https://doi.org/10.31891/2307-5732-2025-355-93
УДК 004.032.26:004.4’22:004.738.5

ТРІСКА РОМАН
Національний університет «Львівська політехніка»
https://orcid.org/0009-0001-6403-1469
e-mail: roman.m.triska@lpnu.ua
ГЕНТОШ ЛЕСЯ
Національний університет «Львівська політехніка»
https://orcid.org/0000-0002-4957-1512
e-mail: lesia.i.mochurad@lpnu.ua

ГІБРИДНИЙ ПІДХІД ДО ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ДЛЯ ЕФЕКТИВНОГО АНАЛІЗУ ВЕЛИКИХ ДАНИХ

Швидке зростання обсягів даних у науці, бізнесі та цифровій інфраструктурі суттєво підвищило значущість аналітики великих даних як рушія аналізу, прийняття рішень та інновацій. З огляду на властиві великим даним характеристики - обсяг, швидкість та різноманітність («3V»), традиційні послідовні обчислювальні моделі вже не відповідають сучасним викликам. У зв’язку з цим паралельні обчислення стають критично важливими, оскільки забезпечують високу продуктивність, масштабованість та енергоефективність у процесі обробки даних.

У статті подано порівняння трьох основних стратегій паралельних обчислень: CPU-систем, GPU-архітектур та розподілених обчислень. Кожна має власні переваги та обмеження. Центральні процесори (CPU) забезпечують багатопотокове виконання задач загального призначення, але обмежені кількістю ядер і пропускною здатністю пам’яті. Графічні процесори (GPU) надають масивний паралелізм, придатний для інтенсивних обчислень, але мають обмеження у пам’яті й додаткові витрати на передавання даних. Розподілені системи (наприклад, Apache Spark, Dask, Ray) дозволяють горизонтальне масштабування та забезпечують еластичність і відмовостійкість, хоча потребують складної координації між вузлами.

У цій статті аналізується ефективність цих парадигм у реальних сценаріях, спираючись на бенчмарки Spark і Dask, застосування GPU-прискорення в аналітичних фреймворках і гібридні моделі з MPI та OpenACC. Оцінено продуктивність обчислень у різних контекстах, що дозволяє визначити доцільність кожного підходу залежно від задачі.

Новизна дослідження полягає у запропонованій концепції гібридного фреймворку, що поєднує всі три стратегії в єдину багаторівневу архітектуру. CPU відповідають за оркестрацію та легкі обчислення, GPU за паралельну обробку інтенсивних навантажень, а розподілені системи за масштабування та обробку великих обсягів даних. Такий підхід дозволяє суттєво підвищити ефективність використання ресурсів, та продуктивність системи.

Ключові слова: паралельні обчислення, аналітика великих даних, планування завдань, високопродуктивні обчислення, оптимізація конвеєра даних.

TRISKA ROMAN, HENTOSH LESIA
Lviv Polytechnic National University

HYBRID APPROACH TO PARALLEL COMPUTING FOR EFFICIENT BIG DATA ANALYTICS

The rapid growth of data volumes in science, business, and digital infrastructure has significantly increased the importance of big data analytics as a driving force for analysis, decision-making, and innovation. Given the inherent characteristics of big data - volume, velocity, and variety (the “3Vs”), traditional sequential computing models no longer meet current challenges. As a result, parallel computing has become critically important, offering high performance, scalability, and energy efficiency in data processing.

This article compares three main parallel computing strategies: CPU-based systems, GPU architectures, and distributed computing. Each has its advantages and limitations. Central processing units (CPUs) provide multithreaded execution for general-purpose tasks but are limited by the number of cores and memory bandwidth. Graphics processing units (GPUs) offer massive parallelism suited for intensive computations but face memory constraints and data transfer overhead. Distributed systems (such as Apache Spark, Dask, and Ray) enable horizontal scaling and provide elasticity and fault tolerance, although they require complex inter-node coordination.

