МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 49.00.00.000 ПЗ

Група ІПм-24-3

Сидорів Дмитрій

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Сидорів Дмитрій Тарасович

(прізвище, ім'я, по батькові)

УДК 004.9

(індекс)

МАГІСТЕРСЬКА РОБОТА

Методи аналізу однорідних і неоднорідних навантажень в системах реалізації високопродуктивних обчислень

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Сидорів Д.Т.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Піх Володимир Ярославович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**

доц. В.В. Бандура

“ 04 ” вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Сидоріву Дмитрію Тарасовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Методи аналізу однорідних і неоднорідних навантажень в системах реалізації високопродуктивних обчислень ”

керівник проекту (роботи) Піх В.Я., к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 05 ” листопада 2025 р. № 695/7

2. Строк подання студентом проекту (роботи) 15 грудня 2025 р.

3. Вихідні дані до проекту (роботи) Формальні моделі і методи побудови інформаційних технологій реалізації високопродуктивних обчислень

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

1. Аналіз проблематики розподілу в системах високопродуктивних обчислень
2. Алгоритми та стратегії розподілу навантаження у HPC
3. Методологія аналізу трас та характеристика робочого навантаження HPC-систем
4. Розробка методології аналізу однорідних та неоднорідних навантажень в HPC-системах

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Схематичне зображення архітектури високопродуктивної обчислювальної системи (рис. 1.1)
2. Компоненти кластера (рис. 1.2)
3. Програмний стек HPC (рис. 1.3)
4. Конвеєри аналізу трас (рис. 1.4)
5. Виділення CPU у загальній черзі системи класу 1 (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2025 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2025	виконано
2	Аналіз проблематики розподілу в системах високопродуктивних обчислень	01.10.2025	виконано
3	Алгоритми та стратегії розподілу навантаження у HPC	17.10.2025	виконано
4	Методологія аналізу трас та характеристика робочого навантаження HPC-систем	02.11.2025	виконано
5	Розробка методології аналізу однорідних та неоднорідних навантажень в HPC-системах	19.11.2025	виконано
6	Рекомендації щодо політики оптимізації ресурсів	02.12.2025	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2025	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 75 с., 15 рис., 8 табл., 45 джерел.

Тема: Методи аналізу однорідних і неоднорідних навантажень в системах реалізації високопродуктивних обчислень

Метою роботи є розроблення комплексної методології аналізу однорідних і неоднорідних навантажень у високопродуктивних обчислювальних системах для оптимізації ресурсів та планування конфігурацій HPC-систем.

Об'єктом дослідження є робоче навантаження, що формується в сучасних високопродуктивних обчислювальних системах із гетерогенною архітектурою.

Предметом дослідження є методи аналізу, класифікації та моделювання однорідних і неоднорідних навантажень у HPC-системах, а також їх вплив на ефективність використання обчислювальних ресурсів.

Результати дослідження

В роботі запропоновано універсальні критерії диференціації навантажень для гетерогенних систем, а також формалізовано параметричну модель синтезу навантаження, що відтворює особливості використання GPU.

Висновок

Розроблена методологія є універсальним інструментом для аналізу, моделювання та вдосконалення HPC-систем, а також відкриває перспективи для подальших досліджень у напрямі інтелектуальних систем управління навантаженнями.

ВИСОКОПРОДУКТИВНІ ОБЧИСЛЕННЯ, РОБОЧЕ НАВАНТАЖЕННЯ, GPU, КЛАСИФІКАЦІЯ ЗАДАЧ, СТАТИСТИЧНИЙ АНАЛІЗ, СИНТЕЗ НАВАНТАЖЕННЯ, ГЕТЕРОГЕННА АРХІТЕКТУРА, ПЛАНУВАННЯ РЕСУРСІВ, ПРОФІЛЮВАННЯ.

ABSTRACT

Master Thesis: 75 pp., 15 fig., 8 tab., 45 sources.

Topic: Methods for analyzing homogeneous and heterogeneous loads in high-performance computing systems

The purpose of the work is to develop a comprehensive methodology for analyzing homogeneous and heterogeneous loads in high-performance computing systems for resource optimization and configuration planning of HPC systems.

The object of the study is the workload that is formed in modern high-performance computing systems with heterogeneous architecture.

The subject of the study is methods for analyzing, classifying and modeling homogeneous and heterogeneous loads in HPC systems, as well as their impact on the efficiency of using computing resources.

Research results

The work proposes universal criteria for load differentiation for heterogeneous systems, and also formalizes a parametric load synthesis model that reproduces the features of GPU use.

Conclusion

The developed methodology is a universal tool for analysis, modeling and improvement of HPC systems, and also opens up prospects for further research in the direction of intelligent load management systems.

HIGH-PERFORMANCE COMPUTING, WORKLOAD, GPU, TASK CLASSIFICATION, STATISTICAL ANALYSIS, LOAD SYNTHESIS, HETEROGENEOUS ARCHITECTURE, RESOURCE PLANNING, PROFILING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HPC - High-Performance Computing

CPU - Central Processing Unit

GPU - Graphics Processing Unit

GPGPU - General-Purpose computing on Graphics Processing Units

AVX - Advanced Vector Extensions

SM - Streaming Multiprocessor

CUDA - Compute Unified Device Architecture

OpenMP - Open Multi-Processing

MPI - Message Passing Interface

OpenACC - Open Accelerators

HPS - High-Performance Storage

IOPS - Input/Output Operations Per Second

FPGAs - Field-Programmable Gate Arrays

DC - Data Center

ВСТУП

Актуальність теми.

Системи високопродуктивних обчислень (High Performance Computing, HPC) стали ключовою технологічною основою сучасної наукової та інженерної діяльності, забезпечуючи можливість виконання складних обчислень, моделювання та аналізу великих обсягів даних. Інтенсивний розвиток гетерогенних архітектур, зокрема широке впровадження графічних прискорювачів, значно ускладнив характер робочих навантажень, що надходять до HPC-систем. У таких умовах традиційні методи аналізу та управління навантаженням виявляються недостатніми, оскільки не враховують високий ступінь неоднорідності задач, змінність структури подачі заявок і специфіку взаємодії між різними обчислювальними компонентами. Відсутність універсальних підходів до опису поведінки навантажень у гібридних системах створює суттєві бар’єри для ефективної експлуатації ресурсів та оптимального планування.

У сучасних HPC-кластерах дедалі важливішого значення набуває підтримка збалансованого використання CPU та GPU, проте аналіз реальних трас свідчить про часті приклади недовикористання апаратних прискорювачів або, навпаки, їх перевантаження на тлі простою процесорних вузлів. Характер навантаження формують різні групи користувачів, чиї задачі відрізняються за рівнем паралелізму, часовими патернами, інтенсивністю обчислень і потребами в пам’яті. Таким чином, для підвищення продуктивності системи необхідно мати інструменти, здатні всебічно аналізувати навантаження, виявляти закономірності та аномалії, а також синтезувати репрезентативні моделі для тестування конфігурацій.

Дослідження, представлене у цій роботі, спрямоване на розроблення комплексної методології аналізу однорідних і неоднорідних навантажень, яка враховує архітектурну специфіку HPC-систем, поведінкові характеристики задач і особливості використання графічних прискорювачів. Запропонована

методологія охоплює багаторівневий збір даних, статистичне профілювання, класифікацію задач, виявлення аномалій і синтез робочого навантаження. Отримані результати мають на меті забезпечити універсальний інструментарій для оптимізації використання обчислювальних ресурсів і планування майбутніх систем.

Актуальність роботи зумовлена стрімким зростанням складності HPC-навантажень, їх неоднорідністю та переходом сучасних обчислювальних систем до гетерогенних архітектур, що поєднують різні типи апаратних ресурсів. Традиційні підходи до аналізу навантаження, орієнтовані переважно на однорідні CPU-системи, не забезпечують необхідного рівня точності у випадку багатовузлових конфігурацій з GPU, прискорювачами нового покоління та складними інтерконектами. На практиці це призводить до неефективного використання ресурсів, збільшення часу очікування задач та зниження загальної продуктивності системи. Дослідження реальних трас демонструє, що існуючі планувальники не завжди здатні враховувати поведінкову характеристику задач, а відсутність глибокого аналізу суттєво ускладнює адаптацію системи до змінних умов.

Актуальність роботи, присвяченої методам аналізу однорідних і неоднорідних навантажень (workloads) у системах високопродуктивних обчислень (HPC), визначається стрімкою еволюцією обчислювальних архітектур та зростаючими вимогами до ефективності наукових і промислових симуляцій. Сучасні HPC-системи, включаючи пета- та екзаскейл-кластери, є принципово гетерогенними. Вони поєднують різні типи обчислювальних ресурсів: багатоядерні CPU, графічні прискорювачі (GPU), Field-Programmable Gate Arrays (FPGA) та спеціалізовані тензорні процесори (TPU). Ефективний розподіл навантаження (Load Balancing) і планування (Scheduling) є ключовими для досягнення максимальної продуктивності. Якісний аналіз навантаження є необхідною передумовою для розробки досконалих планувальників.

Зростаючий попит на використання GPU у сферах машинного навчання, моделювання фізичних процесів, обробки зображень і чисельної оптимізації формує новий тип навантажень, що має власні правила поведінки та вимоги до інфраструктури. Крім того, у процесі еволюції кластерів виникає потреба у засобах прогнозного аналізу та синтезу навантажень, які здатні відтворити їх структуру і слугувати основою для тестування масштабованості систем. Тому розроблення методології, що дозволить описати, класифікувати та моделювати неоднорідні навантаження, є важливим науковим і практичним завданням. Актуальність визначається також необхідністю оптимізації енергоспоживання, збалансованого використання CPU та GPU і планування майбутніх HPC-систем з урахуванням реальних характеристик навантажень.

Таким чином, дослідження є своєчасним і критично важливим для переходу до ефективної експлуатації екзаскейл-систем, забезпечуючи високу продуктивність і масштабованість сучасних ресурсомістких застосувань.

Метою роботи є розроблення комплексної методології аналізу однорідних і неоднорідних навантажень у високопродуктивних обчислювальних системах для оптимізації ресурсів та планування конфігурацій HPC-систем.

Об’єктом дослідження є робоче навантаження, що формується в сучасних високопродуктивних обчислювальних системах із гетерогенною архітектурою.

Предметом дослідження є методи аналізу, класифікації та моделювання однорідних і неоднорідних навантажень у HPC-системах, а також їх вплив на ефективність використання обчислювальних ресурсів.

Завдання дослідження

1. Проаналізувати проблематику розподілу навантаження в HPC-середовищах.

2. Дослідити архітектурні особливості CPU- та GPU-орієнтованих систем.

3. Виконати аналіз реальних трас і журналів диспетчерів ресурсів.

4. Сформувати статистичні характеристики навантаження для різних класів HPC-систем.

5. Створити метод виявлення аномалій у роботі користувацьких задач.

6. Розробити методологію синтезу робочого навантаження на основі репрезентативних періодів.

Методи дослідження

У роботі застосовано методи статистичного аналізу, класифікаційного моделювання, багаторівневого збору даних, аналізу часових рядів, інструменти моніторингу HPC-систем, методи синтезу навантаження, а також підходи до виявлення аномалій та аналізу журналів диспетчерів ресурсів.

Наукова новизна отриманих результатів

Запропоновано цілісну методологію аналізу однорідних і неоднорідних навантажень, яка поєднує багатошарову структуру збору даних, класифікацію задач, виявлення аномалій та синтез репрезентативного навантаження. Розроблено новий підхід до вибору репрезентативного періоду, що забезпечує достовірну реконструкцію поведінки реальних трас. Отримані результати дозволяють визначити закономірності еволюції навантаження та включити їх у моделі прогнозування та оптимізації.

Практичне застосування результатів

Результати дослідження можуть бути використані адміністраторами HPC-систем для оптимізації розподілу ресурсів, планувальниками для створення адаптивних політик призначення задач, а також дослідницькими центрами для моделювання конфігурацій майбутніх кластерів. Метод синтезу навантаження може застосовуватися для тестування продуктивності та масштабованості систем. Класифікація навантажень і виявлення аномалій дозволяють підвищити ефективність використання GPU та зменшити простої ресурсів.

Структура магістерської роботи. Представлена робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 75 сторінок, і містить 15 рисунків, 8 таблиць, перелік використаних джерел із 45 позицій.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ РОЗПОДІЛУ В СИСТЕМАХ ВИСОКОПРОДУКТИВНИХ ОБЧИСЛЕНЬ

1.1. Аналіз та моделювання робочого навантаження в системах високопродуктивних обчислень

Хмарні обчислювальні системи (Cloud Computing Systems) та високопродуктивні обчислювальні системи (High-Performance Computing, HPC), зокрема суперкомп'ютери, вимагають значних інвестицій ресурсів (обладнання, енергоспоживання, ліцензування) та висококваліфікованого персоналу для їх підтримки та ефективної експлуатації.

Високопродуктивні обчислювальні системи або High-Performance Computing (HPC), — це технології та системи, які об'єднують значну кількість обчислювальних ресурсів для виконання надзвичайно складних і об'ємних завдань, що вимагають швидкості, недосяжної для звичайних комп'ютерів чи робочих станцій.

Основна ідея HPC — це паралельні обчислення. Замість того, щоб один процесор виконував завдання послідовно (одне за одним), завдання розбивається на безліч менших частин, і ці частини одночасно обробляються багатьма процесорами (або іншими обчислювальними пристроями).

Ключові компоненти HPC-системи:

1. Кластери (Compute). Групи окремих комп'ютерів, званих вузлами (nodes), які працюють разом. Це можуть бути сервери, обладнані потужними центральними процесорами (CPU) та/або графічними процесорами (GPU).

2. Високошвидкісне мережеве з'єднання (Networking). Дуже швидкісні канали зв'язку (наприклад, InfiniBand або високошвидкісний Ethernet) між вузлами та системою зберігання, що забезпечують мінімальну затримку (низьку латентність) для обміну даними між паралельними процесами.

3. Системи зберігання (Storage): Швидкісні системи для зберігання та доступу до величезних обсягів даних, які необхідні для обчислень.

4. Суперкомп'ютери. Найпотужніші HPC-системи у світі, які часто є масивними кластерами з тисяч вузлів, здатні виконувати квадрильйони (10 в степені 15) операцій за секунду (петафлопс).

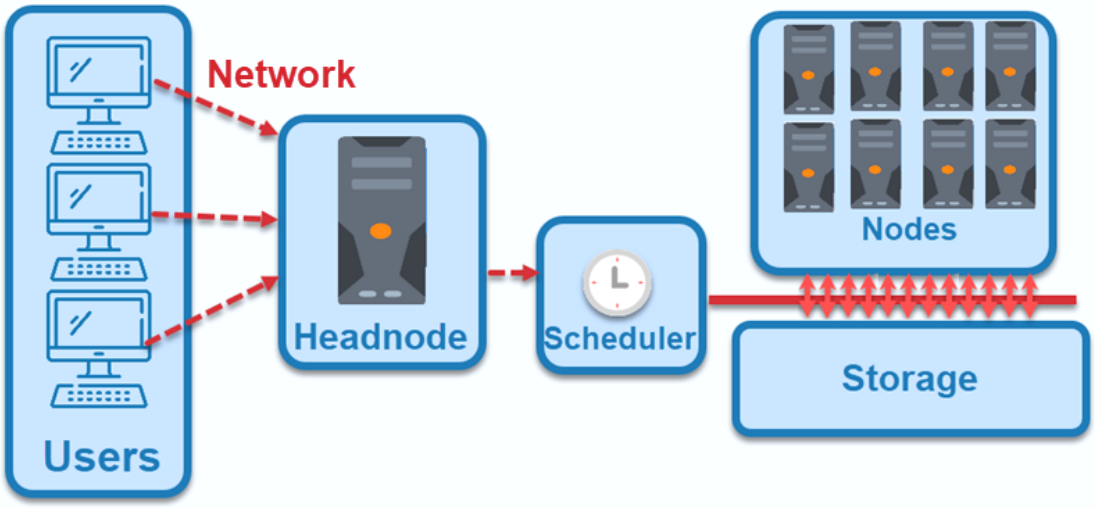

Рис. 1.1. Схематичне зображення архітектури високопродуктивної обчислювальної системи

На цій схемі (рис. 1.1) представлені такі ключові елементи:

- обчислювальні вузли - окремі сервери або комп'ютери, які виконують обчислювальну роботу.

