 3 \cdot 9 = 28,35 \approx 28 \text{ м}.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору ґрунту $К_П$ [12]:

$$R_П = 0,366 \cdot (\rho \cdot К_П / L_П) \cdot \lg(2 \cdot L_П^2/(B \cdot t)) =$$

$$= 0,366 \cdot (40 \cdot 5/28) \cdot \lg((2 \cdot 28^2)/(0,06 \cdot 0,6)) = 12,2 \text{ Ом}.$$

де

$К_П$=5 – табличне значення кліматичного коефіцієнта питомого опору ґрунту для відповідної кліматичної зони для з'єднуючої полоси [12]:

B = 60 мм = 0,06 м – ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [12]:

$$R = (R_О \cdot R_П)/(R_О \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (23 \cdot 12,2)/(23 \cdot 0,6 + 9 \cdot 12,2 \cdot 0,62) = 3,48 \text{ Ом}.$$

де

$\eta_П$ = 0,6 – табличне значення коефіцієнта екранування з’єднуючої полоси [12].

Умова $R \leq R_{ЗН}$ виконується (3,48 ≤ 4).

Оскільки при 9 вертикальних електродах R суттєво менше $R_{ЗН}$, зменшимо кількість вертикальних електродів N до 8 і виконаємо перерахунок. У результаті остаточно отримали: R=3,91 Ом при кількості вертикальних електродів N=7.

7.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп’ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

8 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

– Досліджена система адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

– На основі отриманих результатів досліджень створена програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Галіченко Т.О. Дослідження та програмна реалізація системи адаптивного управління та забезпечення відмовостійкості розподілених обчислень у хмарній тестовій платформі GENI // Збірник праць молодих науковців ЦНТУ. – Вип. 17. – Кропивницький: ЦНТУ, 2027.

2. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

3. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

4. Vintenko, B., Myronets, I., Smirnov, O., Smirnova, T. «Enhancing the Reliability of Input Data for Nuclear Power Plant Operator Support Systems». *International Conference on Sensors, Devices and Systems. SDaS 2025. Lecture Notes in Electrical Engineering,* 2026. vol 1617. pp 3-12.

5. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing,* 2025. pp. 21-61.

6. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications,* 2025, pp. 193–224.

7. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

8. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security

enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

9. Богаченко, С., Смірнов, О., Буравченко, К., Смірнова, Т., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С. «Недоліки: ресурсів мобільних та edge-платформ при розробці програмного забезпечення для реалізації та оптимізації моделей машинного навчання». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2026, т. 2 № 34, С. 677-686.

10. Вінтенко Б.Ю., Смірнова Т.В., Миронець І.В., Смірнов О.А., Буравченко К.О. «Метод оцінки функціональної стійкості комп'ютерно-орієнтованих процедур системи підтримки оперативного персоналу АЕС». *Системи управління, навігації та зв'язку*, 2026, вип. 2(84), С. 62-69.

11. Смірнова Т.В., Усік П.С., Лисенко І.А., Буравченко К.О., Смірнов О.А. «Методологія підтримки технологічних процесів у критичній інфраструктурі з забезпеченням безпеки інформації на основі хмарних технологій». *Безпека інформації.* 2025. Том 31 № 1. С. 49-60.

12. Смірнова Т.В., Буравченко К.О., Добринчук О.А., Смірнов С.А., Якименко Н.М., Смірнов О.А. «Хмарна технологія моніторингу ключових індикаторів ефективності технологічних процесів у критичній інфраструктурі». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. № 12(43), ч. II. С. 36-54.

13. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С. 413-427.

14. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. Вип.

11(42), ч. II. С.52-62.

15. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк, О., Смірнова, Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

16. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

17. Вінтенко Б.Ю., Миронець І.В., Смірнов О.А. Коваленко О.В., Смірнов С.А., Буравченко К.О., Якименко Н.М. «Дослідження інформаційного забезпечення та технологічних регламентів процесів керування критичною інфраструктурою енергоблоку АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка».* 2024. № 1(25), С. 253–278.

18. Смірнова Т.В., Константинова Л.В., Конопліцька-Слободенюк О.К., Козлов Я.О., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження сучасного стану SIEM-систем». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка».* 2024. № 1(25), С. 6-18.

19. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

20. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

21. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О.

«Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

22. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

23. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

24. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

25. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

26. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

27. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

28. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А.,

Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

31. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

32. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

33. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

34. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

35. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

36. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

37. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

38. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

39. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

40. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

41. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

42. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

43. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE*

International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

44. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

45. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

46. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

47. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

48. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

49. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

50. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

51. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

52. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

53. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

54. О.А. Смірнов, П.С. Усік, «дослідження перспектив використання технологічних рішень в мережах 5g» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

55. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

56. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.