The paper analyzes the effectiveness of these paradigms in real-world scenarios, relying on benchmarks for Spark and Dask, the use of GPU acceleration in analytical frameworks, and hybrid models combining MPI and OpenACC. It evaluates computing performance across different contexts, helping to determine the suitability of each approach depending on the task.

The novelty of this study lies in the proposed concept of a hybrid framework that integrates all three strategies into a unified multi-level architecture. CPUs are used for orchestration and lightweight tasks, GPUs for parallel processing of intensive workloads, and distributed systems for scalable processing of large data volumes. This approach significantly enhances resource utilization efficiency and overall system performance.

Keywords: parallel computing, big data analytics, task scheduling, high-performance computing, data pipeline optimization.

Стаття надійшла до редакції / Received 23.05.2025
Прийнята до друку / Accepted 26.06.2025

Постановка проблеми

Вибухове зростання обсягу даних з цифрових платформ, біомедичних датчиків та наукових досліджень зробило аналітику великих даних важливою в таких галузях, як геноміка, фінанси та кібербезпека. Однак різноманітність, обсяг та швидкість таких даних значно перевищують можливості традиційних послідовних систем, що вимагає масштабованих обчислювальних моделей.

Для вирішення цієї проблеми широко використовуються три парадигми паралельних обчислень. Паралелізм на основі центрального процесора спирається на багатоядерну потоковість та фреймворки, такі як OpenMP та MPI, що пропонує гнучкість, але обмежену пропускну здатність. Обчислення на основі графічного

процесора, що підтримуються такими платформами, як CUDA та OpenCL, чудово справляються з паралельними завданнями обробки даних, але вводять додаткові витрати на пам'ять та передачу даних. Розподілені системи, такі як Apache Spark та Dask, масштабують аналітику між кластерами, покращуючи відмовостійкість та розподіл пам'яті, але додають складності мережі та планування.

Кожна парадигма пропонує різні переваги, але жодна окремо не є достатньою для повного спектру робочих навантажень великих даних, особливо враховуючи, що аналітика в реальному часі, енергоефективність та обробка даних стають критично важливими. Це створює потребу в гібридних обчислювальних архітектурах, які інтегрують сильні сторони всіх трьох підходів. Однак проектування таких систем створює нетривіальні проблеми в оркестрації, сумісності та оптимізації продуктивності.

Аналіз досліджень та публікацій

Нещодавні дослідження розкривають конкретні аспекти паралельних обчислень у контексті великих даних. Дослідження розглянуті в [1] порівняли Spark та Dask для завдань нейровізуалізації та виявили, що обидва фреймворки обмежені продуктивністю вводу/виводу та використанням пам'яті. Згідно з [2], прискорення GPU дає приріст продуктивності для великих наборів даних, але страждає від накладних витрат на передачу при менших робочих навантаженнях. [3] підкреслили ефективність MPI порівняно зі Spark в геномному аналізі, але зазначили простоту використання та відмовостійкість Spark. У роботах [4] та [5] запропоновано гібридні або гетерогенні моделі, проте вони вимагають нетривіального планування та системної інтеграції. Також в роботі [6] наголосили на відсутності узгодженості продуктивності в гетерогенних середовищах. Незважаючи на цінні висновки, ці дослідження не пропонують єдиної або порівняльної структури, залишаючи місце для систематизованих рекомендаціях щодо вибору архітектури в аналітиці великих даних.

Формулювання цілей статті

Метою роботи є: проведення порівняльного дослідження стратегій обчислень на основі центрального процесора (CPU), прискорених на графічному процесорі (GPU) та розподілених обчислень для аналітики великих даних, а також розробка гібридної концептуальної основи, що інтегрує переваги кожного з підходів для підвищення продуктивності, масштабованості та адаптивності у різноманітних завданнях обробки даних.