- кластери - група об'єднаних вузлів, що працюють як єдина система.

- високошвидкісне з'єднання - мережева інфраструктура (наприклад, InfiniBand або Omni-Path), яка забезпечує надзвичайно швидкий обмін даними між вузлами.

- сховище - масштабовані системи зберігання даних, необхідні для швидкого доступу до великих масивів інформації.

- планувальник / менеджер ресурсів - програмне забезпечення, яке керує розподілом обчислювальних завдань між вузлами кластера.

HPC дозволяє не просто обробляти дані швидше, але й проводити симуляції з більшою роздільною здатністю та складністю, що веде до нових наукових відкриттів і технологічних проривів.

Робоче навантаження (Workload) системи є ключовим чинником, що прямо впливає як на системну продуктивність (System Performance), так і на якість обслуговування (Quality of Service, QoS) для кінцевих користувачів. З огляду на це, глибоке розуміння та кількісний аналіз характеристик навантаження є критично важливим для організацій, що експлуатують та розвивають такі складні обчислювальні інфраструктури.

У цьому дослідженні ми проводимо емпіричний аналіз трас (логістичних записів) з двох продуктивних суперкомп'ютерних систем (Production HPC Systems). Метою є ідентифікація та характеристика типових патернів їхнього робочого навантаження, а також виявлення конкретних потреб користувачів HPC-ресурсів.

Особлива увага приділяється аналізу використання графічних процесорів (GPU) — ключового компонента сучасних HPC-систем для прискорення обчислень —дослідниками, які представляють різноманітні наукові домени та інженерні дисципліни. Детальний аналіз метрик, таких як тривалість завдань (Job Duration), використання ресурсів (Resource Utilization), розмір паралелізму (Parallelism Size) та інтенсивність використання GPU, дозволяє створити точний профіль типового користувача та його обчислювальних потреб.

На основі отриманих результатів емпіричного аналізу ми виконуємо генерацію синтетичного робочого навантаження (Synthetic Workload Generation). Це синтетичне навантаження, яке точно імітує характеристики реальних трас, має високу релевантність та може бути використане як стандартизований бенчмарк для:

1. Тестування та валідації архітектури та програмного забезпечення майбутніх суперкомп'ютерних систем (Pre-Deployment Testing).

2. Оцінки ефективності планувальників завдань (Job Schedulers) та механізмів керування ресурсами (Resource Management Mechanisms).

Крім того, проведений аналіз дозволяє зробити обґрунтовані висновки (Informed Observations) щодо ефективного та оптимального надання ресурсів

(Effective Resource Provisioning), включаючи стратегії балансування навантаження (Load Balancing), управління чергами (Queue Management) та оптимізації використання гетерогенних ресурсів (GPU/CPU). Це сприятиме максимізації пропускної здатності (Throughput) системи та мінімізації часу очікування (Wait Time) для користувачів.

1.2. Проблеми розподілу навантаження у високопродуктивних обчислювальних системах

Розподіл навантаження є критично важливим аспектом у високопродуктивних обчислювальних системах, таких як кластери та суперкомп'ютери. Його основна мета — оптимально і рівномірно розподілити обчислювальні завдання між доступними ресурсами (процесорами, ядрами, пам'яттю) для максимізації пропускної здатності та мінімізації часу виконання завдань.

Однак, досягнення ідеального розподілу стикається з низкою складних проблем:

1. Нерівномірність робочого навантаження. Це найбільш поширена та фундаментальна проблема. Багато HPC-застосувань (наприклад, моделювання, симуляції) мають непередбачуваний або динамічно змінний обсяг роботи для різних частин даних або різних часових кроків. Окремі процеси або потоки можуть мати суттєво різний час виконання через різну складність обчислень, залежність від даних або умовні переходи.

Також існує ефект "вузького горла" (Bottleneck) - це якщо один вузол або процес отримує непропорційно велике навантаження, він стає "вузьким горлом", змушуючи інші, вже завершені вузли, простоювати (idle), чекаючи на його завершення. Це знижує загальну ефективність системи.

2. Витрати на передачу даних та зв'язок. Ефективний розподіл часто вимагає переміщення даних між вузлами. Переміщення великих обсягів

даних через мережу кластера споживає час (латентність) і обмежує пропускну здатність, що може звести нанівець вигоди від паралелізації.

Завдання можуть мати жорсткі залежності по даних (data dependencies), коли один процес не може розпочатися, доки інший не завершить обробку і не передасть результати. Це створює неявні бар'єри та простої.

Комунікаційні бар'єри - використання колективних операцій (наприклад, MPI_Allreduce, MPI_Barrier) вимагає, щоб усі процеси досягли певної точки перед продовженням, що узалежнює швидкість роботи всієї системи від найповільнішого вузла.

3. Динамічність та адаптивність. Вибір між статичним і динамічним розподілом навантаження:

- статичний розподіл. Навантаження визначається та фіксується перед початком виконання. Він простий у реалізації, але неефективний для динамічних завдань, оскільки не може реагувати на зміни під час виконання.

- динамічний розподіл: Навантаження перерозподіляється під час виконання програми. Це дозволяє досягти кращої рівномірності, але сам механізм перерозподілу (моніторинг, прийняття рішення, міграція даних/процесів) додає значні додаткові витрати (overhead), які можуть бути більшими, ніж вигода від балансування.

4. Зернистість та міграція. Це стосується розміру обчислювального блоку та складності його переміщення.

Дрібна Зернистість (Fine Granularity) - якщо завдання дуже дрібні, витрати на їх координацію та зв'язок між вузлами можуть домінувати над часом обчислення.

Міграція процесів/потоків - переміщення активного процесу або його стану з одного вузла на інший для балансування навантаження є технічно складним і витратним з точки зору пам'яті та часу, особливо якщо процес має великий робочий простір (великий обсяг пам'яті).

5. Географія та топологія мережі. Фізична структура кластера впливає на розподіл.

Нерівномірний доступ до пам'яті (NUMA). Навіть в межах одного вузла різні ядра мають різний час доступу до різних областей пам'яті (локальна vs. віддалена). Ігнорування NUMA-архітектури може призвести до уповільнення.

Топологія мережі. Сучасні HPC-системи використовують складні топології (наприклад, Torus, Fat Tree). Розподіл завдань має враховувати, які вузли фізично "близькі" один до одного, щоб мінімізувати затримку при обміні даними між ними. Неправильне розміщення може призвести до перевантаження окремих каналів.

Успішне управління розподілом навантаження в HPC вимагає використання складних, часто гібридних, алгоритмів, які враховують як обчислювальні потреби, так і комунікаційні витрати.

1.3. Особливості архітектури систем високопродуктивних обчислень

Архітектури високопродуктивних обчислень (HPC) складаються з кількох ключових будівельних блоків, які працюють разом, створюючи потужне та масштабоване обчислювальне середовище, здатне вирішувати складні обчислювальні завдання. Ці будівельні блоки можна розділити на апаратні компоненти та програмні компоненти.

Давайте розглянемо кожен із цих блоків і те, як вони використовуються для створення HPC-архітектури:

1. Локальна HPC-архітектура (on-premises HPC architecture)

Високопродуктивні обчислення (HPC) — це метод обробки великих обсягів даних та виконання складних обчислень на високих швидкостях. Локальна HPC-архітектура передбачає встановлення та управління HPC-системами у власному центрі обробки даних (ЦОД) організації.

А. Апаратні компоненти

Обчислювальні вузли (Compute Nodes): кластер комп'ютерів, кожен з яких має кілька процесорів, виділені схеми та локальну пам'ять, що

виконують обчислення. Приклади включають масштабовані процесори Intel Xeon, процесори AMD EPYC або процесори на базі ARM, такі як Ampere Altra. Ці процесори забезпечують високу кількість ядер та розширені функції, як-от інструкції AVX-512, для прискорення HPC-навантажень.

Системи з GPU-прискоренням - графічні процесори (GPU) забезпечують можливості паралельної обробки для підтримки завдань, які можуть бути паралелізовані як на ядрах CPU, так і на GPU. Деякі GPU включають NVIDIA A100 або A40, AMD Instinct MI100 або MI200, які забезпечують масові можливості паралельної обробки для прискорення завдань ШІ, машинного навчання та наукових обчислень.

Високопродуктивне сховище (HPS) - паралельні файлові системи або високошвидкісні контролери сховища підвищують швидкість доступу до даних та їхньої передачі. Деякі приклади HPS включають паралельні файлові системи Lustre, IBM Spectrum Scale (GPFS) або BeeGFS. Ці файлові системи використовують кілька серверів сховища та високошвидкісні мережі для забезпечення надзвичайно високої пропускної здатності та показників IOPS.

Комутатор InfiniBand - з'єднує та полегшує зв'язок між усіма вузлами в кластері. Комутатори Mellanox/NVIDIA InfiniBand, такі як серія Quantum 8700, забезпечують високошвидкісні міжз'єднання з низькою затримкою між обчислювальними вузлами.

Фізичні приміщення та живлення - фізичний простір, необхідний для розміщення HPC-обладнання, включаючи джерело живлення та інфраструктуру охолодження.

Програмовані вентильні матриці - апаратне прискорення, що налаштовується, для середовищ, які вимагають високопродуктивної обробки з низькою затримкою.

Вузли віддаленої візуалізації (Remote Visualization Nodes) переносять завдання візуалізації з основних обчислювальних вузлів, щоб зберегти обчислювальну ефективність.

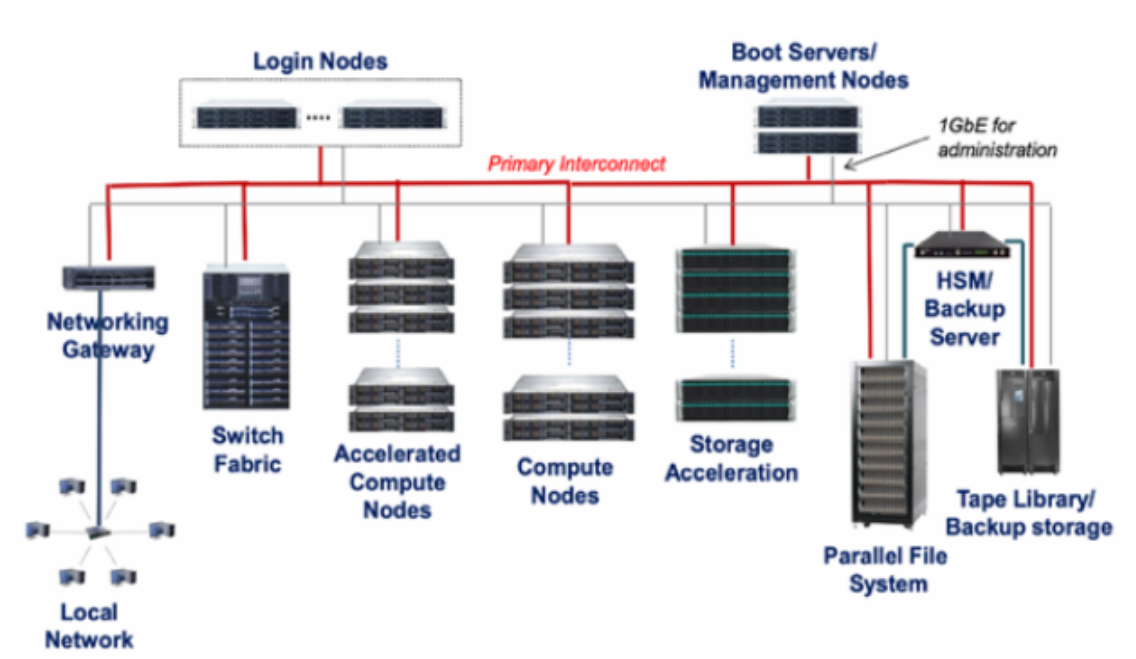

Рис. 1.2. Компоненти кластера

Б. Програмні компоненти

Планувальники HPC (HPC Schedulers) - спеціалізоване програмне забезпечення, яке оркеструє спільні обчислювальні ресурси, забезпечуючи ефективне використання вузлів. Приклади планувальників включають SLURM та Univa Grid Engine. Ці планувальники ефективно керують спільними обчислювальними ресурсами та робочими навантаженнями.

Програмне забезпечення для паралельних обчислень (Parallel Computing Software): керує паралельними завданнями, забезпечуючи ефективну координацію та синхронізацію. Приклади включають OpenMP, MPI, CUDA та OpenACC. Ці бібліотеки та фреймворки дозволяють розробникам паралелізувати свій код для ефективного виконання в HPC-системах.

Програмне забезпечення для управління даними (data management software): оптимізує управління системними ресурсами, обробляє зберігання, вилучення, організацію та переміщення даних у середовищі HPC. Прикладами є IBM Spectrum Scale, DataWarp від Cray та BeeGFS. Ці

програмно-визначені рішення для зберігання оптимізують розміщення, рівні та переміщення даних.

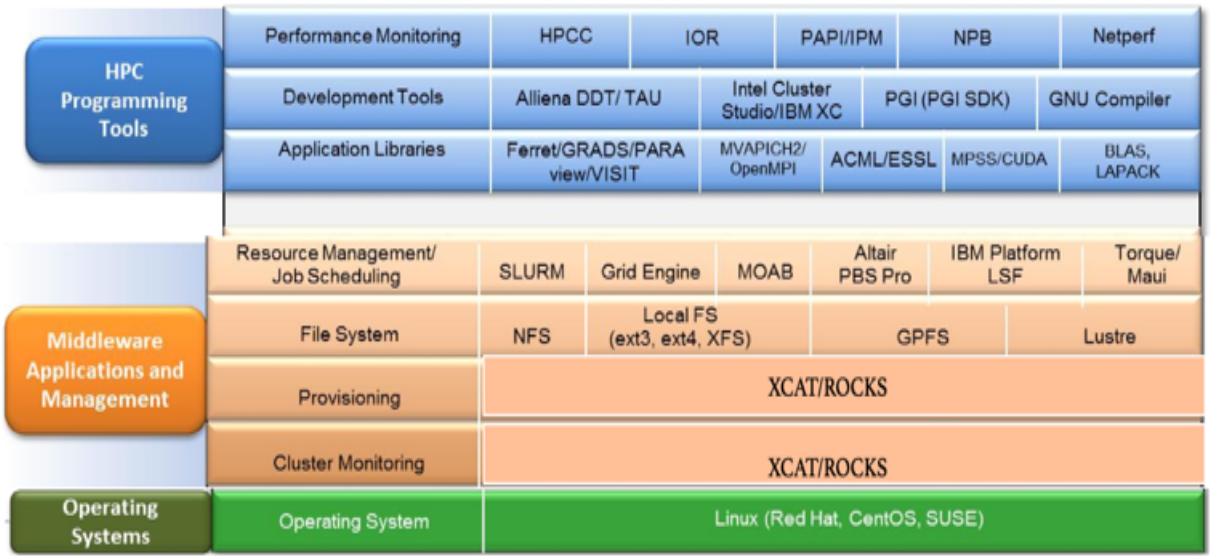

Рис. 1.3. Програмний стек HPC

Для створення HPC-архітектури, ці будівельні блоки збираються та налаштовуються відповідно до специфічних вимог цільового робочого навантаження застосунку. Обчислювальні вузли взаємопов'язуються за допомогою високошвидкісних міжз'єднань та організовуються у кластер. Системи зберігання даних забезпечуються для зберігання вхідних даних, проміжних результатів та вихідних даних, згенерованих обчислювальними завданнями.

Операційна система, проміжне програмне забезпечення (middleware) та інструменти розробки встановлюються та налаштовуються для підтримки паралельних обчислень, планування завдань та розробки застосунків. Мережева інфраструктура налаштовується на основі обраної топології мережі, з розгорнутими комутаторами, маршрутизаторами та інструментами мережевого управління для забезпечення ефективного зв'язку між обчислювальними вузлами. Нарешті, HPC-застосунки розробляються або переносяться на архітектуру з використанням моделей та бібліотек

паралельного програмування, а оптимізації продуктивності застосовуються за допомогою інструментів розробки та методів профілювання для максимізації обчислювальної ефективності.

Створення масштабованих систем та програмних архітектур для HPC ставить численні виклики та міркування. Серед цих викликів балансування навантаження, стійкість (resilience) та енергоефективність відіграють вирішальні ролі у визначенні загальної продуктивності та стійкості HPC-архітектур.