Виклад основного матеріалу

В аналітиці великих даних обсяг і складність інформації вимагають не лише швидших обчислень, але й розумнішого структурування самих обчислень. Паралельні обчислення забезпечують основу для цього зрушення, дозволяючи виконувати завдання одночасно, а не послідовно. Щоб краще зрозуміти, як паралельні стратегії застосовуються в середовищах великих даних, важливо розрізняти три основні парадигми паралелізму: паралелізм завдань, паралелізм даних та конвеєрний паралелізм. Ці концептуальні моделі слугують основою для архітектурних стратегій, що використовуються в системах на основі центрального процесора, графічного процесора та розподілених системах.

Паралелізм завдань передбачає одночасне виконання різних обчислювальних завдань. Ці завдання можуть виконувати різні операції над одним або різними наборами даних. Ця модель добре підходить для універсальних процесорів і зазвичай реалізується за допомогою бібліотек потоків, таких як OpenMP [7], або методів паралельного програмування, таких як пули потоків. Багатоядерні процесори добре справляються з незалежними завданнями, такими як паралельні операції вводу/виводу або різні етапи конвеєра попередньої обробки.

Паралелізм даних означає виконання однієї й тієї ж операції над багатьма елементами даних одночасно. Ця парадигма природно поєднується з графічними процесорами (GPU), які містять тисячі легких ядер, оптимізованих для паралельного виконання однієї й тієї ж інструкції на різних фрагментах даних. Паралелізм даних широко використовується в машинному навчанні, матричних операціях та обробці зображень. Платформи GPU, такі як CUDA та OpenCL, побудовані безпосередньо навколо цієї концепції [8].

Конвеєрний паралелізм розділяє завдання на дискретні етапи, де вихід одного етапу стає входом для наступного [9]. Кожен етап може оброблятися паралельно, особливо коли система може буферизувати проміжні результати. Ця модель поширена в розподілених системах та конвеєрах інженерії даних, де етапи прийому, перетворення та аналізу даних виконуються одночасно на різних вузлах або мікросервісах.

Взаємозв'язок між типом паралелізму та архітектурою є критично важливим при оцінці архітектурних стратегій: погана узгодженість між моделлю паралелізму та архітектурою системи призводить до недостатнього використання, затримки та неефективності витрат. Узгодження типів паралелізму та архітектури систем наведено в таблиці 1.

Таблиця 1

Узгодження типів паралелізму та архітектурних стратегій

Тип паралелізму	Архітектура що підходить	Загальні інструменти та фреймворки	Приклади використання
Паралелізм завдань	CPU (багатоядерний, потоковий)	OpenMP, MPI, pthreads, Intel TBB	Паралельне виконання різноманітних ETL-завдань
Паралелізм даних	GPU (тисячі малих ядер)	CUDA, OpenCL, Tensor Cores	Навчання Deep Learning Models
Конвеєрний паралелізм	Distributed systems (багатовузлові)	Apache Spark, Dask, Ray, Flink	Конвеєри потокової передачі даних, обробка журналів

Сучасні *процесори (CPU)* оснащені кількома ядрами та векторними блоками виконання, що дозволяє виконувати паралелізм на рівні потоків та операції SIMD (одна інструкція, кілька даних). У конвеєрах великих даних системи на базі процесорів обробляють різноманітні робочі навантаження, включаючи логіку з високим навантаженням на керування, попередню обробку даних та оркестрацію завдань. Основні характеристики та аналіз CPU архітектури наведено в таблиці 2.

Таблиця 2

Аналіз архітектури на базі CPU

Ключові технології	Сильні сторони	Обмеження
• OpenMP, Intel TBB, pthreads • MPI (для кластерів процесорів)	• Гнучка модель програмування. • Підходить для змішаних робочих навантажень та логіки розгалуження. • Добре підтримується у високопродуктивних обчисленнях (HPC) та мовах загального призначення (C/C++, Java, Python).	• Обмежена масштабованість поза межами кількості сокетів/ядер. • Приріст продуктивності зменшується зі збільшенням потоків (закон Амдала). • Вузькі місця в пропускній здатності пам'яті.