Балансування навантаження є суттєвим для рівномірного розподілу обчислювальних завдань між ресурсами у середовищі паралельних обчислень. Неоднорідний розподіл робочого навантаження може призвести до недовикористання ресурсів та появи вузьких місць продуктивності, що перешкоджає масштабованості. В HPC-архітектурах балансування навантаження стає особливо складним через динамічну природу робочих навантажень та неоднорідність обчислювальних ресурсів.

1.4. Аналіз еволюції робочого навантаження та оптимізація ресурсного забезпечення в високопродуктивних обчислювальних системах

1.4.1. Актуальність аналізу навантаження в HPC-середовищах

Суперкомп'ютерні системи продуктивного масштабу (Production-Scale Supercomputing Systems) вимагають значних капіталовкладень (у апаратне забезпечення та операційних витрат на персонал та підтримку. Для досягнення максимальної системної продуктивності та ефективного використання ресурсів, глибоке розуміння характеристик робочого навантаження є фундаментально важливим для оптимального проєктування системи та її подальшої експлуатації.

Оскільки характеристики навантаження є динамічними та еволюціонують протягом життєвого циклу розгорнутої системи, їх адекватне

розуміння потребує аналізу історичних даних (трас) за весь термін експлуатації системи.

Нещодавнє інтеграція гетерогенних прискорювачів, зокрема графічних процесорів (GPU), у великомасштабні кластерні системи [2] додало додатковий рівень складності до традиційного аналізу навантаження. Організації, які підтримують такі продуктивні кластерні системи, повинні володіти вичерпною інформацією про потреби своїх користувачів та характеристики навантажень, які вони генерують. Крім того, симуляції на основі трас [3] є ключовим методологічним підходом для ефективного тестування та оцінки життєздатності нових архітектурних конфігурацій. Отже, достовірне знання специфічного для системи навантаження є необхідним не лише для ефективної експлуатації поточної інфраструктури, але й для точного планування та забезпечення ресурсами майбутніх систем.

1.4.2. Впровадження GPU та виклики аналізу використання

Швидке впровадження та значні інвестиції у GPU як обчислювальних прискорювачів у HPC-середовищі викликали підвищений інтерес у дослідницькій спільноті. Незважаючи на велику кількість досліджень у галузі комп'ютерних наук щодо різноманітних сценаріїв їх використання [1, 3], залишається недослідженим питання щодо фактичного масштабування їх використання вченими, які представляють різні наукові дисципліни. Оскільки ці користувачі є основними споживачами ресурсів багатьох продуктивних суперкомп'ютерних систем, кількісний аналіз їх використання GPU, або його відсутність, становить значний інтерес для організацій, що займаються проєктуванням та підтримкою подібних інфраструктур.

На основі детального аналізу наявних трас було розроблено синтетичне робоче навантаження (Synthetic Workload), яке репрезентативно моделює патерни використання кластерів ARC. Це синтетичне навантаження може бути використано для проведення симуляційних експериментів з метою

тестування майбутніх архітектурних конфігурацій та оцінки їх здатності ефективно обслуговувати навантаження, унікальне для систем.

Спираючись на емпіричні спостереження, отримані в результаті аналізу трас, були сформульовані практичні рекомендації щодо оптимізації використання поточних ресурсів та стратегічного планування забезпечення майбутніх ресурсів.

1.5. Опис інфраструктури високопродуктивних обчислень та управління ресурсами

Центри обчислень для досліджень (ADRC) є підрозділами в академічних або дослідницьких установах, які здійснюють операційне управління загальноустановчими суперкомп’ютерними ресурсами та інфраструктурою високопродуктивних обчислень (HPC).

Основна місія ADRC полягає у:

- Наданні сучасних обчислювальних ресурсів та засобів візуалізації.

- Забезпеченні технічної підтримки та лідерства для сприяння обчислювальним дослідженням.

- Стимулюванні розвитку знань і навичок у сфері обчислювальних інструментів та технік серед науковців, викладачів та студентів [4].

Для ефективного планування та управління завданнями (Job Scheduling and Management) в HPC-кластерах зазвичай застосовується дворівнева програмна архітектура:

- диспетчер ресурсів (Resource Manager), наприклад, TORQUE (Terascale Open-Source Resource and QUEue Manager) [3], який використовується для фізичного виділення обчислювальних вузлів та їх компонентів (ядер, пам'яті, прискорювачів) під користувацькі завдання.

- планувальник кластера (Cluster Scheduler) функціонує над диспетчером ресурсів. Він приймає запити від користувачів, визначає їх пріоритет та час запуску, а також керує чергами завдань.

Більшість операційних даних для аналізу трас збирається саме за допомогою цих двох програмних продуктів.

1.5.1. Гібридна HPC-система класу 1 (референсна архітектура)

Референсна HPC-система класу 1 є продуктивною 42-вузловою гібридною системою CPU-GPU, що працює під управлінням стандартної дистрибуції Linux (наприклад, CentOS Linux 5.5).

Кожен вузол оснащений чотирма восьмиядерними процесорами (наприклад, AMD Magny Cour з тактовою частотою 2,3 ГГц), що забезпечує сукупну обчислювальну потужність у 1344 ядра. Кожен вузол має значний обсяг оперативної пам'яті (наприклад, 64 ГБ), що відповідає щільності 2 ГБ на ядро.

16 вузлів системи додатково оснащені двома графічними процесорами (наприклад, NVidia S2050 Fermi) кожен. Ця гетерогенна конфігурація призначена для обчислень та візуалізації великих наборів даних, причому GPU використовуються переважно для прискорення візуалізаційних завдань. Вузли з'єднані високошвидкісною мережею (наприклад, InfiniBand з чотирикратною швидкістю передачі даних, 40 Гбіт/с).

Система має стабільне локальне сховище обсягом 40 ТБ.

На цій системі застосовуються політики обмеження ресурсів (Resource Limits):

- Загальна черга: обмеження для звичайних користувачів, наприклад, до 256 ядер одночасно.

- Черга GPU: обмеження до 192 ядер на користувача.

- Обмеження часу виконання: максимальний час виконання завдання становить 300 годин.

1.5.2. Прискорений GPU-кластер класу 2 (референсна архітектура)

Референсний GPU-кластер класу 2 є більш масштабною системою, що складається з 204 вузлів з прискоренням на GPU (наприклад, працює під

управлінням CentOS 6.3). Кожен вузол містить два шестиядерні процесори (наприклад, Intel Xeon E5645 з тактовою частотою 2,4 ГГц), що сумарно дає 2448 ядер.

Загальний обсяг пам'яті на вузол становить 24 ГБ (2 ГБ на ядро). Кожен вузол обладнаний двома графічними процесорами (наприклад, NVidia M2050), що робить його повністю прискореним кластером.

Аналогічно класу 1, вузли використовують високошвидкісне з'єднання InfiniBand (40 Гбіт/с).

Кластер має дві окремі черги завдань з різними політиками:

- Звичайна черга - призначена для типових завдань.

Обмеження розміру: максимум 32 вузли та 384 ядра на користувача.

Обмеження часу виконання: 72 години.

- Довга черга - призначена для особливо тривалих завдань.

Обмеження розміру: суворіші обмеження, наприклад, максимум 8 вузлів та 96 процесорів.

Обмеження часу виконання: продовжений ліміт до 144 годин.

На відміну від систем з розділеними чергами, у цьому кластері ресурси між чергами можуть розподілятися, але завданням у довгій черзі може надаватися пріоритет або інші переваги. Цей кластер, як правило, починає свою роботу як дослідницька система з обмеженим доступом для ключових дослідників, з очікуванням розширення доступу до ширшої спільноти після завершення етапу тестування.

1.6. Алгоритми та стратегії розподілу навантаження у HPC

Стратегії розподілу навантаження у високопродуктивних обчислювальних системах (HPC) поділяються на дві основні категорії: статичні (коли розподіл визначається до початку виконання) та динамічні (коли навантаження коригується під час виконання).

1. Статичні Стратегії (Static Load Balancing)

Статичні стратегії ідеально підходять для застосувань, де робоче навантаження є передбачуваним та рівномірним або має стабільну структуру. Вони мають мінімальні витрати (overhead) на адміністрування.

Таблиця 1.1.

Статичні стратегії балансування

Стратегія	**Опис**	**Застосування**
Геометрична декомпозиція	Простір даних (наприклад, 2D або 3D сітка) рівномірно поділяється між процесами/вузлами. Кожен процес відповідає за фіксовану, рівну частину.	Рішення диференціальних рівнянь, обчислювальна гідродинаміка (CFD) на регулярних сітках.
Циклічний (Round-Robin)	Завдання призначаються процесам по черзі (P1, P2, P3, P1, P2, P3...). Припускає, що всі завдання мають приблизно однаковий час виконання.	Обробка великих наборів незалежних завдань (embarrassingly parallel).
Вагове призначення (Weighted)	Завдання призначаються вузлам пропорційно їхній обчислювальній потужності (наприклад, вузол із 8 ядрами отримує вдвічі більше завдань, ніж вузол із 4 ядрами).	Гетерогенні кластери, де вузли мають різну апаратну конфігурацію.

Перевага: дуже низькі комунікаційні витрати, простота реалізації.

Недолік: нездатна реагувати на динамічний дисбаланс під час виконання, що призводить до простою.

2. Динамічні стратегії (Dynamic Load Balancing)

Динамічні стратегії необхідні, коли час виконання обчислень залежить від даних або змінюється непередбачувано. Вони працюють, постійно або періодично, моніторячи стан вузлів і мігруючи навантаження від перевантажених вузлів до недовантажених.

Важливий компроміс полягає в тому, що ефективність динамічних стратегій безпосередньо залежить від витрат на моніторинг та міграцію. Якщо ці витрати занадто високі, вони можуть звести нанівець вигоду від кращого балансування.

Таблиця 1.2.

Динамічні стратегії балансування

Стратегія	Опис	Адресовані проблеми
Алгоритм "Крадіжки роботи" (Work Stealing)	Процес, який завершив свою роботу (недовантажений, stealer), активно запитує або "краде" завдання з черги перевантаженого процесу (victim).	Високий динамічний дисбаланс, особливо у неструктурованих застосуваннях.
Алгоритм "Поділу роботи" (Work Sharing)	Перевантажений процес (sender) ініціює передачу частини своїх завдань іншому, менш завантаженому процесу (receiver).	Зменшення простою, реакція на раптові піки навантаження.
Дифузійні алгоритми	Навантаження розглядається як "тепло", яке має рівномірно розподілитися по системі. Вузол обмінюється інформацією про свій стан лише з найближчими сусідами і передає частину навантаження, якщо сусід менш завантажений.	Мінімізація комунікаційних витрат (локальний зв'язок), підходить для великих кластерів.
Перерозподіл на основі графів (Graph Partitioning)	Складні структури даних (наприклад, нерегулярні сітки) моделюються як графи. Використовуються бібліотеки (наприклад, Metis, ParMetis) для перерозподілу розділів графа, щоб мінімізувати вагу розрізу (зв'язок) і максимізувати рівномірність навантаження.	Мінімізація зв'язку та рівномірність навантаження при нерегулярних обчисленнях.

Розподіл навантаження (Load Balancing) у контексті HPC-систем є процедурою, спрямованою на забезпечення рівномірного використання обчислювальних ресурсів з метою мінімізації часу простою (idling) та оптимізації загальної пропускної здатності системи. Алгоритми класифікуються відповідно до моменту прийняття рішення про розподіл: статичні (заплановані до виконання) та динамічні (адаптивні під час виконання).

1. Статичні методи розподілу навантаження

Статичні алгоритми ґрунтуються на апріорному знанні про структуру завдання та обчислювальні характеристики системи. Вони демонструють ефективність у випадках детермінованих або регулярних обчислень.

А. Декомпозиція області

Цей метод застосовується переважно до завдань, що працюють з регулярними просторовими сітками або геометричними об'єктами (наприклад, симуляції методом скінченних елементів).

Принцип: обчислювальна область ділиться на N непересічних підобластей, де N — кількість доступних процесорів.

Механізм: поділ зазвичай відбувається рівномірно (наприклад, блокова або циклічна декомпозиція). Кожен процесор відповідає за обчислення у своїй підобласті. Зв'язок між процесорами відбувається лише вздовж межування підобластей, що природно мінімізує комунікаційний трафік завдяки локальності даних.

Б. Декомпозиція графа

Використовується для моделювання нерегулярних обчислень та задач, структурованих як граф (наприклад, обробка неструктурованих сіток).

Принцип: Завдання моделюється як неорієнтований граф G=(V,E), де вершини V представляють обчислювальні одиниці (навантаження), а ребра E — комунікаційні залежності.

Механізм: алгоритм (наприклад, заснований на евристиках Кернігана–Ліна або Фідлера) націлений на поділ множини вершин V на N підмножин, забезпечуючи дві критичні умови:

1) рівновагу навантаження (сума ваг вершин у кожній підмножині приблизно однакова),

2) мінімізацію перерізу (сума ваг ребер, що з'єднують вершини різних підмножин, є мінімальною).

Це мінімізує міжпроцесорну комунікацію.

2. Динамічні методи розподілу навантаження

Динамічні алгоритми є адаптивними і необхідні для недетермінованих застосувань, де навантаження на процесор може змінюватися протягом

виконання. Вони вимагають механізмів моніторингу та міграції навантаження.

А. Дифузійні алгоритми

Ці методи використовують концепцію поступового вирівнювання навантаження через локальну взаємодію, подібно до фізичного процесу дифузії.

Принцип: навантаження вузла розглядається як "концентрація". Якщо навантаження вузла Pi відрізняється від навантаження його сусіда Pj, відбувається двостороння міграція навантаження, яка намагається встановити локальну рівновагу.

Механізм: обмін інформацією та міграція навантаження відбувається лише між безпосередньо зв'язаними вузлами. Наприклад, різниця в навантаженні $\Delta L = Li - Lj$ може бути використана для визначення обсягу роботи, який має бути переданий від більш завантаженого вузла до менш завантаженого. Це забезпечує високу масштабованість, оскільки не вимагає глобальної синхронізації.

Б. Алгоритми на основі токенів

Використовуються в системах із послідовною комунікацією або складною топологією.

Принцип: спеціальний токен (маркер) циркулює через усі вузли системи, збираючи інформацію про їхнє поточне навантаження.

Механізм: вузол, який отримує токен, аналізує зібрані дані про стан системи, приймає глобальне або квазіглобальне рішення щодо необхідності перерозподілу, і, можливо, ініціює міграцію навантаження. Це забезпечує глобальну узгодженість інформації про навантаження ціною додаткової латентності, пов'язаної з циркуляцією токена.

Вибір між статичним та динамічним підходами є компромісом, що ґрунтується на двох ключових характеристиках обчислювального завдання: передбачуваність навантаження та вартість комунікації.

Висновки до розділу

У першому розділі було встановлено, що сучасні системи високопродуктивних обчислень працюють в умовах постійно зростаючої складності, що робить аналіз і моделювання робочого навантаження ключовим аспектом їх ефективного функціонування. Детальний розгляд проблем розподілу навантаження показав, що вони є наслідком нерівномірного надходження задач, різнорідності типів обчислень та динамічної зміни вимог користувачів. З’ясувалося, що гетерогенність ресурсів, зокрема поєднання CPU і GPU, створює додаткові труднощі у визначенні оптимальних політик планування. Аналіз архітектур високопродуктивних систем продемонстрував, що структурні особливості впливають на формування навантаження не менше, ніж програмні механізми керування. У роботі підкреслено важливість розуміння специфіки взаємодії між компонентами HPC-системи, оскільки навіть незначні зміни можуть призводити до суттєвих коливань у продуктивності. Особливий акцент зроблено на ролі GPU як ключових елементів сучасних обчислювальних платформ, що потребують спеціалізованих методів аналізу їх завантаження.