Графічні процесори пропонують величезні можливості паралельної обробки даних, спочатку розроблені для графіки, але тепер широко використовуються для обчислень загального призначення. Маючи тисячі ядер, графічні процесори ідеально підходять для завдань з високим обчислювальним навантаженням, таких як глибоке навчання, матричні операції та аналіз зображень. Основні характеристики та аналіз GPU архітектури наведено в таблиці 3.

Таблиця 3

Аналіз архітектури на базі GPU

Ключові технології	Сильні сторони	Обмеження
• CUDA, OpenCL, ROCm • Тензорні ядра (для прискорення машинного навчання) • Бібліотеки: cuDF, RAPIDS, PyTorch, TensorFlow	• Масивний паралелізм для однорідних операцій. • Висока пропускна здатність пам'яті. • Чудово підходить для навчання моделей машинного навчання та наукових симуляцій.	• Висока вартість передачі даних (CPU-GPU). • Ємність пам'яті обмежена порівняно з оперативною пам'яттю CPU. • Менш ефективний для невеликих або складних завдань.

Розподілені системи виконують завдання на кількох машинах для обробки наборів даних, що перевищують ємність одного вузла. Популярні фреймворки, такі як Spark, Dask та Ray, абстрагують розподіл даних, відмовостійкість та управління пам'яттю, що спрощує масштабування аналітики даних. Аналіз архітектури на базі розподілених систем, та її основні характеристики наведено в таблиці 4.

Таблиця 4

Аналіз архітектури на базі розподілених систем

Ключові технології	Сильні сторони	Обмеження
• Apache Spark, Dask, Ray • Kubernetes, Hadoop YARN, Airflow (для оркестрації)	• Горизонтальна масштабованість. • Відмовостійкість та стійкість. • Прості у використанні API та абстракції даних (RDD, масиви Dask).	• Високі мережеві накладні витрати на перетасування даних. • Навантаження на пам'ять та накладні витрати на збирання вантажу (особливо в Spark). • Потрібне налаштування, щоб уникнути простою інфраструктури або дисбалансу завдань.

Для змістовного порівняння систем на основі центрального процесора, систем з прискоренням на графічному процесорі та розподілених систем зазвичай використовується кілька основних показників:

Пропускна здатність (завдань/сек або МБ/сек): Скільки елементів даних або операцій можна обробити за одиницю часу.

Затримка (мс/операцій): Затримка на завдання, важлива для програм реального часу або інтерактивних програм.

Обсяг пам'яті: Загальне споживання пам'яті під час виконання робочого навантаження.

Енергоефективність (операцій/ват): Потужність, необхідна для виконання робочого навантаження, особливо актуальна для високопродуктивних обчислень та вартості обчислень в хмарі.

Масштабованість: Наскільки добре масштабується продуктивність з більшою кількістю ядер, графічних процесорів або вузлів.

Складність розгортання: Наскільки легко чи складно впроваджувати, налаштовувати та підтримувати систему у виробництві.

На рис. 1 наведено візуальне порівняння стратегій паралельних обчислень.