РОЗДІЛ 2. МЕТОДОЛОГІЯ АНАЛІЗУ ТРАС ТА ХАРАКТЕРИСТИКА РОБОЧОГО НАВАНТАЖЕННЯ СИСТЕМ ВИСОКОПРОДУКТИВНИХ ОБЧИСЛЕНЬ

2.1. Методологія аналізу робочого навантаження

Головний фокус дослідження зосереджено на аналізі трас (Workload Traces) з метою отримання корисних статистичних даних про характеристики завдань (Job Characteristics), які виконуються на продуктивних суперкомп'ютерних системах. Хоча точна природа та ідентифікація використовуваних користувачами програмних застосунків часто залишається конфіденційною або невідомою, такі статистичні показники, як рівень використання процесора (CPU Utilization), вимоги до оперативної пам'яті та тривалість виконання, можуть бути ефективно використані для кількісної характеристики HPC-навантаження. Крім того, ці статистичні дані є критично важливими для подальшого симулювання навантаження (Workload Simulation) при проєктуванні та тестуванні майбутніх хмарних (Cloud) або гетерогенних HPC-систем.

Для спрощення та стандартизації аналізу, завдання були класифіковані на три категорії за критерієм часу виконання:

1. Завдання секунд (Short-Burst Jobs) - завдання з часом виконання до п'яти хвилин. Ця категорія типово включає невеликі, імпульсні обчислення, а також тестові запуски або завдання-заглушки (Placeholder Jobs), які не передбачають значного обчислювального навантаження.

2. Завдання хвилин (Medium-Duration Jobs) - завдання, тривалість яких коливається від п'яти хвилин до однієї години. Ці завдання являють собою середньотривалі обчислювальні процеси, які виконують реальну наукову роботу, але не вимагають тривалого часу виконання.

3. Завдання годин (Long-Running Jobs) - категорія включає всі завдання, тривалість яких перевищує одну годину. Це довготривалі завдання, які характеризуються значними вимогами до часу роботи та зазвичай є основою великих наукових обчислень.

2.2. Огляд досліджуваних трас

Для аналізу було використано три основні типи трас, отриманих з системи управління ресурсами (Resource Management System) та планувальника завдань (Job Scheduler) досліджуваних кластерів.

Траса виділення ресурсів являє собою збірник операційних журналів диспетчера ресурсів за тривалий період (наприклад, 5 місяців експлуатації). Вона містить інформацію про виділені ресурси для користувацьких завдань, включаючи:

- Час виконання (Walltime);

- Виділення обчислювальних вузлів;

- Кількість процесорів (ядер) на вузол;

- Використання CPU, що визначається як сумарний час, витрачений усіма виділеними ядрами на виконання користувацького коду.

Траса була розділена для диференційованого аналізу завдань, що виконуються на загальній черзі (CPU-лише), та завдань, що виконуються на черзі з підтримкою GPU. Важливо зазначити, що, оскільки ця траса фіксує лише виділені ресурси, вона не розрізняє стани "система вимкнена" та "система простоює, але доступна" (idle state).

Траса використання GPU - є короткостроковим (наприклад, тижневим) зрізом, зібраним спеціально для кількісної оцінки використання GPU на вузлах з прискорювачами. Збір даних здійснювався за допомогою:

- планувальника для отримання статистики щодо виділення процесорів під завдання.

- інтерфейсу управління системою NVidia (компонент CUDA SDK) для безпосереднього вимірювання метрики використання GPU.

Показник використання GPU виражався як відсоткова частка часу, протягом якого GPU був активно задіяний у виконанні користувацьких обчислювальних ядер (User Kernels).

Траса планувальника – це остання траса — це короткостроковий зріз даних з планувальника, зібраний на кластері, що перебуває на етапі тестування (Pre-Production System) або з обмеженим доступом. Вона містить дані про виділення ресурсів та час виконання завдань. Порівняння цієї траси з даними продуктивних систем дозволяє ідентифікувати відмінності у характері навантаження між тестовими (Early-Access) та повноцінними продуктивними HPC-системами.

Рисунок 2.1 детально ілюструє конвеєри аналізу трас — послідовність етапів, використаних для трансформації різнорідних журналів та трас у стандартизований формат придатних для аналізу даних.

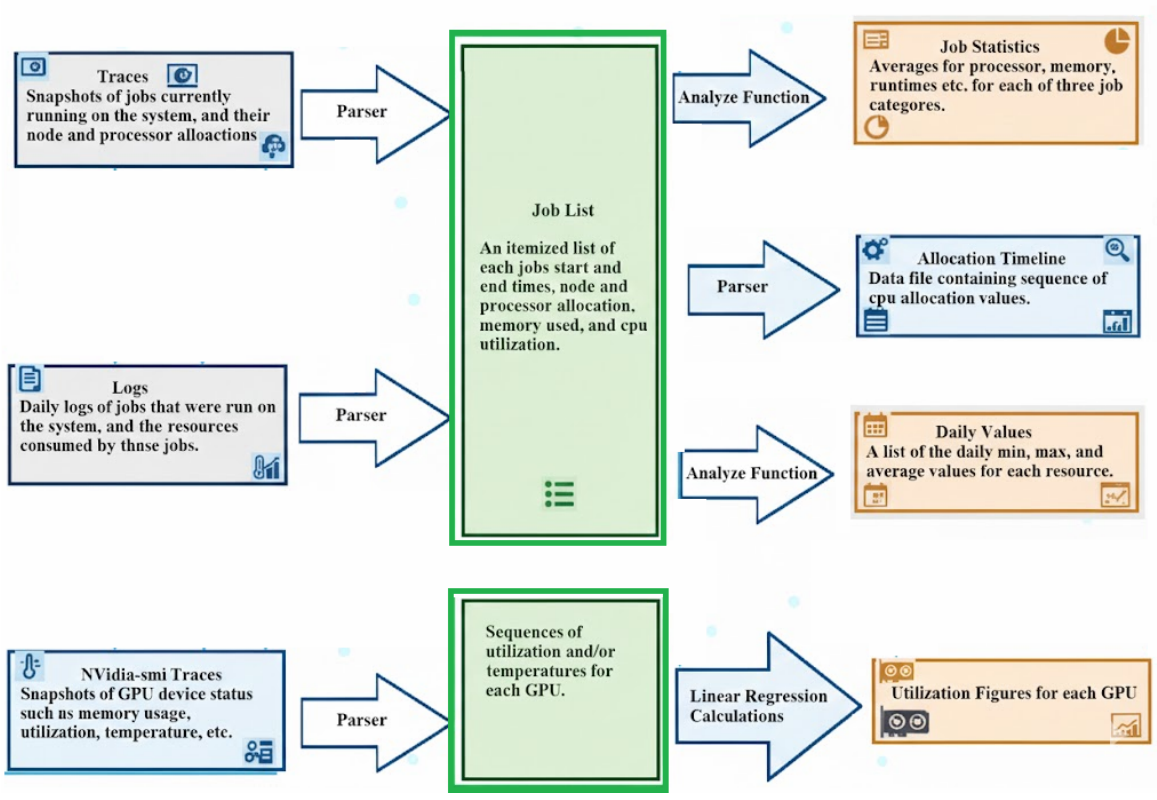

Рис. 2.1. Конвеєри аналізу трас

2.3. Характеристика робочого навантаження на загальнопризначеній HPC-системі

2.3.1. Аналіз журналів диспетчера ресурсів

Траса, зібрана з журналів диспетчера ресурсів загальнопризначеної черги, охоплює понад 37 тисяч завдань за період спостереження. Кількісний аналіз цих даних дозволяє встановити патерни використання ресурсів та типові характеристики обчислювальних завдань.

На рисунку 2.2 проілюстровано динаміку виділення ядер CPU протягом усього певного періоду моніторингу.

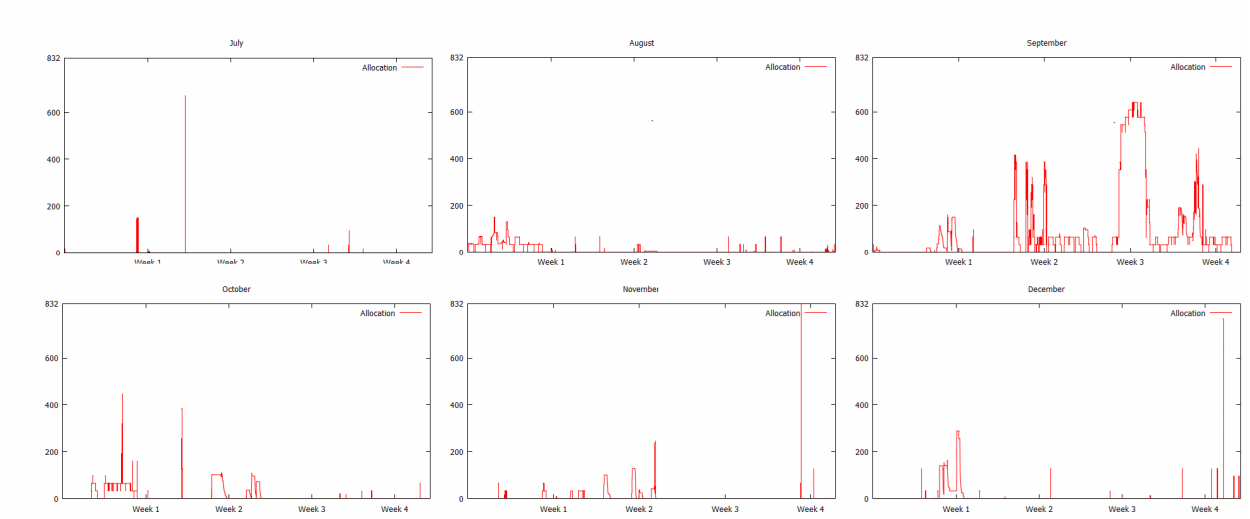

Рис. 2.2. Виділення CPU у загальній черзі системи класу 1

Спостерігаються чіткі фази використання:

1. Початкова фаза.

Протягом першого місяця функціонування системи фіксується мінімальне використання (Underutilization). Більшість поданих завдань є невеликими, а система перебуває у стані низької завантаженості.

2. Фаза зростання (активація).

Спостерігається різке зростання використання, що встановлює регулярний патерн завантаження, який періодично досягає понад 80%

доступних ресурсів. Це зростання часто корелює з початком академічного або дослідницького циклу.

3. Сезонні коливання.

Використання має тенденцію до зниження наприкінці академічного року та під час літніх місяців, з наступним відновленням активності.

4. Спорадичні піки.

Протягом деяких періодів спостерігаються випадкові, але інтенсивні піки активності, за якими слідують тривалі періоди низького використання або повної бездіяльності.

Загальний час активного використання (Active Utilization Time), тобто час, протягом якого було виділено принаймні одне ядро, становить 25% від загального часу моніторингу.

Таблиця 2.1 представляє статистичні показники для завдань, класифікованих за їхньою тривалістю (секунди, хвилини, години).

Таблиця 2.1.

Статистика завдань загальнопризначеної черги

Категорія завдань	**Кількість завдань**	**Середній час виконання (сек)**	**Середня кількість процесорів**	**Середнє використання пам'яті (ГБ)**
Секунди	12001	142 (σ = 99.2)	10 (σ =57.6)	19 (σ =237)
Хвилини	8814	1608 (σ =1086)	11 (σ =59.1)	28 (σ =277)
Години	13865	15155 (σ =21322)	4 (σ =16.7)	4 (σ =22.4)
> 8 Годин	1855	-	-	-

Незважаючи на високе стандартне відхилення (σ), середній розмір завдання (кількість виділених процесорів) є відносно малим для всіх трьох категорій, не досягаючи повного обсягу ядер одного обчислювального вузла. Це свідчить про те, що переважна більшість завдань значно менша за встановлені жорсткі обмеження на кількість ядер, доступних для

користувачів. Як показано на рисунку 2.3, дисперсія завдань концентрується навколо невеликих кластерів (менше 100 ядер).

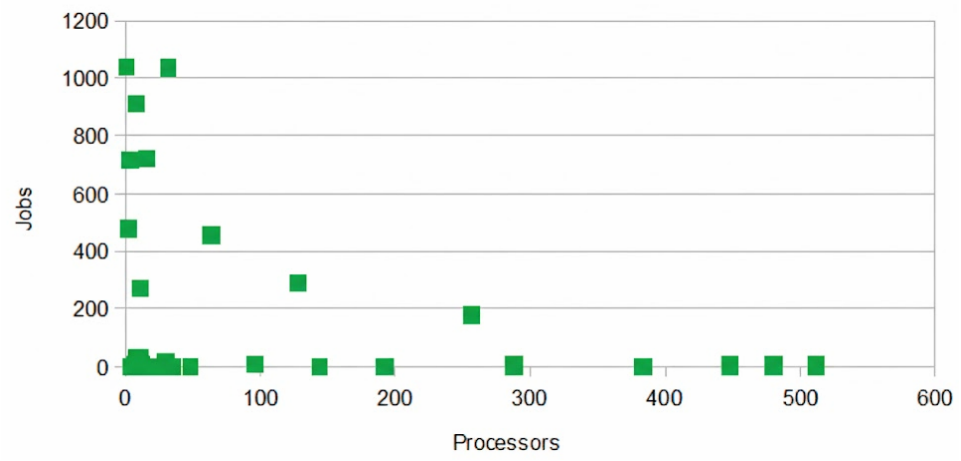

Рис. 2.3. Кількість проти розміру завдань у загальній черзі системи класу 1

Проведемо аналіз використання пам'яті. Тут спостерігається цікава аномалія: завдання категорії "Хвилини" демонструють пропорційно найбільше використання пам'яті порівняно з іншими категоріями. Завдання "Секунди" використовують приблизно 2 ГБ на ядро (що відповідає системному наданню), тоді як завдання "Години" використовують ще менше, близько 1,2 ГБ на ядро. Це може вказувати на те, що завдання середньої тривалості часто є інтенсивними за пам'яттю (Memory-Bound).

2.3.2. Аналіз еволюції використання системи

Для кращого розуміння еволюції використання системи було виключено початкові місяці з мінімальною активністю, що підвищило загальний час активного використання до 29,2%. Були обчислені щоденні мінімальні, максимальні та середні показники виділення CPU (рисунок 2.4).

Результати показують, що хоча система часто переживає короткочасні піки, які споживають майже всі доступні ядра, ці піки є недостатньо тривалими, щоб істотно вплинути на середнє щоденне виділення. Лише у виняткових випадках середнє виділення наближається або перевищує 50%.

Цей факт підкреслює, що системні ресурси часто запитуються невеликими порціями, що призводить до недостатнього використання загальної доступної потужності.

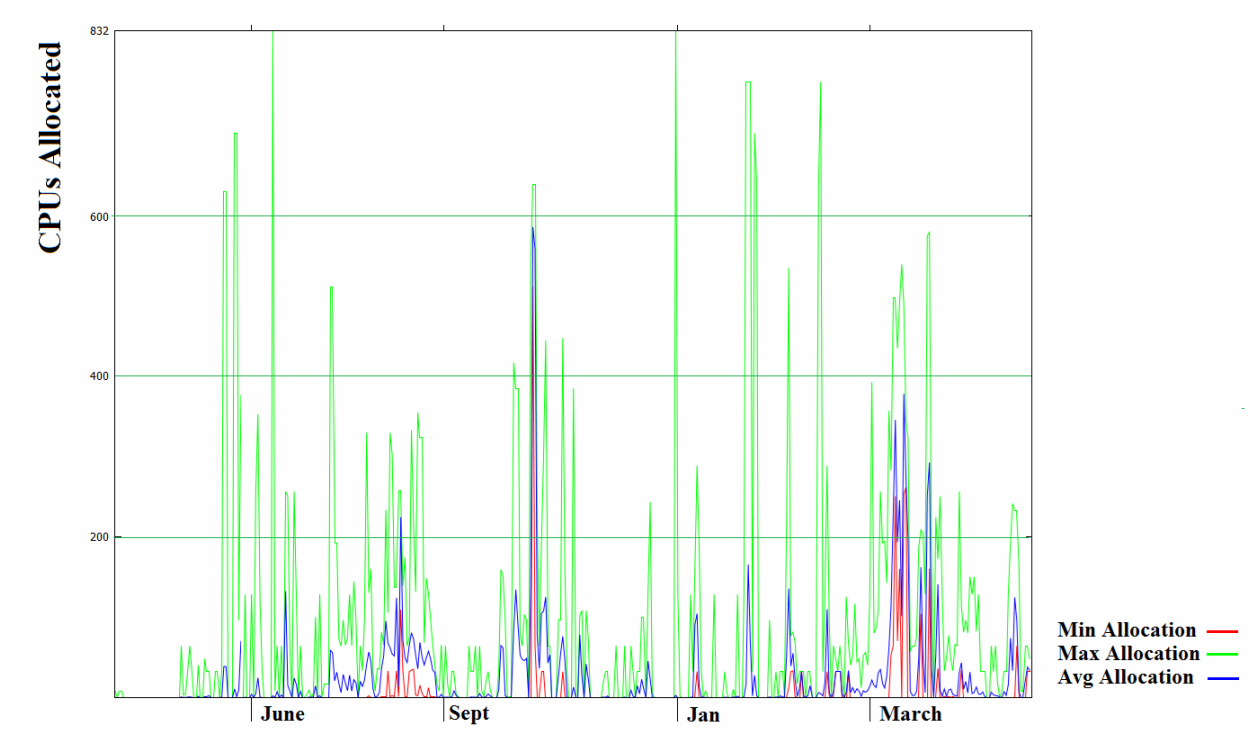

Рис. 2.4. Щоденне виділення CPU

Для оцінки фактичного використання системи (Resource Efficiency) було обчислено середнє щоденне використання на основі ефективної кількості ядер, задіяних у виконанні користувацького коду.

$$\textit{Ефективні CPU} = \textit{Час Виконання} \times \textit{Кількість Запитуваних Процесорів}$$

$$\textit{Ефективність Використання} = (\textit{Час Використання CPU} / \textit{Ефективні CPU}) \times 100\%$$

$$\textit{Ефективна Кількість Ядер} = \textit{Ефективність Використання} \times \textit{Кількість Запитуваних Ядер}$$

Рисунок 2.5 порівнює середнє щоденне виділення та середнє щоденне використання.

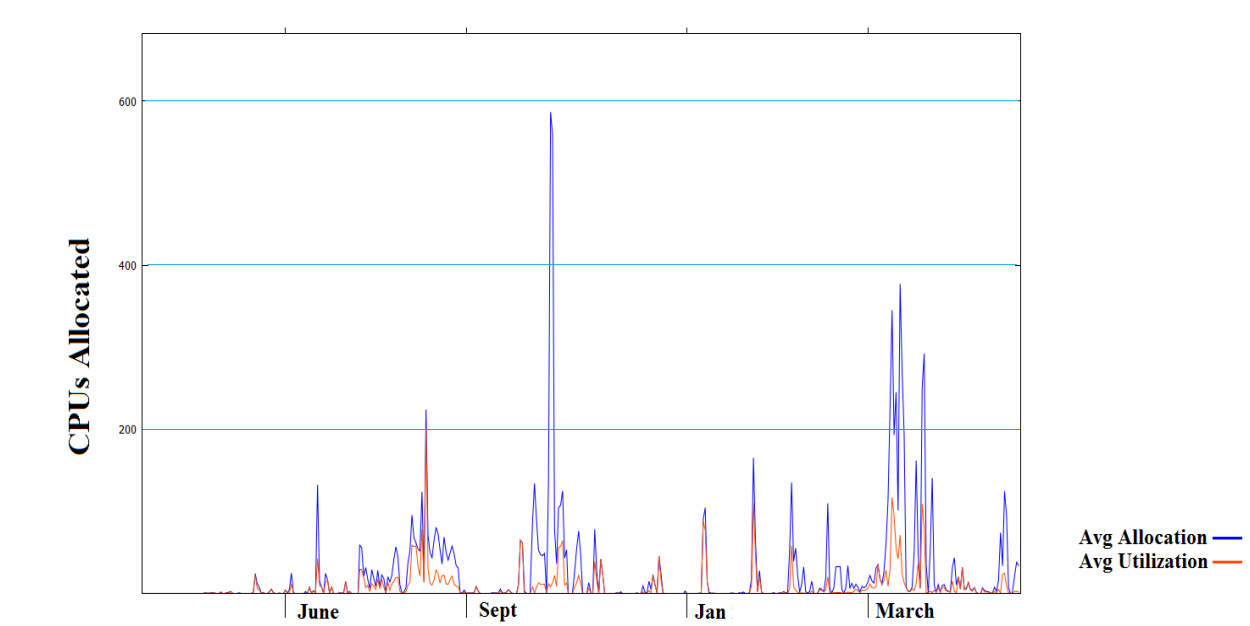

Рис. 2.5. Щоденне виділення CPU по відношенню до загального використання CPU

Виявлено, що середнє використання зазвичай є значно нижчим за середнє виділення. Хоча цей розрив може частково пояснюватися факторами, що не контролюються операторами системи (наприклад, неефективна оптимізація коду користувачів), це має важливі наслідки для стратегій надання системних ресурсів та політик планування.

Отже, навантаження на загальнопризначену чергу переважно складається з невеликих завдань (12 ядер або менше) з переважною тривалістю менше восьми годин. Використання пам'яті в основному відповідає наданим 2 ГБ / ядро.

Запити на системні ресурси є спорадичними, з тривалими періодами повної бездіяльності, а виділені ресурси часто використовуються не повністю.

2.4. Дослідження робочого навантаження на гетерогенному GPU-кластері

2.4.1. Аналіз журналів диспетчера ресурсів

Цей розділ аналізує дані, отримані з тієї вибірки журналів диспетчера ресурсів, що й у попередньому розділі, але сфокусовано на черзі, що підтримує графічні процесори.

Траса охоплює біля 6 тисяч завдань, а зафіксований час активного використання ресурсів (Active Utilization Time) становить 15%. Хоча ці журнали фіксують лише виділення ядер CPU і не містять даних про використання самих GPU, вони слугують важливим індикатором загальної активності в гетерогенній підсистемі. Слід враховувати, що ця черга є меншою за загальнопризначену, і складається з 512 ядер на 16 вузлах.

На рисунку 2.6 проілюстровано виділення ядер CPU на вузлах, оснащених GPU, протягом періоду спостереження.

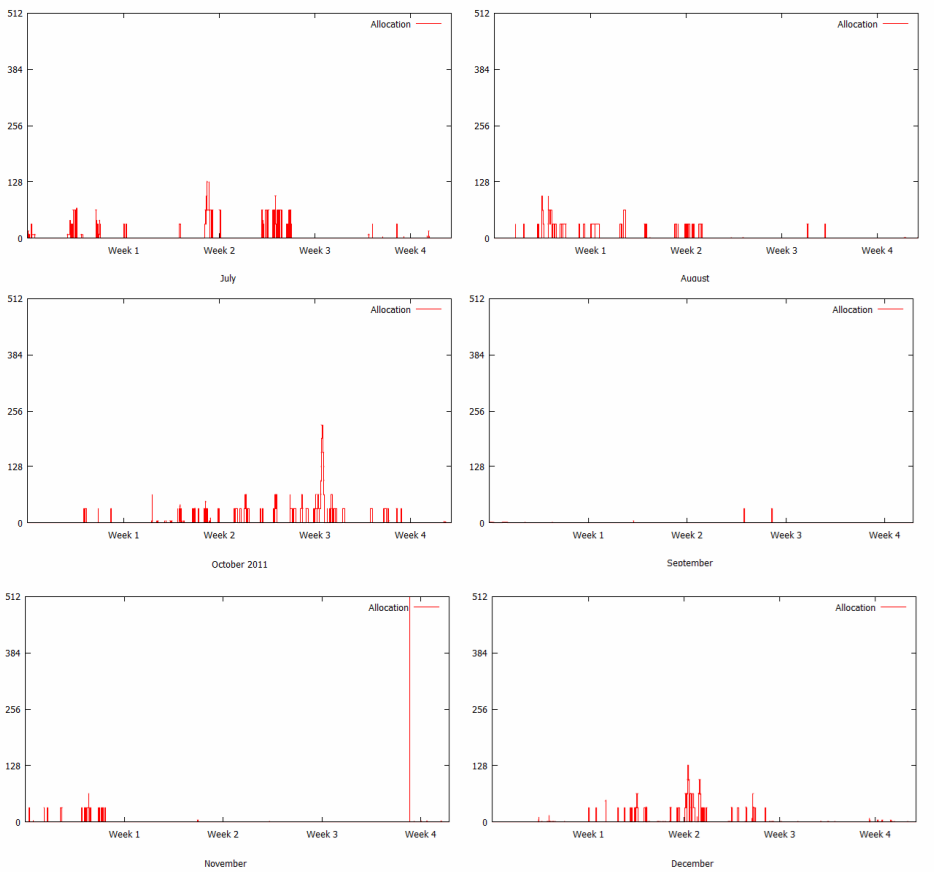

Рис. 2.6. Динаміка виділення ядер CPU на вузлах

На відміну від загальнопризначеної черги, черга GPU демонструє значне використання вже на ранніх етапах експлуатації системи, що свідчить про негайний попит на прискорені обчислення.

Щодо періоду інтенсивного використання, то спостерігається досить постійне та інтенсивне використання протягом перших шести місяців, після чого активність починає знижуватися. За винятком поодиноких піків, використання черги GPU стає надзвичайно рідкісним протягом тривалого часу. Надалі фіксуються лише спорадичні періоди низької або середньої активності, що тривають до кінця траси.

Ця динаміка свідчить про менш послідовне використання ресурсів у гетерогенній підсистемі, де періоди високого попиту чергуються з тривалими фазами низької завантаженості.

Статистика завдань у черзі GPU, класифікованих за часом виконання, представлена у таблиці 2.2.

Таблиця 2.2.

Статистика завдань черги з прискорювачами

Категорія Завдань	**Кількість Завдань**	**Середній Час Виконання (сек)**	**Середня Кількість Процесорів**
Секунди	2606	120 (σ =81)	20 (σ =47)
Хвилини	2333	998 (σ =758)	36 (σ =57)
Години	599	16210 (σ =24260)	23 (σ =38)

Проведемо аналіз розподілу тривалості. Спостерігається значно менша частка довготривалих завдань (категорія "Години") порівняно із загальнопризначеною чергою. Це вказує на те, що користувачі гетерогенної черги переважно виконують короткочасні або середньотривалі обчислення. Середній час виконання секундних та хвилинних завдань є порівнянним із загальною чергою, тоді як годинні завдання мають дещо більший середній час виконання.

Середня кількість виділених процесорів на завдання є помітно більшою у всіх категоріях порівняно з аналогічними завданнями в загальнопризначеній черзі. Цей висновок є контрінтуїтивним, оскільки:

- Черга GPU має менший загальний обсяг ядер.

- Встановлені обмеження на максимальну кількість ядер для одного користувача часто є суворішими або меншими у черзі GPU.

Можна було б очікувати, що завдання у більш спеціалізованій та обмеженій черзі будуть меншими. Це аномальне явище потребує подальшого дослідження у контексті фактичного використання GPU, оскільки більший розмір завдання за CPU може бути пов'язаний з певними патернами гетерогенного програмування або вимогами до даних.

Отже, черга з прискорювачами демонструє менш стабільний патерн використання з тривалими періодами низької активності після початкового інтенсивного використання. Завдання тут переважно короткочасні, і, незважаючи на менший обсяг ресурсів у черзі, вони вимагають більшої середньої кількості ядер CPU на завдання, ніж завдання у загальнопризначеній черзі.

2.5. Аналіз використання графічних процесорів у гетерогенних HPC-системах

Обчислення загального призначення на графічних процесорах (GPGPU) є однією з ключових тенденцій у високопродуктивних обчисленнях (HPC), що викликала значний інтерес у дослідницькій спільноті. Метою даного аналізу є кількісна оцінка фактичного впровадження та використання GPU вченими різних галузей для задоволення їхніх суперкомп'ютерних потреб.

Для цього було зібрано тижневу трасу використання на вузлах, призначених для обробки завдань з прискорювачами. Важливо зазначити, що через операційні обмеження або технічне обслуговування, дані представляють меншу кількість GPU (наприклад, 30 пристроїв на 15 вузлах).

2.5.1. Збір даних та методологія

Збір даних про використання здійснювався за допомогою інтерфейсу управління системою NVidia (nvidia-smi). Це утиліта командного рядка, що входить до складу комплекту розробника CUDA та дозволяє моніторинг і управління GPU. Ключовий показник — використання GPU — виражався як відсоткова частка часу, протягом якого пристрій активно виконував користувацькі обчислювальні ядра. Статистика щодо виділення CPU та параметрів завдань збиралася з планувальника.

Продуктивний характер HPC-системи та, як наслідок, консервативна політика оновлення програмного забезпечення ускладнювали збір даних. На момент моніторингу вузли працювали з різними версіями програмного стека (наприклад, CUDA 3.1, 3.2, 4.0), причому явна підтримка збору детальних даних про використання була інтегрована лише у новіших версіях (наприклад, з CUDA 3.2). Це обмежувало пряме отримання даних про використання лише частиною GPU (наприклад, 8 пристроїв), тоді як дані про температуру були доступні для всіх пристроїв.

Спостерігається пряма кореляція між температурою GPU та його фактичним використанням. Середні температури простою були різними для GPU, встановлених у різних слотах (наприклад, 58◦C у слоті 0 та 51◦C у слоті 1). Теоретично, наявність даних про температуру для всіх пристроїв та явних даних про використання для підмножини дозволяє розрахувати кореляційну функцію для екстраполяції використання. Однак, через характер отриманих даних, розрахунок точних функціональних рівнянь був неможливий.

2.5.2. Аналіз результатів

Розглянемо динаміку виділення CPU. Рисунок 2.7 ілюструє виділення CPU на вузлах черги GPU протягом тижневої траси. Черга демонструє активне виділення ресурсів, причому більшість спожитих ресурсів була задіяна серією великих завдань, ініційованих обмеженою кількістю

унікальних користувачів (наприклад, двома). Паралельно спостерігалася подача колекції менших, короткотривалих завдань.

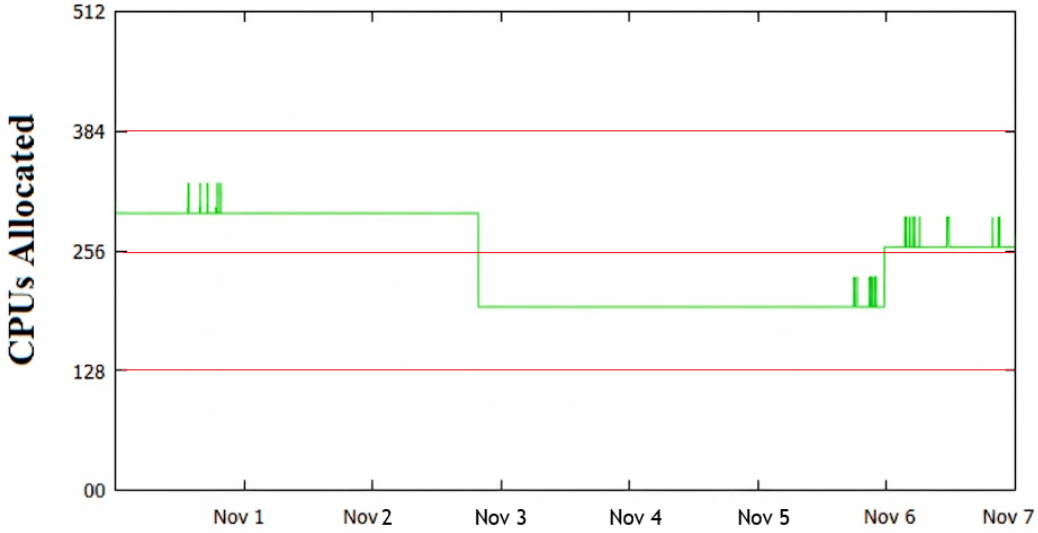

Рис. 2.7. Виділення CPU у черзі GPU системи класу 1, траса використання

Щодо фактичного використання GPU, то рисунок 2.8 демонструє середнє використання 8 GPU, для яких були доступні явні дані, а рисунок 2.9 показує усереднені температури GPU в різних слотах протягом траси.