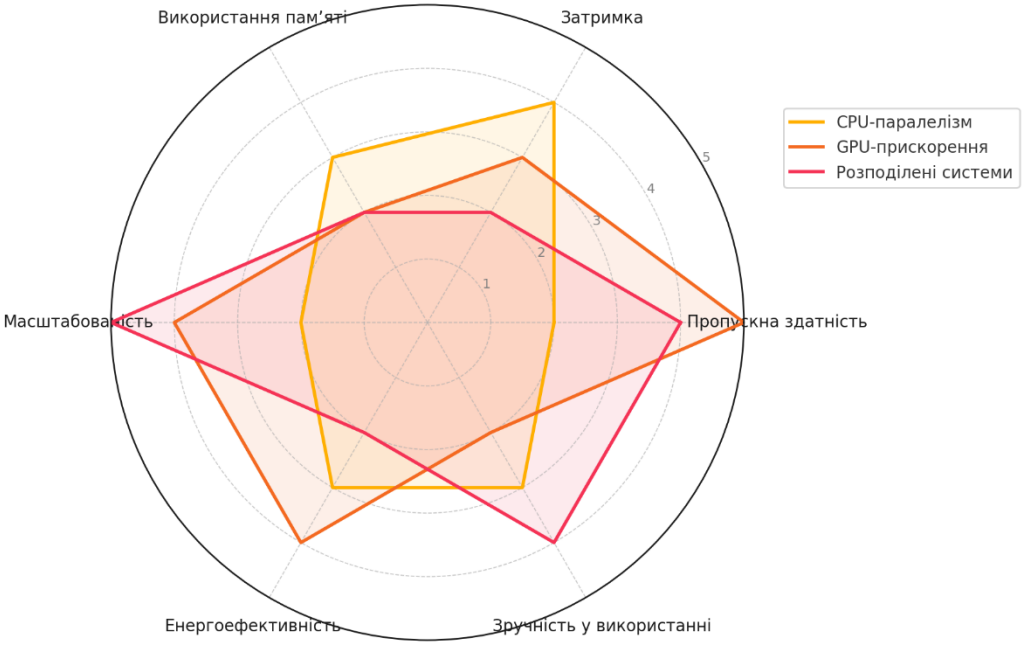

Рис. 1. Порівняння стратегій паралельних обчислень для аналітики Big Data

Хоча системи на основі центрального процесора (CPU), графічного процесора (GPU) та розподілені системи пропонують унікальні переваги для аналітики великих даних, інтеграція цих стратегій у цілісний та ефективний конвеєр залишається нетривіальним завданням. Реальні робочі навантаження рідко чітко узгоджуються з єдиною архітектурою. Натомість вони часто включають різноманітні фази, такі як прийом даних, попередня обробка, перетворення, навчання та обслуговування, які могли б отримати користь від різних обчислювальних парадигм. Однак поєднання цих парадигм створює новий набір технічних та операційних проблем, які необхідно вирішити, перш ніж їхній спільний потенціал буде повністю реалізовано.

Основні виклики інтеграції систем, які базуються на різних архітектурах:

- *Фрагментація ланцюжка інструментів.* Окремі API та середовища виконання для центрального процесора (CPU), графічного процесора (GPU) та розподілених компонентів.
- *Накладні витрати на передачу даних.* Дорогі передачі між хост-пристроями та перестановки між вузлами.
- *Статичне планування.* Більшість планувальників ігнорують неоднорідність обладнання.
- *Невідповідність продуктивності.* Мінливість між платформами та розмірами завдань.
- *Високі операційні витрати.* Складне розгортання, моніторинг та налагодження систем які поєднують декілька архітектур.

Проблеми, окреслені вище, демонструють, що жодна окрема архітектура – центральний процесор (CPU), графічний процесор (GPU) або розподілена система – не здатні задовольнити всі вимоги сучасної аналітики великих даних. Однак сильні сторони кожної парадигми доповнюють одна одну, що свідчить про те, що ретельно інтегрована гібридна модель може пом'якшити окремі обмеження, одночасно максимізуючи продуктивність системи, адаптивність та масштабованість.

Як рішення, пропонується гібридна концептуальна платформа зображена на рис. 2, розроблена для інтелектуального розподілу завдань обробки даних між гетерогенними обчислювальними ресурсами. Вона спрямована на узгодження правильного робочого навантаження з правильним механізмом, використовуючи центральні процесори для оркестрації та завдань вводу-виводу з низьким обсягом обчислень, графічні процесори для високопродуктивних обчислень та розподілені системи для масштабування обчислювальної потужності між вузлами.

Гібридна структура базується на таких основних принципах:

- *Архітектурна спеціалізація:* Зіставлення типів завдань з оптимальними обчислювальними ресурсами (наприклад, використання графічних процесорів (GPU) для ядер глибокого навчання, центральних процесорів (CPU) для розбору JSON або оркестрації).
- *Рівні конвеєра:* Розділення процесу аналітики на логічні рівні - попередня обробка, трансформація, моделювання, постобробка - кожен з яких відповідає окремій архітектурі.
- *Розумне планування:* Використання рівня проміжного програмного забезпечення (middleware) або апаратно-залежного планувальника для динамічного призначення завдань на основі доступності ресурсів та профілю робочого навантаження.

- *Мінімізоване переміщення даних:* Проєктування конвеєра для зменшення непотрібних передач даних (наприклад, збереження проміжних результатів у пам'яті графічного процесора, де це можливо).
- *Прозора абстракція:* Надає змогу спеціалістам з обробки даних та інженерам визначати логіку конвеєра без необхідності вручну керувати низькорівневою оркестрацією.

Запропонована архітектура складається з чотирьох основних рівнів:

1. Захоплення та оркестрація (рівень процесора):
    a. Обробляється багатоядерними процесорами.
    b. Відповідає за розбір вхідних даних, валідацію, перетворення форматів та координацію конвеєрів.
    c. Використовує оркестратори на основі Python або Java, OpenMP, легкі фреймворки (наприклад, Ray для маршрутизації завдань).
2. Паралельні обчислення (рівень графічного процесора):
    a. Опрацьовує навчання моделей машинного навчання, великомасштабні матричні операції, паралельна інженерія функцій даних.
    b. Використовує CUDA, cuDF, RAPIDS, TensorFlow або Flink + TornadoVM.
    c. Ці завдання об'єднуються в пакети та передаються масово, щоб уникнути накладних витрат PCIe.
3. Масштабоване виконання (розподілений рівень):
    a. Розподіл завдань між вузлами за допомогою Apache Spark, Dask або Ray.
    b. Ролі: розподілений ETL, перетасування даних, тривалі пакетні завдання та важкі агрегації.
    c. Вузли можуть розміщувати лише апаратне забезпечення на базі центрального процесора або змішане обладнання на базі центрального процесора та графічного процесора.
4. Програмне забезпечення проміжного рівня планування (додатковий логічний рівень):
    a. Логіка маршрутизації завдань на основі метаданих та статистики ресурсів.
    b. Готові до майбутньої динамічної адаптації за допомогою планувальників на основі машинного навчання.