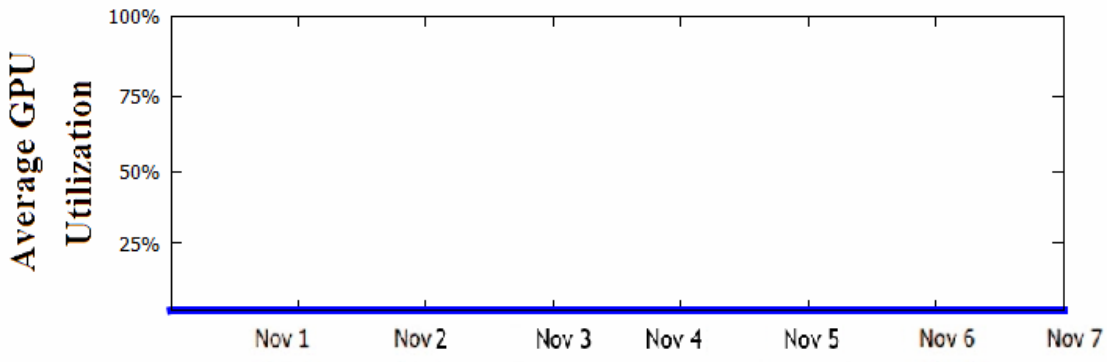

Рис. 2.8. Середнє використання GPU

Незважаючи на активне виділення ядер CPU та виконання завдань протягом тижня, GPU були суттєво невикористані (фактичне використання

наближалося до нуля). Це є значним висновком, що вказує на проблему адаптації новітніх обчислювальних парадигм.

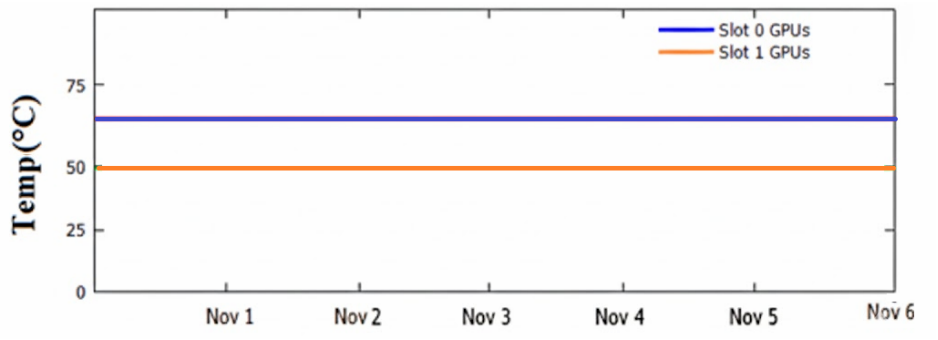

Рис. 2.9. Середні значення температури GPU в різних слотах

Основними потенційними причинами недовантаження GPU є:

1. Бар'єр програмування.

На відміну від традиційного програмування, паралельне програмування GPGPU є іншою, часто більш складною парадигмою, яка вимагає спеціалізованих знань (наприклад, мови та моделі CUDA/OpenCL). Цільова аудиторія користувачів HPC-систем, яка часто складається з дослідників у галузях ядерної інженерії, хімічної інженерії, фізики та математики, може не володіти необхідними навичками для ефективної розробки та адаптації коду для GPU.

2. Відсутність готових додатків.

Дослідники можуть використовувати комерційне або галузеве програмне забезпечення, яке ще не оптимізоване або не підтримує прискорення на GPU.

Отже, хоча GPGPU пропонує унікальні можливості для прискорення обчислень, що призвело до його стрімкого розвитку в комп'ютерній науці, фактичне використання GPU в гетерогенній підсистемі дослідниками з різних наукових дисциплін наразі є вкрай низьким. Ця критична проблема недовикористання вказує на необхідність цільових освітніх програм та

політик стимулювання для підвищення ефективності використання високоцінних прискорених ресурсів.

2.6. Порівняльний аналіз робочого навантаження з використанням тестового середовища

З метою порівняння характеристик робочого навантаження на продуктивній системі (наприклад, системі класу 1) з активністю у тестовому/попередньо-продуктивному середовищі (Pre-Production System), було зібрано тижневу трасу користувацьких завдань з планувальника, що функціонує на тестовому GPU-кластері (кластер класу 2).

Оскільки дані були отримані з інструменту користувацького простору, вони включають лише статистику щодо виділення ядер CPU та часу виконання (Walltime). Ця траса використовується насамперед для оцінки відносних рівнів активності та типології завдань, що виконуються на двох архітектурно різних системах.

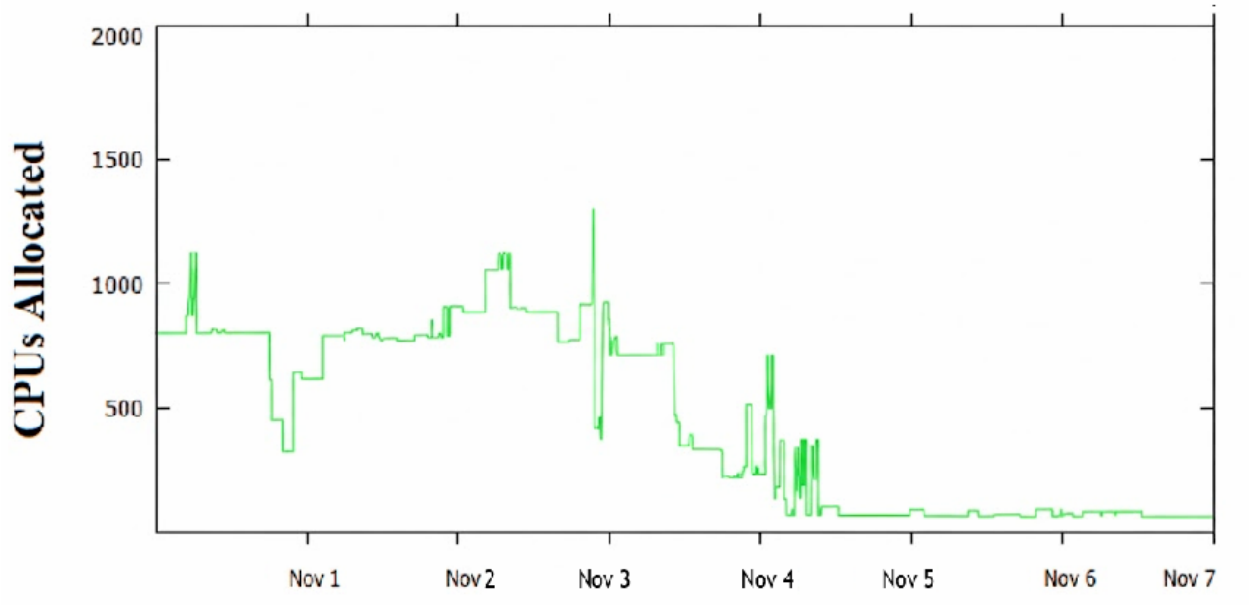

Рис. 2.10. Динаміка виділення ядер CPU на тестовому кластері протягом тижневого періоду

Рисунок 2.10 ілюструє виділення ядер CPU на тестовому кластері протягом тижневого періоду моніторингу. Незважаючи на свій попередньо-продуктивний статус, система демонструє постійне використання протягом усього періоду траси.

Фаза змінності. У першій половині періоду спостереження виділення ресурсів швидко коливається.

Фаза стабілізації. Протягом останніх трьох днів використання стабілізується.

Загалом за тиждень було зафіксовано біля 200 користувацьких завдань.

Таблиця 2.3 представляє статистику завдань, класифікованих за часом виконання (секунди, хвилини, години).

Таблиця 2.3.

Статистика завдань тестового кластера

Категорія завдань	**Кількість завдань**	**Середній час виконання (сек)**	**Середня кількість процесорів**
Секунди	75	120 (σ =105)	20.8 (σ =47.0)
Хвилини	76	960 (σ =726)	37.2 (σ =57.9)
Години	59	114240 (σ =122078)	24.7 (σ =38.2)

Ключові відмінності від продуктивної системи (Класу 1):

1. Розмір завдань (кількість CPU).

Завдання всіх категорій на тестовому кластері вимагають значно більшої кількості процесорних ядер порівняно з продуктивною системою. Особливо виділяються довготривалі завдання ("Години"), де середнє завдання використовує більш ніж у шість разів більше ядер, ніж аналогічне завдання на продуктивній системі.

2. Тривалість завдань.

Середній час виконання довготривалих завдань на тестовому кластері є набагато більшим (у середньому в сім разів), ніж на продуктивній системі. Цей факт є парадоксальним, оскільки максимальне обмеження часу

виконання (Walltime Limit) у довгій черзі тестового кластера становить лише 144 години, тоді як у загальній черзі продуктивної системи — 300 годин.

Отже, тестовий кластер демонструє постійний, хоча й помірний, рівень використання (виділення менше 50% протягом траси). Навантаження характеризується значно більшими потребами в ресурсах (більше ядер та довший час виконання) у порівнянні з продуктивною системою. Це може вказувати на те, що:

- на тестовому кластері виконуються великомасштабні, оптимізовані застосунки, що вимагають більшого паралелізму.

- користувачі, які мають доступ до тестового середовища (наприклад, ключові дослідники), виконують більш обчислювально інтенсивні експерименти.

Висновки до розділу

У другому розділі було проведено ґрунтовний аналіз методології дослідження робочого навантаження, що дозволило сформувати чіткі підходи до збору, обробки та інтерпретації даних з реальних HPC-систем. Дослідження трас показало, що робоче навантаження формується під впливом широкого спектра чинників, серед яких ключову роль відіграють тип задач та інтенсивність їх подання. Аналіз журналів диспетчера ресурсів дав можливість виявити характерні патерни поведінки користувачів, включно з піковими періодами активності та нерівномірністю використання ресурсів. Було встановлено, що загальнопризначені HPC-системи демонструють суттєві коливання навантаження, що вказує на необхідність гнучкого підходу до планування. Також дослідження показало, що гетерогенні GPU-кластери мають власну специфіку, зокрема залежність продуктивності від типу задач і рівня оптимізації програмного коду. Аналіз використання графічних процесорів підтвердив існування значного дисбалансу між доступними ресурсами та фактичним попитом на них. Збір і

опрацювання телеметрії GPU дозволили визначити, що навіть за високої середньої завантаженості існують періоди недовикористання прискорювачів. Порівняння різних HPC-середовищ показало, що ефективність роботи системи значною мірою залежить від структури поданих задач, а не лише від характеристик апаратного забезпечення.

РОЗДІЛ 3. РОЗРОБКА МЕТОДОЛОГІЇ АНАЛІЗУ ОДНОРІДНИХ ТА НЕОДНОРІДНИХ НАВАНТАЖЕНЬ В HPC-СИСТЕМАХ

3.1. Опис етапів пропонованої методології

На основі емпіричних даних та проведеного аналізу трас пропонується комплексна методологія дослідження робочого навантаження для високопродуктивних обчислювальних систем. Ця методологія враховує специфіку як однорідних (гомогенних, CPU-орієнтованих), так і неоднорідних (гетерогенних, з використанням прискорювачів) обчислювальних середовищ. Метою методології є створення точних статистичних профілів користувачів для оптимізації надання ресурсів та планування.

3.1.1. Багаторівнева архітектура збору даних

Першим етапом методології є впровадження ієрархічної системи моніторингу, яка здатна захоплювати дані на різних рівнях абстракції системи. Враховуючи виявлені розбіжності між виділенням (allocation) та використанням (utilization), критично важливим є розділення потоків даних:

1. Рівень планувальника (Scheduler Level):

Джерело: журнали систем управління ресурсами.

Цільові метрики: час подання завдання (T_{submit}), запитуваний час виконання ($T_{walltime}$), кількість запитуваних вузлів/ядер (N_{cores}), статус завершення.

Призначення: аналіз однорідних навантажень та адміністративних політик черг.

2. Апаратний рівень (Hardware Level):

Джерело: інструменти профілювання "in-band" (наприклад, nvidia-smi).

Цільові метрики: фактичне завантаження CPU (%), завантаження GPU (%), температура кристала, використання пам'яті (VRAM/RAM).

Призначення: аналіз неоднорідних навантажень та оцінка ефективності використання прискорювачів.

3.1.2. Критерії диференціації та класифікація навантажень

Для коректного моделювання необхідно застосувати процедуру кластеризації завдань. Методологія передбачає класифікацію за двома векторами:

1. Вектор ресурсоємності (Resource Intensity):

Однорідні завдання характеризуються високою кореляцією між виділеними ядрами та використанням пам'яті. Основний обмежувальний фактор — пропускна здатність пам'яті або тактова частота CPU.

Неоднорідні завдання характеризуються асиметричним навантаженням. Виявлено патерн "High CPU Allocation / Low GPU Utilization", що вимагає окремої категорії для завдань, які блокують ресурси прискорювачів без їх активного використання (Idle Holding).

2. Вектор часової шкали (Temporal Scale):

Завдання категорії "Секунди" (Short-Burst): <5 хвилин. Слугують індикаторами налагодження коду або помилок конфігурації.

Завдання категорії "Хвилини" (Medium): 5 хв−1 година.

Завдання категорії "Години" (Long-Running): >1 години. Формують основне корисне навантаження системи.

3.1.3. Статистичне профілювання та виявлення аномалій

Етап аналізу передбачає розрахунок статистичних моментів для кожної категорії навантаження. Для однорідних навантажень ключовим показником ефективності (E_{hom}) є відношення часу виконання користувацького коду до зарезервованого часу:

$$E_{hom} = \frac{\sum(T_{CPU\_active})}{\sum(N_{cores} \times T_{walltime})}$$

Для неоднорідних навантажень вводиться Коефіцієнт Ефективності Прискорення (E_{het}), який враховує простій GPU:

$$E_{het} = \alpha \cdot U_{GPU} + (1 - \alpha) \cdot U_{CPU}$$

де U — усереднене використання ресурсу, а α — ваговий коефіцієнт, що визначає пріоритетність використання дорогих ресурсів (GPU). Низьке значення E_{het} при високому запиті ресурсів сигналізує про необхідність втручання (навчання користувачів або оптимізація коду), як це було виявлено у випадку з низьким використанням CUDA-ядер. Ці ядра є ключовим компонентом архітектури паралельних обчислень CUDA (Compute Unified Device Architecture), яка дозволяє використовувати GPU не лише для обробки графіки, а й для загальних високопродуктивних обчислень (GPGPU).

3.1.4. Синтез навантаження для валідації систем

Завершальним етапом методології є генерація синтетичних трас. На відміну від простого відтворення історичних даних, пропонується використовувати стохастичну модель генерації, яка базується на визначених розподілах ймовірностей:

1. Моделювання прибуття завдань. Використання розподілу Пуассона для імітації часу між надходженням завдань (λ) для кожної категорії (секунди, хвилини, години).

2. Параметризація ресурсів. Призначення атрибутів (кількість ядер, пам'ять) на основі нормального розподілу (μ,σ), отриманого з емпіричного аналізу пікових періодів.

Ця методологія дозволяє перейти від реактивного аналізу логів до проактивного моделювання (Simulative Forecasting), що є критично важливим для обґрунтування інвестицій у майбутні гетерогенні архітектури та запобігання проблемам недостатнього використання ресурсів.

3.2. Представлення архітектури потоку даних методології

Для ефективної реалізації методології аналізу пропонується конвеєрна архітектура обробки даних (Data Processing Pipeline), що складається з чотирьох послідовних фаз. Цей процес забезпечує перетворення "сирих" логів у валідовані моделі навантаження.

Фаза 1. Агрегація та Синхронізація (Data Ingestion & Synchronization). Процес розпочинається з паралельного збору даних із двох незалежних джерел. Критичним етапом тут є часова синхронізація (Time_Sync), оскільки системні логи та апаратна телеметрія можуть мати різні частоти дискретизації.

Вхід: логи планувальника (події submit/start/end) + Часові ряди апаратних сенсорів (CSV/RRD).

Дія: нормалізація часових міток та об'єднання даних у єдиний запис (Job Record), де кожному ID завдання відповідає профіль використання ресурсів.

Фаза 2. Класифікація та векторизація (Classification & Vectorization). Об'єднані записи проходять через фільтр класифікації. Система визначає тип навантаження на основі запитаних ресурсів (R_{req}).

Логіка наступна: якщо R_{req}(GPU) > 0, завдання маркується як Герогенне. В іншому випадку — Однорідне.

Далі відбувається розподіл за часовими категоріями (секунди, хвилини, години) для формування окремих статистичних вибірок.

Фаза 3. профілювання ефективності (Efficiency Profiling). Для кожного класу обчислюються метрики ефективності.

Для однорідних: аналіз співвідношення Requested CPU vs Used CPU. Виявлення "зомбі-процесів" (високий walltime, нульове навантаження).

Для неоднорідних: кореляційний аналіз. Порівняння часу утримання GPU з часом активного виконання ядер (Kernel Execution Time). Це дозволяє виявити неефективний код, який блокує прискорювачі.