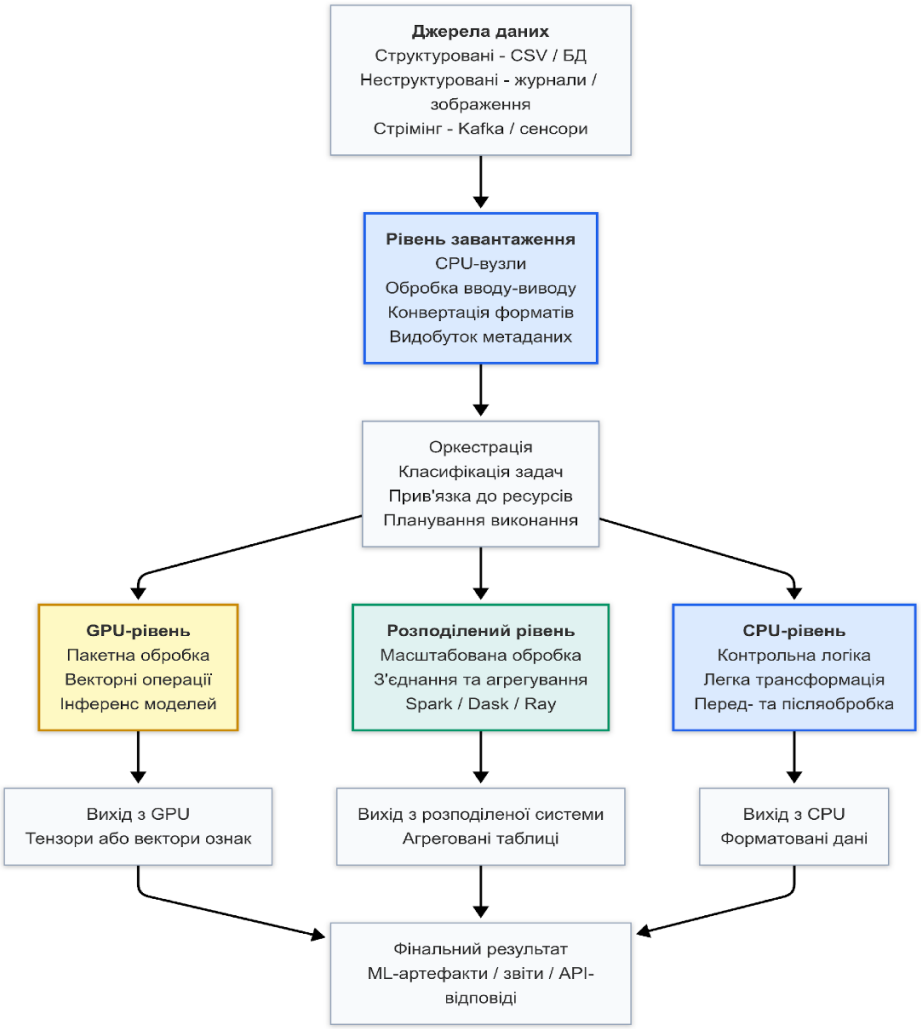

Рис. 2. Гібридна архітектура паралельних обчислень для аналітики великих даних із автоматизованою маршрутизацією задач

Очікувані переваги запропонованої архітектури:

- *Оптимізація продуктивності* – кожне завдання виконується на найбільш ефективному обладнанні.
- *Масштабованість* – підтримується як горизонтальне, так і вертикальне масштабування.
- *Енергоефективність* – уникає надмірного використання потужних графічних вузлів для завдань з низьким обчислювальним ресурсом.
- *Адаптивність* – підходить для пакетних завдань, потокових конвеєрів та інтерактивної аналітики.
- *Гнучкість до змін* – може інтегрувати нове обладнання (наприклад, FPGA, прискорювачі штучного інтелекту) в межах однієї логіки планування.

Висновки

Підсумовуючи, інтеграція стратегій паралельних обчислень – систем на основі центрального процесора (CPU), систем з прискоренням на графічному процесорі (GPU) та розподілених систем (distributed systems) - являє собою важливий крок вперед в еволюції аналітики великих даних. Оскільки обсяги даних та вимоги до обробки продовжують зростати, жодна окрема обчислювальна архітектура не може ефективно задовольнити всі вимоги до робочого навантаження. Гібридна структура, запропонована в цій статті, надає практичне рішення, узгоджуючи конкретні завдання з найбільш ефективними ресурсами, тим самим покращуючи продуктивність, зменшуючи накладні витрати та підвищуючи масштабованість. Цей багаторівневий підхід не тільки підвищує адаптивність конвеєрів обробки даних, але й закладає основу для впровадження майбутніх апаратних інновацій. Подальші дослідження та вдосконалення гібридних стратегій допоможуть встановити найкращі практики для проектування надійних, ефективних та інтелектуальних систем аналітики великих даних.

Література

1. Dugré K., Suleiman I., Lawrence A. Performance comparison of Dask and Apache Spark on HPC systems for neuroimaging // arXiv, 2023. – arXiv:2406.01409. DOI: https://doi.org/10.48550/arXiv.2406.01409.

2. Toorpu R. Comparative assessment of GPU-accelerated vs. CPU-based databases: Architecture, performance, and cost implications // International Journal of Cloud Computing and Data Management. – 2025. – Vol. 6, № 1. – P. 4–16. DOI: https://doi.org/10.33545/27075907.2025.v6.i1a.79.

3. Abuín J.M., Rivas-García L., Jimeno A. Big Data in metagenomics: Apache Spark vs MPI // PLOS ONE. – 2020. – Vol. 15, № 10. – e0239741. DOI: https://doi.org/10.1371/journal.pone.0239741.

4. Alshahrani A., Al-Gudaifi A. Accelerating Spark-Based Applications with MPI and OpenACC // Complexity. – 2021. – Article ID 9943289. DOI: https://doi.org/10.1155/2021/9943289.

5. Xekalaki P., Collari F., Emmi C. Enabling Transparent Acceleration of Big Data Frameworks Using Heterogeneous Hardware // Proceedings of the VLDB Endowment. – 2022. – Vol. 15, № 13. – P. 3869–3882. DOI: https://doi.org/10.14778/3565838.3565842.

6. Shin J., Lee H., Choi K. Large-Scale Data Computing Performance Comparisons on SYCL Heterogeneous Parallel Processing // Applied Sciences. – 2020. – Vol. 10, № 5. – Article 1656. DOI: https://doi.org/10.3390/app10051656.

7. Mochurad L., Kryvinska N. Parallelization of Finding the Current Coordinates of the Lidar Based on the Genetic Algorithm and OpenMP Technology. Symmetry. 2021; 13(4):666. https://doi.org/10.3390/sym13040666.

8. Mochurad L. Implementation and analysis of a parallel kalman filter algorithm for lidar localization based on CUDA technology. Front. Robot. AI 11:1341689, 2024, https://doi.org/10.3389/frobt.2024.1341689.

9. Navarro A., Asenjo R., Tabik S. and Cascaval C. Analytical Modeling of Pipeline Parallelism, 2009 18th International Conference on Parallel Architectures and Compilation Techniques, Raleigh, NC, USA, 2009, pp. 281-290, https://doi.org/10.1109/PACT.2009.28.

References

1. Dugré K., Suleiman I., Lawrence A. Performance comparison of Dask and Apache Spark on HPC systems for neuroimaging // arXiv, 2023. – arXiv:2406.01409. DOI: https://doi.org/10.48550/arXiv.2406.01409.

2. Toorpu R. Comparative assessment of GPU-accelerated vs. CPU-based databases: Architecture, performance, and cost implications // International Journal of Cloud Computing and Data Management. – 2025. – Vol. 6, № 1. – P. 4–16. DOI: https://doi.org/10.33545/27075907.2025.v6.i1a.79.

3. Abuín J.M., Rivas-García L., Jimeno A. Big Data in metagenomics: Apache Spark vs MPI // PLOS ONE. – 2020. – Vol. 15, № 10. – e0239741. DOI: https://doi.org/10.1371/journal.pone.0239741.

4. Alshahrani A., Al-Gudaifi A. Accelerating Spark-Based Applications with MPI and OpenACC // Complexity. – 2021. – Article ID 9943289. DOI: https://doi.org/10.1155/2021/9943289.

5. Xekalaki P., Collari F., Emmi C. Enabling Transparent Acceleration of Big Data Frameworks Using Heterogeneous Hardware // Proceedings of the VLDB Endowment. – 2022. – Vol. 15, № 13. – P. 3869–3882. DOI: https://doi.org/10.14778/3565838.3565842.

6. Shin J., Lee H., Choi K. Large-Scale Data Computing Performance Comparisons on SYCL Heterogeneous Parallel Processing // Applied Sciences. – 2020. – Vol. 10, № 5. – Article 1656. DOI: https://doi.org/10.3390/app10051656.

7. Mochurad L., Kryvinska N. Parallelization of Finding the Current Coordinates of the Lidar Based on the Genetic Algorithm and OpenMP Technology. Symmetry. 2021; 13(4):666. https://doi.org/10.3390/sym13040666.

8. Mochurad L. Implementation and analysis of a parallel kalman filter algorithm for lidar localization based on CUDA technology. Front. Robot. AI 11:1341689, 2024, https://doi.org/10.3389/frobt.2024.1341689.

9. Navarro A., Asenjo R., Tabik S. and Cascaval C. Analytical Modeling of Pipeline Parallelism, 2009 18th International Conference on Parallel Architectures and Compilation Techniques, Raleigh, NC, USA, 2009, pp. 281-290, https://doi.org/10.1109/PACT.2009.28