Фаза 4. Стохастичне моделювання (Stochastic Modeling). На фінальному етапі емпіричні дані конвертуються у ймовірнісні розподіли (λ для частоти, μ,σ для ресурсів).

Вихід: генерація файлу синтетичного навантаження (Synthetic Workload Trace), придатного для симуляторів (наприклад, для симулятора Alea або SimGrid).

Нижче наведена блок-схема, що візуалізує алгоритм методології, демонструючи розгалуження аналізу для різних типів навантажень та інтеграцію результатів у синтетичну модель.

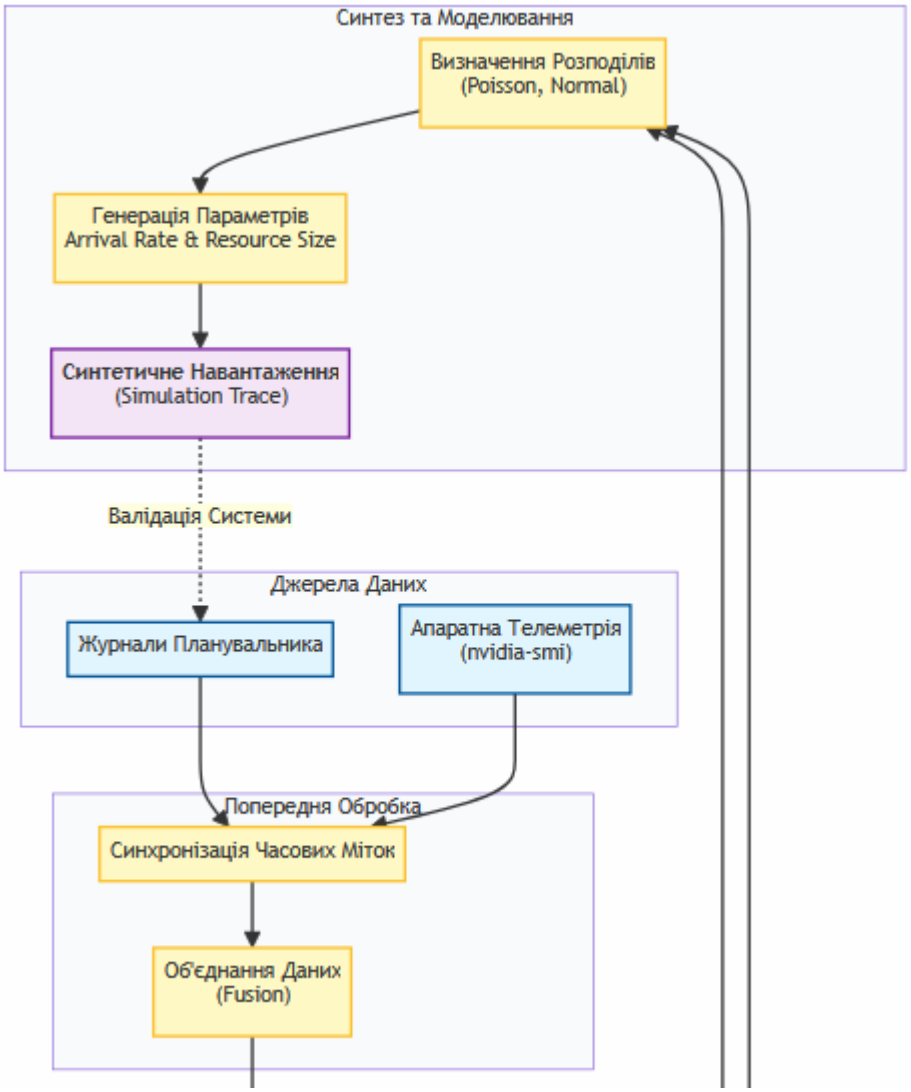

Рис. 3.1. Алгоритм пропонованої методології аналізу однорідних та неоднорідних навантажень в HPC-системах (початок)

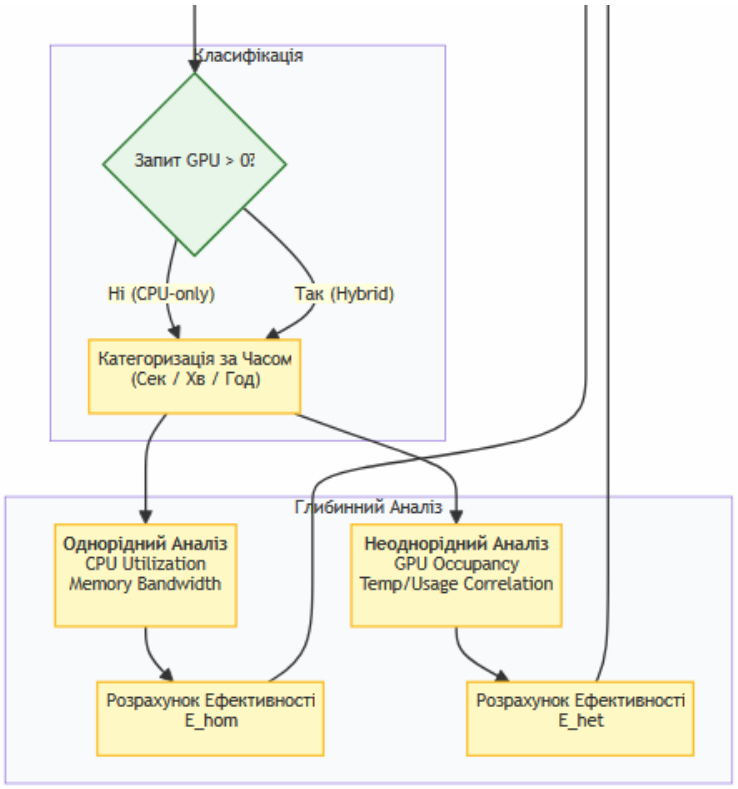

Рис. 3.1. Алгоритм пропонованої методології аналізу однорідних та неоднорідних навантажень в HPC-системах (завершення)

Цей алгоритм демонструє замкнутий цикл методології:

1. Ліва гілка (CPU-only) - фокусується на традиційних метриках пропускної здатності.

2. Права гілка (Hybrid) включає специфічні метрики для GPU, такі як кореляція температури та завантаження.

3. Фінальний блок показує, що метою аналізу є не просто звітність, а створення активного інструменту (синтетичного навантаження) для тестування майбутніх систем (Feedback Loop)

3.3. Методологія вибору репрезентативного періоду для синтез робочого навантаження

Синтез робочого навантаження (Workload Synthesis) є критичним етапом для валідації та бенчмаркінгу архітектури та політик планування

ресурсів майбутніх високопродуктивних обчислювальних систем (HPC). Для забезпечення релевантності синтезованого навантаження, необхідно обрати репрезентативний часовий проміжок з історичних трас експлуатації існуючих систем.

Критерії вибору часового проміжку:

1. Інтенсивність та постійність використання.

Необхідно ідентифікувати період, який характеризується найбільш інтенсивним та сталим рівнем використання ресурсів. Це дозволить моделювати пікове навантаження (Peak Load), яке має бути стійко підтримане новою системою.

2. Часова релевантність.

Обраний період повинен бути відносно недавнім у контексті загальної експлуатації системи. Це гарантує, що синтезоване навантаження точно відображає поточні обчислювальні потреби (Computational Demands) і характеристики завдань актуальної спільноти користувачів.

3. Масштаб.

З огляду на поточне спостереження щодо низького або середнього використання наявних HPC-ресурсів, обране пікове навантаження може бути використане для тестування гіпотетичного налаштування меншого масштабу, яке планується розгорнути.

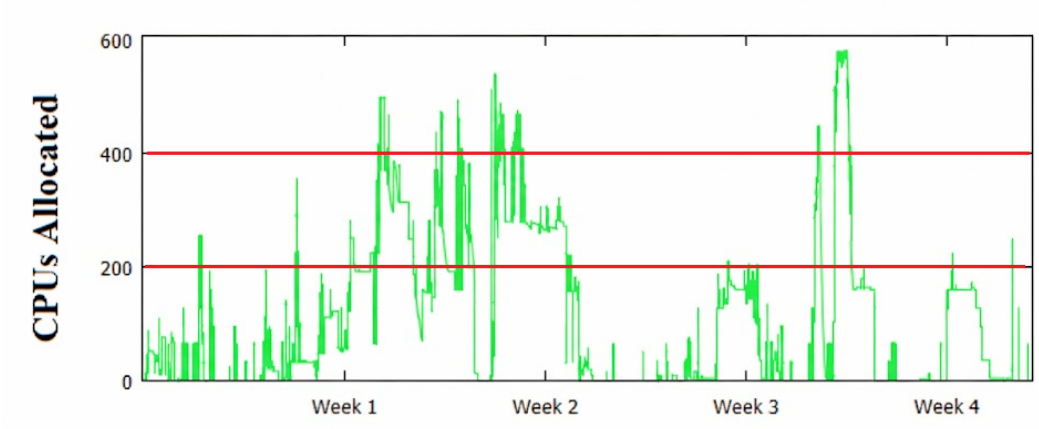

Рис. 3.2. Вибір часового проміжку для синтезу навантаження

З цією метою було обрано двотижневий проміжок з журналів диспетчера ресурсів загальнопризначеної черги продуктивної HPC-системи. Цей період продемонстрував оптимальне поєднання високої інтенсивності та часової близькості до моменту моделювання, що робить його найкращим кандидатом для створення синтетичного навантаження, що відображає гіпотетичний піковий сценарій для нових систем.

3.4. Статистична характеристика та параметри синтезу робочого навантаження

Після вибору найбільш репрезентативного часового проміжку пікового навантаження (15 днів), було проведено детальний статистичний аналіз цього періоду, аналогічний загальному аналізу трас, з метою параметризації синтетичного навантаження.

Завдання було розділено на ті самі три категорії (секунди, хвилини, години) для обчислення середнього часу виконання, кількості виділених процесорів та використаної пам'яті. Статистика, що характеризує навантаження у вибраний період, представлена у таблиці 3.1.

Таблиця 3.1.

Статистика завдань за обраний часовий проміжок

Категорія завдань	**Кількість завдань**	**Середній час виконання (сек)**	**Середня кількість процесорів**	**Середнє використання пам'яті (ГБ)**
Секунди	270	58 (σ=65.6)	30.7 (σ=56.8)	13 (σ=100.6)
Хвилини	623	1411 (σ=741.2)	12 (σ=31.8)	12 (σ=35.7)
Години	775	41655 (σ=36608)	4 (σ=15.1)	5 (σ=29.36)

Аналіз показує, що:

- довготривалі завдання ("години") становлять найбільшу частку (понад 45%) від загальної кількості завдань у піковий період.

- Завдання категорії "секунди" демонструють найбільший середній розмір за кількістю виділених процесорів, але мають найкоротшу середню тривалість.

Для створення достовірного синтетичного навантаження (Synthetic Workload), придатного для симуляційного тестування (Simulation-Based Testing), необхідна більш детальна інформація, яка відображає стохастичну природу використання системи. Було визначено два ключові додаткові параметри:

- Швидкість подачі завдань (arrival rate): для відображення цього стохастичного процесу, вибраний 15-денний період був розділений на 360 годинних інтервалів. Була обчислена середня кількість завдань, поданих на годину, для кожної з трьох категорій. Це дозволяє моделювати реалістичний потік запитів до планувальника.

- Питоме використання пам'яті: припускаючи, що використання пам'яті корелює з розміром завдання, а не є незалежною змінною, була обчислена середня кількість пам'яті, споживаної на одне ядро (ГБ/ядро), для кожної категорії.

Ці статистичні параметри, необхідні для генерації синтетичного навантаження змінної довжини, представлені у таблиці 3.2.

Таблиця 3.2.

Статистика для синтетичного навантаження

Категорія Завдань	**Середня кількість завдань/година**	**Середній час виконання (сек)**	**Середня кількість процесорів**	**Середня пам'ять/ядро (ГБ)**
Секунди	0.73 (σ=2.3)	51.8 (σ=65.6)	30 (σ=56.8)	0.543 (σ=3.98)
Хвилини	1.62 (σ=13.31)	1487 (σ=741.2)	12 (σ=31.8)	1.11 (σ=0.592)
Години	2.02 (σ=9.92)	40021 (σ=36501)	4 (σ=15.1)	1.24 (σ=0.535)

Ці параметри становлять повний набір статистичних даних, необхідних для створення синтетичного робочого навантаження для симуляційного

тестування потенційних майбутніх конфігурацій HPC-систем. Здатність точно прогнозувати ефективність (Performance Prediction) обслуговування унікального, специфічного для користувачів навантаження має надзвичайну цінність для організацій, які здійснюють значні інвестиції часу та фінансових ресурсів у придбання та розгортання нових HPC-платформ.

3.5. Рекомендації щодо політики оптимізації ресурсів у високопродуктивних обчислювальних системах

Цей розділ представляє набір практичних рекомендацій, спрямованих на вирішення проблем, пов'язаних із недостатнім використанням ресурсів (Underutilization) у високопродуктивних обчислювальних системах (HPC), а також на підвищення ефективності їхнього надання та планування. Недостатнє використання системи (наприклад, системи класу 1) до такої міри, що спостерігається, вказує на можливість значної економії фінансових ресурсів установи за рахунок зниження енергоспоживання та операційних витрат на обслуговування.

3.5.1. Оптимізація використання обчислювальних ресурсів (CPU)

Як було показано в попередніх розділах, середнє виділення ресурсів та фактичне використання виділених ядер CPU часто є дуже низькими відносно загальної потужності системи.

Стратегії покращення завантаженості:

1. Динамічне введення вузлів в експлуатацію (Power Gating):

- Виділити обмежену підмножину вузлів для постійної роботи (24/7), яка обслуговуватиме типове базове навантаження.

- Решту вузлів тримати в режимі низького енергоспоживання або вимкненими, вводячи їх в експлуатацію лише на вимогу (On-Demand) у разі виникнення потреби у виконанні великих або пакетних завдань.

2. Консолідація завдань через віртуалізацію (VM Consolidation):

- Через низький коефіцієнт використання виділених ресурсів CPU, робоче навантаження є ідеальним кандидатом для розміщення користувацьких завдань на віртуальних машинах (ВМ).

- Запуск кількох ВМ на одному фізичному вузлі значно підвищить загальне використання вузла. Окремі завдання навряд чи відчують негативний вплив на продуктивність через їхню низьку внутрішню ефективність використання ресурсів.

- Використання технологій міграції ВМ у реальному часі (Live Migration) дозволило б гнучко масштабувати доступні ресурси. Зі зростанням попиту можна активувати додаткові вузли, заморожувати ВМ з низькозавантажених вузлів, переносити їх на нові активні вузли та відновлювати (розморожувати) їхнє виконання.

3.5.2. Стимулювання використання GPU

Незважаючи на наявність вузлів з прискорювачами, фактичне використання GPU виявилося вкрай низьким. Для підготовки до розгортання майбутніх гетерогенних систем, критично важливо визначити причини такої недозавантаженості та навчити користувачів ефективно використовувати ці ресурси.

Пропозиції для технічного втручання:

- Надати користувачам, які не є фахівцями у комп'ютерних науках, необхідні навички для розробки або адаптації їхніх наукових застосунків для використання ресурсів GPU.

- Слід розглянути можливість перенесення (porting) існуючих, широко використовуваних застосунків, які зараз виконуються на CPU, для використання можливостей GPU.

Приклад. MATLAB є ідеальним кандидатом, оскільки завдання MATLAB становлять значну частину загального навантаження. Існують легко інтегровані бібліотеки для програмування GPU в MATLAB [15], які

дозволяють використовувати прискорювачі без глибокого знання специфіки низькорівневого GPGPU-програмування.

3.5.3. Рекомендації щодо надання майбутніх систем

На основі спостережених характеристик навантаження на існуючих системах (де домінує велика кількість невеликих завдань з низьким використанням ресурсів), рекомендується переглянути стратегію масштабування інфраструктури.

1. Масштабування вниз (Scaling Down).

Замість слідування тенденції до придбання все більших і більших кластерів, рекомендується масштабувати нові системи вниз за загальною кількістю вузлів. Основна мета масштабування вниз — оптимізація витрат та енергоспоживання. Коли попит на ресурси падає (наприклад, вночі або у вихідні), система автоматично або вручну зменшує кількість активних одиниць.

2. Оптимізація ядра.

Надавати перевагу кластерам з меншою кількістю вузлів, але оснащених меншою кількістю, але більш швидкими процесорними ядрами та/або новішими, більш ефективними архітектурами. Це краще обслуговуватиме потреби користувачів, які в основному подають невеликі завдання.

3. Консолідація та віртуалізація.

Менша, але швидша система залишається сумісною з вищезгаданою моделлю консолідації через ВМ, максимізуючи використання дорогих ресурсів.

4. Аналіз пам'яті.

Спостережуване використання пам'яті, що відповідає приблизно 2 ГБ на ядро, вказує на те, що подальше значне збільшення наданої пам'яті на ядро (наприклад, до 5 ГБ/ядро і більше) без радикальної зміни характеру

навантаження є фінансово необґрунтованим і призведе до хронічного недовикористання інвестицій.

3.6. Узагальнення по розділу і перспективи подальших досліджень

У цьому дослідженні було успішно здійснено збір та аналіз трьох трас робочого навантаження (Workload Traces), отриманих з продуктивних суперкомп'ютерних систем (Production HPC Systems). Проведений аналіз дозволив характеризувати патерни навантаження та визначити обчислювальні потреби науковців та дослідників із різних галузей знань.

Кількісно встановлено типові характеристики завдань (тривалість, розмір, використання ресурсів) на гетерогенних та загальнопризначених HPC-платформах.

На основі емпіричних трас було розроблено синтетичне робоче навантаження (Synthetic Workload), яке є репрезентативним для продуктивних систем, що використовуються науково-дослідною спільнотою. Це навантаження є цінним інструментом для симуляційного тестування (Simulation-Based Testing) та валідації архітектурних рішень майбутніх HPC-систем.

Сформульовані обґрунтовані рекомендації щодо політики, спрямовані на підвищення коефіцієнта використання (Utilization Rate) існуючих обчислювальних ресурсів, зокрема, через імплементацію стратегій віртуалізації та динамічного надання ресурсів. Також надані рекомендації стосовно оптимального масштабування (Scaling) та конфігурації майбутніх систем з метою кращого обслуговування специфічного навантаження користувачів.

Подальші дослідження будуть зосереджені на поглибленні аналізу навантаження та практичному застосуванні отриманих результатів:

1. Планується розширити аналіз на новіші або менш досліджені системи (наприклад, кластер класу 2). За умови співпраці з системними

адміністраторами, які наразі обмежують доступ до детальних журналів (через статус системи, що тестується), можна буде провести більш глибокий аналіз специфіки використання CPU та GPU після випуску системи для загального доступу.

2. Проведення комплексного аналізу навантаження на всіх системах обчислювального центру дозволить порівняти відносні рівні використання та ефективність застосування різних архітектур, що підтримуються установою, а також краще зрозуміти, як користувачі адаптуються до гетерогенних середовищ.

3. Фінальним кроком стане використання розробленого синтетичного навантаження для запуску симуляційних експериментів з метою тестування та оцінки потенційних конфігурацій майбутніх HPC-систем. Це дасть змогу установі точно адаптувати наступне покоління інфраструктури до конкретних потреб своєї користувацької бази, забезпечуючи оптимальні інвестиції та максимальну продуктивність.

Висновки до розділу

У третьому розділі було запропоновано комплексну методологію аналізу однорідних і неоднорідних навантажень, яка базується на багаторівневому зборі даних, класифікації задач і статистичному профілюванні. Розгляд етапів побудови методології показав, що успішне моделювання робочого навантаження неможливе без узгодження різних джерел інформації та чіткої структури обробки даних. Особливе значення має представлена архітектура потоку даних, яка забезпечує повний цикл від збору телеметрії до синтезу репрезентативних навантажень. Сформульовані критерії класифікації задач продемонстрували свою здатність диференціювати навантаження за ключовими характеристиками, що дозволяє точніше оцінювати поведінку системи. У роботі доведено, що виявлення аномалій є важливою складовою аналізу, оскільки дає змогу вчасно

ідентифікувати неефективні або неправильні сценарії використання ресурсів. Метод синтезу навантаження дав можливість створити моделі, які відтворюють поведінкові особливості реальних систем без надмірного спрощення. Вибір репрезентативного періоду став ключовим етапом, що дозволив отримати збалансовану вибірку для формування синтезованих трас. Проведений статистичний аналіз підтвердив, що параметри навантаження можуть суттєво варіювати залежно від типу системи та користувацької активності. Запропоновані рекомендації щодо оптимізації ресурсів містять практичні підходи до підвищення ефективності використання CPU та GPU у гетерогенних середовищах.

ВИСНОВКИ

У магістерській роботі проведено дослідження методів аналізу однорідних і неоднорідних навантажень у високопродуктивних обчислювальних системах (HPC), що охоплює теоретичні, прикладні та методологічні аспекти моделювання, збору та інтерпретації даних про робоче навантаження. Отримані результати дозволили сформувати цілісну методологію, здатну забезпечити підвищення ефективності використання гетерогенних ресурсів HPC-систем та підґрунтя для оптимізації технологічних та інфраструктурних рішень у цьому домені.

По-перше, у результаті аналізу проблематики розподілу навантаження встановлено, що сучасні HPC-системи зазнають впливу широкого спектра факторів, серед яких ключовими є різнорідність апаратних компонентів, варіативність типів задач, зміна структури поданих заявок та збільшення частки задач із використанням прискорювачів. Дослідження архітектурних особливостей гібридних систем показало, що зростання ролі GPU суттєво загострює питання балансування навантаження, потребуючи адаптивних механізмів аналізу та прогнозування. Було також обґрунтовано, що зростання неоднорідності робочих навантажень підсилює актуальність створення універсальних моделей для опису поведінки користувачів і ресурсів у таких системах.

По-друге, у межах другої частини роботи реалізовано поглиблений аналіз реальних журналів диспетчерів ресурсів двох класів HPC-систем: загальнопризначеної багатовузлової та гетерогенної GPU-орієнтованої. Проведене профілювання дозволило виявити характерні закономірності використання ресурсів, зокрема нерівномірність навантаження процесорів, низьку завантаженість GPU у певні періоди, а також циклічність активності користувачів. Була сформована статистична характеристика навантаження, що охоплює розподіли тривалості задач, обсяги споживання ресурсів, рівень паралелізму, часові патерни та еволюцію навантаження. Порівняльний аналіз

між системами продемонстрував, що гетерогенні системи потребують відмінних підходів до інтерпретації навантаження через специфічні режими роботи GPU та їхню чутливість до типів задач.

По-третє, у роботі сформульовано комплексну методологію аналізу однорідних і неоднорідних навантажень, яка поєднує багаторівневий збір даних, класифікацію задач, статистичне профілювання, виявлення аномалій та синтез навантаження. Розроблена архітектура збору і обробки даних забезпечує можливість інтеграції різних джерел інформації — від системних логів до телеметрії прискорювачів. У роботі визначено ключові критерії диференціації навантажень, зокрема за характером паралелізму, профілем використання ресурсів, часовими змінами та відповідністю типовим сценаріям HPC-додатків.

По-четверте, метод синтезу робочого навантаження, запропонований у роботі, дає змогу формувати репрезентативні моделі для тестування продуктивності та масштабованості систем. Визначено правила вибору репрезентативного періоду, що забезпечують відтворення характеристик реального навантаження без втрати його структурної складності. Сформовані параметри синтезу дозволяють не лише моделювати поведінку задач у середовищі HPC, але й прогнозувати наслідки зміни конфігурацій системи.

По-п’яте, розроблені рекомендації щодо оптимізації використання ресурсів HPC-систем орієнтовані на підвищення ефективності розподілу навантаження. Зокрема, було обґрунтовано доцільність застосування політик гнучкого планування CPU-ресурсів, методів стимулювання активного використання GPU та підходів до адаптивного масштабування гетерогенних компонентів. Запропоновані рекомендації також враховують перспективу розвитку HPC-інфраструктури, включно з впровадженням нових типів прискорювачів та посиленням вимог до енергоефективності.

Узагальнення результатів роботи засвідчує, що проблема аналізу однорідних і неоднорідних навантажень потребує системного підходу, який передбачає інтеграцію методів статистичного аналізу, моделей поведінки

користувачів, параметричного профілювання та інтелектуальних механізмів виявлення закономірностей. Запропонована в магістерській роботі методологія забезпечує підвищення точності оцінки навантажень, покращення ефективності планування та сприяє оптимізації гетерогенних HPC-систем різного класу.

У підсумку, результати дослідження формують теоретичну й практичну основу для подальших робіт у напрямах прогнозного аналізу навантаження, інтелектуального розподілу ресурсів, автоматизованого профілювання задач та створення розширених моделей гібридних обчислювальних систем.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Scalable Systems and Software Architectures for High-Performance Computing on cloud platforms / Risshab Srinivas Ramesh // https://www.arxiv.org/pdf/2408.10281
2. Performance Modeling and Workload Analysis of Distributed Large Language Model Training and Inference . - https://arxiv.org/html/2407.14645v1
3. Adams, R., & Mitchell, T. Designing Adaptive Restaurant Management Systems Using Microservice Architecture. Journal of Software Engineering Practices, London: Springer, 2023, pp. 45–59.
4. Brown, S., & Keller, J. A Comparative Study of Mobile Ordering Platforms in the Food Service Industry. International Journal of Information Systems Research, New York: Elsevier, 2022, pp. 101–117.
5. Carter, L., Nguyen, P. Database Optimization Techniques for Real-Time Business Applications. Proceedings of the International Conference on Data Technologies, Berlin: IEEE, 2021, pp. 88–97.
6. Dawson, A., & Lin, Y. User-Centered Interface Design for Restaurant Automation. Human–Computer Interaction Review, Chicago: ACM Press, 2022, pp. 64–80.
7. Evans, R. Modeling Workflows in Restaurant Management Systems Using UML. Software Modeling and Simulation Journal, Boston: Elsevier, 2023, pp. 122–138.
8. Foster, G., & Hall, T. Security Challenges in Cloud-Based POS Systems. Cybersecurity and Information Protection Journal, Amsterdam: Springer, 2021, pp. 73–90.
9. Green, M. Integrating Inventory Tracking Modules into Management Platforms. Proceedings of the Conference on Intelligent Systems, Rome: IEEE, 2022, pp. 33–44.

10. Harris, K., & O’Neill, D. Evaluation of Performance Metrics in Web-Based Management Applications. Journal of Web Technologies, Toronto: ACM, 2023, pp. 150–166.
11. Smith, J. Adopting Agile Methodologies for Business Application Development. International Journal of Agile Systems, Zurich: Springer, 2021, pp. 11–27.
12. Johnson, P. Improving Restaurant Operations through Automated Scheduling Systems. Business Automation Journal, London: Taylor & Francis, 2022, pp. 88–103.
13. Kim, S., & Zhang, H. Data Integrity in Distributed Business Databases. Journal of Database Engineering, Singapore: IEEE, 2023, pp. 49–63.
14. Lewis, M. Assessment of Usability in Enterprise Management Software. User Experience Studies Review, San Francisco: ACM, 2021, pp. 32–48.
15. Martin, G., & Duarte, C. Integration Patterns for Third-Party Payment Gateways. Proceedings of the International Conference on Software Integration, Barcelona: Springer, 2023, pp. 97–115.
16. Nelson, R. Automated Reporting Tools for Small and Medium Enterprises. Journal of Business Informatics, Oxford: Elsevier, 2022, pp. 58–71.
17. Owens, T., & Patel, S. Machine Learning Techniques for Predicting Customer Demand in Restaurants. AI Applications Journal, Tokyo: IEEE, 2023, pp. 74–92.
18. Park, J. Optimizing Menu Recommendation Systems with Real-Time Analytics. Journal of Data-Driven Business Solutions, Seattle: Springer, 2021, pp. 119–137.
19. Quinn, L. Role of Object-Oriented Analysis in Modern Information Systems. Software Engineering Review, Berlin: Taylor & Francis, 2023, pp. 201–214.
20. Roberts, C., & Ahmad, Y. Evaluating Reliability of Cloud-Hosted Business Systems. Journal of Enterprise Cloud Computing, Sydney: ACM, 2022, pp. 17–36.

21. Stevens, D. Applying Scrum to the Development of Management Platforms. International Journal of Agile Practices, London: Wiley, 2023, pp. 55–72.
22. Turner, B., & Lee, A. Backend Optimization for High-Load Web Applications. Journal of Distributed Systems, New York: IEEE, 2021, pp. 98–113.
23. Underwood, T. Testing Strategies for Enterprise Web Applications. Journal of Software Quality Assurance, Paris: Springer, 2023, pp. 143–158.
24. Vasquez, R., & Chen, D. Implementation of Modular Architectures in Business Systems. International Software Architecture Journal, Rome: Elsevier, 2022, pp. 66–81.
25. White, J. Automated Order Processing Systems in Restaurant Environments. Journal of Digital Business Technologies, Denver: ACM, 2021, pp. 50–67.
26. Xu, F., & Morales, M. Improving Data Synchronization in Cross-Platform Applications. Journal of Mobile Systems Engineering, Stockholm: IEEE, 2023, pp. 91–108.
27. Young, E. Performance Evaluation Techniques for Web-Based Platforms. Computing Systems Review, London: Springer, 2022, pp. 127–142.
28. Zahir, K., & Adams, J. Adoption of REST APIs in Business Application Integration. Journal of Web Architectures, Amsterdam: Elsevier, 2021, pp. 34–49.
29. Allen, S. Cloud Scalability Models for Enterprise Management Software. International Journal of System Scalability, Tokyo: IEEE, 2023, pp. 200–214.
30. Brooks, A. Digital Transformation in the Food Service Sector. Journal of Service Industry IT, Chicago: Wiley, 2021, pp. 15–29.
31. Clark, P., & Omar, L. Reliability Analysis of Modern POS Systems. Proceedings of the Conference on Secure Technologies, Vienna: IEEE, 2023, pp. 52–70.

32. Daniels, H. Service-Oriented Approaches to Business Software Development. International Journal of SOA Engineering, Brussels: Springer, 2022, pp. 81–95.
33. Ellis, T. Enhancing Customer Interaction through Digital Ordering Interfaces. User Interaction Systems Journal, Boston: ACM Press, 2021, pp. 139–154.
34. Fisher, J., & Wang, R. Monitoring Tools for Enterprise Application Health. Journal of IT Infrastructure, Berlin: Taylor & Francis, 2023, pp. 44–59.
35. Gomez, E. Innovations in Restaurant Automation Using IoT Devices. IoT Engineering Review, New York: IEEE, 2022, pp. 73–88.
36. Hart, D., & Lee, S. Comparative Analysis of Frameworks for Web Application Development. Journal of Applied Computing, Milan: Elsevier, 2021, pp. 160–177.
37. Ivanova, K. Security Protocols for Protecting Customer Data in Web Systems. Cyber Defense Journal, Toronto: Springer, 2023, pp. 112–130.
38. Johansen, M. Workflow Automation Models in Business Management Systems. Journal of Enterprise Workflow Solutions, Copenhagen: Wiley, 2022, pp. 37–53.
39. King, H., & Zhao, Q. Scalable Designs for Multi-User Web Systems. Proceedings of the Web Engineering Congress, Dublin: ACM, 2023, pp. 21–39.
40. Lopez, A. Evaluating UX Approaches in Modern Management Platforms. Human-Centered Computing Journal, San Diego: Elsevier, 2021, pp. 55–70.
41. Miller, C., & Stone, F. Modular Approaches to ERP System Design. Enterprise Resource Systems Journal, Madrid: Springer, 2022, pp. 80–96.
42. Rivera, P. Advanced Data Processing Pipelines for Business Analytics. Journal of Intelligent Data Systems, Sydney: IEEE, 2023, pp. 100–118.
43. Shi, Y., Bhowmick, S., & Wu, C. (2018). A Scalable Decentralized Dynamic Load Balancing Framework for High Performance Computing. IEEE Transactions on Parallel and Distributed Systems, 29(1), 160-173.

44. Sadayappan, P., Shen, X., & Ponnusamy, R. (2007). Load Balancing for Scientific Applications on Heterogeneous Systems. IEEE Transactions on Parallel and Distributed Systems, 18(6), 762-776.
45. Al-Jassani, M., & Hammadi, H. (2019). Token-Based Dynamic Load Balancing in Large-Scale Distributed Systems. International Journal of Advanced Computer Science and Applications (IJACSA), 10(4), 1-10.